Network Management Performance Analysis
And Scalability Tests: SNMP vs. CORBA

Qiang Gu, Alan Marshall
Advanced Telecommunications System Laboratory,
School of Electrical & Electronic Engineering,
The Queen’s University of Belfast,
Ashby Building, Stranmillis Road, BT9 5AH Belfast
Northern Ireland, U.K. +44 (0) 2890274248
{qiang.gu, a.marshall}@ee.qub.ac.uk

Abstract The performance of network management applications becomes an increasingly important factor as the scale of the networks they have to manage increases. This paper presents a series of comparative performance tests on two platforms used for network management: SNMP and CORBA. In terms of network performance, the two platforms are used to achieve the same network management tasks. The bandwidth consumed and the time taken to complete the tasks is compared. Both systems were tested by manipulating different types of management objects (single or tabular) under different network environments. The results also reveal the bottlenecks in each system and the reasons for them. The paper compares both systems and identifies their respective advantages and disadvantages. The work shows that their different features will decide what kind of application they are mostly suitable for.

Key Words – CORBA, SNMP, Performance, Network Management

1 Introduction
The Common Object Request Broker Architecture (CORBA) from OMG is the most widely used Distributed Processing Environment (DPE). Although CORBA has demonstrated an elegant flexibility and programmability, its performance and scalability is still questioned [11]. There have been a number of studies on analysing and improving a CORBA platform’s performance [1][2][3]. Using CORBA in network management is also a very active area [4][5][6]. However, these studies have tended to concentrate on using CORBA to achieve increased functionality, and very little work has been done on investigating CORBA’s performance in a specific application domain such as network management.
This paper attempts to answer part of the above question by comparing CORBA with a traditional network management system, SNMP. Section 2 provides the theoretical comparison between CORBA and SNMP. Section 3 describes a comprehensive set of tests on SNMP and CORBA. Section 4 and 5 gives the conclusion of this paper and suggest some possible future work.

In this paper, when CORBA is mentioned, the term client/server is used. When SNMP is mentioned, the term manager/agent is used. Because this paper is focused on the comparison between SNMP and CORBA, the CORBA client can be regarded as the equivalent of the manager in SNMP. For the same reason, the CORBA server is the equivalent of the SNMP agent.

## 2 Comparison between SNMP and CORBA

<table>
<thead>
<tr>
<th></th>
<th>SNMP</th>
<th>CORBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management object Structure</td>
<td>MIB Tree Static, Part of Standard</td>
<td>Naming Service Dynamic, Depend on Implementation</td>
</tr>
<tr>
<td>Object Description Language</td>
<td>SMI</td>
<td>IDL</td>
</tr>
<tr>
<td>Object Identification</td>
<td>OID</td>
<td>IOR</td>
</tr>
<tr>
<td>Management Object Access</td>
<td>Sequential</td>
<td>Random (Depend on the method)</td>
</tr>
<tr>
<td>Presentation Layer</td>
<td>ASN.1</td>
<td>CDR</td>
</tr>
<tr>
<td>Transport Protocol</td>
<td>UDP</td>
<td>TCP</td>
</tr>
</tbody>
</table>

There has to date been very little comparative performance analysis of network management systems. This section gives a theoretical comparison between SNMP and CORBA. The comparison focuses on the performance impact of their different characteristics.

Manipulation of large MIB tables is one of the most common tasks in network management, and it is also shown to cause significant delay. [13] Provides an algorithm on using basic SNMP operations (GET, GET-NEXT, GET-BULK) to achieve faster search in a large MIB table. Instead of retrieving sequential data from the agent, the manager uses binary search on the table. However, before the target can be found, the manager still needs to receive a certain number of SNMP messages (average is \( \log_2(n) \)) from the agent. Compared to this approach, a new function can be defined in the CORBA server. This function will return the entry
matching the criteria. The search is performed on the server. Only the target entry will be sent back to the client.

2.1 Information Models
In the SNMP protocol suite, the Structure of Management Information Version 2 (SMIv2) defines the information model of the protocol [7]. According to SMIv2, “Management information is viewed as a collection of management objects, residing in a virtual information store, termed the Management Information Base (MIB)”. Some MIB data is in the form of simple variables (e.g. sysDescr), while other data can be in tabular form (e.g. ifTable). MIBs are organized as tree structures. This reflects the information hierarchy inside managed devices. In order to identify the data in the MIB, a unique integer ID is assigned to every child of a parent node in the tree. Each node in the tree will have a unique Object Identification (OID), which is the sequence of integer IDs from the root to the current node. Because of historical reasons, the OID of all management objects begin with 1.3.6.1 (iso.org.dod.internet). This has introduced redundancy: A sender will needs to add these four bytes at the beginning of every OID even though the receiver always knows this header. There is currently a recommendation for OID compression [8].

In CORBA, the Interface Description Language (IDL) is used to describe the data structure and the operation on the data. Tabular data is represented as a sequence of structures. The objects in CORBA are identified and located with their Interoperable Object References (IOR). Because CORBA is not designed for any specific application, the organization of data is beyond the basic standards. However, CORBA has provided several services (e.g. Naming Service) to help organise data in the system.

2.2 Protocol Operations
One goal in designing the traditional management protocol is to use a small set of protocol operations (primitives) to support unlimited management objects and manipulations on them. In SNMP, all the management tasks are achieved by retrieving (GET) or updating (SET) the management objects’ values. A variation of GET, GET-NEXT, is used to when the OID of
target object is unknown. Although the conceptual structure of management objects is a tree, during the protocol interoperation, the objects are actually in a sequential order. Another variation of GET, GET-BULK, was introduced in SNMPv2 to improve the efficiency of retrieving large tables.

The approach of CORBA is very much different from SNMP. Every object can have an arbitrary number and type of operations. It is entirely dependent on the application’s requirement. The user can even add some more operations later to satisfy new demand. However, every single CORBA message can only support one method invocation on one object.

2.3 Presentation Layer
Both SNMP and CORBA were designed to accommodate a certain extent of heterogeneity. They both include the presentation layer in their protocol stacks. SNMP uses an adapted subset of OSI's Abstract Syntax Notation One (ASN.1), and CORBA uses the Common Data Representation (CDR). From a performance point of view, ASN.1 and CDR are more or less the same. However it should be noted that the presentation layer could introduce a significant delay, especially when CPU power is a factor (i.e. bottleneck).

2.4 Transport Mechanism
Classical SNMP uses UDP between manager and agent. In the exchange of shorter messages, UDP has better round trip delay. UDP also requires less system resources than TCP. RFC 3430 [14] has introduced another option: SNMP over TCP. According to the RFC: “SNMP over TCP is intended to be used when the size of the transferred data is large since TCP offers flow control and efficient segmentation. The transport of large amounts of management data via SNMP over UDP requires many request/response interactions with small-sized SNMP over UDP messages, which causes latency to increase excessively.”

CORBA mainly uses TCP. It can thus be expected that CORBA will have a greater resource usage than SNMP (e.g. too many open TCP connections). However, in a manner similar to
SNMP over TCP, CORBA should have better latency during the transport of large amounts of management data. Table 1 gives a summary of comparison between SNMP and CORBA.

3 Tests and Analysis
This section presents the set of tests used for comparison. In each test, SNMP and CORBA are used to achieve same management task, and their respective performances compared.

3.1 Test Environment

3.1.1 Software
The SNMP programs in the tests are based on UCD SNMP 4.2.3. The CORBA programs are based on TAO [10] minimum CORBA platform. There are a number of interesting studies from the TAO project that compare, analyze and improve the performance of CORBA platform [11][12].

3.1.2 Hardware
In the test, the SNMP manager/CORBA client will be run on a Windows XP PC with Pentium III, 498MHz CPU and 256 MB RAM. The SNMP agent/CORBA server will be run on a Windows 2000 PC with AMD Duron, 800MHz CPU and 256 MB RAM. These two machines are both connected to a Nortel Business Policy Switch 2000.

3.1.3 Software Tools
Several software tools are used in this test to monitor and measure the performance.

- **Ethereal Network Analyzer:** This tool is used to monitor the network traffic. The captured data can be used to measure the bandwidth and find out the bottleneck.

- **Rational Quantify:** The Rational Quantify is extremely useful to pinpoint application performance bottlenecks. Quantify can show what functions have been invoked, how many times they have been invoked, how long each invocation lasted. Most importantly, Quantify shows the time percentage of each function in the whole execution.

3.2 Test Systems
There are three different types of network management systems to be compared.
Network traffic volume and latency are investigated. Appendix 1 provides the test data used.

3.3 Test 1: Updating and Retrieving Single Management Object
In this test, the performance of manipulating single management objects was investigated. Since the bandwidth caused by retrieving or updating management objects are similar, only one of them (retrieving) was tested. Two scenarios are investigated. In the first, the SNMP manager/CORBA client retrieves only one single object’s value. In the second scenario, the values of multiple objects are retrieved. The GET-BULK operation in SNMP V2 will only have better performance over SNMP V1 when retrieving data in sequential order, however for generality we consider the MIB data to be retrieved randomly. For this reason, only SNMP V1 and CORBA were compared.

3.3.1 Traffic Volume

<table>
<thead>
<tr>
<th></th>
<th>One Single Object (bytes)</th>
<th>Three Single Objects (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client-&gt;Server</td>
<td>Server -&gt;Client</td>
</tr>
<tr>
<td>SNMP</td>
<td>71</td>
<td>70</td>
</tr>
<tr>
<td>CORBA</td>
<td>332</td>
<td>192</td>
</tr>
</tbody>
</table>

Table 2 shows the traffic volume used to retrieve one object and three objects. The number given is based on IP packets size. It can be seen that CORBA uses much more traffic to obtain the objects’ value. There are several reasons for this:

- CORBA uses TCP. The connection and disconnection of the TCP link will cost some traffic. For example, in the case of retrieving one single object, 192 bytes (58% of total traffic) were used by TCP link connection and disconnection. This is more than the application data. Because the same TCP connection can be reused, when using CORBA to retrieve three objects’ values, the traffic is less than three times that of retrieving one
object value. However, this can only happen when successive CORBA methods are invoked before IIOP time out. Otherwise, the TCP link will be disconnected. Any subsequent CORBA interoperation will then need another new TCP link.

- In the multiple objects scenario, SNMP can have even better performance. SNMP supports retrieving multiple objects’ value during one round of PDU message exchange. Compared to the retrieval of one object, the only data added is payload data. However, CORBA can only acquire one object’s value during every round of IIOP message exchange.

- SNMP uses UDP to carry its PDU. In the case of CORBA, user data will be encapsulated into IIOP message and then TCP packet’s data field. CORBA’s overhead is larger than SNMP. In SNMP, 54 bytes are used for IP, UDP and SNMP headers. In CORBA, the overhead can vary depending on CORBA’s service content and some other factors. In this test where no special CORBA service content is provided, the overhead from client to server is 107 bytes and the overhead from server to client is 64 bytes.

3.3.2 Latency

<table>
<thead>
<tr>
<th></th>
<th>One Single Object (ms)</th>
<th>Three Single Objects (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP</td>
<td>0.62</td>
<td>0.73</td>
</tr>
<tr>
<td>CORBA</td>
<td>6.3</td>
<td>7.8</td>
</tr>
</tbody>
</table>

Table 3 shows the latency measurement for SNMP and CORBA. Unsurprisingly, SNMP has a much shorter latency than CORBA. A large portion of CORBA’s latency is the result of the establishment of a TCP connection. For a task involving three single objects, three rounds of message exchanges will be needed for CORBA, whereas SNMP only needs one round.

Two points should be mentioned here:

- Management applications are not as time-critical as tasks such as routing. The CORBA application has managed to keep the delay under 10ms, which should be acceptable for most management tasks.
• In a real running environment, depending on the specific application, some changes can be made to improve CORBA’s performance. If a group of objects are usually accessed together, a new method can be defined in CORBA server to return the value of all the objects in the group. As the result, only one message exchange is needed. Because the methods is well known to both client and sever, the identification of each individual object is not included in the message. However, in the case of SNMP, the OID of every object is always necessary. When the number of objects in the group increases, CORBA will produce less traffic volume than SNMP. The trade-off of this improvement is that more code will be needed especially in the sever, which will increase the memory consumption.

3.4 Test 2: Updating and Retrieving Management Data in tables
In this test, the performance of manipulating table objects will be investigated. For the same reason as test 1, only the retrieval of table objects is examined.

![Figure 1 Increasing of BGP Routing Table Size [15]](image)

The dramatic change of network size and complexity has hugely increased the size of some management tables. An interesting example is the routing table (ipRouteTable in SNMP), especially the BGP routing tables in the core routers. Figure 1 is taken from Geoff Huston’s presentation in Internet2 Joint Techs Workshop, January 2001. He is also the author of RFC 3221, “Commentary on Inter-Domain Routing in the Internet”. The figure shows the increase of BGP routing table from year 1994 to year 2002. It can be seen that by the year 2002, the
typical size of BGP routing tables in core router have already reached more than 100,000 entries. These kind of large tables have become a challenge to the network management system.

In network management, one common task is to periodically poll the management data from every device in a domain. Most management data is in the form of tables, sometimes very large tables. The size of tables can significantly affect the traffic volume and delay of the task. As the number of devices in a domain increases, the polling task can be a major bottleneck. Consequently, a better mechanism for retrieving the table data can make a significant difference in performance.

This test shows how the variation of table size (table’s row number) can affect the performance of management system.

3.4.1 Traffic Volume
Figure 2 and Figure 3 show the traffic volumes between SNMP manager/CORBA client and SNMP agent/CORBA server when retrieving table data. The number of rows was varied from 1 to 2000. Every figure has two sub-figures. The sub-figure on the left shows the traffic volumes when rows number is between 1 and 100. The sub-figure on the right shows the traffic volumes when rows number is between 200 and 2000.

In this test, CORBA uses much less traffic than SNMP V1 and V2. As the table size grows, CORBA’s advantage becomes even more obvious. For example, in the case of 200 rows, CORBA uses 488 bytes from client to server in contrast to 4,702 bytes for SNMP V2 (almost 10 times) and 115,846 bytes for SNMP V1 (almost 240 times). In the opposite direction (from server to client), CORBA uses 10,872 bytes in contrast to 37,566 bytes for SNMP V2 (more than 3 times) and 121,418 bytes for SNMP V1 (more than 11 times). When the row number grows to 2000, CORBA uses 1,808 bytes from client to server whereas SNMP V2 uses 46,750 bytes (almost 26 times) and SNMP V1 uses 1,167,630 bytes (almost 650 times)! From server to client, CORBA uses 111,112 bytes whereas SNMP V2 uses 385,102 bytes (about 3.5 times) and SNMP V1 uses 1,225,818 bytes (more than 11 times). In this test, only the data
in the table is regarded as payload. Table 4 gives a list of the payload percentage for each system.

![Table 4 Payload Percentages](image)

<table>
<thead>
<tr>
<th></th>
<th>200 rows</th>
<th>2000 rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP V1</td>
<td>4.4%</td>
<td>4.4%</td>
</tr>
<tr>
<td>SNMP V2</td>
<td>24.6%</td>
<td>24.0%</td>
</tr>
<tr>
<td>CORBA</td>
<td>91.5%</td>
<td>92.1%</td>
</tr>
</tbody>
</table>

The differences between CORBA and SNMP can be explained from several aspects:

- In CORBA, a function was defined and implemented on the server side to return the entire table. At the beginning of the poll (after the TCP link was established), the client only needs to send an IIOP message to invoke this function. All subsequent traffic is merely table data (in TCP PDUs) from the server, and the TCP ACKs from the client. When the number of rows is increased, the traffic added from client to server is just more TCP ACKs. So even to retrieve a table with 2000 rows, the traffic from client to server is still very low. Because SNMP’s operation set is not extensible, SNMP cannot provide any new method suitable to the need of a specific management object. The consequence is
that SNMP uses a very inefficient way to retrieve the table data. SNMP V1 manager uses GET-NEXT operation and the OID prefix of the table. The polling task starts with acquiring the OID and value for the first data item (column 1, row 1 in the table). The OID obtained will be used in the next round of message exchange to retrieve the value and OID of the second data item (column 1, row 2 in the table). These message exchanges continue until the end of the table. As the result, the number of messages exchanged equals the table’s column number times its row number. For example, to retrieve a table with 2000 rows, CORBA uses 43 IP (TCP) packets from client to server and 77 IP (TCP) packets from server to client. SNMP V1 uses $2000 \times 8 = 16000$ IP (UDP) packets in both directions. Improvements to SNMP V1 can reduce the number of message exchanged down to the size of the number of rows. SNMP V2’s GET-BULK operation can further reduce the number of messages exchanged. However, it is still far less efficient than CORBA because each UDP packet is independent and as there is no flow control mechanism for UDP, the client has to make sure that the return data from server will not exceed the maximum UDP packet size, which is not very easy to estimate. The client can only take a very conservative approach. In the UCD implementation of SNMP, by default, GET-BULK only acquires 25 data items (UDP size) from the agent during every round of message exchange. In the case of 2000 rows, there are 1282 messages in both directions.

- A deeper investigation of packet contents reveals that CORBA’s packets are mostly very large. Almost all the IP packets from server have reached MTU (1,500 bytes). The IP packet size for SNMP V1 varies from 87 to 105 bytes depending on the size of user data. The IP packet size for SNMP V2 varies from 511 to 986 bytes. Obviously, the larger packet size means higher payload efficiency.

- In the task of retrieving an entire table, neither client nor server needs to know the identification of every single data item inside the table. In the CORBA system, only the raw table data will be sent from CORBA server to client. TCP’s flow control and message
sequencing guarantees correct ordering of the table data. However, in the SNMP RESPONSE message the agent has to include the OID for every data item. In a typical example, 4 bytes counter data might have an OID of 16 bytes. This means only 20% of the data is actually payload.

3.4.2 Latency
Latency is not a significant problem in manipulating single objects. Both CORBA and SNMP can archive reasonably small delay. However table objects normally have much more data than single objects and consequently, their manipulation can cause a major delay problem. In this test, the delay caused by network communication and the delays caused by computer computation are compared separately. This is helpful for isolating and locating any performance bottlenecks.

The test takes the following approach. Firstly, the client and sever are run on the same machine. In this case, the network bandwidth can be thought unlimited and the computer computation is the only cause of delay. In the second step, client and sever are placed on two machines and connected via 100 Mbps links to the Nortel BPS2000 switch. By comparing the results of these two steps, the bottleneck can be located. In the third step, the 100Mbps links will be replaced by 10 Mbps links. It will be helpful to investigate how CORBA and SNMP applications perform in various bandwidth environments. In every step, the same application was run 100 times and the average value obtained.

Figure 4 Local Delay of Retrieving Table
Figure 5 Remote Delay of Retrieving Table (100M link)

Figure 6 Remote Delay of Retrieving Table (10M link)

Figure 4 shows the delay of retrieving table data on local machine. Figure 5 shows the delay of retrieving table data on remote machine via a 100Mbp/s links. Figure 6 shows the delay of retrieving table data on remote machine via a 10Mbp/s links. From comparison of these figures, the following may be observed:

- In all three (local, 100Mbp/s links, 10Mbp/s links) steps, CORBA experiences higher delay than SNMP V1 and SNMP V2 when table rows are less than 20. After that, CORBA’s delay is lower than SNMP. As the row number increases, this advantage becomes increasingly obvious. From 1 to 2000 rows, CORBA’s delay has only increased from 12 ms to 50 ms. In contrast, the delay of SNMP V1 has increased from 2 ms to 4757 ms. In the case of SNMP V2, the delay has increased from 1.4 ms to 4472 ms.

- In all the runs, SNMP V2 always has lower delay than SNMP V1. However, when the row number has reached more than 700, the delay of SNMP V2 increases faster than SNMP V1. In fact, we found out that SNMP V2 had worse delay than SNMP V1 when the row number is more than 3000.
The results show that computation delay is the main bottleneck of SNMP applications, especially SNMP V2 applications. It can be seen that the local delay is even higher than the 100 Mbps links remote delay when the row number is more than 500. The explanation is that both server and client processes were using the same CPU. Either one, or both of them has a large amount of computations. It seems that the change on network link has rather little effect on the application delay. For example, the delay has only increased by about 12% when retrieving a 2000 rows table over 10Mbps links. On the other hand, different link speeds can decide where the CORBA application’s bottleneck is. When the network links are 100Mbps, the delay is almost the same as local delay. In this case, the computation delay is the bottleneck. However, running the same CORBA application over 10Mbps links resulted in a much higher delay. For example, the delay of retrieving a 2000 rows table via 10 Mbps links is about three times the duration spent on retrieving the same table from local machine or via a 100 Mbps link. This shows that the network delay becomes the bottleneck.

By using Rationale Quantify, a better understanding can be obtained on the cause of certain bottlenecks. Quantify was used to profile the execution of both client and server applications for SNMP V1, SNMP V2 and CORBA. Three table sizes were tested: 20, 200 and 2000.

Quantify can give a list of all the functions invoked along with the times they have been invoked and the duration. In this paper’s tests, functions can be classified into three categories:

- **Category 1**: The functions with very short running duration. These types of functions are ignored in the analysis because they don’t play a significant role in the application delay.

- **Category 2**: The functions with constant long running duration. This means that the number of rows in the table doesn’t affect the duration of these functions.

- **Category 3**: The functions whose duration increases as the row number increases. This duration can be significant.
Table 5 through Table 8 shows the Quantify’s profiling results for the CORBA application. A large portion of delay was caused by the functions from category 2. In the tests of retrieving tables with less than 200 rows, the client spent more than 50% of the time on these types of functions (Table 5). Function CORBA_ORB::string_to_object was used to parse the IOR string. In IOR, the remote machine’s address is normally given as domain name. This function also includes the enquiry routine for the DNS server to obtain the remote machine’s IP address. Function TAO_GIOP_Twoway_Invocation::start was used to establish the TCP connection between client and server. Table 6 gives a list of client functions from category 3. The function TAO_GIOP_Twoway_Invocation::invoke was used to exchange IIOP messages with server process on TCP link. This function’s duration actually reflects the network delay. The other functions were used by presentation layer routine. These functions are O(R) where R is the row numbers in the table. On the server side, most computation was spent on the presentation layer routine (the first 5 functions in Table 8).

Table 9 through Table 12 show the Quantify’s profiling results for SNMP V1 and SNMP V2. There is no function from category 2. This explains the reason why SNMP applications have very short delay when retrieving the table with few rows. Function NTWaitForSingleObject on the client side reflects the network delay. It can be seen that most functions are O(R). However, there are some exceptions in the lists worth attention. In SNMP V2, the number of calls and the time of function strcmp on client side are O(C*R^2) where C is the number of columns in the table. This exception can explain why the delay of SNMP V2 increases faster as row number is increased. Compared to SNMP V1, when the computation delay of strcmp becomes more significant than the network delay reduced, the overall delay of SNMP V2 will eventually become larger than SNMP V1. A further investigation of the code shows the reason why SNMP V2 calls strcmp much more times than SNMP V1. As mentioned in section 2.2, during the exchange of SNMP messages, the data in MIB table is represented as sequential items instead of tabular structure. However, the manager will need to reconstruct the data received back into tabular format. In the case of SNMP V1, manager uses GET-
NEXT to retrieve a single data item from the agent table. Before the arrival of agent’s reply, the manager already knows where to save the incoming data. Compared to SNMP V1, SNMP V2 manager can use GET-BULK to retrieve a list of data items. However, SNMP V2 manager has no knowledge where to save them in its memory space. It needs to call `strcmp` many times (dependent on the size of the table) to locate the appropriate position (row and column) for every data item in the list.

By checking the statistics on the agent side, it can be seen that both SNMP V1 and SNMP V2 agents have called the function `snmp_oid_compare` many times. They are both about $O(C\times R^2)$. In fact, despite the conceptual difference between GET-BULK and GET-NEXT, the SNMP agent normally uses one single set of basic functions to serve both of them. On receiving GET-BULK requests, the agent will just call the basic function of GET-NEXT for
certain times (according to the parameter in GET-BULK requests). SNMP V2 is the same as SNMP V1 in terms of agent complexity. One common routine of SNMP agent is to convert table structure into data items in sequential order. In the GET-NEXT request, the manager provides an OID. The agent needs to search the whole table to find the data item whose OID is the next in this sequential order. The function `snmp_oid_compare` was used to for the search. The number of calls on this function depends on the size of table.

### Table 9 SNMP V1 Functions in Category 3 (Client Side)

<table>
<thead>
<tr>
<th>Function</th>
<th>20 Rows</th>
<th></th>
<th>200 Rows</th>
<th></th>
<th>2000 Rows</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Calls</td>
<td>Time (ms)</td>
<td>Calls</td>
<td>Time (ms)</td>
<td>Calls</td>
<td>Time (ms)</td>
</tr>
<tr>
<td>NtWaitForSingleObject</td>
<td>21</td>
<td>4.87579</td>
<td>201</td>
<td>59.88488</td>
<td>2001</td>
<td>3077.22400</td>
</tr>
<tr>
<td>snmp_read</td>
<td>21</td>
<td>3.34186</td>
<td>201</td>
<td>26.28863</td>
<td>2001</td>
<td>256.23660</td>
</tr>
<tr>
<td>snmp_send</td>
<td>21</td>
<td>2.24539</td>
<td>201</td>
<td>23.41526</td>
<td>2001</td>
<td>199.42460</td>
</tr>
<tr>
<td>sprint_value</td>
<td>160</td>
<td>1.04851</td>
<td>1600</td>
<td>10.51261</td>
<td>16000</td>
<td>105.41380</td>
</tr>
<tr>
<td>Malloc</td>
<td>161</td>
<td>0.41802</td>
<td>1601</td>
<td>5.61363</td>
<td>16001</td>
<td>57.93994</td>
</tr>
<tr>
<td>NtDeviceIoControlFile</td>
<td>21</td>
<td>0.41158</td>
<td>201</td>
<td>4.18174</td>
<td>2001</td>
<td>43.80784</td>
</tr>
<tr>
<td>snmp_add_null_var</td>
<td>168</td>
<td>0.34238</td>
<td>1608</td>
<td>2.06821</td>
<td>16008</td>
<td>19.60241</td>
</tr>
<tr>
<td>Realloc</td>
<td>2</td>
<td>0.00975</td>
<td>20</td>
<td>0.53609</td>
<td>200</td>
<td>16.16901</td>
</tr>
<tr>
<td>snmp_free_pdu</td>
<td>21</td>
<td>0.31928</td>
<td>201</td>
<td>1.61586</td>
<td>2001</td>
<td>15.08310</td>
</tr>
<tr>
<td>sprint_oid</td>
<td>20</td>
<td>0.14406</td>
<td>200</td>
<td>1.44285</td>
<td>2000</td>
<td>14.45249</td>
</tr>
</tbody>
</table>

### Table 10 SNMP V2 Functions in Category 3 (Client Side)

<table>
<thead>
<tr>
<th>Function</th>
<th>20 Rows</th>
<th></th>
<th>200 Rows</th>
<th></th>
<th>2000 Rows</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Calls</td>
<td>Time (ms)</td>
<td>Calls</td>
<td>Time (ms)</td>
<td>Calls</td>
<td>Time (ms)</td>
</tr>
<tr>
<td>NtWaitForSingleObject</td>
<td>7</td>
<td>3.49196</td>
<td>65</td>
<td>50.32933</td>
<td>641</td>
<td>2645.778</td>
</tr>
<tr>
<td><code>strcmp</code></td>
<td>1660</td>
<td>0.03112</td>
<td>160600</td>
<td>2.99137</td>
<td>16006000</td>
<td>297.7468</td>
</tr>
<tr>
<td>snmp_read</td>
<td>7</td>
<td>2.45573</td>
<td>65</td>
<td>18.98845</td>
<td>641</td>
<td>181.0465</td>
</tr>
<tr>
<td>sprint_objid</td>
<td>161</td>
<td>1.16508</td>
<td>1601</td>
<td>11.60445</td>
<td>16001</td>
<td>116.1772</td>
</tr>
<tr>
<td>sprint_value</td>
<td>160</td>
<td>1.04851</td>
<td>1600</td>
<td>10.51261</td>
<td>16000</td>
<td>105.4138</td>
</tr>
<tr>
<td>snmp_send</td>
<td>7</td>
<td>1.10197</td>
<td>65</td>
<td>9.74953</td>
<td>641</td>
<td>101.5205</td>
</tr>
<tr>
<td>malloc</td>
<td>182</td>
<td>0.54171</td>
<td>1802</td>
<td>7.18821</td>
<td>18002</td>
<td>85.19605</td>
</tr>
<tr>
<td>NtDeviceIoControlFile</td>
<td>7</td>
<td>0.14928</td>
<td>65</td>
<td>1.37398</td>
<td>641</td>
<td>14.65163</td>
</tr>
<tr>
<td>snmp_free_pdu</td>
<td>7</td>
<td>0.2269</td>
<td>65</td>
<td>1.29122</td>
<td>641</td>
<td>11.44029</td>
</tr>
</tbody>
</table>

### Table 11 SNMP V1 Functions in Category 3 (Server Side)

| Function                        | 20 Rows |                | 200 Rows |                | 2000 Rows |                |
|                                | Calls   | Time (ms)      | Calls    | Time (ms)      | Calls     | Time (ms)      |
| snmp_oid_compare               | 2666    | 0.75658        | 170606   | 57.29104       | 16106006  | 5546.39500     |
| NtDeviceIoControlFile          | 64      | 1.91490        | 604      | 19.60717       | 6004      | 200.20880      |
| Memmove                         | 1126    | 1.55431        | 10846    | 14.88535       | 108046    | 148.20140      |
| Memcpy                          | 1188    | 0.27869        | 11808    | 2.77685        | 118008    | 27.77178       |
| Sendto                          | 21      | 0.25039        | 201      | 2.60289        | 2001      | 25.14542       |
| compare_tree                    | 1372    | 0.13873        | 13612    | 1.37633        | 136012    | 13.75223       |
A portion of CORBA application’s delay is used for establishing TCP connections. It is very common that a manager will retrieve more than one table during every polling on remote device. In this case, the retrieval of subsequent tables after the first one can reuse the same TCP link. Table 7 shows the delay of retrieving multiple tables in one poll. All the tables have the same structure. Each table has only 10 rows. It can be seen that after retrieving 3 tables, CORBA has the lowest delay.

![Figure 7 Remote Delay of Retrieving Multiple Tables](image)

### 4 Conclusion and Related Works

CORBA’s performance in network management is a common concern. However as the size of managed networks increase the scalability of traditional network management applications such as SNMP also presents issues. In this paper, the performance of SNMP and CORBA systems are studied and compared when considered for network management applications.

The results show that SNMP and CORBA demonstrate very different characteristics. When manipulating small amounts of data (e.g. single object, small table), SNMP costs less network bandwidth and has less latency than CORBA. This is mainly because of different
transportation layer protocol they use: UDP for SNMP or TCP for CORBA. However, for a management application, CORBA’s latency is still acceptable (less than 10ms). Manipulating large tables is another typical management task. This is very different from traditional primitive-driven control of remote objects. A large percentage of delay in SNMP applications is caused by the conversion between conceptual data format (e.g. tabular format) and the sequential data format, which can be communicated and interpreted by the SNMP entities. In this type of task, CORBA consumes much less bandwidth and experiences less delay than SNMP. The results show that this type of application benefits from CORBA’s flexible distributed programming architecture.

In general, both SNMP and CORBA have their advantages and disadvantages. Which one to choose or what trade-off to make is entirely dependent on the specific system. CORBA is more suitable in applications where there is a large amount of data to be exchanged and memory consumption is not a major concern. However, in systems where only a small amount of management data is to be exchanged and the memory is restricted, SNMP will have better performance.

5 Future Work
Some other performance and scalability issues are not addressed in this paper. One of them is the application size and memory consumption of each system. Another is the exhaustion test. For example, users can try to add as many objects as possible until eventually the system crashes of memory overflow, CPU degrading or bugs in the code. In another example, users can send as many requests as possible from different manager/client until the latency deteriorates to a very low level. We believe these can be some interesting topics for future research.

Acknowledgment
The authors would like to thank the Industrial Research and Technology Unit (IRTU) of Northern Ireland for their financial support for the research carried out in this work. The
technical support from NITEC (Northern Ireland Telecommunications Engineering Centre, of Nortel Networks) is also gracefully acknowledged.

Reference
Appendix I: Test Data

Only a small portion of code is given here for demonstration and comparison. The test has also defined several single manager objects. The code of their definition is omitted here.

```
QubEntry ::=  
  SEQUENCE {   
    qubTabIndex INTEGER,   
    qubTabDescr DisplayString,   
    qubTabIpAddr IpAddress,   
    qubTabCounter1 Counter,   
    qubTabCounter2 Counter,   
    qubTabCounter3 Counter,   
    qubTabCounter4 Counter,   
    qubTabCounter5 Counter   
  }
```

**Code Segment 1 SNP MIB Definition of the Table Structure**

```
typedef sequence<octet,4> IpAddressType;
struct QubEntry { 
  unsigned long qubTabIndex;
  string qubTabDescr;
  IpAddressType qubTabIpAddr;
  unsigned long qubTabCounter1;
  unsigned long qubTabCounter2;
  unsigned long qubTabCounter3;
  unsigned long qubTabCounter4;
  unsigned long qubTabCounter5;
};
typedef sequence<QubEntry> qubEntry_Seq;
readonly attribute qubEntry_Seq qubTable;
```

**Code Segment 2 CORBA IDL Definition of the Table Structure**


