Analysis of Errors in Network Load Measurements

Stanislav Belenki Sven Tafvelin
Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, Sweden
{belenki,tafvelin}@ce.chalmers.se

ABSTRACT

The paper identifies elements in network monitoring systems that cause errors in the load measurements found in recent reports on network statistics from an academic backbone network. Two types of network monitors are investigated: counter-based and packet capturing. The paper explains how to assign an accuracy term to the load values in case of counter-based monitors and how to eliminate distortion in the case of packet capturing monitors. The paper also suggests an MIB to reduce the counter-based measurement error.

1. INTRODUCTION

From the moment the Internet appeared it has been measured in different ways in order to keep it working and plan its further development. Perhaps the most fundamental parameter of any communication network is its load. Measured load values provide indications of the overall performance of the network. It is on the basis of these values that a deeper analysis of the network is often carried out. For example, a load value exceeding 60% of the nominal capacity of a network link may indicate that the link is overloaded. Thus, it is necessary to identify the cause of the overload: an increase in the number of the users, misbehavior of a network protocol or a mistake in the routing.

The most well understood network load measurement is the periodic one. The measurement is based on periodic accumulation of the number of octets transmitted over the observed link over a certain time interval. The resulting load samples reflect different behavior aspects of the network depending on the length of the time interval. Setting the interval to one hour may thus help identify the busy periods of the network or link over a day. On the other hand, load samples collected with a periodicity of several seconds or a few minutes enable keeping track of statistical properties of the traffic that are otherwise averaged in more coarsely sampled statistics. The statistical properties or alarms derived from the finely sampled load values can be used e.g. for security purposes [1].

The measured values of the network load must be accurate enough to justify network management decision based on them. In case these values are distorted due to some flaw in the measurement system, it is difficult, if not impossible, to rely on them and the decisions they lead to. Thus, it is important to identify sources of the distortion, as well as its character and magnitude, in order to develop distortion-free measurement systems [2].

There are two major ways to obtain periodic samples of the network load. The first method relies on polling the network nodes for counters of octets transferred over interfaces of the nodes. The other method derives the load samples from packet or flow traces collected by such systems as tcpdump [3], OCXmon [4] or NetFlow [5]. The first method is used by the SNMP (Simple Network Management Protocol) and requires the network nodes to run an SNMP agent that serves the polls and maintains the counters. The second method usually employs some dedicated hardware that sniffs packets transmitted over the link to which it is connected and saves the time stamped packet headers in some log files. The log files are then processed by summing the sizes of the packets for each time interval. The systems that use the second method provide much more detailed information on the traffic than do the former systems and can use a single packet trace to produce the periodic network load samples with different sample interval length. However, these systems also require much greater resources than the systems that use the first method. Thus, collecting the traces usually requires a dedicated host with comparatively large storage capacity and a transmission medium splitter (e.g. optical) per each observed link in case the transmission medium is not shared. Furthermore, the use of flow-based systems like NetFlow limits the range of the sampling interval because the statistics are maintained per flow. This makes impossible sampling of the load over time intervals smaller than the duration of the shortest flow . At the same time the first method might require only one host that collects values of the octet counters from a number of routers serving a whole intranet. On the other hand queries issued by the host to the routers require the latter to consume their resources (e.g. CPU cycles) to process the queries while handling the traffic itself.

This paper presents an investigation into the two methods for obtaining the network load samples in order to identify sources of errors of the network load values generated by the systems based on these methods. The investigation includes theoretical and quantitative analyses of the errors. In a number of real life measurements performed in the study the measurement systems used are described in detail to assure that no error other than that induced by the method it employs distorts the load values.

The general problem of accurate measurements of Internet behavior is not new and has been investigated in a number of ways ([6], [7], [8]), although the problem of accurate load measurement has not received any significant attention. Nevertheless, authors of [9] explicitly mention the ability of TCP/IP stack implementation to affect the results of their study, meanwhile perhaps diminishing the significance of this distortion with respect to the low speed network the authors were studying.

The paper is organized as follows. Sections 2 and 3 describe the operation of each type of monitor and elements of the monitors that induce errors along with real life measurements of the errors. Section 4 provides a discussion of the results obtained in the previous sections, and section 5 concludes the paper.

2. COUNTER-BASED LOAD MEASUREMENTS - SNMP AGENT/CLIENT VERSION 3.5-3 CARNEGIE MELLON UNIVERSITY

2.1 The model

An SNMP[10] (Simple Network Management Protocol) traffic monitoring system includes an SNMP agent located at a host or a network node and an SNMP management station usually located at a workstation. The agent is responsible for maintaining a management database known as MIB, which contains, among other data, configuration, performance, fault parameters, and other statistics of the host or node where it is installed. The performance parameters also include statistics on network interfaces of the host/node, namely the number of octets passed through the interface in both directions, counters of packets that belong to specific network protocols, as well as other data of this type. Sometimes the SNMP management station and the SNMP agent are located at the same host. We, however, will investigate the case where the agent is placed at a network node and the management station is a remote workstation, which is common in cases where the agent is located at a network node.

To obtain load values from a remote SNMP agent, the management station sends SNMP GetRequest messages to the agent every T seconds. The agent receives the request, processes it, and sends back an SNMP GetResponse message containing the values of the requested parameters. The monitoring station then subtracts the number of reported octets from the value resulting from the previous poll. (Wrap around of the counters is taken into account when needed.) The resulting number of octets is attributed to the last interval T and this entry is logged into a file. The resulting interactions are shown in Figure1.

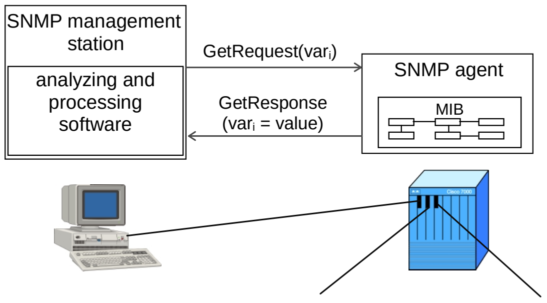
SNMP management station
analyzing and processing software
GetRequest(var$_i$)
GetResponse (var$_i$ = value)
SNMP agent
MIB

Figure 1. Interaction between an SNMP management station and an SNMP agent

However, a closer look at these interactions and their timing reveals details that can affect the accuracy of the logged data. Figure 2 illustrates this by showing how the interactions are decomposed into the following series of events: before the GetRequest is actually sent to the agent, it is delayed in the socket and interface queues of the management station (*MS Tx*); the transmission of the request to the agent‘s node adds another delay (*Tx Ms* $\rightarrow$ *Ag*); there the request is queued in the network interface device driver and then in its socket queues (*Ag Rx* altogether); finally the request is delivered to the SNMP software. The software performs the necessary inquiries and constructs an SNMP GetResponse containing the requested data (*Ag SNMP*). The response is transferred to the management station in a fashion equivalent to the transfer of the request from the management station to the agent’s node (phases from *Ag Tx* to *MS Rx*). The time required by an agent to read the relevant counters of the network node (*Ag SNMP*) depends on the network node‘s load situation. The higher the load, the longer the time required to read the counters. Also the network between the agent and the monitoring station may experience different load conditions during the phases *Tx MS*$\rightarrow$*Ag* and *Tx Ag* $\rightarrow$ *MS*.

All these effects introduce variable delays in all the above interactions. Thus, the *real time* between two successive polls is distorted in two ways. First, it is shifted by the minimum possible time it takes to process the phases from *MS Tx* to *Ag SNMP*.

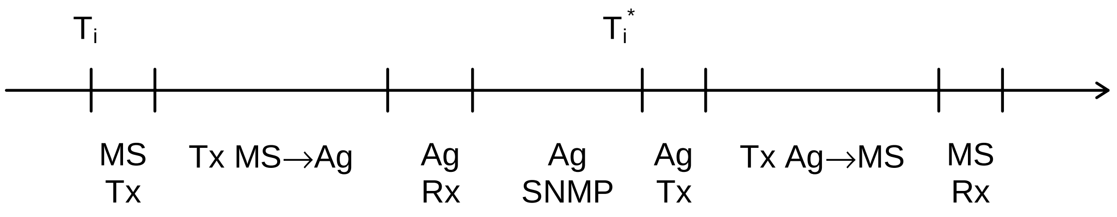

Figure 2. Chain of events within SNMP request-response interaction

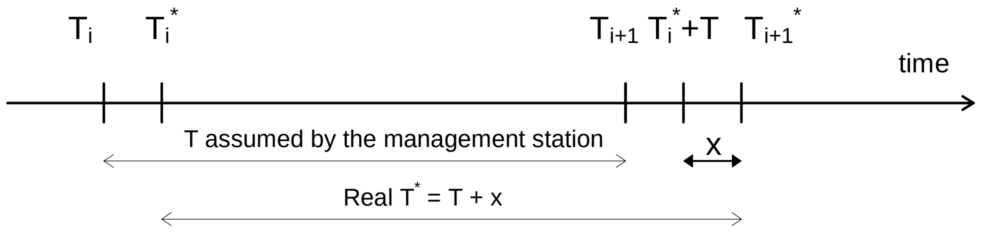

Figure 3. Two successive SNMP request/response interactions

Second, it is changed by the difference between the agent- and transmission-induced delays in successive interactions. For example, if the processing time of the request generated at time T_i was faster by x seconds than that of the request issued at time T_{i+1}, the real time interval would be T+x (the interval ends at time instance T^*_{i+1}). Packets processed by the node during these x seconds are mistakenly accounted for in the time interval (T_{i+1}-T_i, see Figure 3).

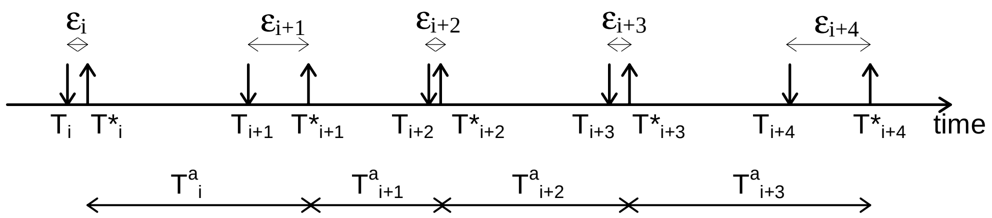

Figure 4. A number of successive GetRequest-GetResponse interactions

The above observations can be extended to intervals that consist of more than two successive polls of the counter. Figure 4 shows an example where the downwards arrows are instances of the GetRequest generation by the management station and the upwards arrows are instances of the corresponding counter reads by the agent. Let's denote the time between T_i and T^*_i of the ith GetRequest-GetResponse interaction (Figure 2) as ε_i and assume that the ith load sample derived from the ith and the i+1st values of the counter is the first load sample in the sequence shown by Figure 4. Then, $x_i = \varepsilon_{i+1} - \varepsilon_i$, and the actual time period between the ith and the i+1st counter reads is $T^a_i = T + x_i$. In analogy the i+1st interval is $T^a_{i+1} = T + x_{i+1}$, where $x_{i+1} = \varepsilon_{i+2} - \varepsilon_{i+1}$. Thus if $\varepsilon_{i+1} > \varepsilon_i$, then $T^a_i > T$. Excess packets accounted for in this load sample result in the calculation of a load value that is too high compared to the true value. Conversely, if $\varepsilon_{i+1} < \varepsilon_i$, then $T^a_i < T$. In this scenario, the computed load value for the interval T may miss some octets (this results in a load value lower than the actual one), if packets have arrived between time instances $T_i + \varepsilon_{i+1}$ and $T_i + \varepsilon_i$. On the other hand, should $\varepsilon_{i+1} = \varepsilon_i$, the calculated load value would be correct. Note that averaging the load calculation over a number of successive polls decreases the error. If ε_i and ε_j refer to the first and the last load values of the averaged sequence, the difference between both limits the error.

2.2 Measurements

A number of tests were performed with four different routers (three of Chalmers university's backbone, and one of the Swedish national academic backbone). All the routers were queried by GetRequest messages issued by a single central workstation. The tests were carried out late afternoon on several days within a one week time frame, with each test running for approximately one hour. GetRequests were issued once every two seconds. The queried objects were an ICMP packet counter and the CPU load of the router. The management station was running tcpdump (see section 3) while sending and receiving the SNMP packets. The linux kernel was modified to enable microsecond precision of the tcpdump time stamps. The packet traces were later used to calculate the delay differences of successive queries. The data retrieved from single routers

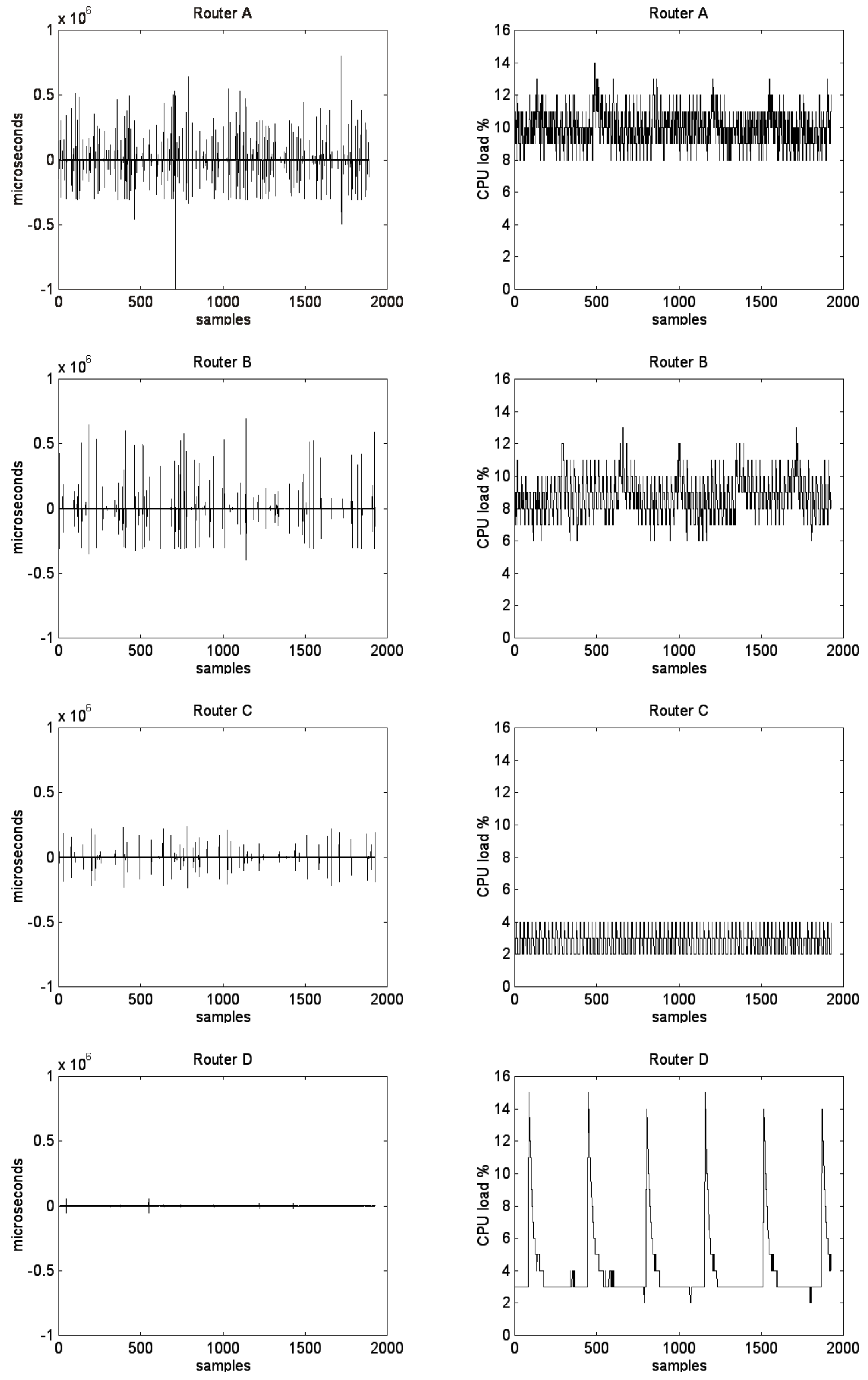

Figure 5. Measured delay difference and CPU load

did not change significantly from one day to another. Some of the time results are shown in Figure 5, where the delay difference (i.e. x) is plotted aside with the CPU load of the corresponding router. The top two rows of the plots correspond to two boundary routers of the university backbone (Routers A and B), the third row shows data from a Swedish national academic backbone router (Router C), and the bottom row contains plots from one of the university's backbone routers (Router D). It should be noted, that the SNMP queries of Router C had to pass through one of the two boundary routers. Nevertheless the delay differences measured for Router C are of a lower magnitude than those of the boundary routers. This allows us to conclude that the delay differences at Router C are not significantly affected by the traffic conditions at Routers A and B. Figure 6 shows a simplified topology of the network nodes and links between them. Although the CPU load of Router D was not lower than that of Router C, the delay differences of the SNMP queries associated with Router D are significantly lower in magnitude and variation than those of Router C. The four measurements suggest, that a higher variation of the CPU load results in a higher magnitude of the delay difference. It should be noted, that Router D was located closest to the workstation issuing the GetRequest messages. One could argue, that the delay difference may be proportional to the number of hops between the SNMP agent and the management station. However, the two boundary routers are closer to the workstation than the router from the Swedish national academic backbone (Router C), whereas the delay difference associated with the latter is less significant than that associated with the boundary routers. This favors the conclusion that the magnitude and frequency of delay differences were mainly caused by internal router mechanisms. The two top plots show a number of occasions where the delay difference exceeded 500 milliseconds. If the time interval T for the load calculation was less than 50 seconds, these events could distort the corresponding load calculation by introducing an error of over 1%.

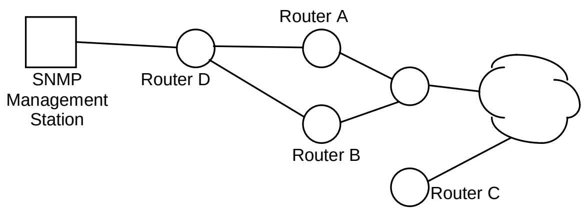

Figure 6. Simplified topology of the network between the routers and the workstation

3. PACKET CAPTURING MONITORS

3.1 The Model

The packet capturing monitors capture all or some of the packets on the network link to which they are connected. The capturing is achieved by setting the network interface card (NIC) into promiscuous mode by one method or another. For example, most Ethernet NICs can be set to the promiscuous mode by a command. If the link is a shared media, the connection does not require any modifications. However, when the media is dedicated, some sort of tapping hardware, e.g. optical splitter, is required. The monitor's software attaches a time stamp to the header of every packet it gets from the NIC and saves the resulting data structure in a log file. There are two key aspects in this procedure of obtaining the packet and time stamping. The first aspect is the level of the network stack of the operating system where the time stamp is attached, and the second is the precision of the time stamp. These two aspects affect the accuracy of the time stamp.

The possible components of an operating system, Linux 2.0.36 in particular, where the time stamp can be attached to the packet header are: the NIC interrupt routine, the NIC bottom half handler, the socket-level software, and, finally, the monitoring software operating at the user level. The further from the interrupt function the time stamping is done, the more the time stamp value of the packets is increased as compared with its real arrival time. The increase comes from the processing of the packets at each of the stages. Consequently, alternating loads imposed on the operating system can produce alternating differences between the real arrival time instance and the time stamp value. In the linux kernel, the NIC interrupt function records the arrival time of every packet into the corresponding field of the socket buffer structure. The structure is used to pass the packet further up the network stack [11]. The field is then read by tcpdump with the help of libpcap library routines.

The influence of the time stamp precision is quite explicit: the higher the precision, the more exactly one can identify the arrival instance of every packet. The value recorded in the socket buffer data structure by the interrupt function is read from a hardware clock in the computer. In the original version of the interrupt routine, the value has a precision of 10 milliseconds, but some minor adjustments to the code allow for microsecond precision of the value.

Figure 7 shows an example of tcpdump output format. The first field is the time stamp in format hours:minutes:seconds.microseconds, the rest is Ethernet source and destination addresses, the Ethernet type, length of the Ethernet frame and higher level protocol data.

To obtain values of the link load over a period of T seconds from a tcpdump output, one must write a simple script. The script would use the time stamp of the first packet from the output to derive the time origin for the sequence of the periods of T seconds.

Assume a tcpdump output longer than one sampling period T. The analyzing script scans the output and accumulates the number of octets in packets that have arrived within the sampling time intervals. The accumulated number of octets can later be divided by T to produce the link capacity utilization over the period. Let a packet be the first falling into one of the sampling intervals. That is, the time stamp (i.e. the clock value) of this packet is greater than T*n, the beginning of the nth sampling interval. Therefore all octets contained in the packet will be accounted for in this interval. However, it is possible that the packet had begun to arrive before T*n. Thus, the front part of the packet does not belong to the nth interval but is accounted for in it by the script. At the same time, the front octets of the packet are excluded from the previous interval value.

```
20:52:23.772446 0:e0:29:1b:c3:82 0:e0:4f:60:48:c0 0800 60: 129.16.20.181.1673 > 207.25.71.22.80: . ack 21454 win 8048 (DF)
20:52:23.777616 0:e0:29:1b:c3:82 0:e0:4f:60:48:c0 0800 60: 129.16.20.181.1673 > 207.25.71.22.80: . ack 24374 win 8760 (DF)
20:52:23.778548 0:e0:29:1b:c3:82 0:e0:4f:60:48:c0 0800 60: 129.16.20.181.1673 > 207.25.71.22.80: . ack 27294 win 8760 (DF)
20:52:23.851792 0:e0:29:1b:c3:82 0:e0:4f:60:48:c0 0800 60: 129.16.20.181.1679 > 209.67.3.82.80: . ack 2863519882 win 8760 (DF)
20:52:23.852849 0:e0:29:1b:c3:82 0:e0:4f:60:48:c0 0800 419: 129.16.20.181.1679 > 209.67.3.82.80: P 0:365(365) ack 1 win 8760 (DF)
20:52:23.874430 0:e0:29:b:cf:a ff:ff:ff:ff:ff:ff 0800 243: 129.16.20.173.138 > 129.16.255.255.138:udp 201
20:52:23.889148 0:e0:4f:60:48:c0 9:0:7:ff:ff:ff 020b 537: 3 > 2 at-lap#73 512
```

Figure 7. An example of tcpdump output

Let us denote the size of the left part of a packet hit by the beginning of a sampling interval as X, and the size of the left part of the packet hit by the end of the sampling interval as Y. Assign z = X - Y. If z is not zero, the load value of the time interval becomes erroneous. If z is more than the amount of empty space (i.e. the idle time times the link capacity) of the link during the interval, the resulting link utilization over the interval will be more than 100%. Even if z is smaller than the amount of the empty space or it is negative, it still imposes an error on the load value. Figure 8 illustrates the scenario. If the load values are averaged over a time period substantially longer than T, the significance of the error is reduced.

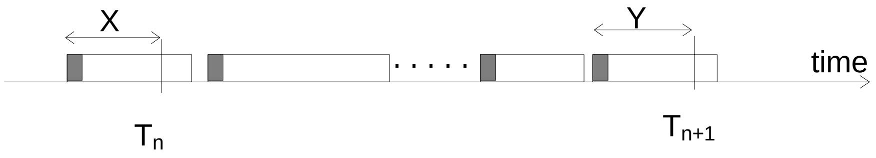

Figure 8. The distortion as an effect of processing tcpdump output

However, if the sampled values are used to report the peak load, one must take into account the error, whose maximal value is the number of octets in the longest possible packet.

To form an idea about the behavior of the distortion it was decided to calculate the probability of the event where distortion z takes certain values. The algorithm shown below was used to calculate the probability over the entire positive range of the distortion z = [0, MAX] (the case for the negative range is symmetric), where MAX is the maximum packet length:

```
for(z = 0 to MAX)
    for(L = z to MAX)
            for(R = 0 to MAX){
              if(L > 0 & R > 0)
        prob[z]+ = P{L}P{R}P{X - Y = z}$\lambda^2$;
             if(R = 0 & L > 0)
               prob[z]+ = P{L} P{X = z}$\lambda(1-\lambda)$;
             else
               prob[z]+ = $(1-\lambda)^2$;
            }
```

Here L is the length of the packet hit by the start of the interval, and R is the length of the packet hit by the end of the interval. P{X - Y = z} was calculated with the assumption that the start and the end of the interval hit every packet uniformly within each packet's length. That is, if a B-octet long packet is hit, the probability P{X = x} (or P{Y = y}) is 1/B. Thus P{ X - Y = z } = $\frac{1}{L}\frac{1}{R}(n-1)$, where n is the number of points on the line X - Y = z [12]. P{L} and P{R} are probabilities that packets of length L and R appear on the link. The probabilities were taken from an experimental packet length histogram derived from a wide area network packet trace [13]. Figures 9 and 10 show the histogram. Figure 10 shows the frequency in logarithmic scale to stress the few occurrences of packets longer than 1500 bytes (in the oval region). The resulting probability distribution functions of the distortion are shown in Figures 11 and 12, where the former assumes 100% utilization of the link and the latter assumes 60% utilization. As can be seen, when the utilization is less than 100% the probability P{z ≠ 0} scales by a factor of about λ^2, while the probability of zero distortion takes on an increase of $(1-\lambda)^2$.

Assume a link with nominal capacity of C bits/sec, and a time interval over which the load values are calculated equal to T sec. The relative distortion with respect to the interval can then be found as z/(TCλ), where λ is the link utilization during time interval T. With T = 10 sec, C = 34 Mbit/sec, λ = 0.6 and z = 700 octets, the relative distortion is $2.7*10^{-3}$ % of the real load value.

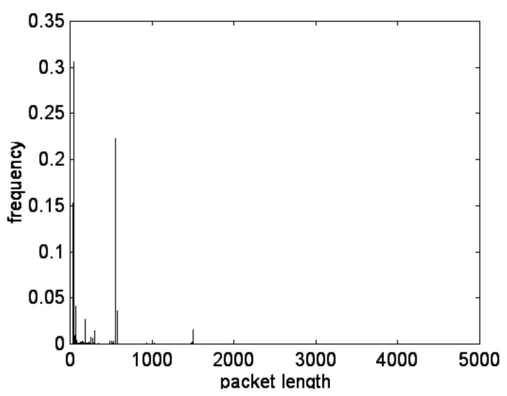

Figure 9. Experimental packet length histogram

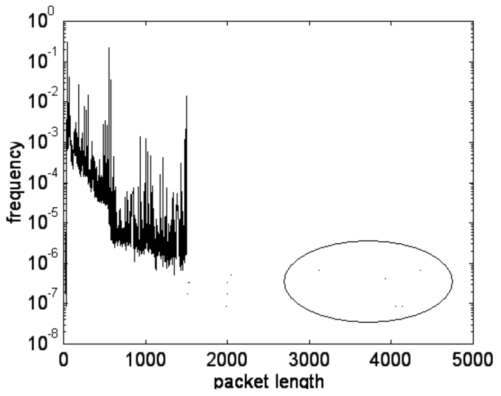

Figure 10. Experimental packet length histogram, logarithmic scale

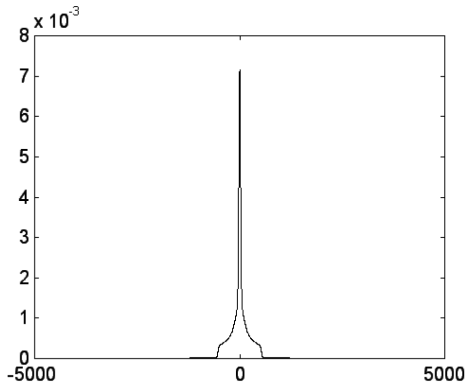

Figure 11. Probability distribution function of the distortion, 100% load

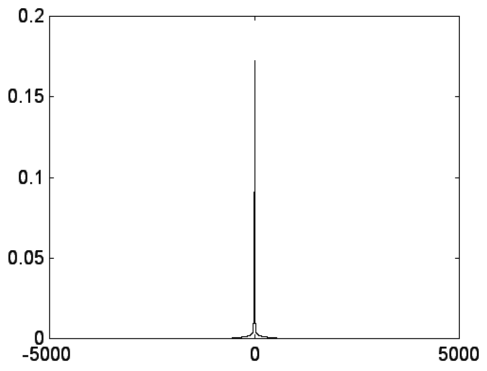

Figure 12. Probability distribution function of the distortion, 60% load

3.2 Measurements

A one hour tcpdump trace was analyzed with help of a simple parsing script. The trace was obtained from the department's local area network. It would be interesting to sniff links between the routers measured in the previous section. Comparisons of the counter-based measurements and the tcpdump measurements may reveal other sources of errors. However, we do not have the possibility to directly access the routers and sniff their links. Although we have a router for experimental purposes, we do not have the means to create a realistic heavy load on it. The script used to process the departmental tcpdump trace detected the cases where the edges of the sampling periods hit the packets. The sampling period was chosen to be 5 seconds; the observed link was a full duplex 10 Mbit Ethernet. Figure 13 shows the measured error, and figure 14 shows the corresponding 5-second load values. Figure 15 shows a histogram of the packet sizes in the trace. About a quarter of all of the packets were 1514 octets long. This makes the fraction of such packets greater than in the histogram used to calculate the probability density function of the error. This difference explains higher than expected levels of the error. Figure 16 shows the relative error calculated as z_i/l_i, where z_i is the error of sample i and l_i is the value of sample i. As was expected, the highest error caused little impact on the high load values, whereas the same error can cause an impact of more than 3% on low load values.

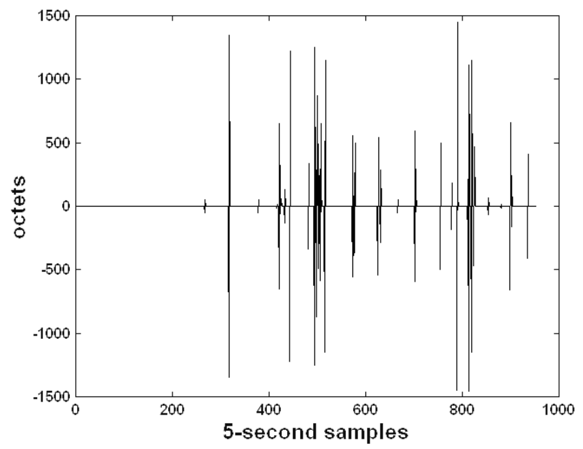

Figure 13. Experimental distortion of load values derived from a packet trace

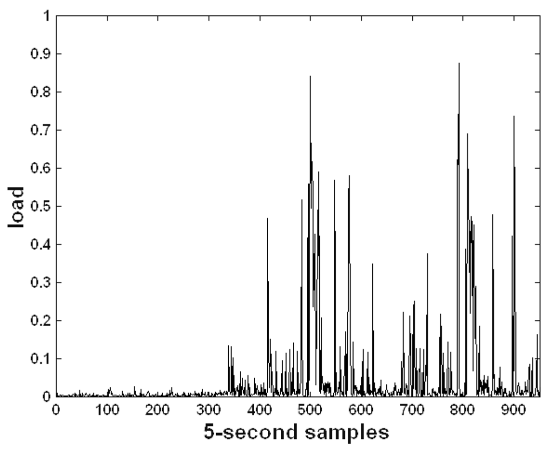

Figure 14. Relative load observed from the packet trace

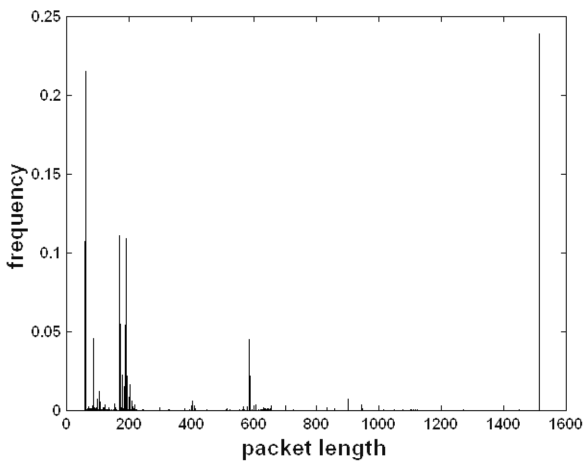

Figure 15. Packet length histogram of the packet trace

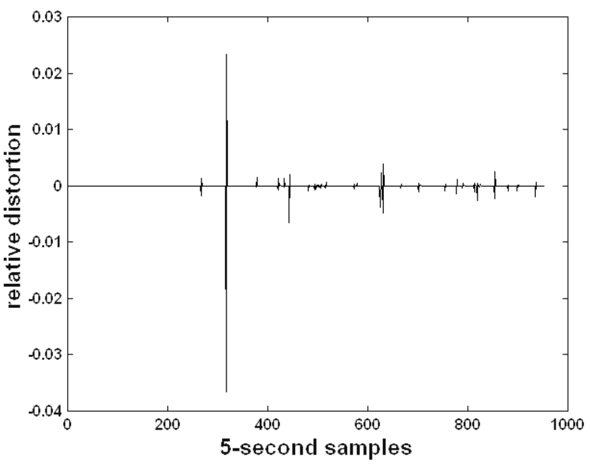

Figure 16. Experimental relative distortion

4. DISCUSSION

Both types of network monitors examined were found to be capable of distorting values of the network load. It is of course natural to try to eliminate or reduce the errors. Let us start with the counter-based monitors. There are temporal uncertainties in the measurement process that make it impossible to eliminate the error. Consider two GetRequest-GetResponse interactions. Assume that the management station can precisely measure the delay, ε, between the request and the response. The agent reads the counter at the *Ag SNMP* phase of the interaction. The phase is preceded and succeeded by the station-to-agent phases and the reverse phases, respectively. Since the duration of all of the phases can vary, even equal response-request delays of two successive polls do not guarantee that there were T seconds between the reads. That is, while total sums of the phases of two interactions are equal, the individual phases can be of a different duration. And the differences may cause shifts of the read instances with respect to each other.

Still, there is a way to calculate the range of the error. Assume that the delays of two successive request-response interactions are ε_i and ε_{i+1}. There are two worst cases that represent the shortest and the longest possible time distance between the two counter reads. The longest possible distance can be produced if the station-to-agent phases in the first interaction and the agent-to-station phases in the second interaction are of the shortest possible duration. It is also required that the first read is done early in the *Ag SNMP* phase, while the second read is done late in the phase. The shortest possible distance appears if the delays of the phases in the previous case are exchanged with each other. Let us denote all the phases on the left from *Ag SNMP* by S→A, and all the phases on the right from *Ag SNMP* by A→S. The longest distance is found as $T_{i+1} + (\varepsilon_{i+1} - min(A \rightarrow S)) - T_i - min(S \rightarrow A) = T + \varepsilon_{i+1} - min(A \rightarrow S) - min(S \rightarrow A)$. The shortest distance between the reads is $T_{i+1} + min(S \rightarrow A) - T_i - (\varepsilon_i - min(A \rightarrow S)) = T - \varepsilon_i + min(S \rightarrow A) + min(A \rightarrow S)$. The derivation of the marginal error terms from these equations is straight-forward. Assume that the link is 100% - loaded during the time that is added or removed from interval T. The error of the measurement is then in the range $[-(\varepsilon_i - min(S \rightarrow A) - min(A \rightarrow S))C;(\varepsilon_{i+1} - min(A \rightarrow S) - min(S \rightarrow A))C]$. C is the link speed. Thus, if the delays and the $min(S \rightarrow A)$ and $min(A \rightarrow S)$ terms are known, the management stations can calculate the error range. If ε is significantly larger (which is our experience) than the terms $min(S \rightarrow A$ and $min(A \rightarrow S)$, these terms can be neglected. The station can attach the range to the load value to indicate the possible error in it.

There are other methods to reduce the error in the counter-based systems. [14] proposes to fetch the local system time of the network node together with the counter value. The SNMP variable sysUpTime used in that method stores the time in seconds. As was shown by measurements in section 2, distortion of the polling interval takes values of up to half a second. Therefore, the value of the system up time is not precise enough to reduce or eliminate the error. Another possible improvement could be to attach the management station to the network node via a dedicated link, e.g. the serial port. This method eliminates part of the error caused by variations in the transmission delays. However, the method does not eliminate the part of the error caused by changes in the node's load. Furthermore, such a solution potentially requires the management station to be in the geographical vicinity of the monitored network node. This can be impossible in cases of remote monitoring.

We propose an idea that would allow network nodes to periodically read the counter values using the nodes' own timing facilities. The management stations would instruct the network nodes on the variables that they want to poll and on the time period of the polling. If the nodes consider this polling a priority job by e.g. running the procedure on an interrupt timer (the transmission of the GetResponse itself can be of low priority), the accuracy of the measurements is significantly improved. Implementation of this idea can be carried out through a modification of the Alarm group of RMONv1 [10]. The resulting polling group would then look like this:

```
pollTable
        pollEntry
                pollIndex
                pollInterval
                pollVariable
                pollOwner
                pollMangementTarget
                pollStatus
```

Here, pollIndex is a field used to identify a particular entry in the poll table; pollInterval stores the number of seconds of the polling interval; pollVariable identifies the object id (oid), i.e. the variable, that is polled; pollOwner contains a textual description of the management station; pollManagementTarget contains an index for the SNMP-TARGET-MIB[15] that stores data necessary to identify the management station on the network; pollStatus indicates the validity of the entry. If such an MIB is implemented, the management station sends a single SetRequest message containing values for all the objects except pollStatus (which is managed by the node's SNMP agent). The pollIndex value in the request is an increment of the previous row pollIndex value. The agent validates the values and sets pollStatus to the "valid" value. It then starts to collect the value of the variable and sends it to the management station in a GetResponse message every pollInterval seconds. If necessary, a more precise time interval can be defined by the addition of a pollIntervalMsec object to the table that would store the microsecond part of the polling interval. Moreover, the solution excludes the loss of the GetRequest packets issued by the management station.

The significance of the load measurement errors varies with the application of the measurements. Apparently, the error displayed in the measurements in section 2 is most likely negligible when the sampling period is e.g. one hour. An example of an application in which the error can be significant is the use of the measurements to control actions such as alarms or policing. The shorter the sampling interval, the higher the probability that the action will be affected by the error. One area in which the importance of the error is of no doubt is the correctness of the measurement system. If the source and magnitude of the error are not known, the operation of the entire measurement system can be questioned.

The potential error magnitude discovered in this study applies only to the network configuration used. As can be seen from the measurements performed, the tested nodes were not heavily loaded. At the same time, the distortion of the polling interval that was revealed was quite significant. Thus it is possible that heavily loaded network nodes can produce more severe distortions of the polling.

As for the packet capturing monitors, the following approach can be used to eliminate the error: if (Δ = [(first packet arrival time) - (packet length)/(speed of the link)] < T_i), then add to the previous load value (T - Δ)(speed of the link) bits and subtract the same number of bits from the current load value.

5. CONCLUSIONS

This paper presents the results of our work on the accuracy analysis of network load measurement systems. The two most common types of network monitors – counter-based and packet capturing - were investigated in order to find the elements that may be responsible for causing erroneous values of the periodic network load samples. A linux-based implementation of each of the systems was studied.

It was found that both systems are capable of inducing errors in the load values. While the errors of the packet capturing systems are limited by the number of octets in the longest packet, counter-based monitors were found to be potentially capable of unlimited distortion of the load values.

We developed a solution that allows elimination of the errors in the case of the packet capturing monitors by using the packet length field in the packet headers. In the case of the counter-based monitors an error range was defined that takes into account two extreme cases of the polling interval error. These cases are caused by uncertainty in the exact composition of the error. To calculate the error range, the management station needs to know the delays in the GetRequest-GetResponse interactions. The station also needs to know the minimum transmission delays between the SNMP application and the agent software. A modification to the alarm RMON MIB group was also suggested that would allow the management station to issue a single request for a number of periodic responses from the agent. In combination with the agent process running on a timer interrupt, the modification can reduce the error in this type of monitor. The advantage of the proposed MIB group over the RMON alarm group is twofold: it allows for precise load sampling and it offloads the alarm function from the router, transferring it to the management station.

It should be noted that the distortion inherent to the packet capturing monitors can also be present in counter-based monitors. This is true if the agent operates like a packet capturing monitor except that it does not time stamp the packets that it counts. In this case it is impossible to eliminate the distortion by the method described above.

Although packet capturing monitors provide an exact picture of the network load, their demand for dedicated resources does not allow them to be used exclusively. The proposed MIB for periodic polling of object values maintained by the network nodes makes possible retrieval of the values in the exact periodic manner. This makes such a solution preferable, where the errors induced by the edges of the sampling interval hitting packets are negligible.

6. ACKNOWLEDGMENTS

We thank the anonymous reviewer for providing helpful comments. We also thank Chalmers University Networking Group for the equipment used in the measurements as well as for interesting discussions on the subject.

7. REFERENCES.

[1] Phillip A. Porras, Alfonso Valdes. *Live traffic analysis of TCP/IP Gateways.* In proceedings of the 1998 ISOC Symposium on Network and Distributed System Security

[2] V. Paxson, G. Almes, J. Mahdavi, M. Mathis. *rfc 2330 Framework for IP Performance Metrics*.

[3] V. Jacobson, C. Leres, and S. McCanne, tcpdump available via anonymous ftp to ftp.ee.lbl.gov

[4] National Laboratory for Applied Network Research, Measurement and Operation Analysis Team, http://moat.nlanr.net/

[5] Net Flow home page available via http to http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nda/index.htm

[6] Vern Paxson. *On Calibrating Measurements of Packet Transit Times*. Proceedings of SIGMETRICS'98.

[7] Guru Parulkar, Douglas Schmidt, Eileen Kraemer, Jonathan Turner, and Anshul Kantawala. *An Architecture for Monitoring, Visualization, and Control of Gigabit Networks*. IEEE Network, September/October 1997

[8] K. Claffy, G. Polyzos and H-W. Braun, "Measurement Considerations for Assessing Unidirectional Latencies." Internetworking: Research and Experience, Vol. 4, No. 3, September 1993

[9] Timo Alanko, Markku Kojo, Heimo Laamanen, Mika Lijeberg, Marko Moilanen, Kimmo Raatikainen. *Measured Performance of Data Transmission Over Cellular Telephone Networks.* ACM Computer Communication Review, Vol. 28, No. 2, October 1995.

[10] William Stallings. *SNMP, SNMPv2, and RMON Practical Network Management. Second Edition*. Addison-Wesley Publishing Company, Inc. 1996. ISBN 0-201-63479-1

[11] Alessandro Rubini. *Linux Device Drivers.* O'Reilly and Associates, Inc.1998. ISBN 1-56592-292-1

[12] Yannis Viniotis. *Probability and Random Processes for Electrical Engineers*. WCB/McGraw-Hill. 1998. ISBN 0-07-067491-4

[13] National Science Foundation Cooperative Agreement No. ANI-9807479, the National Laboratory for Applied Network Research, available via http to http://www.nlanr.net/NA/Learn/packetsizes.html

[14] Tobias Oetiker, Dave Rand, *MRTG: Multi Router Traffic Grapher*. http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html

[15] D. Levi, P. Meyer, B. Stewart. *SNMP Applications*, rfc 2573