ANALYSIS OF EFFECTS OF NETWORK LATENCY ON MULTIMEDIA DATA TRANSFER IN BROADBAND COMMUNICATIONS NETWORKS

Daniel Z. Lenardic [1], Branka Zovko-Cihlar [2], Mislav Grgic [2]

[1] KPMG Croatia, Kaptol Centar, Nova Ves 11, Zagreb, CROATIA, daniel.lenardic@kpmg.hr
[2] University of Zagreb, FER, Unska 3 / XII, Zagreb, CROATIA
[2] University of Zagreb, FER, Unska 3 / XII, Zagreb, CROATIA

Abstract: *This paper describes the issues relating to the analysis of the effect of network latency on TCP/IP protocol behavior in a multimedia computer network.*

Keywords: *TCP/IP, analysis, latency, multimedia, computer, network*

1. INTRODUCTION

TCP/IP family of network protocols was designed decades ago, in the early days of Internet. At the time, the main concern of building a network protocol was simplicity (to be easily implemented in existing computer systems), flexibility (so it could be adapted to various communications technologies and media) and resiliency (required to be able to support communications over unreliable, slow communications links that were the norm at the time).

Although never designed to do so, TCP/IP protocols are today used to provide multi-gigabit high-speed communications over fiber optics links, wireless media, satellites and a multitude of other communications technologies and media. On the other hand, TCP/IP is also used to transfer a variety of real-time multimedia data streams and applications, like voice, video, telephony and similar, for which it was also not very well prepared at its outset.

Even more fundamental are the requirements that these new transmission technologies and data streams put forth. Quality of Service, like guaranteed bandwidth, maximum loss or delay, parameters, assignment of priorities and allocation/reallocation of network resources to various data streams, and various other new requirements, have requested completely new sets of functionality to the basic TCP/IP.

In this article, we will describe several fundamental issues related to the research and analysis of how TCP/IP protocols behave in the modern, multimedia and data flow saturated, network environment, with specific analysis of the influence of network latency on the TCP/IP protocol operation.

2. INTRODUCTION TO TCP PROTOCOL OPERATION

The introduction to how TCP protocol operates can be summarized with two illustrations of the design idea behind it. The first one is self-clocking, which means that TCP/IP protocols do not require an external synchronization source. While that is not absolutely true for any TCP/IP communication (as some underlying communications

technologies on which TCP/IP can be used are synchronous in nature), TCP/IP provides clocking and flow control functions through a feature appropriately referred to as the "TCP/IP sliding window". The second important feature of TCP/IP, related to network congestion control, is termed as "additive increase, multiplicative decrease", and concerns the back-off algorithm TCP/IP uses when congestion on the network is detected.

3. "TCP/IP SLIDING WINDOW" BASED SELF CLOCKING

The "sliding window" operation of TCP/IP protocol provides the flow control mechanism in TCP/IP networks. Clocking in TCP protocol is achieved through the use of packet acknowledgements (ACK packets), which means that, in principle, for each data packet that the sending TCP station transmits over the network, the receiving TCP station should send back an ACK acknowledgement packet. If we, for the sake of this exercise, assume that the initial data packet (which carries useful data and is thus usually of much larger size than the simple ACK packet) and the ACK packet take exactly the same time (T) to traverse the communications network from the transmitting station to the receiving station and back, the total time needed before the next data packet could be sent to the network is 2*T.

Thus, it is clear that the TCP network operation in this case would be very inefficient, as this form of flow control would necessitate long periods of inactivity on the transmitting end before the next data packet is sent to the network. Thus the concept of "sliding window" has been established.

In this case, the transmitting station and the receiving station go through the following process at the initiation of communication:

1 The transmitting and receiving stations negotiate initial communication setup. In particular, the receiving station determines the maximum batch of packets it is prepared to receive before sending out an ACK packet (the "maximum window size").

2 The transmitting station starts the communication, by sending out one data packet and waiting for its acknowledgement. Upon receiving the ACK packet, the transmitting station increases the size of the batch in which it sends the data packets, for example by one, so it now transmits two data packets in sequence (thus, the "window size" parameter is increased).

3 After the receiving station receives the first of the two data packets in sequence, it can detect the new "window size" parameter and refrain from sending the ACK packet before both packets in are received. Then, one ACK packet is sent back, informing the transmitter that both packets arrived at the destination.

4 Further on, as long as all packet batches are timely "group acknowledged" by the receiver, and while "window size" remains smaller or equal to the "maximum window size" allowed by the receiver, the process of increasing the "window size" continues. The size of increments in which the transmitter increases the "window size" is appropriately termed as the aggressiveness of the transmitting TCP/IP protocol stack implementation.

5 Ideally, once the "maximum window size" has been reached, the TCP communication will remain at this, most efficient mode of operation, as the maximum connection bandwidth that the receiver can support is reached.

6 If, due to congestion, or unreliable connection, there are lost packets in a batch sequence, the TCP back-off algorithm will be initiated, and the "window size", and thus the used bandwidth will decrease.

4. ADDITIVE INCREASE, MULTIPLICATIVE DECREASE

As we explained in the previous section, the flow control operation of TCP/IP provides the general framework for gradually determining the appropriate connection bandwidth and rate of transmitting data packets. The communication in the described case is one-way only, but the example can be duplicated to provide for the two-way transfer.

If the connection experiences network congestion or packet loss, the TCP back-off algorithm is initiated. Depending on various standardized TCP protocol implementations (most popular are UC Berkeley developed, BSD Unix based, TCP Tahoe, TCP Reno, TCP Vegas...), the following back-off behavior can be experienced:

1 If the receiving station misses a packet in the expected sequence, it will wait for a predetermined timeout period, and then send a partial ACK packet.

2 When the transmitting station receives a partial ACK packet, or the timeout period for receiving an ACK expires, it will either drop the "window size" back to one packet, cut the "window size" in half, decrease the "window size" in some increment, or resend the missing packet in sequence (quick retransmit behavior). The particular behavior depends on TCP protocol implementation and is not specifically standardized (that is, two different implementations can interoperate).

3 If the congestion, or packet loss, continues, the transmitting station will continue to decrease the "windows size". Once the network becomes responsive and reliable again, the process of increasing the "window size" will recommence.

5. IMPACT OF NETWORK LATENCY

Explained flow control and congestion detection mechanisms of TCP present a problem for multimedia communications, as it is common (especially for aggressive TCP implementations) that the connection is constantly increasing bandwidth until it reaches congestion, upon which the bandwidth is rapidly decreased, and then the process is reiterated. Example of bandwidth variation and network congestion can be seen in the following example – two data flows using the same data transfer algorithm (TCP Reno in the example shown on Fig. 1 and Fig. 2) with the same network delay and transmission initiation time will increase bandwidth until they consume about one half of the bandwidth each, and would therefore reach a stable network state (as shown in Fig. 1).

In case that one of the data flows has greater latency than the other, the data flow with lower latency will increase its windows size, and thus bandwidth consumption, at a much

higher rate, and would therefore penalize the greater latency data flow. Also, as the total available network bandwidth is consumed, both TCP Reno connections continue to fight for the bandwidth, continually pushing the network in and out of congestion:

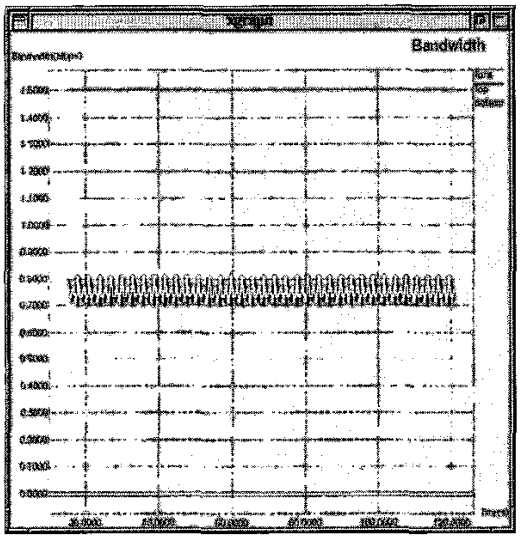

Fig. 1. TCP Reno with the same delay

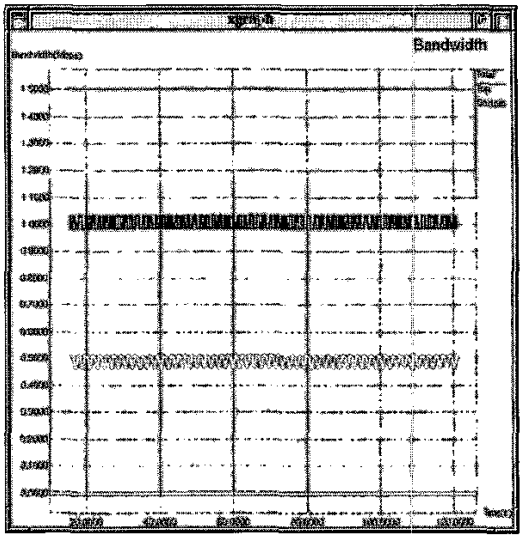

Fig. 2. TCP Reno with different delays

This behaviour of TCP Reno testifies that TCP Reno exhibits considerable unfairness towards longer delay connections in the network. The impact of such behavior on multimedia communications can be extremely disruptive and result in very poor quality of video or audio signals on a network that, otherwise, has quite sufficient capability (in terms of bandwidth, delay and delay variation) to support it. The solution to these problems can be sought on two fronts:

1 By introducing connection control protocols, which will provide in-band (one example is *diff-serv* protocol) or out-of-band (example is RSVP or RTP/RTCP protocols) management of a multimedia connection. This would include setting of traffic priorities, negotiating quality of service parameters etc.

2 The other option is researching better algorithms for congestion detection and control within the TCP protocol implementations. As the general framework of protocol operation enables us to implement various algorithms for behavior in state of congestion, it is possible to implement new, adaptive behavior algorithms to improve the efficiency and intelligence of TCP operation. Examples of work in these areas include TCP SACK, RED, ECN and other protocols.

Our research interest in this area is the second described approach, that is, specification of requirements of and possible development of an intelligent TCP algorithm, which would include adaptive behavior and intelligent detection of the network state (defined through parameters such as bandwidth, delay, delay variation, congestion etc), and thus quickly reach and maintain the optimal "window size" in a particular TCP/IP connection. For that purpose, we are researching various TCP protocol implementations and their behavior in

various network topologies and levels of congestion, with the use of NS-2 network modeling and simulation software package.

As the network is usually very dynamic, rapidly changing its state through creation and dissolution of a large number of various flows of traffic, the optimal window size, and therefore the bandwidth capacity consumed by a single traffic flow, is continuously changing for non-guaranteed traffic flows. As the detection of the change in network state usually occurs with a significant delay with regard to the actual time it occurred, traffic flows can detect the change in network state too late (not executing the TCP backoff algorithm in time to prevent network congestion), and thus drive the network into congestion, which usually indiscriminately affects all traffic flows on a given network segment and thus decreases overall network performance.

6. FAIRNESS BETWEEN CONNECTIONS WITH DIFFERENT DELAYS

If we continue this exploration of the behaviour of various TCP/IP data transfer algorithms in the environment of contemporary multimedia data flows with different latencies, we can observe the following results:

Both TCP Reno and TCP SACK show considerable bias towards lower latency data flows, with the difference between overall bandwidth consumption control in favour of the TCP SACK (compare Fig. 2 showing the instability of overall network bandwidth consumption for TCP Reno with Fig. 4 showing the stable overall consumption for TCP SACK, which does not push the network into congestion although the lower and higher latency data flows continuously fight for bandwidth which lower latency flow monopolises similarly as TCP Reno does);

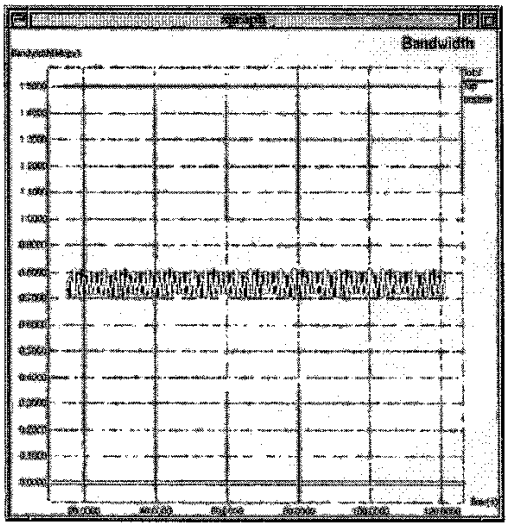

Fig. 3. TCP SACK with the same delay

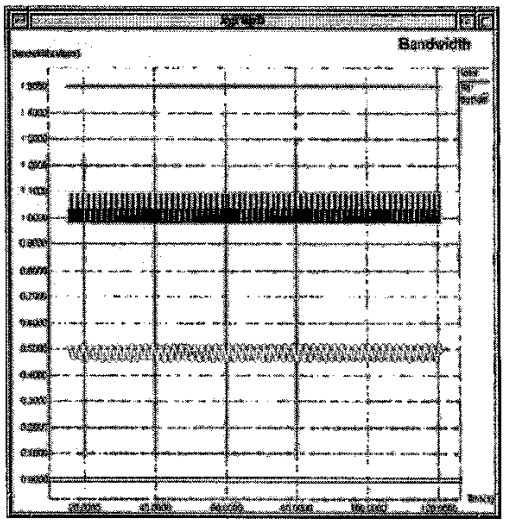

Fig. 4. TCP SACK with different delays

TCP Vegas exhibits much improved fairness between data flows with different delays, in case that both data flows use the TCP Vegas data transfer algorithm.

In that situation, as we can observe from Fig. 5 and Fig. 6, both the fairness of bandwidth allocation between the data flows and the stability of overall network bandwidth consumption is preserved for data flows with different latencies. Although for data flows with large differences in latency, there is some preference in bandwidth allocation toward the lower latency data flow, this preference results only in incremental difference in the allocation of bandwidth, as can be observed in Fig. 6.

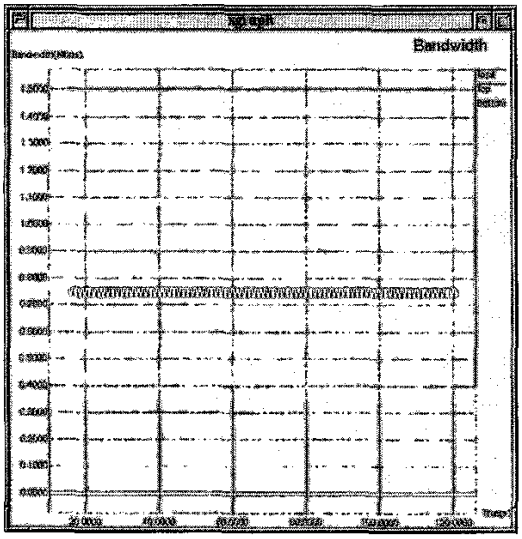

Fig. 5. TCP Vegas with the same delay

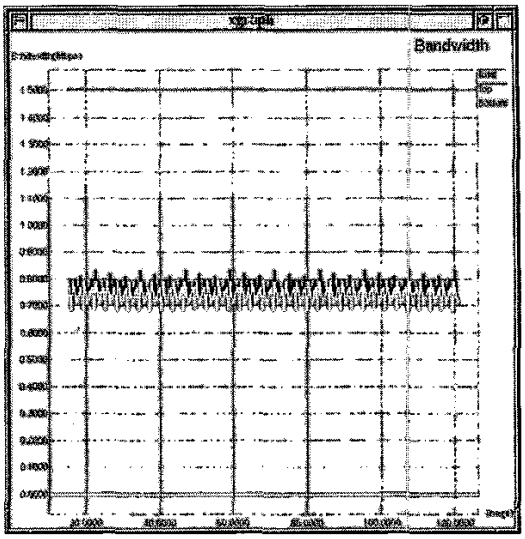

Fig. 6. TCP Vegas with different delays

The reason for this fairness of TCP Vegas towards the flows with different latencies lies in the mechanism of TCP Vegas in increasing its window size, and therefore network bandwidth consumed. While other TCP data transfer algorithms base their windows size increases on simple AIMD algorithms, based on the measurement of packet receipt acknowledgements, and continuously push the network into congestion, Vegas approaches the bandwidth increases much more cautiously and in a more complex manner.

7. CONCLUSION

From the presented analysis, we can conclude that both TCP Reno and TCP SACK exhibit significant bias towards low latency data flows, significantly penalising higher latency data flows with lower achievable throughput. In addition, TCP Reno also presents unstable overall network throughput consumption, with continuous fluctuation of the network in and out of congested state.

While TCP Vegas presents a high level of fairness in throughput utilisation of lower and higher latency data flows, it still exhibits serious degradation when Vegas and non-Vegas TCP implementations are present in the same environment. In that case, non-Vegas TCP

implementations, due to their higher aggressiveness in increasing network bandwidth utilisation, consume disproportionately higher amount of bandwidth than TCP Vegas, thus mostly eliminating the fairness advantages of the TCP Vegas implementation.

This behaviour is expected due to understanding of the AIMD algorithm governing the increase in throughput sent to the network by the TCP protocol at the network end-stations. Modifications and improvements to the AIMD algorithm used in various TCP protocol implementations are responsible for the behaviours of these implementations in the environment analysed. In particular, mostly the dependence on measuring the return trip time (RTT) of network packets as the main time parameter for increasing the sending network bandwidth in Reno and SACK implementations is responsible for favouring of lower latency, and thus lower RTT time, connections.

Mechanisms for improving the described behaviour in the environments with numerous data flows and highly varying latency levels (environment that is definitely exhibited by the global Internet) are the subject of our further research. Specifically interesting are external mechanisms, such as router queue management algorithms, which do not require changes in the TCP implementations in the end user computers.

As we do not envision that latency-friendly TCP implementations such as TCP Vegas are about to be universally adopted in all equipment in the foreseeable future, and given that Vegas implementations do not exhibit expected improvements when mixed with previous TCP implementations, we believe that these external mechanisms are the only feasible option at present.

REFERENCES

[1] D. Lenardic, B. Zovko-Cihlar, M. Grgic, *Analysis of Network Buffering Effects on TCP/IP Protocol Behavior*, COST-279 Meeting, Dubrovnik, 23-24 January 2003, TD-03-014

[2] D. Lenardic, B. Zovko-Cihlar, M. Grgic, *Analysis of TCP/IP Protocol Behavior in a Congested Multimedia Computer Network*, Proceedings of the 9th International Workshop on Systems, Signals and Image Processing, IWSSIP 2002, Manchester, United Kingdom, 07-08 November 2002, pp. 59-65

[3] D. Lenardic, B. Zovko-Cihlar, *Modeling and Simulation of Modern Computer Networks*, Proceedings of the 8th International Workshop on Systems, Signals and Image Processing, IWSSIP 2001, Bucharest, Romania

[4] Low, Peterson and Wang, *Understanding TCP Vegas – a Duality Model*, Caltech, 2000

[5] Kevin Fall, Kannan Varadhan et al., *The ns Manual*, UC Berkeley, 2000