

# A Passive Method for Estimating End-to-End TCP Packet Loss

Peter Benko and Andras Veres

Traffic Analysis and Network Performance Laboratory, Ericsson Research, Budapest, Hungary  
{Peter.Benko, Andras.Veres}@eth.ericsson.se

**Abstract**—This paper presents a passive end-to-end loss monitoring method, which relies on traffic monitoring at a core or ingress router interface. The monitoring node captures the packets of TCP connections generated by end-hosts. Based on the seen sequence number pattern the loss ratios are estimated for the two segments of end-to-end path divided by the monitor. This feature is especially useful if the monitoring node is placed at the border of an autonomous system, e.g., at the egress point of an ISP. When applied to mobile Internet access, packet losses on the radio interface can be distinguished from losses in the core network or in the public Internet. The packet loss estimation algorithm is rigorously validated using simulations and testbed measurements. Additionally, measurement results and an application example in a dial-up ISP are presented.

## I. INTRODUCTION

Packet loss ratio is one of the most important Key Performance Indicators (KPI) for an Internet Service Provider (ISP). It is especially important if the ISP offers business grade services with performance guarantees. Although most ISPs collect statistics about packet losses from the routers, this information provides only a “local” view. The problem with collecting statistics from the routers is twofold. On the one hand, summing up per-hop counters does not necessarily yield the end-to-end loss. On the other hand, an ISP has control over only a single segment (e.g., over an Autonomous System) of the end-to-end connection, therefore its managing horizon is limited.

TCP, the dominant protocol in the Internet [1], includes a complex adaptive algorithm, which ensures reliable data transfer over unreliable paths. There have been a number of papers published analyzing how the path characteristics (e.g., loss) affect TCP dynamics [2][5][6]. Our paper discusses the inverse problem: by observing TCP dynamics we infer the loss properties of the end-to-end path. An existing tool, Tstat [7] addresses a similar problem, but it measures the number of TCP retransmissions, which is not equal to the loss ratio.

In this paper a method is described, which can be used by ISPs to efficiently monitor the end-to-end loss ratios of individual TCP connections or traffic aggregates. A passive monitor listens to TCP packets on the data flow direction. The algorithm running in the monitor estimates the packet loss ratios separately for the two end-to-end path segments separated by the monitoring point. Since loss estimation is performed on a per-TCP basis, the direction and the location of the problem can be deduced.

The packet loss estimation algorithm can provide the operator with the following valuable information:

- If the end-to-end loss ratio satisfies the service level

agreements,

- or there is a problem on a particular path in the ISP's own domain,
- and/or there is a problem on a particular path in the outside Internet.

The described method is based on heuristics that aim to estimate the most probable sample path of the TCP state-machines in the two communicating end-hosts. We explain these heuristics in detail with examples. Since heuristics always lack a general proof, we explore a large portion of the practically relevant state space by simulations. In addition, the algorithm is evaluated in a testbed. We analyze the precision of the algorithm by performing several thousands of transactions for each configuration. The outcome of the analyses is that the algorithm is capable of estimating the packet loss ratio for both path segments independently of the position of the losses with reasonably high precision. The analyses cover the practical ranges of loss ratios from 0.1% to 10%.

There are several versions of TCP using different congestion control algorithms, which apply different rules to cope with packet losses. A comparative analysis can be found in [3] and in [4]. We found that the precision of the algorithm is acceptable for all TCP versions used in practice. Due to the space limitations, the results of these simulations are not discussed in detail in the paper.

The paper is organized as follows: we sketch a “naive” loss estimation algorithm in Section II. After identifying its limitations, an improved algorithm is presented in Section III, which tries to solve the problems of the “naive” algorithm. A rigorous evaluation of the improved algorithm is presented in Section IV. To demonstrate its feasibility, Section V reports measurement results from an operational ISP network. We conclude the paper in Section VI.

## II. A “NAIVE” LOSS ESTIMATION ALGORITHM

The monitoring point divides the network into two segments

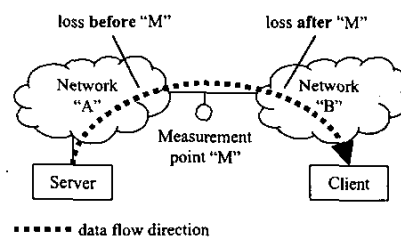


Fig. 1. The measurement scenario

as shown in Fig. 1. The client in network "B" downloads a file from the server in network "A" by opening a TCP connection. The packets flow through the measurement point "M", which lies at the border of network "B" and network "A". The task of "M" is to estimate the end-to-end TCP packet loss ratio between the server and the client by examining the sequence numbers of the packets belonging to the TCP connection. Although observing acknowledgment packets would help estimating losses, we chose not to use them. The reason is that due to asymmetric routing, data packets and acknowledgements often traverse different paths.

In the following, we identify a few basic, simple rules that can be used to detect the losses. First, we classify packet losses by the location of the loss. A packet can be lost either before reaching the measurement point (i.e., in network "A"), or after passing it (in network "B"). Fig. 2 shows a sample TCP sequence number pattern seen at the measurement point, illustrating the characteristic effects of packet losses.

A packet loss before the measurement point can be detected at point "M" by observing an out-of-order TCP segment that "fills a hole" in the data sequence (in Fig. 2, segment  $S_2$  fills the hole caused by the loss of segment  $S_1$ ). Similarly, a repeated sequence number may signal that a packet is lost after passing the measurement point. This can be seen in Fig. 2 where the retransmitted segment  $S_4$  repeats the sequence number of the lost segment  $S_3$ .

Based on these observations, an initial, "naive" algorithm can be constructed that detects packet losses by maintaining a history of sequence numbers seen and comparing whether a packet has been seen or it fills a hole. Although this algorithm is obviously simplistic, the used heuristics guess packet losses correctly in a wide range of practical cases.

In order to evaluate its precision, we implemented the algorithm and tested it on packet traces obtained from simulations. The simulation setup consisted of just three nodes: the Server, the Client, and the Monitor "M". The delay and link rate was set to 100 ms and 64 kbps for both segments respectively. We simulated the NewReno version of TCP [8].

During the simulation, the client opened subsequent TCP connections to download a fixed length file from the server (the maximum window size was set to 64 Kbytes). Packets were dropped independently and with equal probability on both path segments (in Network "A" and in Network "B").

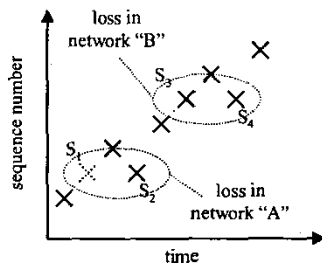


Fig. 2. Detecting packet loss from the sequence number pattern

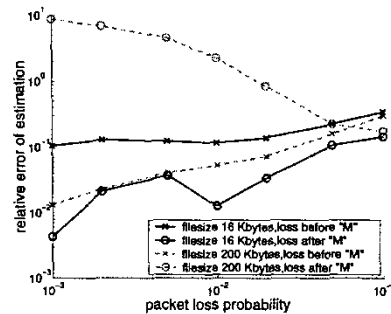


Fig. 3. Simulation results for the "naive" algorithm

The relative error of the packet loss estimation algorithm was calculated and plotted against the loss probability ranging between 0.1% and 10%, see Fig. 3.

The simulation results show that the error of loss ratio estimation after the measurement point was higher for large files. Additionally, in case of estimating packet losses before the measurement point, the algorithm is less accurate for small files. In the extreme worst case, the relative error of the algorithm reached 10, which means that the absolute error was approximately 1%. Although in absolute values this error does not seem to be too high, in many practical Internet configurations, packet losses below 1% should also be possible to monitor.

### III. IMPROVED ALGORITHM FOR TCP PACKET LOSS ESTIMATION

The reason why the previously described simple algorithm makes numerous errors is that packet losses, depending on when and where they happen, may result in a large variety of sequence number patterns not covered by the algorithm. Multiple losses within the same window, different versions (Tahoe, Reno) and implementations of TCP also increase the complexity of the problem.

In the measurement point "M", it would be too difficult - and in most cases even impossible - to take all possible sequence number sample paths into consideration. In the following, instead of pursuing this, we try to identify the scenarios that play an important role for packet loss ratio estimation.

We found that four major effects can explain the inaccuracy of the algorithm:

1. **Spurious timeouts:** In case of a sudden rise in the round-trip time, a TCP sender may timeout even if no packet loss has occurred. The sender then retransmits the segment believed to be lost and waits for its acknowledgement. When the acknowledgement for the *original* segment arrives, all subsequent segments will be retransmitted. The loss estimation algorithm will then incorrectly assume that the repeated segments are retransmissions of lost packets. This effect can explain

the significant error seen in the case of packet loss estimation after the measurement point (see Fig. 3).

2. **Mass losses:** If an entire window of data is lost before reaching the measurement point, the algorithm will be unable to detect it because of the missing out of order segments at point "M". That is, the algorithm will not see the "hole" to be filled and will not recognize the retransmissions. Since this effect is more likely to happen if the window size is small (e.g., just after opening a connection, or after a timeout), it can account for the error experienced for small files (see Fig. 3).
3. **Load balancing:** If per-packet load balancing is applied, the packets traverse different paths thus the monitoring point may not see all packets. The holes are erroneously interpreted as indications of losses before the monitoring point.
4. **Reordering:** Packet reordering may happen if the monitoring point is after the merging point of load balancing paths. In this case, the algorithm erroneously assumes that the holes in the sequence numbers are indications of packet losses before the monitoring point.

Besides the above-mentioned causes and symptoms, there can be additional cases where the simple estimation algorithm fails to detect losses precisely. In order to improve the algorithm, it is essential to realize that we are not interested in determining the exact number of lost packets; we only need to calculate the *loss ratio* instead. The idea is then to choose a subset of packets for which it can be determined, with relatively high probability, whether and where the packets were lost. We call these packets *significant packets*. The estimation is unbiased if packet losses happen independently, that is, the loss ratio of significant packets is equal to the loss ratio of all packets. Although the algorithm is based on a decreased number of packets, the significant packet set is expected to be large enough for estimating the loss ratio.

The set of significant packets is constructed by disregarding packets for which the loss detection would be too uncertain. In our algorithm, a significant packet is the packet that holds a particular data segment for the first time. In other words, retransmissions of a data segment are not considered to be significant. This is because the loss of a first transmission can be detected with quite high probability based on the retransmissions seen at the measurement point, but the loss of the retransmission itself is hard to recognize.

In order to cope with the problem of losing an entire window of data, we also have to exclude the first and the last data segment of a TCP connection when estimating packet loss before the measurement point. The reason for this is that if the first transmission of these segments is lost before "M", they are not seen at all. This solution is expected to increase the precision of the loss estimation before the measurement point.

We improved the algorithm further by making it capable of excluding the effect of spurious timeouts. This is achieved by continuously searching the history for segments that are

believed to be lost in a row. If the number of consecutive segments lost exceeds a pre-defined threshold (e.g., three), the algorithm assumes that a spurious timeout has occurred and removes all the questioned segments from the significant packet set. This solution is expected to increase the precision of the loss estimation after the measurement point.

When the monitoring point detects that it has not seen all the packets of a successfully ended TCP connection, it assumes that load balancing was applied for the flow, and removes the packets from the significant set. This solves problem 3.

In order to deal with the last problem, namely reordering, the packets that have likely been reordered are also removed from the significant set. A reliable detection of reordering can be based on observing the *IP Id* field in the IP header of consecutive packets. Although there are several different ways how operating systems set this field, most of them increase the IP Id value with every packet sent in a TCP connection (Windows 95/98/NT/2000, most Linux, FreeBSD versions, and Solaris as of today). That is, if we observe a decrease, it is likely due to reordering on the path.

#### IV. ANALYSIS OF THE IMPROVED ALGORITHM

In order to check how these improvements influence the precision of the algorithm, we carried out simulations with the same setup as shown in Fig. 1. We were interested in the relative error of the estimation under various parameter sets. We define the relative error as the absolute difference of the estimated loss and the real loss divided by the real loss ratio:

$$error_{relative} = \frac{|loss_{real} - loss_{estimated}|}{loss_{real}} \quad (1)$$

Since the algorithm is based on heuristics, to prove its applicability, a rigorous test has been performed covering a large portion of the practically relevant parameter space.

In Fig. 4 an example of our results can be seen. The first row shows the results for the loss estimation for the path segment before "M", while the second row shows them for after "M". In each column a different version of the estimation algorithm is used. The first is the basic algorithm, described in section II. The second and the third are the improved algorithms without and with spurious timeout detection, respectively. The plots show a heat map demonstrating the dependence of the relative error on the file size and on the packet loss probability. Darker colors stand for higher errors.

It can be seen that the relative error of the basic algorithm is below 20% at the majority of the parameter space. It is worth noting that although the relative error of the loss ratio estimation after "M" can be high at small loss ratios, the absolute error may be still low (i.e., a relative error of 5 stands for a 0.5% absolute loss ratio difference at a loss probability of 0.001).

As we look at the graphs from left to right, we can see that there are characteristic changes and significant improvements

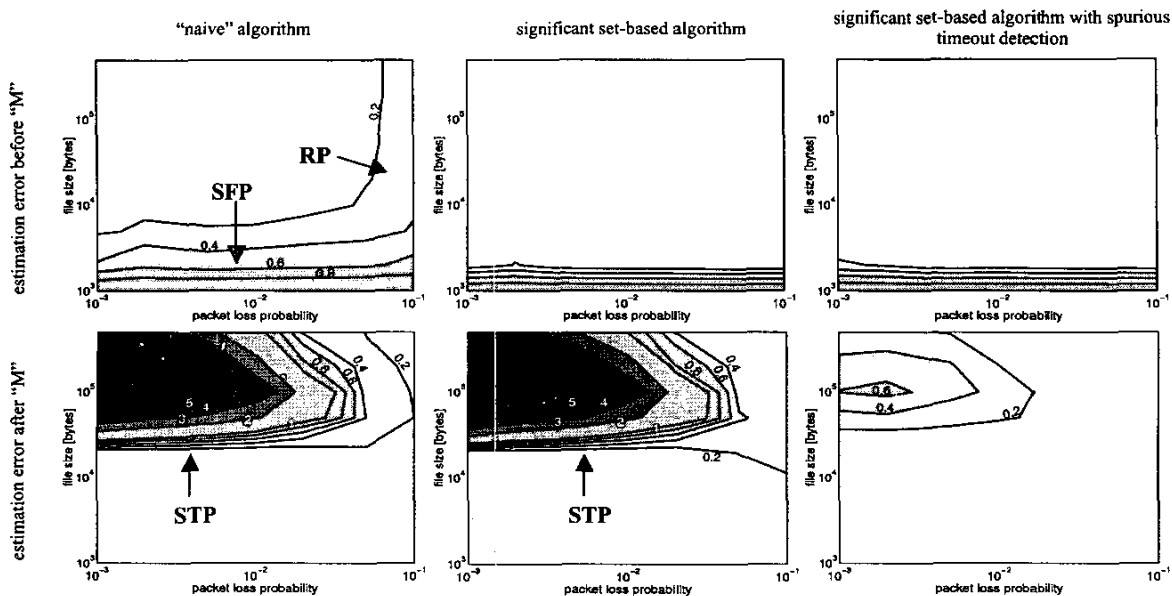


Fig.4. Simulation results with different algorithms

in each step.

Comparing the graphs on the left with the middle graphs, the most important change is that the error of the loss estimation before "M" is almost completely vanished for large loss ratios ("Retransmission Problem", RP area in Fig. 4). This is due to the introduction of the significant set, which improved the estimation when there are considerable amount of multiple or mass losses. We can also note that removing the first and the last segment of a TCP connection from the significant set decreased the estimation error for short files. However, the error for *extremely* short files is still present, because a loss before "M" can be detected only for file sizes exceeding two maximum segment sizes (mss). ("Short File Problem", SFP area in Fig. 4).

The largest error is for the loss ratio estimation after "M" in case of large files and small loss probabilities, for which the left and middle graphs show no improvement. This large error is due to retransmissions caused by spurious timeouts. When the loss ratio is low, there is a higher probability for the congestion window to open up. As a consequence, when a spurious timeout occurs, more packets will be retransmitted, which increases the estimation error ("Spurious Timeout Problem", STP area in Fig. 4). By removing packets from the significant set that would indicate consecutive losses, the precision of the algorithm can be significantly improved. This can be observed on the right bottom graph. An interesting phenomenon is that the error resulting from spurious timeouts decreases for large files. This can be explained by the fact that spurious timeouts tend to occur at the start of a TCP connection. When a new connection is opened, TCP has no information on the round-trip time, therefore its

retransmission timeout (RTO) timer may be calculated incorrectly.

Fig. 5 shows the sensitivity of the algorithm to different levels of asymmetry of the packet loss probabilities before and after "M". The values range between 0.1-10%. For the investigated file sizes 16 Kbytes and 200 Kbytes the estimation error was negligible for the loss ratio before "M". (This is why only the results for the side after "M" are shown.) The loss estimation error is higher if the file size is larger, and when the loss probabilities before and after "M" differ significantly. The largest error is measured in the extreme case when the loss ratio is 10% before "M" and 0.1% after "M".

Further tests have been performed for several TCP versions (Tahoe, NewReno) and in case of correlated losses. For the practically important configurations, the relative error was below 10% for 90% of the loss state space.

## V. TESTBED AND ISP RESULTS

We built a testbed to analyze the algorithm in a controlled, but real environment. The testbed consists of Linux PCs interconnected by a 100 Mbps Ethernet hub. Between the server and the client hosts, the user can configure the link rate, packet delay and random packet loss. We used a modified *shaper* kernel module and the *NIST Net* [9] program to set the desired parameters for the network segments before and after the monitoring host.

The same configuration was tested as we used for the simulations. The loss probabilities before "M" were set gradually from 0.1% to 10%, the loss after "M" was set to

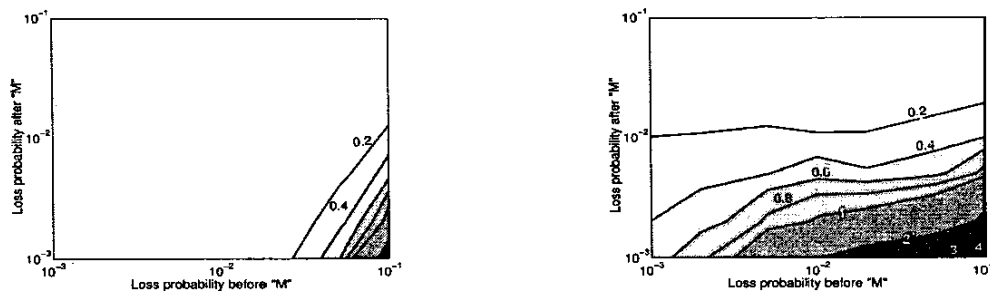


Fig.5. Relative error of the loss estimation for the path segment after "M" for short (left) and large (right) files

0.1% and 10%. The results are summarized in Table 1. The results indicate that the loss ratios are estimated quite precisely in the testbed as well even for very asymmetric cases.

TABLE 1 - RESULTS FROM THE TESTBED (FILE SIZE 100 KBYTES)

Loss before "M"	Loss after "M" 0.1 %		Loss after "M" 10 %	
	Estimation after "M"	Estimation Before "M"	Estimation after "M"	Estimation before "M"
0.1 %	0.07 %	0.10 %	9.53 %	0.10 %
0.2 %	0.12 %	0.26 %	9.95 %	0.23 %
0.5 %	0.11 %	0.45 %	10.05 %	0.59 %
1 %	0.09 %	1.10 %	9.64 %	1.05 %
2 %	0.15 %	1.91 %	9.77 %	1.92 %
5 %	0.13 %	5.17 %	10.14 %	4.86 %
10 %	0.15 %	9.82 %	9.46 %	10.05 %

Finally, we present results from an ISP network. The monitor was placed to the egress point of the ISP network. The loss estimation algorithm was used to study the effect of upgrading the ISP's Internet access link rate. In Fig. 6, the downlink TCP loss ratio before the monitoring point is shown for a period of three days before and after the network upgrade (each value represents a 10 minute average). The high estimated packet loss ratios outside the ISP's domain

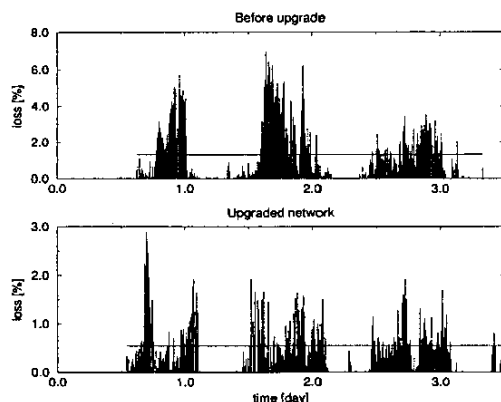


Fig. 6. Loss probability in a dial-up ISP network

indicate the congestion of the ISP's leased line. It can be seen that both the average and the peak loss ratio decreased after the upgrade of this link. The average loss ratio changed from 1.31% to 0.55% whereas the peaks lowered from 5-6% to 2-3%.

## VI. CONCLUSION

We have proposed a passive monitoring based end-to-end TCP packet loss estimation algorithm. We introduced the notion of significant packet set, which decreases the estimation error by only including packets, for which the algorithm is able to determine, with high probability, whether the packet was lost or not. To improve the accuracy, the effects of spurious timeouts, mass losses, load balancing and packet reordering are considered. All-round, rigorous simulations have been carried out that showed that the algorithm is capable of estimating the packet loss ratio with high precision for the practically relevant parameter space. The algorithm has been implemented and tested in a testbed. We have found that the measurement results are consistent with the simulations. Finally, we have demonstrated the applicability of the algorithm with results from an operational ISP network.

## REFERENCES

- [1] J. Postel, "Transmission Control Protocol," RFC 793, September 1981.
- [2] L. Zhang, S. Shenker, and D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," Proc. ACM SIGCOMM '91, pp. 133-147, September 1991.
- [3] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," Computer Communication Review, vol. 26, pp. 5-21, July 1996.
- [4] V. Paxson, "Automated packet trace analysis of TCP implementations," Proc. ACM SIGCOMM '97, pp. 167-179, September 1997.
- [5] J. Mogul, "Observing TCP dynamics in real networks," Proc. ACM SIGCOMM '92, pp. 305-317, August 1992.
- [6] V. Paxson, "End-to-end Internet packet dynamics," Proc. ACM SIGCOMM '97, pp. 139-152, September 1997.
- [7] Tstat TCP Statistic and Analysis Tool <http://verza.polito.it/>
- [8] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, April 1999.
- [9] NIST Net network emulator home page <http://snad.ncsl.nist.gov/itg/nistnet/>