

TCP-Based Bulk Data Transfer

The movement of bulk data over long-haul networks has become integral to modern computational science. Raw instrument data, computational data sets, and visualization data must be moved between institutions to take advantage of computational and other resources. Historically, bulk data transfer has been plagued by poor performance for a variety of reasons. These include improper configuration of the sending and receiving hosts and packet loss caused by ‘dirty’ network links, firewalls, small buffers, and other factors. In most cases, however, large data sets can be moved long distances using today’s networks with minimal effort.

Throughput

Moving large (tens of Gigabytes or more) data sets across the network is difficult using the default TCP settings on most hosts. Hosts are configured by default to allow them to function well for most common uses, e.g. reasonable performance on an Ethernet LAN, and reasonable performance when connecting to Web servers on the commodity Internet. TCP window settings in the range of 32kB to 64kB accomplish these goals, and these are typical default settings. These values are not at all appropriate for high-speed bulk data transfer across wide area networks. For example, a 200Mbps TCP stream over an 80 millisecond path (the continental US is approximately 80msec wide in terms of network round-trip-time) requires TCP buffers of 2MB. Using 32kB buffers, throughput would be limited by TCP to 3.2Mbps per stream, even if the network had more bandwidth available. Wide area bulk data transfer typically requires data rates not achievable with default settings. Note that the relationship between data rate, TCP window (“buffer”) size, and round-trip time (RTT) is commonly called the Bandwidth-Delay Product:

$$\text{TCP_window} = \text{data_rate} * \text{RTT}$$

This relationship is described more thoroughly in many TCP tuning documents, including <http://dsd.lbl.gov/TCP-tuning/> and <http://www.psc.edu/networking/projects/tcptune/>.

The following table illustrates the data rate required to move a data set of a particular size in a given amount of time. The red shaded areas indicate single-stream throughput rates greater than 10Gbps, while the green shaded areas indicate data rates that should be straightforward to achieve.

10PB	300,240.0 Gbps	25,020.0 Gbps	3,127.5 Gbps	1,042.5 Gbps	148.9 Gbps	34.7 Gbps
1PB	30,024.0 Gbps	2,502.0 Gbps	312.7 Gbps	104.2 Gbps	14.9 Gbps	3.5 Gbps
100TB	2,932.0 Gbps	244.3 Gbps	30.5 Gbps	10.2 Gbps	1.5 Gbps	339.4 Mbps
10TB	293.2 Gbps	24.4 Gbps	3.1 Gbps	1.0 Gbps	145.4 Mbps	33.9 Mbps
1TB	29.3 Gbps	2.4 Gbps	305.4 Mbps	101.8 Mbps	14.5 Mbps	3.4 Mbps
100GB	2.9 Gbps	238.6 Mbps	29.8 Mbps	9.9 Mbps	1.4 Mbps	331.4 Kbps
10GB	286.3 Mbps	23.9 Mbps	3.0 Mbps	994.2 Kbps	142.0 Kbps	33.1 Kbps
1GB	28.6 Mbps	2.4 Mbps	298.3 Kbps	99.4 Kbps	14.2 Kbps	3.3 Kbps
100MB	2.8 Mbps	233.0 Kbps	29.1 Kbps	9.7 Kbps	1.4 Kbps	0.3 Kbps
	5Min	1H	8H	24H	7Days	30Days

Table 1. Data rates required for data set movement in a given time interval

This table shows that while the default TCP settings of most hosts may not be appropriate, significant throughput can be achieved in a few hours with modest resources. Two good approximations are that one can move about 1 Terabyte per 100Mbps of throughput per day, and one can move 10 Gigabytes per 1Mbps of throughput per day. The first requires slightly greater than Fast Ethernet speeds (due to protocol overhead, Fast Ethernet is capable of maximum throughput rates in the 85 to 90 Mbps range). The second is easily achievable with a laptop.

Packet Loss

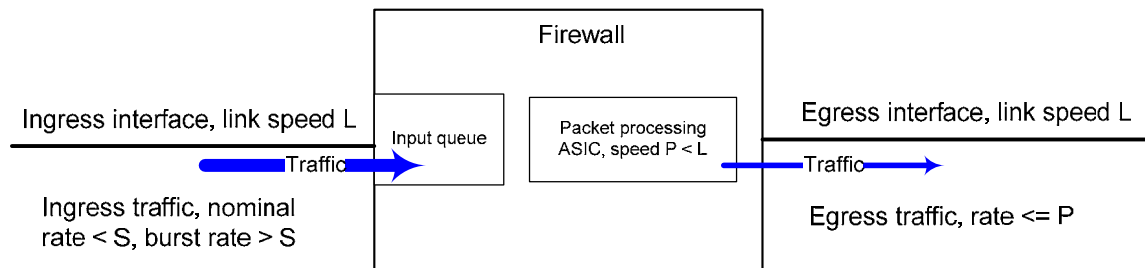
Sometimes well-configured hosts cannot achieve the necessary data rates, even over apparently uncongested networks. In most cases, this is caused by packet loss. Packet loss is most often caused by firewalls and network link aggregation in devices with small packet buffers. Other causes include ‘dirty’ network links (links with constant, low-level packet loss) and failing hardware. Locating the source of packet loss is often an involved process, since the entire path must be examined – people with the ability to monitor (and potentially reconfigure) all the network devices in the path must be made aware of the problem, and must participate in finding its cause. This can be made simpler by conducting tests to well-known hosts (e.g. the ESnet Performance Testers, see <https://performance.es.net/>) in an attempt to isolate the problem. For example, if a user on the West Coast can get the same excellent TCP throughput to the ESnet performance testers in Sunnyvale and Chicago, but gets terrible performance regardless of TCP settings when testing with the performance tester in New York, it might be that there is a problem between Chicago and New York. On the other hand, if performance declines as the user progressively tests farther afield (throughput declines as RTT increases) regardless of TCP settings, it is likely that the source of packet loss is close to the site host, i.e. the packet loss is likely to be present in the part of the network common to all tests.

Let us examine the two most common causes of packet loss (firewalls and aggregation devices with inadequate buffering) in more detail. In order to understand the problem, one has to realize that a TCP flow rarely runs at its “average” speed, and even then only after the flow has been running for some time. When observed closely, it is apparent that most high-speed TCP flows are composed of bursts and pauses, especially during their

first few minutes. These bursts are often very close to the maximum data rate for the sending host's interface. This is important, because it means that a 200Mbps TCP flow between hosts with Gigabit Ethernet interfaces is actually composed of short bursts at or close to 1Gbps with pauses in between.

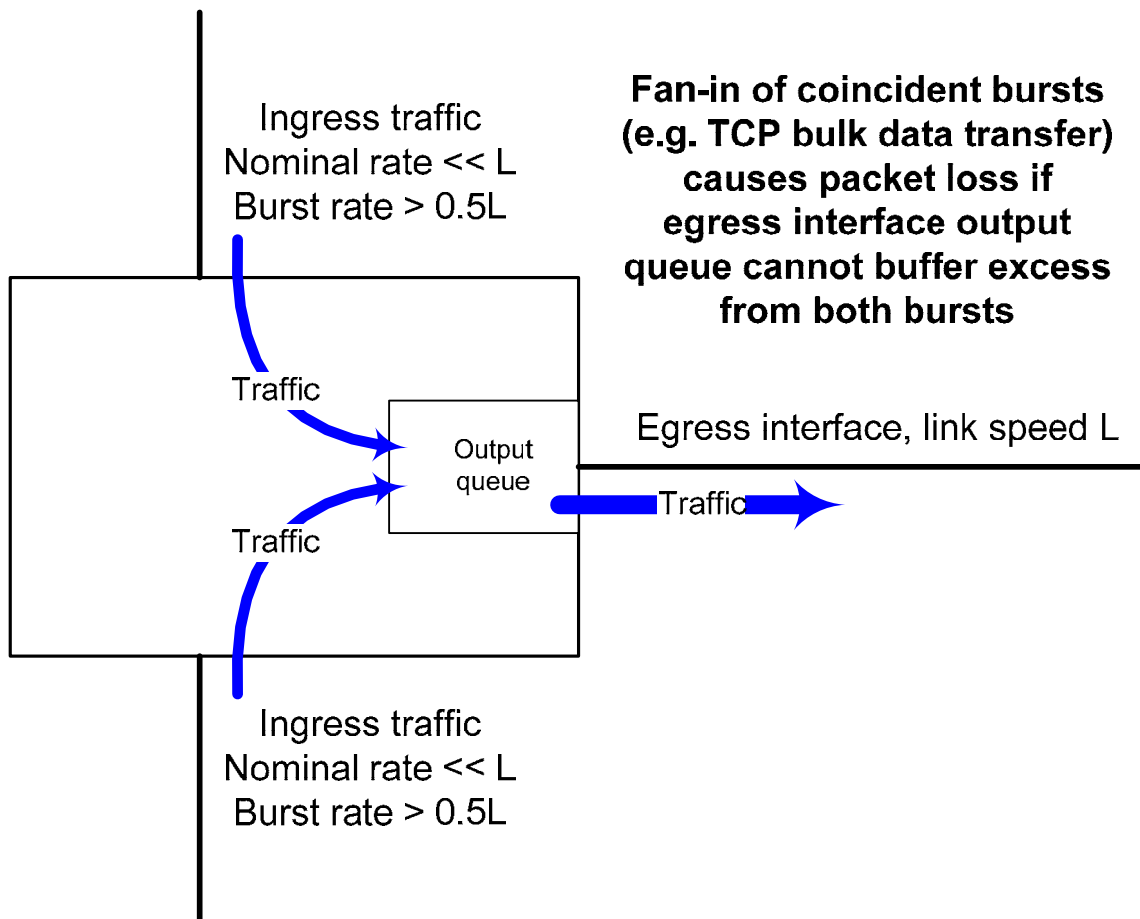
Firewalls are often slower than the link speed of their network interfaces (e.g. many firewalls with Gigabit Ethernet interfaces have a maximum throughput rate of 800 Mbps). This causes a problem when a host with a network interface that is faster than the firewall's internal processor attempts to send data through the firewall (remember that TCP bursts typically occur at or near the maximum data rate of the sending host's interface). Since the firewall must buffer the traffic bursts sent to it by the data transfer host until it can process all the packets in the burst, input buffer size is critical. Firewalls often have small input buffers (firewalls are typically designed to scale to large numbers of low-speed flows, rather than a few high-speed data flows, since large numbers of low-speed flows is a typical Enterprise traffic profile). If the firewall's input buffers are too small to hold the bursts from the data transfer host, packet loss will result – often causing severe performance problems.

Firewall with packet processor slower than ingress link speed causes packet loss if ingress interface input queue is unable to buffer excess burst

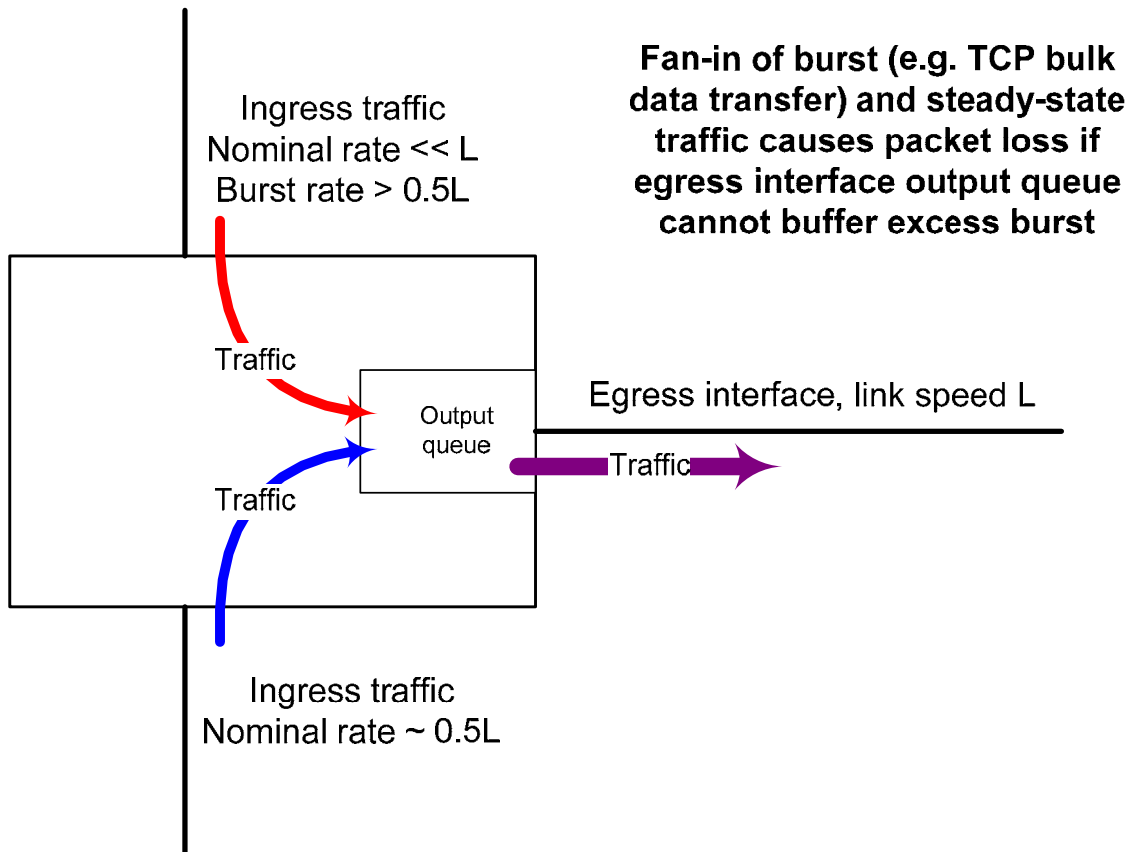


In this case the limiting factor is the combination of input buffer size and processing speed of the firewall, not TCP window size or network link speed, though TCP window size will have an effect on the size of the burst sent by the data transfer host.

Aggregation (“fan-in”) problems are related to the firewall problem in that they, too, result from the combination of the burstiness of TCP traffic and small buffers on network devices. However, the fan-in problem arises when multiple traffic flows from different ingress interfaces on a switch or router are destined for the same egress interface. If the speed of the sum of the bursts arriving at the switch is greater than the speed of the device’s egress interface, the device must buffer the traffic. If the device does not have sufficient buffer space, it must drop some of the traffic, causing performance problems. This situation is particularly common in inexpensive LAN switches - since high-speed packet memory is expensive, cheap switches often don’t have enough buffer space to handle anything except LAN traffic.



Note that the fan-in problem is not unique to coincident bursts. If a burst from a single flow arrives at a rate greater than the rate available on the egress interface due to existing non-bursty traffic flows, the same problem exists as in the following diagram:



Recovery from packet loss events is important to why packet loss matters so much. TCP interprets packet loss as congestion, and assumes that it must reduce its transfer rate. However, when TCP encounters packet loss it often backs off drastically, and increases its data transfer rate very slowly after the loss event (typically, the TCP window opens up at the rate of one packet per round trip which is very slow on a long-latency path). However, when a TCP data transfer first begins sending data, the window opens up very rapidly (typically exponentially, with TCP sending 2,4,8,16 ... packets per round trip) until either the TCP window is full or a loss event occurs. This is called the “slow start” phase. If packet loss is encountered during slow start as is common (TCP is sending successively larger bursts at or near line rate) the connection enters congestion avoidance immediately and then opens up at one packet per round-trip.

The following table shows some examples of this behavior – for a given transfer speed, it shows the typical time required to ‘get back’ to that speed after a loss event, and the data transferred during the recovery time. An obvious consequence of this is that if one attempts to transfer a file a few gigabytes in size in the face of packet loss, the transfer

could spend most of its time in congestion recovery as a result of a single loss event resulting from a switch with small buffers halfway across the world. If a scientist is moving a data set consisting of many such files, the impact can be significant.

Transfer speed	Recovery time	Data sent during recovery
10Mbps	8.6 sec	5.5MB
100Mbps	86 sec	537MB
200Mbps	3 min	4.8GB
500Mbps	7 min	13.4GB
1Gbps	14 min	53.5GB

Table 2. Time to fully recover from congestion loss and data sent during recovery period

Jumbo Frames

Jumbo frames (typically 9000 bytes, where standard Ethernet frames are 1500 bytes) are helpful in two different ways. One is simple enough – for a given data rate, jumbo frames reduce the packet rate by a factor of 6. This means that the load on the sending and receiving hosts due to packet rate (an example of this is per-packet interrupt handling) is also reduced. The other clear win from jumbo frames is that, since recovery from loss events happens on a per-segment basis, increasing the segment size by a factor of 6 also increases the recovery rate from packet loss events. This behavior is a component of the so-called Mathis Equation,¹ which states that for constant RTT and packet loss, the available bandwidth is proportional to the maximum segment size of the TCP connection (MSS).

$$Bandwidth \leq \left(\frac{MSS}{RTT} \right) \frac{1}{\sqrt{p}}$$

The ESnet backbone is fully capable of carrying jumbo frames, as are the peerings between ESnet and the other major research and education networks.

¹ M. Mathis, J. Semke, J. Mahdavi, T. Ott, "**The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm**", *Computer Communication Review*, volume 27, number3, July 1997.
<http://www.psc.edu/~mathis/papers/index.html>

Host Tuning

Tuning a host for wide area data transfer is a multi-variable problem. TCP window sizes must be configured on a per-destination basis. The best way to do this is to use a data transfer application that allows per-destination configuration of TCP parameters, since host kernels are often not this flexible (both GridFTP and HSI allow this).

Documentation describing host TCP tuning is widely available – good examples are <http://dsd.lbl.gov/TCP-tuning/> and <http://www.psc.edu/networking/projects/tcptune/>. Understanding the performance of the host's local storage is also important – it doesn't matter if you have a pair of fast hosts with a clean 10Gbps network between them if you're using slow disks. The important thing to remember is that host tuning is a per-host-pair exercise, i.e. it doesn't matter if one side of the connection is perfectly configured if the host at the remote end of the connection is set to defaults.

It is important to note that configuring host TCP parameters so that the host will attempt to use all the bandwidth theoretically available to it (e.g. 622Mbps for a data transfer over a path where an OC12 is the smallest link) is almost never a good idea. It is likely that the data transfer will be required to share the network with other traffic, and allowing a big, bursty TCP flow to try and use all the bandwidth will almost certainly result in reduced performance for all the traffic. A good rule of thumb is to tune a host for half the average available bandwidth – this is likely to succeed in the near term, and unlikely to require immediate reconfiguration as network conditions change.

Putting It All Together

In the end, the goal is to get the data moved so that the scientists can get their work done. Many disciplines have current requirements to move data sets in the range of a few hundred gigabytes to a few tens of terabytes – these operations should be possible with current networks, without resorting to shipping tapes, hard drives or RAID boxes by truck or air. For example, using the throughput table, we can see that moving 10TB of data can be accomplished in a week using about 150Mbps, or about ¼ the bandwidth of an OC12 (622Mbps). Please note that ESnet staff are experienced in solving TCP performance problems, and if a user or project is having trouble moving the data they need to move to get their science done (especially if they are shipping tapes or disks to move data), users are welcome to contact trouble@es.net – we will do whatever we can to help get a data set moved, or provide assistance in setting up long-running data movement processes.