SANDVINE

TCP Optimization:

Opportunities, Key Performance Indicators, and Considerations

The Transmission Control Protocol (TCP) is the engine of the internet. Due to its ubiquitous use in video streaming, web browsing, file transfers, communications, and other application types, TCP typically represents 85% to 90% of fixed access internet traffic and as much as 96% of mobile internet traffic.

However, for a number of reasons explained in this document, TCP performance on today's networks is sub-optimal. Far from a minor inconvenience, this reality costs network operators enormous sums due to inefficient use of expensive resources, poor quality of experience (QoE), and other factors.

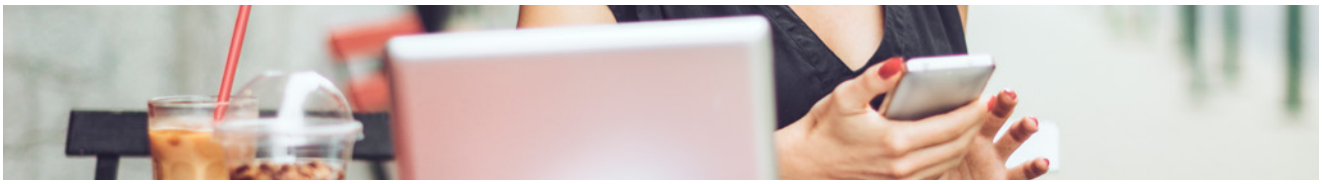
By taking control of TCP with an optimization solution, an operator can improve TCP execution in five ways, by:

- Reducing the time to reach available bandwidth
- Maintaining available bandwidth
- Adapting to changes in available bandwidth
- Handling packet loss in last-mile networks
- Handling congestion in last-mile networks

By doing so, an operator (and their customers) benefit from dramatically improved TCP performance, including:

- Higher average TCP transfer speeds
- Lower and more consistent TCP round-trip times
- Lower retransmission rates
- Increased goodput
- Improved network efficiency

By truly understanding the ways in which TCP performance can be optimized, the key performance indicators (KPIs) by which these optimizations can be measured, and the related considerations, a network operator will be in a position to make an informed decision and choose the most suitable solution.



INTRODUCTION TO TCP ACCELERATION

Due to its ubiquitous use in video streaming, web browsing, file transfers, communications, and other application types, TCP typically represents 85% to 90% of fixed access internet traffic and as much as 96% of mobile internet traffic[1].

However, despite TCP's omnipresence, network operators have no direct control over it. This lack of control is especially notable when compared against other aspects of the network:

- **Voice:** the voice network is planned, built, maintained, and optimized by the voice carrier
- **IP:** the last-mile infrastructure is planned, built, maintained, and optimized by the operator

With these networks, the network operator is in complete control of the architecture, the planning, the routing, etc. From an OSI layer perspective, for instance:

- **Layer 1:** Single-mode or multi-mode fiber? 10 GE links or 100 GE?
- **Layer 2:** MPLS? VLAN?
- **Layer 3:** How should we set up our BGP routes?
- **Layer 4:** The realm of TCP, is often overlooked. Why?

The likely culprit is that TCP is an end-to-end protocol, and the operator has no direct control over the endpoints[2].

It's important to acknowledge that TCP has enjoyed a long history of tremendous success, due in large part to the generic nature of TCP. Because TCP does not have end-to-end control, it makes almost no assumptions about the underlying network: a TCP sender neither knows nor considers if it's transmitting over satellite, GPRS, LTE, fiber, DOCSIS, DSL, etc. To succeed amid all this uncertainty, TCP has to be very conservative.

However, just because a TCP server can't know the last-mile method of transportation (or the subscription plan of the recipient, or other important factors), that doesn't mean that this knowledge can't be put to use by another element. As this paper explains, by taking control of TCP and by taking into account important network and environmental characteristics, TCP performance can be dramatically improved.

If TCP is so successful, then why would an operator ever want to exert control? It turns out that the very characteristics that made TCP so successful (e.g., generic behavior, conservative nature, relative simplicity) lead to sub-optimal performance in modern networks.

A BRIEF HISTORY OF TCP

TCP traces its origins back to the literal early days of the internet. In 1974, the IEEE published a paper by Vint Cerf and Bob Kahn that described a protocol for sharing resources using packet-switching among the nodes of an interconnected network[3].

The Transmission Control Program described within the paper was later divided into two parts[4]:

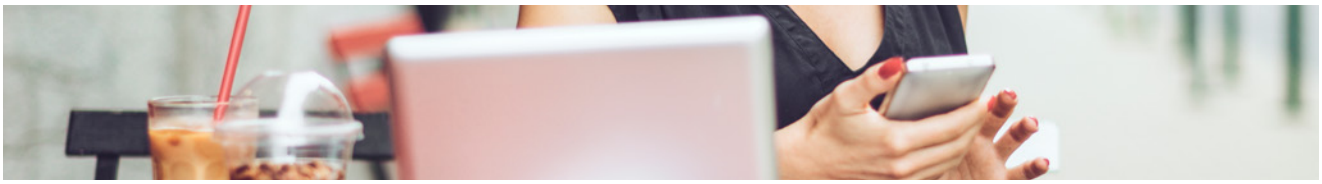
- The Transmission Control Protocol at the connection-oriented layer
- The Internet Protocol at the internetworking (datagram) layer

1 These statistics as of July, 2016, as measured during the course of preparing a Global Internet Phenomena Report

2 For the most part, but there are exceptions of course. For instance, an operator has control over content being delivered from an operator's server to a set-top box, but such circumstances represent a tiny portion of internet traffic.

3 "A Protocol for Packet Network Intercommunication"

4 This origin is also the reason why we still, today, say "TCP/IP"



TCP'S MAIN FUNCTIONS

The purpose of TCP is to provide efficient, reliable, ordered, and error-checked data delivery between endpoints over an IP network, while abstracting the connection details so that the endpoints don't need to know anything about the network links.

More specifically, TCP fulfills five functions to enable data transmission:

1. **Preserve Order:** TCP uses sequence numbers that correspond to each byte, which allows reordering if data arrives out-of-order.
2. **Retransmission:** Cumulative acknowledgement numbers and, later, selective acknowledgement (SACK) lets a sender know which data has successfully arrived at the receiver/destination.
3. **Rejection:** A checksum verifies headers and data payload; if the checksum indicates an error, then the packet is rejected. A rejection is interpreted by the sender as a packet loss.
4. **Receiver-Side Flow Control:** The receiver has control over how much data it is prepared to receive.
5. **Sender-Side Congestion Control:** The sender owns adjusting the sending rate in response to (perceived) network congestion.

Of the five functions outlined above, the first three focus primarily on transmission reliability, while the final two address transmission speed, efficiency, and fairness.

CONGESTION CONTROL ALGORITHMS

In the early years, things worked well, but as network speeds increased a problem emerged: congestion within the network led to widespread congestive collapse, a condition that reduced TCP throughput by orders of magnitude.

In response, TCP was bolstered with congestion control algorithms; these algorithms allow TCP to make adjustments to data sending rates in response to network conditions. An important consequence of TCP's origins on reliable, wired networks is a guiding assumption that packet drops were an indicator of network congestion.

Over the years, many TCP congestion control algorithms[4] (both open and proprietary) have been developed, in a seemingly endless game of catch-up in response to new access network technologies and internet trends.

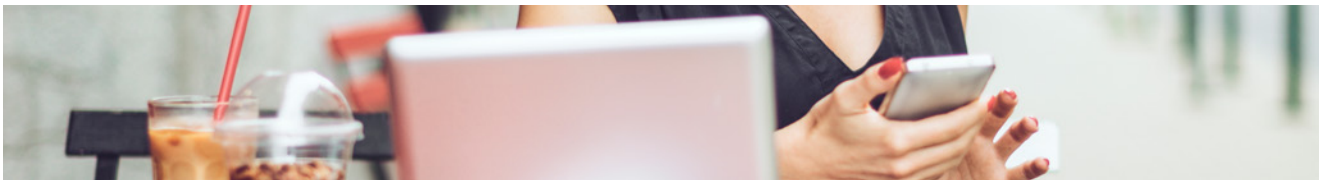
In general, these algorithms exist on a spectrum, with their general position based upon what they take as an indicator of network congestion (i.e., packet loss vs latency) and their more specific positions determined by their approaches to recovery from congestion.

Spectrum of TCP Congestion Control Algorithms

THE PROBLEMS WITH TCP

In modern networks, many algorithms are in use, but of course there's no guarantee that different endpoints support the same algorithms or TCP tuning options (the general behavior of these algorithms and function of the parameters are explained deeper in this document). Additionally, counter to the original assumptions when TCP was born, packet loss can't be trusted to indicate congestion (for instance, packets might simply be lost in the radio network) and congestion can't be assumed to cause packet loss (for instance, packets might be queued up in a memory buffer).

4 As expected, you can read about them over at Wikipedia: https://en.wikipedia.org/wiki/TCP_congestion_control



Furthermore, while the congestion control algorithms adjust their behavior in response to measured network characteristics, they lack direct knowledge of the network (e.g., access technology, dynamic capacity, etc.) and more 'big picture' characteristics (e.g., multiple TCP flows for a single user, many users on a shared access network, content being sent from multiple servers, etc.); as we shall see, direct knowledge opens up a range of optimization possibilities.

So, despite the undeniable evolution of TCP, improvements can still be made.

At a high level, the goal of TCP (and a TCP optimizer) is to make sure the sender's/server's in-flight data rate matches the bandwidth delay product (BDP) of the network. The BDP is the mathematical product of the instantaneously available network bandwidth and the network latency.[6]

Despite the best efforts of many talented researchers and engineers, because TCP lacks end-to-end control and visibility, there are times when the in-flight data rate falls below the BDP, and times when it exceeds the BDP.

When TCP is Too Slow

In many situations, TCP is too conservative (read: slow). For instance, a network might have capacity available right now, but a new TCP flow takes a little while to speed up. This conservative behavior wastes available capacity.

Or consider TCP's conservative response to what it interprets as signs of network congestion, like packet drops: this behavior is useful, generally, but there are times when TCP misinterprets normal network behavior and unnecessarily slows down. As with a new flow, it takes the existing flows a little while to speed up again, even when there's ample network capacity available right now.

This problem arises on all access networks, with varying impact to users. For instance, consider a few packets dropped in the radio network, which causes a TCP server to dramatically slow it's transmission; a user on a 3G network might get frustrated that her download drops from 15 Mbps to 7.5 Mbps to less than 4 Mbps to less than 2 Mbps and then slowly climbs back up; a user on an LTE network might be furious that his 60 Mbps drops to 7.5 Mbps... when all-the-while the radio access network (RAN) has capacity to spare.

In both of these cases, TCP itself becomes a bottleneck, and the enormously expensive infrastructure built by the operator isn't being utilized optimally.

When TCP is Too Fast

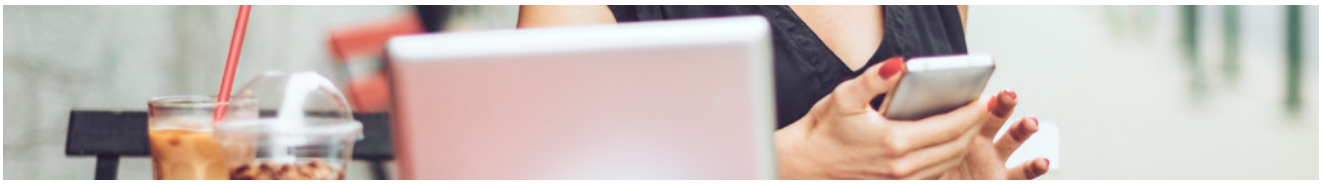
There are also situations when TCP sends data too fast, and by doing so it inadvertently overwhelms network resources (in particular, the buffers/queues in access network resources). When this situation – called bufferbloat – occurs, it results in a huge spike in latency and has a major negative impact on quality of experience, particularly for interactive applications that rely on real-time feedback mechanisms.

Additionally, micro-bursts[7] can instantaneously overwhelm buffers, leading to packet drops that are then (mis)interpreted by TCP congestion control algorithms as signs of network congestion.

Whether buffers are gradually or instantly overwhelmed hardly matters at all to users, who experience in both cases inconsistent speeds.

6 People are often surprised by how 'small' a BDP is, even on a fast network. To illustrate, consider an LTE network with a server close to the network core. If this network has a round-trip time latency of 25 ms and available bandwidth of 60 Mbps, then the BDP = 60 Mbps * 25 ms = 7,500 KBps * 0.025 s = 187.5 KB. That is, at any point in time, this network can only support 187.5 kilobytes of in-flight data for TCP – as a whole, across all concurrent TCP connections!

7 Wikipedia has a (surprisingly) short explanation at: https://en.wikipedia.org/wiki/Micro-bursting_(networking)



WHAT IS TCP OPTIMIZATION?

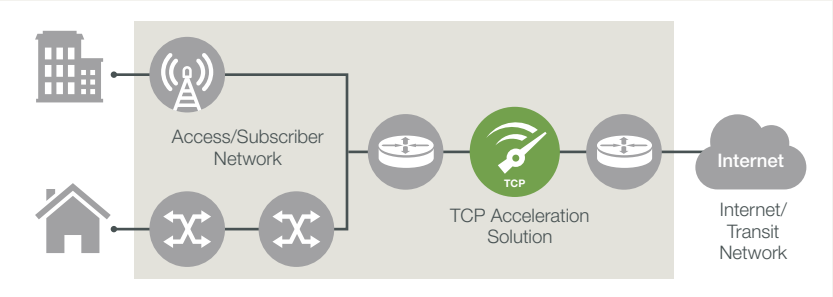
TCP optimization (often used synonymously with "TCP acceleration") is the name given to (and the desired result of) the application of a variety of techniques to 'take control' of TCP connections.[8] TCP optimization solutions come in a few different forms, for instance:

- "one-box" solutions that intersect the data path (see **Figure 1** below); broadly, these come in two types:
  - Terminating TCP proxies
  - Transparent TCP bridges
- "two-box" solutions that reside at two endpoints; these are best suited for environments when the service provider (or another entity) has complete control over both endpoints (for instance, WAN optimization for a multi-premise business), and won't be examined in this paper

Generally, a one-box TCP optimization solution is deployed between TCP endpoints, communicating with the server on behalf of the client, and with the client on behalf of the server. Such a solution can be deployed with complete visibility into a user's traffic, as opposed to any individual servers or CDNs, which only 'see' what they themselves are delivering. As such, a one-box TCP optimization solution applies network-specific knowledge to make TCP work better in the unique and dynamic network environment.

Figure 1

The TCP acceleration solution is deployed between TCP endpoints



Opportunities to Improve TCP Performance

In general, there are five 'opportunities' in which TCP acceleration can improve TCP performance[9]; these opportunities are described in the subsections below.

OPPORTUNITY 1: Reducing the Time to Reach Available Bandwidth

Because a TCP server doesn't know the characteristics of the network over which it is delivering content (e.g., last-mile access type, a user's subscription plan, the congestion level, etc.), TCP is designed to probe the network to determine the available bandwidth.

This phase, known as the TCP SlowStart, is represented in **Figure 2** (on the following page). A part of TCP's congestion control algorithm[10], TCP SlowStart attempts to determine empirically what the BDP is. In simple terms, this phase gradually sends increasing amounts of data until one of several conditions is met, as explained below, at which point it changes its behavior to suit the network characteristics.

SlowStart begins with a congestion window (cwnd) Size of one, two, or 10 segments.[11] The congestion window, maintained by the sender (i.e., the server) is one of the factors that determines how many segments can be outstanding (i.e., in-flight) at any point in time. In Figure 2, the starting Size is two segments.

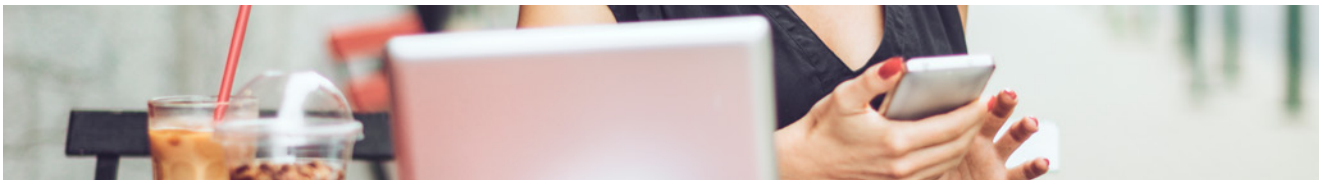
The value of the congestion window increases for each segment acknowledged, effectively doubling witih each acknowledgement packet (ACK).

8 Some solutions also include media optimization features (e.g., video trans-rating or trans-coding, image caching, etc.); strictly speaking, these techniques aren't TCP optimizations, and, in any case, the well-documented rise of encryption renders them useless. They won't be mentioned again in this paper.

9 There are additional areas in which TCP can behave sub-optimally, but these five are the ones that are most generally applicable and subject to optimization.

10 The algorithm is specified by RFC 5681.

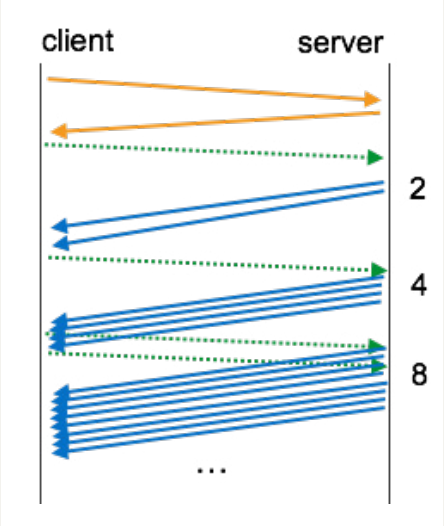
11 Over the course of a year, the initial congestion window was increased from 1 segment, to 2 segments, to 4 segments; in 2013, Google proposed allowing the initial congestion window to be set as high as 10, and this value has received widespread adoption.



In **Figure 2** below, the first ACK (represented by the dashed green line) completes the handshake, while the next acknowledges receipt of the first two data segments. Upon receipt of that second ACK, the congestion window doubles to four segments; upon receipt of the next ACK, the congestion window doubles again to eight segments.

Figure 2

TCP SlowStart, with a starting congestion window Size of 2 segments



This doubling continues until a packet loss is detected, the client's advertised receive window (rwnd) becomes the limiting factor, or the slow start threshold (ssthresh) is reached.

Typically, if a packet loss is detected, then the server interprets this loss as a sign of network congestion and reduces the data transmission rate – Congestion Avoidance. If ssthresh is reached, then the server switches from SlowStart into a linear growth algorithm, during which the cwnd increments by one segment for each round-trip time (RTT).

All of this explanation is to highlight a few things:

- the higher the round-trip time, the longer the ramp-up takes (whether measured in milliseconds or by the number of RTTs that have completed)
- the more bandwidth that's available (i.e., the ceiling to the transmission rate), the more RTTs it takes to ramp-up to maximum transmission rate
- the more RTTs it takes to ramp-up to maximum transmission rate, the more data is delivered during the course of ramping up

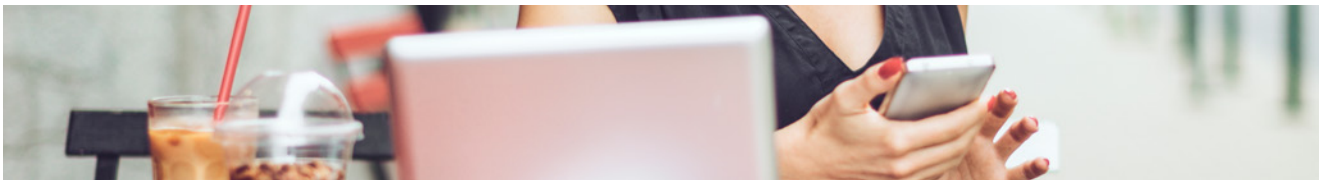
Ultimately, the maximum achievable TCP throughput depends primarily on two factors: the round-trip time and the amount of data to be delivered (because if a data transmission completes during the SlowStart phase, then the top speed was never reached). To illustrate this point, Table 1 shows a few examples.

Table 1: Example TCP Performance and Influencing Factors

Access Technology	Example Real-World Performance	Example RTT	Minimum File Size to Reach Speed
Cable	100 Mbps	10 ms	~1 MB
3G	15-25 Mbps	50 ms	1-1.5 MB
LTE	45-70 Mbps	25 ms	1-2 MB
LTE Advanced	250 Mbps	25 ms	7-9 MB

The time to reach available bandwidth is especially important for mobile operators, because the SlowStart phase all but guarantees that the radio access network and allocated radio resources are underutilized during the beginning of the connection.[12]

12 Network operators generally pay enormous spectrum license and maintenance fees, and invest huge sums to build and expand the radio access network – without realizing that sub-optimal TCP behavior can have a massively deleterious effect on network performance

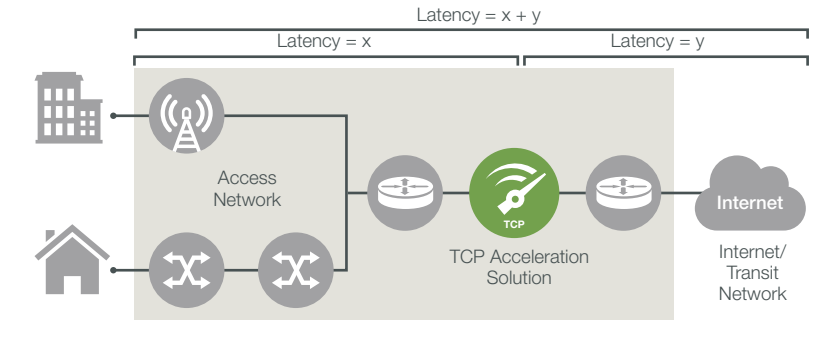


The access network latency is dependent upon the access infrastructure (e.g., DOCSIS, Fiber, 3G, LTE, etc.) and the state of the resources (e.g., congested, suffering from bufferbloat, capacity to spare, etc.), while the internet latency is dependent upon a whole host of factors, but mostly upon the geographic distance to the server.

TCP acceleration reduces the time to reach available bandwidth by 'splitting' the latency between the access network and the transit network (see Figure 3 below), and applying techniques to optimize the performance of each 'side' of the connection.

Figure 3

The TCP acceleration solution 'splits' the latency between access network and the transit network



While TCP acceleration benefits practically any network, the impact of latency-splitting is even more profound when the internet-side latency is large. Each of the examples in Table 1 assumes a relatively close TCP server; if the server was farther away, then the RTT would be higher and the minimum file size to reach speed would increase dramatically.

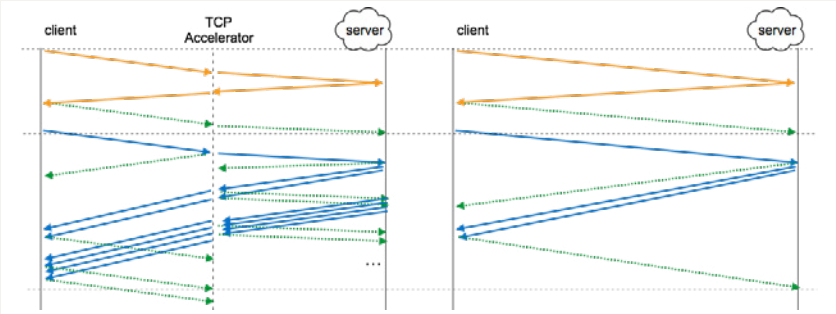
Because the congestion window is only increasing by one segment for every round-trip, a long round-trip means a painfully slow increase in the data sending rate. For example, if the cable example in the table instead used a server that is 150 ms away, then the minimum file size to reach 100 Mbps would need to be at least 20 MB.

Pre-Acknowledging Data

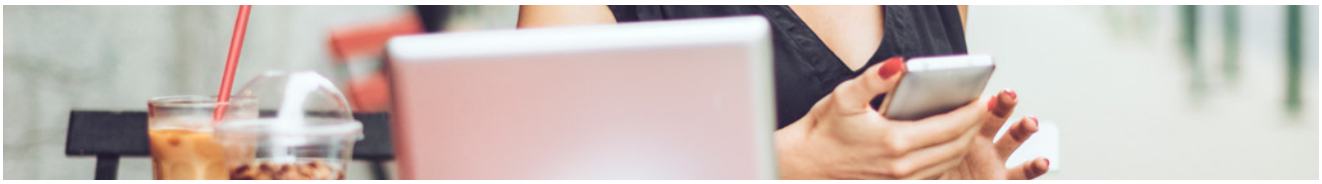
One technique to accelerate TCP flows relies on the TCP acceleration solution to pre-acknowledge data on behalf of the client (as pictured in **Figure 4** below).

Figure 4

TCP acceleration speeding up the SlowStart phase



Comparing the two diagrams at the point in time marked by the bottom horizontal dashed line, we see that with acceleration splitting the latency and ACKing on behalf of the client, all the data (i.e., six data packets) of the second RTT is already delivered; without acceleration, the ACKs of first RTT have only just arrived back at the server, and only two data packets have arrived at the client. Because the server sees the ACKs sooner, it sends data sooner, and the whole SlowStart phase is accelerated.



Were this the only mechanism by which SlowStart could be sped up, then it would only be of use in networks with relatively high access latency. Indeed, this is a common (erroneous) conclusion reached by netowrk operators who key in on the 'latency splitting' explanation.

However, some TCP acceleration solutions employ additional techniques to accelerate SlowStart, for even better results. For instance, techniques exist to ensure that the sender's cwnd growth during SlowStart is fully exponential, which is important because some TCP/IP stacks behave sub-optimally during this phase; furthermore, shielding the server from access-side network events can prevent an early transition from SlowStart to Congestion Avoidance.

Additionally, the TCP acceleration solution needs to ensure that the Receive Window advertised to the sender is large enough to match the network's BDP.

OPPORTUNITY 2: Maintaining Available Bandwidth

Once the available bandwidth is reached, for efficiency's sake (and for the sake of QoE) it should be maintained whenever the network can sustain it.

Unfortunately, TCP's congestion control algorithms misinterpret non-congestion events as congestion, and slow down unnecessarily as a result of the sender decreasing its congestion window. Typically, the erroneous slowdowns are triggered by packet re-ordering[13], jitter, or pauses (e.g., due to a cell handover in a mobile network), and non-congestion related packet losses (e.g., due to micro-bursts).

With respect to maintaining transmission speeds, the objective of a TCP acceleration solution is to ensure that the server does not unnecessarily decrease its sending rate and, if it does – for instance, due to internet-side packet loss – to help the sender recover quickly. In practice, this result is achieved by shielding the sender from the idiosyncrasies of last-mile networks (particularly wireless networks, where packet reordering and drops are a fairly normal occurrence).

Beyond maintaining consistent throughputs, such shielding increases network efficiency by preventing spurious retransmissions; with the right TCP acceleration solution in place, the retransmission rate is approximately equal to the actual packet loss rate.

OPPORTUNITY 3: Adapting to Changes in Available Bandwidth

All networks exhibit dynamically changing capacity as a result of changing demands, shared resources, and aggregation layers, and this variation presents another opportunity to improve TCP performance:

- If available bandwidth decreases, then the server should back off immediately
- If more bandwidth becomes available, then the server should use it immediately

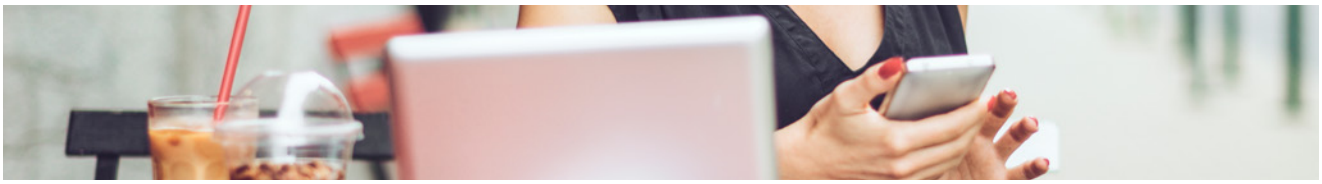
Once again, a TCP optimization solution can apply its knowledge (e.g., of the network, of the user's plan, etc.) to achieve these desired results, by leveraging its receive-side control of the server's sending rate.

OPPORTUNITY 4: Handling Packet Loss in Last-Mile Networks

In today's access networks, packet loss rarely indicates congestion. These last-mile networks are usually over-dimensioned and capacity is added before sufficient chronic congestion as to incur packet loss would appear.

That's not to say that congestion doesn't occur, but rather that it doesn't typically cause packet drops. When part of the access network does become congested, then increased queuing (i.e., buffering) within the access network resources appears first, well before actual packet loss; when part of the backhaul network becomes congested, then the best approach is to rely on an intelligent traffic management solution to match the aggregate traffic flow to the (dynamically changing) size of the bottleneck.

13 In practice, we routinely encounter packet re-ordering in radio networks



If the cause isn't congestion, then why do packets still go missing? In practice, there are many 'natural' causes, including:

- Interference and signal coverage issues, in wireless networks: essentially, giving up on attempts to retransmit on L1/L2
- Cell/Access point handovers
- Queue overflows due to excessive buffering (i.e., bufferbloat)
- Queue overflows due to micro-bursts
- Eth/IP/TCP checksum errors
- Damaged cables, faulty memory, or other hardware problems

If a packet is lost due to something other than genuine congestion, then there isn't any point in spending too much time recovering from it. But, as we have seen before, TCP is very conservative: left to its own algorithms, the recovery proceeds relatively slowly. A TCP acceleration solution can expedite this recovery in much the same manner that it can reduce the time to reach available bandwidth.

OPPORTUNITY 5: Handling Congestion in Last-Mile Networks

The previous section dealt with packet loss resulting from something other than congestion, but what can be done when congestion genuinely appears?

Congestion arises when the bandwidth into a resource (ingress) exceeds the available bandwidth out of the resource (egress). As mentioned above, networks today are engineered to avoid packet loss and, combined with the low cost of memory, this engineering goal has led to access network resources that have large buffers/queues that can temporarily store the excess data until it can be delivered. Packets are only dropped within the resource when the buffers overflow. However, while queuing significantly reduces packet drops, it comes at the cost of large delays in packet delivery, leading to increasing round-trip times. In fact, when properly calibrated[14], RTT is a great indicator of the degree of queuing in a network resource.

By preventing excessive queuing, a TCP acceleration solution can keep effective network latency low and can prevent the packet drops that result from buffer overlows.

OPPORTUNITY 6: The Need to Address QUIC

QUIC is a UDP based, Google provided protocol that has been designed to deliver data at lower latency rates than those of TCP. The protocol has gained some market traction with YouTube and other popular applications using it for transport.

QUIC does not work in a coordinated fashion with TCP, so it can have a serious and negative impact on TCP traffic performance. Left unchecked, QUIC has a tendency to overwhelm the same network buffers used by TCP, degrading the performance of TCP based services that are on their way to users.

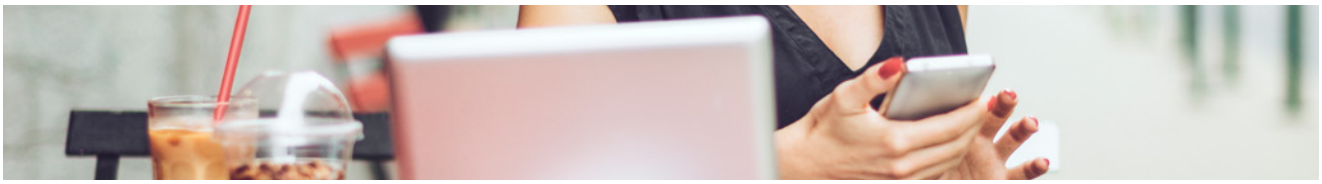
Before choosing a TCP optimization solution, make sure that the solution addresses both TCP and QUIC traffic management to ensure that applications running over both protocols can continue to perform as intended.

KEY PERFORMANCE INDICATORS

The previous section outlined six opportunities to improve TCP's performance in today's networks; this section examines five key performance indicators (KPIs) and their expected behavior when TCP is optimized.

The first two KPIs (TCP Transfer Speeds and TCP Round-Trip Times) follow directly from the objectives of TCP acceleration, while the last three (Retransmission Rates, Goodput, Network Efficiency) measure secondary effects. Other metrics will also show improvement, but this paper focuses on KPIs that can be directly applied to TCP.

14 Since every network is different, being built of different combinations of access resources supplied by different vendors



TCP Transfer Speeds

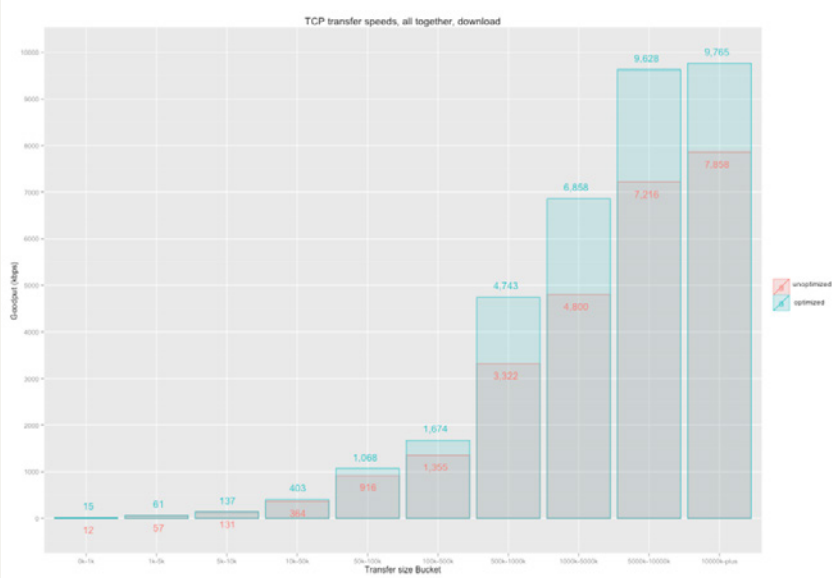
TCP optimization is meant to ensure that TCP isn't a bottleneck in the network, thereby allowing TCP to utilize available bandwidth. If this result is achieved, then (on average) TCP transfer speeds should increase, and that matters for a few reasons:

- Faster transfer speeds contribute to better quality of experience, whether from 'real' experience, from speed test utilities, or from users more frequently achieving advertised speeds
- Faster transfer speeds indicate more efficient utilization of the access network
- Faster transfer speeds contribute to transfers being completed sooner, freeing up access resources for other users (this point is especially important in mobile access networks)

Measuring the average TCP transfer speeds with and without TCP acceleration in place lets an operator compare the overall impact. For instance, **Figure 5** below shows that speeds in this European 3G/4G network improved by as much as 43% with TCP acceleration in place.

Figure 5

Impact of TCP optimization on TCP Transfer Speeds in a 3G/4G network



TCP Round-Trip Times

As described previously, there are circumstances in which TCP sends data too fast, overwhelms buffers/queues, and increases effective latency. Therefore, measuring the round-trip time lets an operator verify that latency is kept in balance by a TCP acceleration solution.

Maintaining low round-trip times is especially important for network operators, because high round-trip times (i.e., high latency) correlates significantly with poor QoE; in particular, this effect is true when the network is used simultaneously (e.g., by multiple users behind a router or NATing device, by multiple applications on a single client), as is almost always the case.

With the right TCP acceleration solution in place, the effective latency can be reduced to a level approximating the base access network latency. **Figure 6** (on the following page) shows the positive impact of TCP acceleration on the latency in a 2G network: with TCP acceleration enabled (blue), the Average RTT is cut in half and shows much less variation than when TCP acceleration is not enabled (red).

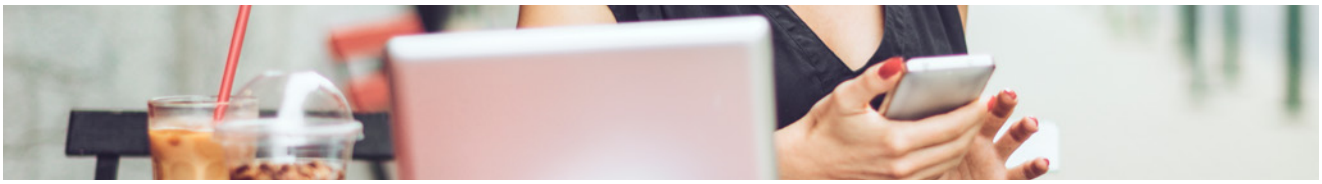
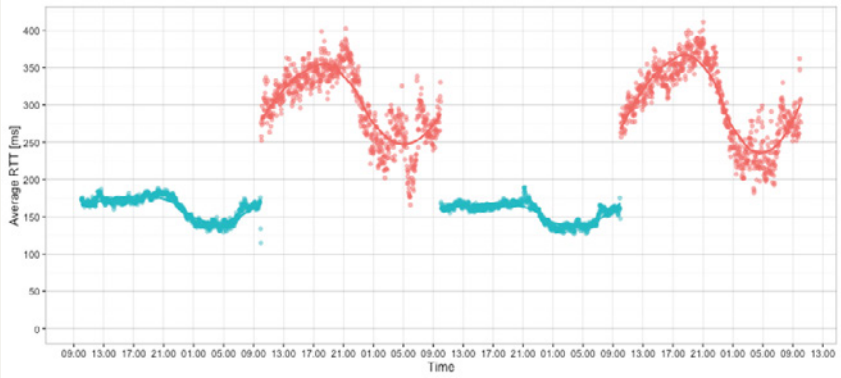


Figure 6

Impact of TCP optimization on Round-Trip Times



Beyond the improved average subscriber QoE, a secondary (but still very significant) benefit of both reducing RTT and improving its consistency becomes apparent with careful examination of **Figure 7** below. This figure contains five graphs, each of which shows the queuing time (Y-axis) versus the load (X-axis, expressed in percent of utilization) on a particular eNodeB in a mobile network. The blue dots plot the queuing time with TCP acceleration enabled, with the red dots showing the time without TCP acceleration enabled.

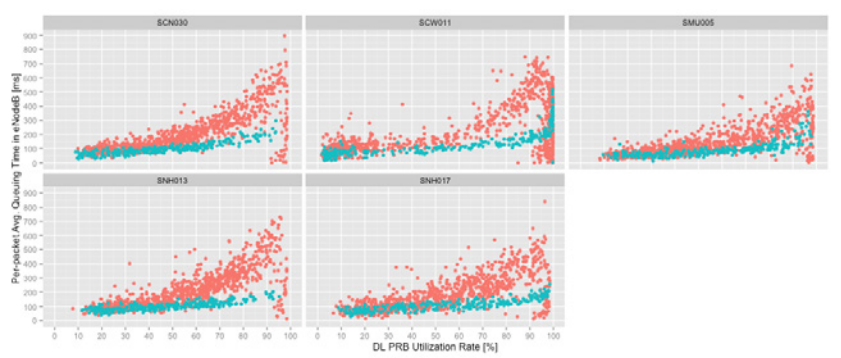
These graphs show that TCP acceleration maintains good latency even up to 95% of utilization. Without TCP acceleration, the latency at a much more moderate load (say, 60%-70%) exceeds the latency experienced by the accelerated samples when the eNodeB is at 95% utilization.[15]

Practically, this impact means that operators are able to safely run their access network resources 'hotter'; in the short term, doing so lets an operator accommodate short-term surges in demand and, in the long-term, lets an operator defer or redirect capital investments.

Retransmission Rates

Figure 7

Impact of TCP optimization on eNodeB queuing time versus resource load



As explained previously, many retransmissions are spurious (especially in wireless networks), rather than resulting from actual packet loss. With TCP acceleration in place – and able to better interpret network glitches – retransmission rates should fall.

Figure 8 (on the following page) shows the positive impact of TCP acceleration, as evidenced by the dramatically lower retransmission rate when TCP acceleration is enabled (blue) versus when it is disabled (red). Note, also, that the blue clusters exhibit greater consistency; by contrast, without TCP acceleration active, retransmissions surge during the network's periods of peak subscriber and bandwidth utilization – precisely the times when network efficiency is most critical to ensuring a positive QoE.

15 "What about the red dots in the bottom corner of the graphs?", you ask? That's a fair question. The "Per-Packet Avg Queuing Time" is calculated by dividing the sum of the total queuing time experienced by all packets by the total number of packets that the eNodeB transmitted, within a sample period. In the overload scenario (90%+ utilization), the eNodeB struggles to accurately count these basic measurements (for instance, dropped packets contribute towards the packet count but not to the queuing time), and the metric itself becomes meaningless.

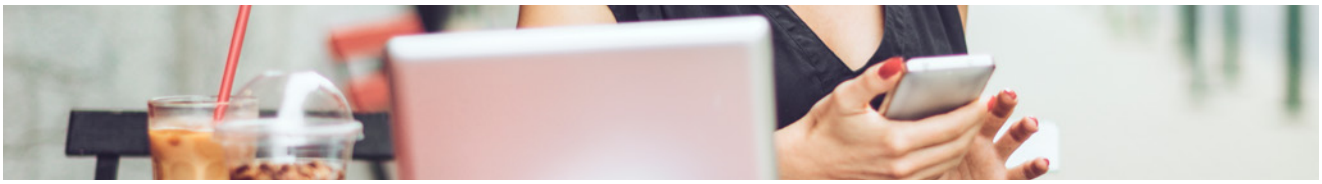
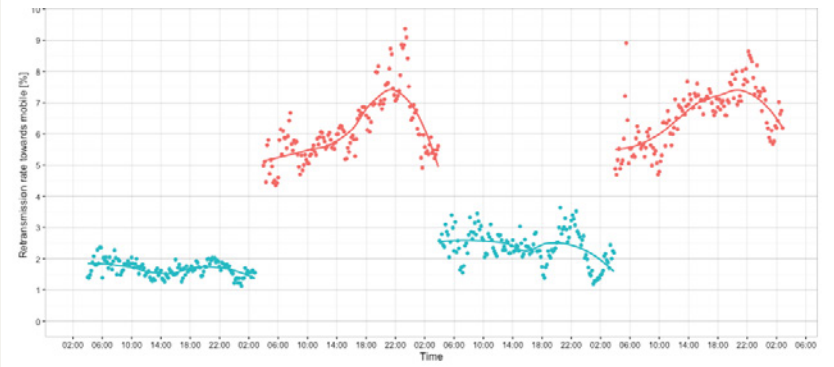


Figure 8

Impact of TCP optimization on Retransmission Rate



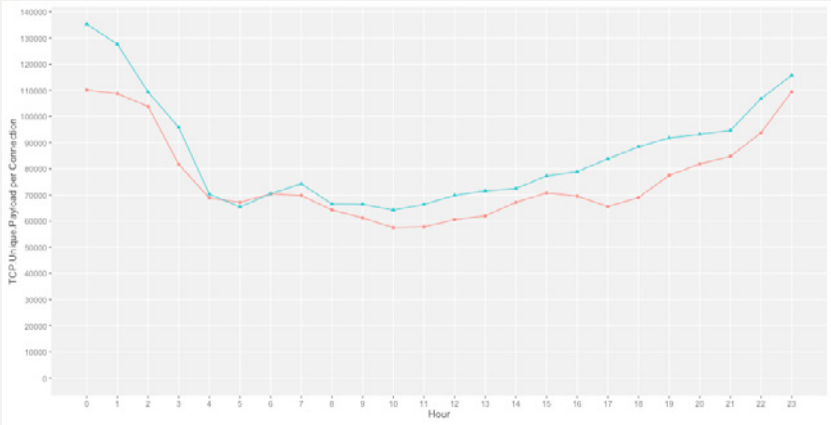
Goodput

Goodput, defined as the payload without retransmissions, can be monitored by measuring the unique application data volume per TCP connection. With retransmissions reduced, speeds increased, and other factors improved, TCP acceleration should result in increased goodput. **Figure 9** below shows what this result looks like: with TCP acceleration in place (blue), unique application data per TCP connection is consistently higher than when acceleration isn't active (red).

Network Efficiency

Figure 9

Impact of TCP optimization on Goodput

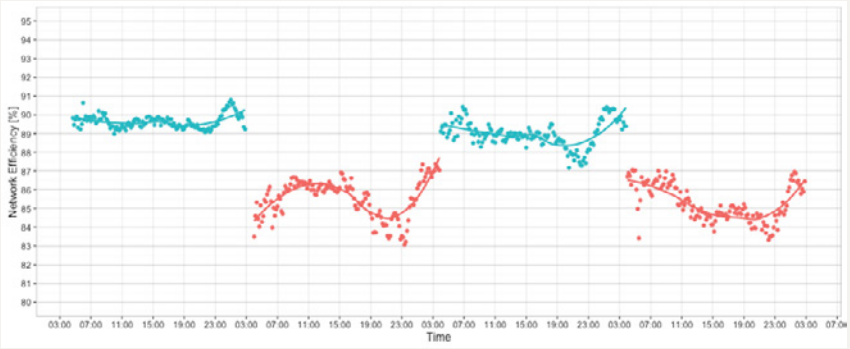


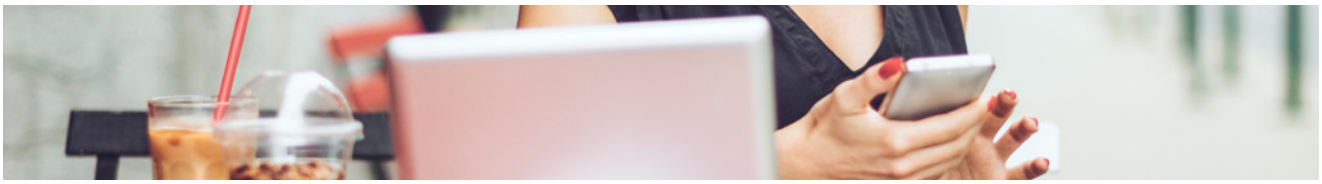
It follows that by increasing goodput as a proportion of total throughput, the overall efficiency of the network is improved, and that's what we see in **Figure 10** below: with TCP acceleration enabled, network efficiency rises to 90%, versus ~85% without TCP acceleration, and exhibits greater consistency. Both of these results have positive impacts for users and for operators.

ADDITIONAL CONSIDERATIONS

Figure 10

Impact of TCP optimization on Network Efficiency





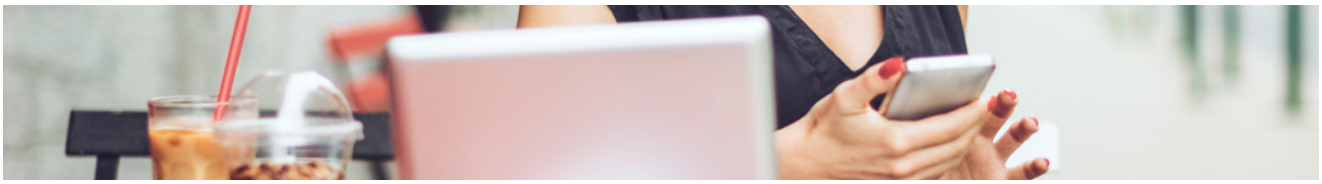
Having explored the opportunities to improve TCP's performance, and the KPIs that showcase its effect, it's time to look at additional considerations. Broadly, these can be divided into two families: General Considerations, and Solution-Specific Considerations.

GENERAL CONSIDERATIONS

A number of general considerations are explained in Table 2.

Table 2: General considerations when evaluating TCP acceleration

General Consideration	Description/Explanation
Should we simply use an open source TCP stack and build our own solution?	Many network operators, particularly those with skilled or powerful engineering or IT departments, inevitably wonder if they're not better off just building their own solution. After all, there are open source TCP stacks available, so why not just install one on a server, maybe tune it a bit, and voila. In practice, TCP acceleration products deliver meaningfully superior results over open source stacks and, beyond those significant benefits, the products include an assortment of other positives (e.g., much more rapid deployment timelines, dedicated support organization, engineered and optimized for deployment in networks, relatively free of organization politics, etc.). Of course, an operator should ask any potential vendors exactly how they stack up.
How does TCP optimization fit into our overall traffic optimization strategy?	TCP optimization is incredibly complementary to other traffic optimization solutions, including intelligent traffic management and content delivery networks. By combining these solutions, an operator can take advantage of the best of all worlds: TCP acceleration to operate on layer-4, traffic management to manage aggregate traffic profiles and ensure fair sharing of bandwidth during congestion events, and content delivery as close to the subscriber edge as is practical. The only caveat is that because TCP acceleration needs to take into account things like packet drops and network latency, some care must be taken to ensure the TCP acceleration solution and traffic management solution are working together, rather than inadvertently against each other. For instance, if a traffic management solution queues or deliberately drops packets, then the TCP accelerator might misinterpret these events. As is often the case, the recommendation is simply for operators to talk to their vendors.
I'm a participant in a Multi-Operator Core Network (MOCN) – if I use TCP optimization, then will that benefit my competitors?	Because TCP acceleration does not manage the aggregate amount of traffic flowing on the network, it's an ideal solution for MOCN arrangements in which multiple operators share/divide network resources. That is, unlike with traffic management solutions, an operator can deploy TCP acceleration without suffering from the unintended consequence of 'freeing up' capacity for the other network operators sharing the network.



SOLUTION-SPECIFIC CONSIDERATIONS

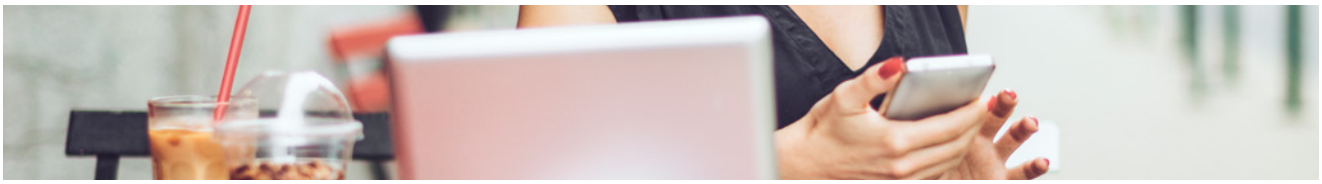
Besides the obvious considerations that emerge from the previous sections (e.g., "Please explain, for each of the five opportunities to improve TCP performance that were outlined above, how your solution does so."), a number of solution-specific questions should be answered by any TCP acceleration vendor. These questions are listed in Table 3.

Table 3: Solution-specific considerations when evaluating TCP acceleration

Solution Consideration	Description/Explanation
Is the solution layer-2 transparent, or does it terminate TCP connections?	There are arguments for and against either approach; at the very least, by asking the question the operator can comprehensively understand the proposed solution and its behavior in the network. To get the ball rolling, here are seven advantages of the transparent bridge approach, versus a terminating proxy: Transparent = Safe: Any level of non-transparency may break something in an obscure way.* For instance, a client makes a connection attempt and the terminating proxy answers it, even though the server is not available or does not exist. A terminating proxy must account for every single possible circumstance, whereas a transparent bridge is, well, transparent. While the goal is to optimize all TCP traffic, the real world contains all sorts of obscurities (e.g., VPN solutions that look like TCP but aren't, sensors with buggy hardware TCP/IP stacks, the expanding Internet of Things). A transparent solution can examine the SYN and SYN/ACK and then determine whether or not to apply optimization; a terminating proxy terminates and then is stuck with the consequences. A transparent solution will not block experimentation with, and adoption of, new TCP features.** If such a solution observes a new/unrecognized TCP option, then it does not attempt to optimize the connection. A transparent solution can opt-out of optimizing a connection at any time when the endpoints are synchronized, and the connection continues to work. Closely related to #4, a transparent connection is fail-safe: all synchronized connections continue to work even if the solution falls into physical bypass. It is much easier to upgrade a live transparent solution – there is zero network impact during maintenance work, so no maintenance windows are required. If a transparent solution is deployed in a network environment where there is asymmetric routing, then (unlike a terminating proxy) the routing is not broken.
Does the solution accelerate upstream traffic and downstream traffic?	While downstream traffic represents the majority of traffic by volume, upstream traffic shouldn't be ignored. Slow upstream connections are especially apparent to mobile users when they try to upload media to social media or transfer media via instant messaging applications.
Does the solution rely on any 'media optimization' or content caching to deliver results?	Media optimization solutions (e.g., trans-rating, trans-coding, image compression, etc.) and content caching (i.e., caches that attempt to cache a specific piece of content so that it can be delivered locally when requested again by someone else) rely on access to the content payloads, but the continued rise of content encryption renders such solutions essentially useless – if not immediately then in the near future. Network operators would be wise to avoid making any business investments based on claims of long-term media optimization or content caching efficacy.
Does the solution accelerate encrypted traffic?	True TCP acceleration (i.e., not media optimization or content caching) works on the TCP layer; it doesn't rely on any knowledge of the payload, so it doesn't matter if the payload is encrypted or not.
How does the solution scale?	Scalability – from small to large – is a concern in any telecommunications solution, but due to the memory and processing requirements of TCP acceleration, many solutions reach a ceiling that can severely limit the operator's deployment options.
How does the solution handle asymmetric traffic?	Closely related to the issues around scale are those around traffic asymmetry. Network operators should ask their potential vendors how the proposed TCP acceleration solution handles asymmetric traffic; otherwise, the deployment options or the effectiveness of the solution might be limited.

* For some great explanations and real-world obscurities, check out Juho Snellman's blog; in particular, Mobile TCP optimization – lessons learned in production: https://www.snellman.net/blog/archive/2015-08-25-tcp-optimization-in-mobile-networks/#slide-7

** For instance, see Network support for TCP Fast Open, by Christoph Paasch of Apple. In his study, he found that TCP Fast Open was prevented in 20% of the networks examined.



CONCLUSIONS

In modern networks, many TCP algorithms are in use, but of course there's no guarantee that different endpoints support the same algorithms or TCP tuning options (the general behavior of these algorithms and function of the parameters are explained deeper in this document).

Additionally, counter to the original assumptions when TCP was born, packet loss can't be trusted to indicate congestion (for instance, packets might simply be lost in the radio network) and congestion can't be assumed to cause packet loss (for instance, packets might be queued up in a memory buffer).

Furthermore, while the congestion control algorithms adjust their behavior in response to measured network characteristics, they lack direct knowledge of the network (e.g., access technology, dynamic capacity, etc.) and more 'big picture' characteristics (e.g., multiple TCP flows for a single user, many users on a shared access network, content being sent from multiple servers, etc.); as we shall see, direct knowledge opens up a range of optimization possibilities.

So, despite the undeniable evolution of TCP, improvements can still be made.

By taking control of TCP with a TCP optimization solution that takes into account important network and environmental characteristics, an operator (and their customers) benefit from dramatically improved TCP performance.

In general, there are five 'opportunities' in which TCP acceleration can improve TCP performance; these opportunities are described in the subsections below.

Table 4: The Five 'Opportunities' for TCP Optimization

Optimization Opportunity	Description/Explanation
Reducing the time to reach available bandwidth	Help TCP get up to maximum speed faster, during the SlowStart phase.
Maintaining available bandwidth	Consistently maintain the maximum throughput, by preventing TCP from reacting to non-congestion events.
Adapting to changes in available bandwidth	Speed up and slow down, as needed, in response to dynamic network conditions and events.
Handling packet loss in last-mile networks	There are many 'natural' reasons why packets go missing in today's last-mile networks, and this phenomenon is not limited to wireless access networks. A TCP optimization solution can insulate the sender from these events, preventing unnecessary slowdowns.
Handling congestion in last-mile networks	By preventing excessive queuing, a TCP acceleration solution can keep effective network latency low and can prevent the packet drops that result from buffer overflows.

There are five KPIs that can be used to measure the effectiveness and impact of a TCP optimization solution. The first two KPIs (TCP Transfer Speeds and TCP Round-Trip Times) follow directly from the objectives of TCP acceleration, while the last three (Retransmission Rates, Goodput, Network Efficiency) measure secondary effects.

Of course, other metrics and indicators (like general QoE) will also show improvement.

Table 5: The Five KPIs for TCP Optimization

KPIs		Description/Explanation
Direct Effects	TCP Transfer Speeds	If TCP is no longer a network bottleneck, then (on average) TCP transfer speeds should increase.
	TCP Round-Trip Times	There are circumstances in which TCP sends data too fast, overwhelms buffers/queues, and increases effective latency. Therefore, measuring the round-trip time lets an operator verify that latency is kept in balance by a TCP optimization solution.
Secondary Effects	Retransmission Rates	Many retransmissions are spurious, rather than resulting from actual packet loss. With TCP optimization in place – and able to better interpret network glitches – retransmission rates should fall.
	Goodput	With retransmissions reduced, speeds increased, and other factors improved, TCP optimization should result in increased goodput.
	Network Efficiency	By increasing goodput as a proportion of total throughput, the overall efficiency of the network is improved

Before choosing a TCP optimization solution, a network operator should consider some additional general and solution-specific factors, including: the overall traffic optimization strategy, any potential impact of internet encryption, solution scalability, and solution transparency.

By truly understanding the ways in which TCP performance can be optimized, the KPIs by which these optimizations can be measured, and the related considerations, an operator will be in a position to make an informed decision and choose the most suitable solution.

ABOUT SANDVINE

Sandvine helps organizations run world-class networks with Active Network Intelligence, leveraging machine learning analytics and closed-loop automation to identify and adapt to network behavior in real-time. With Sandvine, organizations have the power of a highly automated platform from a single vendor that delivers a deep understanding of their network data to drive faster, better decisions. For more information, visit **sandvine.com** or follow Sandvine on Twitter at **@Sandvine.**

SANDVINE

USA
2055 Junction Avenue
Suite Number 105
San Jose,
CA, 95131
USA

EUROPE
Svärdfiskgatan 4
432 40 Varberg,
Halland
Sweden
T. +46 340.48 38 00

CANADA
408 Albert Street,
Waterloo,
Ontario N2L 3V3,
Canada
T. +1 519.880.2600

ASIA
RMZ Ecoworld,
Building-1, Ground Floor,
East Wing Devarabeesanahalli,
Bellandur, Outer Ring Road,
Bangalore 560103, India
T. +91 80677.43333