Network and Cybersecurity
Assignment 2

Kristoffer Klokker

2022

Contents

1 Problem - Congestion Control in TCP

1.1 Describe TCP Cubic congestion control. Include a comparison to earlier versions (TCP Reno or TCP Tahoe), and explain why Cubic is chosen as the default in many operating systems today

TCP congestion controls a limit on the amount of traffic send by a TCP connection. This limit is known as the congestion window and is denoted *cwnd*.
cwnd is a variable used to getting the finding the limit of data through the network. The sending rate will roughly be equal to $\frac{cwnd}{RTT} bytes/sec$. To find the *cwnd* value different approaches, TCP Reno, TCO Tahoe and Cubic all uses the first phase slow start.

1.1.1 Slow start

Slow start works by setting $cwnd = 1$. When an acknowledgment then is gotten *cwnd* is increased by 1 maximum segment size (MSS).
This will therefore create an exponential growth.
Once a loss is detected a new variable slow-start threshold *ssthresh* is set equal to $cwnd/2$. Then *cwnd* is set to 1 and slow start is repeated.
In the event that *cwnd* then get equal or higher than *ssthresh* it will continue to next phase Congestion avoidance.

1.1.2 Congestion avoidance

Congestion avoidance works by using a different growing rate for *cwnd* such the last maximum throughput W_{max} will be hovered around.
For TCP Tahoe and Reno the *cwnd* is incremented every RTT.
For TCP Cubic a variable k is chosen which is the time which it should take a cubic function to reach the last known W_{max}.
In case of a loss the *ssthresh* value is set equal to half *cwnd* and depending on type of congestion control different approaches is done.
In case of a loss (triple duplicate ACKs or timeout) TCP Tahoe set *cwnd* to 1 and goes to slow start.
For TCP Reno and Cubic in case of timeout the $cwnd = cwnd/2$.
In case of triple duplicate ACKs TCP Reno and Cubic goes to the Fast Recovery phase and set $cwnd = ssthresh + 3MSS$

1.1.3 Fast recovery

For every duplicate ACKs that follows the *cwnd* is increased by 1.
This is done until the missing package ACK is received, an it returns to the Congestion avoidance phase.
In case of a timeout *ssthresh* is updated to half *cwnd* and *cwnd* is set to 1, from which Slow start is performed.

1.1.4 TCP Cubic's advantages

TCP Cubic has the same advantages over TCP Tahoe as TCP Reno.
By not setting *cwnd* to 1, a faster recovery is done in slow start. Moreover has the Fast recovery which helps not doing a complete reset in case of duplicate and faster recovers to the expected W_{max}.
TCP Cubic has the advantage over TCP Reno by using a smart congestion avoidance function. By using a cubic function will it be assumed that limit for W_{max} will be constant and therefore faster reach the constant and stay around it. In case of W_{max} had gotten bigger, then the cubic function will grow faster than the linear functions and faster reach the new W_{max}.

2 Problem - RDI and Pipelining in TCP

The scenario is:

- There are 6 data segments (Numbered 1-6)
- The first time an ack if sent for segment 2, it is lost
- The first time segment 4 is sent, it is lost
- The window size is fixed at 3

2.1 Draw a timing diagram for RDT pipelining in TCP given the scenario described above. Make sure your diagram show when data is delivered to the application layer. Include a description where you point out the TCP specific elements.

In the diagram 1 is buffer colors representing:

- Darkblue is the sent segments

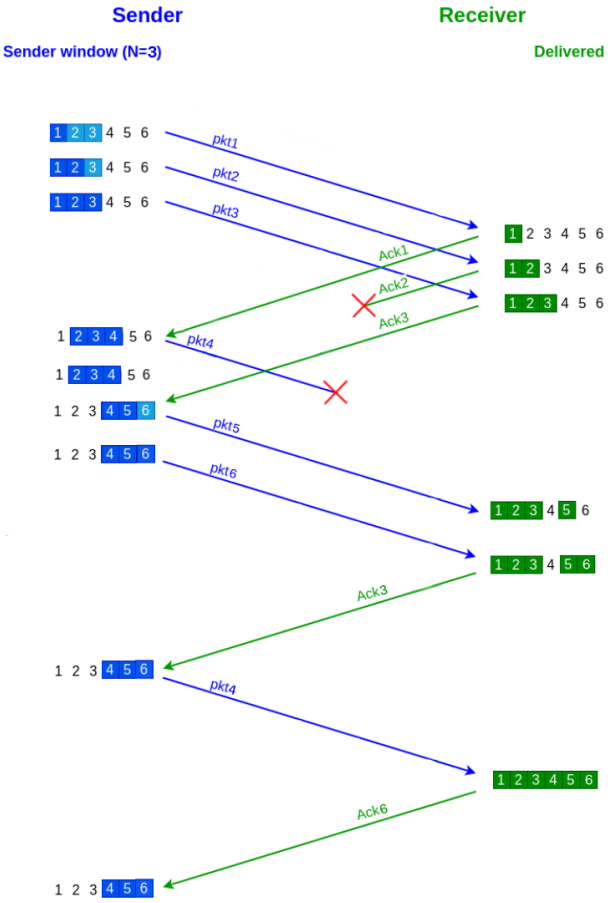

Figure 1: RDT pipelining in TCP

- Lightblue is the segments in the sending window
- Dark green are the received segments which is in the receiver buffer ready for the application layer to stream.

When the acknowledgement 2 is lost, will commutative acknowledgement tell the sender that segment 2 was received, due to receiving acknowledgement 3.
When the segment 4 is lost is assumed that the receiver will buffer segment 5 and 6, from this it can be seen that segment 3 is acknowledged again indicating that segment 4 was lost. When segment 4 is then received the buffered segments are still used such the last segment 6 is acknowledged. This is again using commutative acknowledgement.

3 Problem - TLS

Consider the picture in figure 2 of a package captured form wireshark. Answer the questions below the image.

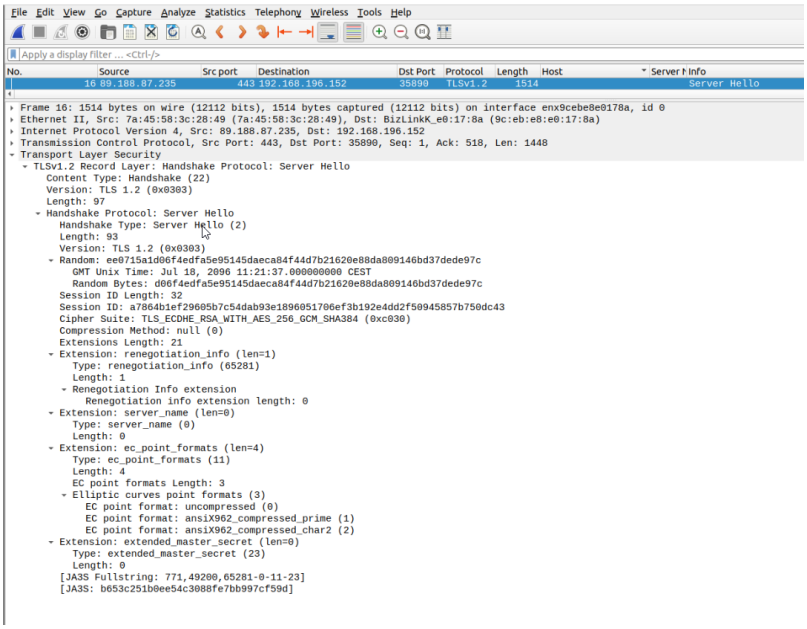

Figure 2: Wireshark capture

3.1 What is the overall role of the highlighted packet (packet No 16)?

The packets role is the answer to a client hello.
The server hello package contain:

- Choice of cryptography,- and hashing algorithms
- SSL certificate
- Random string of bytes (nonce)

3.2 What was the length of the data in the previous packet?

While starting the connection and sending the SYN flag, the ack number will be 1.

It can be seen that the current ACK number is 518, therefore making the last package sent of size 517 bytes.

3.3 What version of TLS is being used, and which cipher suite is used?

The TLS version used is 1.2
The ciphers used are:

- Key Exchange - ECDHE (Elliptic Curve Diffie-Hellman Ephemeral)
- Authentication - RSA (Rivest Shamir Adleman)
- Encryption - AES (Advanced Encryption Standard) 256 with Galois/-Counter mode
- Hash - SHA (Secure Hash Algorithm) 384

3.4 Who selected the cipher suite, the client of the server?

In TLS the server always selects cipher suite, which it desires from the offered ones by the client.

3.5 What does each parts of the cipher suite provide to TLS?

- Key Exchange - The keys used for encryption and decryption of package data
- Authentication - The authentication algorithm is used to ensure authentication between client and server
- Encryption - The chosen algorithm for encrypting and decrypting the package data to ensure confidentiality
- Hash - The hashing method for ensuring integrity of the data