Myth Buster #1: TCP vs. UDP

Under the Covers and at the Application Layer

Frequently, transferring large volumes of data, either in huge video files or enormous directories of variable-sized much smaller files, creates a question among IT managers regarding whether to use data movement software based upon the TCP or UDP protocols.

TCP users complain that UDP is non-deterministic, connectionless, and lossy, while UDP users complain that TCP is slower and nobody needs all of its features anyway.

Both are wrong.

So the real questions are: has anyone actually seen and/or experienced UDP packet loss in actual use? Is the degeneration of UDP any better or worse than TCP in a congested environment? TCP may also experience issues in times of congestion, but at least you know about it, some would say. How do these issues manifest in the software products available to move the data from *here* to *there*. Where is the truth, and why does it matter? Are there other factors of the underlying protocol that limit a product's usefulness? Indeed, there are.

So, let's start with some basics and go on from there.

1. It is essential to keep in mind that User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are simply the underlying transport mechanisms (software methods or "protocols" provided by the operating system) in accordance with industry standards and supported by hardware vendors in their routers, switches, and network interface cards and chips (NICs). It is rare that raw TCP or UDP is used to move data; software vendors surround both protocols with an application layer of additional code that, in their view, improves performance, provides additional features, or overcomes short-comings of the basic protocols.

2. The service level agreement (SLA) between bandwidth providers and users usually specifies a minimum level of assurance that your data will make it from source to destination—that is, be free from packet loss to the stated values. These SLAs are usually less than 0.1% packet loss on copper-based infrastructure, or between 0.01% - 0.001% on fibre-based infrastructure.

> With an SLA of 0.01% - 001%, 2% packet loss would be 200 to 2,000 times the SLA! (Please note below the potential packet loss of UDP.)

> Keep in mind the SLA covers each particular set of endpoints, e.g., if your facility is on the East Coast and you need to move data to the West Coast, your local endpoint is usually within a 50-mile radius of your facility. From there, your data moves to the Internet, encountering switches and routers (termed "hops") along the way to the destination, where another set of SLAs exist that may or may not be the same as the source, but probably close enough not to matter much.

> SLAs refer to TCP. No one can predict UDP packet loss or make representations of its frequency.

3. UDP is a very simplistic protocol. There is nothing from a user's perspective that UDP allows that TCP cannot also do, but the reverse is quite different. There are many things that TCP can do or allows that UDP cannot. What are these differences, how do they affect data movement and other desirable processes, and why should you care? Refer to *Figure 1*, below.

Figure 1: TCP vs. UDP

TCP	UDP
Connection	
TCP is a connection-oriented protocol. As a message makes its way across the network from one computer to another, they communicate with one another. This is a connection based transmission. The receiver acknowledges that the message has arrived.	UDP is a connection-less protocol. This is not connection based, which means that one program can send a load of packets to another and that would be the end of the relationship.
Ordering	
TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
Reliability	
There is absolute guarantee that the data transferred remains intact and arrives in the exact order in which it was sent.	There is no guarantee that the messages or packets sent would reach the destination at all. Only if the message arrives in the basic integrity check performed, e.g., a low-level checksum.
Data Streaming	
Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries, which are honored upon receipt, but the receiver is unaware if packets have been dropped or lost along the way.
Flow Control	
TCP does flow control. TCP handles reliability and congestion control.	UDP does not have flow control and is unaware if packets are received.
Error Checking	
TCP does error checking and has recovery options.	UDP does error checking, but has no recovery options.

As can be seen in *Figure 1*, the comparison of UDP and TCP is useful in determining which is better suited for a particular application. Not surprisingly, the following addresses Aspera FASP® and TransferSoft HyperTransfer. The following examination is focused on how the underlying protocol limits or expands application usefulness.

Ordering

Packet (or Datagram) sequence can be critical in some applications.

- **HyperTransfer** data is received and reordered in the exact sequence read from the host.
- **Aspera** packets are received and recorded in random order, leading to high storage latency with reading the received data at the destination.

Reliability

100% guaranteed reliable data delivery.

- **HyperTransfe**r delivers data with 100% reliability—first time, every time.
- **Aspera** has been observed losing up to 44% of packets sent on first transfer. The Aspera software then collects a list of missing packets (this process is performed, oddly enough, using TCP) and then attempts to resend those packets that weren't received the first time. This process may repeat for quite some time depending upon packet loss percentage and the size of the data transfer.

  In such an extreme case, the retransmissions not only take additional time to complete, but 44% (or more) data has to be transferred before the data is safely at its destination, and the received data is finally useful.

Streaming

TCP streaming vs. UDP Datagrams has multiple consequences.

- **HyperTransfer** uses TCP streaming to launch multiple simultaneous internal streams with the effect of incredible speed, packet-loss avoidance, strong and effective data reduction techniques (delta difference compression as well as standard compression techniques, in addition to extremely strong encryption, and strong check-summing, all without loss of performance.

  Linked with HyperTransfer's Parallel I/O and parallelization of all data transforms, a powerful case is made for TCP Streaming vs. UDP Datagram.
- **Aspera**, using UDP datagrams (limited to 64 KB maximum), makes any type of data reduction practically impossible, and the small datagram size dramatically reduces encryption and check-summing performance.

Flow Control

Flow control enhances productivity.

- **HyperTransfer's** enhanced congestion control algorithms and multi-threaded streaming protocol result in reduced packet loss and extreme performance gains compared to any UDP-based transfer application. Since HyperTransfer automatically load-balances with other TCP-based processes, operation on LANs is perfectly supported, running at pre-determined maximums, or load-balancing with all other processes on an equal basis.
- **Aspera**, with no flow control—other than metering packets at reduced rate to play fairly with TCP, which does, in fact, work properly—makes operation on LANs limited and problematic.

TCP Expands Additional Functionality

Moreover, with TCP as the underlying protocol, additional functions become a reality! For example:

- Port forwarding enables any TCP-based application to be enhanced with HyperTransfer feature extensions—WAN acceleration, encryption, data compression, and delta differencing deduplication. No programming or alteration to existing applications is required to use this facility. Furthermore, port forwarding works with NFS!
- Using streaming TCP allows HyperTransfer to fully support UNIX/Linux pipe functions to extend user applications through this simple and well-understood technique.
- Data reduction: On-the-fly data deduplication of large files and huge directories of files:
  - Identify and transfer only the data that has changed.
  - Keep file systems in sync and/or upload to cloud for hybrid environments.
- Lastly, recall that TCP is the very backbone of the Internet, encompassing nearly every user application from web browsing to email to database replication, SSH, and so on, and yes—data transfer as well: FTP, SCP, among others. UDP is generally used for video on demand (or video streaming in general), VOIP (Voice over IP) telephony, and multi-user gaming.