Net Mobile-Cop: A Hybrid Intelli-Agent Framework to Manage Networks

Mydhili K Nair
Department of
Information Science and Engineering
M.S Ramaiah Institute of Technology
Bangalore, India
Email: mydhili.nair@gmail.com

Chandan Bhosle
Department of
Information Science and Engineering
M.S Ramaiah Institute of Technology
Bangalore, India
Email: chandanbhosle@gmail.com

V Gopalakrishna
Integra Micro Systems
Bangalore, India
Email: vgopi@integramicro.com

***Abstract*—More than a decade ago, MA(Mobile Agents) were introduced for NM(Network Management) of distributed computer networks in tandem with the design paradigm of *Code-Shipping* the agents to remote node, where they are programmed to collect, analyze and process data locally. However, even today, most networks employ SNMP (Simple Network Management Protocol), an inherently *Client-Server* based stable and proven protocol, which uses the paradigm *Data-Shipping*. It suffers from the major drawback that when the network traffic increases, the manager is overloaded due to excessive processing. Real synergy could be achieved if we adopt a hybrid approach that brings the best of both paradigms. In this paper we present *Net Mobile-CopHybrid NM Framework*, prototyped on Aglet Mobile Agent System and AdventNet SNMP Package, which imbibes this synergy. We also introduce a novel method to dynamically configure managed nodes using intelligent agents. We also provide quantitative evaluation leading to useful tips to the sys-admin as to when to toggle between SNMP and MA usage.**

***Index Terms*— Mobile/Intelligent Agents, Network Configuration, Management, Monitoring, Management by Delegation, Health Functions, Table Filtering, AdventNet, Aglets, SNMP**

I. Introduction

Computer Network Management is traditionally done by using a centralized **NMS** (**N**etwork **M**anager **S**ystem). It constantly checks for ***F**ault*, *C**onfiguration*, ***A**ccounting*, ***P**erformance* and ***S**ecurity*, called ***'FCAPS'*** monitoring, defined by ISO-OSI as the five pillars in the framework of network management [14].

SNMP*(**S**imple **N**etwork **M**anagement **P**rotocol)* is the most popular protocol used in such a centralized NMS, using polling between the manager and managed nodes. It is a client-server approach where each poll is actually a **RPC** *(**R**emote **P**rocedure **C**all)* to the remote SNMP Agent. This paradigm is called *Data-Shipping* [11]. In this paper, we present a quantitative analysis of the volume of data that is generated in a network if we rely only on SNMP, causing information bottleneck at the managing node.

James White [1] has introduced **MA** *(**M**obile **A**gents)* as a strategy for distributed applications. MA is a program, which migrates from host to host in a heterogeneous computer network, leveraging on the strength of *Remote Programming* [10] as opposed to *Client-Server Programming* used in SNMP based management. This paradigm is called *"Code-Shipping"* [11] where the MA migrates to the managed node and executes what it was coded to do (E.g. Checking threshold values). This decentralizes network management adopting the popular technique **MbD** (**M**anagement by **D**elegation) [15]

In this paper, we propose a framework called ***'Net Mobile-Cop Hybrid NM (Network Management) Framework'***, marrying the principles of *Code-Shipping* with intelligent MA (*intelli-agents*) and *Data-Shipping* using traditional SNMP. Thus, we highlight, how our framework chooses to combine the best of both design paradigms making it a flexible, decentralized network monitoring architecture.

This paper is organized as follows, with Section 2 giving an overview of SNMP using Data Shipping. Section 3 explains how MAs are used for Code-Shipping.Section 4 is the longest one in this paper where the crux of our framework's architecture, dynamic configuration of a managed node by an *intelli-agent* and calculation of HF (Health Functions) are explained. Section 5 presents a Quantitative Evaluation while Section 6 gives a succinct review of Related Work [5][6][7][8][9][13] carried out in this area, and how our work fills its voids. We conclude in Section 7 summarising our present work and throwing light on the future directions planned.

II. Data-Shipping Approach of Traditional SNMP

In a typical SNMP based centralized management, the managing node makes ***Requests (GETxxx)*** to the SNMP agent at the managed node, that queries the **MIB**(Management Information Base), having the state of network parameters. If we need to retrieve an SNMP Table (E.g. IP routing or Software process running table) consisting of hundreds of entries,we require at least one *get-next* request operation per table row [6].Each get-next operation has to be completed before the next one can start. Imagine the amount of network latency that will creep in, especially when managing a remote LAN! The SNMP table reflect different updates at multiple points in time, due to the sluggishness encountered in getting the values of the individual **OIDs** *(**O**bject **ID**entifiers)* of

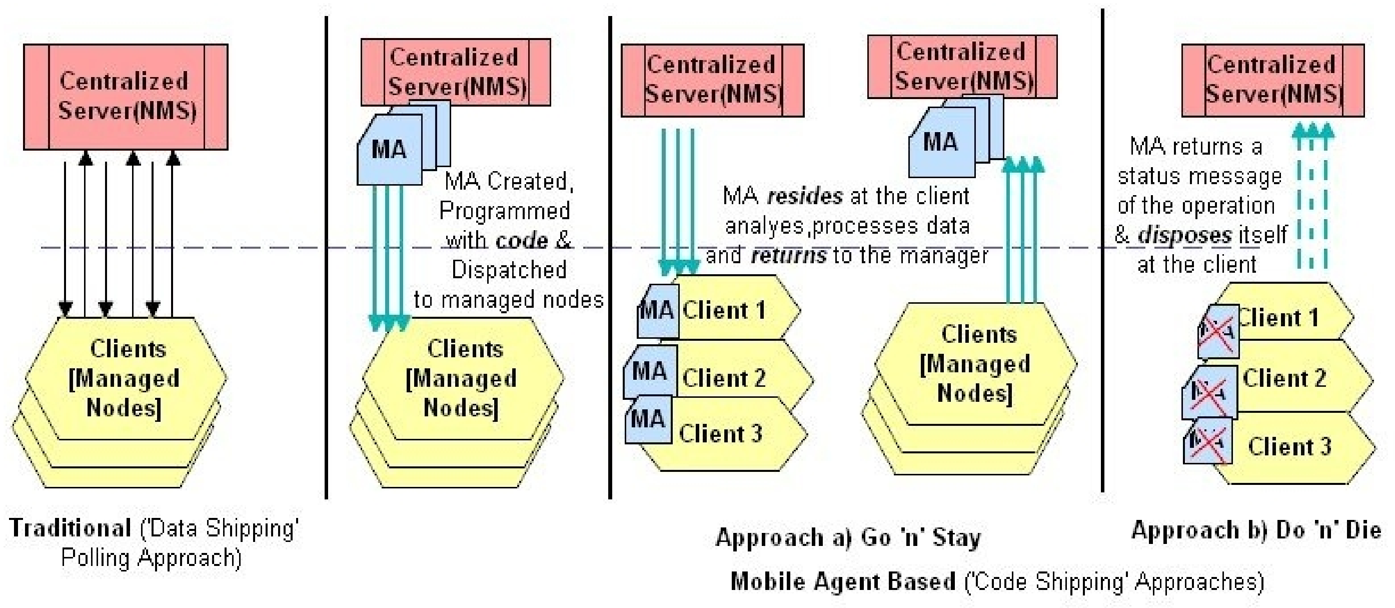

Centralized Server(NMS)
Clients [Managed Nodes]
Traditional ('Data Shipping' Polling Approach)
MA
MA Created, Programmed with code & Dispatched to managed nodes
MA resides at the client analyes,processes data and returns to the manager
Client 1
Client 2
Client 3
MA returns a status message of the operation & disposes itself at the client
Approach a) Go 'n' Stay
Approach b) Do 'n' Die
Mobile Agent Based ('Code Shipping' Approaches)

Fig. 1. Traditional SNMP vs. MA showing our approach Do ‘n‘ Die to configure managed nodes

the MIB.SNMPv2c provides *get-bulk* request to counter this, with max-repetition (max number of successive values to be returned). But, the sys-admin has to guess its value, smaller value causes too many message exchanges and larger value causes the SNMP PDU size to overshoot above the 64 KB limit [8].

III. Code-Shipping with Mobile Agents

Fig.1 diagrammatically [16] represent the difference between SNMP with Data-Shipping, as against MA with Code-Shipping. We introduce a new approach for dynamic configuration of the managed nodes, namely **DnD** ***(Do ‘n’ Die)***, taking the principle of *MbD* (Management by Delegation) [15] a step further.

- DnD (Do ‘n’ Die): We have named our methodology *DnD (Do ‘n’ Die)* to be in tune with the approaches proposed by Gavalas et al [6]. Here, the MAs dispatched by the manager are *‘intelli-agents’*. In case, configuration of a node is required, they take decisions autonomously. As shown in Fig.1, the MA reports a success or failure back to the manager and disposes itself. Thus, our framework prevents the dangers introduced by agent-minions [16], causing malicious code to spread.

Gavalas et al [6] has proposed two strategies, namely *“Get ‘n’ Go”* and *“Go ‘n’ Stay”* which we highlight next.

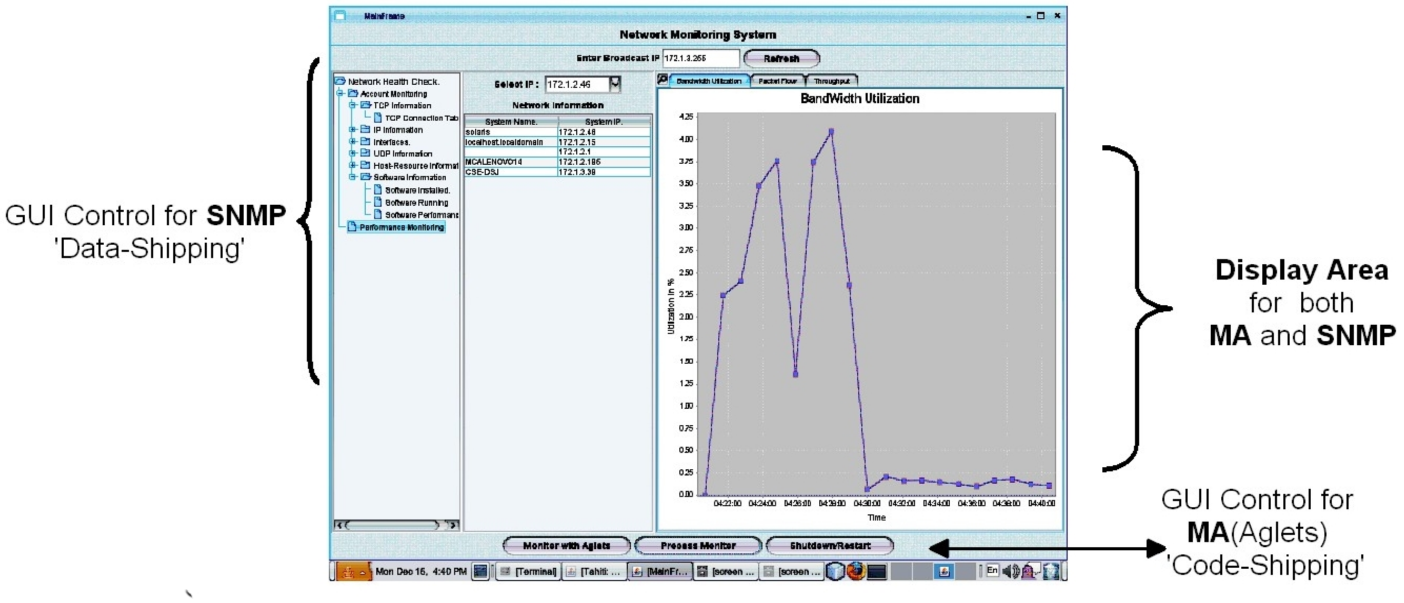
- GnG (Get ‘n’ Go) [6]: Here the network is partitioned into many sub-nets and one MA is assigned to each. The manager defines a **‘polling duration’** for each sub-net, creates the MA and dispatches it. The MA visits each managed node, collects the data, stores the state of the parameters and calculated values within it, then migrates to the next node and returns to the manager with the processed data after the polling duration. This strategy is popularly called the *‘Itinerary Model‘* [7] of MA migration. By this, we can pinpoint things like the most heavily loaded **NIC** *(**N**etwork **I**nterface **C**ards)* in the network.
- GnS (Go ‘n’ Stay) [6]: This approach follows the *‘Broadcast Model’* [7] of MA migration. The manager creates and dispatches as many MAs as the number of nodes it wants to manage. The MA resides there for a number of polling intervals, collecting and analyzing data samples, before returning to the manager. We use this strategy when analysis of data is to be done off-line, like recording the resource (CPU/memory) or bandwidth utilization peak on a node in an observation period.

IV. Net Mobile-Cop Hybrid NM Framework

When we dispatch MAs, the resource consumption at the managed nodes running the **MAEE** (Mobile Agent Execution Environment) increases [11]. If a network element to be managed does not provide the sufficient resources to run a mobile agent, e.g. a switch or a router, we need to install a traditional SNMP agent on it [5]. It can be now managed either from the central manager node or from a MA from a nearby-managed node running the MAEE. Thus, using our framework we are able to manage a network element with traditional SNMP installed in it as well as distribute the delegation of management using MA, impeccably integrating any legacy NMS with our framework, depicted in Fig.3.

A. ‘Net Mobile-Cop Hybrid NM Framework‘ Architecture

The focus of our *Net Mobile-Cop Hybrid NM Framework* is to examine the impact of combining the traditional Client-Server SNMP approach of “Data-Shipping” with that of the



System Name.	System IP.
solaris	172.1.2.46
localhost.localdomain	172.1.2.15
	172.1.2.1
MCALENOVO14	172.1.2.186
CSE-DSJ	172.1.3.38

Fig. 2. Execution Result: Screenshot of Net Mobile-Cop GUI depicting the Hybrid Approach of framework

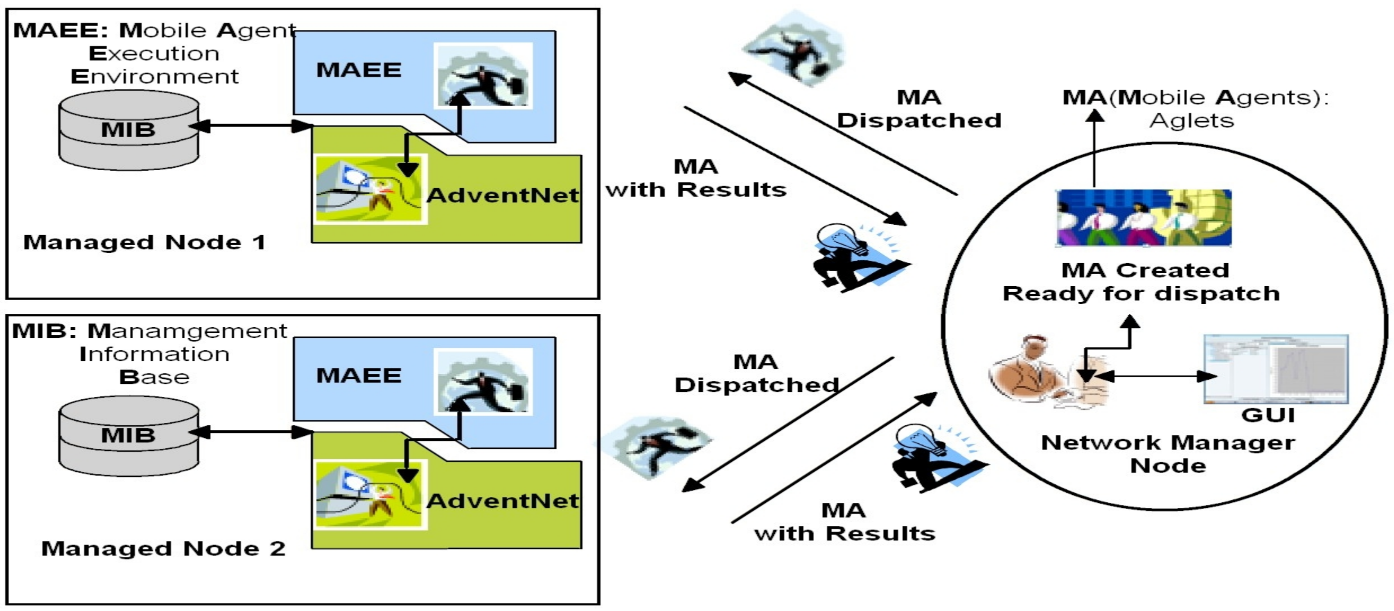


Fig. 3. Net Mobile-Cop Hybrid Framework Overall Architecture (Broadcast Model of Aglets)

Remote Programming approach of "Code-Shipping" using *'intelli-agents'* to effectively manage distributed networks.

Development Environment: We used **ASDK 2.0.2***(Aglets [4] **S**oftware **D**evelopment **K**it)* as the MAEE, **AdventNet SNMP API** [3] Std.4 having Java APIs to communicate between a manager and its managed node, *SUN's NetBeans 6.5* as *Java* (JDK 1.6)IDE and a mixture of NetBeans 6.5 and *Umbrello 1.4* to do the UML Modeling. The SNMP agents at the manager and managed nodes operate on SNMPv1 and SNMPv2c with the MIBs adhering to *RFC 1213* and HOST-RESOURCE MIB. The framework GUI was developed in *Java Swing 1.1.1* and graphing tool was *JFreeChart 1.0.9*.

Fig.3 above,shows that the Manager Node creates a *pool of Mobile Agents*ready for dispatch. The sys-admin is given a **GUI**, having hybrid controls of using traditional SNMP as well as Mobile Agents to do the Network Management tasks.To dispatch MAs to a particular managed node, the pre-requisite is that the node must have a **MAEE***(Mobile **A**gents **E**xecution **E**nvironment)*. Once the MA reaches the managed node, it gets the Network Management Parameters by talking to the **MIB** through the AdventNet SNMP API which acts as the SNMP Agent at the managed node.

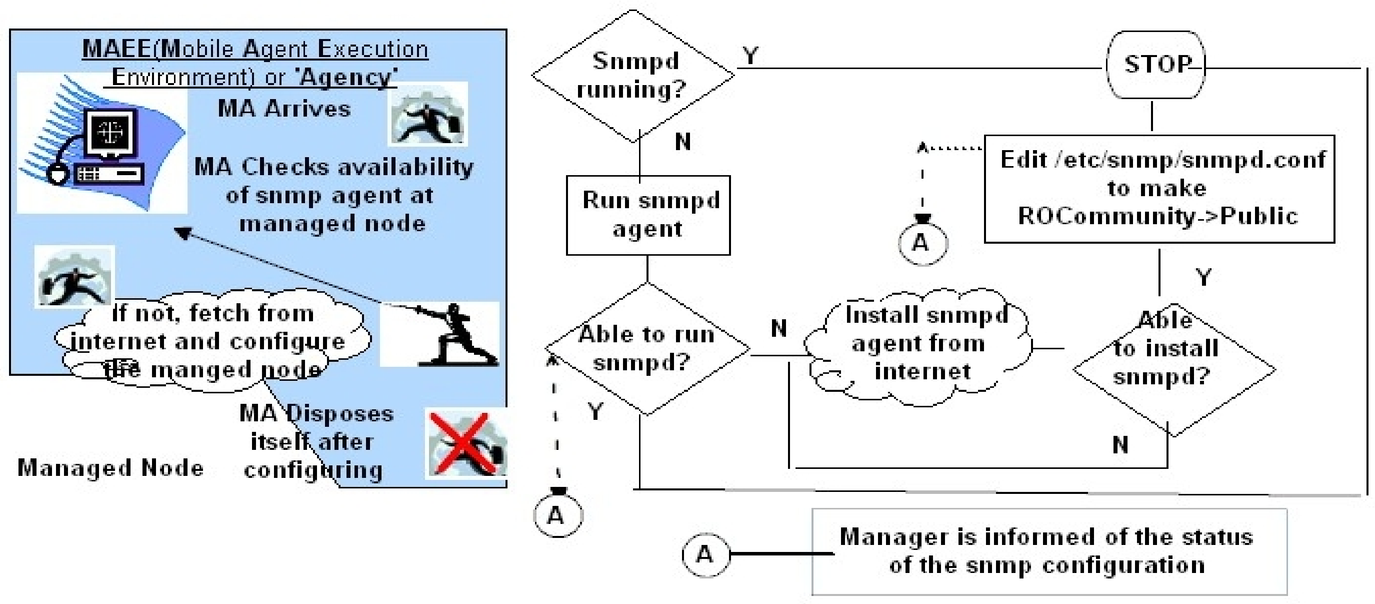

MAEE(Mobile Agent Execution Environment) or 'Agency'
MA Arrives
MA Checks availability of snmp agent at managed node
If not, fetch from internet and configure the manged node
MA Disposes itself after configuring
Managed Node
Snmpd running?
Y
N
STOP
Run snmpd agent
Edit /etc/snmp/snmpd.conf to make ROCommunity->Public
A
Able to run snmpd?
N
Install snmpd agent from internet
Y
Able to install snmpd?
Y
N
A
A
Manager is informed of the status of the snmp configuration

Fig. 4. Do 'n' Die by Intelligent Mobile Agents and Part of Flow Chart showing Dynamic SNMP Configuration)

As shown in Fig.2 above, the sys-admin is given a GUI to choose two modes of execution, namely,

- Using SNMP, we do *network account and performance monitoring on a* ***continuous basis***. The managing node polls all nodes under its purview periodically. This poll RPC call returns raw values to the manager, which has to now, process this data to meaningful information.

- Using MAs, we do *performance monitoring of the network on an* ***on-demand basis***. The manager creates the MAs and keeps them ready for dispatch in an MA pool. The sys-admin dispatches Aglets [4] either to a particular machine or broadcasts to all managed nodes, where the MA creates an SNMP session through *AdventNet[3]*, to talk to the *snmpd*, which in turn fetches the information from the MIB.

B. Advantage of Health Function Calculation using Aglets

An aggregation of variables is required to calculate a cumulative factor, called **HF**(***H****ealth* ***F****unction*), indicating the state or efficiency of a managed node. In the formula depicted in Fig.5 below, **ifSpeed**, **ifInOctects** and **ifOutOctects** of the interfaces group in MIB are used to compute the percentage utilization at an interface over a time interval. Here, the first set of values of Fig.5 implies that *ifSpeed* is the bandwidth of the interface, *ifInOctects* is the bytes *received* at time **x**, *ifOutOctects* is the bytes *sent* at time **x** and *(y-x)* is the polling interval. Refer Fig.2, above, for the result of this HF calculation, shown here as the graph of *bandwidth utilization* of a managed node

Here, the Aglets [4] collect and process the data, locally at the managed nodes. Only the *calculated values* are returned to the manager. Thus, the MAs remove processing load at the manager. In the same context, if conventional SNMP is used, the manager needs to send a get-request for **ifInOctects** and **ifOutOctects** at time **x** and repeat the same for time **y** and calculate the HF cumulative value. When we have to monitor a large number of machines, this is a huge processing overhead at the manager node. The formula for *bandwidth utilization* is given in the Fig.5 below.

C. Configuring the Network Element

It is essential to have **'snmpd'***(SNMP Daemon)* running at the managed nodes for our framework to work. Therefore, we configure the managed nodes that do not have *snmpd*, using Aglets [4]. Our novel strategy is *"Do 'n' Die"* mentioned in Section 3. As depicted in Fig.4, the Aglets [4] used are *'intelli-agents'*(Intelligent Mobile Agents) which first checks the availability of the SNMP agent at the managed node. If not present, they attempt to configure dynamically. Once they **"Do"** what they are programmed to do, they **"Die"**, that is they dispose themselves at the managed node.

The steps for configuration are depicted in Fig.4 above, in the form of a flow-chart. Firstly, the managing node attempts to find the nodes it is managing in the LAN.This is done by the process called **'Auto-Discovery'** where an *ICMP*(**I**nternet **C**ontrol **M**essage **P**rotocol) packet is broadcast to the network. At the end of this process, the **IP**(Internet Protocol) of the managed nodes becomes known to the managing node.

Then, the *Mobile Agents*(Aglets[4]) are dispatched to these nodes. The Aglets check whether **snmpd** is already installed, if so, finds out if it is running. If for some reason *snmpd* is not running, the aglet attempts to start it. If *snmpd* fails to start, the aglet will assume that the snmpd is corrupted. It

$$U(t) = \frac{((ifInOctects_y - ifInOctects_x) + (ifOutOctects_y - ifOutOctects_x)) \times 8 \times 100}{(y - x) \times ifSpeed} \quad (1)$$

Fig. 5. Formula for Bandwidth Utilization

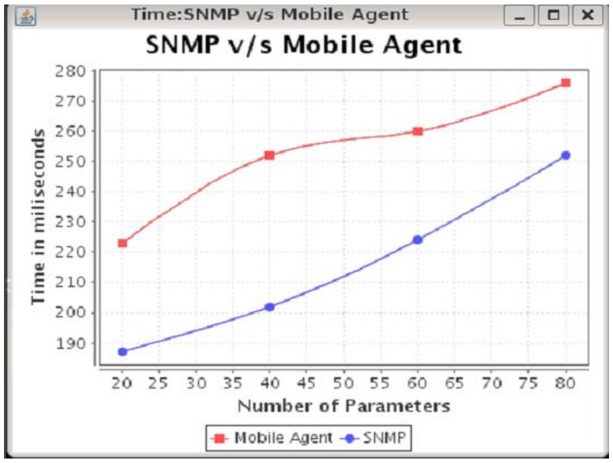


Fig. 6. Response Time Comparison between Mobile Agents and SNMP

TABLE I
RESPONSE TIMES OF SNMP AND AGLETS

Number of Parameters	Response Time	
	SNMP	MA(Aglets)
20 Parameters	187	**223**
40 Parameters	202	**252**
60 Parameters	224	**260**
80 Parameters	252	**276**

then dynamically downloads snmpd from the Internet. We have configured our aglet to install in the default path **/etc/snmp**. The aglet will then edit the *snmpd.conf* file to make the *RO Community* for snmpd **'Public'**. This is done in order to ensure that the communication paradigm is set between the manager and its nodes.

V. QUANTITATIVE EVALUATION AND ANALYSIS

In our framework, we give the flexibility of choosing between SNMP and MA to the sys-admin. Nevertheless, we give him a few recommendations as to when to stick to using SNMP and when to use MA, so he can exploit the full potential of the hybrid nature of this framework imbibing both the strategies. For our test bed, we consider a network of 5 managed nodes. The manager node is a SUN Blade 4000 series machine. All managed nodes are Intel X86 2.4 GHz, 512 MB RAM connected to a 10/100 Mbps Ethernet network, running SUN?s Solaris 10 version of UNIX. For the graphs shown below, we have used *JFreeCharts*.

A. Response Time

Table 1 above shows the response times to do account monitoring operations for **20**, **40**, **60** and **80**. parameters or **OIDs***(Object Identifiers)*. The timestamps are calculated using JDK 1.6 *System.currentTimemillis()*.In Fig.6 above, we see that **SNMP** *wins hand-down* with respect to response time, because the Aglets require time to migrate to the managed nodes. On an average, it takes *200 milliseconds*, while SNMP uses RPC to communicate remotely with the managed nodes, making the response time almost instantaneous. Thus, we prove that there is an inherent overhead associated with the creation and migration of *Mobile Agents* across managed nodes of a network.

B. Packet Size at the Managing Node for Network Health Function Calculation

Next, we used SNMP and MAs to calculate the cumulative value of the HF described in Section 4 B above. Table 2 below, shows the amount of data transferred to and from the managed node. We use this to monitor the values of *ifInOctects* and *ifOutOctects* at the **NIC***(Network Interface Card)*. We kept the *Polling Duration* constant for one set of values, changing the *Polling Interval*. The **Polling Duration** is the total duration during which we monitor the managed node. The PI **(Polling Interval)** is the interval of time the MA*(Mobile Agent)* interacts with the managed node's **snmpd**.

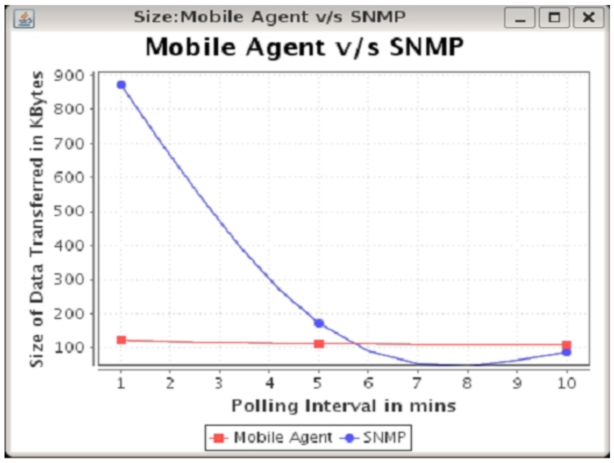
The experiment was conducted for a Total Duration of **1 hour** and **2 hours**. The number of parameters (MIB Objects) polled was kept constant as **80**. This was done to analyse the amount of data trasferred between the managed node and manager node when we use Mobile Agents and SNMP respectively.

The frequency of polls was increased rapidly. For example **6**, **12** and **60** polls for **one** hour and **12**, **24** and **120** polls for **two** hours. The first row of the Table 2 is interpreted below.

- Polling Durstion = **1 hour**
- Polling Interval = **10 mins**.
- Therefore, total number of polls made by the SNMP manager to the managing node = **6**.
- Number of remote requests from manager to managed node for SNMP using RPC = **12**.
- Number of remote replies from RPC for SNMP Poll request from manager to managing node = **12**
- Migration time for mobile agents to remote managed nodes = **200 ms**(average).
- Number of remote polls (request / reply) made by manager to managing node using mobile agents = **0**.
- Number of local polls made by the mobile agent at the managed node = **6**.

TABLE II
COMPARISON IN AMOUNT OF DATA TRANSFERRED OVER WIRE BETWEEN SNMP AND MOBILE AGENTS

Polling Duration	Polling Interval	No. of Parameters Polled	Response Time	
			SNMP	MA(Aglets)
1 Hour	10(6polls)	80	87.19	108.35
	5(12polls)	80	170.29	112
	1(60polls)	80	871.32	121.8
2 Hour	10(12polls)	80	178.56	112.15
	5(24polls)	80	362.72	115.2
	1(120polls)	80	1732.6	119.35



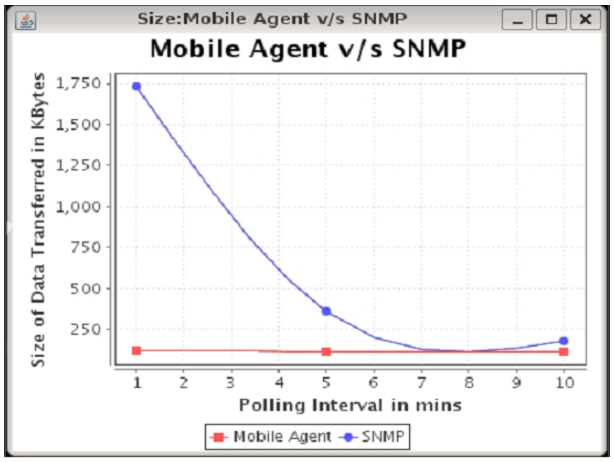


Fig. 7. Graphical Comparison of Size of Mobile Agents and SNMP with Polling Durations of 1 and 2 hours respectively.

We represent the values of Table 2 graphically, in Fig.7 using *JFreeChart*.From Table 2 and Fig.7, we observe that the amount of data transferred for Mobile Agents ***remains almost constant***, for various fast and slow polling intervals. This is because, the Mobile Agent migrates to the managed node, which has the MAEE*(Mobile Agent Execution Environment)* set up for it to execute.

The *Mobile Agent*, in our case, *Aglet*, talks to the MIB of the node **locally**, through the AdventSNMP Agent API deployed there. The exchange of MIB parameters between the Mobile Agent and the managed node happens at the node. The Mobile Agent then processes the data gathered, while it resides at the managed node. Only the ***processed data*** is returned back to the managing node.

In contrast, when the Polling Interval is less, meaning the number of polls is more, the amount of data transferred for conventional SNMP is more. This is because, in SNMP, we make many **RPC***(Remote Procedure Calls)* invocations during the Polling Duration to fetch the data remotely. These RPCs are made between the SNMP Agents at the managed and managing nodes.

In SNMPv1, to fetch each parameter, we need to make a *get-request* with its OID*(Object Identifier)*. Even if we use *get-bulk* of SNMPv2C, the size of the **SNMP PDU***(Protocol Data Unit)* may exceed beyond the threshold value of **64 KB** which is usually set for a *10/100 Mbps* Ethernet LAN.

Thus, in this case, ***the MA wins!*** This is because the MA calculates the cumulative value of the HF from the data it gathers locally from the SNMP agent at the managed node. This means, the mobile agent does not return with the raw data, but spends time at the managed node to locally process and calculate this raw data into meaningful HF value.

In addition, there is only **one** mobile agent sent to the managed node, for every Polling Duration. This Mobile Agent returns to the manager node, with the *calculated value* of the **HF***(Health Function)* after the Polling Duration.An example of a Health Function is given in the Formula for *Bandwidth Utilization* given in Fig.2 above. The execution result of the *Bandwidth Utilization* at one such managed node in our network is shown in Fig.5 above.

VI. Related Work

Gavalas [6] presents two approaches for MAs to tackle the problem of volume of data transfer, namely, *“Get ‘n’ Go”* and *“Go ‘n’ Stay”*. Both these strategies does only *Network Monitoring*. In our framework, we introduce a novel strategy called *“Do ‘n’ Die”* (Refer Section 3) to go beyond just **“monitoring”** the network, but **“managing”** it using *“intelligent mobile agents”*. Also, in *“Go ‘n’ Stay”*, the resource consumption at the managed node increases because the Mobile Agent resides at the node. In our strategy, once the task is done, the Mobile Agent is destroyed, demonstrating optimal usage of resources at the managed node.

Zapf [5] has proposed a hybrid model, called *NetDoctor*, based on *AMETAS* (**A**synchronous **M**essage **T**ransfer **A**gent **S**ystem. The Java-SNMP APIs required for this described in the paper are based on AMETAS. They claim that they provide seamless integration with any other Java SNMP package such as AdventNet[3], but have not shown any proof of compatibility or results. We use *well-known tried and tested* AdventNet Java SNMP API making our solution generic and portable.

Kona [7] and Arora [8]’s paper title gives the impression that their framework is used for *network management*, again like [6] the whole paper focuses on *network monitoring*. We have exploited the intelligence capability of mobile agents with our *“Do ‘n’ Die”* strategy that facilitates *network management* apart from simply *monitoring*. Therefore, unlike our work, presented in this paper, the intelligence capability of MAs is not fully exploited in both these implementations.

Pualiafito’s[13] gives an implementation called *MAP*(**M**obile **A**gent **P**latform), using two types of Mobile Agents, one called *Daemon Agents* residing permanently on the managed node, calculating Health Functions periodically. These calculated values of HFs are given to *Messenger Agents* which are dispatched to these nodes. Though a very impressive strategy, this method suffers from the disadvantage that it has the potential to increase resource utilization at the managed node. In our work, we introduced a novel *“Do ‘n’ Die Strategy”* (Section 3) to overcome the drawback of a potential increase in resource utilisation at the managed node, by ensuring that the Mobile Agent *“dies”* after it finishes what it is programmed to *“do”*.

Iwan [9]’s has presented a *simulation approach* using a network simulation toll called *OPNET Modeler*. They used it to test the applicability of mobile agents in a network with parameters simulated by the tool. This work is takes a very idealistic network where the links and nodes have *no load* and the links are *error free*. Such a situation is practically impossible to fathom in a real network scenario. We have tested our Net Mobile-Cop Framework on *real test beds*, not a simulated environement giving accurate, real-time results.

VII. Conclusion and Future Scope

The focus of our work was to effectively project the need to revive the need to employ Intelligent Mobile Agent technology for managing large distributed heterogeneous networks.

The interest in mobile agents as a design paradigm for distributed systems seems to have dwindled over the past decade [11], with the number of research groups working on mobile agent related research topics becoming smaller.

One of the reasons for this diminished interest could be the overhead associated with the creation and maintenance of mobile agents and the inherent non-reliability of its execution. It is certainly not a wise decision to design any architecture that only has mobile agents, however intelligent they are programmed to be. Especially so, for managing distributed systems such as heterogenous computer networks that we have chosen as a test bed to implement this work.

Therefore, one of the first and foremost conclusions we give here, is reiterate the need for a *Hybrid* framework combining plain SNMP as well as Mobile Agents. This is why we have named our *‘Net Mobi-Cop’* architecture as a Hybrid one, where the System Administrator can choose either of the paradigms, namely SNMP or Mobile Agents to manage his network.

Secondly, we confirm that the principle MbD (**M**anagement by **D**elegation) [15] is here to stay with mobile agents ruling the roost. The inherent advantage of this technique is the avoidance of the *’One Point Failure and Dependance Syndrome’* with the entire Network Monitoring and Management decisions to be done by the centralized managing node in traditional SNMP Managed Networks. With the agile powers of intelligent agents capable of travelling from one managed node to the other; Network Management tasks can be delegated to these *Mobile Agents*.

We have proved this with our implementation of *“Do ‘n’ Die”* strategy where the *Aglets* takes an autonomous decision to configure a managed node in which it finds *snmpd* either not running or corrupted. Thus, the managing node has *‘delegated’* the responsibility of making its managed node snmp enabled. Thus, with this work, we have demonstrated how *“Remote Programming”[10]* can be used to build *intelligent mobile agents*.

In our future work, we plan to add more intelligence into the Aglets to *analyse* **F**ault and **S**ecurity related MIB parameters and take autonomous, intelligent decisions without depending on the managed node for every action.

Thirdly, in our work, the focal point was to give a prototype implementation, tested on real networks.This allowed us to do network management by a seamless integration of traditional SNMP and Intelligent Mobile Agents. It gave us an ample opportunity to study the behaviour of mobile agents in actual local area networks, rather than simulated environments as done by some contemporary work such as [8].

In our future work we plan to test aglets behaviour over *HTTP*(**H**yper **T**ext **T**ransfer **P**rotocol) as opposed to its inherent *ATP*(**A**glet **T**ransfer **P**rotocol). We plan to choose real networks for this work also, so that the results are accurate and precise.

Lastly, we have given a bunch of useful tips (Refer Section 5) to the system administrator, when to choose between SNMP and Aglet strategies. This makes our framework is inherently robust wherein the sys-admin can fall back on one of the paradigms which is bound to work if the other fails.

This paper presents the first implementation of our *Net Mobile-Cop Framework*. Here, we have focused on the ***C****onfiguration*, ***A****ccounting* and ***P****erformance* aspects of **FCAPS [14]**. In our ongoing second phase, we plan to cater ***F****ault* and ***S****ecurity* aspects.

We also plan to employ advanced techniques like *SNMP Table Filtering* and explore the use of *customized* **HF** (Health Functions).As mentioned above, we also plan to do a comparative study of the difference in the usage of *HTTP* (Hypertext Transfer Protocol) with *ATP* (Aglet Transfer Protocol).We hope to explore other areas of management where *Mobile Agents* can be used like *Network Tomography*.

Acknowledgment

We have demonstrated our framework in ***SUN Tech Days*** held at Hyderabad in Feb 2009. We would like to thank the Technical Architects and the University Relations Team of SUN Microsystems, Bangalore. The strict milestone reviews conducted by them, to select our work for demonstration for the University Track of Sun Tech Days is greatly appreciated. We would also like to thank Mahesh V. Ghatage and Damodaran for their valuable technical inputs and the numerous frenzied discussions related to Java and Network Management! We are also indebted to Dr.Aswatha Kumar for providing the facilities of High Performance Computing, Sun Technologies and other Labs which we could use as test-beds for our work.

References

[1] J. E. White, “Mobile agents“ in Software Agents (J. Bradshaw, ed.), pp. 437-472, Menlo Park, CA: The MIT Press, 1996.

[2] Stallings W., “SNMP, SNMPv2, SNMPv3 and RMON 1 and 2“, 3rd ed., Addison-Wesley, 1999.

[3] The Advent Net Library. http://www.adventnet.com.

[4] Aglets Mobile Agents http://aglets.sourceforge.net

[5] Zapf M., Herrmann K., Geihs K., “Decentralized SNMP Management with Mobile Agents“, Proceedings of the 6th IFIP/IEEE Int. Symposium on Integrated Network Management (IM?99) May 1999.

[6] Gavalas D., Greenwood D., Ghanbari M., O?Mahony M.,“Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology“, Computer Communications, 23(8), April 2000.

[7] Manoj Kumar Kona and Cheng-Zhong Xu,“A Framework for Network Management using Mobile Agents“, Parallel and Distributed Processing Symposium. Proceedings International, IPDPS 2002.

[8] Vipan Arora and Harpreet Kaur Bajaj,“Effective Network Monitoring Using Mobile Agents“, Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007).

[9] Iwan Adhicandra, Colin Pattinson, Ebrahim Shaghouei,“Using Mobile Agents to improve performance of Network Management Operations.“,Proceeding of 4th Annual Symposium of Postgraduate Networking Conference,2003

[10] Christos Bohoris, George Pavlou, Antonio Liotta,“A Hybrid approach to Network Performance Monitoring based on Mobile Agents and CORBA“, SpringerLink LNCS, pages 151-162, Jan 2002

[11] Peter Braun, Ingo Mueller, Ryszard Kowalczyk, Steffen Kern and Friedrich-Schiller, “Attacking the Migration Bottleneck of Mobile Agents“, Faculty of Information and Communication Technologies Centre for Intelligent Agents and Multi-Agent Systems, Technical Report: SUTICT-TR2005.01, January 2005

[12] Raquel Trillo, Sergio Ilarri and Eduardo Mena, “Comparison and Performance Evaluation of Mobile Agent Platforms“, ICAS ’07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems,June 2007

[13] Antanio. Puliafito and O. Tomarchio, “Advanced Network Management Functionalities through the use of Mobile Software Agents“, in Proceedings of Workshop on Intelligent Agents for Telecommunication Applications (IATA?99), LNCS, vol.1699, August 1999.

[14] Mani Subramanian,“Network Management: Principles and Practice“, 2008 Edition, Perason Education

[15] Yemini Y., Goldszmidt G., Yemini S.,“Network Management by Delegation“, Proceedings of the 2nd International Symposium on Integrated Network Management, pp. 95-107, 1991.

[16] Niranjan Suri, “Mobile Agents“, Institute for Human and Machine Cognition, University of West Florida, Proceedings of the fourth international conference on Autonomous agents, pp 163-165,2000.

[17] C. Bohoris, G. Pavlou, H. Cruickshank,“Using Mobile Agents for Network Performance Management“, Journal of Network and Systems Management, pp 147-169, 2006

[18] McCloghrie K., Rose M.,“Management Information Base for Network Management of TCP/IP based internets: MIB-II“, RFC 1213, 1991.