PROCEEDINGS OF SCIENCE

TCPDelay: Constant bit-rate data transfer over TCP

Stephen Kershaw*
The University of Manchester
E-mail: Stephen.Kershaw@manchester.ac.uk

Richard Hughes-Jones
The University of Manchester
E-mail: R.Hughes-Jones@manchester.ac.uk

Transmission Control Protocol (TCP) is a reliable transport protocol which guarantees that data sent will be perfectly replicated at the receiver. In order to deliver on this guarantee, TCP transmits data according to well defined rules which create reliable transfers and attempt to ensure that our traffic can fairly co-exist with other data flows. However, this can result in delayed data transmission and highly variable throughput. We investigate the use of TCP for the transfer of real-time constant bit-rate data and report on the effects of the protocol on the flow of data. We discuss the requirements for TCP to be successfully applied in this situation and the implications for applications such as e-VLBI, where we are more concerned with timely arrival of data than guaranteed delivery.

Experiments show that for a lossy TCP connection using standard bandwidth-delay sized buffers the packet arrival times for a constant bit-rate flow diverge away from real-time arrival. Increasing the buffer sizes, by orders of magnitude in some situations, allows timely arrival of data with only temporary, though possibly lengthy, delays.

Lighting the Blue Touchpaper for UK e-Science - Closing Conference of ESLEA Project
March 26-28, 2007
Edinburgh

*Speaker.

PoS(ESLEA)002

1. Introduction

Transmission Control Protocol (TCP) is the most widely used transport protocol on the Internet, due in the most part to its reliable transfer of data. However, it is not ideal to use for constant bit-rate applications because TCP throughput can vary wildly in a lossy environment. Many applications using constant bit-rate data transfer desire timely arrival of data but the rate fluctuations of TCP mean that timely arrival of data is not guaranteed. We examine the effect of packet loss on packet arrival times and investigate whether packet loss and the consequent effect on throughput delays the data irrecoverably. The performance of TCP from the perspective of data arrival time will determine the suitability for real-time applications, such as e-VLBI.

Electronic Very Long Baseline Interferometry (e-VLBI) is a technique used for high-resolution observations in radio astronomy which involves the transmission of constant bit-rate data streams which are generated in real-time. Timely arrival of data is a fundamental requirement of e-VLBI and data are often transmitted using TCP, hence tests were conducted using constant bit-rate flows at rates of up to 512 Mbit/s to be representative of e-VLBI observations.

2. Transmission Control Protocol

2.1 Properties of TCP

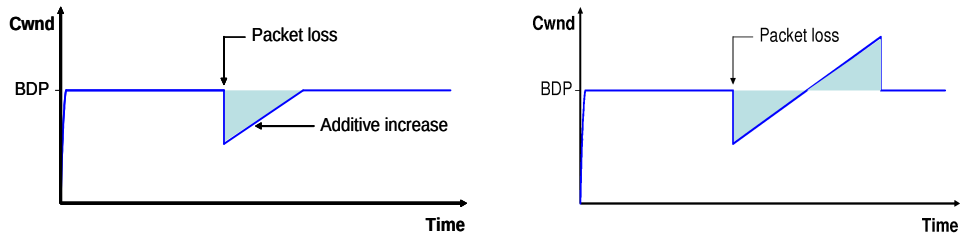
TCP is connection-oriented and reliable, ensuring that data sent will be perfectly replicated at the receiver, uncorrupted and in the byte-order sent. From the perspective of the application TCP ensures that the byte stream sent is the same as the byte stream received. Data corruption is detected by checksums and the receipt of all data (reliability) is ensured by using automatic repeat-request (ARQ), whereby the receiving system sends messages (ACKs) back to the sending system to acknowledge the arrival of data and hence indicate the missing data to be retransmitted. TCP assumes lost data packets are due to network congestion and attempts to mitigate congestion by varying the transmit rate - a process known as *congestion avoidance*, of great importance and described in more detail later.

2.2 High performance TCP

To make effective use of TCP, especially with high-capacity networks, it is often necessary to tune certain parameters. The end-hosts maintain windows over the data and to use the full capacity of a link the windows must be sized to the *bandwidth-delay product* (BDP) to allow sufficient "in-flight" unacknowledged segments [1]. In this investigation, a desired constant bit rate *CBR* was considered, where bandwidth delay product, *BDP*, is expressed as:

$$BDP = CBR \cdot RTT \tag{2.1}$$

where RTT = round-trip time. With windows sized to the BDP, steady line-rate throughput is achievable if we have no packet losses and so this practice is a generally recommended step to tune a TCP connection. In the event of packet loss the size of one window, the congestion window, on the sending host is adjusted to limit the maximum instantaneous throughput.



(a) BDP buffer size (b) Expanded buffer size

Figure 1: Theoretical action of TCP congestion avoidance with CBR data transfer. Comparing (a) and (b) we see the effect of dramatically increased buffer size. The shaded area of delayed data in (a) is compensated for in (b), transmitted faster than the constant bit-rate, having been buffered on the sending host

2.3 Reaction to loss

When TCP detects a lost packet it is assumed that the loss was due to network congestion and TCP enters a congestion avoidance phase, altering the achievable transmit rate dramatically by adjusting the congestion window. This feature of TCP was implemented to prevent congestion collapse of the Internet where competing flows reduce the useful throughput to zero. It is the congestion avoidance behaviour of TCP that creates problems for constant bit-rate flows.

The standard NewReno response to congestion is a decrease of the congestion window by a factor of 2, followed by an additive increase of 1 packet per round-trip time. This gives the throughput a characteristic sawtooth shape when a packet loss is detected - a sudden dramatic reduction of the congestion window, followed by a gradual linear increase. Considering this sawtooth congestion avoidance response, as shown in Figure 1, the amount of data that is delayed can be calculated.

2.4 Delayed data

When a packet is lost, the equations of TCP congestion avoidance determine both the decrease of the congestion window and the rate of increase. If we consider a pre-loss thoughput of *CBR*, it can be calculated that the time taken to regain *CBR* throughput is given by:

$$t_{recovery} = \frac{CBR \cdot RTT^2}{2MSS} \tag{2.2}$$

where *MSS* is the maximum segment size, the maximum amount of data that TCP can encapsulate in one packet.

The shaded triangular area in Figure 1(a), whose presence is due to a packet loss, has area proportional to the amount of data that has been delayed. The area is proportional to the recovery time and can be represented simply as:

$$\frac{CBR^2 \cdot RTT^2}{8MSS} \tag{2.3}$$

For applications like e-VLBI, where data are transferred over large distances at high rates it is essential to note from Equation 2.3 that the delayed data scales with the square of the throughput and the square of the round-trip time.

3. Constant bit-rate data over TCP

It is often said in the literature that TCP is largely unsuitable for real-time applications and constant bit-rate flows because of the variable rate of TCP over a lossy connection due to congestion avoidance [2],[3],[4].

If CBR data is streamed over TCP, as with some multimedia applications or e-VLBI, the reduced throughput due to packet loss leads to a data arrival rate on the receiver of less than the original constant bit-rate. If the processing or playback at the application level is not to stall then sufficient data must be stored in a playout buffer to compensate for the lower data-rate at the transport level, allowing CBR data arrival at the application level. This is common practice for streaming multimedia applications, requiring an initial buffering period and hence a delay between the start of the transfer and the start of the playback.

The situation of bulk data transfer is quite well researched and understood,[5],[6], in contrast to the equivalent situation but where the CBR data is generated and transferred in real-time. When a CBR data stream is generated in real-time and cannot be stalled then we must transfer the data at a steady CBR else we have to either discard or buffer the data at the sending end.

3.1 Regaining timely arrival

If we temporarily store the delayed data on the sending host and can subsequently transfer it faster than the constant bit-rate then we should be able to regain timely arrival of data at the receiving host. We require the data to be buffered and the maximum window size must permit transfers at a rate higher than the CBR. In the investigation that follows, both functions are performed using the socket buffers in Linux.

Data from a Linux application, destined for a network, is buffered in a socket buffer, the size of which we can specify through kernel and application parameters. The socket buffer in Linux serves two purposes: to retain data for windowing and also as an application buffer, designed to isolate the network from effects of the host system, such as scheduling latency of the Linux kernel. Therefore, on the sending host, the socket buffers which are an integral part of TCP/IP in the Linux kernel can be used to buffer the data that is delayed in the event of a loss.

4. Experimental configuration

4.1 Network setup

The network links used to test constant bit-rate performance over TCP were dedicated fibre optic lightpaths with connections to UKLight in the UK peering with NetherLight in the Netherlands. The links were tested to have a very low bit-error rate, allowing loss-free data transfers with stable delay and jitter characteristics, making for an ideal protocol testing configuration. The lightpaths were used to connect to hosts in Manchester, Jodrell Bank Observatory and JIVE (Joint Institute for VLBI in Europe, Dwingeloo, Netherlands) with dedicated point-to-point 1 Gbit/s connections. From Manchester the round-trip times (RTT) were 1 ms and 15 ms for Jodrell and JIVE respectively, with a connection from Manchester to Jodrell, looped back in the Netherlands giving a RTT of 27 ms.

The computers used as end-hosts were server-quality SuperMicro machines, with all configurations tested to give 1 Gbit/s throughput using UDP/IP or TCP/IP over Gigabit Ethernet interfaces. The systems used Intel Xeon CPUs and were running Red Hat or Fedora distributions of Linux. Test were performed using kernel versions 2.4.20 and 2.6.19 with negligible difference in TCP performance seen between kernel versions. All systems were equipped with onboard Intel e1000 Gigabit Ethernet ports.

4.2 Diagnostic software

TCPdelay [7] is an application written by Richard Hughes-Jones, used to conduct tests using memory-to-memory TCP streams, sending data to a socket at regular intervals so as to attain a specified average data rate, emulating a CBR data stream. TCPdelay measures timings of packet sending and arrival at the application level, allowing measurement of whether the data stream is arriving at the receiver in a timely manner.

In order to gain more insight into the behaviour of TCP the *web100* kernel patch was used. Web100 [8] is a kernel patch which provides extended TCP instrumentation, allowing access to number of useful TCP related kernel variables, such as the instantaneous value of the congestion window.

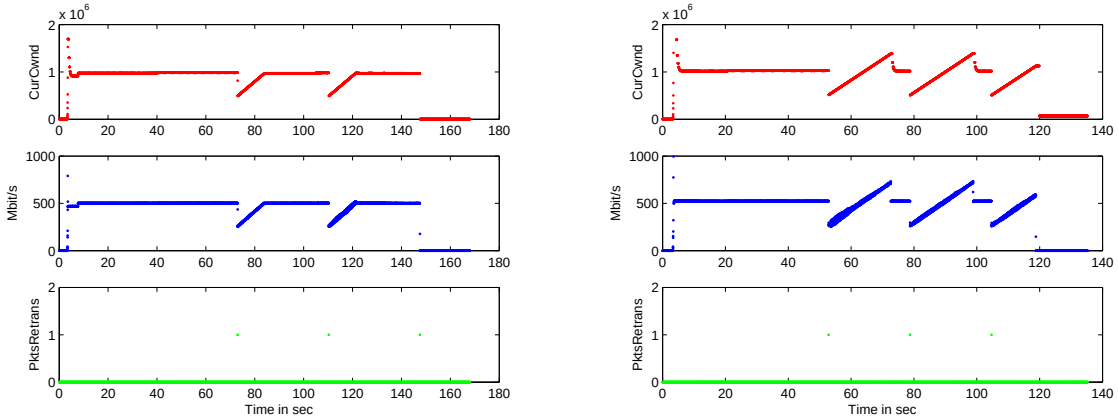
Packet loss on the test networks is rare, so we simulated packet loss in the receiving hosts using a Linux kernel patch to discard packets at a configurable, regular rate.

5. Results

Using the often recommended socket bandwidth-delay product buffer size, the behaviour of a 512 Mbit/s CBR TCP stream over a lossy 15 ms connection was studied with 0.9 Mbyte (BDP) socket buffers. The observed behaviour is shown in Figure 2(a). In the event of a lost packet (deliberately dropped on the receiving host) we see the reliable TCP protocol retransmitting a packet (lowest plot) and we see the expected congestion window evolution, as detailed earlier and illustrated in Figure 1(a). The rapid decrease and additive increase of the congestion window is apparent, with recovery of the constant bit-rate transfer taking around 10 seconds. We see an amount of delayed data of around 160 Mbyte, in agreement with Equation 2.3 when delayed acknowledgements are accounted for.

Data is further delayed with every subsequent packet lost, the cumulative effect of multiple losses shown in Figure 3, which demonstrates the effect of loss rate on message arrival time. The lowest curve in Figure 3 shows the observed timely arrival of data, with higher curves showing lossy transfers diverging rapidly away from this ideal. As one may expect, with the throughput dipping below the desired rate many times and never exceeding it, the amount of delayed data increases and the data arrives later as the duration of the transfer increases.

Figure 1(b) shows the same network configuration of the test in Figure 1(a) but with increased socket buffers of 160 Mbytes, which is the calculated amount of delayed data. As explained previously, the intention was that the delayed data was stored in the TCP socket buffer, to eventually be transmitted at a rate in excess of the constant bit-rate. We see in Figure 1(b) that we initially have the same post-loss behaviour as (a) but the buffered data means that we can transmit faster as the transfer from the buffer memory is not limited to the constant bit-rate. Once the buffered data has



(a) 0.9 Mbyte (BDP) buffer

(b) 160 Mbyte (180x BDP) buffer

Figure 2: Plots of TCP parameters, logged using web100. Kernel patch used to drop packets.
Top: TCP Congestion window (bytes)
Middle: Achieved throughput (Mbit/s)
Bottom: Number of packets retransmitted

been exhausted, we transmit new data at the CBR once more, as seen in the figure. For the duration of the sawtooth the receiver experiences delayed data arrival, but subsequent data arrives in a timely manner once more, until the next loss. In this situation, with a constant bit-rate of 512 Mbit/s and a 15 ms RTT, we can use a 160 Mbyte buffer on the sending side to allow timely delivery to be resumed at the receiver.

Instead of never resuming timely arrival and the departure for timely arrival becoming increasingly worse with time, which is the situation with conventionally sized buffers, we can use larger buffers to instead suffer only a temporary period of delayed data arrival. One must consider however the logistics of providing such large buffers and be able to cope with the temporary period of delay.

6. Conclusions

In a lossy environment, using TCP/IP to transfer CBR data with normal TCP buffer settings (BDP) leads to delayed data arrival. The delay is to the entire stream of data as all data arriving after the first loss will be delayed, with subsequent losses compounding the problem. For an application such as e-VLBI this not acceptable and can lead to a loss of correlation and lower quality results as multiple streams become unsynchronised.

In theory, to regain timely delivery of data following a loss, it is necessary to store the delayed data and subsequently transmit it at a rate exceeding the constant bit-rate to achieve an average CBR throughput. This can be demonstrated in practice in a relatively simple manner by using the socket buffers for this temporary data storage. The practicalities of providing buffers depend strongly on the parameters of the application and network. For a 512 Mbps flow over a connection with a round trip time of 15 ms we are required to buffer 160 Mbytes of data. The scaling apparent

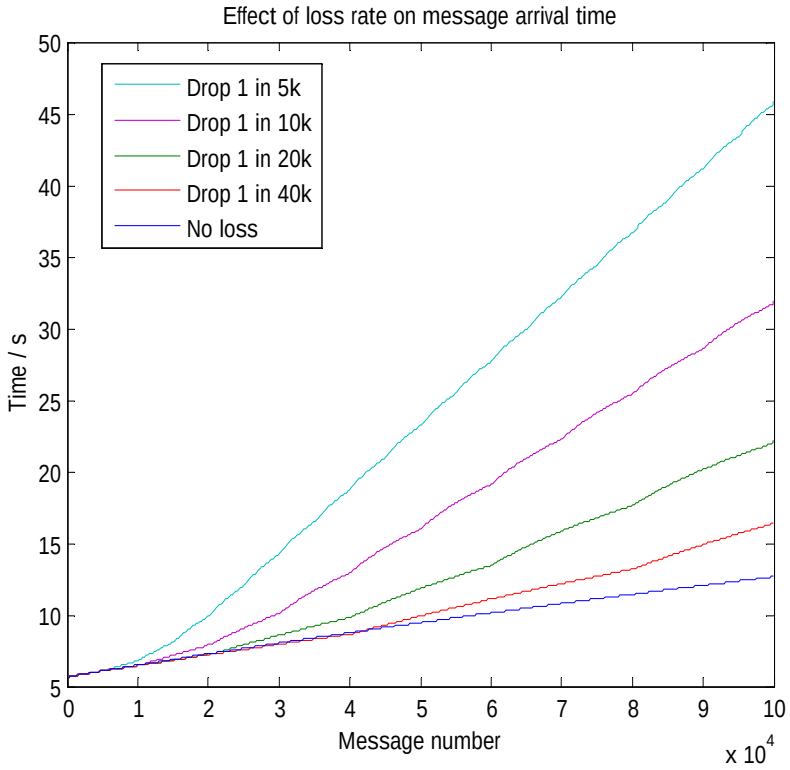


Figure 3: The effect of packet loss on message arrival time. Manchester to Jodrell Bank, looped through Amsterdam, 27ms RTT. TCP buffer size 1.8 Mbytes (BDP)

in Equation 2.3 is an important consideration, with transatlantic distances requiring the buffering of upwards of 5 Gbytes of data and temporary departure from timely delivery of tens of minutes. This will often prove impractical, with alternative protocols or TCP variants being options to consider.

References

[1] W. R. Stevens, *TCP/IP illustrated: the protocols*, Addison-Wesley Publishing Company, Reading, Mass., 1994

[2] K. Li et. al, *The Minimal Buffering Requirements of Congestion Controlled Interactive Multimedia Applications*, *Lecture Notes in Computer Science*, 2158, 2001

[3] C. Krasic, K. Li, J.Walpole, *The Case for Streaming Multimedia with TCP*, *Lecture Notes in Computer Science*, 2158, 2001

[4] B. Wang et. al., *Multimedia streaming via TCP: An analytic performance study*, *Performance Evaluation Review*, 32, 2004

[5] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm*, in *Computer Communications Review*, 27(3), July 1997

[6] J. Padhye et. al., *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, in proceedings of *ACM SIGCOMM*, September 1998.

[7] R. Hughes-Jones, *TCPdelay Home Page*. Available at :
http://www.hep.man.ac.uk/u/rich/Tools_Software/tcpdelay.html

[8] Various authors, *The Web100 Project*. http://www.web100.org/