Integrating Android Devices into Network Management Systems based on SNMP

Fernando Hidalgo
Escuela de Computación
Central University of Venezuela
Caracas, Venezuela

Eric Gamess
Laboratorio de Comunicación y Redes
Central University of Venezuela
Caracas, Venezuela

***Abstract*—Mobile devices are becoming essential for today life. In developed countries, about half of the people have a smartphone, resulting in millions of these electronic devices. Android is the most popular operating system for smartphones and other electronic devices such as tablets. Hence, for network administrators, it is essential to start managing all the Android based devices. SNMP is the de facto standard for network administration, where agents that are running in managed devices are polled by management stations. Some primitive tools have already been developed to transform an Android device as a basic management station. However, so far, there is no SNMP agent for this operating system. In this paper, we develop the first SNMP agent for Android. We also propose an SNMP benchmark to study the SNMP traffic that can be supported by our SNMP agent over some real and actual Android devices. The results obtained show that it is realistic to integrate mobile Android devices in network management systems since they can handle a high number of SNMP requests in a reasonable period of time.**

Keywords—Network Management Systems; SNMP; Android; Performance Evaluation; Benchmarks

I. Introduction

All over the world, millions of cell phones have been sold. In developed countries, smartphone penetration among cell phone users are around 45% in US and 50% in Europe, resulting in a huge number of new devices with a high processing power.

Even if the majority of smart phones and tablets are owned and operated by people, for an organization it can be very useful to integrate these devices in its network administration system. That is, the organization could manage the smart mobile devices that are used for its operation, and the smart mobile devices of its employees with their consent.

The industry standard for network administration is called Simple Network Management Protocol (SNMP) [1][2], and has been around for more than two decades now. SNMP is based on the client/server model, where clients are also referred as network management stations and servers are agents running in the administrated devices. In SNMP, management stations regularly poll agents for information. Even though SNMP is widely spread, it is still rare in these new mobile devices. Most of the SNMP applications proposed for mobile devices are small management tools that allow a limited monitoring of classical network devices such as servers, switches, or routers from the mobile devices.

There are several Operating Systems (OSs) proposed for mobile devices, such as Symbian [3] from Nokia, BlackBerry OS [4] from BlackBerry Limited, iOS [5] from Apple, Windows Phone [6] from Microsoft, and Android [7][8] from Google. Recently, Android has taken the most important part of the market with this OS installed in more than 70% of the smartphones shipped. To the best of our knowledge, there is no SNMP agent for Android, making it impossible for network administrators to incorporate these devices to their monitoring systems. In this work, we develop an SNMP agent for Andriod. It implements many of the SNMP objects and can be used to integrate Android mobile devices to network management systems. To study the SNMP traffic that can be handled by our SNMP agent in real Android based devices, we also develop an SNMP benchmarking tool to get two important metrics (Response Time and Reply Request Ratio). The results obtained show that it is realistic to integrated mobile Android devices in network administration systems since they can handle a high number of SNMP requests in a reasonable period of time.

The rest of this paper is structured as follows. Section II presents the related work. In Section III, we introduce SNMP, the de facto network management protocol. Android is shortly described in Section IV. The development of our SNMP agent is discussed in Section V. Section VI is focused on evaluating the SNMP traffic that can be handled by our agent in real life Android devices. Finally, the paper concludes in Section I.

II. Related Work

Despite of its popularity among developers and users, just a few applications have been released for Android in the field of network administration based on SNMP. SNMP MIB Browser, created by ZOHO Corporation, can be downloaded for free from Google Play, formerly known as the Android Market. Google Play is a distribution platform for applications for the Android operating system operated by Google. SNMP MIB Browser enables users to browse/view the MIB data of SNMP enabled network devices such as servers, switches, routers, etc., from an Android based device. Users can load any standard MIB and fetch values from the managed devices to show the MIB data in an intuitive manner, just by clicking an object, a group of objects, or a table. It supports all the versions of SNMP (v1, v2c, v3). In the case of SNMPv3 [9], it implements MD5 and SHA algorithms for authentication, and DES, 3DES, AES-128, AES-192 and AES-256 algorithms for encryption.

SNMP Manager is similar to SNMP MIB Browser. That is, it is another basic application for Android that allows users to browse/view MIBs. Its support for SNMPv3 is actually in development. However, it has additional features such as trap reception and transmission. It can be freely downloaded from Google Play.

ezNetScan[1] is a free application developed by VR Software Systems Pvt Ltd from India, for network administration for mobile devices. It has been ported to Android and can be downloaded from Google Play. ezNetScan can display basic information of WiFi networks, scan the WiFi networks the Android device is connected to, and display information about the other connected devices, reporting their IP addresses, hostnames, MAC addresses, and more. Through SNMPv1 or SNMPv2c, ezNetScan can do basic operations and collect information about monitored devices such as operating system, file system capacity, installed applications, and running processes. Other features of ezNetScan include TCP port scanning, ping, and traceroute.

SNMP Traffic Grapher is a basic application that can be downloaded from Google Play. It allows real-time graphing of two SNMP Object Identifiers (OIDs) at the same time. The idea is to monitor the download and upload streams that transit the interface of a classical network device such as a server, switch, or router. The first OID will be graphed in green (download) and the second in blue (upload). The application retrieves the SNMP data with SNMPv2c.

SNMP Trap Agent[2] is a commercial product developed by Maildover LLC that provides a limited monitoring of Android phone's performance (percentage of busy CPU), usage (battery charge level, percentage of free memory), and location (latitude and longitude). It uses SNMP traps to send the selected information to the management stations. Even if it allows some flexibility for administration, it is not an SNMP agent since it is restricted to SNMP traps and cannot respond to SNMP get- and set-requests (GetRequest, GetNextRequest, and SetRequest).

As reported in this section, most of the development done so far is limited to SNMP managers, i.e., applications to monitor network devices. Unlike this previous work, our work bring the first implementation of an SNMP agent to Android.

III. Simple Network Managment Protocol

Simple Network Management Protocol [10][11] (SNMP) is a protocol for network management defined by the Internet Engineering Task Force (IETF) that is widely used since it is simple and easy to implement. SNMP is an application layer protocol that facilitates the exchange of management information between agents (managed devices) and Network Management Systems (NMSs). NMSs are also called managers. It is part of the TCP/IP protocol suite and uses User Datagram Protocol (UDP) as a transport protocol. Agents listen to queries on UDP port 161, while NMSs received traps on UDP port 162.

SNMPv1 specifies five core Protocol Data Units (PDUs): GetRequest, GetNextRequest, SetRequest, GetResponse and Trap. GetRequest is sent by a manager to retrieve the value of some objects managed by an agent. GetNextRequest is used iteratively by a manager to get tables or subtrees from administrated systems such as the Address Resolution Protocol (ARP) cache, or the routing table. SetRequest is used by a manager to modify an object in a managed device. GetResponse is sent by agents to respond with data to get-requests (GetRequest, GetNextRequest) and set-requests (SetRequest). Trap is used by agents to report an alert or other asynchronous events to managers. SNMPv1 does not allow manager-to-manager interactions as SNMPv2c and SNMPv3 do.

SNMPv2c is a revised version of SNMPv1 and includes improvements in the areas of performance, manager-to-manager communications, and error-handling. Three new PDUs were added in SNMPv2c: GetBulkRequest, InformRequest, and Report. The purpose of GetBulkRequest is to request the transfer of a potentially large amount of data including, but not limited to, the efficient and rapid retrieval of large tables. Compared to GetNextRequest, GetBulkRequest minimizes the number of requests and responses necessary to complete the transfer. InformRequest is sent by a manager to provide management information to a remote manager. Usage and precise semantics of Report are not specified in [2]; therefore, any SNMP administrative framework making use of this PDU must define it. The SNMPv2c improved error-handling includes expanded error codes that distinguish different kinds of error conditions; these conditions are reported through a single error code in SNMPv1.

A Management Information Base [11] (MIB) is a formal description of a set of network objects that can be managed using SNMP. Standard minimal MIBs have been defined (MIB-I, MIB-II, Host Resources MIB, etc), and vendors often have private enterprise MIBs. MIB-I [12] was defined to manage TCP/IP-based internets. MIB-II, defined in [13], is basically an update of MIB-I. Another fundamental concept of SNMP is the notion of Object Identifiers (OIDs). An OID is a tag that allows a management entity to refer unambiguously to a particular object. OIDs are allocated in a tree fashion and described in the MIB. The value of the OID is a sequence of integers that refers to a particular traversal of the object tree.

IV. Android Operating System

Android Inc. was founded in Palo Alto, California in October 2003 to initially develop an advanced operating system for digital cameras. However, with the goal of reaching a bigger market, Android Inc. diverted its efforts to produce a smartphone OS to rival those of Nokia, BlackBerry Limited, Apple, and Microsoft. The first version of Android was unveiled in November 2007. It is based on the Linux kernel, and designed primarily for touchscreen mobile devices such as smartphones and tablets. It is also used in televisions, games consoles, digital cameras, and many other electronic devices.

On August 2005, Google acquired Android Inc. Part of the success of Android is due to its license. Android source code is released by Google under the Apache License, which allows the software to be freely modified and distributed by device manufacturers, wireless carriers, and the community.

[1] http://www.eznetscan.net
[2] http://www.maildover.com/eurotrap.html

The user interface of Android is based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching and reverse pinching to manipulate on-screen objects. Internal hardware such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented.

As of May 2011, Android has become the leader of the market for mobile OSs, having the largest installed base in almost all the countries of the world. More than one million applications for Android are available to be downloaded from Google Play. Also, it is by far the most popular platform for mobile OSs developers. For all the previous reasons, we decided to develop an SNMP agent for this well accepted architecture.

V. SNMP Agent For Android

Our SNMP agent runs in background and was developed by extending class `Services` from the Android Application Programming Interface (API). It is configured through a GUI (Activity) and some of the parameters that can be set include: the version of SNMP, the read-only community, the read-write community, and the UDP port where the agent is listening (see Figure 1). Users can start and stop the service through a simple switch in the GUI.

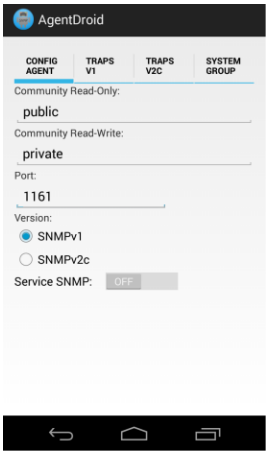

Fig. 1. Interface to Configure the Android SNMP Agent

For the implementation of SNMP, we used a Java package called `SNMP Package`[3]. Our agent supports both, SNMPv1 and SNMPv2c. We implemented all the SNMP messages: GetRequest, GetNextRequest, GetBulkRequest, SetRequest, GetResponse, Trapv1, and Trapv2c. We also added a module for the configuration of some important OIDs of the System Group, such as `sysContact`, `sysLocation`, and `sysName` (see Figure 2).

We implemented all the OIDs of MIB-II in our agent. It also supports part of Host Resources MIB [14], such as the Installed Software Group (hrSWInstalledTable), allowing the

Fig. 2. Interface to Configure Important Objects of the System Group

monitoring of the actual version of the installed applications or the installation of new software. The storage of read-write OIDs is done in a database kept in the system.

We made exhaustive tests to validate our SNMP agent for Android using SNMP JManager [15], a famous Java based manager that we developed a few years ago, and freely available from SourceForge[4]. We also captured the traffic with Wireshark [16] to validate the PDUs generated by our agent.

VI. Performance Evaluation

To evaluate the performance of our SNMP agent for Android, we decided to test its capacity over different Android based devices. Two metrics are important when evaluating the performance of an SNMP agent: response time and Reply Request Ratio (RRR). Response time is the time elapsed between sending an SNMP request (GetRequest, GetNextRequest, or SetRequest) and receiving the corresponding reply (GetResponse). RRR is the ratio between the number of replies received and the number of requests sent. This last metric is very useful when evaluating the number of requests that can be handled in one second or to study the behavior of the agent in stressed conditions. To the best of our knowledge, there is no tool developed so far to evaluate the performance of an SNMP agent. Hence, we developed our own benchmarking tool.

For the response time, it is important to report a meaningful time, i.e., a time that is not based on the sending of a unique SNMP request and the reception of its associated reply. A response time based on a unique SNMP request will have an important error, due to (1) clock precision and (2) the possibility of heavy OS processes being executed during the test, such as disk swapping, which will alter significantly the final results. Hence, our benchmark to measure the response time is based on the client/server model. Basically, an SNMP request (GetRequest, GetNextRequest, or SetRequest) with a fixed number of OIDs is sent from the benchmarking tool (client) to the agent (server) a number of times.

[3] http://jsevy.com/snmp

[4] http://sourceforge.net/projects/snmpjmanager

As soon as the reply arrives to the benchmarking tool, the next request is sent, so there is no idle time introduced by the benchmarking tool between two consecutive requests. We take a timestamp before and after the interchange. The difference of the timestamps is divided by the number of time the SNMP request is done, to get the average response time. For the response time test, our benchmarking tool has two parameters: (1) the list of OIDs to be fetched or modified, and (2) the number of time the request must be done before obtained the total response time.

In the case of the RRR test, our benchmarking tool has three parameters: (1) the list of OIDs to be fetched or modified, (2) the frequency of the requests or the number of requests that must be done in a second, and (3) the duration of the experiment.

For a better explanation, let say that the frequency is 5 and the duration is 30 seconds. Therefore, the benchmarking tool will be sending 5 requests every second, during 30 seconds, i.e., a total of 150 requests will be sent during the experiment. The benchmarking tool will also count the number of received replies, which must be less than or equal to 150. At the end of the experiment, the RRR is computed by dividing the number of replies received by the number of requests sent.

We tested the performance of our agent with two different Android devices: (1) a Galaxy Nexus i9250 phone and (2) a Galaxy Tab 8.9 tablet. We chose these devices since they are widely spread all over the world. We also used some PCs to run the benchmarking tools. The specifications of the devices are in Table I. We developed our benchmarking tool with Java, so we can run it in all the platforms that have a Java Virtual Machine (JVM).

TABLE I. SPECIFICATIONS OF DEVICES USED IN EXPERIMENTS

	Device		
	Phone	***Tablet***	***PC***
Brand	Samsung	Samsung	Hewlett Packard
Model	Nexus i9250	Tab 8.9	HP xw4600
Processor	Dual-Core 1.2 GHz	Dual-Core 1.0 GHz	Core 2 Duo 2.6 GHz
RAM	1 GB	1 GB	4 GB
OS	Android v4.3	Android v3.0	Windows 7 Pro

A. *Results for the Response Time*

Fig. 3 depicts the response time obtained with the Nexus i9250 phone. We have three curves representing the response time for GetRequest, GetNextRequest, and SetRequest messages.

We varied the number of OIDs in the requests from 1 to 10. As expected, the response time of the GetRequest is the smallest, since we only fetch the value of the OIDs in the Android device. GetNextRequest messages first search for the next OID and then fetch its value. SetRequest messages modify the value of the OID, and modifying is usually longer than fetching.

The SetRequest experiments have a better performance than the GetNextRequest when the SNMP messages have a small number of OIDs, and the behavior is inverted for higher number of OIDs. Also note that all the three curves are increasing with the number of OIDs as expected, with SetRequest increasing with the fastest rate.

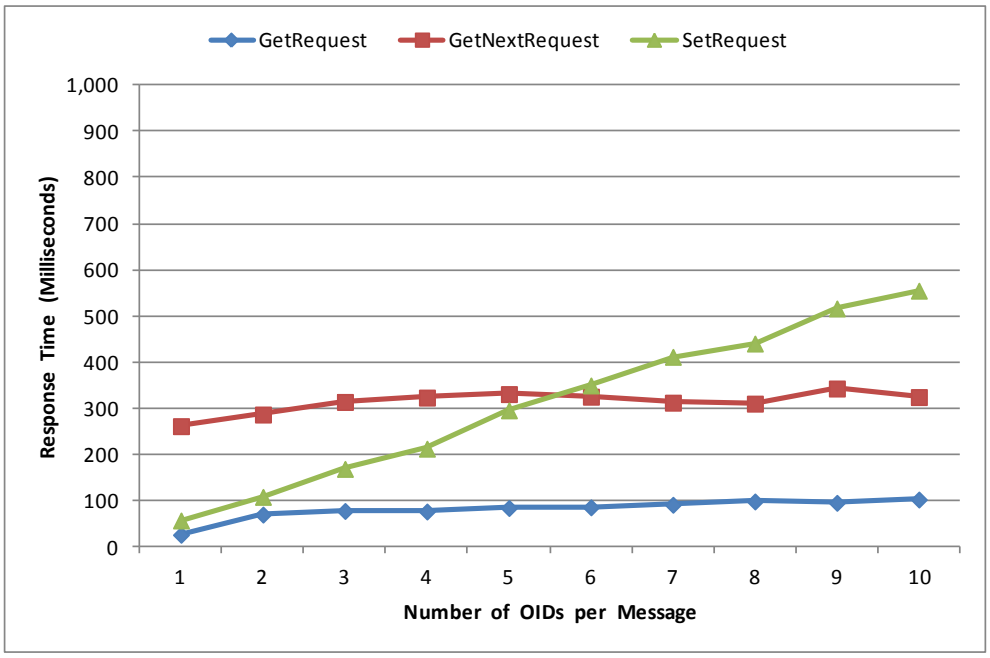

Fig. 3. Response Time for Different SNMP Messages in the Nexus i9250 Phone

Fig. 4 shows the response time obtained with the Tab 8.9 tablet. We have three curves representing the response time for GetRequest, GetNextRequest, and SetRequest messages. We varied the number of OIDs in the requests from 1 to 10. The results obtained are similar to the one of the Nexus i9250 phone (see Fig. 3). However, the response time is a little

smallest (better in this case) for the phone since it has a better processor than the tablet.

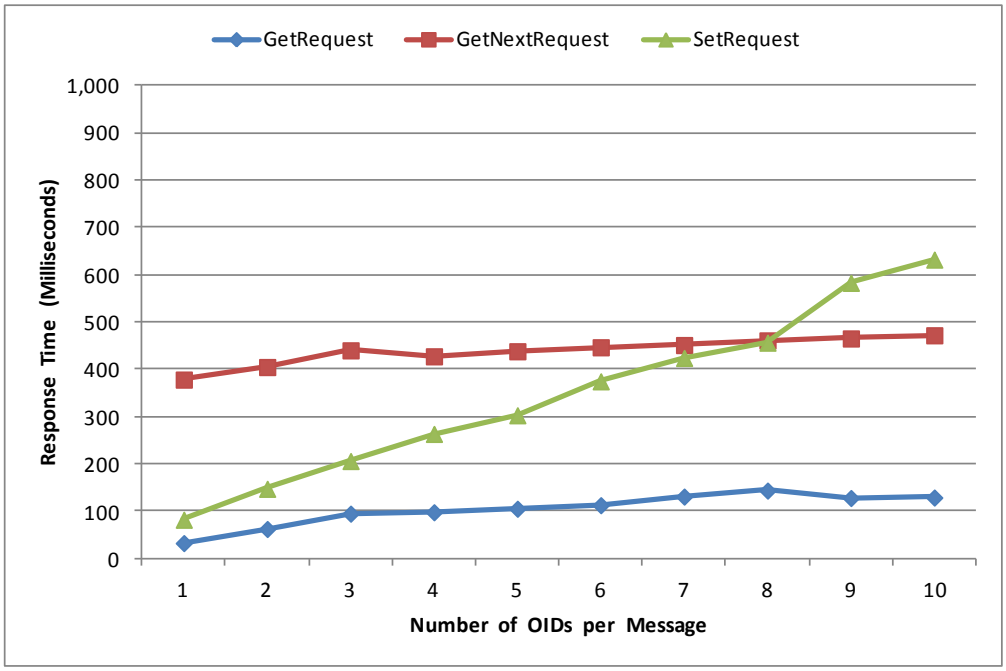

Fig. 4. Response Time for Different SNMP Messages in the Tab 8.9 Tablet

Results for the Reply Request Ratio

Fig. 5, Fig. 6, and Fig. 7 show the results obtained for the RRR, for GetRequest messages with 1, 3, and 6 OIDs, respectively. We varied the number of GetRequest messages sent by second according to the following values: 1, 2, 4, 6, 8, 10, 12, 14, and 16.

For each value of the number of GetRequest messages sent by second, we have six bars. The first two bars represent the RRR for the Nexus i9250 phone and the Tab 8.9 tablet, respectively, with one PC sending GetRequest messages. The following two bars (third and fourth) represent the RRR for the Nexus i9250 phone and the Tab 8.9 tablet, respectively, with two PCs sending GetRequest messages at the same time. In other words, the Android device receives twice the number of requests in one second, from two different sources. The last two bars (fifth and sixth) represent the RRR for the Nexus i9250 phone and the Tab 8.9 tablet, respectively, with three PCs sending GetRequest messages at the same time. In other words, the Android device receives three times the number of requests in one second, from three different sources. We can observe from these figures than the Nexus i9250 phone has a better RRR than the Tab 8.9 tablet, as expected, since it has a better processor. The idea of these experiments is to stress the Android devices and to see how much SNMP GetRequest traffic can be handled by our agent before getting saturated.

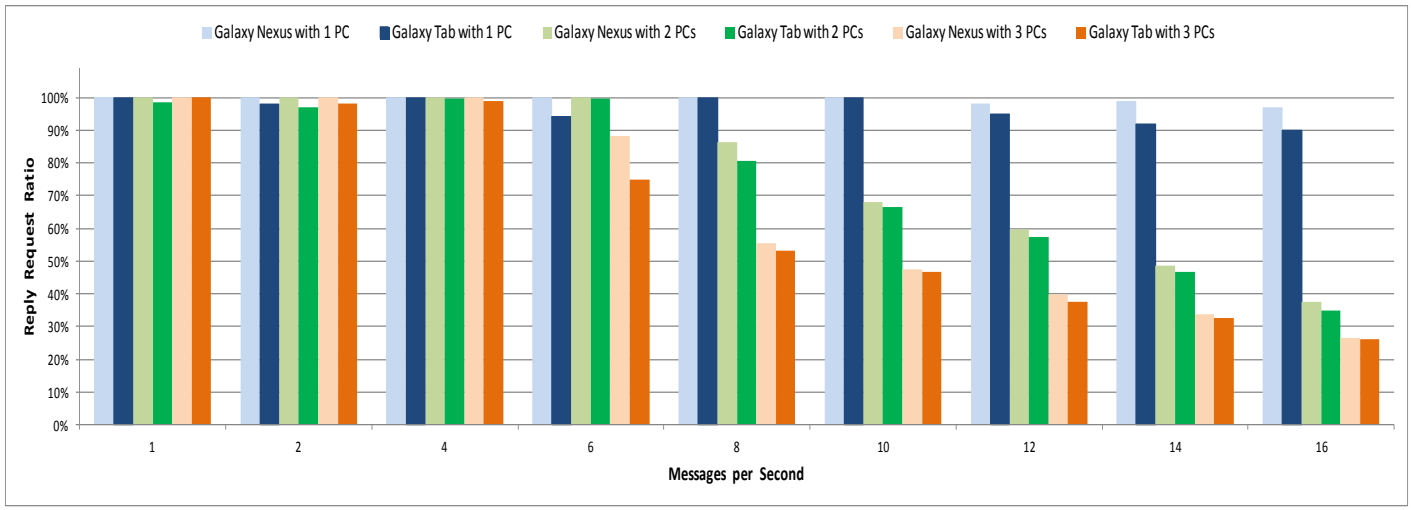

Fig. 5. RRR for GetRequest Messages with 1 OID

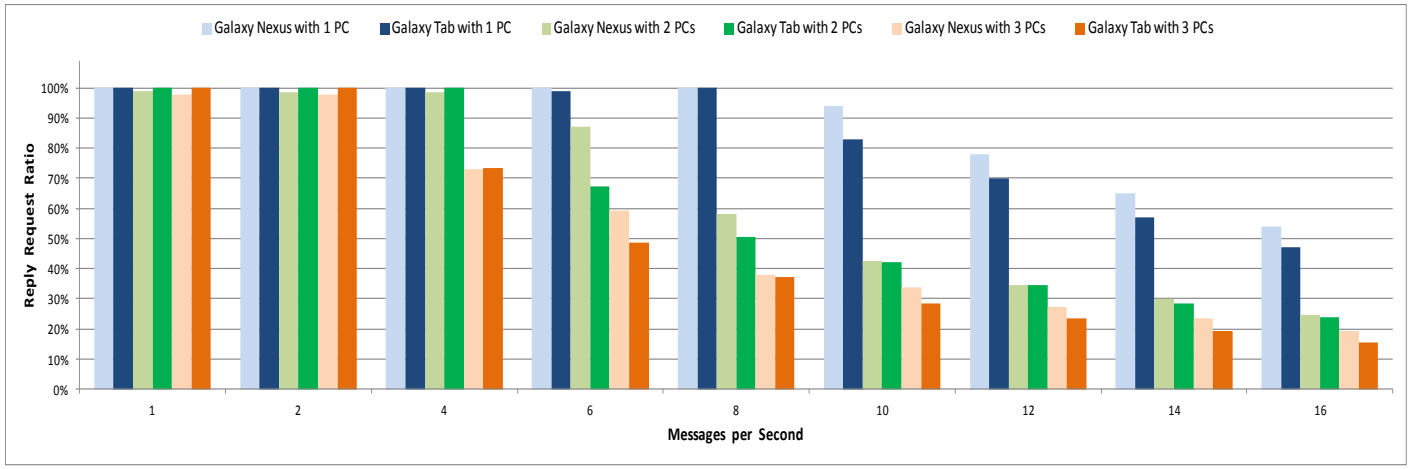

Fig. 6. RRR for GetRequest Messages with 3 OIDs

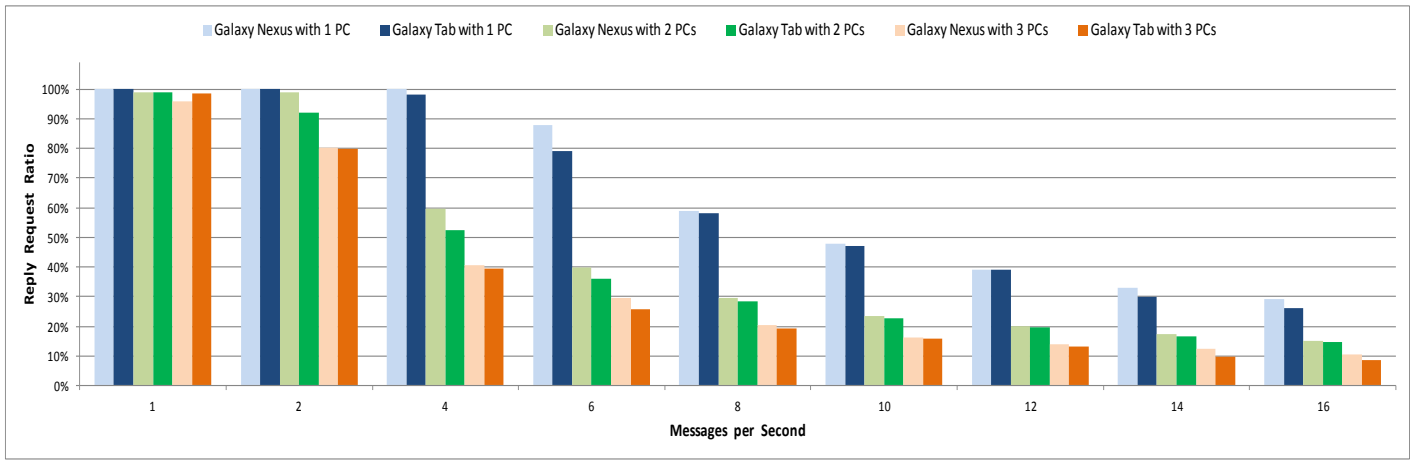

Fig. 7. RRR for GetRequest Messages with 6 OIDs

Fig. 8, Fig. 9, and Fig. 10 show the results obtained for the RRR, for GetNextRequest messages with 1, 3, and 6 OIDs, respectively. We varied the number of GetNextRequest messages sent by second according to the following values: 1, 2, 4, 6, 8, 10, 12, 14, and 16. Similarly to the GetRequest experiments, we can observe from these figures than the Nexus i9250 phone has a better RRR than the Tab 8.9 tablet, as expected, since it has a better processor. That is, the Nexus i9250 phone can handle a bigger number of SNMP requests before been saturated.

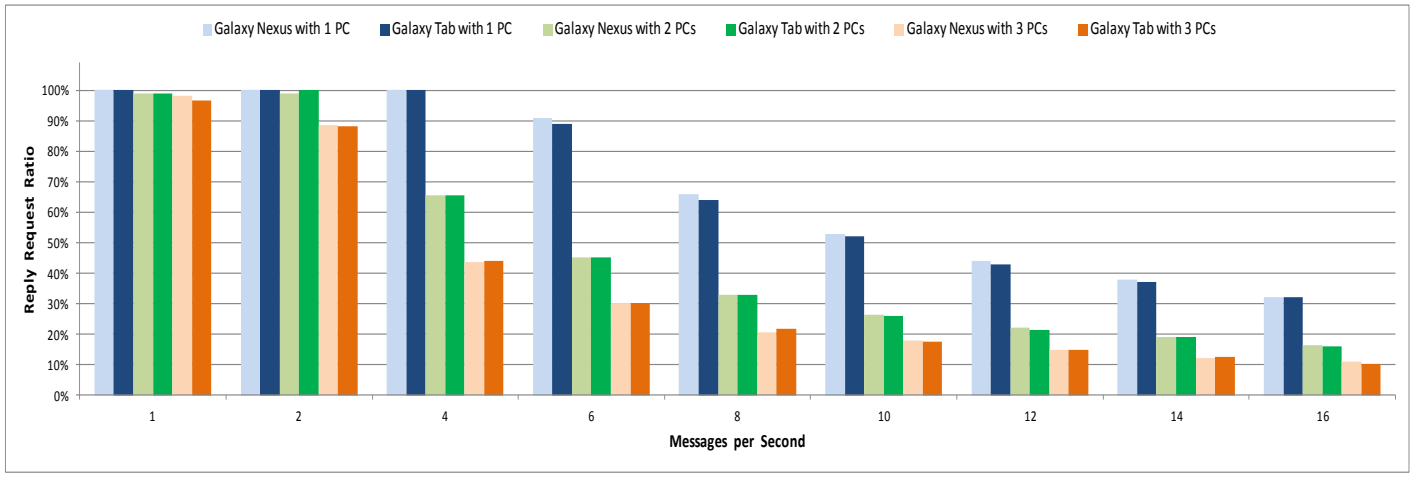

Fig. 8. RRR for GetNextRequest Messages with 1 OID

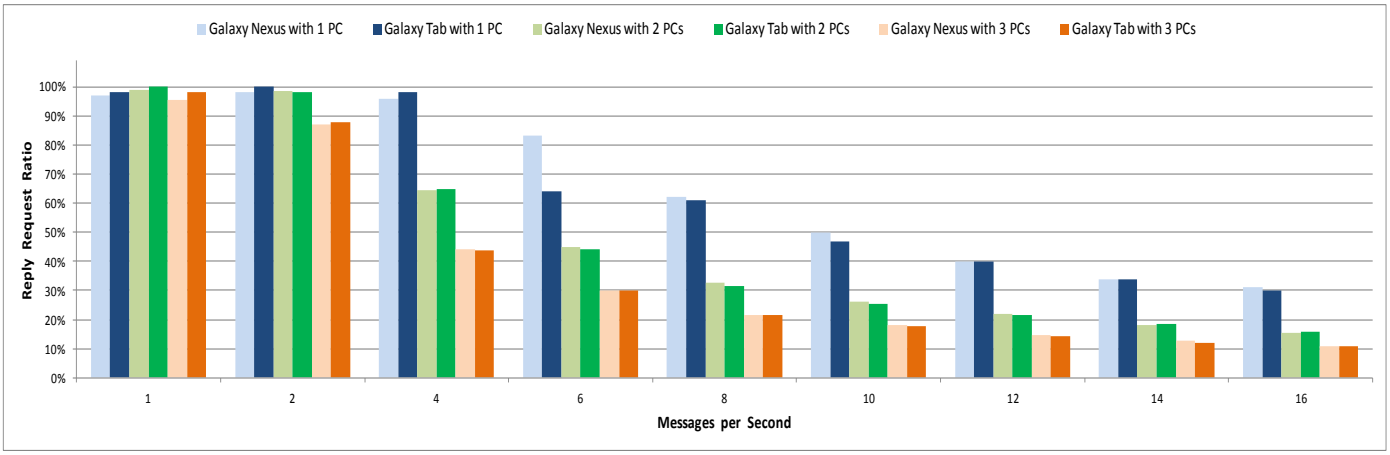

Fig. 9. RRR for GetNextRequest Messages with 3 OIDs

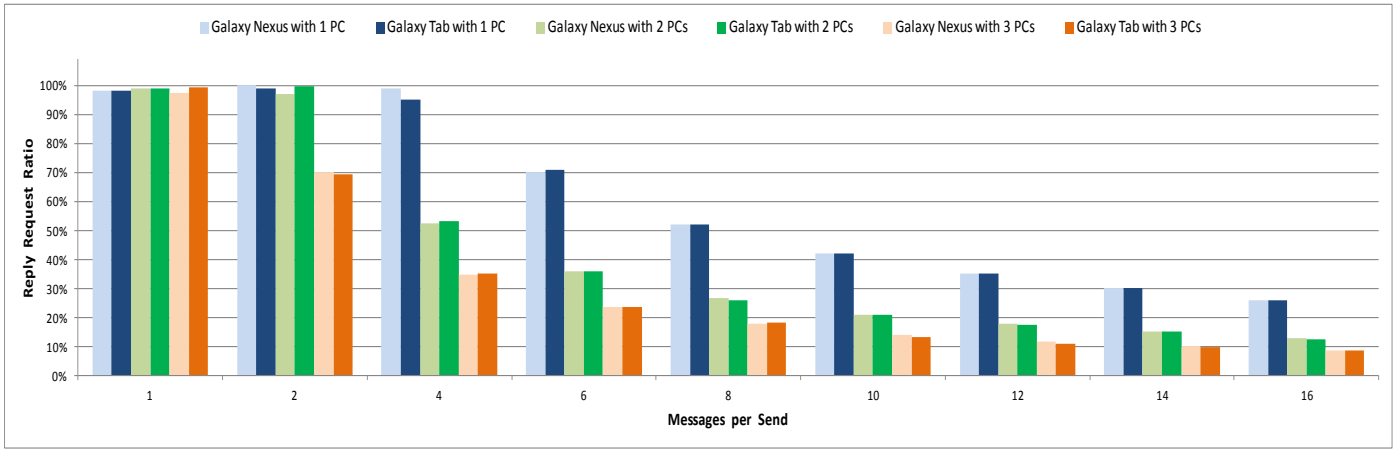

Fig. 10. RRR for GetNextRequest Messages with 6 OIDs

Fig. 11 shows the result of the RRR when varying the number of SNMP requests sent by second to the Nexus i9250 phone from 3 PCs, by using the following values: 1, 2, 4. 6, 8, 10, 12, 14, and 16 requests per seconds. We have 3 curves for GetRequest, GetNextRequest, and SetRequest messages. As expected, GetRequest has the best performance of the requests. GetNextRequest and SetRequest are showing similar results.

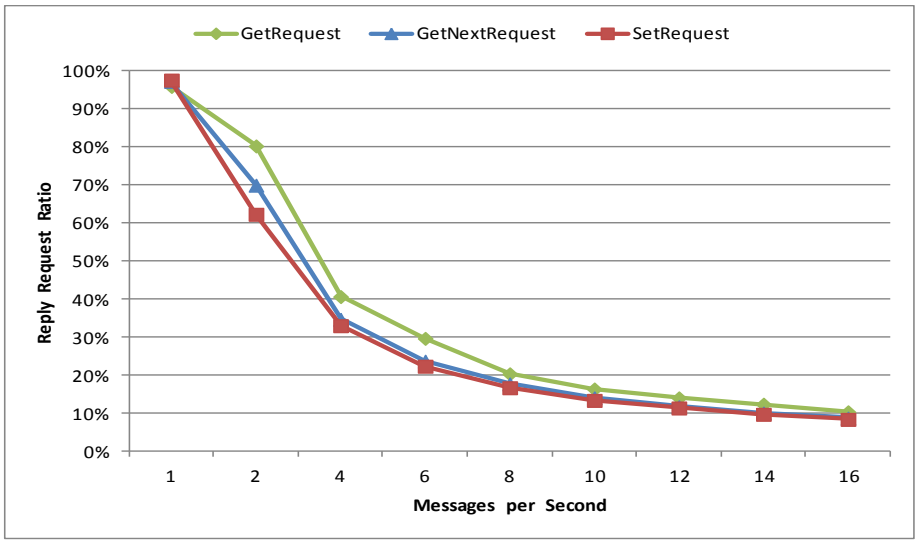

Fig. 11. RRR for GetRequest, GetNextRequest, and SetRequest Messages Sent to the Nexus i9250 Phone with 3PCs and 6 OIDs

Fig. 12 shows the results of the RRR when varying the number of SNMP requests sent by second to the Tab 8.9 tablet from 3 PCs, by using the following values: 1, 2, 4. 6, 8, 10, 12, 14, and 16 requests per seconds. We have 3 curves for GetRequest, GetNextRequest, and SetRequest messages. The results are similar to the ones obtained for the Nexus i9250 phone (see Fig. 11). However, the RRR is better in the case of the Nexus i9250 phone, due to the better processor. These results also show that common Android devices can manage a high volume of SNMP request in a short period of time.

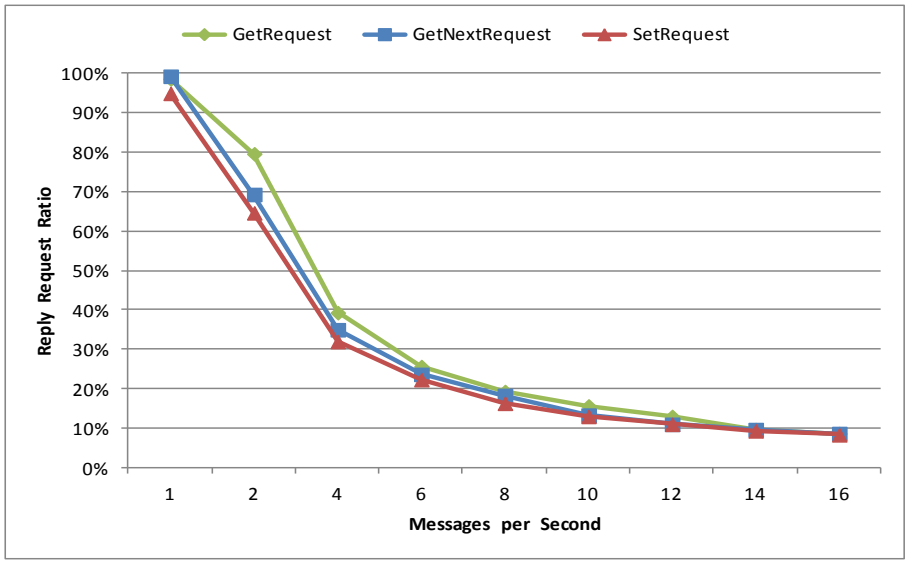

Fig. 12. RRR for GetRequest, GetNextRequest, and SetRequest Messages Sent to the Tab 8.9 Tablet with 3PCs and 6 OIDs

VII. Conclusions and Future Work

Million of Android devices have been sold all over the world, making Android the favorite mobile OS. Due to the growing interest in these devices, changes in network administration to integrate them in NMSs are required. Since SNMP is wildly spread in NMSs, an SNMP agent for Android is the first step for the inclusion of Android devices in monitoring systems. In this paper, we presented the first implementation of an SNMP agent for Android. Our agent has support for SNMPv1 and SNMPv2c. With our agent, users can retrieve almost all the OIDs of MIB-I, MIB-II, and Host Resources MIB.

To validate the agent and to show that the actual Android mobile devices have enough power to be integrated in a management system, we also proposed a performance benchmarking tool for SNMP. Our tests showed that common Android devices (Nexus i9250 phone and Tab 8.9 tablet) can be integrated in NMSs, since they can handle a high number of SNMP request in one second.

As future work, we plan to extend our agent to support SNMPv3 [9]. Today, authentication and privacy are important due to the numerous security threads in networks, especially in wireless networks where the radio waves spread in all directions.

Acknowledgment

We want to thank the CDCH-UCV (Consejo de Desarrollo Científico y Humanístico) which partially supported this research under grant number: PG 03-8066-2011/1.

References

[1] J. Case, M. Fedor, M. Schoffstall, and J. Davin, A Simple Network Management Protocol (SNMP), RFC 1157, May 1990.

[2] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, Introduction to Community-based SNMPv2, RFC 1901, January 1996.

[3] M. Wilcox, Porting to the Symbian Platform: Open Mobile Development in C/C++, 1st Edition, Wiley, November 2009.

[4] A. Ludin, Learn BlackBerry 10 App Development: A Cascades-Driven Approach, Apress, 1st Edition, March 2014.

[5] J. Conway, A. Hillegass, C. Keur, iOS Programming: The Big Nerd Ranch Guide, 4th Edition, Big Nerd Ranch Guides, February 2014.

[6] A. Whitechapel and S. McKenna, Windows Phone 8 Development Internals, Microsoft Press, July 2013.

[7] R. Meier, Professional Android 4 Application Development, Wrox, 3rd Edition, May 2012.

[8] P. Deitel, H. Deitel, and A. Deitel, Android for Programmers: An App-Driven Approach, Prentice Hall, 2nd Edition, January 2014

[9] W. Stallings, SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Addison-Wesley Professional, 3rd Edition, October 2013.

[10] J. Gateau, Extending Simple Network Management Protocol (SNMP) Beyond Network Management: A MIB Architecture for Network-Centric Service, Naval Postgraduate School, Monterey, California, March 2007.

[11] L. Walsh, SNMP MIB Handbook, Wyndham Press, March 2008.

[12] K. McCloghrie and M. Rose, Management Information Base for Network Management of TCP/IP-Based Internets, RFC 1156, May 1990.

[13] K. McCloghrie and M. Rose, Management Information Base for Network Management of TCP/IP-based Internets: MIB-II, RFC 1213, March 1991.

[14] S. Waldbusser and P. Grillo, Host Resources MIB, RFC 2790, March 2000.

[15] G. Ayala, P. Poskal, and E. Gamess, SNMP JManager: An Open Source Didactic Application for Teaching and Learning SNMP v1/2c/3 with Support for IPv4 and IPv6, In proceedings of the seventh Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2009), San Cristóbal, Venezuela, June 2009.

[16] L. Chappell and G. Combs, Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide, 2nd Edition, Laura Chappell University, March 2012.