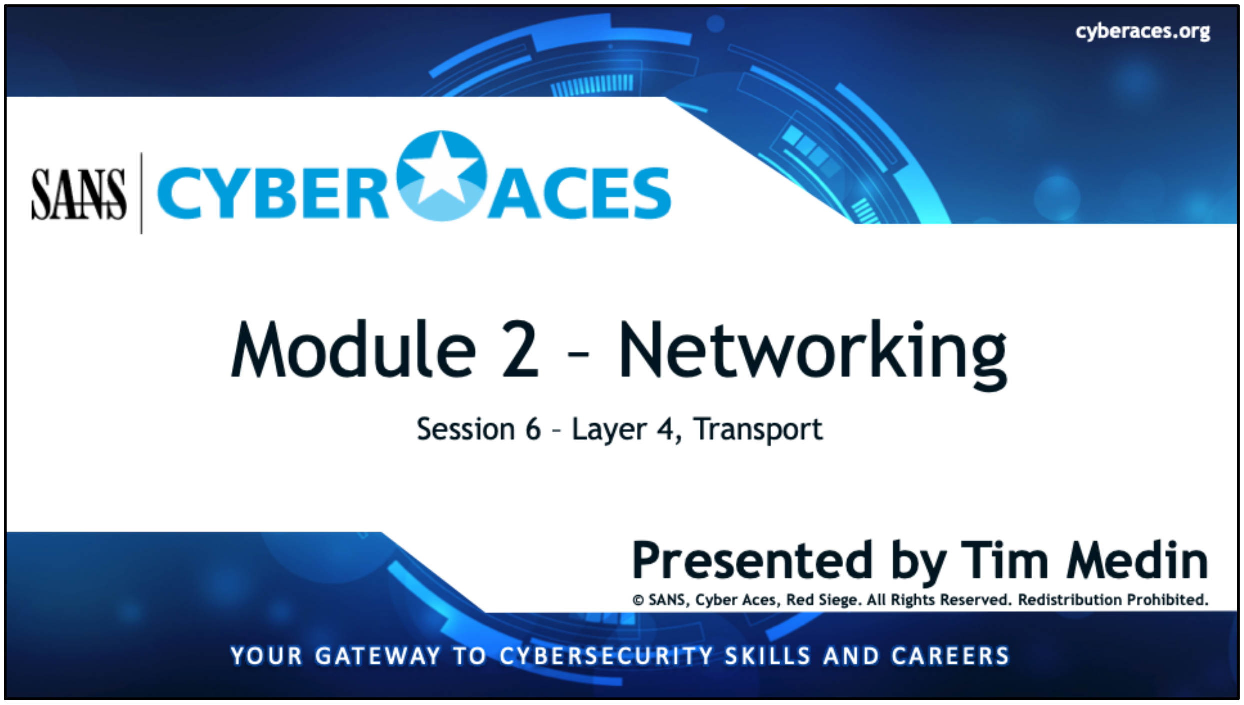


cyberaces.org

SANS | CYBER ACES

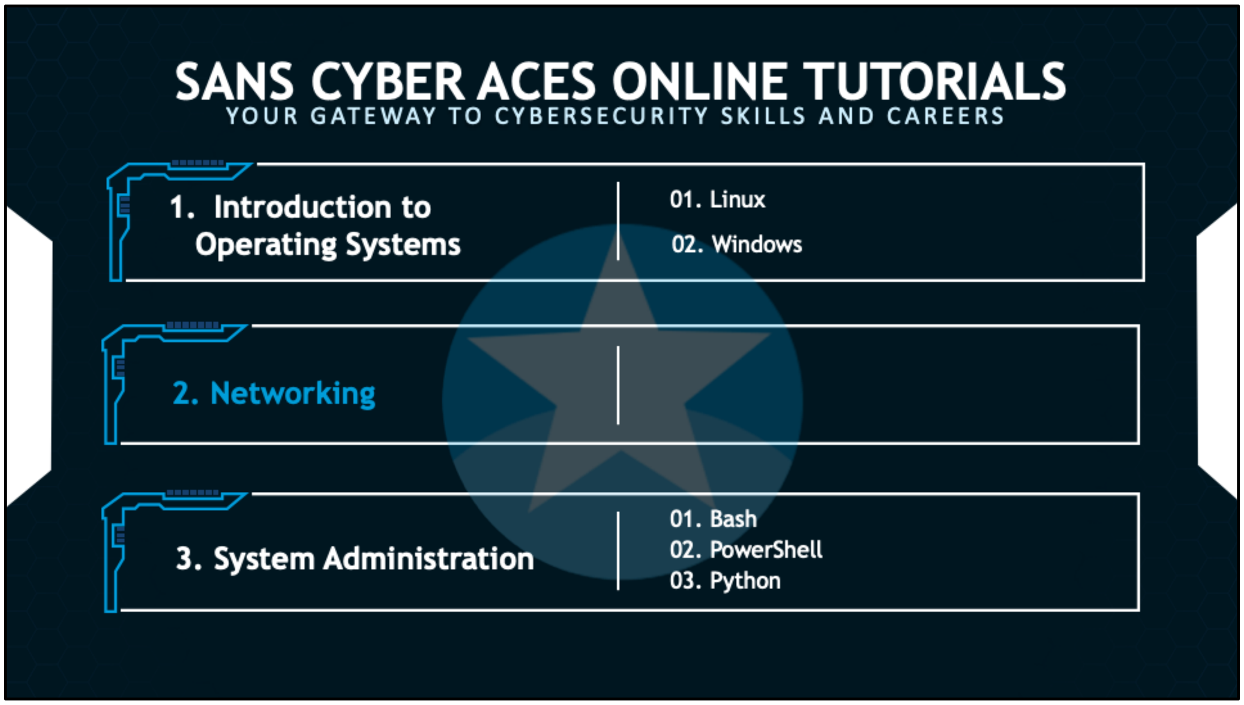
Module 2 – Networking

Session 6 – Layer 4, Transport

Presented by Tim Medin

YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

Welcome to Cyber Aces, Module 2! A firm understanding of network fundamentals is essential to being able to secure a network or attack one. This section provides a broad overview of networking, covering the fundamental concepts needed to understand computer attacks and defenses from a network perspective.



SANS CYBER ACES ONLINE TUTORIALS

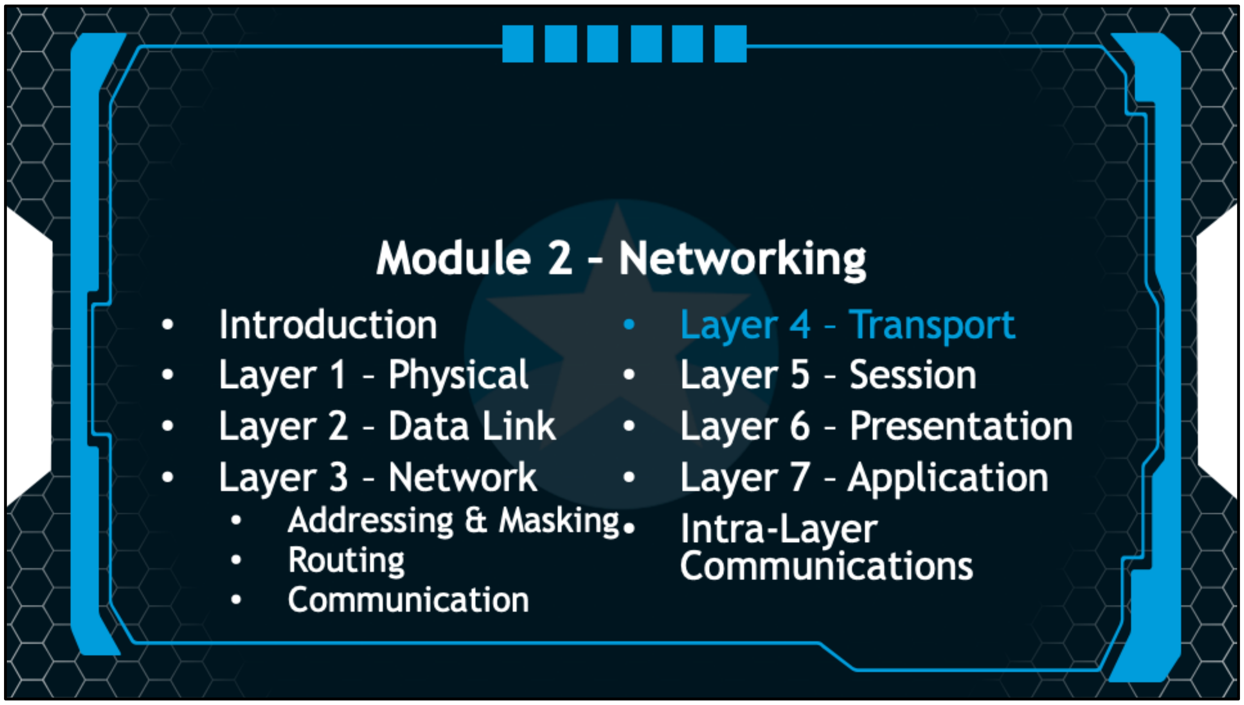
YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

1. Introduction to Operating Systems
   - 01. Linux
   - 02. Windows
2. Networking
3. System Administration
   - 01. Bash
   - 02. PowerShell
   - 03. Python

This training material was originally developed to help students, teachers, and mentors prepare for the Cyber Aces Online Competition. This module focuses on the basics of networking. This session is part of Module 2, Networking.

The three modules of Cyber Aces Online are Operating Systems, Networking, and System Administration.

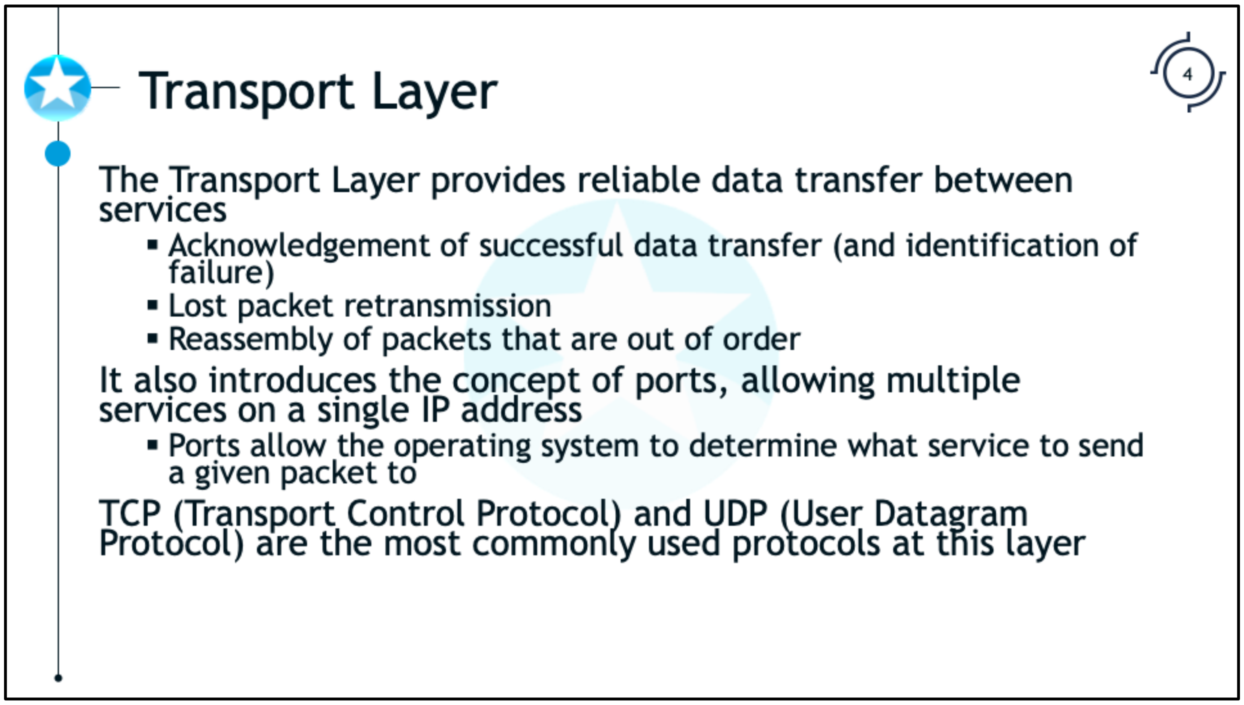
For more information about the Cyber Aces program, please visit the Cyber Aces website at https://CyberAces.org/.



Module 2 - Networking

- Introduction
- Layer 1 - Physical
- Layer 2 - Data Link
- Layer 3 - Network
  - Addressing & Masking
  - Routing
  - Communication
- Layer 4 - Transport
- Layer 5 - Session
- Layer 6 - Presentation
- Layer 7 - Application
- Intra-Layer Communications

In this section, you'll learn about the Transport Layer. We'll cover the concept of ports, as well as TCP and UDP, which are essential parts of any modern network.



Transport Layer

The Transport Layer provides reliable data transfer between services

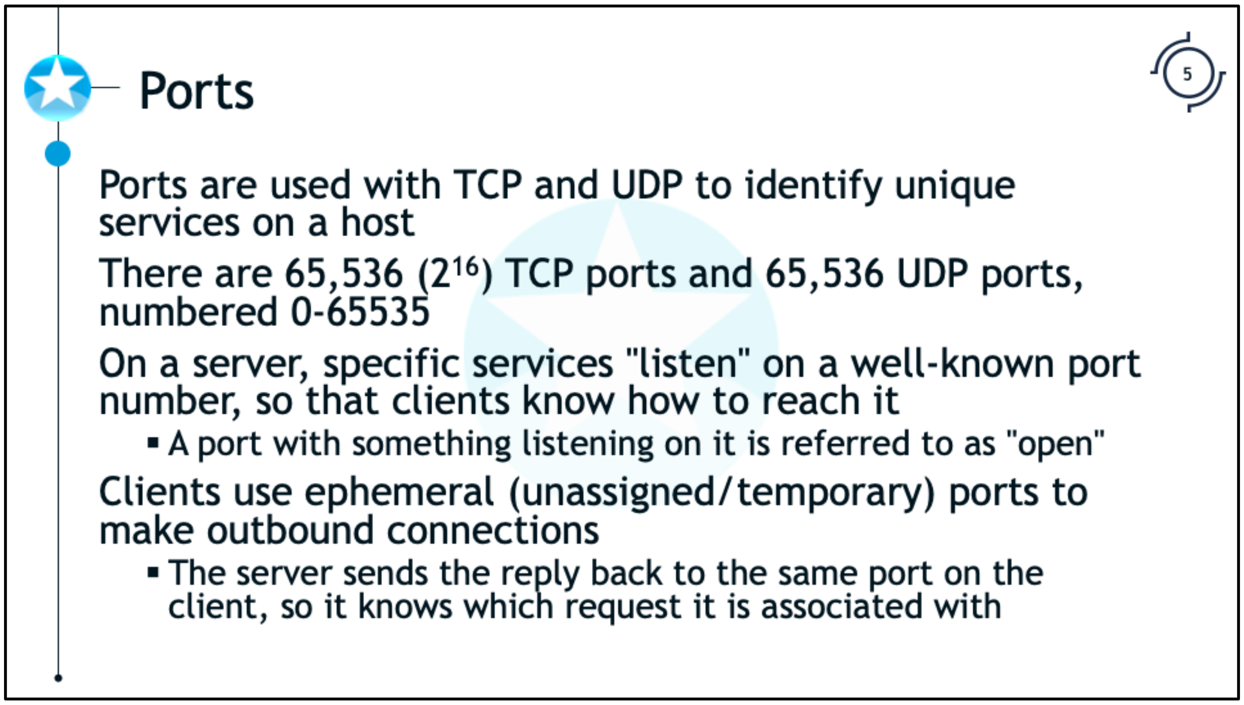
- Acknowledgement of successful data transfer (and identification of failure)
- Lost packet retransmission
- Reassembly of packets that are out of order

It also introduces the concept of ports, allowing multiple services on a single IP address

- Ports allow the operating system to determine what service to send a given packet to

TCP (Transport Control Protocol) and UDP (User Datagram Protocol) are the most commonly used protocols at this layer

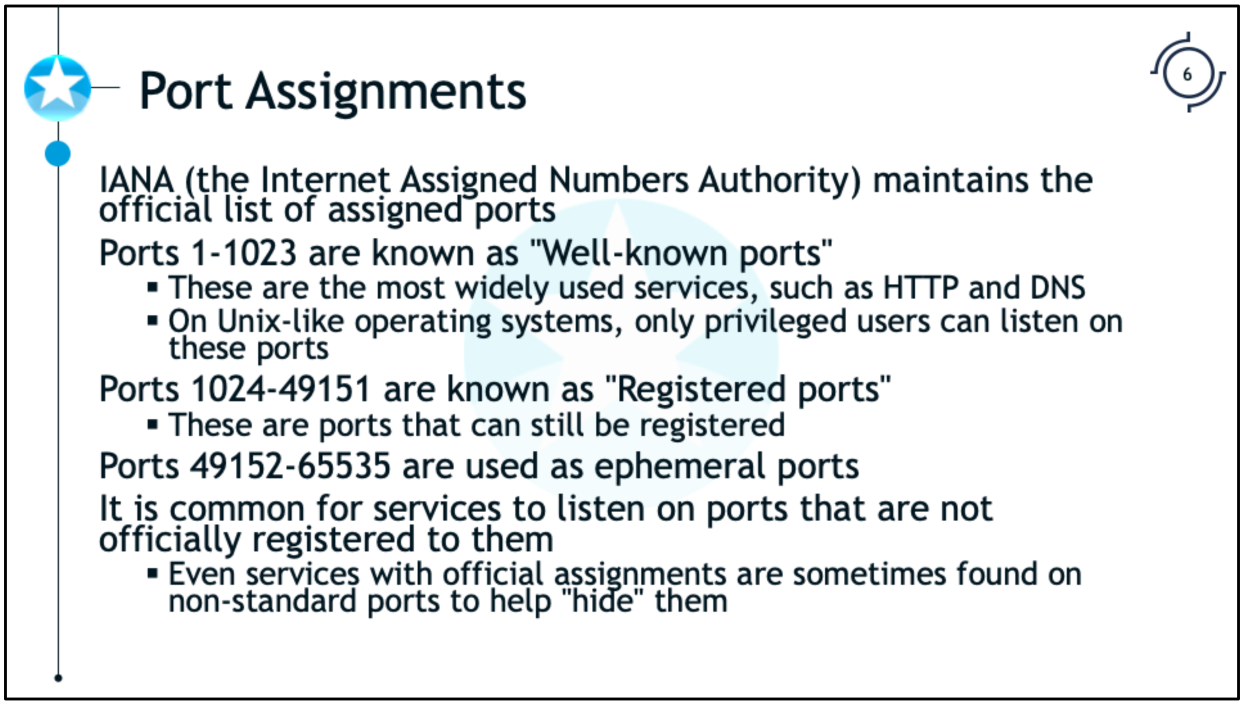
With a reliable Network Layer in place, we now have a way of sending a stream of packets back and forth between two machines. But there are significant limitations to what we can do with just a networking layer. For example, if you want to access multiple services such as Web, E-mail and SSH on a remote computer, the networking layer doesn't provide a means of matching packets to a service. Also, if we send several related (non-fragmented) packets between two hosts and they arrive out of order, the Networking Layer doesn't provide a way to unscramble the message at the destination. The Transport Layer solves both of these problems and more. TCP (Transport Control Protocol) is the king of the Transport Layer, but there are several important protocols that operate at the Transport Layer including TCP, UDP (User Datagram Protocol) and SCTP (Steam Control Transmission Protocol). TCP and UDP introduce the concept of PORTS that identify unique services on host. When we combine an IP address with a PORT, our computer can now establish a SOCKET between two different hosts.



Ports

- Ports are used with TCP and UDP to identify unique services on a host
- There are 65,536 (2^{16}) TCP ports and 65,536 UDP ports, numbered 0-65535
- On a server, specific services "listen" on a well-known port number, so that clients know how to reach it
  - A port with something listening on it is referred to as "open"
- Clients use ephemeral (unassigned/temporary) ports to make outbound connections
  - The server sends the reply back to the same port on the client, so it knows which request it is associated with

Ports are a bit like doors on a system, providing different services their own door to use. Ports can also be thought of like apartment numbers within a building, where the building's street address would be the IP address. Without port numbers, there would be no way for the operating system to determine which service a given packet should be sent to. There are 65,536 (2^{16}) TCP ports and 65,536 UDP ports, numbered 0-65535 (port 0 is reserved and not generally used). On a server, specific services "listen" on a pre-defined port number, so that clients know in advance how to reach it. For example, HTTP (web) servers generally listen on port 80, so web browsers always connect to port 80 by default. When a port has a service listening on it, it is called "open". Conversely, a port without anything listening on it is called "closed". When clients make outbound connections, they send traffic from an ephemeral (temporary) port, and listen on that same port for the response. This way, the operating system can keep track of which responses are associated with particular requests.



Port Assignments

IANA (the Internet Assigned Numbers Authority) maintains the official list of assigned ports

Ports 1-1023 are known as "Well-known ports"

- These are the most widely used services, such as HTTP and DNS
- On Unix-like operating systems, only privileged users can listen on these ports

Ports 1024-49151 are known as "Registered ports"

- These are ports that can still be registered

Ports 49152-65535 are used as ephemeral ports

It is common for services to listen on ports that are not officially registered to them

- Even services with official assignments are sometimes found on non-standard ports to help "hide" them

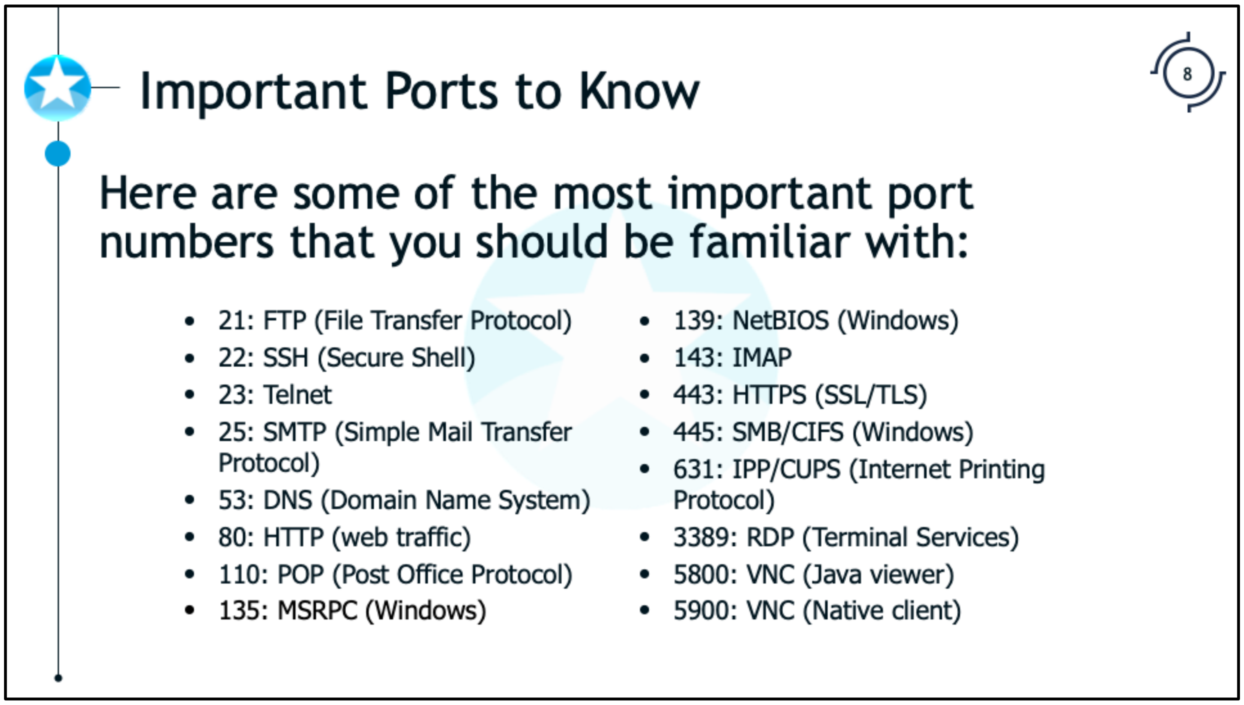
IANA, the Internet Assigned Numbers Authority, maintains the official registry of assigned ports. Maintaining a central list allows all services to know where to find other services (in fact, a copy of the assigned ports is found in the "services" file in most operating systems), and also helps prevent conflicts. However, there are many common services without official port registrations.

Ports 1-1023 are known as "Well-known ports". These are the most widely used core networking services, such as HTTP, DNS, and SSH. On Unix-like operating systems (including Linux and Mac OS X), only privileged users can listen on these ports. This is a security protection that ensures regular users can't set up a rogue service.

Ports 1024-49151 are known as "Registered ports". These are ports that still have official assignments from IANA, but are not as important (or as old) as the services on the well-known ports.

Ports 49152-65535 are used as ephemeral ports, and cannot be officially registered to a service. These are generally used for outgoing connections. However, it should be noted that this convention is not strictly followed by all operating systems. Some versions of Linux use the range 32768-61000, and older versions of Windows (Windows Server 2003 and earlier) use 1025-5000.

Many services use ports that are not officially registered to them. Even services with official assignments are sometimes found on non-standard ports, either to help "hide" them from potential attackers (security through obscurity), or simply to offer similar services (such as a web-based management interface). SSH can often be found on high-numbered ports to help hide it from automated password-guessing attacks. Of course, changing the port number alone should not be considered secure; is is simply one extra layer of security.



Important Ports to Know

Here are some of the most important port numbers that you should be familiar with:

- 21: FTP (File Transfer Protocol)
- 22: SSH (Secure Shell)
- 23: Telnet
- 25: SMTP (Simple Mail Transfer Protocol)
- 53: DNS (Domain Name System)
- 80: HTTP (web traffic)
- 110: POP (Post Office Protocol)
- 135: MSRPC (Windows)
- 139: NetBIOS (Windows)
- 143: IMAP
- 443: HTTPS (SSL/TLS)
- 445: SMB/CIFS (Windows)
- 631: IPP/CUPS (Internet Printing Protocol)
- 3389: RDP (Terminal Services)
- 5800: VNC (Java viewer)
- 5900: VNC (Native client)

Here is a list of a few of the most important port numbers that you should be able to recognize:

21 FTP: File Transfer Protocol; an unencrypted protocol used to transfer files

22 SSH: Secure Shell; an encrypted protocol used to access remote machines for system administration, file transfer, creating an encrypted tunnel to another system or network, etc.

23 Telnet: unencrypted protocol used for remote administration; SSH should generally be used instead

25 SMTP: Simple Mail Transfer Protocol; used to transmit e-mail across the Internet)

53 DNS: Domain Name System; hostname to IP address resolution (and vice versa)

80 HTTP: Hypertext Transfer Protocol; an unencrypted protocol used to access web pages

110 POP: Post Office Protocol; one of two primary protocols used to download e-mail from mail server

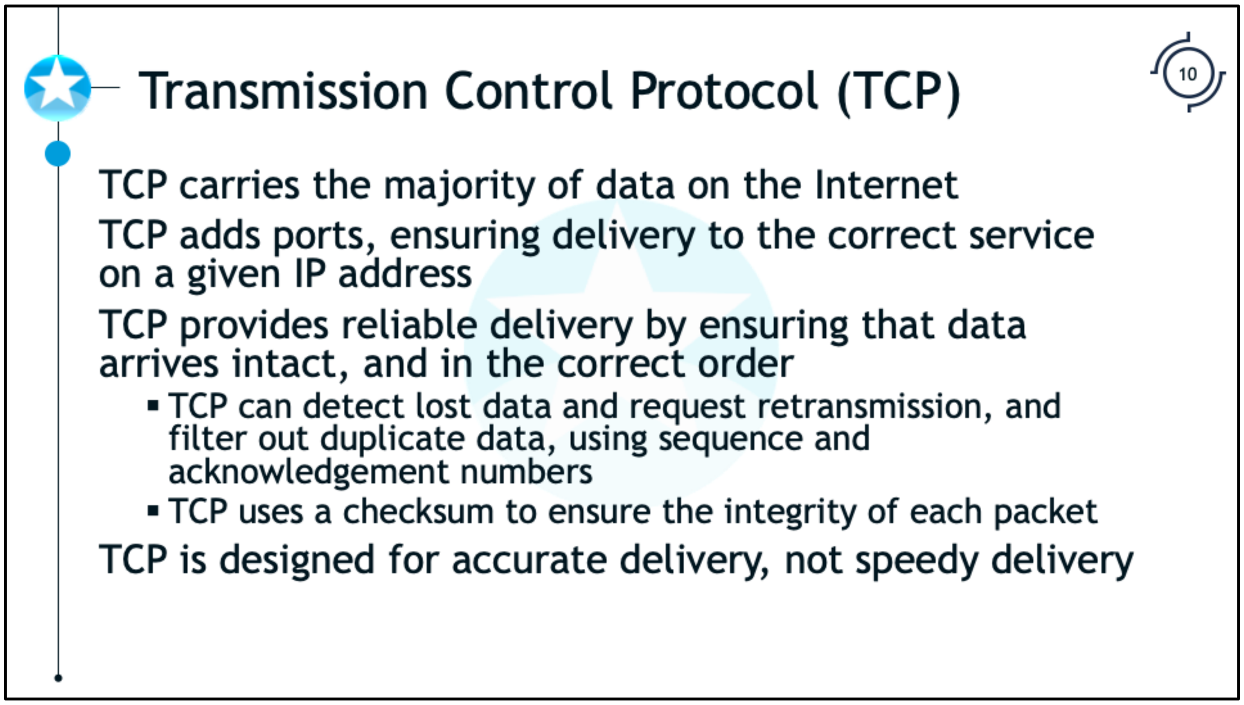
135 MSRPC: used for Windows networking (NetBIOS over TCP) and Remote Procedure Call

139 NetBIOS: used for Windows networking (NetBIOS over TCP)

143 IMAP: Internet Message Access Protocol; used to access e-mail stored on a mail server

443 HTTPS: HTTP over an encrypted channel using SSL/TLS

445 SMB/CIFS: used for Windows networking (SMB/CIFS over TCP)

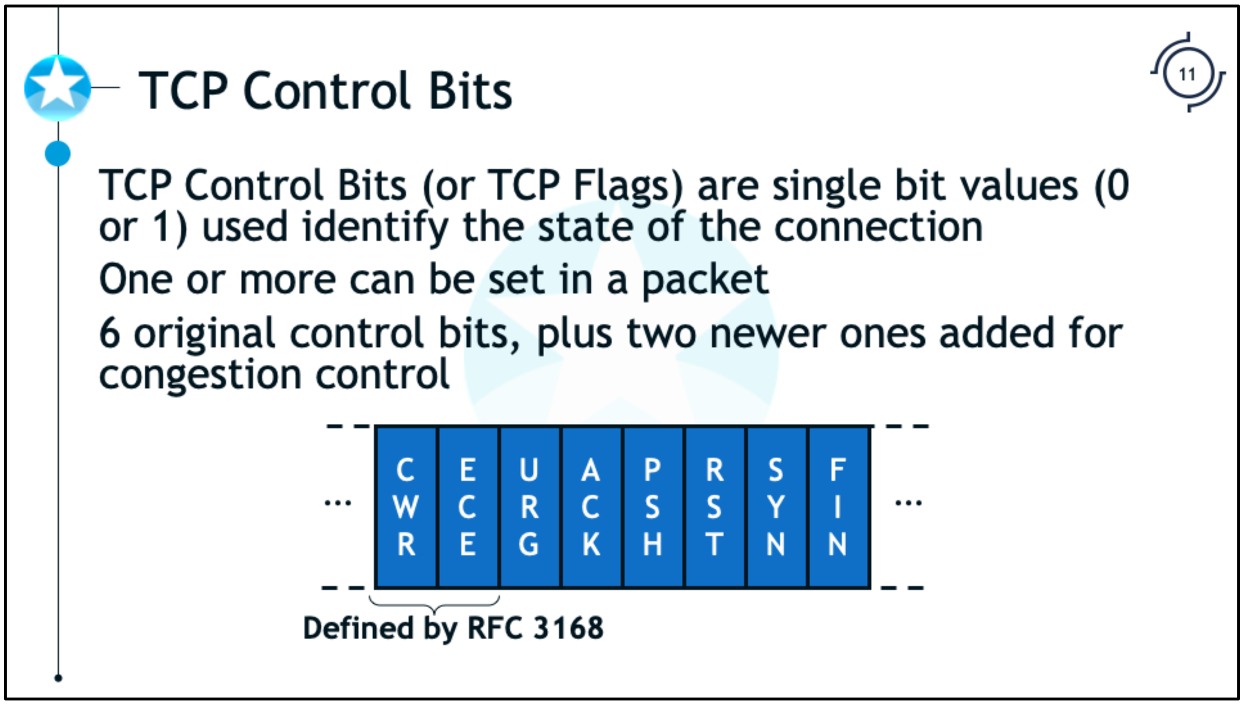


Transmission Control Protocol (TCP)

- TCP carries the majority of data on the Internet
- TCP adds ports, ensuring delivery to the correct service on a given IP address
- TCP provides reliable delivery by ensuring that data arrives intact, and in the correct order
  - TCP can detect lost data and request retransmission, and filter out duplicate data, using sequence and acknowledgement numbers
  - TCP uses a checksum to ensure the integrity of each packet
- TCP is designed for accurate delivery, not speedy delivery

TCP carries the majority of the data on the Internet today. TCP adds PORTS to the IP addresses established at the network layer by IP. TCP uses a "3 way handshake" to establish a connection between two hosts and provide "reliability" by tracking the data that flows between the hosts. TCP tracks the flow of data with SEQUENCE and ACKNOWLEGEMENT numbers on each packet. Sequence numbers and acknowledgement numbers are like tracking numbers on packages shipped through UPS. They can be used to detect if packets arrive in the wrong order (which could happen if packets take different paths on a network), or to detect if certain packets didn't make it through or got sent more than once. TCP also uses a checksum to ensure the data integrity of each packet.

TCP is designed more for accurate data transmission than speedy data transmission; it has a lot of overhead. For applications where timely delivery is more important, such as video streaming, other protocols may be more appropriate.

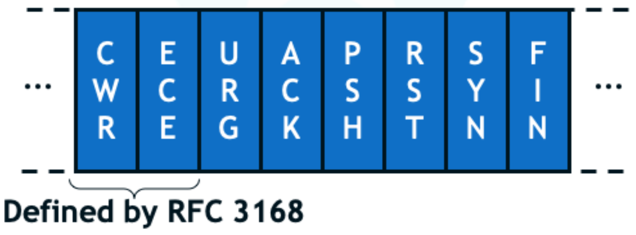


TCP Control Bits

TCP Control Bits (or TCP Flags) are single bit values (0 or 1) used identify the state of the connection

One or more can be set in a packet

6 original control bits, plus two newer ones added for congestion control



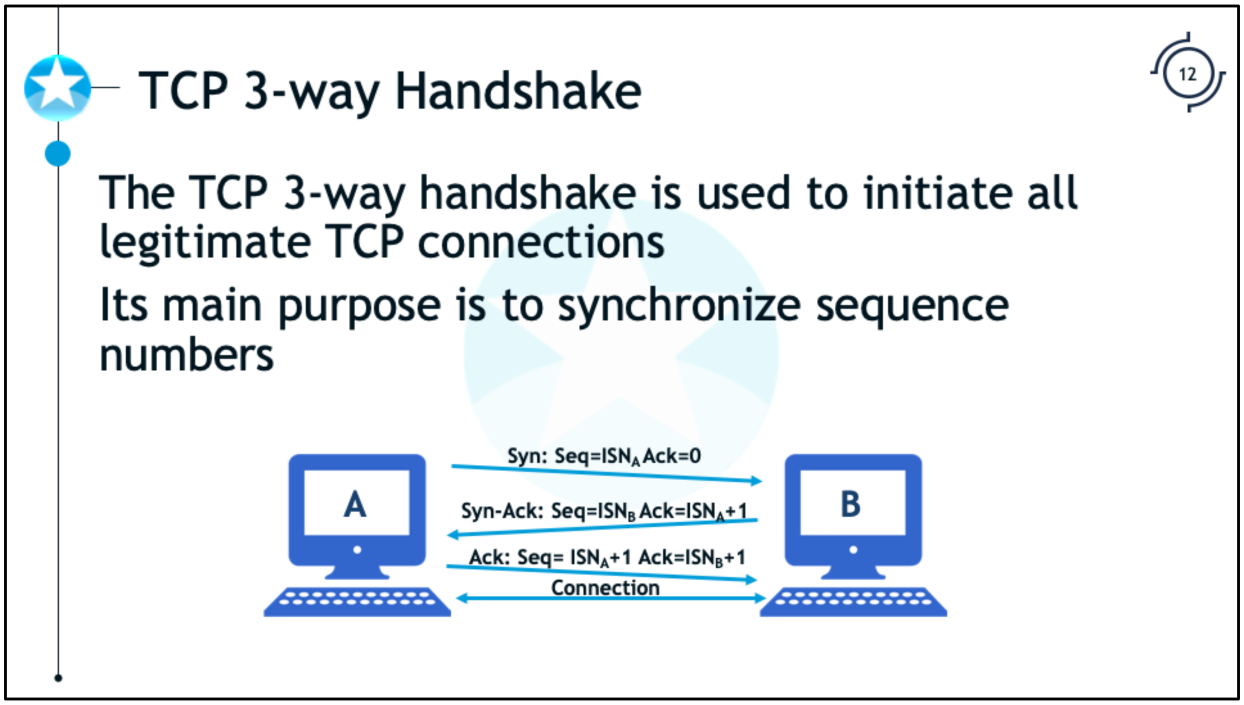
The TCP Control Bits (part of the TCP header) help identify the state of the TCP connection and which components of the TCP connection the given packet is associated with. There are six traditional TCP Control Bits, with 2 newer extended ones defined by RFC 3168. Each control bit can have a value of 0 or 1 (each one is just one bit long). The six traditional control bits include:

- SYN: The system should synchronize sequence numbers. This Control Bit is used during session establishment.
- ACK: The Acknowledgment field is significant. Packets with this bit set to 1 are acknowledging earlier packets.
- RST: The connection should be reset, due to error or other interruptions.
- FIN: There is no more data from the sender. Therefore, the session should be gracefully torn down.
- PSH: This bit indicates that data should be flushed through the TCP layer immediately rather than holding it and waiting for more data.
- URG: The Urgent Pointer in the TCP header is significant. There is important data there that should be handled quickly.

Note that this list doesn't show the Control Bits in the order in which they appear in the packet. Instead, we have sorted them in a more memorable fashion. The two additional control bits are CWR and ECE, which are:

- CWR: Congestion Window Reduced, which indicates that, due to network congestion, the queue of outstanding packets to send has been lowered.
- ECE: Explicit Congestion Notification Echo, which indicates that the connection is experiencing congestion.

Each of these control bits can be set independently of the others. Thus, we can have a single packet that is simultaneously a SYN and an ACK.



TCP 3-way Handshake

The TCP 3-way handshake is used to initiate all legitimate TCP connections

Its main purpose is to synchronize sequence numbers

A
B
Syn: Seq=ISN_A Ack=0
Syn-Ack: Seq=ISN_B Ack=ISN_A+1
Ack: Seq= ISN_A+1 Ack=ISN_B+1
Connection

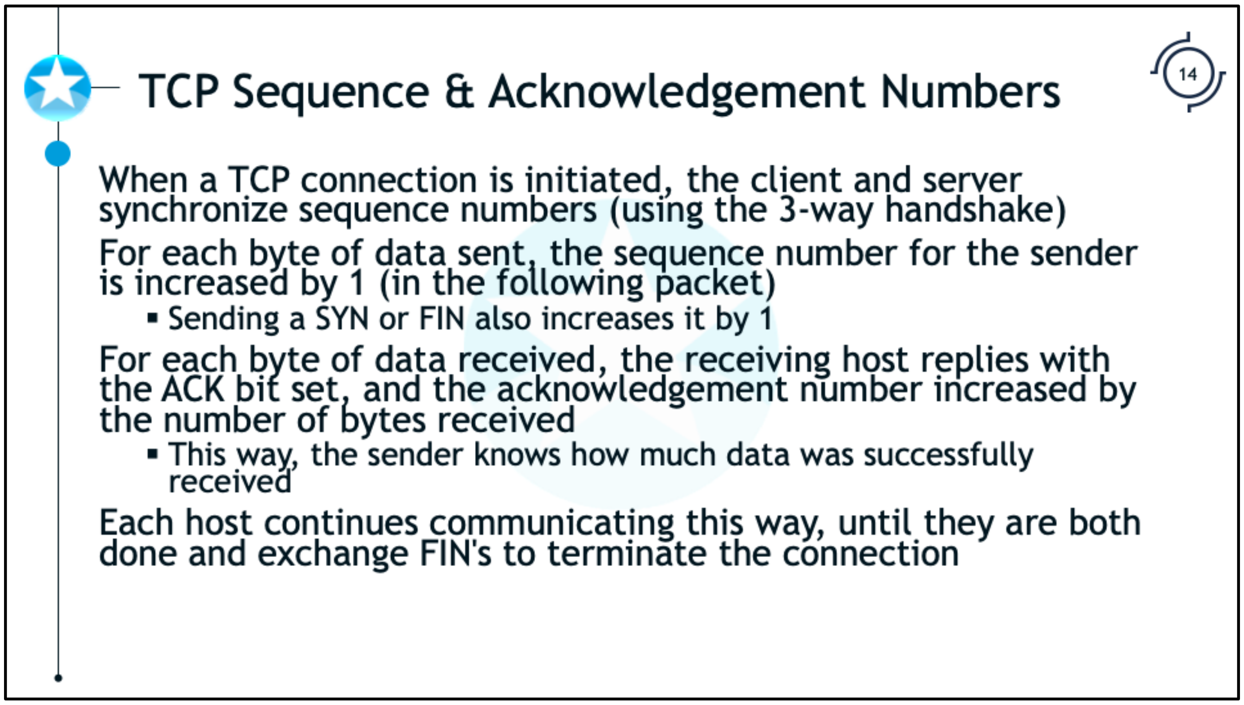
Every legitimate TCP connection begins with the TCP three-way handshake, which is used to exchange sequence numbers so that lost packets can be retransmitted and packets can be placed in the proper order.

If machine A wants to initiate a connection to machine B, it will start by sending a TCP packet with the SYN Control Bit set. This packet will include an initial sequence number (which we'll call ISN_A because it comes from machine A), which is 32-bits long and typically generated in a pseudo-random fashion by the TCP software on machine A. The ACK number (another 32 bits in the TCP header) is typically set to zero, because it is ignored in this initial SYN. Some operating system variants may make this ACK number non-zero. Either way, it is ignored by the destination machine.

If the destination port is open (that is, there is something listening on that port), it must respond with a SYN-ACK packet back (a packet that has both the SYN and ACK Control Bits set at the same time). This packet will have a sequence number of ISN_B, a pseudo-random number assigned by machine B for this connection. The SYN-ACK packet will have an acknowledgment number of ISN_A+1, indicating that machine B has acknowledged the SYN packet from machine A.

To complete the three-way handshake, machine A responds with an ACK packet which has a sequence number of ISN_A+1 (it's the next packet, so the sequence number has to change from the value in the original SYN packet). The acknowledgment number field is set to ISN_B+1, thereby acknowledging the SYN-ACK packet.

We have now exchanged sequence numbers. All packets going from A to B will have increasing sequence numbers starting at ISN_A+1, going up by a value of 1 for each byte of data transmitted in the payloads of A to B packets. Likewise, all responses back from B will have sequence numbers starting at ISN_B+1 and going up for each



TCP Sequence & Acknowledgement Numbers

When a TCP connection is initiated, the client and server synchronize sequence numbers (using the 3-way handshake)

For each byte of data sent, the sequence number for the sender is increased by 1 (in the following packet)

- Sending a SYN or FIN also increases it by 1

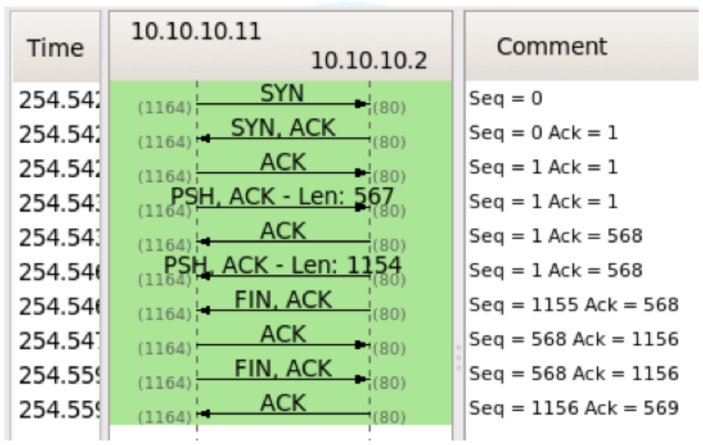
For each byte of data received, the receiving host replies with the ACK bit set, and the acknowledgement number increased by the number of bytes received

- This way, the sender knows how much data was successfully received

Each host continues communicating this way, until they are both done and exchange FIN's to terminate the connection

During the 3-way handshake, the client and server synchronize their sequence numbers (note that they don't use the same sequence numbers, they simply exchange sequence numbers so they can keep track of each other). From that point on, the sequence number is increased by 1 for each byte of data sent by a host. So, for example, if host A sends 5 bytes of data and has a sequence number of 30, the next packet host A sends will have a sequence number of 35. Meanwhile, for each packet of data received, the receiving host responds with a packet with the ACK bit set, and the acknowledgement number increased by the number of bytes received. In the previous example, the host receiving 5 bytes of data with a current acknowledgement number of 1 would reply with an acknowledgement number of 6. By comparing these numbers, both sides of the connection are able to know if any data was lost, if packets were received in the wrong order, etc. Finally, when the exchange is over, the server will send a packet with the FIN and ACK bits set to let the client know that it is done transmitting data. The client will reply with an ACK, followed by its own FIN/ACK. Then, the server will reply with an ACK, and the connection is closed.

Example TCP Transaction

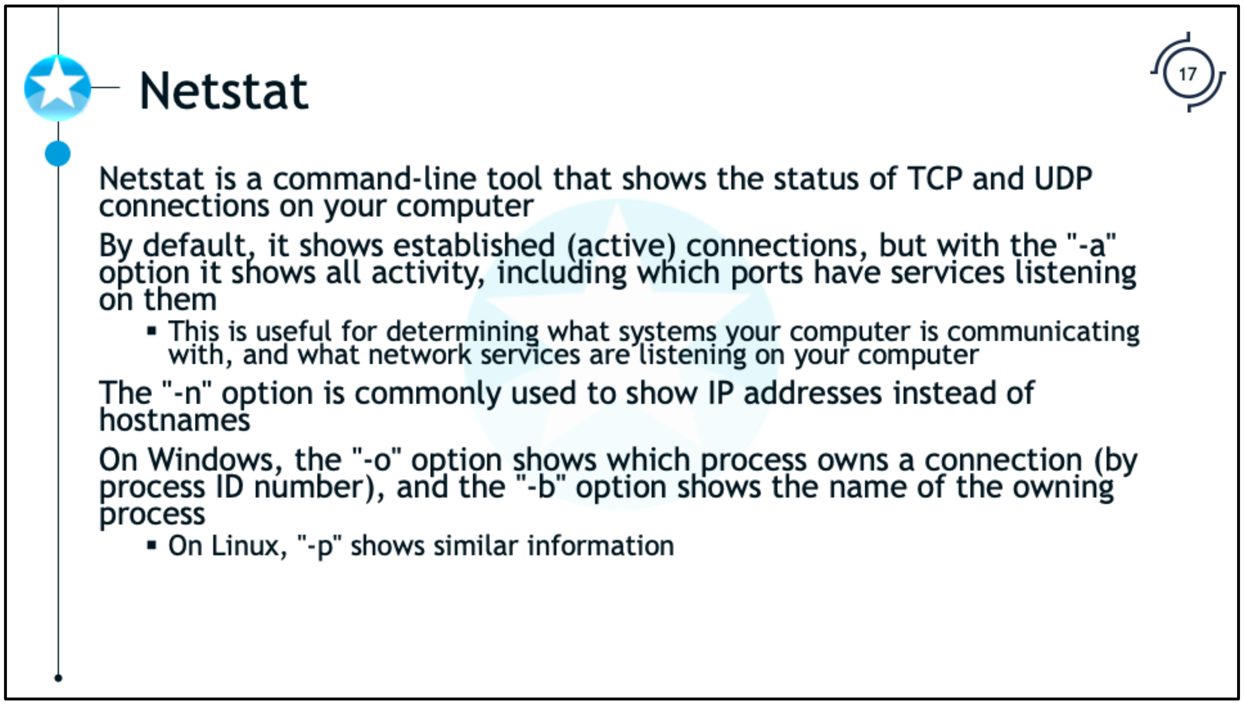


Time	10.10.10.11		10.10.10.2	Comment
254.54	(1164)	SYN →	(80)	Seq = 0
254.54	(1164)	← SYN, ACK	(80)	Seq = 0 Ack = 1
254.54	(1164)	ACK →	(80)	Seq = 1 Ack = 1
254.54	(1164)	PSH, ACK - Len: 567 →	(80)	Seq = 1 Ack = 1
254.54	(1164)	← ACK	(80)	Seq = 1 Ack = 568
254.54	(1164)	← PSH, ACK - Len: 1154	(80)	Seq = 1 Ack = 568
254.54	(1164)	← FIN, ACK	(80)	Seq = 1155 Ack = 568
254.54	(1164)	ACK →	(80)	Seq = 568 Ack = 1156
254.55	(1164)	FIN, ACK →	(80)	Seq = 568 Ack = 1156
254.55	(1164)	← ACK	(80)	Seq = 1156 Ack = 569

Here is a simple TCP transaction, as illustrated by Wireshark's "Flow Graph" tool. Each line represents a single packet. The client that initiated the connection is on the left, and the server is on the right. Packets flowing from the client to the server are represented by an arrow pointing to the right, and packets flowing from the server to the client are represented by an arrow pointing to the left. Note that Wireshark displays relative sequence and acknowledgement numbers by default, to make it easier to follow a series of packets.

The first three packets are the TCP 3-way handshake. The client (10.10.10.11) sends a TCP packet with the SYN bit set from port 1164 to the server (10.10.10.2), port 80 (meaning this is likely an HTTP request), with a sequence number of 0. The server then replies with the SYN and ACK bits set, with its own sequence number (shown as zero since it's relative) and an acknowledgement number of 1 (since it's acknowledging the first packet). Then, the client complete the handshake by replying with just the ACK bit set, increasing its sequence number to 1 and keeping its acknowledgement number at 1.

The fourth packet is the beginning of the data transfer. The client sends a request to the server that is 567 bytes long. This packet has the PSH and ACK bits set, and still has a sequence and acknowledgment number of 1 (since no other data has been exchanged since the handshake). The server then responds with the ACK bit set, and an acknowledgement number of 568, which serves to acknowledge that it has received the first 567 bytes. Since the packet it received had the PSH bit set, it processes it right away, and then sends a response. The response also has the PSH and ACK bits set, has a length of 1154, and still has the sequence number set to 1 (since this is the first data it has sent since the handshake) and acknowledgement number set to 568 (since that is still how much data it has received so far from the client).

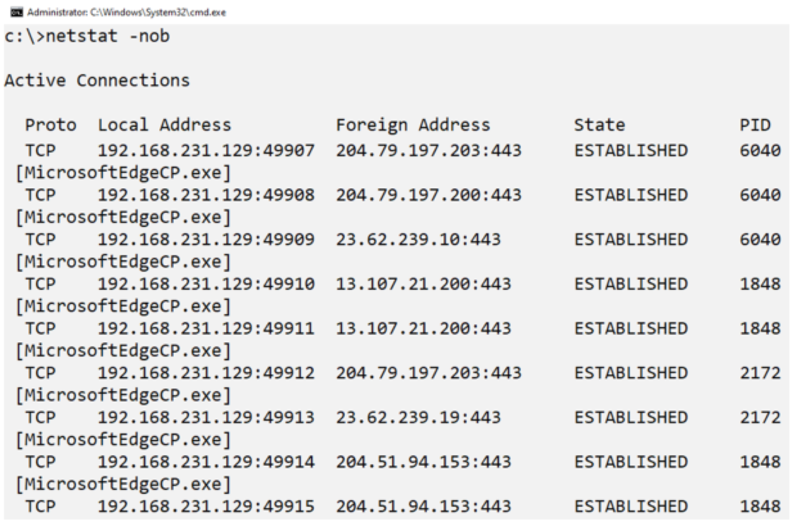


Netstat

- Netstat is a command-line tool that shows the status of TCP and UDP connections on your computer
- By default, it shows established (active) connections, but with the "-a" option it shows all activity, including which ports have services listening on them
  - This is useful for determining what systems your computer is communicating with, and what network services are listening on your computer
- The "-n" option is commonly used to show IP addresses instead of hostnames
- On Windows, the "-o" option shows which process owns a connection (by process ID number), and the "-b" option shows the name of the owning process
  - On Linux, "-p" shows similar information

Netstat is a very useful command-line tool that shows the current status of TCP and UDP connection on your computer. By default, it shows established (or active) connections (and on Linux, it will show a ton of information on sockets). The "-a" option tells Netstat to show all activity, including which ports are open with services listening on them. This is useful for determining what systems your computer is communicating with, and what network services are listening on your computer. The "-n" option is commonly used to show IP addresses instead of hostnames (without "-n", Netstat will attempt to do reverse DNS lookups of each IP address). On Windows, the "-o" option can be used to show the process ID (PID) number of the process (or program) on the system owns a connection. The "-b" option shows the name of the owning process. On Linux, the "-p" option shows similar information.

Netstat Exercise

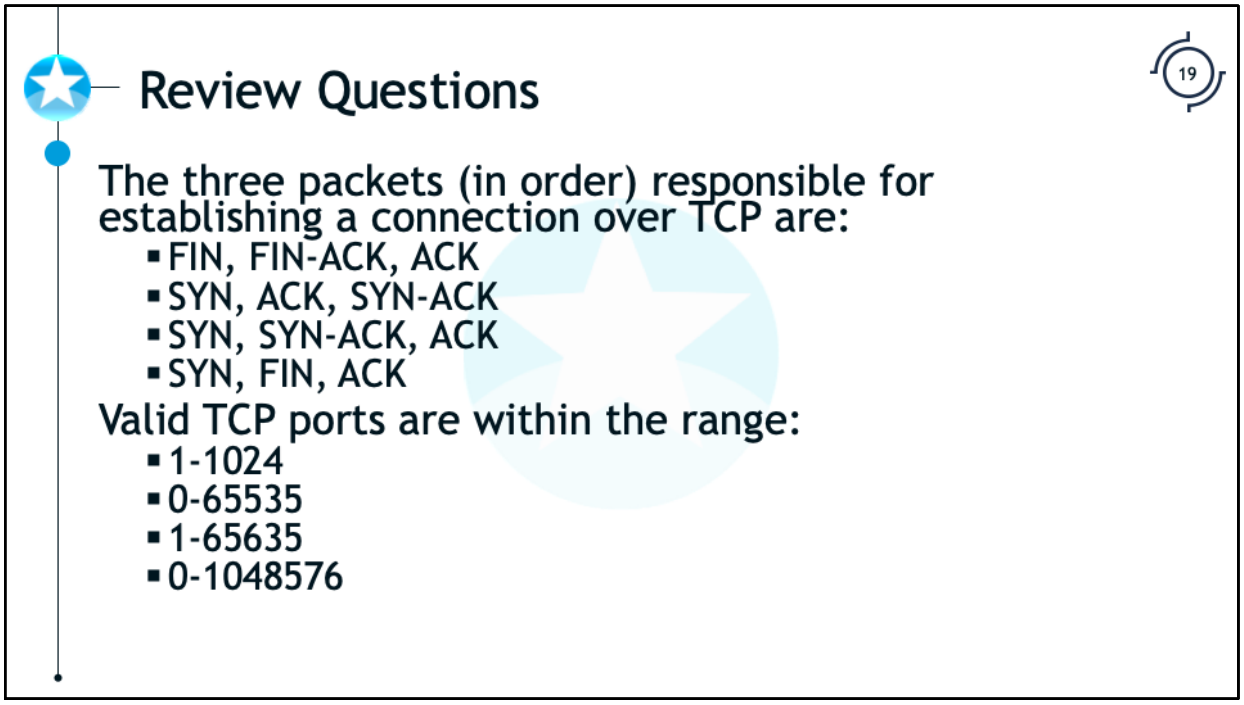

Administrator: C:\Windows\System32\cmd.exe

```
c:\>netstat -nob

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    192.168.231.129:49907  204.79.197.203:443     ESTABLISHED     6040
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49908  204.79.197.200:443     ESTABLISHED     6040
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49909  23.62.239.10:443       ESTABLISHED     6040
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49910  13.107.21.200:443      ESTABLISHED     1848
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49911  13.107.21.200:443      ESTABLISHED     1848
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49912  204.79.197.203:443     ESTABLISHED     2172
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49913  23.62.239.19:443       ESTABLISHED     2172
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49914  204.51.94.153:443      ESTABLISHED     1848
 [MicrosoftEdgeCP.exe]
  TCP    192.168.231.129:49915  204.51.94.153:443      ESTABLISHED     1848
```

On your own computer, open a Command Prompt, and type the command "netstat -nob", but don't press enter yet. Open a web browser and surf to the site **https://isc.sans.org/**. Now, while the page is still loading, quickly go back to the command prompt and press enter. You should see some output similar to the above, showing multiple established connections to port 443 on the isc.sans.org server (204.51.94.153). In the example above, the ephemeral ports used on the local system began at 49907 and were incremented sequentially for each individual connection to the web server. The different connections seen above are likely due to some content being delivered using content delivery networks and cloud-hosted storage. These connections are all associated with "MicrosoftEdgeCP.exe."



Review Questions

The three packets (in order) responsible for establishing a connection over TCP are:

- FIN, FIN-ACK, ACK
- SYN, ACK, SYN-ACK
- SYN, SYN-ACK, ACK
- SYN, FIN, ACK

Valid TCP ports are within the range:

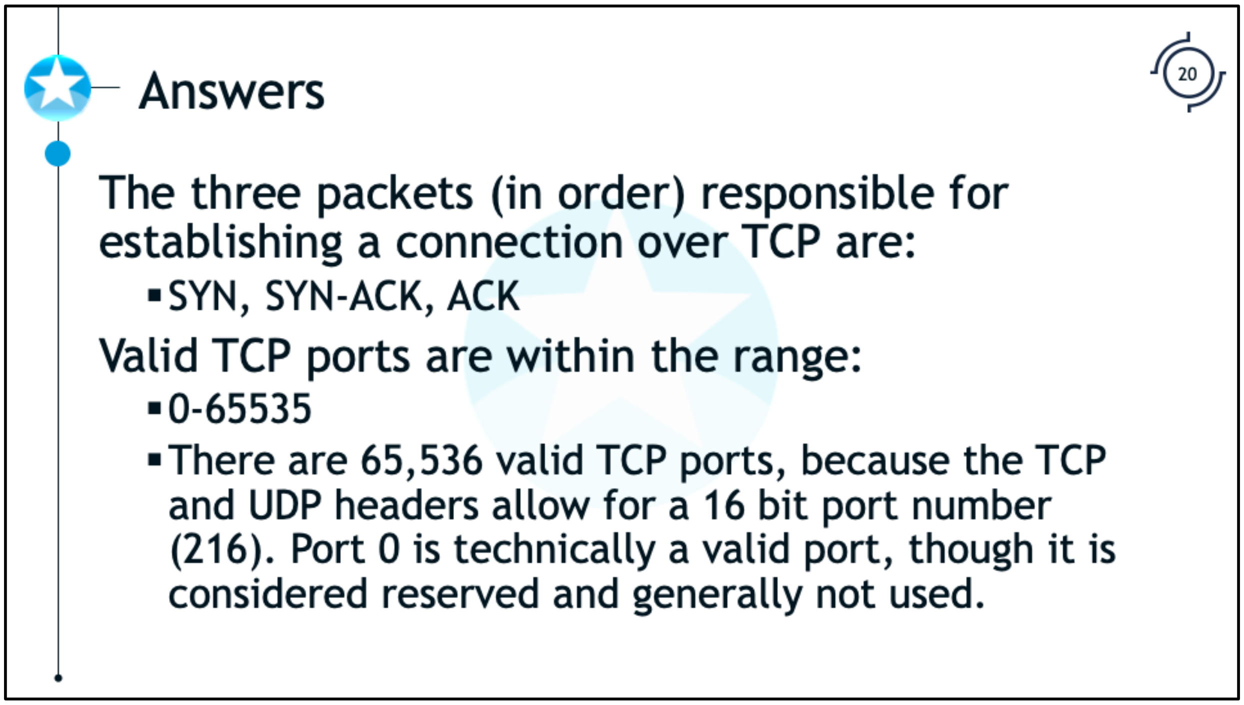
- 1-1024
- 0-65535
- 1-65635
- 0-1048576

The three packets (in order) responsible for establishing a connection over TCP are:

- FIN, FIN-ACK, ACK
- SYN, ACK, SYN-ACK
- SYN, SYN-ACK, ACK
- SYN, FIN, ACK

Valid TCP ports are within the range:

- 1-1024
- 0-65535
- 1-65635
- 0-1048576



Answers

The three packets (in order) responsible for establishing a connection over TCP are:

- SYN, SYN-ACK, ACK

Valid TCP ports are within the range:

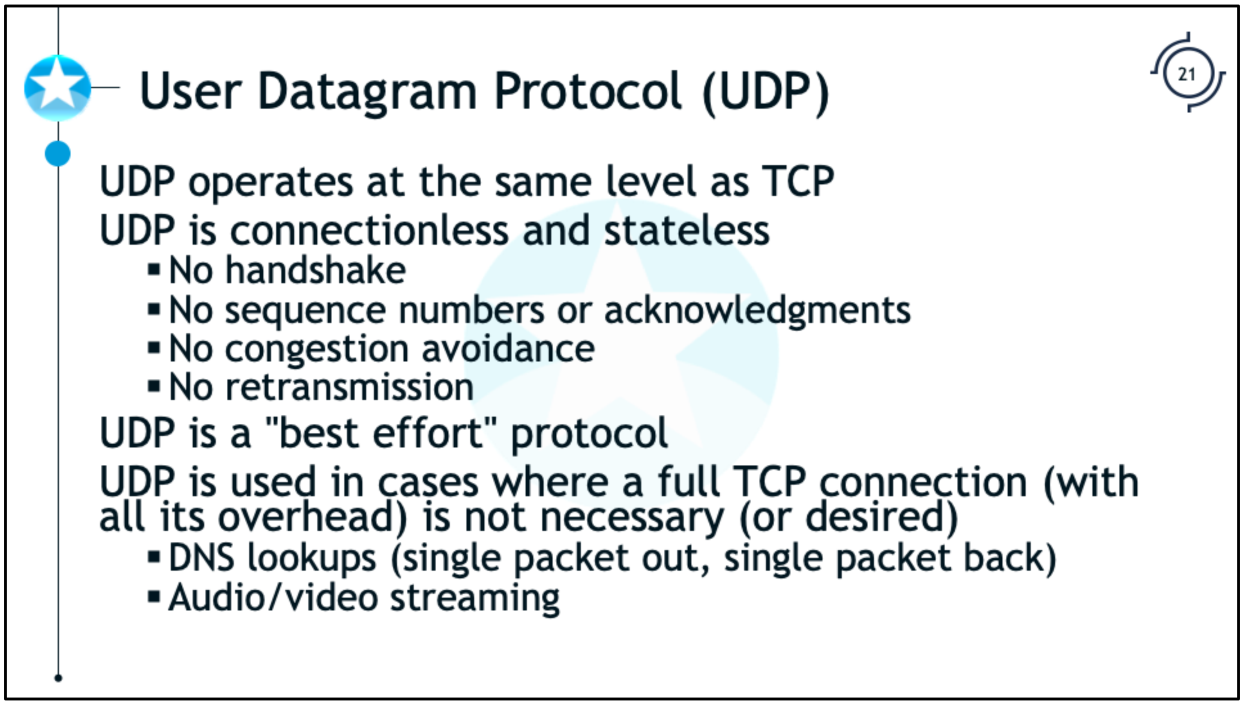
- 0-65535
- There are 65,536 valid TCP ports, because the TCP and UDP headers allow for a 16 bit port number (216). Port 0 is technically a valid port, though it is considered reserved and generally not used.

The three packets (in order) responsible for establishing a connection over TCP are:

- SYN, SYN-ACK, ACK

Valid TCP ports are within the range:

- 0-65535
- There are 65,536 valid TCP ports, because the TCP and UDP headers allow for a 16 bit port number (2^{16}). Port 0 is technically a valid port, though it is considered reserved and generally not used.



User Datagram Protocol (UDP)

UDP operates at the same level as TCP

UDP is connectionless and stateless

- No handshake
- No sequence numbers or acknowledgments
- No congestion avoidance
- No retransmission

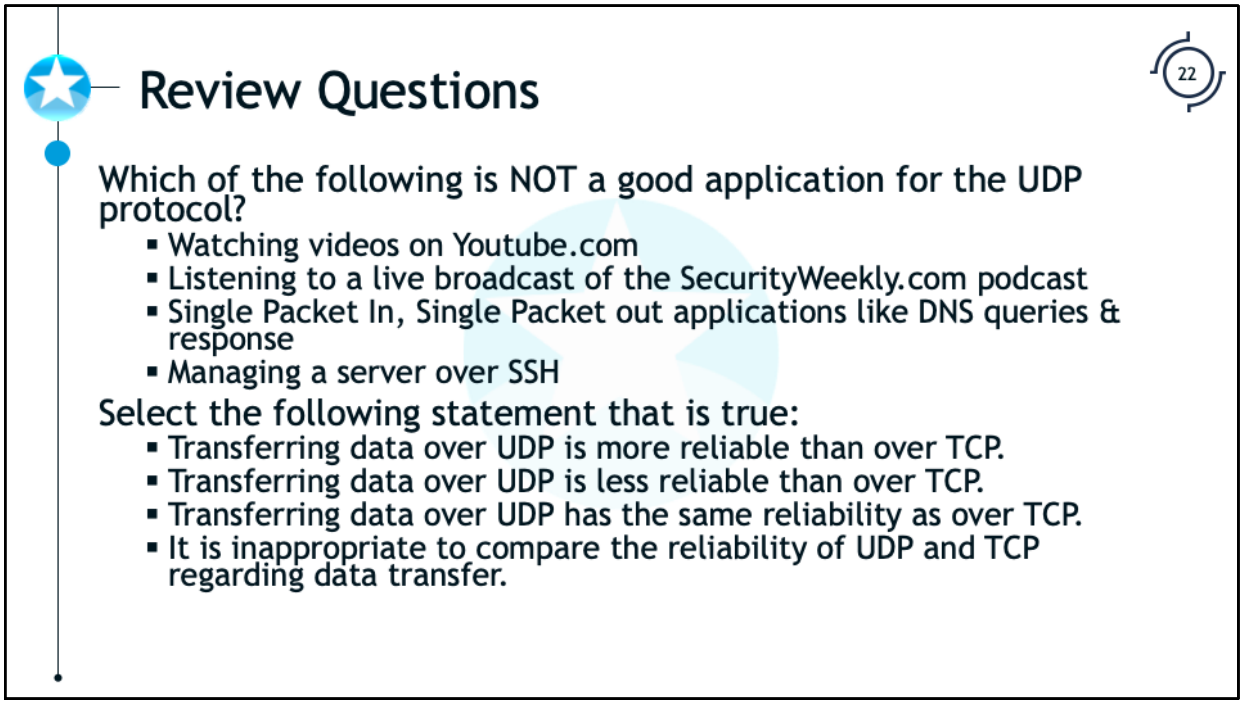
UDP is a "best effort" protocol

UDP is used in cases where a full TCP connection (with all its overhead) is not necessary (or desired)

- DNS lookups (single packet out, single packet back)
- Audio/video streaming

UDP operates at the same level as TCP. UDP also assigns ports to distinguish between different services but it does not use the 3 way handshake or Sequence and Acknowledgement numbers to track packets. As a result, UDP is lightweight and faster than TCP. The speed is gained at the cost of reliability. UDP is a "best effort" protocol: the sender transmits their packets and just hopes that they reach the destination. There will be no indication if a packet doesn't make it, or comes in the wrong order, and there is no mechanism for retransmission of lost or damaged packets (though there is a checksum). There is also no mechanism for congestion avoidance (such as slowing transmission for a slow or congested network link). Rather, the application itself is responsible for ensuring data integrity. For this reason, UDP is sometimes referred to as the "*Unreliable* Datagram Protocol".

UDP is great for applications such as DNS where only a single packet is transmitted in each direction. If I send a single packet DNS request out and don't get a response, I know the sender didn't get it. If I get a response, they did receive my request. So for single packet transmissions, the overhead of session tracking is unnecessary. UDP is also very good for data that will be interpreted by the human brain. With audio and video transmissions, the human brain will compensate for small gaps in the data, so it is much better to allow for them rather than trying to retransmit lost packets. Other common applications and protocols that operate over UDP include RIP (Routing Information Protocol), DHCP, NTP (Network Time Protocol), and TFTP.



Review Questions

Which of the following is NOT a good application for the UDP protocol?

- Watching videos on Youtube.com
- Listening to a live broadcast of the SecurityWeekly.com podcast
- Single Packet In, Single Packet out applications like DNS queries & response
- Managing a server over SSH

Select the following statement that is true:

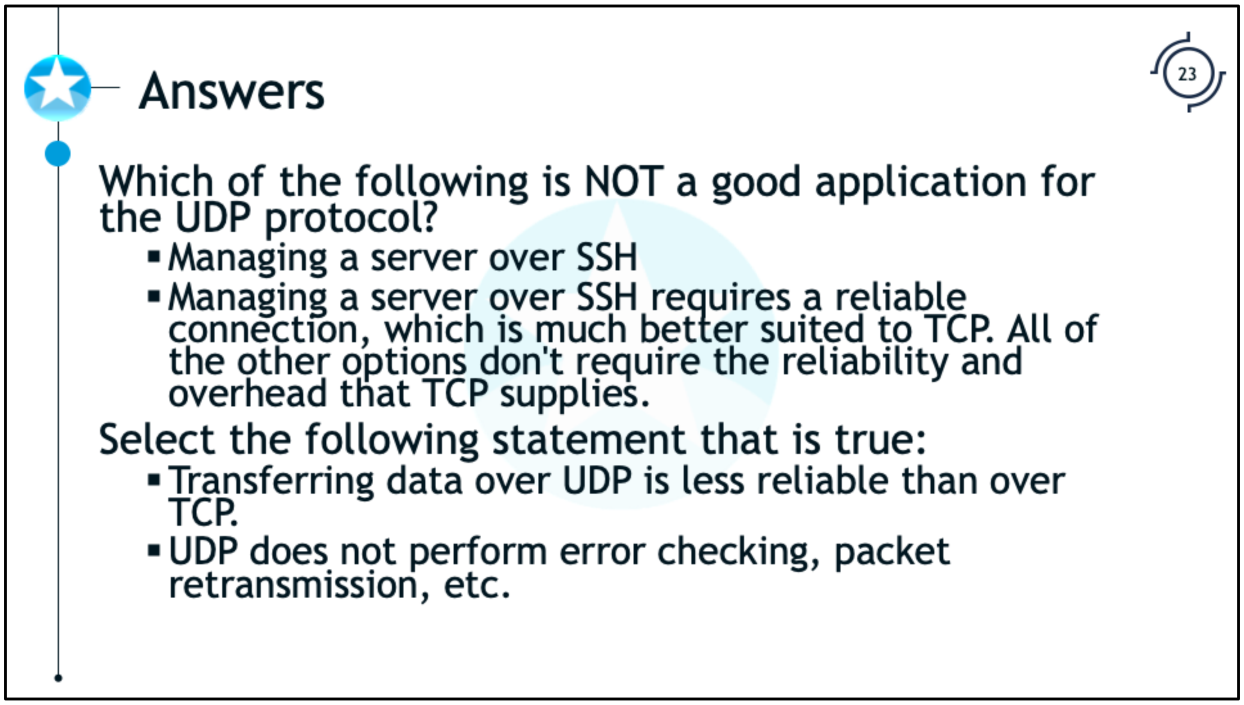
- Transferring data over UDP is more reliable than over TCP.
- Transferring data over UDP is less reliable than over TCP.
- Transferring data over UDP has the same reliability as over TCP.
- It is inappropriate to compare the reliability of UDP and TCP regarding data transfer.

Which of the following is NOT a good application for the UDP protocol?

- Watching videos on Youtube.com
- Listening to a live broadcast of the SecurityWeekly.com podcast
- Single Packet In, Single Packet out applications like DNS queries & response
- Managing a server over SSH

Select the following statement that is true:

- Transferring data over UDP is more reliable than over TCP.
- Transferring data over UDP is less reliable than over TCP.
- Transferring data over UDP has the same reliability as over TCP.
- It is inappropriate to compare the reliability of UDP and TCP regarding data transfer.



Answers

Which of the following is NOT a good application for the UDP protocol?

- Managing a server over SSH
- Managing a server over SSH requires a reliable connection, which is much better suited to TCP. All of the other options don't require the reliability and overhead that TCP supplies.

Select the following statement that is true:

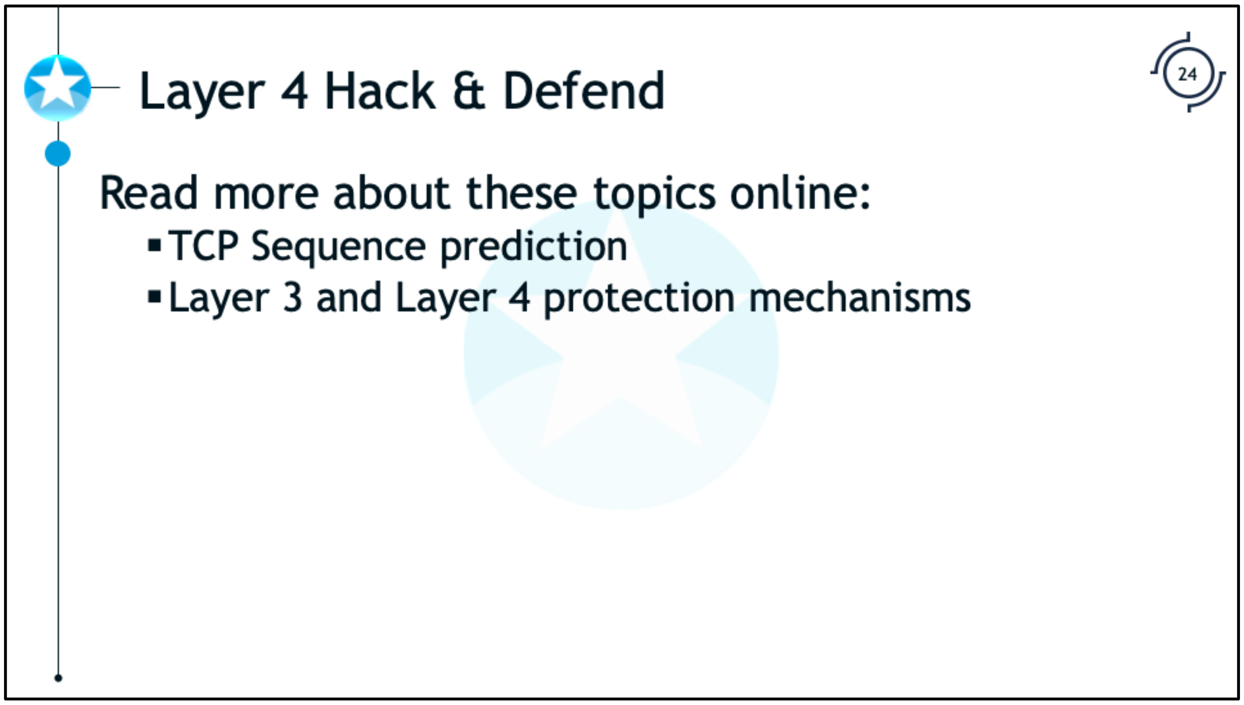
- Transferring data over UDP is less reliable than over TCP.
- UDP does not perform error checking, packet retransmission, etc.

Which of the following is NOT a good application for the UDP protocol?

- Managing a server over SSH
- Managing a server over SSH requires a reliable connection, which is much better suited to TCP. All of the other options don't require the reliability and overhead that TCP supplies.

Select the following statement that is true:

- Transferring data over UDP is less reliable than over TCP.
- UDP does not perform error checking, packet retransmission, etc.



Layer 4 Hack & Defend

Read more about these topics online:

- TCP Sequence prediction
- Layer 3 and Layer 4 protection mechanisms

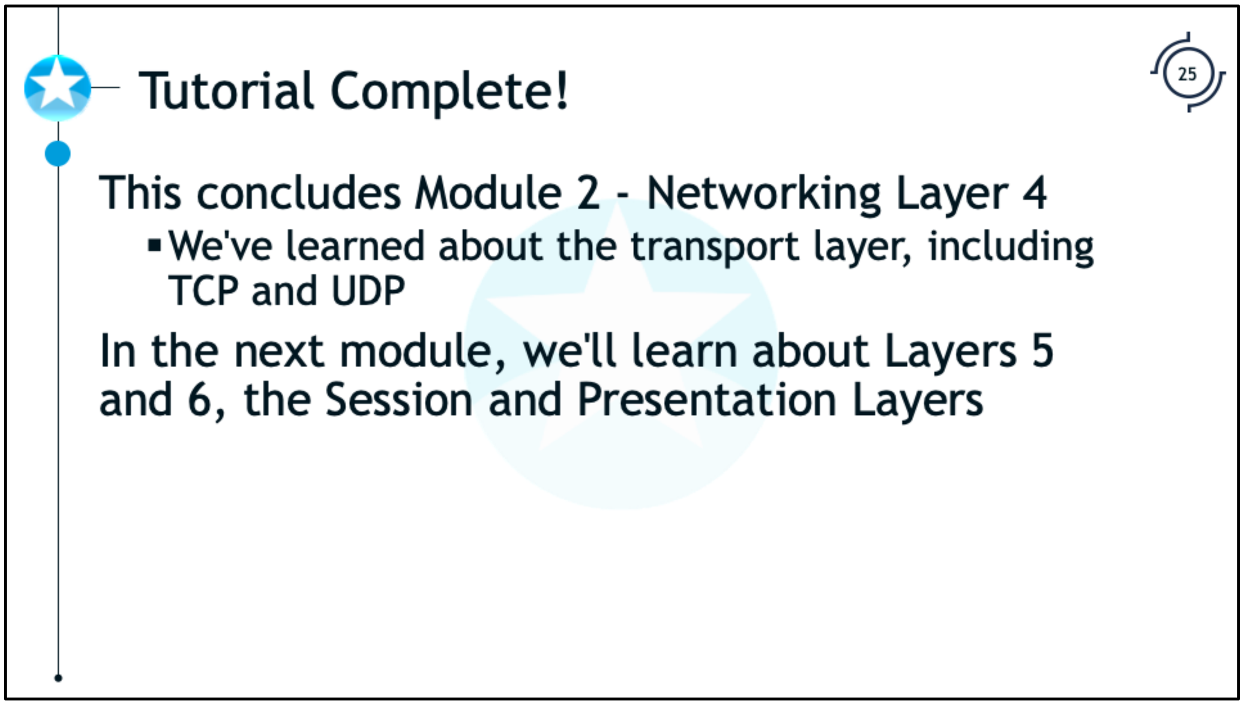
Read more about these topics online:

TCP Sequence prediction:

https://www.redsiege.com/ca/tcp-sequence-prediction

Layer 3 and Layer 4 protection mechanisms:

https://www.redsiege.com/ca/network-infrastructure-defense



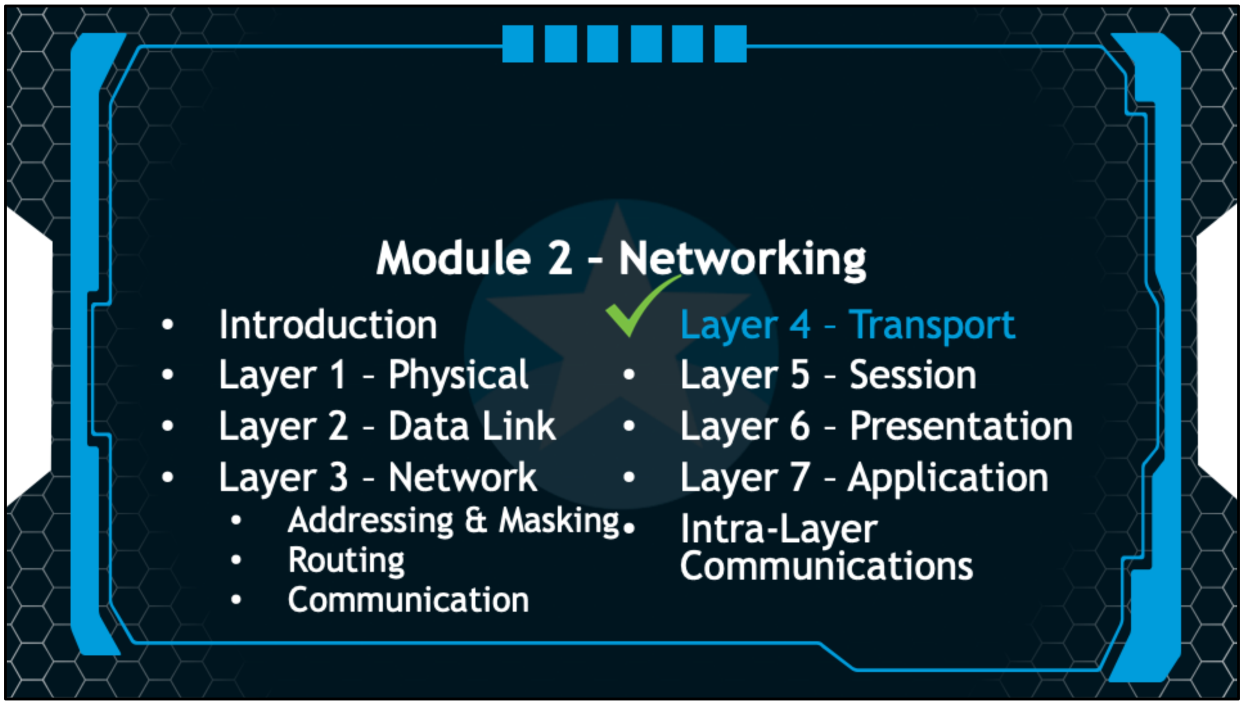
Tutorial Complete!

This concludes Module 2 - Networking Layer 4

- We've learned about the transport layer, including TCP and UDP

In the next module, we'll learn about Layers 5 and 6, the Session and Presentation Layers

This concludes the discussion about Layer 4, the Transport Layer. In the next tutorial we'll discuss the next two layers layer in the OSI model, the Session and Presentation Layers.



Module 2 - Networking

- Introduction
- Layer 1 – Physical
- Layer 2 – Data Link
- Layer 3 – Network
  - Addressing & Masking
  - Routing
  - Communication
- ✓ Layer 4 – Transport
- Layer 5 – Session
- Layer 6 – Presentation
- Layer 7 – Application
- Intra-Layer Communications

In the next session we discuss Layers 5 and 6.