ALBEDO Telecom
www.albedotelecom.com

TCP Throughput Test with xGenius: RFC 6349

Both the RFC 2544 and the eSAM tests are Ethernet and IP tests that provide the same result regardless of the application; they generate fixed frame rates and verify the ability of the network to transmit this traffic with a low impairment level.

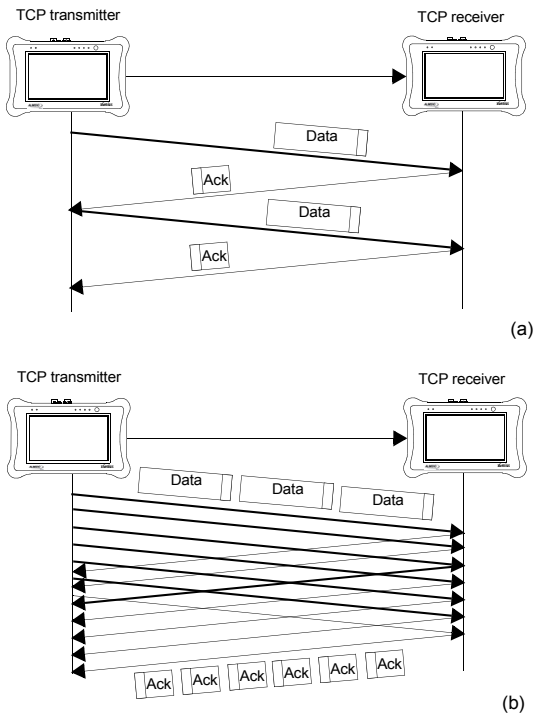

Figure 1.1: (a) No window is used: The transmitter has to wait for the acknowledge message to resume transmission. (b) TCP window mechanism: Continuous transmission can be achieved.

The RFC 2544 and eSAM approach is good for low latency applications (video, voice) based on the *User Datagram Protocol* (UDP) but it has proven to be insufficient for data transfer applications (web, file transfer, e-mail) which are highly sensitive to transmission errors and they are often based on the *Transmission Control Protocol* (TCP). TCP is more complex than UCP. Unlike UDP, TCP is connection oriented, which means that there are TCP handshaking and connection termination procedures. For this reason, TCP endpoints are required to keep status information about open connections. The entity that starts the communication by initiating the handshaking procedure is the TCP client. The TCP server, on the other hand, accepts connections from clients and responds to their queries while the connection is active.

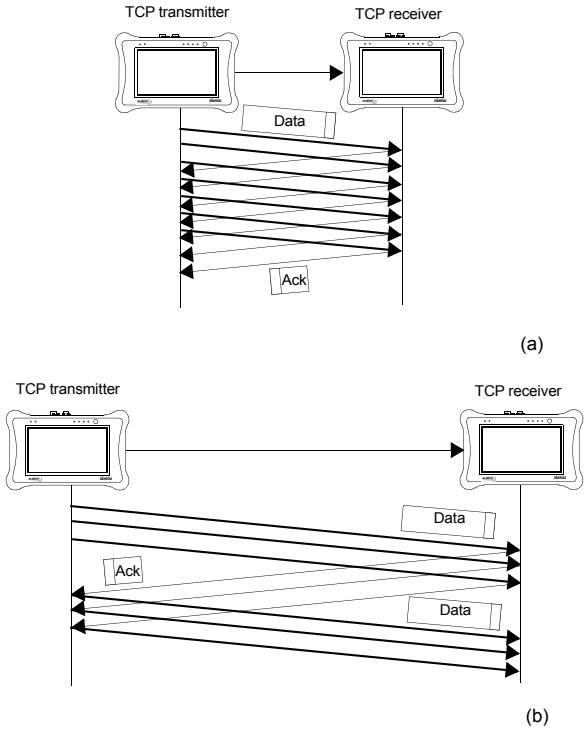

Figure 1.2: The importance of setting a correct window to achieve optimum performance in TCP data transfer: (a) The window is set to a value larger than the BDP and transmission is continuous. (b) If the window is smaller than the BDP the transmitter has to stop at some points and wait for acknowledge messages from the far end.

The mechanism that sets the UDP transmission rate is out of the scope of the protocol itself. The transmitter sets the speed to 100 kb/s, 1 Mb/s, 10 Mb/s or any value required by the application. If the network or the receiver is not able to process the traffic, they just start dropping packets. There is also no way to recover lost data unless a data recovery mechanism is implemented by the application. In TCP, the transmission speed is controlled by the receiver rather than the transmitter as in UDP. The TCP receiver controls which data is to be transmitted by generating acknowledge messages and the transmitter has to stop sending data if the previous messages have not been acknowledged. In order to increase the transmission efficiency, a sliding window mechanism is defined: The TCP transmitter is allowed to generate unacknowledged packets up to a certain limit specified by a window which is defined in terms of certain amount of data often expressed in KB. If the window mechanism is properly implemented, continuous transmission is achieved without the need to wait for each individual acknowledge message. The window mechanism is sensitive to network delay. If latency is increased beyond a limit, the transmitter does not have time to receive acknowledgement messages for transmitted data before the window is empty and it therefore must stop and wait. Something similar happens if the transmission speed increases because then the window empties faster. Given the transmission bandwidth and the *Round Trip Time* (RTT), there is a minimum window size that ensures that data transfer occurs without interruptions. This minimum window size is the *Bandwidth-Delay Product* (BDP). If the window size configured in the endpoint is smaller than the BDP, transmission will not achieve the optimum rate and the bandwidth will be partly wasted. For example, in a 1 Gb/s Ethernet path, with a delay (RTT) of 500 µs, the BDP is 61 KB. If the transmitter configures a transmission window of 16 KB, then the maximum achievable bandwidth will be 262 Mb/s.

The TCP sliding window mechanism allows the TCP to fill the line with a continuous data flow but it is also flexible enough to adapt the flow to varying transmission conditions. When used together with *Automatic Repeat Request* (ARQ), the function that enables TCP to detect and re-transmit lost or corrupted data, the sliding window becomes also a congestion management mechanism that enables the transmitter to adjust the window size on detecting packet loss in order to speed up or down data transfer. There are, actually, many different implementations of the TCP protocol. They differ on how they manage retransmission of lost / corrupted data, on congestion control and in other aspects. The xGenius test unit is based on a TCP implementation known as NewReno.

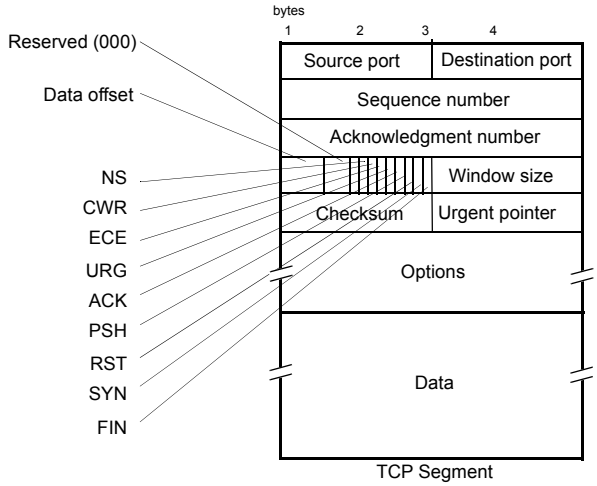

Figure 1.3: TCP segment format

Despite offering potentially different performance levels, all TCP implementations are required to be inter-operable. One of the key aspects of compatibility between them is that they share the same segment structure, which is defined in RFC 793 and it was later complemented by RFC 3040 and RFC 3168. The TCP header is 20 bytes long when no options are appended to it. This length has to be taken into account when the transmission efficiency is rated. For example, in a 1 Gb/s Ethernet link, if TCP is transported over the IPv4 protocol, it achieves at most 949 Mb/s (1518 byte frames). This is because an Ethernet 1 Gb/s interface includes *Maximum Transmission Unit*

(MTU) frames (1518 bytes), the preamble (8 bytes), and the *Interframe Gap* (IFG) (12 bytes), which totals 1536 bytes. However the TCP payload is the MTU minus the Ethernet header (14 bytes), the Ethernet *Frame Check Sequence* (FCS) (4 bytes), the IPv4 header (20 bytes) and the TCP header (20 bytes). Finally, only 1460 bytes are left for user data transmission. In a lossless (no retransmissions) channel, if the TCP sliding window mechanism is properly implemented, the transmission efficiency is the ratio between 1460 bytes and 1518 bytes, which accounts for 94.9% (949 Mb/s in a 1000 Mb/s link). In our previous example about transmission over a 1 Gb/s path with a delay of 500 µs and a window of 16 KB, the real transmission capacity must take into account the frame-to-payload ratio. The result is the product of 262 Mb/s and 94.9% which is only 249 Mb/s.

The previous examples have shown that incorrect TCP settings lead to drastic reduction of net transmission bandwidth. The purpose of the RFC 6349 is to help network administrators to deal with issues that are specific of TCP transmission.

The last point in this short introduction to TCP and the RFC 6349 remembers that this test alone may not be sufficient to provide an accurate diagnostic of the transmission performance and it is often required to run a complementary RFC 2544 or eSAM test before investigating the TCP throughput. Proceeding in this way is beneficial to detect potential issues in the data link and network layers before investigating the transfer layer

1. THE XGENIUS RFC 6349 TEST

xGenius has the ability to generate TCP flows to verify the protocol performance in terms of the metrics defined in standard RFC 6349. Users have control on how the TCP traffic is generated, the metrics to be evaluated and the performance thresholds required to declare a *Pass* or *Fail*.

The RFC 6349 test is made up of three different tests. These tests are described in the following list:

- *Baseline RTT test*: Measures the baseline RTT, the minimum time it takes for a TCP packet to travel from the near end to the far end and back to the near end. The baseline RTT could be also defined as the delay associated to the network under test when it is not experiencing congestion. The test unit generates a low speed TCP connection to measure the baseline RTT a low speed TCP with the purpose of avoid network congestion. The baseline RTT test could be disabled by the user, which is then requested to configure this parameter manually.
- *Window sweep test*: Measures the TCP throughput for different window sizes smaller or equal to the BDP and compares the theoretical and actual results to detect potential issues.
- *TCP throughput test*: For the BDP window, it rates the TCP protocol in terms of Throughput, the transmission speed achieved during the test, the Efficiency, which is related with presence of lost messages and retransmissions and the Buffer Delay which accounts for the latency increase due to congestion.

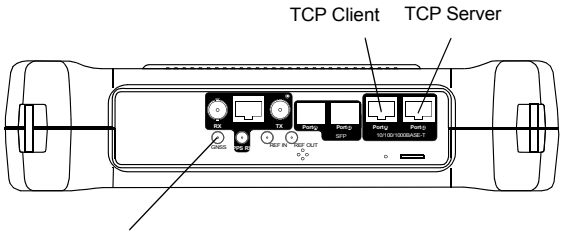

Figure 2 *Albedo xGenius test and clock reference interfaces used in this testing scenario.*

The following sections are a step-by-step description about how to configure and run an RFC 6349 test. To illustrate the ideas behind the test a very simple configuration has been chosen: The test runs in the so called *local bidirectional* mode, which means that the TCP server runs in *Port B* and the client is located in *Port A*. Traffic is bidirectional and flows both from *Port A* to *Port B*

and from *Port B* to *Port A*. Initially, xGenius is considered to be configured to the factory defaults.

Connecting the Unit

Connect the *Port A* RJ-45 port to the network interface where TCP client is going to run and *Port B* to the network interface identified as the server end. Follow these steps to enable the IP protocol in the port:

1. From the *Home* panel, go to *Config*,
   The port configuration panel is displayed.
2. Select *Mode* to enter in the mode selection menu.
3. Choose *IP endpoint* and confirm.

Configuring the Port

When the equipment is configured in *IP Endpoint* mode it behaves like any other IP host in the network. That means that the test unit should be able to receive traffic directed to it or transmit test and signaling traffic to other devices. To do that, users should assign an IP profile to each test port: IP address, network mask, gateway address and DNS address.

The following procedure assumes that most simple test port configuration is used (1000BASE-T interface, no VLANs). Additional settings are required for more complicated connections.

1. From the *Home* panel, go to *Config*,
   The port setup panel is displayed.
2. Select *Port A*.
3. Enter in *Local profile*.
4. Decide whether you want to configure the local IP profile with the help of the DHCP protocol or you want to statically set these profiles by means the *Use DHCP* control.
5. If you have enabled DHCP wait for the tester to get an IP profile from a DHCP server. If DHCP is not enabled, enter a valid *Static IPv4 address*, *Static IPv4 network mask*, *Static IPv4 gateway* and *Static IPv4 DNS address*.
6. Repeat the steps from 1 to 5 but replace *Port A* by *Port B* in step 2.

Configuring the TCP Throughput Test

Now that the test unit is connected to the network and the test interfaces are configured, it is ready to generate and receive traffic. This traffic is configured by following these steps:

1. From the *Home* panel, go to *TEST*,
   The test configuration panel is displayed.
2. Go to *Performance test* and configure *RFC 6349*.
3. Go to *RFC 6349 configuration*.
4. Optionally, configure the *Server TCP port* and the *Path MTU*.
5. Make sure that *Enable RTT test* is configured to *Enabled*.
6. Configure the baseline RTT test duration with the help of the *RTT test duration* control.
7. Enable the window sweep test through the *Enable window sweep test* field.
8. Configure *Step duration* and *Number of steps* to adjust the sweep test duration.
9. Configure the *Expected IR upstream* and *Expected IR downstream* in accordance with the test interface.
10. Leave the *RFC 6349 configuration* and *Performance test* panels.
11. Go to *Test mode*
12. Configure *Test method* to *Local bidirectional*.

Setting the Performance Objectives

The TCP Throughput test based on the standard RFC 6349 provides results in terms of three metrics: Throughput, Efficiency and Buffer delay. All three are described in the RFC 6349 standard:

- *Throughput*: Represents the maximum achievable bit rate between the client and the server entities (or between the server and the client). Only the TCP payload capacity is taken into ac-

count in this results. The Ethernet related fields (IFG, preamble, header, and FCS), IP header and TCP header are not taken into consideration. For this reason, the theoretical TCP throughput is smaller than the nominal L1 channel capacity. For example, a 1 Gb/s Ethernet, has a maximum TCP throughput of 949 Mb/s (1518 byte frames).

- *Efficiency*: It is defined as the ratio of successfully transmitted bytes to the total number of transmitted bytes. Efficiency decreases when data has to be retransmitted and the more retransmissions are necessary, the smaller the TCP efficiency. This parameter is closely related with frame loss.
- *Buffer delay*: This parameter is the ratio between the RTT computed during the throughput test and the inherent baseline RTT. This parameter is sensitive to waiting time in intermediate buffers and congestion.

The test unit provides a *Pass* or a *Fail* result for each of these parameters based on thresholds users can set at their will. The test unit provides one or two results for the throughput, efficiency and buffer delay depending on the test method. One of the results correspond with the upstream and the other with the downstream. The thresholds apply to both of them.

To configure the RFC 6349 thresholds, follow these steps:

1. From the *Home* panel, go to *TEST*,
   The test configuration panel is displayed.
2. Go to *Performance test* and make sure that the *RFC 6349*.is configured.
3. Go to *RFC 6349 objectives.*
4. Configure the value of *Minimum throughput*, *Minimum efficiency* and *Maximum buffer delay*.
   Note: You can set any of these values to 0 if you want them not to generate a *Pass / Fail* result and not to participate in the global RFC 6349 Pass / Fail result.

Retrieving the Results

Once the RFC 6349 test has been configured and the performance objectives are adjusted, the test is ready to run. To start execution press *run* at any moment. You can wait to the end of the test do check the results but you can also get a partial view of these results during test execution. These are the steps required to and display results:

1. From the *Home* panel, go to *RESULTS*,
   The test port results panel is displayed.
2. Select *Port A* to enter in the port specific results.
3. Select *RFC 6349*.
4. Check the *Global status*, *Current test, Current test direction* and *Remaining* fields to get an overview of the current test status.
5. Go to *RTT test* to know the *Baseline RTT* and *Window size*.
6. Go to *Window sweep* and check the *Connections*, *Theoretical throughput* and *Throughput* for each transmission window.
7. Go to the *Throughput* test panel and check the *Status*, *Window size*, *Connections*, *Throughput*, *RTT*, *Efficiency* and *Buffer delay*.