IBM Cloud
White paper

IBM Aspera FASP High-Speed transport

A critical technology comparison to alternative TCP-based transfer technologies

Contents:

Introduction

In this digital world, fast and reliable movement of digital data, including massive sizes over global distances, is becoming vital to business success across virtually every industry. The Transmission Control Protocol (TCP) that has traditionally been the engine of this data movement, however, has inherent bottlenecks in performance (Figure 1), especially for networks with high, round-trip time (RTT) and packet loss, and most pronounced on high-bandwidth networks. It is well understood that these inherent "soft" bottlenecks arcaused by TCP's Additive-Increase-Multiplicative-Decrease (AIMD) congestion avoidance algorithm, which slowly probes the available bandwidth of the network, increasing the transmission rate until packet loss is detected and then exponentially reducing the transmission rate. However, it is less understood that other sources of packet loss, such as losses due to the physical network media, not associated with network congestion equally reduce the transmission rate. In fact, TCP AIMD itself creates losses, and equally contributes to the bottleneck. In ramping up the transmission rate until loss occurs, AIMD inherently overdrives the available bandwidth. In some cases, this self-induced loss actually surpasses loss from other causes (e.g., physical media or bursts of cross traffic) and turns a loss-free communication “channel” to an unreliable “channel” with an unpredictable loss ratio.

The loss-based congestion control in TCP AIMD has a very detrimental impact on throughput: Every packet loss leads to retransmission, and stalls the delivery of data to the receiving application until retransmission occurs. This can slow the performance of any network application, but is fundamentally flawed for reliable transmission of large “bulk” data, for example file transfer, which does not require in-order (byte stream) delivery.

This coupling of reliability (retransmission) to congestion control in TCP creates a severe artificial throughput penalty for file transport, as evident by the poor performance of traditional file transfer protocols built on TCP such as FTP, HTTP, CIFS, and NFS over wide area networks. Optimizations for these protocols such as TCP acceleration applied through hardware devices or alternative TCP improve file transfer throughput to some degree when round-trip times and packet loss rates are modest, but the gains diminish significantly at global distances. Furthermore, as we will see later in this paper, parallel TCP or UDP blasting technologies provide an alternative means to achieve apparently higher throughputs, but at tremendous bandwidth cost. These approaches retransmit significant, sometimes colossal amounts of unneeded file data, redundant with data already in flight or received, and thus take many times longer to transfer file data than is necessary, and cause huge bandwidth cost. Specifically, their throughput of useful bits excluding retransmitted data packets – “goodput” – is very poor. These approaches deceptively appear to improve network bandwidth utilization by filling the pipe with waste, and transfer times are still slow!

For the narrow network conditions under which TCP optimizations or simple blasters do achieve high “good data” throughput, as network-centric protocols, they run up against further soft bottlenecks in moving data in and out of storage systems.

Transporting bulk data with maximum speed calls for an end-to-end approach that fully utilizes available bandwidth along the entire transfer path – from data source to data destination – for transport of “good data” – data that is not in flight or yet received. Accomplishing this goal across the great range of network round-trip times, loss rates and bandwidth capacities characteristic of the commodity Internet WAN environments today requires a new and innovative approach to bulk data movement, specifically, an approach that fully decouples reliability and rate control. In its reliability mechanism, the approach should retransmit only needed data, for 100 percent good data throughput. In its rate control, for universal deployment on shared Internet networks, the approach should uphold the principles of bandwidth fairness, and congestion avoidance in the presence of other transfers and other network traffic, while providing the option to dedicate bandwidth for high priority transfers when needed.

Aspera FASP is an innovative bulk data transport technology built on these core principles that is intended to provide an optimal alternative to traditional TCP-based transport technologies for transferring files over public and private IP networks. It is implemented at the application layer, as an endpoint application protocol, avoiding any changes to standard networking. FASP is designed to deliver 100 percent bandwidth efficient transport of bulk data over any IP network – independent of network delay and packet loss – providing the ultimate high-performance next-generation approach to moving bulk data.

In this paper we describe the alternative approaches to “accelerating” file-based transfers – both commercial and academic – in terms of bandwidth utilization, network efficiency, and transfer time, and compare their performance and actual bandwidth cost to Aspera FASP.

High-speed TCP overview

In recent years, a number of new high-speed versions of the TCP protocol and TCP acceleration appliances implementing these variations have been developed. High-speed TCP protocols recognize the fundamental flaw of AIMD and revamp this window-based congestion control algorithm to reduce the artificial bottleneck caused by it, and improve the long-term average throughput. The most advanced versions of these protocols typically aim to improve the detection of congestion through measuring richer signals such as network queuing delay, rather than increasing throughput until a loss event occurs. This helps to prevent TCP flows from creating packet loss, and thus artificially entering congestion avoidance, and improves the long-term throughput in nearly loss-free networks.

However, the improvement diminishes rapidly in wide area networks, where packet losses due to physical media error or buffer overflow by cross traffic bursts become non-negligible. A single packet loss in these networks will cause the TCP sending window to reduce severely, while multiple losses will have a catastrophic effect on data throughput. More than one packet loss per window typically results in a transmission timeout and the resulting bandwidth-delay-product pipeline from sender to receiver drains and data throughput drops to zero. The sender essentially has to re-slow-start data transmission.

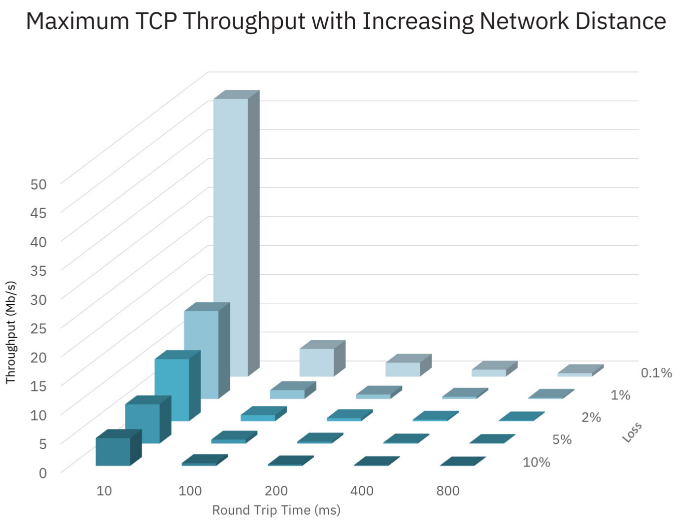


Figure 1: The bar graph shows the maximum throughput achievable under various packet loss and network latency conditions on an OC-3 (155 Mbps) link for file transfer technologies that use TCP (shown in yellow). The throughput has a hard theoretical limit that depends only on the network RTT and the packet loss. Note that adding more bandwidth does not change the effective throughput. File transfer speeds do not improve and expensive bandwidth is underutilized.

In contrast, in Aspera FASP the transmission rate is not coupled to loss events. Lost data is retransmitted at a rate corresponding to the end-to-end desired bandwidth. The retransmission achieves virtually ideal bandwidth efficiency – no data is transmitted in duplicate and the total target capacity is fully utilized.

As shown in Figure 2, the throughput of FAST TCP, one such commercial version of high-speed TCP (which include such variations as CUBIC, H-TCP, BIC, etc.) on a network of one percent packet loss improves the throughput over standard TCP Reno on low latency networks, but the improvement falls off rapidly at higher round-trip times typical of cross-country and intercontinental links. The FASP throughput in contrast has no degradation with increasing network delay and achieves up to 100 percent efficient transmission and an effective file transfer throughput at over 95 percent of the bandwidth capacity. Similarly, as packet loss increases (e.g., at five percent loss or more) the FASP throughput decreases only by the same amount. At higher loss rates the accelerated TCP throughput approximates Reno.

Standard and High-Speed TCP’s reaction to packet loss forces the sender to not only reduce its sending window, leading to an erratic transfer speed, but also pre-empts new packets in the sending window with retransmitted packets to maintain TCPs in-order delivery guarantee. This transmission of new and retransmitted packets in the same TCP sending window entangles the underperforming TCP congestion control with TCP reliability control that guarantees transfer integrity, and unnecessarily handicaps transfer throughput for applications that do not require in-order delivery, such as bulk data.

TCP reliability guarantees that no data is lost (the lost packets will be detected by the receiver and retransmitted by the sender afterwards), and received data is delivered, in-order, to the application. In order to fulfill these two guarantees, TCP not only retransmits the lost packets, but also stalls the earlier-arriving, out-of-order packets (stored temporarily in the kernel memory) until the missing packet arrives, and the received data can be delivered to the application layer, in-order. Given the requirement that the receiver must continue storing incoming packets in RAM until

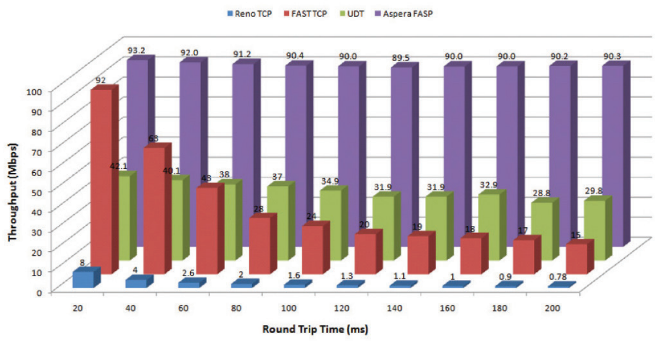


Figure 2: File transfer throughput for 1 GB file comparing Reno TCP, a commercially available high-speed TCP, UDT, and Aspera FASP in a link with medium packet loss (1%). Note that while the accelerated TCP improves Reno throughput on lower latency networks, the throughput improvement falls off rapidly at higher round-trip times typical of cross-country and intercontinental links. The FASP throughput in contrast has no degradation with increasing delay. Similarly, as the packet loss rate increases (e.g., at 5% loss) the FASP throughput decreases only by about the same amount, while high-speed TCP is no better than Reno.

the missing data is received, retransmission is urgent and first priority, and the sending of new data must be slowed in concert. Specifically, on every packet loss event, new packets have to slow down (typically the sending window freezes until lost packets are retransmitted to receiver and acknowledged), waiting for retransmitted packets to fill "holes" in the byte stream at receiving end. In essence the reliability and flow control (or congestion control) in TCP are by design, thoroughly coupled.

Although this type of mechanism provides TCP with a strict in-order byte stream delivery required by many applications, it becomes devastating to applications that naturally do not require strict byte order, such as file transfer, and thus introduces a hidden artificial bottleneck to these applications, limiting their corresponding data throughput.

To make it crystal clear, we can explore a simple example to calculate the throughput loss due to a single non-congestion related packet loss in a High-Speed TCP with a window reduction of one-eighth on each loss. For a Gigabit network with one percent packet loss ratio and 100 ms round-trip delay, every single packet loss causes the rate to reduce by one-eighth (compared with one half in TCP Reno) and it will take 1 Gbps÷8(bits/byte)÷1024(bytes/packet)×100 ms×0.125 (drop ratio/loss)×100ms ≈ 152.6 seconds for the sender to recover the original sending speed (1 Gbps) before the packet loss event. During this recovery period, High-speed TCP loses about 152.6s×1 Gbps×0.125/2 ≈ 8.9 GB throughput because of a single loss event! In the real wide area network, the actual value will be even larger since RTT can be larger due to network queuing, physical media access, scheduling and recovery, etc. Thus it typically takes longer than 152.6 seconds for the sender to recover. Multiple consecutive packet losses will be a catastrophe. A quote from Internet Engineering Task Force (IETF) bluntly puts the effect in this way: "Expanding the window size to match the capacity of an LFN [long fat network] results in a corresponding increase of the probability of more than one packet per window being dropped. This could have a devastating effect upon the throughput of TCP over an LFN. In addition, since the publication of RFC 1323, congestion control mechanisms based upon some form of random dropping have been introduced into gateways, and randomly spaced packet drops have become common; this increases the probability of dropping more than one packet per window."[1]

We note that this rate slowdown or throughput loss is sometimes indeed necessary for byte-stream applications where strict in-order delivery is a must. Otherwise, RAM has to accommodate at least 1 Gbps×100 ms×0.125 ≈ 1.5 MB extra data just to wait for a single lost packet of each TCP connection for at least one RTT in our earlier example. However, this slowdown becomes unnecessary for file transfer applications because out-of-order data can be written to disk without waiting for this lost packet, which can be retransmitted any time at the speed that precisely matches the available bandwidth inside the network, discovered by an advanced rate control mechanism.

Indeed TCP by itself will not be able to decouple reliability and congestion control and thus will not remove this artificial bottleneck unless the purposes of TCP – providing reliable, byte-stream delivery – are redefined by the IETF.[2] The traditional reliance upon a single transmission control protocol for both reliable streaming and non-streaming applications has been proven in practice to be suboptimal for both domains.

UDP-based high-speed solutions

The reliability provided by TCP reduces network throughput, increases average delay and worsens delay jitter. Efforts to decouple reliability from congestion avoidance have been made for years. Due to the complexity of changing TCP itself, in recent years academic and industry practices have pursued application-level protocols that feature separable rate and reliability controls. These approaches use UDP in the transport layer as an alternative to TCP and implement reliability at the application layer. Most such approaches are UDP blasters – they move data reliably with UDP, employing some means to retransmit lost data - but without meaningful consideration of the available bandwidth, and risk network collapse, not to mention collapse of their own throughput. Figure 3 shows the throughput of Rocketstream, a commercial UDP data blaster, when run over a 300 Mbps link with typical WAN conditions (increasing RTT and packet loss).

UDP solutions, including open source implementations Tsunami and UDT (used by products such as Signiant, File

Catalyst, and Sterling Commerce®), have attempted to strengthen congestion control in UDP blasters through simplistic algorithms that reduce transmission rate in the face of packet loss. While the back off can be "tuned" to achieve reasonable performance for specific network pathways on a case-by-case basis, meaning single combinations of bandwidth, round-trip delay, packet loss and number of flows, the design is inherently unable to adapt to the range of network RTT and packet loss conditions and flow concurrency in any real-world Internet network. Consequently, these approaches either underutilize available bandwidth, or apparently "fill the pipe", but in the process overdrive the network with redundant data transmission – as much as 50 percent redundant data under typical network conditions - that wastes bandwidth in the first order, and leads to collapse of the effective file transfer throughput ("goodput") in the second order. Finally, in the process these approaches can leave the network unusable by other traffic as their overdrive creates packet loss for other TCP applications and stalls their effective throughput.

We selected one of the most advanced retransmission (NACK-based) UDP transport solutions, "UDT", re-packaged by commercial vendors, to demonstrate these problems. Specifically, they include:

- Poor congestion avoidance. The dynamic "AIMD" algorithm (D-AIMD) employed in UDT behaves similarly to AIMD, but with a decreasing additive increase (AI) parameter that scales back the pace of rate increase, as the transmission rate increases. This approach fails to recognize the aforementioned key issues of TCP – the entanglement of reliability and rate control – and instead makes the assumption that tuning one parameter can solve the underperformance of AIMD and even TCP. Indeed, a specifically tuned D-AIMD outperforms TCP in one scenario, but immediately underperforms TCP in another. Thus, in many typical wide area networks, the performance of UDT is actually worse than TCP.
- UDT's aggressive data sending mechanism causes dramatic rate oscillation and packet loss, not only undermining its own throughput, but also jeopardizing other traffic and degrading overall network performance. In a typical wide area network where a regular TCP flow (e.g., a HTTP session of a web client) shares bandwidth with a UDT transfer, the TCP flow can potentially experience denial-of-service due to aggressiveness of the UDT flow (Figure 4). Indeed, the possibility of extreme TCP unfriendliness is theoretically studied in the original UDT paper, "Optimizing UDP-based Protocol Implementations, 2005", and the authors propose a specific condition that must be satisfied to avoid this extreme unfriendliness.[2] In reality, for very typical wide area networks (e.g., WAN with 100 ms RTT and 0.1 percent plr), the condition cannot be satisfied and thus this extreme TCP unfriendliness will be inevitable. That means in order to use a UDT-based data movement solution, a regular customer will likely need to invest more time and money on some type of QoS companion to guarantee UDT will not damage the operation of the whole network ecosystem (e.g., web,

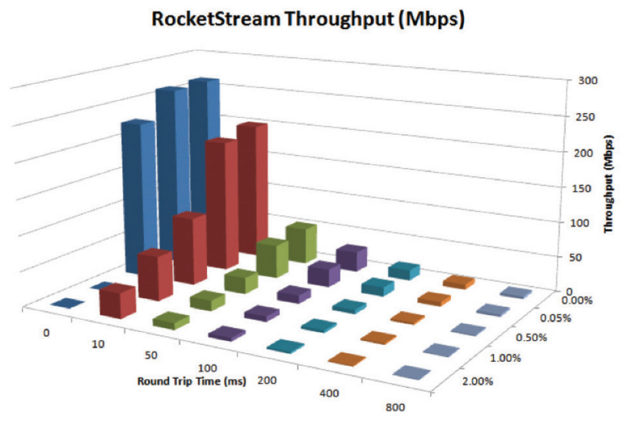


Figure 3: The bar group shows the throughput achieved under various packet loss and network latency (WAN) conditions on a 300 Mbps link for RocketStream. Bars with zero height represent failures of establishing connection between sender and receiver, which is not uncommon to RocketStream when either RTT or packet loss ratio is large.

email, VOIP, network management).

- Aggressive sending and flawed retransmission in UDT leads to lower efficiency of valuable bandwidth and often forces customers to purchase more bandwidth unnecessarily. (The very solution that is intended to better utilize expensive bandwidth actually wastes it.) The large difference between sending rate, receiving rate, and effective file transfer rate in some experiments (Figures 6 and 7) exposes the significant data drops at the router and the receiver primarily due to overly aggressive data injection and the flawed retransmission mechanism of UDT. Measured efficiency ("goodput") drops below 20 percent in some typical wide area networks. That means a 100 percent fully utilized network by UDT uses 80 percent of the bandwidth capacity transmitting redundant (duplicate) data to receiver or transmitting useful data to an overflowed buffer (overdriven by UDT itself).

The "benefit" and "cost" of using UDT to a regular user can be quantified for an accurate comparison. "Benefit" can be measured by the efficient use of bandwidth for transferring needed data (goodput) translating directly into speedy transfer times, while the "cost" can be abstracted as the effort of transferring one needed data packet, defined as how many duplicated copies are transferred to get one needed packet successfully delivered to the application layer at another network end. This cost also implies the induced costs to other transfers, reflected by their loss of fair bandwidth share (Figure 4) and thus their degraded throughputs. Specifically, as already partially reflected in Figure 5, UDT has lower effective transfer throughput (resulting in slow transfers) over a wide range of WAN conditions, and thus brings little benefit to users. And, the associated bandwidth cost due to overdrive and redundant retransmission dramatically affects other workflows.

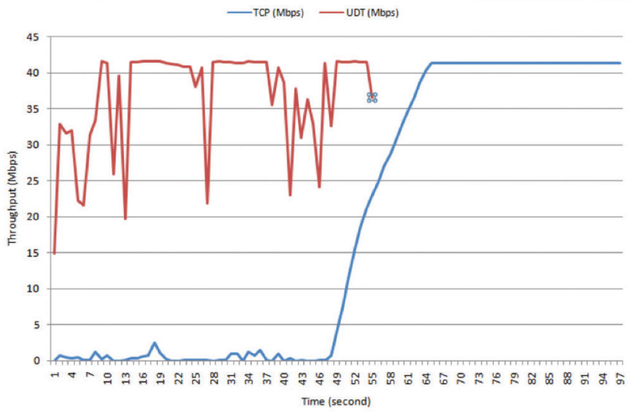


Figure 4: File transfer throughput of a single UDT transfer on a typical T3 link with packet loss ratio and 50 ms RTTs, and the effect of UDTtransfer on a regular TCP flow. The TCP flow is not "visible" for most of its duration until the UDT flow terminates. transfer on a regular TCP flow. The TCP flow is not "visible" for most of its duration until the UDT flow terminates.

Figure 5 shows the overall cost of transmitting one packet by a single UDT transfer on a T3 (45 Mbps) link under different RTTs and packet loss ratios. For most typical wide area networks, one packet transfer needs eight to ten retransmissions. In another words, in order to transfer a 1-gigabyte file, the UDT sender dumps nine to eleven gigabytes into the network in the end. The transfer takes 9 – 11 times longer than necessary, and also causes large packet loss to other flows.

The cost in Figure 5 is caused by the overly aggressive injection rate of the UDT sender and duplicate retransmissions dropped by the UDT receiver. To be more specific, we can define sending cost to reflect the loss due to an overly aggressive injection by the sender and thus packet drops at router, and the receiving cost to reflect duplicate retransmissions dropped at receiver.

More accurately, the sending cost is

$$\text{sending cost} = \frac{\text{total bytes sent - actual bytes received}}{\text{actual bytes received}}$$

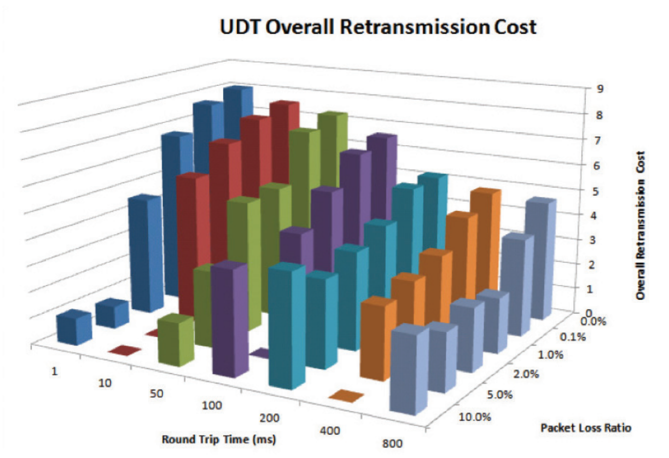


Figure 5: The bar graph shows the retransmissions of a single UDT transfer under different RTTs and packet loss ratios. The height of each bar, the "transmission cost," is the quantity of retransmitted data in units of gigabytes when a 1 GB file is transferred. Bars with zero height represent failures to establish a connection between sender and receiver, which is not uncommon to UDT when either RTT or packet loss ratio is large. Note that up to nine TIMES the original file size is sent and in wasteful retransmissions.

and the receiving cost is

$$\text{receiving cost} = \frac{\text{total bytes received - actual useful bytes}}{\text{actual useful bytes}}$$

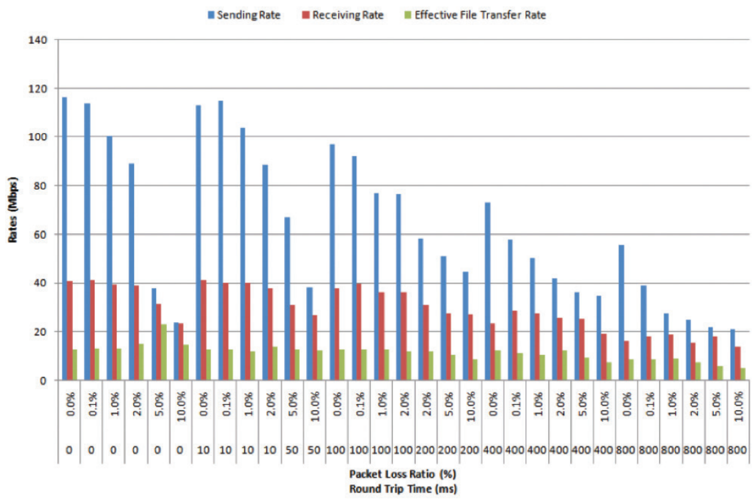
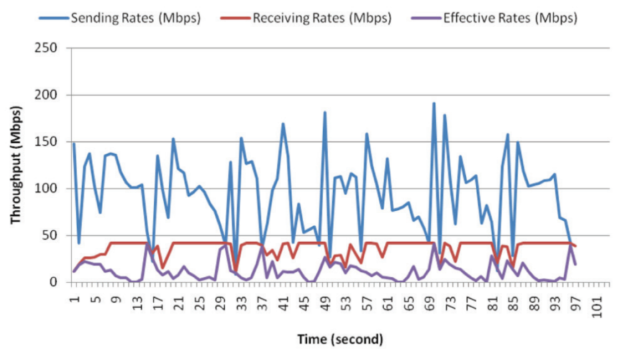
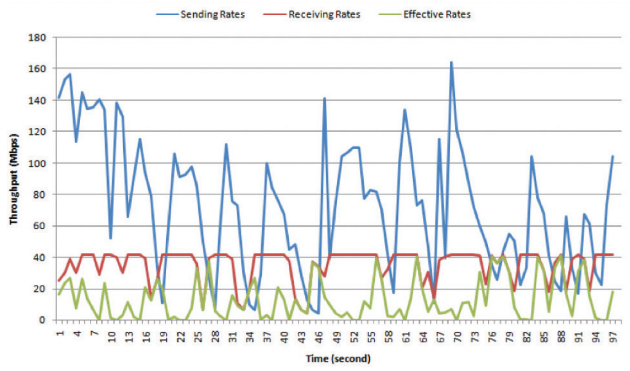


Figure 6: The bar graph shows the sending rates, receiving rates, and effective rates of a single UDT transfer under different RTTs and packet loss ratios on a T3 link. Note that the large difference between sending and receiving rate implies large packet loss on the intervening network path, and the large difference in the receiving and effective rate implies a large number of duplicate retransmissions.

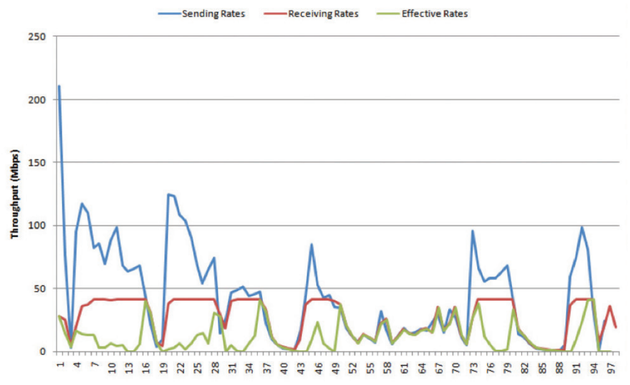
Figure 7: The sending, receiving, and effective receiving rates of a single UDT transfer on a T3 Link with 0%, 1% and 5% packet loss ratios and 100 ms, 200 ms, 200 ms RTTs. The gap between sending and receiving rates implies large amount of data loss at router, while the gap between receiving and effective receiving rates reflects the large number of drops of duplicate retransmissions at the UDT receiver.



(a) UDT transfer on a T3 Link with 0% plr and 100 ms RTT



(b) UDT transfer on a T3 Link with 1% plr and 100 ms RTT



(c) UDT transfer on a T3 Link with 5% plr and 200 ms RTT

Note that the higher the sending cost, the more packets are dropped at the router, while the higher the receiving cost, the more packets are dropped at receiver. Figure 7 shows the sending rates, receiving rates, and effective rates of a single UDT transfer under different RTTs and packet loss ratios on a T3 link. The rates ratios (sending rate to receiving rate and receiving rate to effective rate) will be the defined costs above. We observe that sending rates are persistently higher than receiving rates, which are again persistently higher than effective rate in **all network configurations. These costs drive the network to an operational point where network utilization (defined as throughput divided by bandwidth) is close to one, but the network efficiency (defined as goodput divided by bandwidth) is as low as 15 percent. Consequently, any given file transfer is over six times slower than it should be.**

To be crystal clear, we can verify the above costs through a simple file transfer example under different wide area networks by answering the following performance-related questions:

- How many bytes are to be sent?
- How many bytes are actually sent?
- How many bytes are actually received?
- How long has the transfer taken?
- What is the effective file transfer speed?

Bandwidth (Mbps)	RTT (ms)	plr (%)	How much to be sent (MB)	How much needs to be sent (actual data + inevitable loss by media, MB)	How much data actually sent (MB)	Sending cost (Sender's Overhead, %)	How much data actually received? (MB)	Receiving Cost (Receiver's Overhead, %)	How long does it take? (s)	Effective file transfer speed (Mbps)	Observed Network Utilization	Network Efficiency (Effective Utilization, %)
45	0	0	953.7	953.7	9093.2	314.1%	2195.8	130.2%	625.0	12.8	66.2%	28.4%
45	100	1	953.7	963.2	5941.8	234.9%	1774.0	86.0%	618.0	12.9	54.1%	28.8%
45	400	5	953.7	1001.4	3764.1	150.6%	1501.8	57.5%	830.0	9.6	34.1%	21.4%
45	800	5	953.7	1001.4	3549.9	152.9%	1403.9	47.2%	1296.0	6.2	20.4%	13.7%
100	100	1	953.7	963.2	1413.0	14.0%	1239.8	30.0%	239.0	33.5	44.0%	33.5%
100	200	5	953.7	1001.4	2631.2	19.6%	2200.1	130.7%	571.8	14.0	32.6%	14.0%
300	100	1	953.7	963.2	1060.0	2.1%	1038.4	8.9%	232.0	34.5	12.6%	11.5%
300	200	1	953.7	963.2	1083.0	2.3%	1059.1	11.1%	273.0	29.3	11.0%	9.8%
500	200	1	953.7	963.2	1068.9	1.7%	1051.5	10.3%	252.0	31.8	7.1%	6.4%
500	200	5	953.7	1001.4	1660.9	5.3%	1576.7	65.3%	539.1	14.8	5.0%	3.0%

Table 1: UDT file transfer over typical WANs – high-bandwidth cost and slow transfer rate

Bandwidth (Mbps)	RTT (ms)	plr (%)	How much data to be sent? (MB)	How much needs to be sent (actual data + inevitable loss by media, MB)	How much data actually sent? (MB)	Sending Cost (Sender's Overhead, %)	How much data actually received? (MB)	Receiving Cost (Receiver's Overhead, %)	How long does it take? (s)	Effective file transfer speed (Mbps)	Observed Network Utilization (by receiver)	Network Efficiency (Effective Utilization, %)
45	0	0	953.7	953.7	953.7	0.0%	953.7	0.0%	185.4	43.1	98.5	95.9%
45	100	1	953.7	963.2	963.3	1.0%	953.7	0.0%	187.8	42.6	97.1	94.6%
45	400	5	953.7	1001.4	1002.1	5.0%	954.3	0.1%	197.0	40.6	92.1	90.3%
45	800	5	953.7	1001.4	1003.5	5.1%	955.2	0.2%	197.0	40.6	91.6	90.3%
100	100	1	953.7	963.2	963.3	1.0%	953.8	0.0%	85.0	94.1	96.3	94.1%
100	200	5	953.7	1001.4	1002.4	5.0%	954.5	0.1%	88.9	90.0	91.9	90.0%
300	100	1	953.7	963.2	964.0	1.0%	954.4	0.1%	29.3	273.4	92.6	91.1%
300	200	1	953.7	963.2	964.7	1.0%	955.1	0.1%	29.2	274.3	91.9	91.4%
500	200	1	9536.7	9632.1	9635.0	1.0%	9539.0	0.0%	181.6	440.6	90.6	88.1%
500	200	5	9536.7	10013.6	10018.5	5.0%	9541.2	0.0%	186.9	428.0	88.0	85.6%

Table 2: Aspera FASP file transfer over typical WANs – near zero bandwidth cost and fast transfer rate

The direct consequences of UDT file transfer performance shown in Table 1 are that useful data does not get through the network at all, or that it does so at the price of network efficiency (shown in the eighth column of Table 1), which not only compounds the poor performance, but also causes a denial-of-service for other network traffic by saturating the bandwidth. Note that creating parallel transfers for higher sending rates and network utilization as employed in some UDT and TCP solutions only aggravates bandwidth waste and forces customers to invest in more bandwidth prematurely. The consequent improvement in network utilization and data throughput is little, but the resulting cost (Figure 8) is dramatically increased. Retransmission is increased by another 40 percent with two UDT sessions. For the same example (Table 1), UDT dumps as much as 13 GB to 15 GB data to network in order to successfully deliver a less-than-1 GB file. Solutions using parallel TCP or UDT transfers have similar or even worse performance as shown in Figure 8.

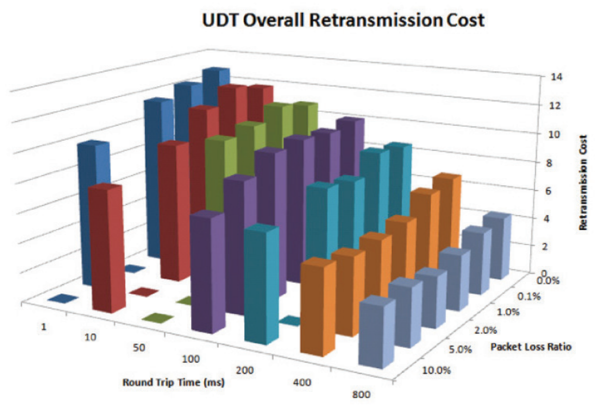


Figure 8: The graph shows the retransmission costs of two parallel UDT sessions for a single 1 GB file transfer under different RTTs and packet loss ratios in a T3 network. The height of each bar, referred to as transmission cost, represents the amount of retransmission in units of gigabytes when the 1 GB file is transferred. Bars with zero height represent failures of establishing connection between sender and receiver, which is not uncommon to UDT when either RTT or packet loss ratio is large. Note that almost 14 GB (14x) the size of the file is retransmitted in the process.

Aspera FASP solution

Aspera FASP fills the gap left by TCP in providing reliable transport for applications that do not require byte-stream delivery and completely separates reliability and rate control. It uses standard UDP in the transport layer and achieves decoupled congestion and reliability control in the application layer through a theoretically optimal approach that retransmits precisely the real packet loss on the channel. Due to the decoupling of the rate control and reliability, new packets need not slow down for the retransferring of lost packets as in TCP-based byte streaming applications. Data that is lost in transmission is retransmitted at a rate that matches the available bandwidth inside the end-to-end path, or a configured target rate, with zero duplicate retransmissions for zero receiving cost.

The available bandwidth inside the path is discovered by a delay-based rate control mechanism, for near zero sending cost. Specifically, FASP adaptive rate control uses measured queuing delay as the primary indication of network (or disk-based) congestion with the aim of maintaining a small, stable amount of “queuing” in the network; a transfer rate adjusts up as the measured queuing falls below the target (indicating that some bandwidth is unused and the transfer should speed up), and adjusts down as the queuing increases above the target (indicating that the bandwidth is fully utilized and congestion is eminent). By sending periodically probing packets into the network, FASP is able to obtain a more accurate and timely measurement of queuing delay along the transfer path. When detecting rising queuing delay, a FASP session reduces its transfer rate, proportional to the difference between the target queuing and the current queuing, therefore avoiding overdriving the network. When network congestion settles down, the FASP session quickly increases according to a proportion of the target queuing and thus ramps up again to fully utilize nearly 100 percent of the available network capacity.

Unlike TCP's rate control, the FASP adaptive rate control has several major advantages: First, it uses network queuing delay as the primary congestion signal and packet loss ratio as the secondary signal, and thus obtains the precise estimation of network congestion, not artificially slowing down over networks with packet loss due to the media. Second, the embedded quick response mechanism allows high-speed file transfers to automatically slow down to allow for stable, high throughput when there are many concurrent transfers, but automatically ramp up to fully, efficiently utilize unused bandwidth for more efficient delivery times. Third, the advanced feedback control mechanism allows the FASP session rate to more quickly converge to a stable equilibrium rate that injects a target amount of queued bits into the buffer at the congested router. Stable transmission speed and queuing delay bring QoS experience to end users without additional investment on QoS hardware or software. Delivery time of data becomes predictable and data movement is transparent to other applications sharing the same network. Fourth, the full utilization of bandwidth, unlike NACK based UDP blasters, introduces virtually no cost to the network and network efficiency is kept around 100 percent.

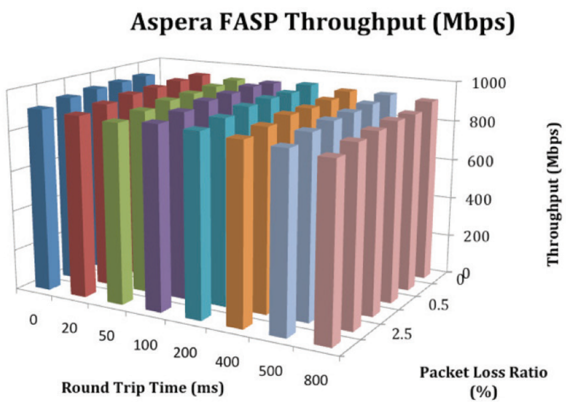


Figure 9: The bar graph shows the throughput achieved under various packet loss and network latency conditions on a 1 Gbps link for file transfer technologies that use FASP innovative transfer technology. Bandwidth efficiency does not degrade with network delay and packet loss.

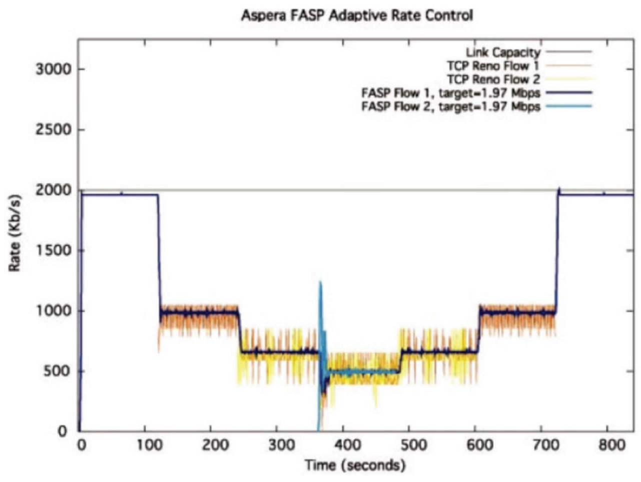


Figure 10: FASP shared link capacity with other FASP and standard TCP traffic, achieving intra-protocol and inter-protocol fairness.

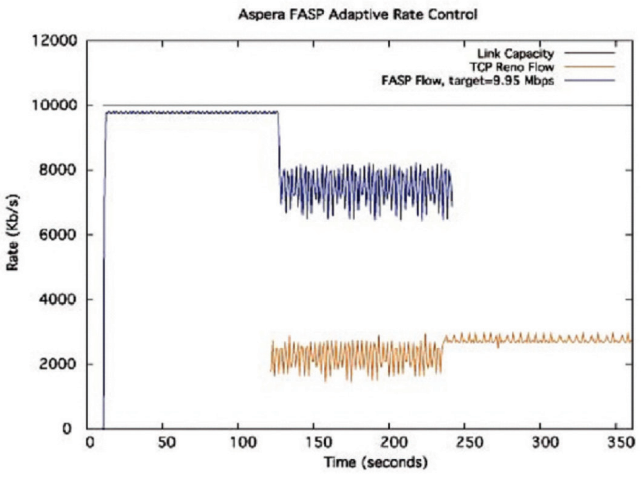


Figure 11: FASP uses available bandwidth when TCP is limited by network condition, achieving complete fairness between FASP flows and with other (TCP) traffic.

In addition to efficiently utilizing available bandwidth, the delay-based nature of FASP adaptive rate control allows applications to build intentional prioritization in the transport service. The built-in response to network queuing provides a virtual "handle" to allow individual transfers to be prioritized/de-prioritized to help meet application goals, such as offering differentiated bandwidth priorities to concurrent FASP transfers.

By removing artificial bottlenecks in network transport and freeing up full link bandwidth to end users, FASP transfers sometimes reveal newly emerging bottleneck points in Disk IO, file systems, and CPU scheduling, etc., which inevitably create new hurdles as the transmission rate is pushed to the full line speed especially in multi-Gigabit networks. The FASP adaptive rate control has been extended to include disk flow control to avoid data loss in fast file transfer writing to a relatively slow storage pathway. A similar delay-based model (patent-pending) was developed for the disk buffer. Due to the different time scales of network and disk dynamics, a two-time-scale design was employed to accommodate both bandwidth and disk speed changes. At a fine-grained, fast time scale, a local feedback mechanism is introduced at the receiver end to accommodate periodic disk slowdown due to operating system scheduling as an example, while at a coarse-grained, slow time scale, a unified delay-based congestion avoidance is implemented for both bandwidth control and disk control, enabling FASP transfers to simultaneously adapt to available network bandwidth as well as disk speed.

File system bottlenecks manifest in a variety of aspects. Indeed, many customers experience dramatically decreased speed when transferring sets of small files compared with transferring a single file of the same size. Using a novel file streamlining technique, FASP removes the artificial bottleneck caused by file systems and achieves the same ideal efficiency for transfers of large numbers of small files. For example, one thousand 2 MB files can be transmitted from the US to New Zealand with an effective transfer speed of 155 Mbps, filling an entire OC-3.

As a result, FASP eliminates the fundamental bottlenecks of TCP- or UDP-based file transfer technologies such as FTP and UDT, and dramatically speeds up transfers over public and private IP networks. FASP removes the artificial bottlenecks caused by imperfect congestion control algorithms, packet losses (by physical media, cross-traffic burst, or coarse protocols themselves), and the coupling between reliability and congestion control. In addition, FASP innovation is eliminating emerging bottlenecks from disk IO, file system, CPU scheduling, etc. and achieves full line speed on even the longest, fastest wide area networks. The result, we believe, is a next-generation high-performance transport protocol that fills the growing gap left by TCP for the transport of large, file-based data at distance over commodity networks, and thus makes possible the massive everyday movement of digital data around the world.

About Aspera, an IBM Company

Aspera, an IBM company, is the creator of next-generation transport technologies that move the world's data at maximum speed regardless of file size, transfer distance and network conditions. Based on its patented, Emmy® award-winning FASP® protocol, Aspera software fully utilizes existing infrastructures to deliver the fastest, most predictable file-transfer experience. Aspera's core technology delivers unprecedented control over bandwidth, complete security and uncompromising reliability. Organizations across a variety of industries on six continents rely on Aspera software for the business-critical transport of their digital assets.

For more information

For more information on IBM Aspera solutions, please visit www.ibm.com/cloud/high-speed-data-transfer.

[1] Jacobson, V., Braden, R., and Borman, D., *TCP Extensions For High Performance,* The Internet Engineering Task Force (IETF®), May, 1992, http://www.ietf.org/rfc/rfc1323.txt

[2] Ibid, 1992.

[3] Gu, Yunhong and Grossman, R., *Optimizing UDP-based Protocol Implementations,* 2005, http://udt.sourceforge.net/doc/pfldnet2005-v8.pdf