A Study of the Behaviour of the Simple Network Management Protocol

Colin Pattinson
School of Computing, Leeds Metropolitan University, Beckett Park, Leeds LS6 3QS UK
E-Mail : c.pattinson@lmu.ac.uk

The long-standing dominance of the Simple Network Management Protocol (SNMP), in its various flavours (though particularly the original SNMPv1) is being challenged by the development of agent-based (and more specifically *mobile*-agent based) network management support. One of the criticisms of SNMP is that it suffers from a common failing of such client-server based paradigms, namely the performance problems caused by all traffic being directed to and from a single location (the network management platform). The behaviour of such client-server applications has been explored through the use of the OPNET simulation system, populating the model using data collected from a study of the way a network management platform is used both to monitor a properly-functioning network and to collect information in response to fault situations. Initial results are presented for version 1 of SNMP showing the behaviour of the resultant system over a network configuration which includes both local and wide area network links.

Keywords : Service modelling, supporting tools & platforms.

1. Introduction

Historically, the development of network management systems (NMSs) has relied heavily on the use of various versions of the Simple Network Management Protocol (SNMP). SNMP's development is well known, and the client-server nature of its operation, with requests for information being issued by a network manager via a network management platform (NMP) to software resident on the systems being managed is also well-documented. [1, 2] Intuitively, the traffic flows generated by SNMP can be related to the tasks typically undertaken in network management. These tasks are of two basic forms, "health function" monitoring, in which the variation of a count (or counters) over time is monitored [2, p.207], and "reactive" mode, in which the (human) network manager is alerted to a real or potential problem, and seeks to collect information to identify, isolate and rectify the cause of that problem [2, p.221]. An example of the first is the monitoring of load (utilisation levels) of a network interface at regular intervals, the second case might arise when the network manager is made aware of loss of connectivity on part of the network, and has to collect additional data to locate the problem more precisely.

The widespread availability of SNMP (particularly version 1 – SNMPv1) has meant that this has become the most common method of carrying out network management. More recently, however, it has been suggested that such methods (which rely on the raw network management data being transported to a central point – the NMP – for processing) are not appropriate for larger networks. In particular, the proponents of mobile agents suggest that this centralisation necessarily leads to scalability problems as network size increases.

O. Festor and A. Pras (Eds.) : 12th International Worshop on Distributed Systems: Operations and Management DSOM'2001 Nancy France, October 15-17, 2001.

In support of this argument, comparisons are made between the bandwidth consumed by an SNMP implementation and that required by a mobile agent to carry out the same task. For example, El-Darieby and Biezczad [3], suggest that a simple polling task in SNMP (checking the status of a network device at 1-minute intervals, to detect the occurrence of some problem as described in Leinwand and Conroy [2]) consumes 720KB per hour. This figure is then used to argue that an alternative mobile agent approach (where the agent is despatched reactively) uses less bandwidth provided there are fewer than 222 such problems in a 60-minute period – a figure unlikely to be reached while the network remains usable.

Baldi and Picco [4] describe the situation in which a SNMPv1 implementation responds to a problem by increasing its activity. "The NMS [Network Management Station] increases its interactions with the devices … increasing congestion. In turn, congestion … is likely to trigger notifications to the NMS" [4].

In the case where the initial problem is caused by, or gives rise to, network congestion, such increased activity can be counter-productive. Baldi and Picco quote a requirement to monitor the activity on a number of network interfaces (50 nodes, each with 30 interfaces, 5 MIB objects per interface) which leads to 335.6KB of data using SNMP. One of the major benefits claimed for mobile agents is that an increase in the management activity does not directly lead to an increase in the bandwidth required to monitor and manage those devices. In the centralised approach, there is a clear relationship between the volume of data transferred and the number of data items requested. Increasing the latter (through adding more managed elements, or by increasing the frequency at which information is gathered) directly affects the former. Gavalas et al [5, 6] show how the bandwidth requirements of a SNMPv1 implementation increase with the frequency of polling, as part of an argument for their proposed table filtering mobile agent.

In much of the mobile agent literature, there is the suggestion that the traditional SNMP approach may place unsupportable loading on the network being managed in circumstances such as those network management applications requiring "frequent polling of several MIB [objects]" described by Puliafito and Tomarchi [7]. Whilst the intuitive arguments presented by these and other authors are clear, the likely continuation of SNMP as a network management paradigm gives rise to the need for a more formal understanding of its behaviour.

It is therefore appropriate to study the operation of SNMP, in order to determine the nature of the load presented by network management tasks. This paper presents an initial analysis of the traffic loads generated by these tasks, and studies their effect on the performance of a typical network topology, through the medium of the OPNET simulation system [8].

2. Network Management Using SNMPv1

The typical NMP offers a broad range of monitoring tools, with performance and fault reports being presented, graphically, as counts versus time plots. The NMP user (the human network manager) is then expected to observe these reports and respond with management actions appropriate for the situation being observed. These reports are provided through the NMP issuing a regular flow of requests to agents, asking for the current value of a particular data object, conceptually part of the agent's management information base (MIB) (for example, the number of packets received by a particular Ethernet card). The response will comprise a near-copy of the request, but with the addition of a value parameter the variable binding or *varbind*) honouring the request. Values are represented in tag-length-value form, so it is possible to identify the number of bytes transferred in order to honour a request for a particular MIB object. *Table 1* shows this for each of the common MIB object types.

MIB Object Type	Number of Bytes
INTEGER + sub-types – assume 32-bit integers	3 – 7 (depending on value)
OCTET STRING + sub-types – assume length < 128	2 + string length
OBJECT IDENTIFIER	2 + (string length – 1)*
IP Address	6

Table 1. Number of bytes per object types.

2.1. Network Monitoring Tasks

Some common NMP tasks entail the monitoring of single MIB objects (a log of the number of TCP segments produced by a given system), whereas others require regular updates of a number of objects (the very common "interface utilisation" task requires a calculation involving the current values of three MIB objects). It is, therefore possible to identify the relationship between the size of SNMP data units (SNMPDUs) travelling to the NMP from the agent and those moving in the opposite direction, for a number of tasks. *Table 2* shows this for the tasks outlined in Leinwand and Conroy [2], in each case, the MIB objects involved are those used in [2]. This table reveals the general pattern of there being more *bytes* of data from agent to NMP than in the reverse direction. The table also reflects the way in which SNMPv1 requests typically carry varbinds containing ASN.1 NULL values, rather than a zero typed value. Therefore each OID-varbind used in the request is 2 bytes long, irrespective of actual data type. Note that the following table assumes the most efficient use of SNMPDUs (e.g. requests / responses for multiple MIB objects are carried in a single SNMPDU). Note also that the number of *packets* does not show the same relationship, as there is a very clear matched request-response pattern – except in the case of lost packets, ignored in this study†.

Note also that this table suggests the average overall SNMPDU size of 90 bytes used by Gavalas et al [7] is an appropriate figure to use.

From the foregoing then, a general picture can be developed of a "typical" network being managed under the SNMPv1 paradigm. An NMP is physically located at some point on the network (probably selected for operational / administrative convenience) and issues periodic requests (get, get_next and set to its agents, who respond as necessary. This is the basis of the health check activity, exemplified by the "Interface utilisation" activity in the first row of *Table 2.* In addition, the agents may issue unsolicited (trap) messages in reaction to events.

* Due to the way in which the first two elements of the object identifier are encoded into a single byte.

† The get_bulk operation of SNMPv2 and SNMPv3 does not show this correlation. In this case, the nature and size of the response may be significantly larger than the request. This is one reason for the focus of this paper on SNMPv1.

Activity	Objects		Variable binding size - Header plus	
	OID length	**#, type**	**Request**	**Response**
Interface utilisation (p.207) *n* = number of interfaces	11 (incl. 1 for index)	3 x Integer	*n* x 3 x (11 + 2) = *n* x 39	*n* x 3 x (11 + *3 to 6*) = *n* x 42 to *n* x 51
Routing fault resolution (pp. 213 – 215) *For each of m nodes*	14 (incl. 4 for index)	3 x IP address[+]	3 x (14 + 2) = 48 *m repeats*	3 x (14 + 6) = 60 *m repeats*
Routing fault resolution (table retrieval by ***get_next*** p. 214) *r* = number of rows *Requires a number of SNMPDUs*	12 (incl. 2 for index)	Integer + string (6) + IPAddr + Integer	*r* each ((12 + 2) *r* each (12 + 2) *r* each (12 + 2) *r* each (12 +2)) = *r* each 14,14,14,14	*r* each ((12 + *3 to 6*) *r* each (12 + 8) *r* each (12 + 6) *r* each (12 +*3 to 6*)) =*r* each 15,20,18,15 to *r* each 18,20,18,20
IP level errors (p.217)	8	8 x Integer	8 x (8 + 2) =80	8 x (8 + *3 to 6*) 88 to 104
IP level performance (p.219)	8	4 x Integer	4 x (8 + 2) = 40	4 x (8 + *3 to 6*) 44 to 54
TCP conn. establishment monitoring (p.225)	8	7 x Integer	7 x (8 + 2) = 70	7 x (8 + *3 to 6*) 77 to 98

Table 2. Bytes per SNMP activity.

3. Research goals

The above discussion gives rise to the following questions relating to the behaviour of the SNMPv1 paradigm in practical use :

- What is the overall loading created by a SNMPv1-based network management system, and does it have any impact on the normal behaviour of the network ?
- Does the location of the NMP (in terms of whether it is situated on a "limb" of the network or somewhere more central) affect the efficiency of the NMP ?
- What are the impacts of error conditions (or more precisely the user and system reactions to such conditions) on the overall data transfer pattern of the network ?

4. The research

In order to address these questions, a study has been undertaken of the data flows generated in a number of SNMPv1 network management scenarios. The "normal" conditions have been identified and described in *Tables 1* and *2* above and studies using a network management training package developed at Leeds Metropolitan University [11] have given an insight into the sort of data patterns generated by human network manager responses to "abnormal" circumstances.

[+] This is the best case scenario – where just the required single table objects are retrieved. The worst case is where complete tables are returned using a ***get-next*** sequence. For the *ipRouteTable*, this requires *r* x 13 ***get_next*** operations, (returning 9 integers, 3 IP addresses and one OID per row).

4.1. A Model of the SNMPv1 NMP-agent Interaction

Combining the information from *Tables 1* and *2* allows the development of a representation of the expected behaviour of the SNMPv1 NMP-agent relationship. There are two elements to this relationship : the regular health check process in which agents are polled and the results used to present behaviour trends; and the data flows produced following a problem report.

4.1.1. Health check activity

In the regular polling case, a situation is assumed in which the following health check trends are being monitored :

1. Utilisation of every device within the network. If there are *m* such devices, each with *i* interfaces, the NMP outputs *m * i* packets, and receives *i* packets from *m* different agents.
2. IP throughput of a sample end system from each subnetwork. If there are *e* such devices, the NMP produces *e* packets, and receives one packet from *e* different agents.
3. Packet loss and error statistics at IP and TCP levels on each server. If there are *s* servers, the NMP produces *s * 2* packets, and receives 2 packets from each of *s* agents. [*]

These activities correspond to *Table 2*'s rows 1, 5 and the combination of 6 and 8 respectively.

The frequency at which such requests are issued is left to the discretion of the network manager, the default settings are 450 sec for Sun's Network Manager [9] and 60 sec for tkined [10]. More frequent polling allows a more detailed trend analysis to be formed, but some (probably circumstance-specific) limit exists beyond which no additional information is provided, and the only effect is increased operational load. For the purposes of this experiment, a situation is postulated where trends 1 and 2 above are being monitored at 450-sec intervals, and trend 3 at 60 sec. This might suggest a network in which general overall performance is acceptable, but there is some concern over data loss in certain situations, loss and error counts are therefore being monitored at various levels to locate the cause of the problem. This gives an overall pattern of at most rather more than one session per minute, in each of which a number of packets are transferred in each direction, for each device. The exact number of such packets will vary according to factors such as the number of interfaces and exact type of device. So, for example, a simple workstation (i.e. neither a server nor a "sample end system") would be monitored for interface utilisation, involving 3 object instances every 450-sec. (See row 1 of *Table 2*). On the other hand, a "server" would be monitored for this activity and packet loss, further, a server would be expected to possess more that one interface (say three), so monitoring its interface activity might require nine objects every 450 sec. Superimposed on this would be a once-per-minute check of TCP and IP behaviour, contributing a further request / response pair every 60 sec. (See rows 6 and 8 of *Table 2*). Therefore, the majority of nodes will be monitored according to row 1 (once every 450 sec), with one per sub network being monitored once per minute.

4.1.2. Fault finding activity

Previous work by the author has resulted in the combination of a simulated network with a commonly available NMP, as described in [11], this tool is used to allow trainee network managers to gain experience of the indications and actions required in carrying out network management. The ability to simulate fault and other situations, while at the same time monitoring the actions of the user, allows us to identify the likely outputs / inputs (in terms of SNMP requests and responses) is particularly useful in the work now being described. It should be noted here that the actual activities observed in these situations are not necessarily the "best" in terms of using the fewest management calls – indeed, observations suggest that the general tendency for a human manager is to collect as much management data as possible, then to filter it at the NMP.

---

[*] Note, the behaviour described here represents a simplistic network manager operation – a more intelligent approach would combine all varbinds in a single SNMPDU.

Responding to ARP corruption involves determining ARP tables from each node. Assume *n* nodes, each table with *r* rows (Integer, String, IPAddr, Integer), therefore the NMP sends *n* * *r* requests, and each of *n* agents returns *r* responses, each request and response is 90 bytes long.

Routing fault detection makes use of the routing table for each node. Assume *n* nodes, each table with *r* rows (IPAddr, Integer, IPAddr, Integer, IPAddr – to retrieve destination, ifIndex, next hop, route type, net mask), therefore the NMP sends *n* * *r* requests, and each of *n* agents returns *r* responses, each request and response is 100 bytes long.

In comparison to the health check scenario, the traffic profile of fault finding is more concentrated, both in terms of overall duration, and volume. The empirical observations referred to above suggest that 1 – 2 minutes might be taken in initial data gathering, and that the quantity of data requested – in comparison with a health monitoring activity - might be tenfold (the emphasis now is on collecting full or partial tables, rather than individual objects). The frequency of "repetition" of fault-finding activity is not easy to quantify, faults typically do not occur according to some regular pattern. For simulation purposes, this activity is deemed to take place at 20-minute intervals, though it is stressed that this simply a means by which more than one such activity can be observed within a simulation.

The loading above, therefore, represents the pattern of SNMP traffic to be represented on the simulated network, when it is behaving normally, and to represent the loading in response to problem situations, which occur infrequently, but which impose a much heavier, though short-lived, loading. This analysis gives rise to *Table 3*, in which the parameters needed to generate representative SNMP traffic are presented. Note that in respect of number of messages, the worst-case situation is adopted, in which each object retrieval requires its own request / response packet pair, no multi-object requests are modelled.

Note these parameters are those required by the OPNET package to determine an application level configuration, for the purposes of this experiment, a "Session" is taken to be the actions required to complete a single task (a complete polling operation or a problem investigation) with a single agent; a "Request" is the information sent *from* the agent *to* the NMP, and a "Response" is that *from* the NMP *to* the agent. Whilst this may appear counter-intuitive, it must be remembered that the SNMP mode of operation has a data flow different from the typical client-server process. In other client-server models, a number of clients issue (usually short) requests to a single server, and receive (usually longer) responses. This is the model used by the OPNET simulator, which requires that there is a single server, which responds to requests generated by a number of clients. However, in the SNMPv1 model, the single NMP issues short requests to a number of agents, receiving long responses in return. Thus, in terms of the number of physical entities, the NMP takes on the role of the more normal server, whilst in data flow terms, the NMP is more like a client. Therefore, the model is developed as above, giving correspondence in both physical layout and overall data flow, whilst recognising that the model actually produces SNMPv1 responses *before* the associated request ! It is argued that this is not significant for the overall results, due to the small size of the requests in comparison to the responses.

Finally, the "Generation Rate" defines the number of packets generated over the lifetime of a session. So, for example, interface utilisation for a two-interface device (*Table 2*, row 1) requires 3 packets (n = 3, the two "real" interfaces plus the software loopback port). For these three packets to be produced within a one-minute "session" requires an hourly rate of 3 * 60 = 180 packets / hour. Similar considerations apply to the other "rate" parameters used.

The present model also assumes that the agent always has relevant raw data to hand, in other words, the (sometimes significant) time taken to fetch data from the monitored device and to construct the appropriate varbind is ignored. Work is in hand to address this shortcoming, although it should be noted that the tasks involved in this work are focussed on data which should be readily

available from the operating system, and therefore retrieved by operating system level calls in some SNMPv1 agent implementations [12].

Grouping	Parameter	Value	
		Health check	**Fault finding**
Session Information	Rate (per hour)	8/76	3
	Rate PDF	Constant	Constant
	Duration (mins)	1.5	1.0
	Duration PDF	Exponential	Exponential
Request Information	Generation Rate (per hour)	1800/180	6000
	Generation Rate PDF	Exponential	Exponential
	Packet Size	90	120
	Packet Size PDF	Poisson	Poisson
Response Information	Packet Size	90	150
	Packet Size PDF	Poisson	Poisson

Table 3. Parameters used in simulation

4.2. Test Scenario

The SNMPv1 model defined above was integrated into an OPNET network model comprising four inter-connected network segments across a 5-km^2 area (such as might be found in a typical University environment). The nodes (30 on each segment) are of the OPNET 10BaseT LAN type, with a set of 5 servers located on one of the segments. Inter-segment links are of 1.544Mbs. A dedicated server node was created as "the NMP" with all workstations in communication with that node for network management purposes only. This NMP node was located on one of the sub networks, so that traffic to and from nodes not also located on that same sub network would need to traverse the inter-network links. Other traffic was simulated using representative choices from the OPNET defined levels (Light, Medium and Heavy). The following parameters were studied :

- **Overall application (SNMP) response time** – to determine the expected SNMPv1 behaviour
- **Response time for a device situated on the same network segment as the NMP**
- **Response time of a device situated remotely from the NMP** – For comparison with the above to study the effects of "remoteness" on NM operations
- **Response time for other applications** – to evaluate the overall effect of SNMPv1 traffic on the network itself.

Three different test situations were established, a control situation with no SNMPv1 traffic, a network in which health functions were being monitored, and one where fault location was being undertaken. All other parameters remained unchanged throughout.

5. Discussion of results

5.1 The pattern of simulated SNMPv1 traffic

Plots of the traffic pattern generated by this simulated SNMPv1 activity show the overall loading generated by both health check and fault finding operations (*Figure 1*). Note the more even (but overall lower) pattern of health check requests and responses, symptomatic of a pattern of such requests. The fault finding activity has higher extreme values, consistent with the profile described earlier. This seems to suggest that, in so far as generated traffic patterns are concerned, this is an appropriate representation of SNMPv1 activity.

Further work is required to confirm this, and to establish the significance of other aspects of agent behaviour (including the time taken by the agent to retrieve information - mentioned earlier in this paper – Section 4.1.2).

5.2 The Wide Area Network results

The overall response times for http access (*Figure 2*) shows that there is little overall impact on the application performance, with the overall response time for HTTP (internet) page recovery being comparable, irrespective of the type of SNMP traffic. The likely explanation for this is that the simulated network has not neared its capacity, and therefore the additional simulated SNMPv1traffic does not cause problems of overloading. Earlier experiments, in which the overall simulated network management activity was significantly - and excessively - higher (by a factor of 5) did reveal some impact, as would be expected with higher overall loading. Further study is required to determine if, and in what circumstances such a quantity of network management traffic might be generated.

Finally, comparison is made of the response time for SNMPv1 activity experienced by a node located on the same subnet as the management platform (subnet_0) (*Figure 3*) and a node located on a different subnet (subnet_0_0) (*Figure 4*). The effect of the traffic having to traverse the slower WAN links does have impact on the response times, as can be seen from the vertical axes of these two graphs. Recall that these response times represent the time between (for example) a *get_request* being issued and the corresponding *get_response* being returned. If sufficient network management information is returned in that single exchange (or if multiple exchanges can be carried out in parallel) then these delays are probably within acceptable bounds. However, some exchanges are typically carried out sequentially (e.g. the traditional "tree walk" using a string of *get_next* operations). Further work is required to determine the circumstances in which this behaviour could become problematic.

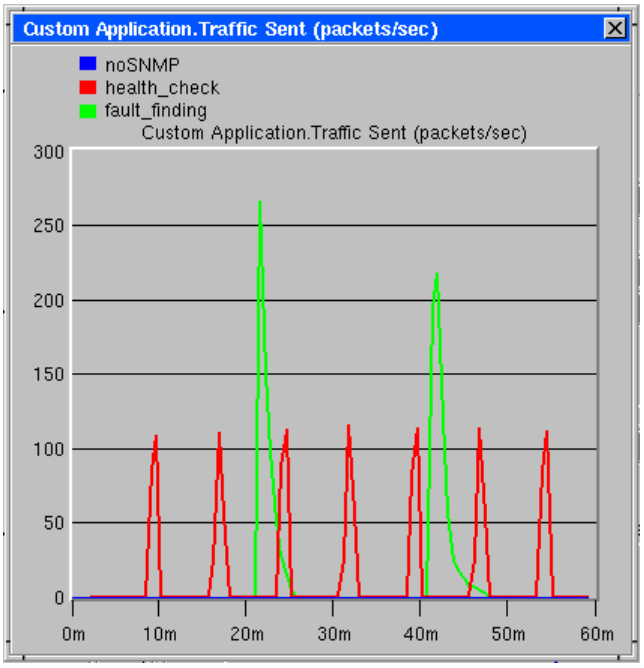


Figure 1. Traffic patterns for SNMPv1

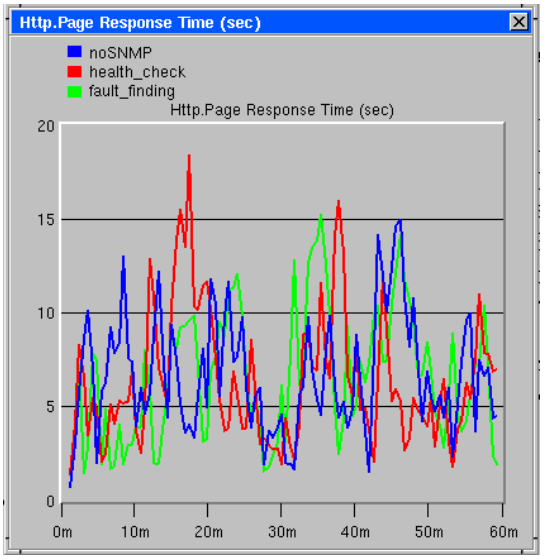
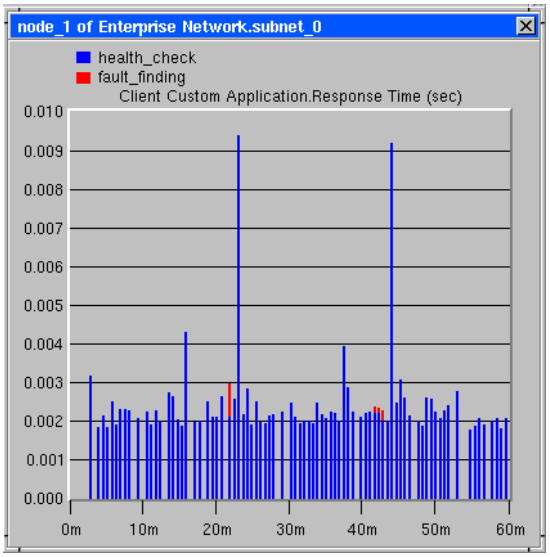
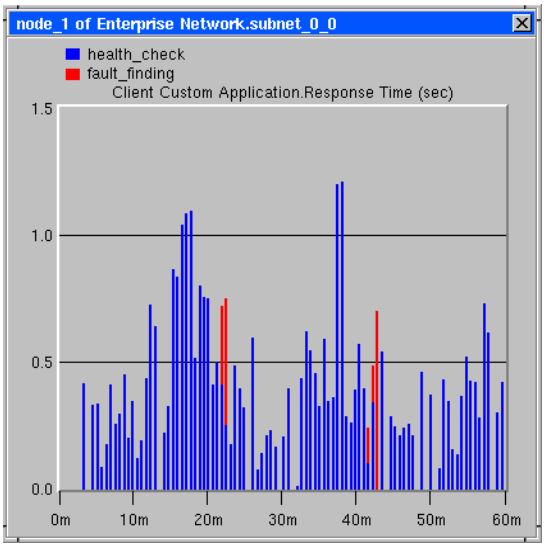


Figure 2. Http Response times for the WAN simulation





Figures 3 and 4. Response times for SNMPv1 activity for local (left) and remote (right) node.

6. Conclusions and future work

This work is only at the initial stages of development, so far the results appear to support the intuitive expectations of SNMPv1 behaviour, and to indicate that the relative locations of NMP and agent are important. However, there is much to be done, in particular, the pattern of packet generation in the fault finding scenario requires further analysis, and comparison with actual fault behaviour. Additionally, further work is planned to develop this study in the following directions :.

- To explore the limitations of the SNMP paradigm, with the intention of determining exactly what the circumstances are under which the use of SNMP to manage a network does create the problems described.

- To extend the model to simulate a more advanced SNMP implementation, in particular, the get-bulk operation introduced in SNMPv2.
- To develop a simulation of the mobile agent approach to network management in order to identify comparative behaviour.

Work has already commenced on each of the above issues, in order to address the very real issue of ensuring effective network management, while minimising the impact of management activity on the behaviour of the network being controlled.

7. Acknowledgements

The supportive and constructive comments of the anonymous reviewers are gratefully acknowledged.

8. References

[1] Network Management Principles and Practice, M. Subramanian, Addison Wesley 2000.

[2] Network Management A Practical Perspective 2nd ed. A. Leinwand, K. F. Conroy, Addison Wesley 1996.

[3] Intelligent Mobile Agents : Towards Network Fault Management Automation M. El-Darieby, A. Biezczad, Proc. 6th IFIP/IEEE International Symposium on Integrated Network Management (IM '99) pp. 611- 622.

[4] Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications M. Baldi, G. Picco Proc. 20th International Conference of Software Engineering (ICSE '98) pp. 146- 155.

[5] Advanced network monitoring applications based on mobile/intelligent agent technology D. Gavalas, D. Greenwood, M. Ghanbari, M. O'Mahony, Computer Communications, Vol. 23 (2000) pp. 720-730, April 2000.

[6] Enabling Mobile Agent Technology for Intelligent Bulk Management Data Filtering, D. Gavalas, M. Ghanbari, M. O'Mahony, D. Greenwood, Proc. DSOM 2000, pp. 623- 636.

[7] Using mobile agents to implement flexible network management strategies, A. Pulafito, O. Tomarchi, Computer Communications, Vol. 23 (2000) pp. 708-719, April 2000.

[8] OPNET Technologies, inc. [Internet] http://www.mil3.com [Accessed 6 August 2001]

[9] Sun Network Manager. Sun Microsystems Inc. system documentation.

[10] Schoenwalder,J. Tkined [Internet] Twente, Twente University. Available from : http://wwwhome.cs.utwente.nl/~schoenw/scotty/ [Accessed 20 March 2000]

[11] A simulated network management information base, C. Pattinson, Journal of Network and Computer Applications, Vol. 23 (2000) pp.93-107, April 2000.

[12] Carnegie-Mellon University snmp agent source code [Internet] London, Imperial College. Available from : ftp://src.doc.ic.ac.uk/computing/comms/tcpip/snmp/cmu-snmp [Accessed 20 March 2000] *The DSOM'2001 Golden Book*, O. Festor, A. Pras , *EIE Journal on Conference Management*, Vol. 15, No 1, Jan. 1997 (http://www.loria.fr/~festor)