A LOSS PAIR APPROACH TOWARDS CONGESTION MANAGEMENT

B. Stojcevska[1], O. Popov[2]

1) Institute of Informatics, Fac. Nat. Sciences, Sts Cyril & Methodius University
PO Box 162, MK-1000 Skopje, Republic of Macedonia

2) Information and Communication Systems, Dep. Information Technology and Media, Mid Sweden University, SE-851 70, Sundsvall, Sweden
E-mail: biljanas@ii.edu.mk oliver.popov@miun.se

Abstract: Numerous versions of the original TCP are implemented in order to avoid the undesirable behaviour within a network. Their main objectives are to provide a stable system operation, prevent appearance of congestion, and even collapse in extreme cases. The evolution of the physical media has brought up nearly error free transmission in wired networks. Nevertheless, this is not the whole picture, since even high performance wired environments have their own problems such as bursty transmissions, disproportion in performance between the end systems, gateways, and links, and heterogeneous infrastructure. The introduction of wireless settings makes the problem much harder due to the high BER, disconnections, and bandwidth limitations. The loss pair method has been successfully used for probing and determining the status of the network. In the focus of the article is its extension in the area of congestion detection, differentiation, and resolution. Hence, the loss pair method is being employed as a TCP mechanism for dealing with some congestion problems. The main advantage is the preservation of the TCP/IP modularity and the lack of infringement on the end-to-end semantics of TCP.

Keywords: TCP modification, loss pair, congestion management

1 Introduction

Internet is the primary global infrastructure for data communications today. It is based on the TCP/IP protocol stack. One of the reasons behind its pervasiveness is its very robust and scalable architecture. However, recent developments in wireless and mobile communications introduced the need for possible modifications of the Internet protocol stack in order to adapt to the new heterogeneous infrastructure that has different attributes with respect to capacity, delay, jitter, and reliability.

The performance of the dominant number of Internet applications directly relies on TCP [6]. The fundamental assumption built in the design of this protocol is that it operates in a fixed network. TCP congestion control in wireline systems is a combination of several mechanisms: the Additive-Increase-Multiplicative-Decrease (AIMD) [1, 2] procedures that decrease the congestion window for a factor of two whenever there is a packet loss, and makes a cautious increase by one segment per round-trip-time; the timer, which is used as the response of last resort in the case of a packet loss, the slow start infusion in order to bring the system in the right operating mode; the acknowledgment mechanism that is used to trigger new transmissions. All of these mechanisms may work in a satisfactory manner in wireline networks, nevertheless the game changes in heterogeneous ones where large number of system segments might be wireless. Simply, a packet loss can be caused by base station handoffs, physical obstacles, interference, noise, and change in the direction of the signal propagation. Consequently, the protocol suffers severe throughput degradation because it inappropriately triggers mechanisms for error control and congestion avoidance. Therefore, there is a plausible need for modifications in the structure of the TCP protocol for making it suitable in the changed networking environment.

The various proposals for improving TCP performance in heterogeneous networks are usually classified in three categories: link-layer solutions, which try to hide the errors of the wireless medium from the upper layers; split connection, where the TCP connection is divided in two parts: a wired and a wireless; TCP variations which introduce new mechanisms to the standard TCP protocol, without affecting the upper layers [5].

2 TCP congestion control modification by using loss pair method

2.1 The loss pair method

The loss pair method [4] was developed as a tool for probing and determining the status of the network based on end-to-end measurements. It appears that its usage is extensible to TCP congestion control. The round trip times of a TCP connection contain information about the underlying network condition. Our basic idea is to use this information for finding out why a segment is lost in the TCP connection.

A loss pair is a pair of packets that travel close to each other through the same path in the network. When one of them is dropped the round trip time of its pair gives information about the network state when the first packet was dropped.

A loss pair is defined as pair of packets p_1 and p_2 and:

1. p_2 initially follows p_1 with time Δ;
2. exactly one of them is dropped;
3. both packets travel on the same path before one of them is dropped.

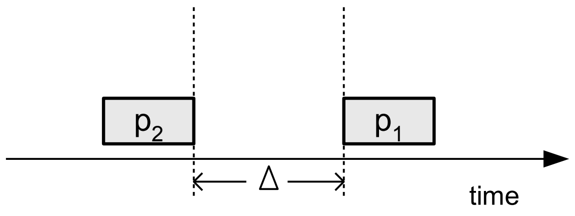

Figure 1: A Loss Pair

A loss pair is formed whenever one of the packets is dropped. It is pointed that it is irrelevant weather p_1 or p_2 is dropped – both cases give the same results. It is assumed that the path in the network and the location of the bottleneck link don't change during the measurement and that most of the losses take place at the bottleneck. These assumptions guarantee that the state of the bottleneck queue was similar when both of the packets passed the bottleneck node. By observing the round trip time of the packet p_2 the reason for dropping p_1 can be found. Namely, it is shown that the round trip times of the loss pairs are tightly grouped around the time necessary to drain the bottleneck queue plus the time of the links traversal.

2.2 The network model

The network path of a TCP connection (Figure 2) can be modelled as a graph with $n+1$ nodes and n links [7]. The nodes $1, 2, \dots, n-1$are routers and the nodes 0 and n are the hosts where the TCP source and destination entities are located.

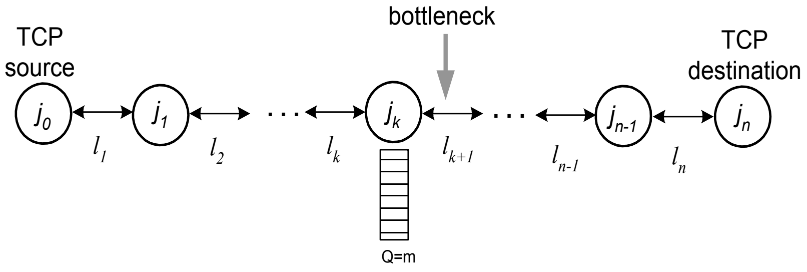

Figure 2: Basic Network Topology

Let all of the TCP segments have the same size and let the acknowledgements travel on the same path as the TCP segments. Let link l_i have capacity C_i where $1 < i \leq n$ and all the queues use FIFO discipline. If $C_k > C_{k+1}$ for k, $1 < k \leq n$, then the link l_{k+1} is a bottleneck and the node j_k can drop packets when it's queue get congested. We assume the following:

P – size of the packets in bits

C_i - l_i capacity

Tq_k(s) – queuing delay for packet s

Td_i – delay at l_i

Tp_i processing time to forward P bits at l_i

Link l_{k+1} is a bottleneck

The value of the round trip time RTT for a segment s can be calculated as:

$$RTT_s = 2 \times \sum_{i=1}^{n} \left(\frac{P}{C_i} + Td_i + Tp_i + Tq_i^{(s)} \right) \qquad (1)$$

The values of Td_i and Tp_i are constant provided there are no changes in the network topology during the connection. Tq_i depend on the number of packets in the queues.

If all queues have infinite capacity, except the queue at j_k, the following holds:

$$RTT_s = 2 \times \sum_{i=1}^{n} \left(\frac{P}{C_i} + Td_i + Tp_i \right) + Tq_k^{(s)} \qquad (2)$$

since $Tq_i(s)=0$ for $i \neq k$.

2.3 Applying the loss pair method

A reception of a duplicate acknowledgement in a TCP connection means that an out of order segment arrived at the destination. We assume that in all cases the reason is a lost segment. The duplicate acknowledgements can be treated as loss pairs. When the sequence number of an acknowledgement is s that means that the segment with sequence number s is lost. The segment $s+1$ that generated the duplicate acknowledgement and the lost segment s form a loss pair where $p_1=s$ and $p_2=s+1$.

The round trip time of the duplicate acknowledgement contains considerable information. The increase of the round trip times is an indication that the bottleneck queue is congested and can be used as an instrument for determining the reason for the packet loss. Intuitively we can conclude that large round trip times would follow congestion losses and in this case activating a congestion mechanism is appropriate. In the opposite case, if a packet loss is followed with small round trip times, we can conclude that the bottleneck queue was not congested when the packet was lost and that the packet was lost on some of the other links on the path.

2.4 The reason behind the loss

The goal of the TCP variations is to find a way to determine the nature of the errors that caused packet losses. If congestion on the path is concluded, the protocol should apply standard congestion control algorithms. If the reason for the loss is found to be a transmission error, then the sender should keep the current sending rate.

Our protocol belongs to the TCP variations group. It observes the sequences of packets that precede and follow a lost packet. If *RTT* values are between the smallest measured *RTT*, *RTTmin*, and the maximum threshold *RTTmax*, then it is very likely to have congestion at the bottleneck. The main problem is: how to define *RTTmax*.

The smallest *RTT* can be expressed as:

$$RTT_{\min} = 2 \times \sum_{i=1}^{n} \left(\frac{P}{C_i} + Td_i + Tp_i \right) \qquad (3)$$

when there is no queuing delay and $Tq_k(s)=0$.

The changes of *RTTs* depend on the changes of Tq_k(s). We have:

$$\overline{RTT} = 2 \times \sum_{i=1}^{n} \left(\frac{P}{C_i} + Td_i + Tp_i \right) + \overline{Tq_k} \qquad (4)$$

where $\overline{RTT} = \frac{\sum_{s=1}^{M} RTT_s}{M}$ is the average value of all measured round trip times RTT_s

and $\overline{Tq_k} = \frac{\sum_{s=1}^{M} Tq_k^{(s)}}{M}$ is the average queuing delay at j_k for M measured values of RTT_s.

If $\Delta_{RTT} = RTT_s - \overline{RTT}$ *(5)*

then: $\Delta_{RTT} = Tq_k^{(s)} - \overline{Tq_k}$ *(6)*

We can conclude that if $\Delta RTT<0$ then $Tq_k(s) < \overline{Tq_k}$, which means that *there was no congestion at the bottleneck when packet s was at the bottleneck* and the congestion window should not be decreased. Finally, we can use $\overline{RTT}$ as the maximum *RTT* threshold *RTTmax*.

3 Experimental results

The previous reasoning was confirmed experimentally, by using the ns [8] network simulator. The modification was implanted in the TCP Reno [3] implementation. We present its performance by using the network given in Figure 3.

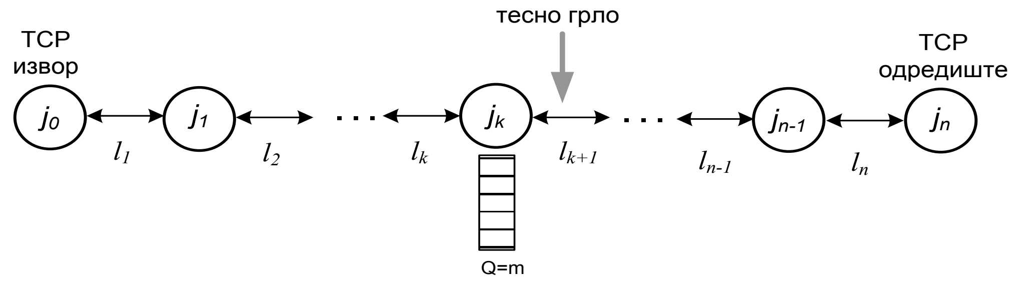

Figure 3: Topology for the Simulation Model

The link r_3-r_4 has a loss module that starts dropping packets at rate 1% at time 10.0 sec. The TCP source starts connection at time 0.0. The simulation lasts 30.0 sec. Table 1 contains the results from the simulation that show significant throughput increase. In Figure 4 we can see a comparison between Reno and modified protocol packet sending during the simulation.

Protocol	No. of sent bytes	Throughput (bits/sec)
Reno	2669000	683612.04
Modification	3187000	814983.28
	Improvement:	131370.24 (19.2%)

Table 1: Reno vs. the TCP modification

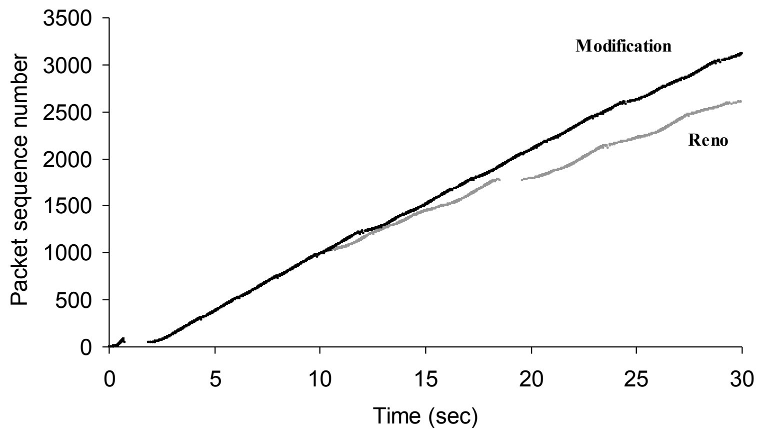

Figure 4: A Comparison of Packet Sending

4 Conclusions

The modification has been tested on various network topologies with lossy links. The results show an increase of the protocol throughput compared to Reno in across the board. The improvements range from 10% to 70%, which indicates a considrable perfromance gain. Moreover, the "new" protocol belongs to the group of TCP variations, hence it does not violate the end-to-end semantics of TCP, preserves modularity, and layer transparency to other layers. The modification has performed well in wireline networks also, which underlines the original TCP/IP philosophy of tranmission infrastructure indepedence.

5 References

1. Chiu, D. M., Jain, R. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Computer Networks and ISDN Systems*, Vol. 17, pp. 1-14, 1991.

2. Jacobson, V., “Congestion Avoidance and Control”, *Proc. of the ACM SIGCOMM Symposium on Communications Architectures and Protocol*s, pp. 314–329, Stanford, USA, pp. 14-17 August 1988.
3. Jacobson, V., “Modified TCP Congestion Avoidance Algorithm”, Technical report, 30 April 1990.
4. Liu, J., Crovella, M., “ Using loss pairs to discover network properties”, *ACM SIGCOMM Internet Measurment Workshop*, San Francisco, CA, November 2001.
5. Popov, O. B., Stojcevska, B., “TCP in Heterogeneous Environments”, *Proc. of the Third CiiT*, Bitola, Macedonia, December 2002.
6. Postel, J., “Transmission Control Protocol”, *RFC 793*, September 1981.
7. Stojcevska, B., “Improving TCP performance in Heterogeneous Environments”, MSc Thesis, Skopje, Macedonia, June 2003.
8. The Network Simulator - ns-2, the VINT project, http://www.isi.edu/nsnam/ns/