TCP Window Scaling

In this lesson, we'll discuss TCP window scaling!

We'll cover the following

Problem: Small Windows Result in Inefficient Use of Bandwidth

From a performance point of view, one of the main limitations of the original TCP design is that the 16-bit **window size** header limits the TCP receive window size to 2^{16} bytes i.e., $65,535$ bytes. That means the sender can send at most $65,535$ bytes before it has to wait for an acknowledgment.

Round Trip Time vs. Bandwidth vs. Throughput

- **Round Trip Time** is the amount of time it takes to send a packet and receive its acknowledgment.
- **Bandwidth** is the rate at which the network can transport the bits.
- **Throughput** is the amount of data that is *actually* transferred from one end-system to another.

So what if the round trip time is short enough to accommodate sending more data without having to wait for acknowledgments? The table below shows the rough maximum throughput that can be achieved by a TCP connection with a

64 KBytes window as a function of the connection's round-trip-time:

RTT	Maximum Throughput
1 msec	524 Mbps
10 msec	52.4 Mbps
100 msec	5.24 Mbps
500 msec	1.05 Mbps

This limitation was not a severe problem when TCP was designed, because at the time the available bandwidth was 56 kbps at best. However, in today's networks where the bandwidth can be in order gigabytes, this limitation is not acceptable.

Transmission and Propagation Delays: Example

- Consider a 1.544 **Mbps** link from one end host to another over a **satellite** link that is 36000 **km** above the surface of the earth.
- It takes $\frac{2 \times 72000 \times 1000}{3 \times 10^8}$ = 480 ms to get to the end host and then for the acknowledgement to return (ignoring the transmission time of the ack). This is the total distance to be covered by two segments divided by the speed of light.
- In the same amount of time, the sender could put $\frac{480 \times 1544000}{1000}$ = 741120 bits = 92640 bytes = 90.5 kB on the network.
- Now, even if the sliding window is at its maximum of 64 kB, the sender will only transmit 64 kB and then wait, sitting idle until an acknowledgement is received from the other side.
- That is a network utilization of $\frac{90.5}{64}$ = 70.71%. If we factor in the overheads of headers from various layers, the utilization drops further.

Note that we don't use satellite links much these days, but effectively, whenever the product of bandwidth and delay is high, which is common with today's high speed networks, we face the same problem. So, if the delay is

small, but the bandwidth is high, the sender can still put out a lot of bytes really quickly on the wire and still have to wait for an ACK sitting idle.

Solution: Larger Windows

To solve this problem, a backward-compatible extension that allows TCP to use larger receive windows was proposed in RFC 1323.

Basic idea: instead of storing the size of the sending window and receiving window as 16-bit integers in the TCB, we keep the 16-bit window size, but introduce a multiplicative scaling factor.

Scaling Factor

As the TCP segment header only contains 16 bits to place the window field, it is impossible to copy the size of the sending window in each sent TCP segment. Instead, the header contains:

$$sending\ window << S$$

where S is the scaling factor (0≤S≤14) negotiated during connection establishment and is specified in the header options field. '$<<$' is the bitwise shift operator. This operation pads zeros at the right and discards the bits on the left essentially multiplying by 2 for each shift.

Deciding a Scaling Factor

The client adds its proposed scaling factor as a TCP option in the SYN segment. If the server supports scaling windows, it sends the scaling factor in the *SYN+ACK* segment when advertising its own receive window. The local and remote scaling factors are included in the TCB. If the server does not support scaling windows, it ignores the received option and no scaling is applied. So scaling only applies when both parties support it.

Note that the protection mechanism of not maintaining state from the *SYN* packet via *SYN* cookies has the disadvantage that **the server wouldn't remember the proposed scaling factor**.

Improvement

By using the window scaling extensions defined in RFC 1323, TCP implementations can use a receive buffer of up to 1 GByte. With such a receive buffer, the maximum throughput that can be achieved by a single TCP entity is delineated in the following table:

RTT	Maximum Throughput
1 msec	8590 Gbps
10 msec	859 Gbps
100 msec	86 Gbps
500 msec	17 Gbps

These throughputs are acceptable in today's networks as there are already servers with 10 Gbps interfaces.

Quick Quiz!

1

Given the window scaling factor of 6 and window size of 3125, how many bytes can the sender send without waiting for an acknowledgement?

A) 100000

B) 200000

C) 2000000

COMPLETED 0%

1 of 2

Did you know that you can measure the round trip time on your network? Yes, it’s true. Let’s see how in the next lesson.