Computer Networks-Lab 09

Instructor: HURMAT HIDAYAT

CL30001 – Computer Networks-Lab

SEMESTER Fall 2022

Computer Networks- Lab 09

OBJECTIVES

After this Lab students shall be able to:

- **Differentiate between TCP and UDP.**
- **Differentiate between TCP header and UDP header.**
- **Understand TCP and UDP header from Wireshark.**

PRE-LAB READING ASSIGNMENT

Remember the delivered lecture carefully.

Table of Contents

Segmentation Explained with TCP and UDP Header

This Lab explains what the segmentation is, how the segmentation works in data communication process, what the TCP and UDP header contain and how the header is used to build a segment.

Both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) protocols work in the Transport layer. Both provide same functionality in different ways. This functionality is the delivering data at the correct destination. While providing this functionality, **TCP focuses on accuracy while UDP pays attention on speed.**

Following five functions are used in data delivery process: -

- Segmentation
- Connection multiplexing
- Connection oriented or connection less delivery
- Reliability through acknowledgement and sequencing
- Flow control through windowing

From these functions, to ensure the accuracy in delivery process, TCP supports all functions while in order to provide the highest possible speed, UDP supports only the second function.

Let's understand each function in detail and compare the way in which both protocols provide it.

Segmentation

Segmentation is the process of dividing large data stream into smaller pieces. This functionality allows a host to send or receive a file of any size over the any size of network. For example, if network bandwidth is 1 Mbps and file size is 100 Mb, host can divide the file in 100 or more pieces. Once a piece becomes less or equal to the network bandwidth in size, it can be transferred easily. Destination host, upon receiving all pieces, joins them back to reproduce the original file.

TCP supports segmentation while UDP does not. It means if an application wants to use the TCP to send its data, it can give the data to TCP in actual size. Based on several conditions such as data size and available network bandwidth, if segmentation is required, TCP does it on its own before packing data for transmission.

But if an application wants to use UDP to send its data, it can't give the data to UDP in actual size. It has to use its own mechanism to detect whether segmentation is required or not. And if segmentation is required, it has to do it on its own before giving data to UDP.

Packing data for transmission

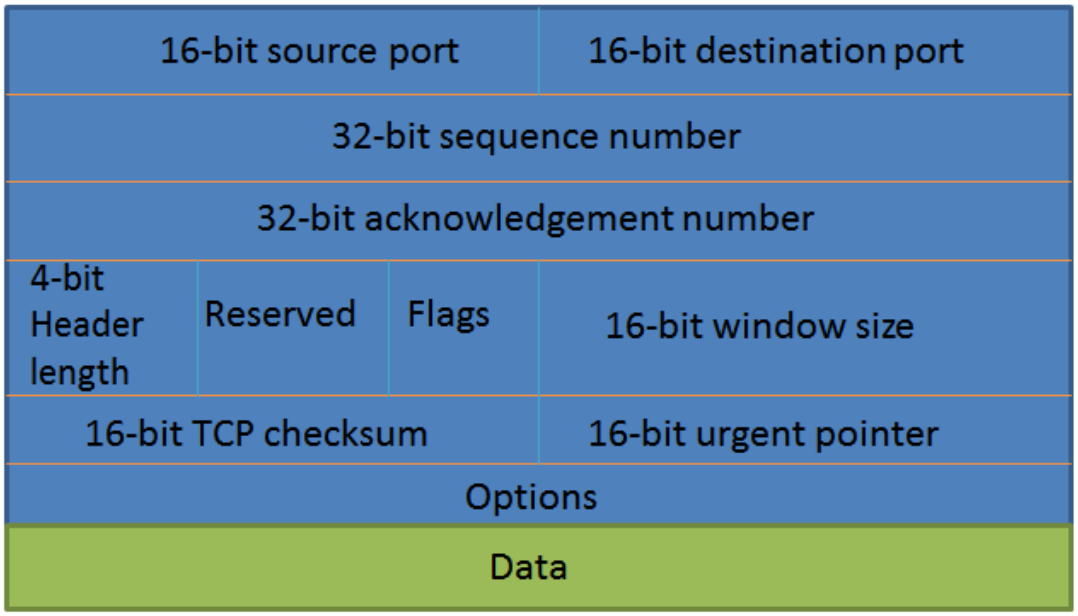
Both protocols pack data in similar fashion. Both add a header with each data piece. A header mainly contains two types of information;

1. The information that is required to send the segment at the correct destination.
2. The information that is required to support the protocol specific features.

Both TCP and UDP add first type of information in same manner. Both use two fields for this information; source port and destination port. Information about the application that is sending the data and the information about the application that will receive the data are added in source port field and in destination port field respectively.

Protocols add second type of information based on the services they offer. TCP offers several protocol specific services such as segmentation, windowing, flow control, etc. To provide these services, it adds the necessary information in the header.

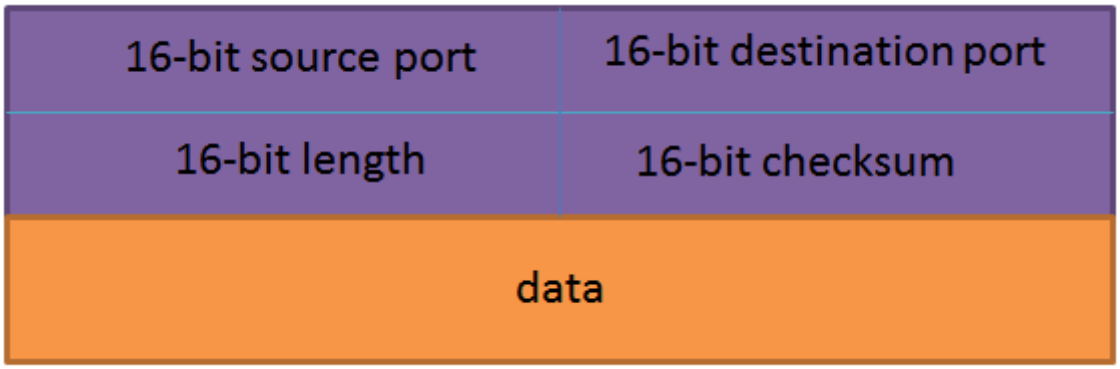
Following figure shows a data piece with the TCP header.



Field	Description
Source port	Used to identify the application that is sending data from the source host
Destination port	Used to identify the application that will receive the data at destination host
Sequence Number	Used to identify the lost segments and maintain the sequencing in transmission.
Acknowledgment Number	Used to send a verification of received segments and to ask for the next segments
Header Length	A number that indicates where the data begin in segment
Reserved	Reserve for future use. Always set to zero.
Code Bits	Used to define the control functions such as setting up and terminating the session
Window size	Used to set the number of segments that can be sent before waiting for a confirmation from the destination.
Checksum	CRC (cyclic redundancy check) of the header and data piece. Checksum is a simple error detection mechanism to determine the integrity of the data transmitted over a network. Communication protocols like TCP/IP/UDP implement this scheme in order to determine whether the received data is corrupted along the network.
Urgent	Used to point any urgent data in the segment.
Options	Used to define any additional options such as maximum segment size
Data	A data piece that is produced from the segmentation. The data is the actual content, such as a string of letters or part of a webpage.

On other hand, UDP neither provides any protocol specific service, nor adds any additional information in the header.

Following figure shows data with UDP header.

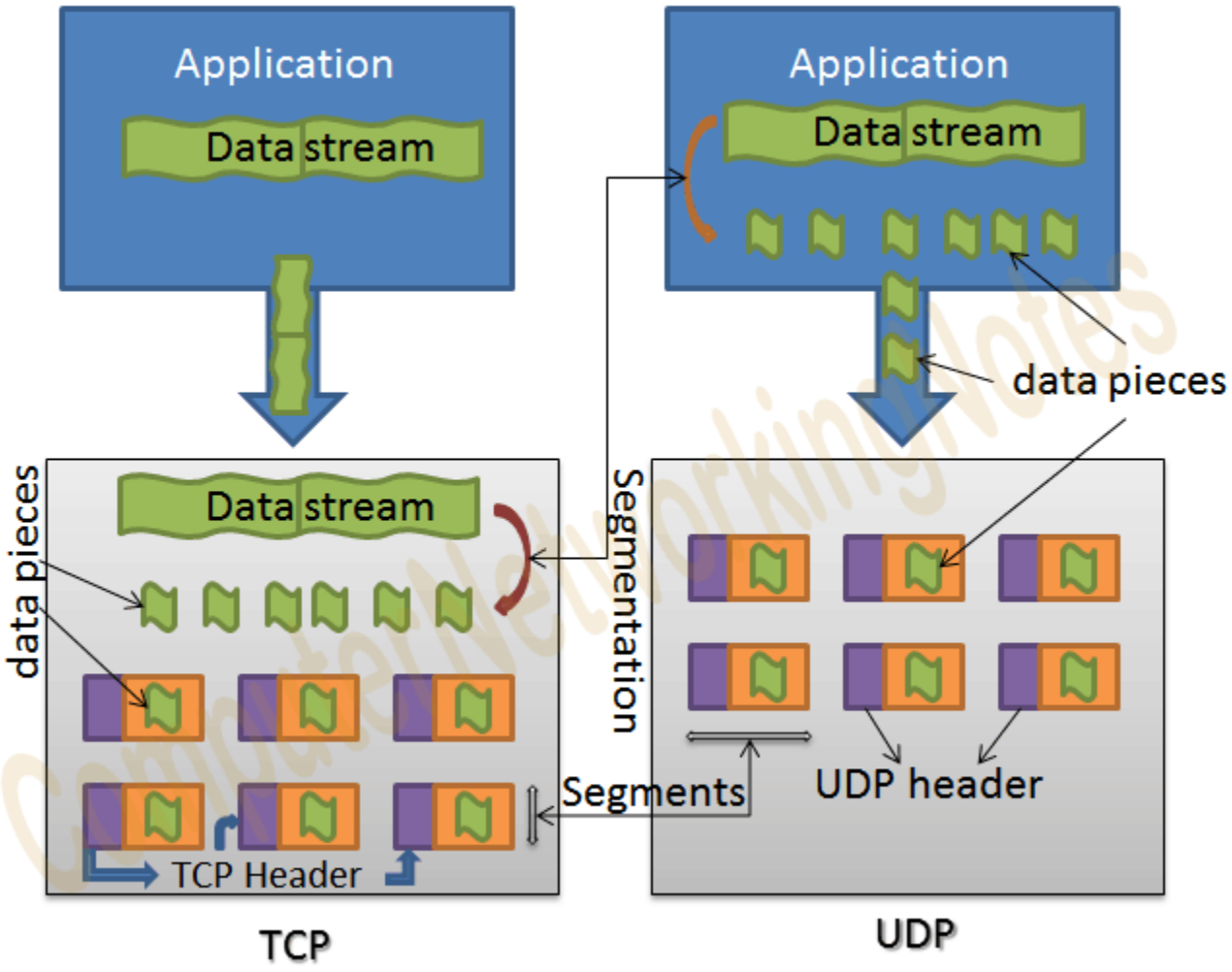


Field	Description
Source port	Port number of the application that is transmitting data from the source computer
Destination port	Port number of the application that will receive the data at destination.
Length	Denotes the length of the UDP header and the UDP data
Checksum	CRC of the complete segment. Checksum is a simple error detection mechanism to determine the integrity of the data transmitted over a network. Communication protocols like TCP/IP/UDP implement this scheme in order to determine whether the received data is corrupted along the network.
Data	Data which it received from the application

Segment

Once a header is attached with the data piece (generated from the segmentation in TCP or received from the application in UDP), it is referred as a segment.

Following figure shows how segmentation works in both protocols.



Key points

- TCP uses segmentation while UDP does not.
- Both protocols use different types of header to pack the data for transmission.
- UDP header contains information only about the compulsory functions and it is 8 bytes in the length.
- TCP header contains information for both compulsory and optional functions. TCP header is 20 bytes and 24 bytes in length without options and with options respectively.

Connection oriented protocol or connection-less protocol

TCP is a connection oriented protocol. Difference between a connection-oriented protocol and a connection-less protocol is that a connection-oriented protocol does not send any data until a proper connection is established. Connection establishment refers to the process of initializing protocol specific features.

TCP, in connection establishment process, initializes sequence and acknowledgement numbers. TCP refers this process as the three-way handshake process.

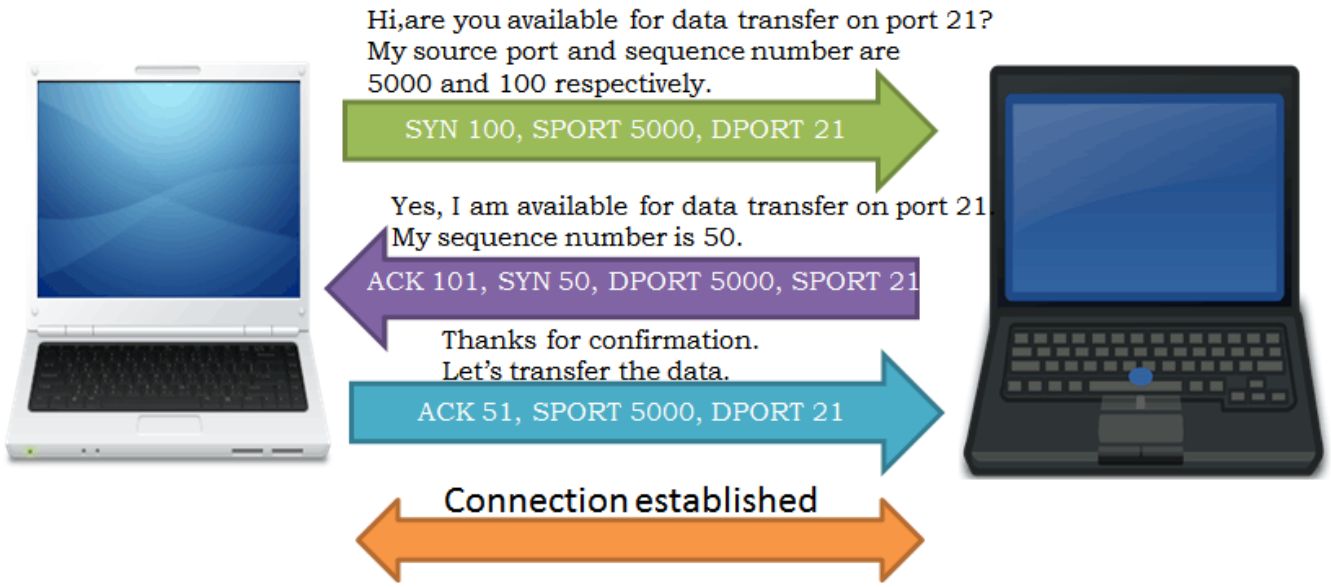
TCP Three-way handshake process

Source sends a SYN (synchronization) segment to the destination. This segment indicates that source want to establish a reliable session with destination.

Destination responds back with a SYN/ACK (synchronization/acknowledgement) segment. This segment indicates that destination received the source's connection request and ready to setup a reliable session with source.

Upon receiving a SYN/ACK segment from destination, source sends an ACK (acknowledgement) segment to the destination. This segment indicates that source received the confirmation from destination and the session is now fully reliable.

Following figure shows an example of Three-way handshake process.



Once the three-way handshake process is done both source and destination can transfer the data.

TCP does not send any data without establishing proper connection. Segments which are used in connection establishment or three-way handshake process contain only the header information that is used to initialize the TCP specific features. These features are explained below.

Reliability through acknowledgement and sequencing

The term TCP reliability covers mainly following items;

- Recognizing and resending lost packets
- Detecting and dropping duplicate packets
- Recognizing and reordering the packets that arrive out of order
- Controlling the overflow of segments

To provide reliability, TCP assigns a sequence number to each sent segment. This number not only helps the destination host in reordering any incoming segments that arrived out of the order but also help in verifying that all sent segments were received.

Acknowledgement numbers are used in opposite direction. These numbers are used to send the verification of received segments, notification of lost segments and acknowledgement for next segments.

Upon receiving all sent segments, to get the next segments, destination sends a segment with a number in the acknowledgment field that is one number higher than the received sequence number.

If any segment lost in transmission, its sequence number is used in acknowledgment field to notify the sender about it.

Both, sequence number and acknowledge number are initialized in three-way handshake process. Source and destination hosts update each other about their sequence number in this process.

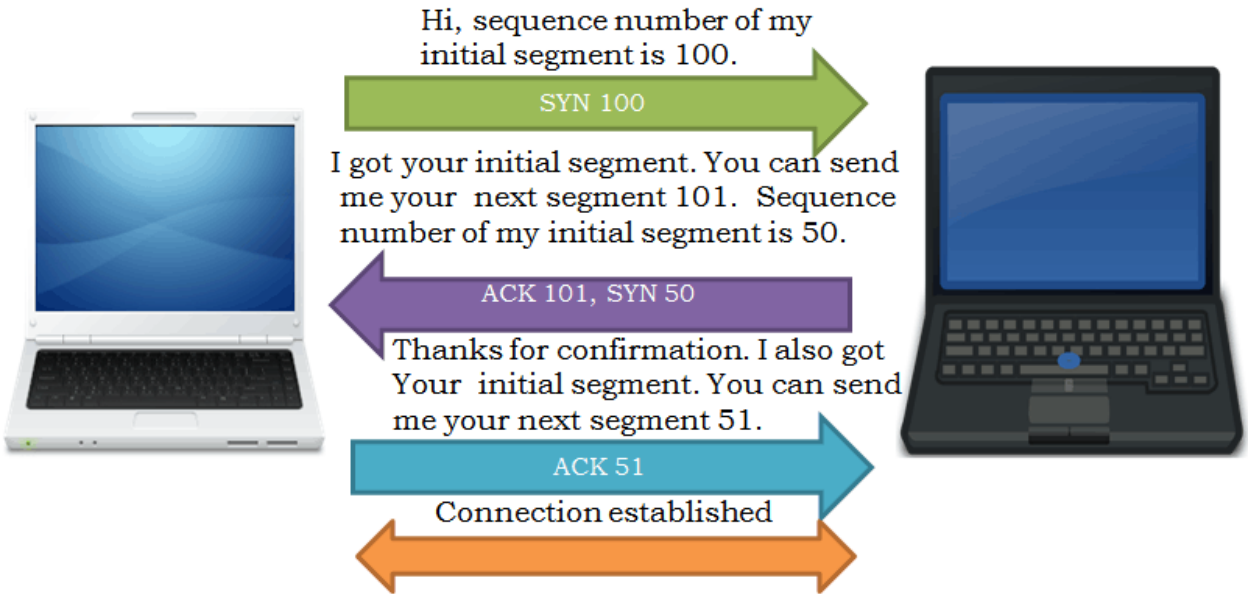
Once source and host know each other's sequence numbers, they use them in data exchange process. Before we take an example of this process, let's understand one more number that is also initialized in three-way handshake process and is used with these numbers.

Flow control through Windowing

Windowing is the process of controlling the flow of segments. It ensures that one host doesn't flood another host with too many segments, overflowing its receiving buffer.

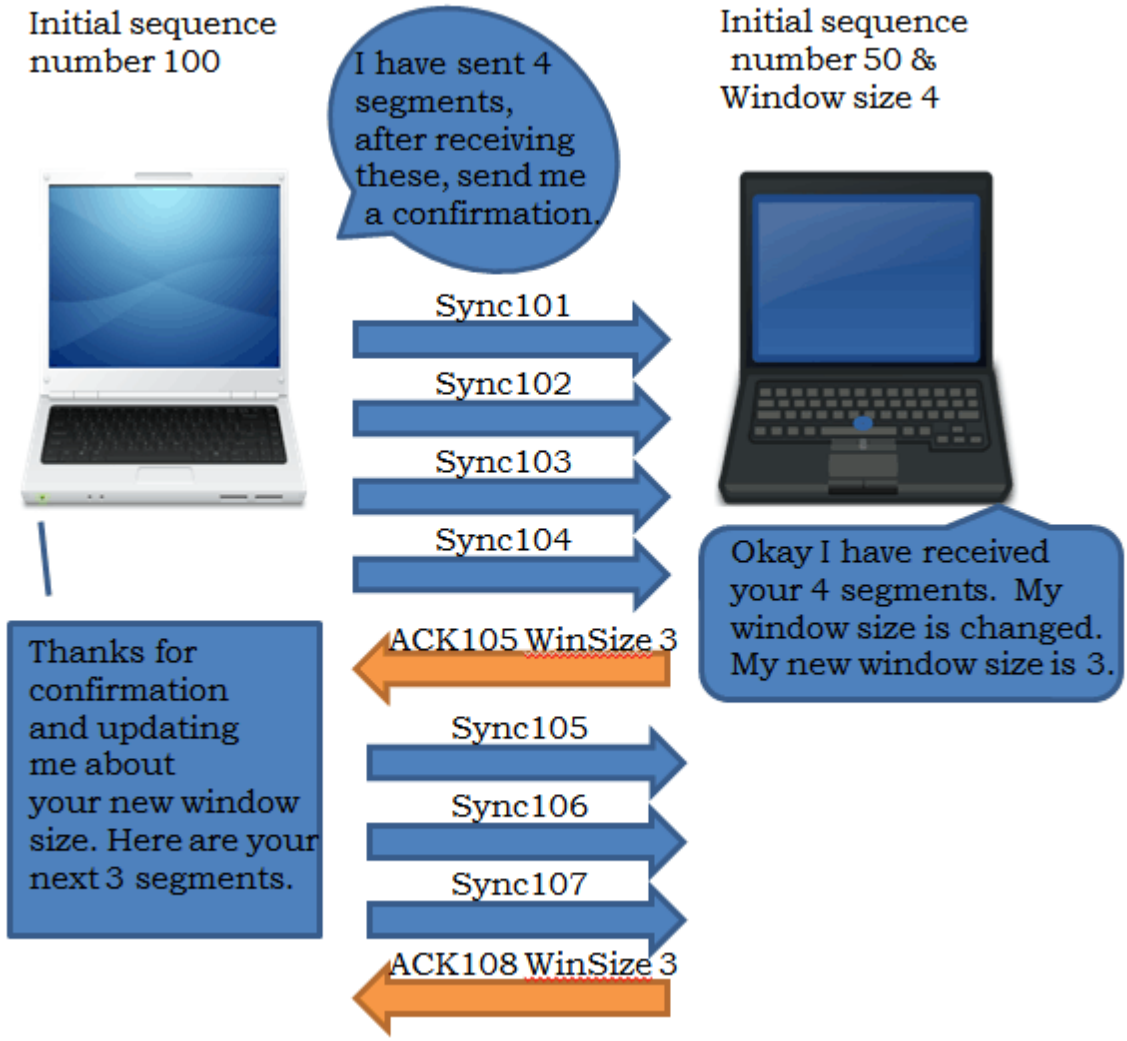
In three-way handshake process, receiver computer, while responding to the initial segment, updates the sender computer about its window size. Window size is the number of segments

that it can store in its buffer. Once sender computer knows the window size of receiver computer, it can control the flow of segments.



For example, if window size of the receiver computer is 4, sender computer sends only the 4 segments. Once 4 segments are sent, it waits for confirmation from receiver computer before sending next 4 segments.

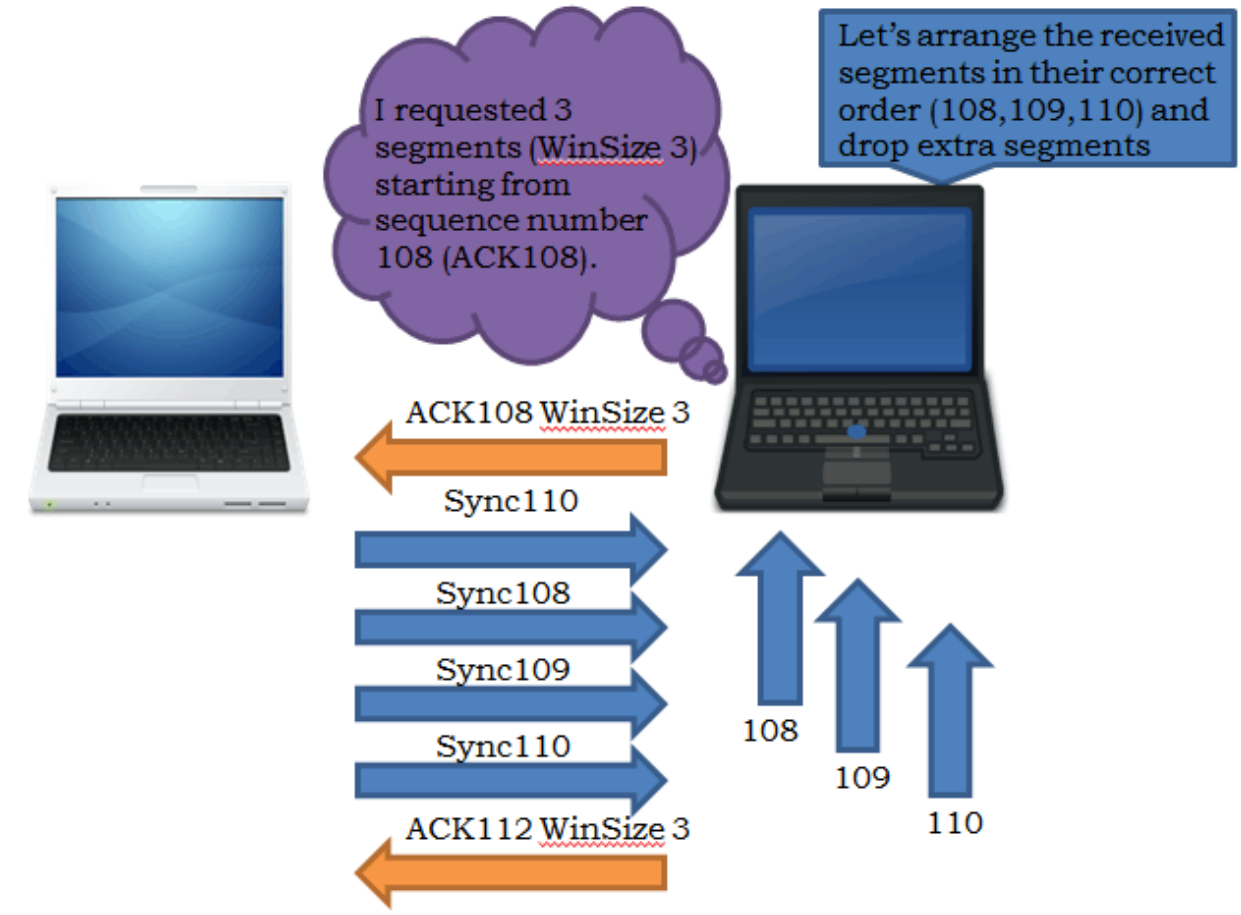
While sending confirmation, receiver computer can change the window size. For example, it can ask sender computer to send more or less segments in next section. This feature is called sliding windowing or dynamic windowing. It allows receiver to control the flow of segments that sender computer can send.



Reordering segments and in correct order and dropping extra segments

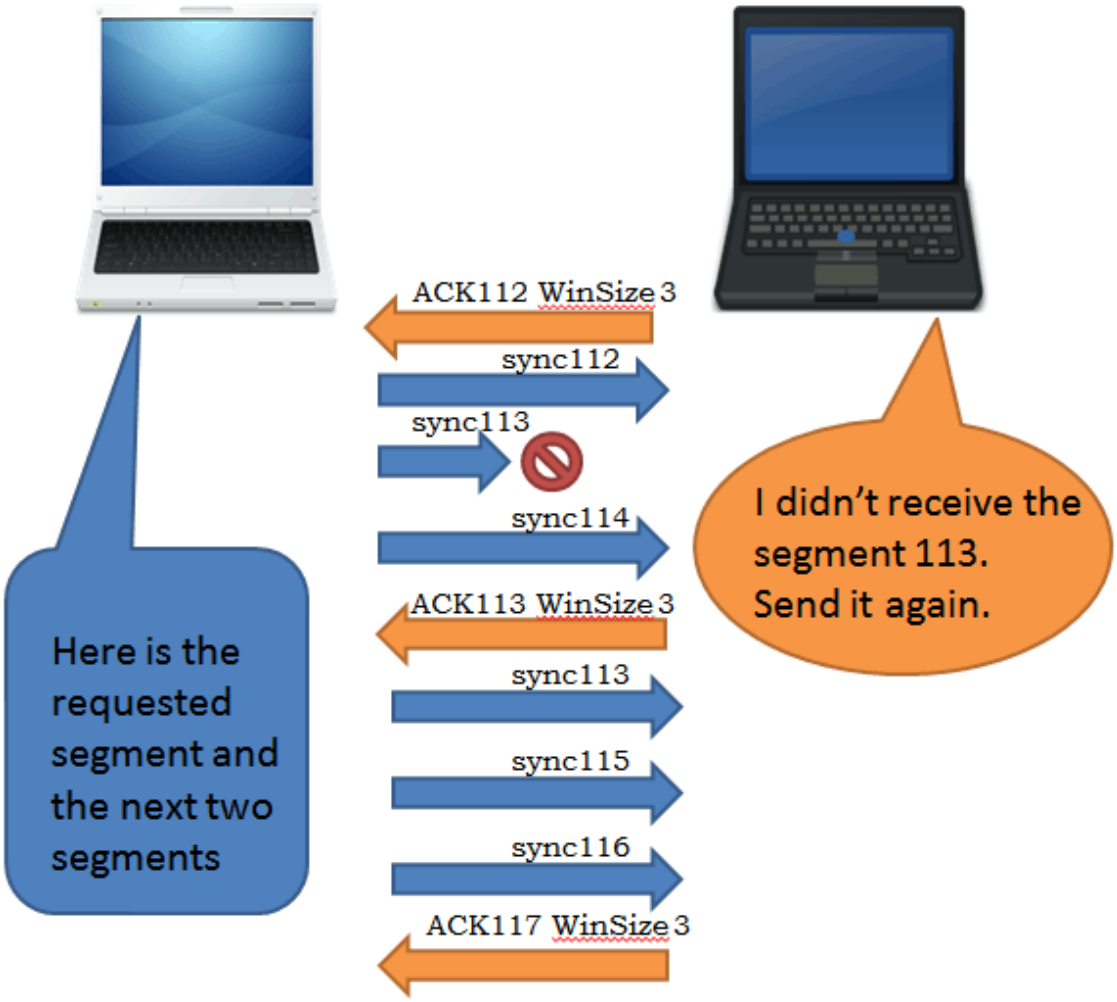
To arrange the arrived segments in correct order, receiver computer uses the sequence numbers of the segments. To detect and drop the duplicate or extra segments, it compares the received segments with the requested segments. For example if receiver computer requested 3 segments by specifying window size 3 in acknowledgment and received 4 segments, it assumes that one segment is arrived extra.

Sequence numbers also help in detecting duplicate segments. If two or more segments have same sequence numbers, they are duplicates. Duplicate segments are dropped.



Recovering lost segments

As mentioned above, receiver computer compares the arrived segments with the expected segments. If it finds any segment is missing, it uses the sequence number of that segment to acknowledge the receiver computer about it. When sender computer receives an acknowledgment of a segment that it has already sent, it assumes that acknowledged segment has lost. While transmitting the next set of segments (segments equal to the window size), it retransmits the lost segments first and new segments later. For example if window size is 3 and lost segment is 1, then the transmitted segments will be; lost segment, new segment and new segment.



Comparing with UDP

UDP is a connection-less protocol. It does not establish any connection or exchange any messages before sending the data.

TCP protocol always starts a session with the three-way handshake process. It means if an application wants to send its data through TCP, it has to wait until the proper connection is established through the three-way hand shake process.

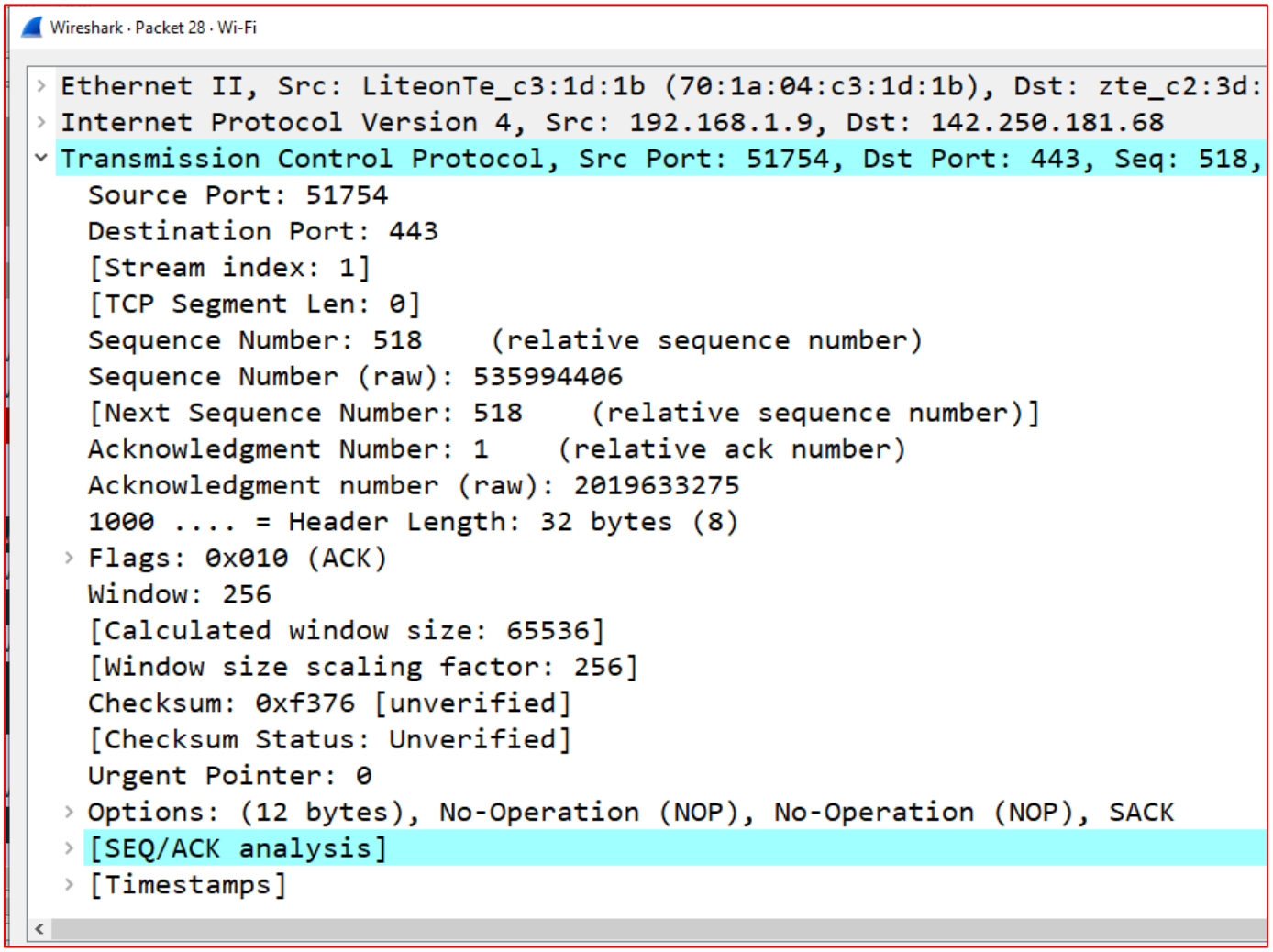
UDP doesn't use any mechanism or process before starting the session. It means if an application wants to send its data through UDP, it can send its data immediately without any delay.

UDP neither sequences the segments nor care about the order in which they are sent to the destination. It also does not take acknowledgement of the sent segments to verify that they

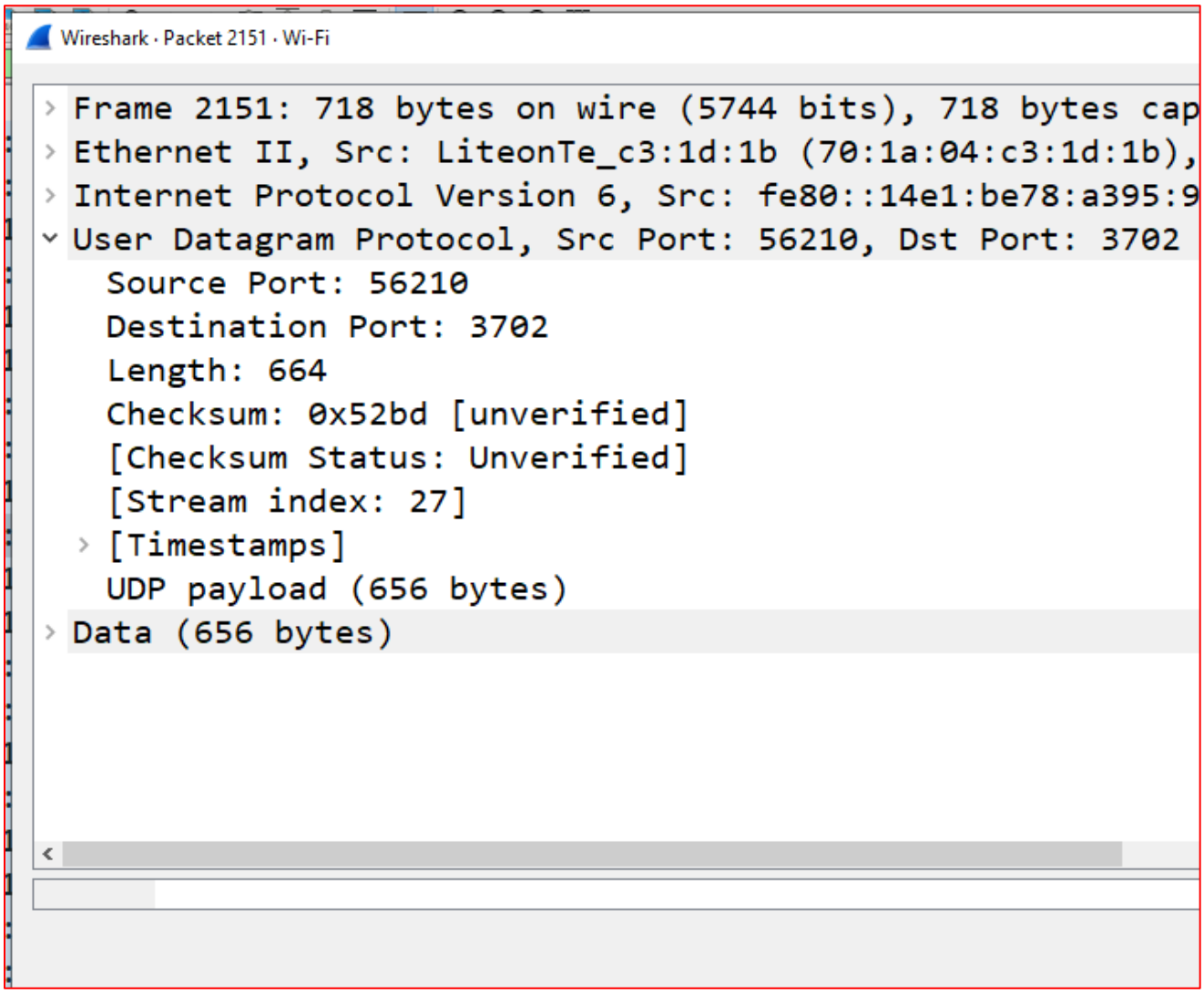
reached at destination. It just sends and forgets about them. Because of this, it is also referred as an unreliable protocol.

UDP takes less bandwidth and uses fewer processing cycles in comparison of TCP.

TCP Header View in Wireshark



UDP Header view in Wireshark



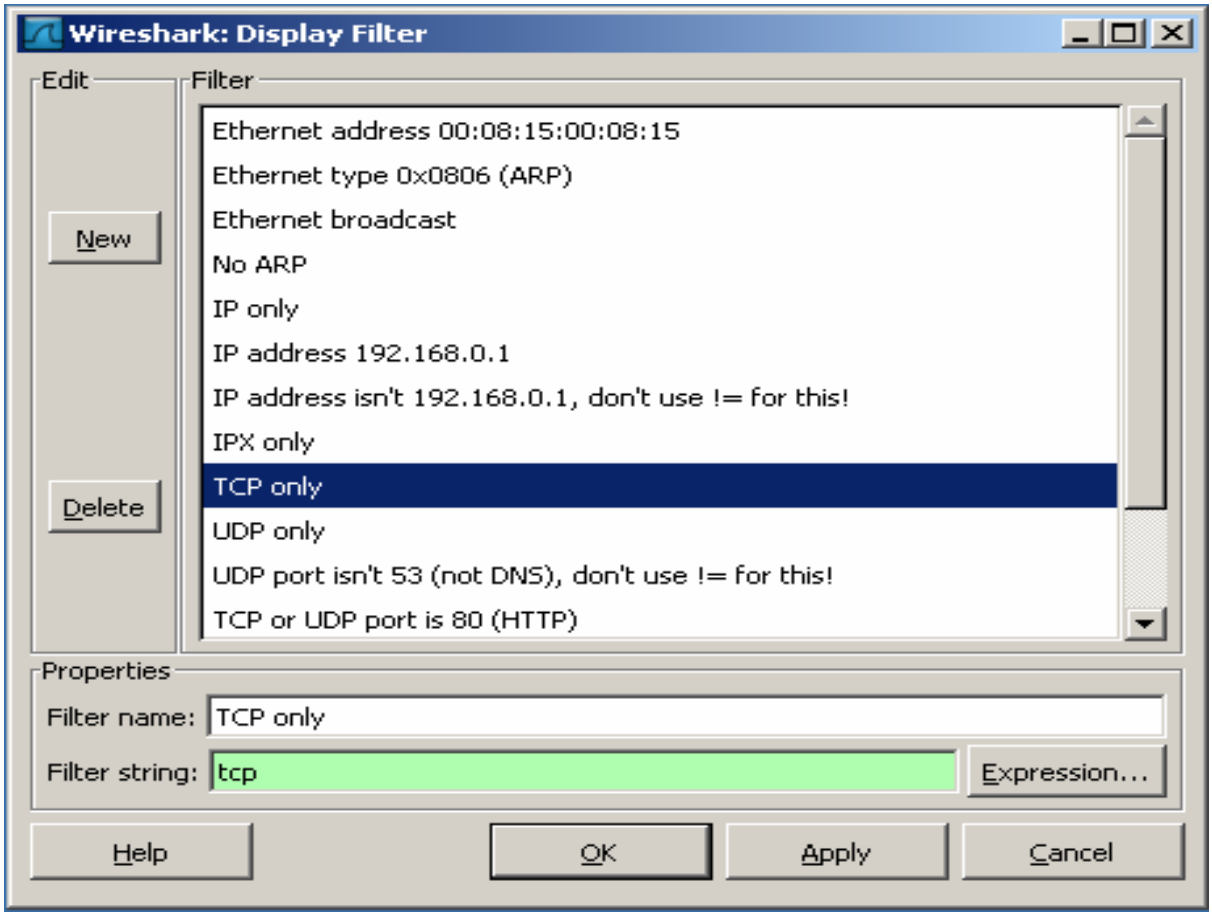
Using Wireshark to Observe the TCP Three-way Handshake:

Given a trace of transferring a 150KB file. The first step is to filter the trace to only view the TCP packets.

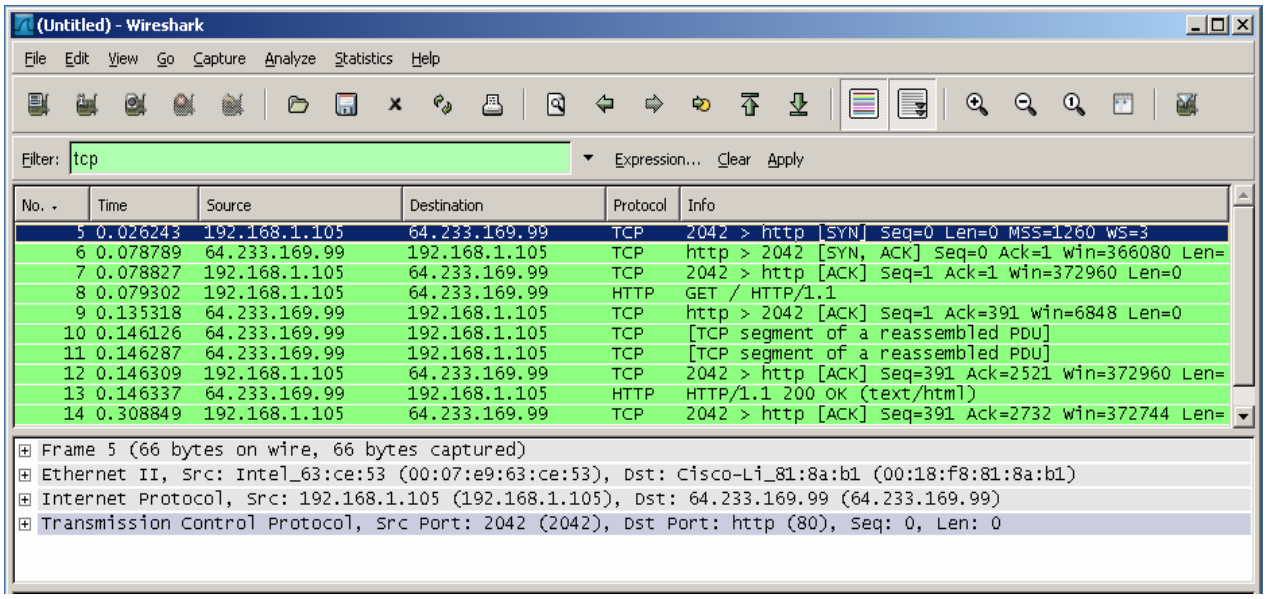
Filter the capture to view only TCP packets:

If you have many packets that are unrelated to the TCP connection, it may be necessary to use the Wireshark filter capability.

a. To use a preconfigured filter, click the Analyze menu option, and then click Display Filters.
b. In the Display Filter window, click TCP only, and then click OK.



c. In the Wireshark window, scroll to the first captured TCP packet. This should be the first packet in the flow.



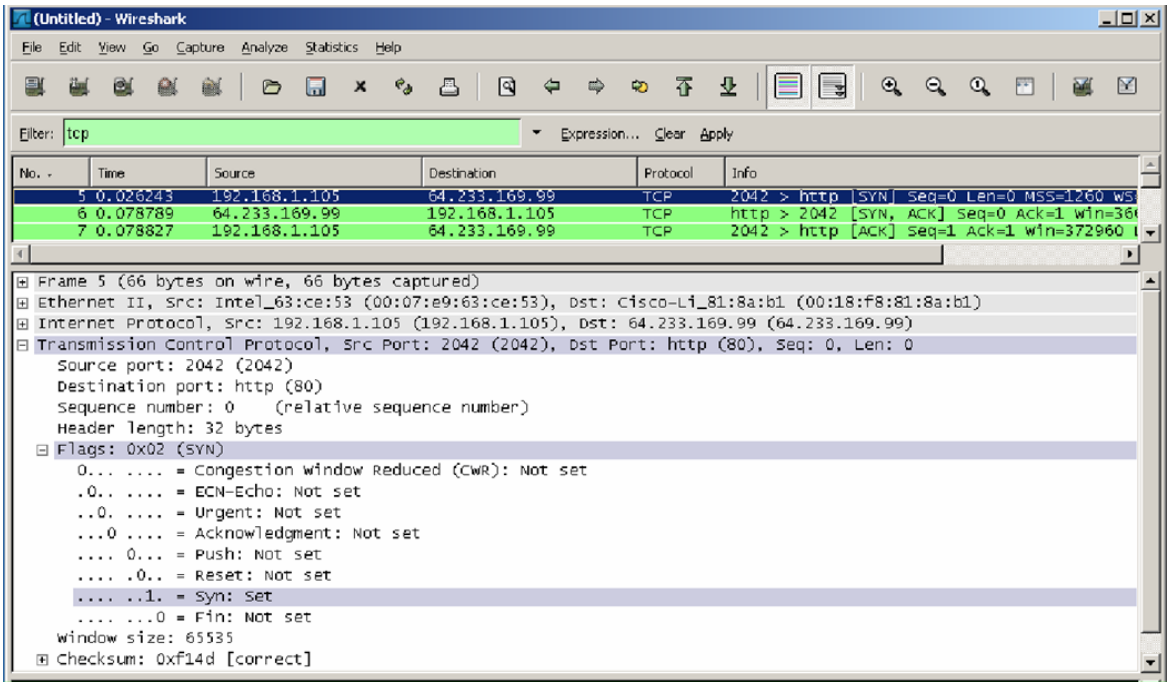
d. In the Info column, look for three packets similar to the first three shown in the window above. The first TCP packet is the [SYN] packet from the initiating computer. The second is the [SYN, ACK] response from the web server. The third packet is the [ACK] from the source computer, which completes the handshake.

Inspect the TCP initialization Sequence

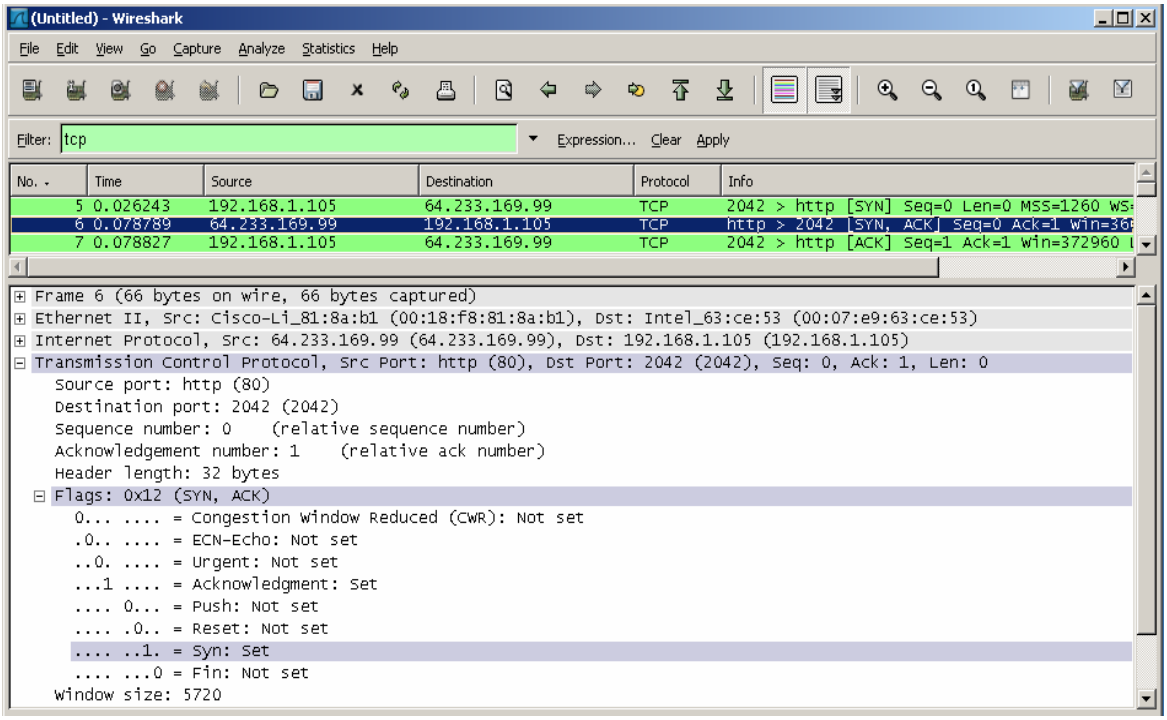
a. In the top Wireshark window, click on the line containing the first packet identified in Step 4. This highlights the line and displays the decoded information from that packet in the two lower windows fill.
Note: The Wireshark windows below were adjusted to allow the information to be viewed in a compact size. The middle window contains the detailed decoding of the packet.

b. Click the + icon to expand the view of the TCP information. To contract the view, click the – icon.

c. Notice in the first TCP packet that the relative sequence number is set to 0, and the SYN bit is set to 1 in the Flags field.

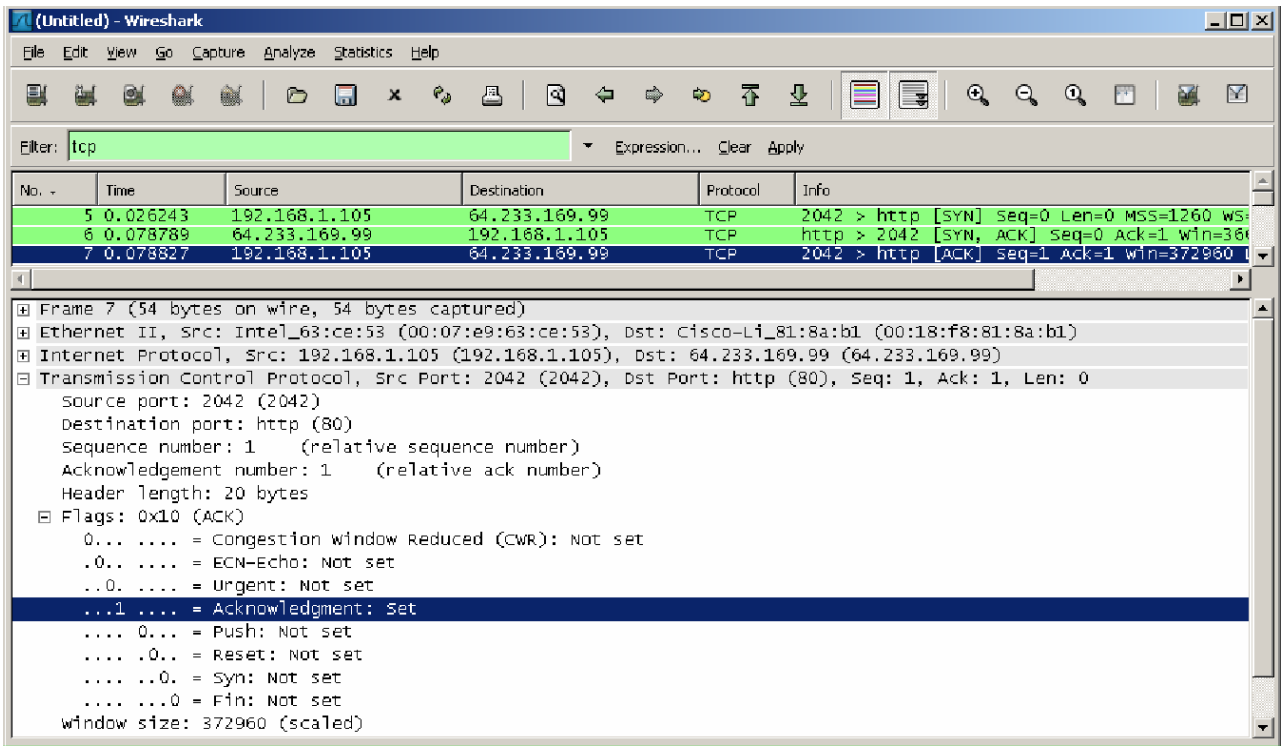


d. Notice in the second TCP packet of the handshake that the relative sequence number is set to 0, and the SYN bit and the ACK bit are set to 1 in the Flags field.



e. In the third and final frame of the handshake, only the ACK bit is set, and the sequence number is set to the starting point of 1. The acknowledgement number is also set to 1 as a starting point. The TCP connection is now

established, and communication between the source computer and the web server can begin.



Lab Task:

Download and Open the trace file here:

Inspect the three-way handshake and answer the following questions:

1. What is the source and destination port numbers?
2. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection? What is it in the segment that identifies the segment as a SYN segment?
3. What is the sequence number of the SYNACK segment sent by the server to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did server determine that value? What is it in the segment that identifies the segment as a SYNACK segment?
4. What is the length of each of the first six TCP segments?
5. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?