

Measuring the Impact of Active Probing on TCP

Andreas Johnsson and Mats Björkman
Department of Computer Science and Electronics
Mälardalen University, Sweden

Abstract

Available bandwidth measurement methods have become more and more accepted to be used when seeking the status of a network path. To measure the end-to-end available bandwidth without access to the path routers, these methods inject UDP based probe packets into the network path. The probe-packet load can transiently be high and thus it is important to study the impact on the existing network flows.

In this paper, we show and discuss our simulation results on how the TCP flows are affected when injecting probe packets with different flight patterns into the network path. We investigate the relation between the amount of injected probe packets and the reduction in TCP performance. Further, we suggest a quantitative definition of the term “network friendly probing”.

1 Introduction

Active measurement methods inject so called probe packets into the network path in order to measure network properties. The range of active measurement methods spans from simple ping applications, that inject a small amount of probe packets, to advanced end-to-end available bandwidth¹ measurement methods that require more measurement samples (referred to as samples in the rest of the paper) and thus inject more probe packets. There are many examples of the latter, such as ABGET [1], BART [2], Pathchirp [3], Pathload [4], Spruce [5] and TOPP [6][7]. Available bandwidth measurement methods typically rely on self-induced congestion. That is, the methods inject UDP based probe packets into the network path with a predefined packet separation to cause network congestion for a short time interval. During the probe-packet traversal of the network, the predefined separation changes due to competing cross traffic [8][9] or

¹The available bandwidth is defined as the unused portion of the bottleneck link capacity during some time interval. The estimate is independent on the network protocols used.

due to the bottleneck spacing effect [6]. The probe-packet separation is measured at a receiver. The separation between two probe packets constitutes a sample. These samples are then used in an estimation algorithm to calculate the end-to-end available bandwidth. The estimates of the available bandwidth can later be used in applications that rely on properties of the path. A good overview of methods is presented in [10].

Efforts have been made previously to develop end-to-end available bandwidth measurement methods and to compare their properties, such as accuracy and speed. However, in this paper we focus on how the cross traffic (i.e. all packets in the network but the probe packets) is affected by the probe packets. When the cross-traffic flow is adaptive, that is the send rate depends on network latency and packet loss, the injected probe packets can adversely decrease the performance of such flows.

In this paper we show and discuss our results on how one important dynamic protocol, TCP, is affected when injecting probe-packet trains with different lengths and intensities into the network path. That is, injecting the probe packets with different flight patterns. Within this study the relation between the number of injected probe packets and the reduction in TCP performance is also examined. These findings then evolve into a discussion and definition of the term “network friendly probing”.

2 Research questions and experimental setup

The aim of this paper is to show and discuss how UDP based probe packets sent in varying flight patterns affect TCP flows, the major transport protocol used on the Internet. The research questions are: “How does network probing affect TCP performance?” and “what is network friendly probing?”.

NS-2 has been used to study the above research questions. A network topology was constructed, shown

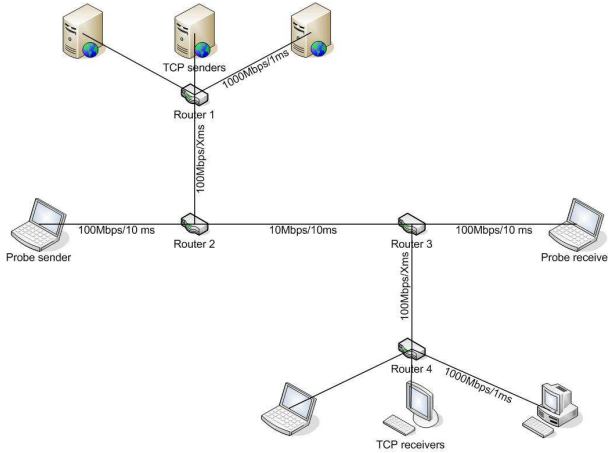


Figure 1: The simulation topology used to study the impact of probe packets on TCP. X in the figure can be 15, 25 or 35 ms.

in Figure 1, which consists of the following nodes: a probe-packet sender and receiver, one bottleneck link (between router 2 and router 3 having a link capacity of 10 Mbps and 10 ms latency) and 1 - 19 TCP SACK (with delayed acknowledgments) senders and receivers (in the figure only three senders and receivers are shown) connected to the bottleneck link via a router (router 1 and router 4). The total latency between a TCP sender and receiver can vary between 42, 62 and 82 ms. X in the figure is 15, 25 or 35 ms depending on the desired latency. The TCP packet size is fixed to 550 bytes.

The router queue size is limited only at the bottleneck link and is fixed to 100 packets (MTU = 1500 bytes). In [11] it is written that a common backbone queue size should be able to store 250 ms of network traffic. This corresponds to approximately 200 maximum size packets on a 10 Mbps link. However, 1) in [11] it is pointed out that queue sizes have shown a decreasing trend in the past and that this trend probably will continue and 2) 250 ms is a rather high delay for a non-backbone 10 Mbps bottleneck link (consider IP-telephony for example). Due to these reasons we have used the above stated queue. Droptail has been chosen to be the router drop policy.

The injection of probe packets is done at the probe-packet sender node. The probe-packet train length and the number of probe-packet trains per second vary. The separation between probe packets inside a probe-packet train is proportional to a bit rate that is uniformly distributed between 5 and 15 Mbps. The probe-packet size is fixed to 1200 bytes.

A synthetic probe-packet generator has been used

in order to circumvent uncertainty due to the use of a specific available bandwidth measurement method. The implementation of BART used in [2] does inject the probe-packets in this random fashion to produce an estimate of the available bandwidth once a second over time. It should be noted that most end-to-end available bandwidth measurement methods use trains or pairs of different length when injecting probe packets. Pathchirp is one exception; it uses trains with exponentially decreasing separation between the probe packets instead of equally sized separations. Pathchirp is discussed in Section 5.

To avoid synchronization in the simulations, random timers are introduced. The timers function as follows: Each TCP flow starts at time $0 + \delta$, where δ is uniformly distributed between 0 and 5 seconds. Further, the calculated separation s between two probe-packet trains is uniformly distributed between $[0.95 * s, 1.05 * s]$. The simulation time is fixed to 100 seconds. Each simulation has run 10 times.

How similar is the simulation topology described above and a real network? The topology can be thought of as a rather simple network, especially since all TCP flows experience the same minimum latency. Further, in a real network the traffic on the bottleneck link will also consist of UDP packets, MAC-layer packets and so on. The TCP flow aggregation on the bottleneck is also an important aspect of this simulation setup. In this paper the aggregation is varied between 1 and 19 flows as described above. Intuitively, the impacts from the probe packets are higher on links with low aggregation. The impact of the probe packets decreases with increasing aggregation towards an asymptotic value.

It is most likely that the results in the simulations point out the direction of the results that can be obtained in a real network. This assumption also holds if the network topology changes, as long as there only is one bottleneck link between the probe-packet sender and receiver. Then the probe packets only cause packet loss and congestion at one point between senders and receivers. Such measurements, in real networks, are left to future work since it requires a large-scale testbed.

3 Results

3.1 Initial results

In this section some initial simulation results are shown to illustrate the impact of probe packets on TCP in order to state that this is an important problem to study.

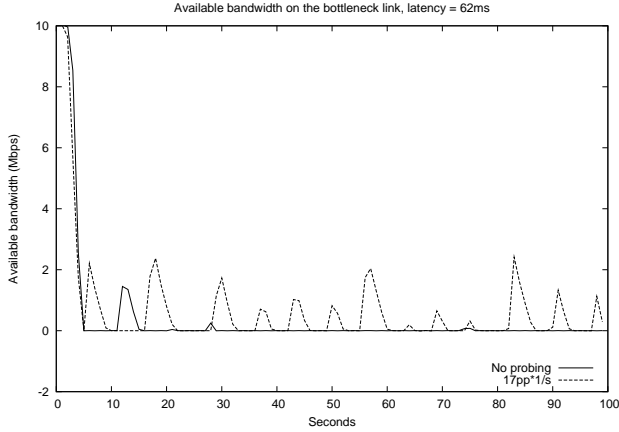


Figure 2: The available bandwidth changes when the TCP flows are affected by the probe packets. The number of aggregated TCP flows is 5.

It is rather obvious that when injecting probe packets, the cross-traffic flows on the bottleneck will have less bandwidth to consume. However, this is not necessarily true. If, for example, a probe-packet sender is injecting one probe-packet train consisting of 17 probe packets once a second (used by e.g. BART and TOPP) the available bandwidth transiently increases. That is, the bandwidth to consume has increased.

The results in Figure 2 originate from a simulation where 5 TCP flows are routed through the bottleneck link. The minimum latency between a TCP sender and a receiver is fixed to 62 ms. The graph shows how the available bandwidth (i.e. the true available bandwidth on the bottleneck link) changes over time in two scenarios: In the first scenario the TCP packets do not have to interact with probe packets while in the second scenario, the TCP packets and probe packets do interact on the bottleneck link. When the TCP backoff mechanism is triggered for a single or multiple flows, an increase in available bandwidth is visible. The important observation from this plot is the following; under the impact of probe packets, the available bandwidth on the link has increased compared to the first scenario, even though the probe-packet sender adds traffic to the bottleneck link.

In Figure 3 the number of transferred bits by the aggregated TCP flows is shown on the y-axis over time. This graph is created using the same simulation results as the one in Figure 2. It is clear that the number of transferred bits decreases when probe packets and TCP packets interact on the bottleneck link, compared to when no probe packets are injected. The number of transferred bits by TCP is decreased

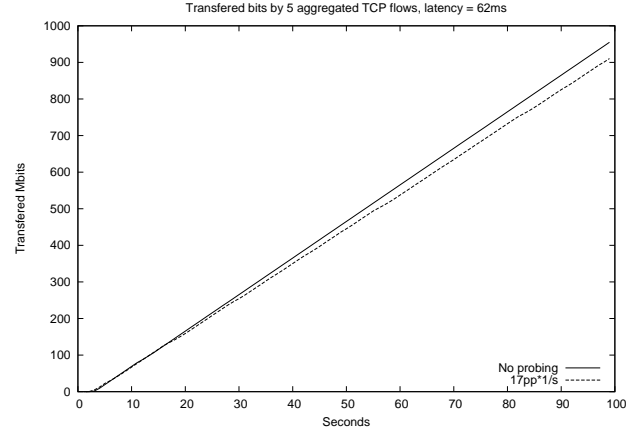


Figure 3: The number of transferred bits decreases when injecting probe packets in the path. The number of aggregated TCP flows is 5.

from approximately 950 Mbit to 900 Mbit during the 100 second simulation. It should be noted that the amount of injected probe traffic during the simulation is $17 * 1200 * 8 * 100 = 16.32$ Mbit. This is less than the decrease of 50 Mbit experienced by the aggregated TCP flows shown in the figure. The important conclusion is that probe-packet trains of length 17 are bursty and have negative impact on TCP performance.

Instead of sending one probe-packet train of length 17 once a second, one could send 16 pairs of probe packets to obtain the same number of samples. Then the plots in Figure 2 and 3 will differ. The question is how much, why and what flight pattern of the probe packets is best from the TCP perspective. In the next section a deeper study of this phenomenon is presented. The probe-packet train length varies along with the latency of the path used by the TCP flows. Such simulations will show the impact of probe packets on the performance of TCP.

3.2 Measuring the impact of active end-to-end probing on TCP

In this section a number of simulation results are presented to illustrate the impact of probe packets on TCP performance. The interpretation of the TCP performance in this paper is the number of transferred bits during a simulation (i.e. 100 seconds).

Each plot in Figure 4 to 6 corresponds to results obtained from using a fixed number of samples. An example: the total number of transferred TCP bits, when injecting one probe-packet train of length 17 once a second is compared to results obtained when

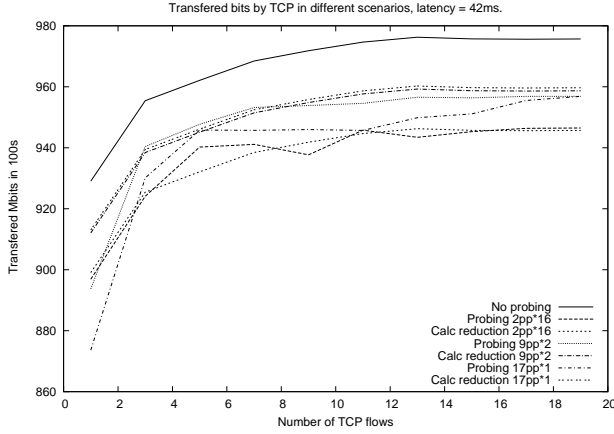


Figure 4: The number of transferred TCP bits in different scenarios.

injecting two probe-packet trains of length 9. In both cases the probing application obtains 16 samples per second. It has been shown that 16 samples per second give accurate estimates, in reasonable time, of the available bandwidth when using available bandwidth tools such as BART and TOPP. One sample is defined as the separation in time between probe packet n and $n - 1$ in a probe-packet pair or train.

If the number of samples per second is fixed, then shorter probe-packet trains will inject a larger number of probe packets compared to a longer probe-packet train. On the other hand, shorter probe-packet trains are less bursty.

The plots in the Figures 4 to 6 show the number of aggregated TCP flows on the bottleneck link on the x-axis. The y-axis describes the TCP performance (i.e. the total number of transferred TCP bits by all aggregated TCP flows) over 100 seconds. In each graph a curve corresponding to the number of transferred bits without impact from probe packets is also plotted. Confidence intervals have been left out in order to improve readability.

The results shown in Figure 4 correspond to a simulation where the minimum latency between TCP senders and receivers is set to 42 ms. The solid line is the TCP performance without impact from probe packets, and hence can be seen as the optimal TCP performance line for the network path used in the simulations. Results from several scenarios are shown along with the optimum line, even though the lines can be hard to distinguish from each other. The scenarios are the following: Injection of 16 probe-packet pairs, 2 probe-packet trains of length 9 and 1 probe-packet train of length 17, once a second respectively. Each

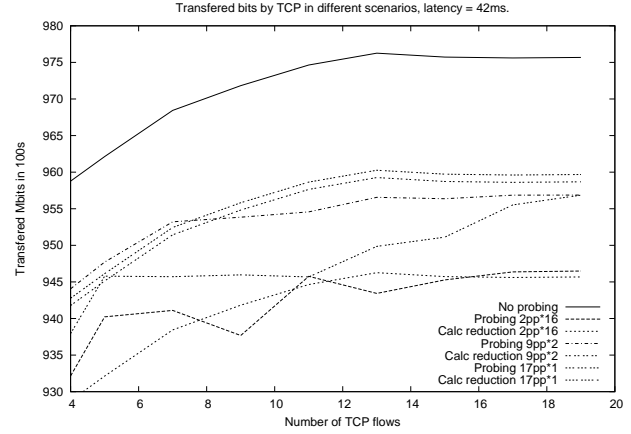


Figure 5: The number of transferred TCP bits in different scenarios.

configuration of the probe-packet flight pattern result in 16 samples per second. With each scenario a reference line corresponding to the calculated reduction in TCP performance with respect to the probe-packet flight pattern is also shown. In this paper a very simple calculation of the reduction in performance has been made. That is, the number of transferred TCP bits without probing minus the number of injected probe-packet bits. This line should be viewed as a reference line and not a correct value of the performance reduction.

To increase readability the case with 1 and 3 aggregated TCP flows are omitted in Figures 5 and 6.

The plot in Figure 5 is the reduced version of the plot shown in Figure 4. From the reduced plot there are several important observations that could be made. Compare the line corresponding to the calculated performance reduction when injecting a probe-packet train of length 17 into the path with the actual performance reduction. The actual reduction in performance is much greater, in a large part of the plot, compared to the calculated reference line. In the two other cases, that is when injecting 16 probe-packet pairs and 2 probe-packet trains of length 9 the difference between calculated reduction and actual reduction in TCP performance is smaller. This is better in some sense, because then the probe-packet sender knows how large the impact from the probe packets is on TCP performance. Thus, it can take that into consideration when estimating the available bandwidth. One more thing should be noted about the plot, and that is that the reduction in TCP performance is less when probing with probe-packet trains of length 9 compared to the case when injecting probe-

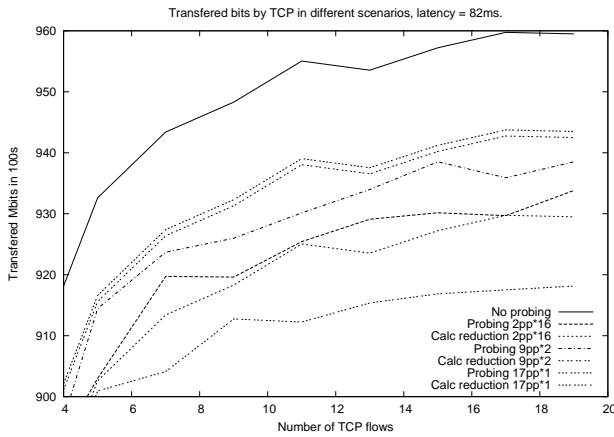


Figure 6: The number of transferred TCP bits in different scenarios.

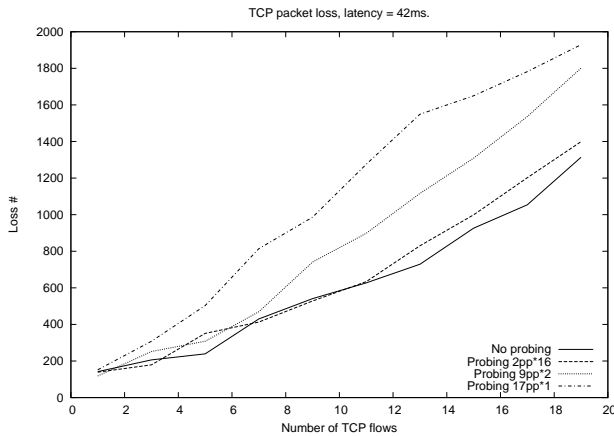


Figure 7: The number of lost TCP packets.

packet pairs or probe-packet trains of length 17. Now, to minimize the impact of the probe packets on TCP one might choose to use medium-length probe-packet trains.

Figure 6 corresponds to simulation results where the latency between TCP senders and receivers has increased to 82 ms. The same pattern is visible here, but perhaps even more clear. The difference between the calculated reduction in TCP performance when probing with one probe-packet train of length 17 and the actual reduction is significant. The differences in the two other scenarios are, as in the previous case, much smaller. If the probe-packet sender intends to minimize the reduction in performance experienced by the TCP flows, it seems to be a good choice to inject probe-packet trains of length 9 two times per second.

Simulations have also been performed using other

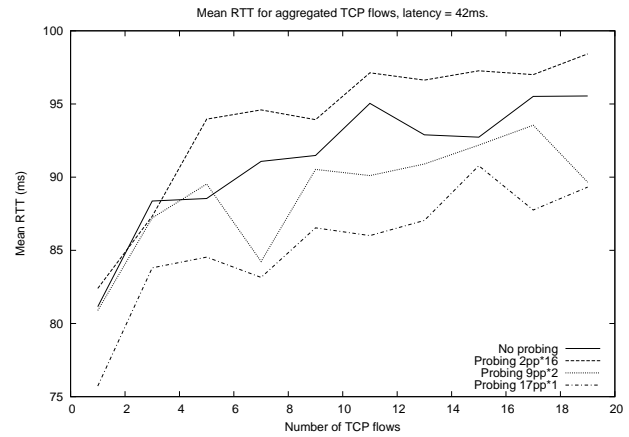


Figure 8: The mean RTT.

probe-packet train lengths (but still keeping the number of samples fixed at 16). However, the reduction in TCP performance does not decrease much further and thus those simulations are left out of this paper.

It has been discussed in the past whether probe-packet pairs or probe-packet trains is the most network friendly flight pattern. As we have shown above, it is not trivial to decide. The reason for this is that the injection of probe packets affects both the TCP loss rate and round-trip times. This is further discussed below.

In Figure 7 the TCP packet loss is shown on the y-axis. The x-axis is the number of aggregated TCP flows over the bottleneck link. The loss is minimized when no probing is performed (as expected). Further, it is seen that the packet loss when injecting probe-packet pairs, 16 times per second, is nearly equally low as in the case with no probing. Increasing the burstiness of the probe packets (i.e. increasing the probe-packet train length) increases the number of lost TCP packets. Based on this one would assume that using the flight pattern of 9 probe-packets twice a second should decrease the TCP performance more compared to the case when injecting probe-packet pairs. However, as seen above this is not the case. Why is that?

In Figure 8 the mean TCP round-trip time is plotted for the different flight patterns. From the plot it is clear that injecting probe-packet pairs increases the TCP round-trip time, which in turn decreases the TCP performance. Injecting probe-packet trains of length 9 on the other hand decreases the round-trip time. This is due to a higher loss rate, which decreases the queuing time in the routers for the TCP packets. Hence the TCP performance is not reduced as much as in the probe-packet pair case. That is, both the TCP loss

rate and mean RTT are affected, in different ways, by the probe packets sent in different flight patterns.

Important conclusions from this section are the following:

- It seems that medium length probe-packet trains (9 packets) are preferable in order to minimize the reduction in TCP performance.
- TCP performance decreases due to packet loss and increased round-trip time caused by the probe packets.
- The available bandwidth can actually increase when sending probe packets with a bursty flight pattern through a network path.

4 Network friendly probing

Many applications, such as games and IP-telephony, use UDP instead of TCP. Some of these applications have to transfer large amounts of data over the network. This can cause congestion in the network, since UDP does not have built-in congestion control mechanisms. Congestion caused by UDP is harmful to other UDP flows as well as to TCP flows sharing the bottleneck. The Data Congestion Control Protocol, DCCP [12], is one attempt to standardize a congestion control protocol for flows that does not need (or performs worse using) TCP.

However, in the case with available bandwidth measurement methods the probe-packet rate (UDP packets) can not be controlled in the same way, since these methods rely on causing congestion. Still, there is a need to make available bandwidth measurement methods as network friendly as possible.

The findings in the previous section provide valuable insights to a quantitative definition of *network friendly probing* for available bandwidth measurement methods. In this paper, the following definition is proposed:

Definition: If 1) the reduction in TCP performance is minimized and 2) during a time interval, the injection of x Mbit of probe packets does not decrease the total amount of transferred TCP bits more than x Mbit, the injection of probe packets is considered network friendly.

Using this definition, different probe-packet flight patterns can be compared to each other when considering network friendliness. Of course, to optimize available bandwidth measurement methods, estimation accuracy and time aspects must also be considered.

Take the plot in Figure 6 as an example. Comparing the case when injecting 16 probe-packet pairs to the case when injecting 2 probe-packet train of length 9 per second, at TCP aggregation 19 on the x-axis; which is the most network friendly alternative to use with respect to the above definition? The relation between calculated and actual reduction in TCP performance is approximately the same. However, it is clear that injecting probe-packet trains of length 9 results in a smaller reduction in TCP performance compared to the probe-packet pair case. Thus, using 2 probe-packet trains of length 9 per second is the best alternative.

From the view of available bandwidth measurement methods it is important to keep a one-to-one relation between the amount of injected probe packets and the reduction in TCP performance, not only due to the definition of network friendly probing above. Consider the following; if a probe-packet sender injects x Mbps of probe traffic while the reduction is higher than x , then the packets injected by the probe-packet sender may actually have increased the available bandwidth. This is a problem that must be addressed (e.g. by being network friendly). Another question that is important to ask is: what was the available bandwidth before the probe-packets actually were injected? Is it possible to calculate it if the one-to-one relation is not fulfilled?

A study of how network friendly state-of-the-art available bandwidth measurement methods are, compared to each other is left to future research.

5 Tools with special flight patterns

One end-to-end available bandwidth tool that does not inject probe packets using probe-packet pairs or trains is Pathchirp. Instead, Pathchirp injects the probe packets in so called chirps where the separation between two successive probe packets is exponentially decreasing. Using this method several probe-packet rates can be scanned in one chirp. Since the scope of this paper is to investigate the impact of active probing on TCP such a flight pattern is of interest.

In Figure 9 it is shown how much the probe packets injected by Pathchirp affect the TCP performance. Lines corresponding to the impact of probe-packet trains of length 9 twice a second are shown for comparison. The number of injected UDP based probe packets by Pathchirp during 100 seconds in different scenarios is shown in Table 1. The probe-packet size is 1200 bytes. The results are means of 10 simulation

No. TCPs	1	3	5	7	9
No. packets	2114	921	767	740	806
No. TCPs	11	13	15	17	19
No. packets	722	631	616	604	562

Table 1: Number of injected probe packets by Pathchirp in different scenarios during 100 second simulations.

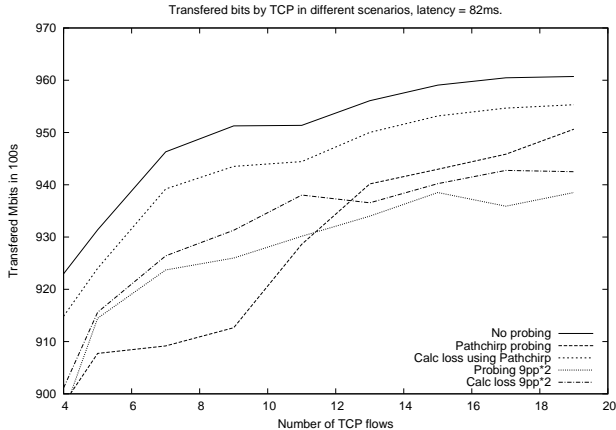


Figure 9: The number of transferred TCP bits in different scenarios.

runs.

From the figure it is clear that Pathchirp also affects the TCP performance in a negative way. At a higher TCP aggregation level the Pathchirp way of injecting probe packets is more network friendly than using probe-packet trains of length 9. These results give a hint of how network friendly Pathchirp is.

However, it is not possible to actually compare the two cases straight off since the Pathchirp curve originates from a real tool that varies the number of injected probe packets depending on the current path conditions while the probe-packet train injection is done by a synthetic probe-packet generator.

6 Discussion

As shown in this paper the impact of the probe packets on TCP performance is larger than the actual number of bits injected. But is it really a big problem?

In Figure 6² the number of transferred bits by 19 TCP flows in one simulation run is decreased from 960 Mbits to 918 Mbits when injecting probe-packet

²The same reasoning can be made for other TCP aggregation levels as well as for the results shown in Figure 5.

trains of length 17. That corresponds to a decrease of the TCP performance close to 4.5%. The calculated reduction for this flight pattern is 1.7%. Further, the procentual difference between the impacts caused by injecting probe packets with different flight patterns is quite small.

When injecting probe-packet trains of length 9, which is the network friendly flight pattern as described above, into the network path the reduction in TCP performance is only 2.2% (in Figure 6 at TCP aggregation level 19). The calculated reduction is 1.9%. *The conclusion is that the impact of the probe packets on TCP performance is quite small as well as the difference between using different flight patterns.* Even though the difference is small it is desirable to use the most network friendly flight pattern in order to reduce the impact on TCP as well as other network flows.

It is still a problem that the TCP performance is decreased by the probe packets. This problem gets more apparent when available bandwidth measurement methods are used in larger scales. That is, when several measurement sessions may take place concurrently over the same network path. Then the probe packets will push back the TCP flows.

How to solve this problem is left to future research. However, viable approaches is to be network friendly. It is also important to investigate whether application traffic can be used as probe packets in order to minimize the overhead on the network.

7 Conclusions

Since TCP is a dynamic protocol that varies the send rate depending on network latency and loss rate, the injection of probe packets will affect its performance.

The impact of probe packets, injected by available bandwidth measurement methods, on TCP has been investigated in this paper. The investigation has been performed in an NS-2 environment.

The results indicate that medium length probe-packet trains (or perhaps chirps) are the flight pattern to be used in order to minimize the reduction in TCP performance.

The reduction in TCP performance is due to the fact that the probe packets cause TCP packet loss, by long probe-packet trains, and increased round-trip time, mainly caused by many probe-packet pairs.

The term “network friendly probing” has been defined and discussed in the paper.

Future work is to compare how network friendly end-to-end available bandwidth measurement methods are compared to each other. It will also be investigated

what estimates will be produced by available bandwidth measurement methods when the cross traffic mainly consists of TCP flows. The relation between network friendliness and measurement accuracy is also an open issue.

References

- [1] Demetres Antoniadis, Manos Athanatos, Antonis Padagogiannakis, Evangelos P. Markatos, and Constantine Dovrolis, "Available bandwidth measurement as simple as running wget," in *Passive and active measurement (PAM) workshop*, Adelaide, Australia, 2006.
- [2] S. Ekelin, M. Nilsson, E. Hartikainen, A. Johnsson, J. Mångs, B. Melander, and M. Björkman, "Real-time measurement of end-to-end available bandwidth using kalman filtering," in *Proceedings to the IEEE/IFIP Network Operations and Management Symposium*, Vancouver, Canada, 2006.
- [3] Ribeiro, Riedi, Baraniuk, Navratil, and Cottrel, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and Active Measurement Workshop*, 2003.
- [4] Manish Jain and Constantinos Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM*, Pittsburgh, PA, USA, Aug. 2002.
- [5] Strauss, Katabi, and Kaashoek, "A measurement study of available bandwidth estimation tools," in *ACM SIGCOMM Internet Measurement Workshop*, 2003.
- [6] Bob Melander, Mats Björkman, and Per Gunningberg, "Regression-based available bandwidth measurements," in *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunications Systems*, San Diego, CA, USA, July 2002.
- [7] Andreas Johnsson, Mats Björkman, and Bob Melander, "An analysis of active end-to-end bandwidth measurements in wireless networks," in *Proceedings to the IEEE/IFIP workshop on End-to-end Monitoring Techniques and Services*, Vancouver, Canada, 2006.
- [8] Attila Pásztor and Darryl Veitch, "The packet size dependence of packet pair like methods," in *Tenth International Workshop on Quality of Service (IWQoS 2002)*, Miami Beach, USA, May 2002.
- [9] Andreas Johnsson, Bob Melander, and Mats Björkman, "On the analysis of packet-train probing schemes," in *Proceedings of the International Conference on Communication in Computing*, Las Vegas, 2004.
- [10] R.S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network Magazine*, 2003.
- [11] Damon Wischik and Nick McKeown, "Part 1: Buffer sizes for core routers," in *ACM/SIGCOMM CCR*, 2005.
- [12] Eddie Kohler, Mark Handley, and Sally Floyd, "Datagram congestion control protocol (dccp)," IETF draft, Dec. 2005, <http://www.ietf.org/html.charters/dccp-charter.html>.