Obtaining Reliable Bit Rate Measurements in SNMP-Managed Networks

Patrik Carlsson, Markus Fiedler, Kurt Tutschku*, Stefan Chevul and Arne A. Nilsson
Dept. of Telecommunications and Signal Processing, Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden. Phone: (+46) 455-385000, fax: (+46) 455-385667
email: {Patrik.Carlsson,Markus.Fiedler,Stefan.Chevul,Arne.Nilsson}@bth.se
*Institute of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany. Phone: (+49) 931-8886641, fax: (+49) 931-8886632
email: tutschku@informatik.uni-wuerzburg.de

Abstract

The Simple Network Management Protocol, SNMP, is the most widespread standard for Internet management. As SNMP stacks are available on most equipment, this protocol has to be considered when it comes to performance management, traffic engineering and network control. However, especially when using the predominant version 1, SNMPv1, special care has to be taken to avoid erroneous results when calculating bit rates. In this work, we evalute six off-the-shelf network components. We demonstrate that bit rate measurements can be completely misleading if the sample intervals that are used are either too large or too small. We present solutions and workarounds for these problems. The devices are evaluated with regards to their updating and response behavior.

Keywords: SNMP, bit rate measurement, MIB, polling, sampling, SNMP agents, response times

1 Introduction

The original Internet was designed as a fault-tolerant network for best effort data transmission. Its capability to support almost any application and the competition in operating efficient networks displaced this relaxation. The future Internet architecture will support Quality-of-Service (QoS) objectives like high throughput, small packet delay, or low delay variation. From the view of network operations, the demand for QoS calls for a high quality performance management architecture. The architecture has to provide reliable and up-to-date network performance information.

The management protocol of today's Internet is SNMP (*Simple Network Management Protocol*) [1, 2, 3]. SNMP applies the manager/agent concept. The manager issues control actions to the agent. The agent processes these commands and returns any results. The information is exchanged via special SNMP Protocol Data Units (SNMP PDUs). The SNMP architecture uses an information model denoted as the Management Information Base II (MIB-II) [4]. This information base specifies syntax and semantics of the stored data. The variables in the information base are denoted as objects. The data types of the MIB-II objects comprise counters, addresses, and strings. Even though the first versions of SNMP and MIB-II have been evolved to more sophisticated variants like SNMPv2 [5], SNMPv3 [6], and various MIB extensions, e.g. [7], SNMPv1 and MIB-II are still being used. Unfortunately, these specifications are still the smallest common denominator when it comes to Internet management. In addition, it is very unlikely that most operators in today's IP-based networks can exchange their equipment quickly. Furthermore, they won't be able to install a large number of special-purpose systems for performance measurements. The existing equipment will still be used for a long time for this task. This paper discusses on the task of obtaining reliable performance measurements from standard, already deployed network equipment.

The system time on SNMP agents is stored in the MIB-II object *sysUpTime* and counted in ticks of tens of milliseconds. The manager can query agents in almost arbitrary intervals. Because of these features, SNMP seems to be well suited for generating input for traffic engineering. The SNMP protocol operates on the typical time scale of network control or an even smaller time scale. In addition, SNMP traffic measurements are non-intrusive, since measurements can be started without service interruption.

An alternative to SNMP based measurements is the use of packet monitoring systems like `tcpdump` [8]. The `tcpdump` software records IP packet streams for off-line analysis. This tool allows for packet stream observations and analysis on small time scales. However, when observing a fully-loaded Gigabit Ethernet link, `tcpdump` produces about 17 MB data per second. The real-time processing of such huge amounts of data is not feasible in operational network environments.

On this background, we evaluate the performance of SNMP agents on six commercially available switches and routers. Our goal is to derive guidelines on how to obtain accurate, reliable, on-line bit rate measurements from these network elements. We identify the sources of errors and unreliability for traffic measurements. In addition, we investigate the updating behavior of the elements and we evaluate their response times for SNMP requests.

The remainder of the paper is organized as follows. Section 2 presents the tool used for SNMP sampling

and some basic formulae for obtaining bit rate measurements. Section 3 discusses the problem of wrapping counters, and Section 4 deals with MIB object update behavior. Section 5 contains a comparison of performance of SNMP agents on six different devices. Section 6 presents the conclusions.

2 SNMP-Based Bit Rate- and Response Time Measurements

SNMP protocol data units (PDUs) are carried via the User Datagram Protocol (UDP) on top of IP. A typical SNMP GetRequest-PDU contains a list of variable identifiers to be polled, while the GetResponse-PDU delivers the identifiers and their values. Especially for performance measurements, the time at which a variable is queried is of outmost importance. Thus, typical requests address traffic counters and the corresponding time stamp.

2.1 Measuring Bit Rates

In order to obtain the bit rate on a link, we monitor the traffic flowing into or out of the interface the link is connected to. The following MIB-II objects are of interest:

- mib-2.interfaces.ifTable.ifEntry.ifInOctets = "The total number of octets received on the interface, including framing characters" [4], henceforth denoted by O_{In};
- mib-2.interfaces.ifTable.ifEntry.ifOutOctets = "The total number of octets transmitted out of the interface, including framing characters" [4], henceforth denoted by O_{Out};
- mib-2.system.sysUpTime = "The time (in hundredths of a second) since the network management portion of the system was last re-initialized" [4], henceforth denoted by T.

The i^{th} response of an agent delivers the tuple $(T(i), O_{\text{In}}(i))$ or $(T(i), O_{\text{Out}}(i))$, depending on which direction is observed on the interface. These samples are stored in the vectors $\vec{T}$, $\vec{O}_{\text{In}}$ and $\vec{O}_{\text{Out}}$, respectively. The bit rate is calculated as

$$\frac{B(i)}{\text{bps}} = \begin{cases} 800\,\frac{O(i)-O(i-1)}{T(i)-T(i-1)} & \text{if } O(i-1) \leq O(i) \\ 800\,\frac{O(i)+(2^{32}-O(i-1))}{T(i)-T(i-1)} & \text{else} \end{cases} \quad (1)$$

The equation has two main problems in real networks. First, it requires that the sysUpTime on the agent is correctly updated every 10^{th} ms. If two consecutive samples have the same time stamp, a division by zero will occur. The next problem is related to the fact that those counters are 32-bit counters in the MIB-II specification [4]. The largest number a 32-bit counter can store before wrapping around is $2^{32} - 1 = 4294967295$. For sysUpTime, this value is equivalent to 497 days. This is usually non-critical, as most devices are normally re-initialized at least once a year. In the case of octet counters, the counter range of 2^{32} equals merely 4 GB. This will cause severe problems in measuring traffic on fast links. A more detailed discussion on this problem of wrapping counters is presented in Section 3.

2.2 Responsiveness of SNMP Agents

The responsiveness is measured as the time it takes from issuing an SNMP GetRequest-PDU until the corresponding GetResponse-PDU is received. This time includes both network transfer time and the processing time in the SNMP agent. Using the notation in Figure 1, the response time is defined as

$$ResponseTime = RespTime - ReqTime \quad (2)$$

The time stamps used here are based on the internal clock of the monitoring station.

2.3 Software Overview

In order to obtain accurate measurements, we implemented a simple measurement application. A flow chart of the software is shown in Figure 1. First of all the internal variables are initialized. The measurements are done in the following way:

1. Get the current time as accurate as possible and store it in the variable $ReqTime$.
2. Send a blocking SNMP GetRequest-PDU to the agent with the requested variables. Blocking means that the software will pause until the GetResponse-PDU arrives from the agent.
3. Get the current time and store it as $RespTime$.
4. Log the response from the agent and the time stamps to a file.
5. From the second sample onward, calculate the current bit rate.
6. Update all variables.
7. Put the software on hold for so long that the next request will be sent $Interval$ seconds after the preceding one. Observe that $Interval$ does not need to be an integer and can be almost arbitrarily small. If the measurement took longer time than $Interval$, then do no sleep.

Once the desired number of measurements has been carried out, close the log files, clean up and exit.

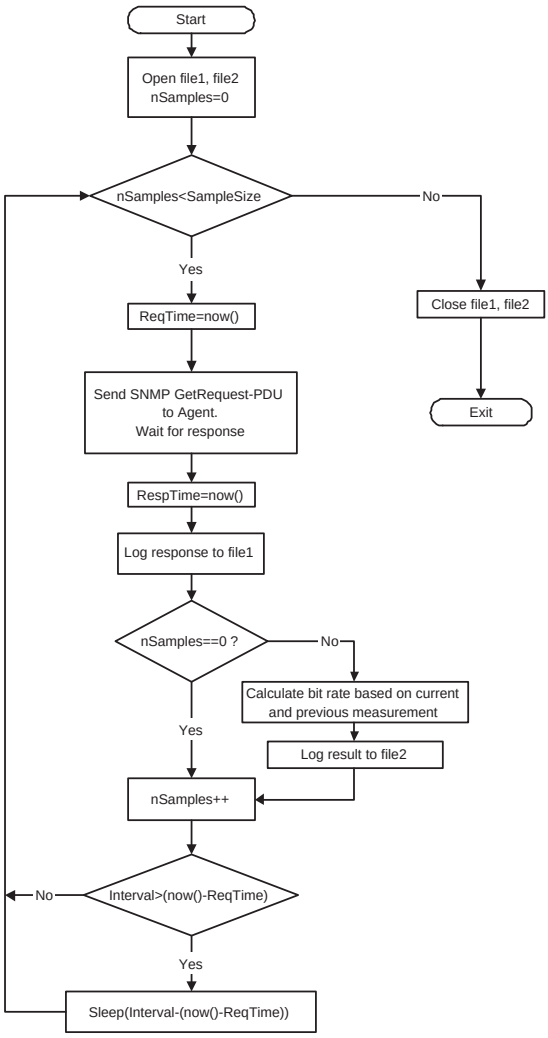

Figure 1: Software flow chart.

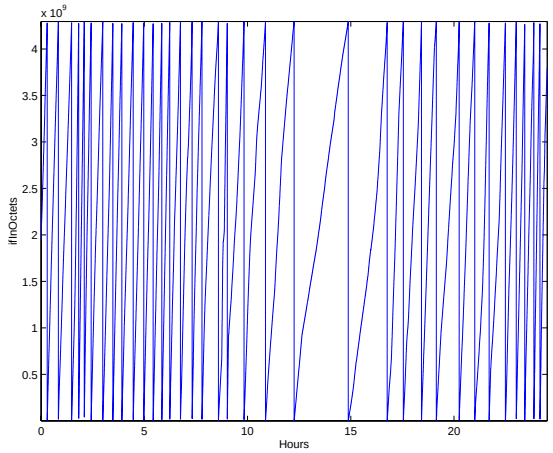

Figure 2: Example of a 32-bit counter wrapping around.

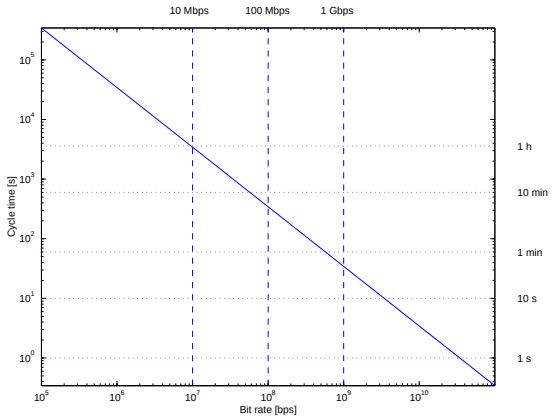

Figure 3: CCT related to the bit rate of a link.

3 Wrapping Counters Problem

We initially encountered the wrapping problem on a 100 Mbps Ethernet interface in the University network. This will occur in every measurement tool that uses 32-bit counters. Figure 2 shows an example of a wrapping counter. The observed interface measured was a Gigabit Ethernet that connects two University campuses. During the measurement, which lasted 24 hours, the utilization was low. The time it takes for a counter to loop is related to both the load and the link capacity of the interface, and in this case the interface had a capacity of 1 Gbps.

Let C denote the link capacity, given in bps, and ρ its current utilization. Then, the Counter Cycle Time (CCT) is given by

$$T_{\text{CCT}} = \frac{2^{32} \cdot 8 \text{ bit}}{\rho\, C} \tag{3}$$

The CCT is minimized, $T_{\text{CCT,min}}$, when the link is fully utilized ($\rho \to 1$). Since we are looking for reliable measurements, we have to account for this worst case (see measurements presented in Section 5). The diagram in Figure 3 shows the cycle time for a given bit rate. For a fully utilized 1 Gbps link the cycle time is found to be around 30 s. The counters for a link with a mean bit rate of 200 Mbps wrap every 200 seconds on average. Using Equation 1, the bit rate will be calculated correctly if and only if the sample interval T_{samp} is smaller than the minimal CCT. Otherwise, there is no way of determining how many times the counter looped in-between consecutive samples.

As an example, let us assume that we are observing a Gigabit Ethernet link with a sample interval of 300 seconds. The first GetResponse-PDU returns a value of $2^{32} - 1 = 4294967295$, and the next one holds 30. From this, it is impossible to determine whether 31 octets (counter looped once) or $n \cdot 4294967296 + 31$ octets (counter looped $n+1$ times, where $n = 1, 2, \ldots, 8$) have been transmitted in reality. So the bit rate could be 0.8 bps or $114, 229, \ldots, 916$ Mbps, respectively. But the reported bit rate would be 0.8 bps.

On a 1 Gbps link, a tool should check the variables at least every 30 seconds. Remember that SNMP runs on

top of UDP, which has no guarantee of delivery. Inturn, this means that packets can disappear. If sampling every 30 seconds, loosing one packet renders the next sample useless. A tool will then not be able to determine if or how many times the counter has looped since the last correct sample. To account for this danger, the sample interval should be reduced to

$$T_{\text{samp,max}} = \frac{T_{\text{CCT,min}}}{\gamma}, \tag{4}$$

where $\gamma \geq 1$ represents a safety factor.

64-bit Counters

The solution for the 32-bit counter problem could be to use the 64-bit counters (mib-2.ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets and .ifHCOutOctets) defined in [7]. This requires the use of SNMPv2 or higher. If this is not available in the device, then the sample frequency simply has to be high enough to detect possible wrappings.

Unfortunately, in many devices, only SNMPv1 agents are available. Despite the fact that this version is already viewed as historic at the time of writing [9], it is still predominant.

4 MIB Update Rate Problem

Another problem we discovered was the slow update frequency of the monitored SNMP variables. Some SNMP agents seem to link internal, always up-to-date octet counters to MIB variables, while others seem to copy counter values into MIB variables based on regular time intervals. One would expect this to happen exactly when sysUpTime is updated. However, we observed that this is generally not the case. On one large switch-router, the update rate of the MIB variables was as slow as once every 10^{th} second. Such a behavior is shown in Figure 4: this graph depicts a magnified part of Figure 2. Please notice the steps, which equal the octet flow over a 10 s period. Since the load was low in this case, the steps are only about 25 MB, as opposed to 1250 MB on a fully utilized link, remember that this value corresponds to 10 s of data flow. Thus, a slow update rate of MIB variables can again cause the measurement to be unreliable.

If the sample interval is smaller than the update interval, the same octet counter value may be sampled several times. Even though this may be of advantage given the possibility of loosing SNMP-PDUs, it will cause Equation 1 to produce erroneous results.

The delayed update of the variables can also cause misleading results. If the update rate is slow, as displayed in Figure 4, once a non-zero measurement is done, the bit rate will be larger than it should be. For instance, if a link is loaded at 80 Mbps, the sample frequency is 1 Hz and the update rate is 10 s, then nine out of ten samples will be zero, and the tenth will indicate a bit rate of 800 Mbps.

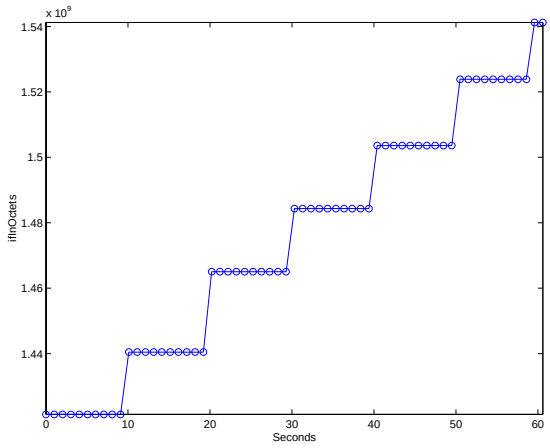

Figure 4: Example of MIB update behavior.

Table 1: Available sample intervals for different safety factors γ on a 1 Gbps link given an update interval of 10 s.

γ	T_{update} [s]	$T_{\text{samp,max}}$ [s]
1	10	34
2	10	17
3	10	11

On a 100 Mbps link, the problem would be discovered. But on a 1 Gbps link, this would perhaps cause costly and unjustified link upgrades. An example of such a behavior is shown in Figure 10. The update interval, T_{update}, defines a minimal sample interval. Hence, the update interval should be determined, which will be done for some switches and routers in Section 5.

In order to achieve correct results, the sample interval has to be chosen according to the following sampling theorem:

$$T_{\text{samp}} \in [T_{\text{update}}, T_{\text{samp,max}}] \tag{5}$$

As indicated by Table 1, this sample interval may have to be chosen very carefully.

Solutions

The update interval needs to be minimized and linked to the sysUpTime variable. This is up to manufacturers to implement.

But even corrupted measurements with $T_{\text{samp}} < T_{\text{update}}$ are feasible for bit rate calculations by averaging. Denote $j = \lceil T_{\text{update}}/T_{\text{samp}} \rceil$. Then the observed bit rate

$$\frac{B(i)}{\text{bps}} = \begin{cases} 800\,\frac{O(i)-O(i-j+1)}{T(i)-T(i-j+1)} & \text{if } O(i-j+1) \leq O(i) \\ 800\,\frac{O(i)+(2^{32}-O(i-j+1))}{T(i)-T(i-j+1)} & \text{else} \end{cases}$$

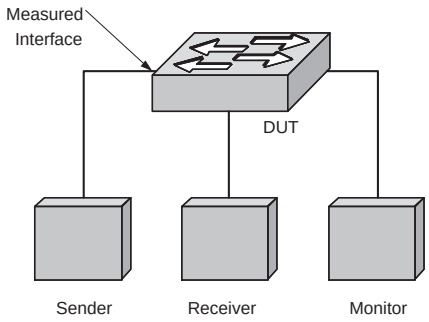

Figure 5: Setup used during switch tests.

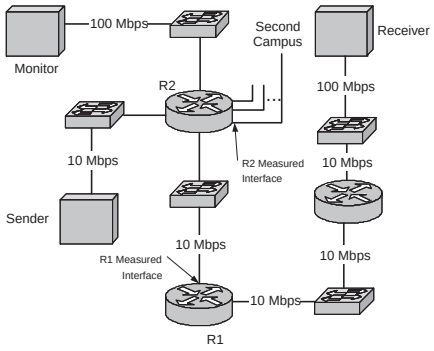

Figure 6: Setup used during router tests.

(6)

can be used either in a moving window $(i = j, j+1, \dots)$ or a jumping window $(i = j, 2j, \dots)$ fashion to correctly calculate the values. In any case, due to the high update interval, the resolution in time is decreased. This means that the observed bit rates are smoothed(low-pass filterd), turning the measurements to be less representative.

5 Device Performance

In this section we show the results of bit rate measurements and response time measurements that have been done on four switches and two routers. In the switch cases, the tests were performed in a closed environment. The setup is displayed in Figure 5. On the device under test (DUT), sender, receiver and monitor are connected to different interfaces. We monitored the ifInOctets counter on the interface that the sender was connected to.

The test was done by sending 15.3 GB of data, which was arbitrarily split into 39 files each $420\,000\,000$ Bytes large, using standard FTP. This is more than the 32-bit counter can handle before wrapping around. The tests on the routers were run while the routers remained in their normal environment. We could not do the same closed environment tests as with the switched because both routers are essential for normal operations of the University network. Figure 6 depicts the setup that was used for the routers. R1 connects a small laboratory, and R2 connects two of the University's campuses. It is well known that the load on any given interface is the sum of all streams passing through it. For router R1 this load was low on the measured interface (ifInOctets counter), causing a long counter wrapping time, so an SFTP session was added to increase the load. In the R2 case this was not necessary, since there already was a significant load on the measured interface. The session originated from "Sender" and terminated in "Receiver". It is also necessary to mention that in the R1 case, at least one of the links that connects the "Sender" to the "Receiver" was a 10 Mbps half duplex link.

The measurement was run for 3600 samples, with

Table 2: Devices that were tested.

Device ID	Number of interfaces	Speed [Mbps]	Ethernet Type
S1	24	10/100	FastEthernet
S2	24	10/100	FastEthernet
S3	24	10/100	FastEthernet
S4	8	10/100	FastEthernet
R1	2	10	Ethernet
R2	18	1000	Gigabit Ethernet

$Interval = 1$ s. Each sample measures the ifInOctet counter for the specific interface that is measured. The monitor was started a couple of seconds prior to the FTP/SFTP transfer. If possible, the DUT was reset before the test, just to get the counters to start from zero. The devices that were tested are listed in Table 2. We look at bit rates and response time statistics for the SNMP queries. A detailed look on the bit rate over a small interval (61 samples) is used for an estimation of the devices' MIB update rates.

5.1 Measured Bit Rates

Time plots of the observed bit rates are shown in Figures 7 to 12. The bit rates were obtained using Equation 1 to allow the identification of erroneous behavior. Also notice that the x-axis holds the sysUpTime variable.

To compare the bit rates obtained from our software, we also determine bit rates on application level, which are smaller than those on the link level. Table 3 presents the results that were calculated as the total number of bits transferred divided by the transfer time. The transfer time is the time between two time stamps, one prior to the FTP call and the other one after the FTP command returned. In the SFTP case, the tool did not provide this possibility, but indicated itself speeds between 400 and 500 KB/s.

The bit rates obtained for S1 (Figure 7) are obviously correct. During activity, the peak values are slightly more than 90 Mbps, but less than 100 Mbps. S2, S3 and S4 (Figures 8 to 10) all indicate a peak bit rate larger than

Table 3: Estimated bit rate from FTP/SFTP tools.

Device	Transfer protocol	Bytes transferred [GB]	Transfer time [s]	Estimated bit rate [Mbps]
S1	FTP	15.3	1894	69.2
S2	FTP	15.3	1914	68.5
S3	FTP	15.3	1899	69.0
S4	FTP	15.3	1930	67.9
R1	SFTP	0.4	N/A	3.7

the link capacity: for S2, it was 105.5 Mbps, for S3, it exceeded 200 Mbps, and finally for S4, it surpassed 140 Mbps.

The abnormal peak of 105.5 Mbps shown by S2 seems to be an irregularity that most probably stems from a lack of synchronization between sysUpTime and the octet counter. This is an internal problem of this SNMP agent.

Besides this, Figures 7 (S1) and 8 (S2) show the highest density of bit rates in the upper part, i.e. around the mean bit rate of approximately 70 Mbps. Turning to Figure 9 (S3), we notice a high density of bit rates in the lower part. The peak bit rate on S3, however, is almost twice as high as expected, which indicates a slow update rate of the MIB variables, approximately 2 s. The latter problem is also visible on S4; a first guess of the update interval could be 1.5 s.

Looking at R1 (Figure 11), we notice an average bit rate of about 4 Mbps on its 10 Mbps interface. This is quite a low value and is probably due to the fact that at least one of the links was operated in half-duplex mode. However the behavior of SFTP also has to be taken into account.

The link in the R2 case is a 1 Gbps link that connects two campuses. The measurement was taken during the afternoon when the load was relatively low, but large enough to not require an additional load. It displays a similar behavior as S3 and S4. Not seen in this figure is that the obtained bit rates are a 10 times larger than they should be. This will be clearly visible in Figure 18 where a detailed view of R2's bit rate is shown. However, the values in Figure 12 seem to be reasonable at first sight, as they are lower than the link capacity.

5.2 MIB Update Rate

The update rate of the MIB is estimated based on detailed bit rate images from the devices, shown in Figure 13 to 18.

There are two reasons that could account for a bit rate measurement being zero.

1. There was actually no data passing through during this interval, which is very unlikely in view of ongoing FTP/SFTP sessions (S1–S4 and R1) or background traffic (R2). Even though there may be

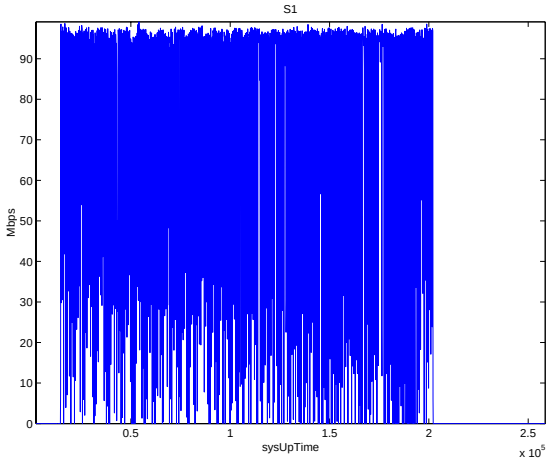

Figure 7: S1 bit rate.

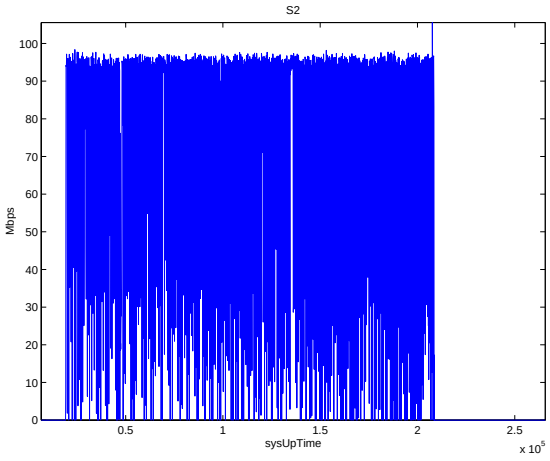

Figure 8: S2 bit rate.

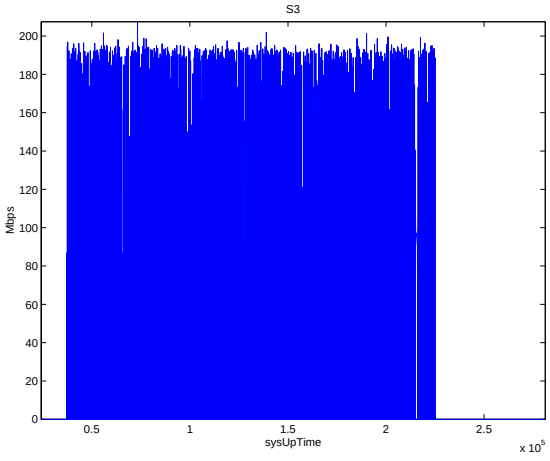

Figure 9: S3 bit rate.

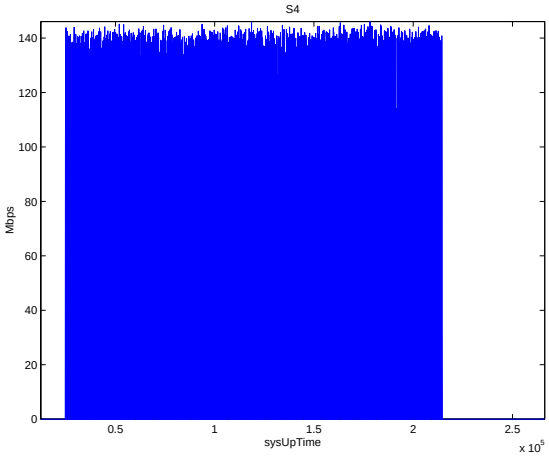

Figure 10: S4 bit rate.

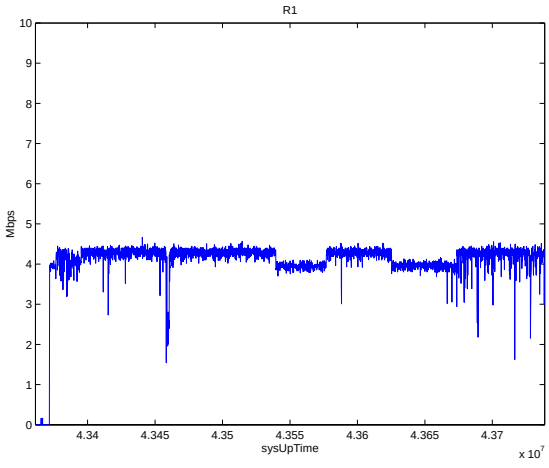

Figure 11: R1 bit rate.

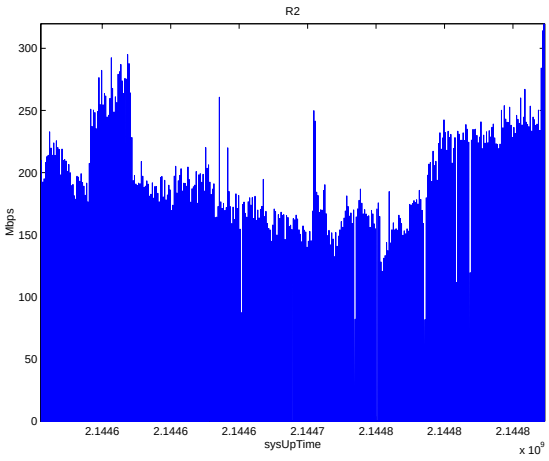

Figure 12: R2 bit rate.

Table 4: Bit rates from Figures 13 to 18.

Device	Minimum bit rate [Mbps]	Mean bit rate [Mbps]	Maximal bit rate [Mbps]
S1	0.0006	71.4	97.6
S2	0.001	73.6	97.4
S3	0.0	74.3	194.2
S4	0.0	71.5	145.8
R1	4.0	4.3	4.5
R2	0.0	19.1	204.6

breaks between the file transfers, they are very unlikely to last for one second.

2. The device has not updated the corresponding MIB variable in time.

At a first glance there seems to be some sort of shared behavior among the switches, the bit rate drops almost to zero. As this is not seen from the figures, Table 4 contains a list of the minimal, mean and maximal bit rates. From this we see that neither S1, S2 nor R1 have minimal bit rate of zero bps, but S3, S4 and R2 do. We also see that the mean bit rates are similar to the bit rates estimated from the FTP/SFTP tools. The maximal bit rates are a different issue: in case of S1, S2, R1 and R2 they are below the link capacity. However, for S3, the maximal estimated bit rate is about two times the link capacity, and for S4, that factor is about 1.5.

Turning focus to Figures 13 to 18, there are two types of drops in the bit rate: some that reache zero and others that do not. The latter are clearly visible in the S1 and S2 cases (Figures 13 and 14), and were also observed using `tcpdump` [8]. They originate from the sender pausing, after having received an acknowledgement that synchronizes the sent and acknowledged bytes. The receiver's advertised window is almost completely open (63712 bytes) at this point.

Since neither S1 or S2 reach zero, both switches seem to have an update rate of at least 1 Hz. Further stress tests with SNMP request intervals down to 10 ms indicated that the variables are almost up-to-date. In the S3 case, the update interval seems to be around 2 s, with the exception of a small interval when sysUpTime $\simeq 1.28 \times 10^5$. Why the switch exhibits an update rate of 1 Hz here is unexplained.

For the S4 switch, the update interval is smaller than 2 s but larger than 1 s. From Figure 16, it is seen that each third sample is zero, indicating an update interval of 1.5 s.

The two routers exhibit totally different update behaviors. R1 seem to update at least once every second, but R2 only once every 10 seconds. In our measurements, R2 obscures its update problem, since the obtained bit rates are below the link capacity (cf. Figures 12 and 18).

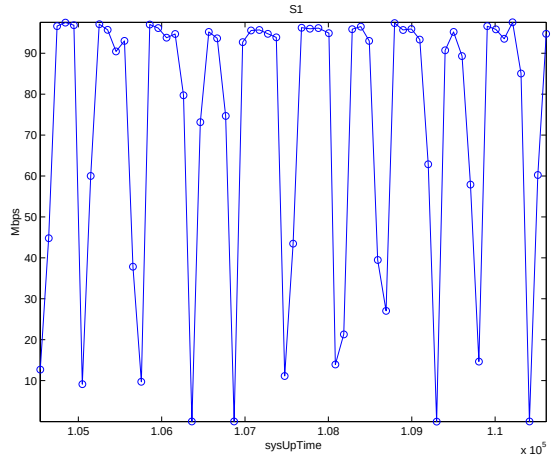

Figure 13: Bit rate for S1 over ∼60 s.

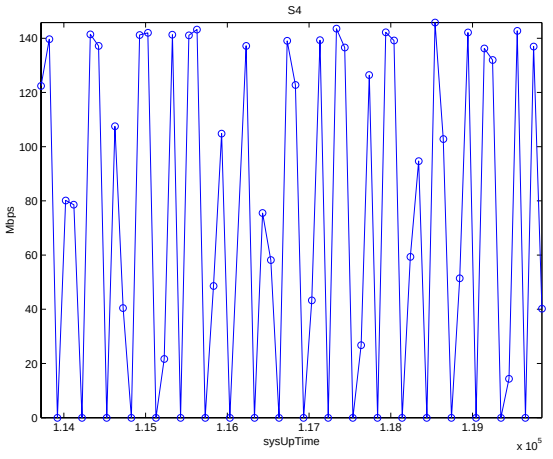

Figure 16: Bit rate for S4 over ∼60 s.

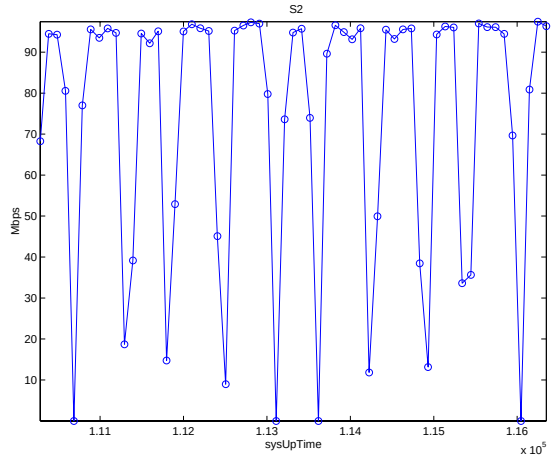

Figure 14: Bit rate for S2 over ∼60 s.

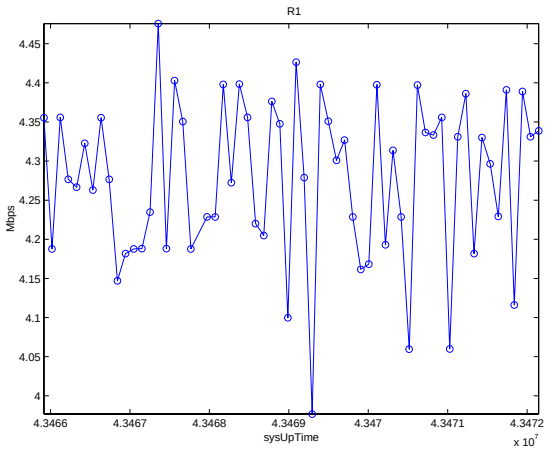

Figure 17: Bit rate for R1 over ∼60 s.

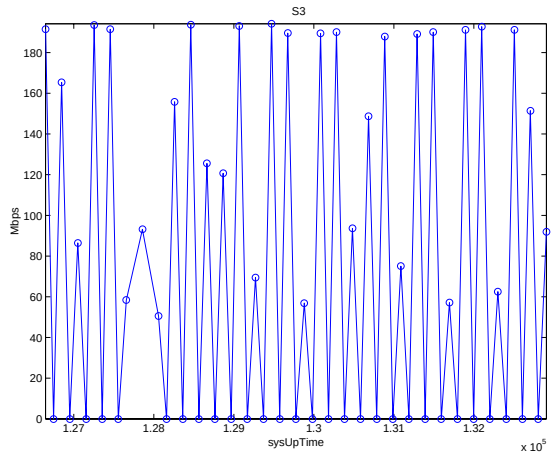

Figure 15: Bit rate for S3 over ∼60 s.

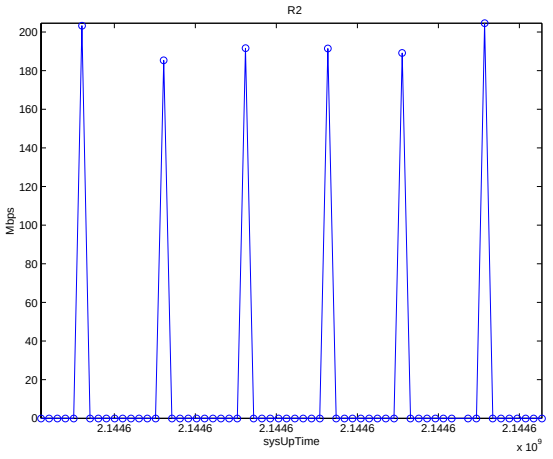

Figure 18: Bit rate for R2 over ∼60 s.

Table 5: Response times for the tested devices.

Device	Minimum response time [ms]	Mean response time [ms]	99 % quantile time [ms]	Maximal response time [ms]
S1	7.0	14.1	200.1	499.7
S2	16.9	25.8	119.7	140.0
S3	8.2	17.9	69.9	80.2
S4	7.8	9.9	10.3	20.0
R1	19.3	33.7	59.5	179.5
R2	2.5	9.7	13.1	20.0

5.3 Response Times

In Table 5, the minimal and maximal response times (Equation 2) for the devices as well as the corresponding means and 99 %-quantiles are shown. The response time histograms are shown in Figures 19 to 24.

Looking on the histogram for S1 in Figure 19, the majority of the samples are located close to 10 ms. Further contributions occur close to multiples of 10 ms, up to 500 ms. S2 has a similar behavior, with most of the samples close to 20 ms and a maximal response time of 140 ms. For S3 the equally distributed peaks are even more noticeable; the majority of samples occurs nearby 10 ms, and the maximal response time is only 80.2 ms.

R1 also shows response times around multiples of 10 ms, with the peak close to 30 ms. It seems that there is a strong connection between response times and the 10 ms time scale of the sysUpTime variable on these devices. However, some response times can be quite large.

Router R2 (Figure 24) and switch S4 (Figure 22) display a different behavior, focusing almost all of their response times to one or two quite small values around 10 ms without exceeding 20 ms. This is also seen from the mean values and 99 %-quantiles in Table 5. This could be an indication that these two devices do not have any problems in completing the GetResponse-PDU rapidly, and thus, outperform the other devices that seem to need more time in their processing of the requests. On the other hand, remember that only S1 and S2 seem to have up-to-date variables.

6 Conclusions

In this paper we have presented some experiences on obtaining reliable bit rate measurements from SNMPv1 agents. The measurements have been obtained on short time scales in real network environments. We demonstrate that it is possible to obtain correct measurements as long as special care is taken to the problems associated with it. When using a 5 minute polling interval SNMPv1 works for link speeds less than or equal to 100 Mbps. On devices with link speeds in excess of 1 Gbps, SNMPv1

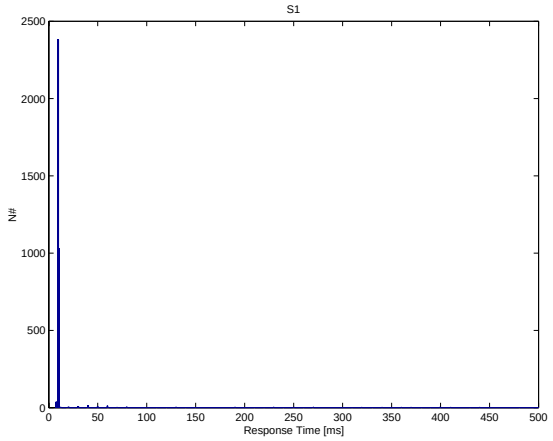

Figure 19: Histogram for S1 response times.

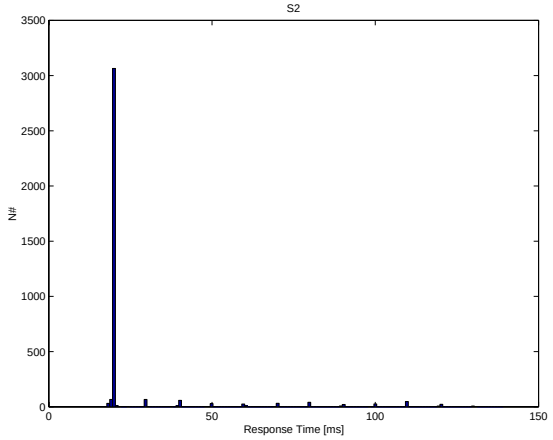

Figure 20: Histogram for S2 response times.

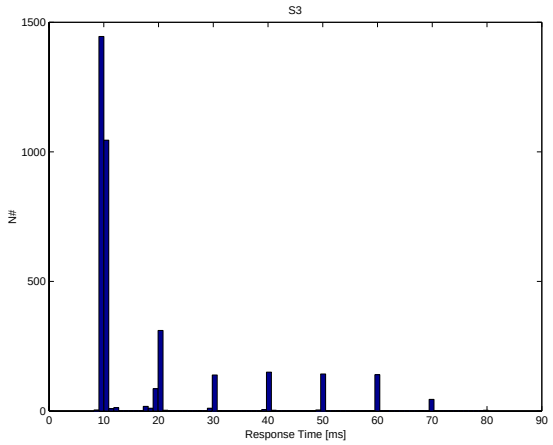

Figure 21: Histogram for S3 response times.

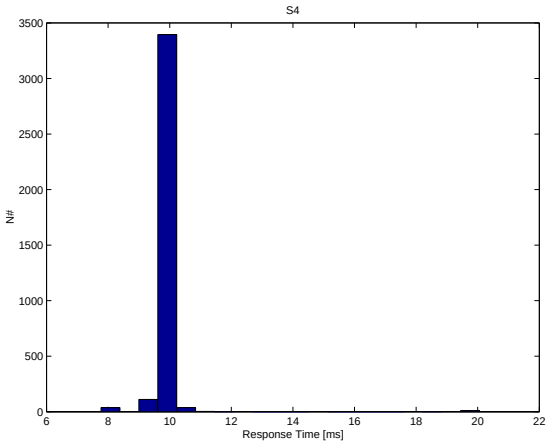

Figure 22: Histogram for S4 response times.

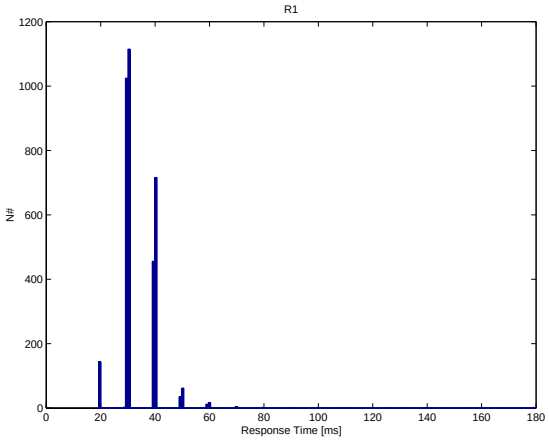

Figure 23: Histogram for R1 response times.

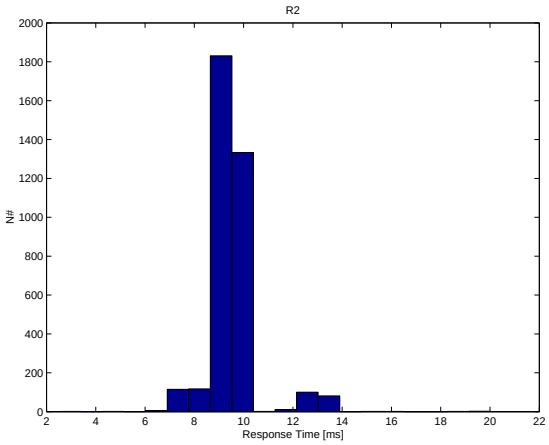

Figure 24: Histogram for R2 response times.

performs badly because of the frequent sampling needed to detect wrapping counters. Even though SNMPv3 is becoming a full Internet standard [9], we expect that SNMPv1 devices will still be used in the future.

None of the devices behaves as expected, i.e. updates its MIB variables synchronized to the sysUpTime variable. There seems to be a compromise between updating behavior and responsiveness. Some devices are good on updating and others on answering. A clarification on how an SNMP agent should behave could be of interest. Otherwise this behavior has to be determined for each and every device, to be able to get correct performance measurements.

It is difficult to do traffic engineering using SNMPv1. However, using the right sampling intervals in conjunction with a separated management network makes it suitable for most traffic engineering and network control issues requiring short sample intervals.

References

[1] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin, *Simple Network Management Protocol (SNMP)*, May 1990, RFC1157, STD0015.

[2] H. Hegering, S. Abeck, and B. Neumair, *Integrated Management of Networked Systems, Concepts, Architectures, and Their Operational Application*, Morgan Kaufmann Publishers, 1^{st} edition, 1999.

[3] M. Subramanian, *Network Management, principles and practice*, Addison-Wesley, 1^{st} edition, 2000.

[4] K. McCloghrie and M.T. Rose, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, March 1991, RFC1213, STD0017.

[5] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, *Introduction to Community-based SNMPv2*, January 1996, RFC1901, EXPERIMENTAL.

[6] B. Wijnen, D. Harrington, and R. Presuhn, *An Architecture for Describing SNMP Management Frameworks*, April 1999, RFC2571,DRAFT STANDARD.

[7] K. McCloghrie and F. Kastenholz, *The Interfaces Group MIB*, June 2000, RFC2863, Draft Standard.

[8] "Tcpdump public repository," http://www.tcpdump.org.

[9] "Snmp research: Snmpv3 goes full standard," http://www.snmp.com/news/snmpv3_fullstandard.html.