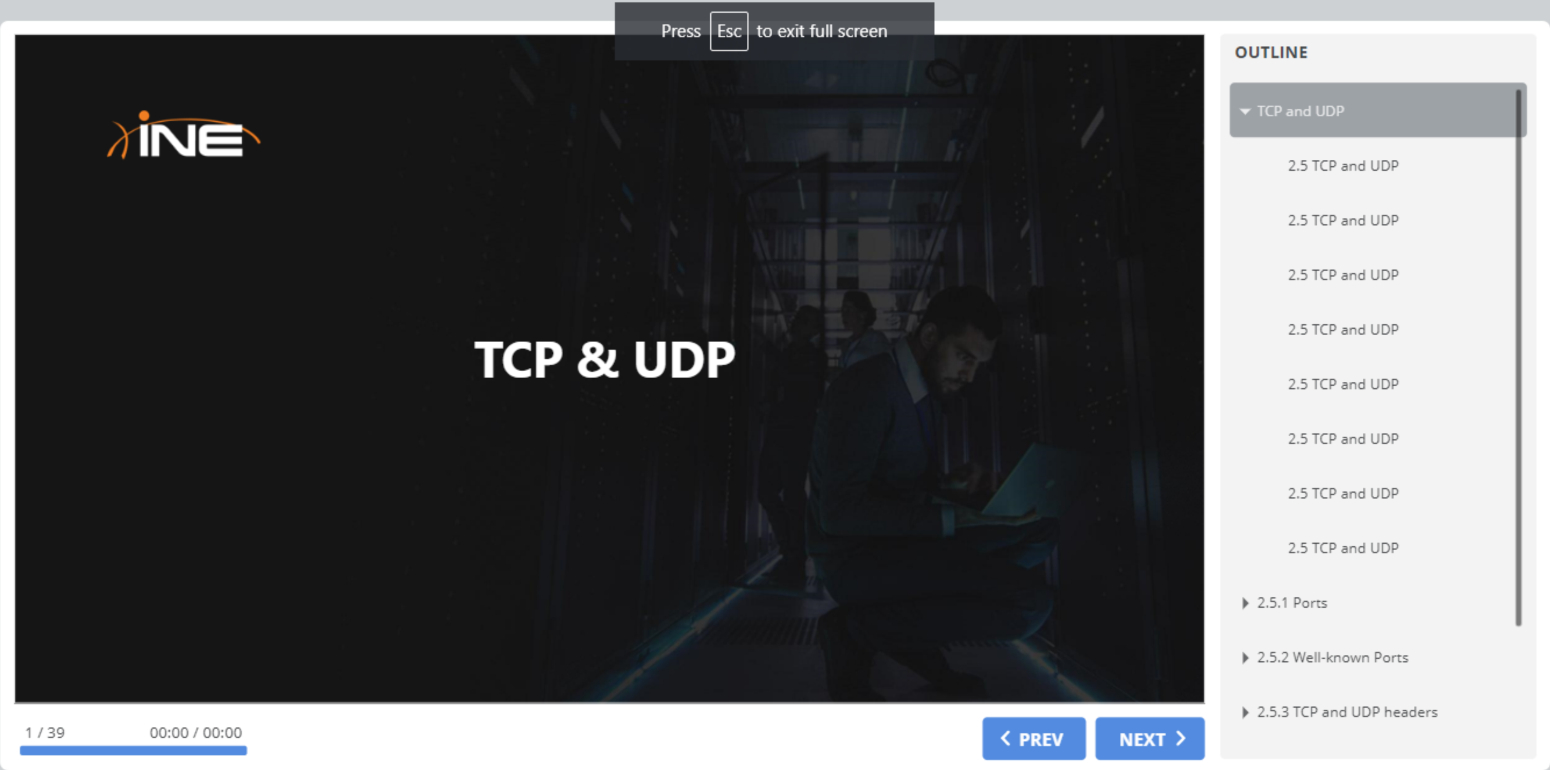
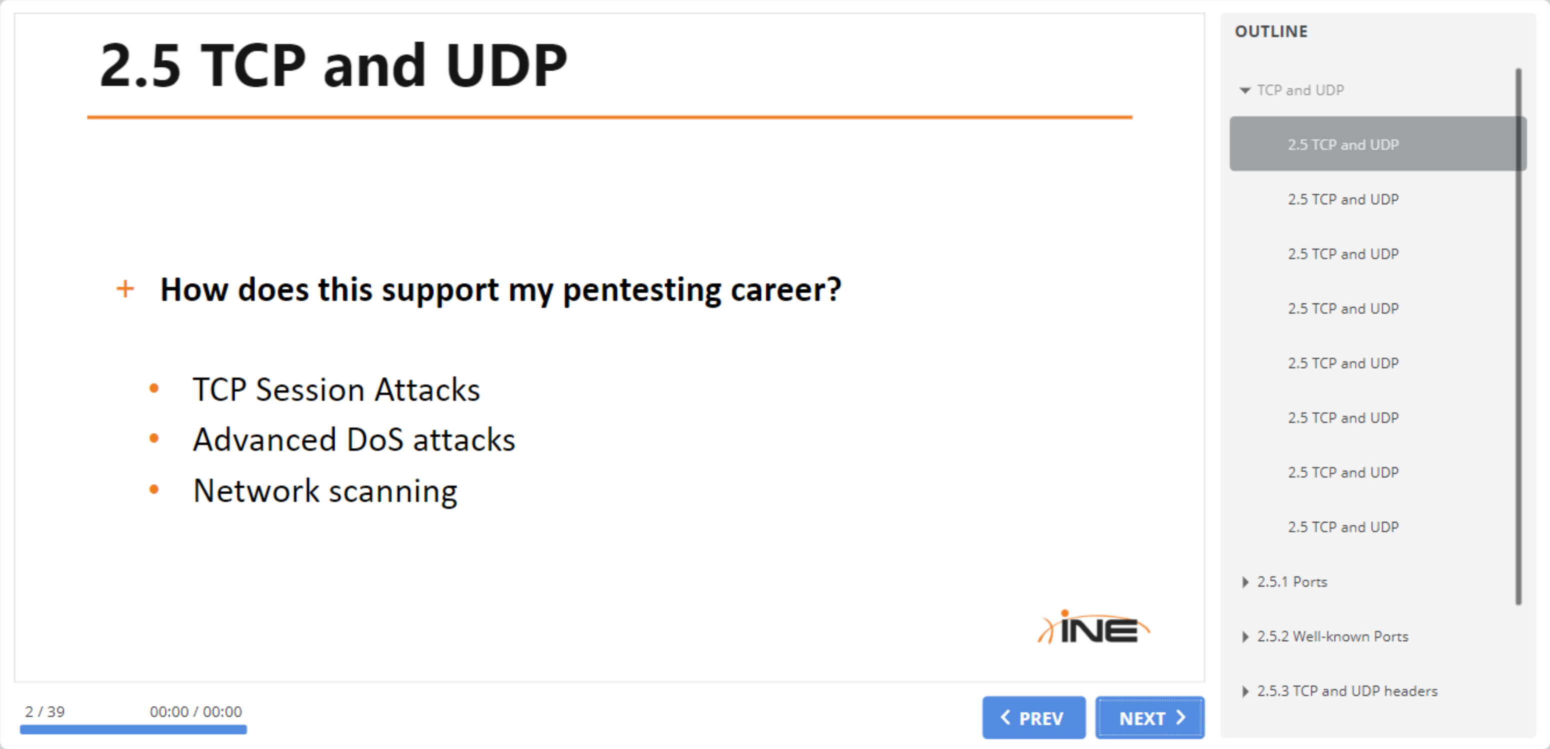


Press Esc to exit full screen
INE
OUTLINE
TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5 TCP and UDP
2.5.1 Ports
2.5.2 Well-known Ports
2.5.3 TCP and UDP headers
1 / 39
00:00 / 00:00
PREV
NEXT

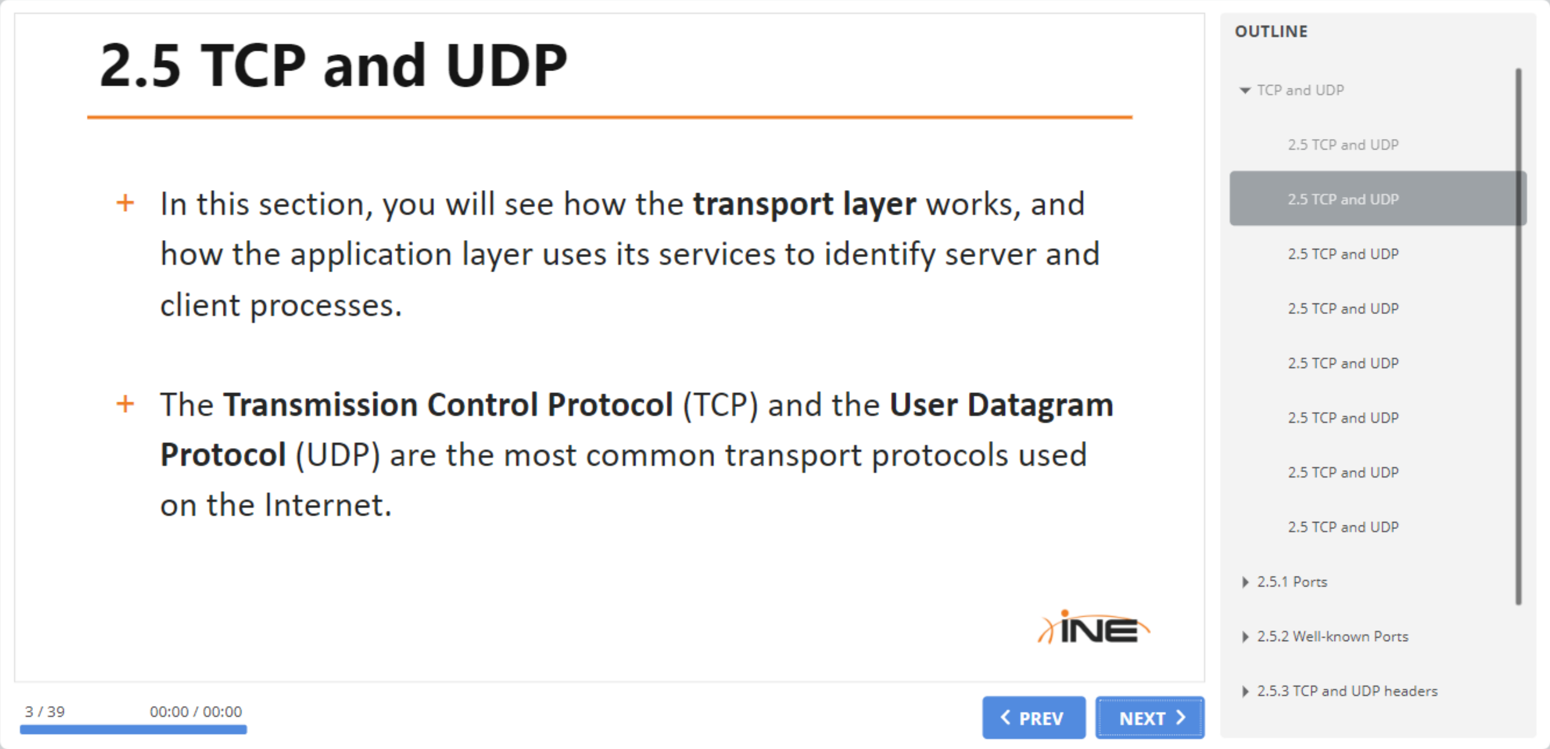
TCP & UDP



2.5 TCP and UDP

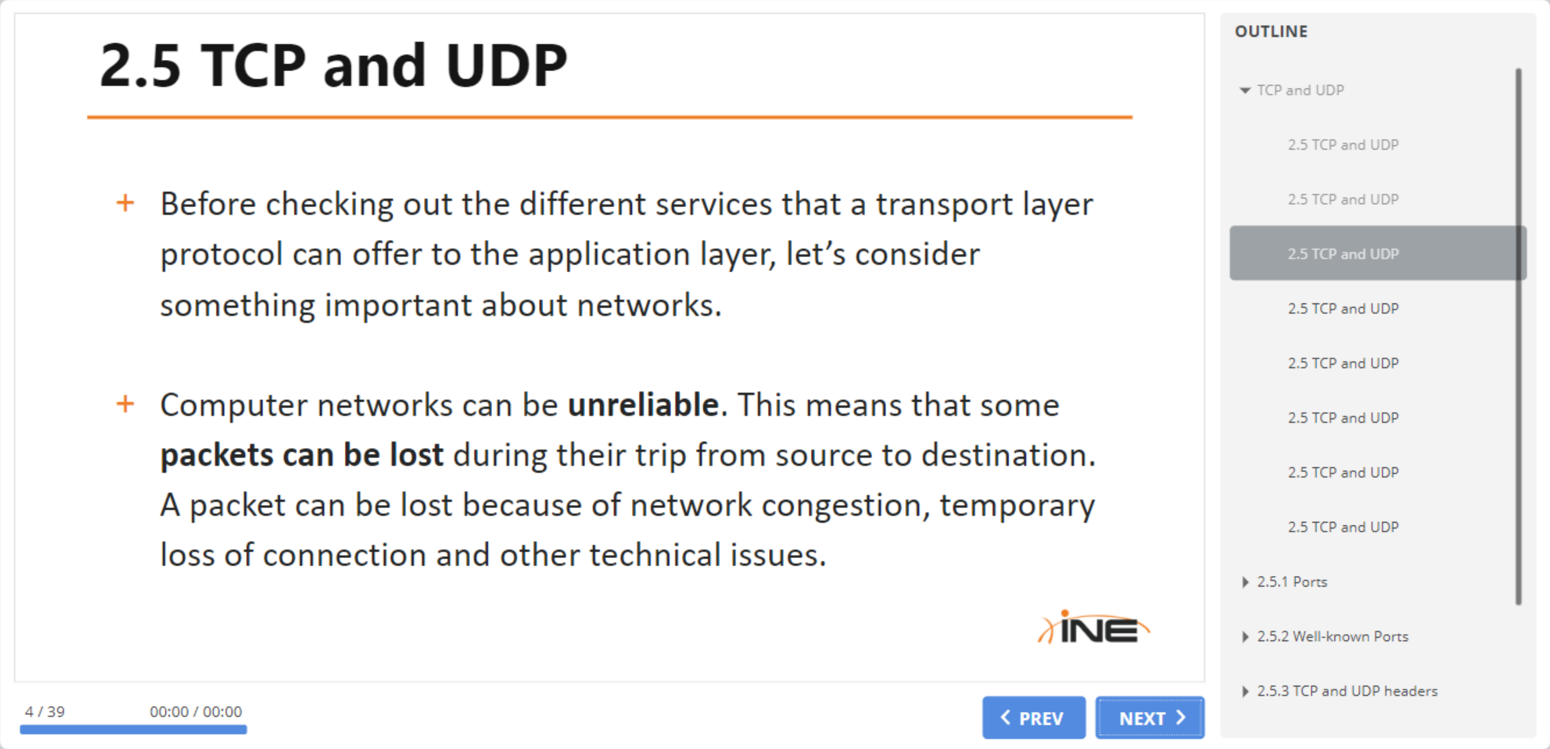
+ **How does this support my pentesting career?**

  - TCP Session Attacks
  - Advanced DoS attacks
  - Network scanning



2.5 TCP and UDP

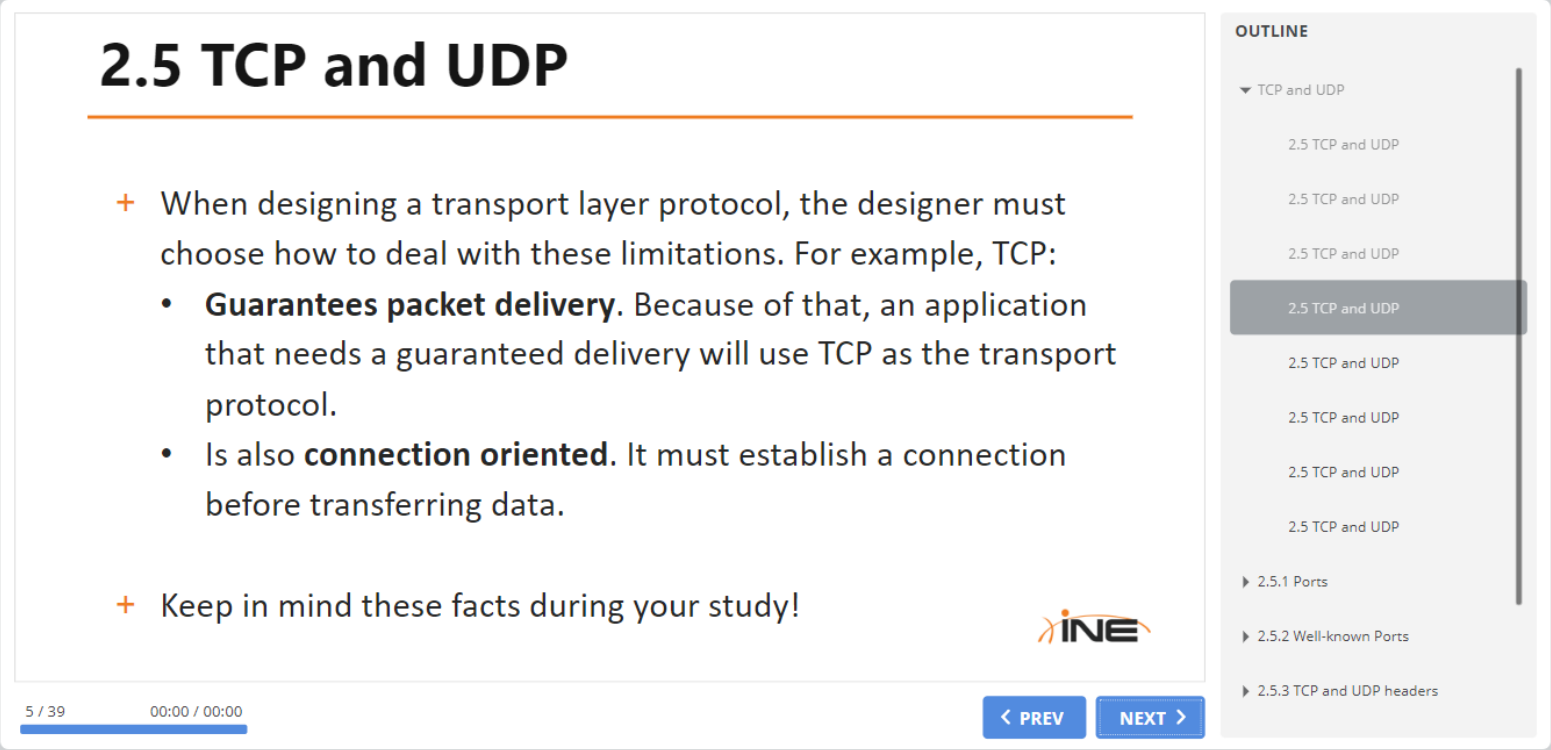
- In this section, you will see how the **transport layer** works, and how the application layer uses its services to identify server and client processes.
- The **Transmission Control Protocol** (TCP) and the **User Datagram Protocol** (UDP) are the most common transport protocols used on the Internet.



2.5 TCP and UDP

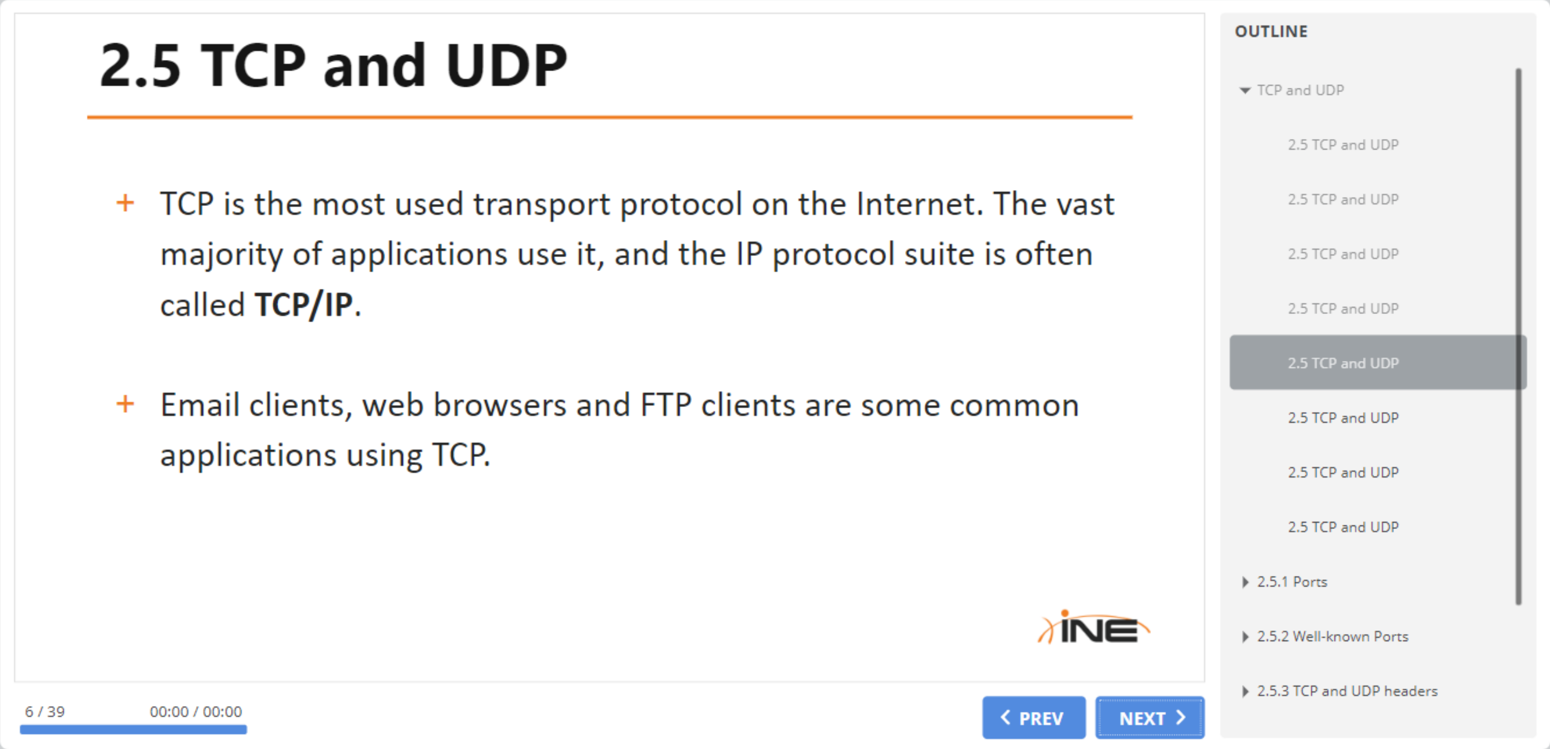
- Before checking out the different services that a transport layer protocol can offer to the application layer, let's consider something important about networks.

- Computer networks can be **unreliable**. This means that some **packets can be lost** during their trip from source to destination. A packet can be lost because of network congestion, temporary loss of connection and other technical issues.



2.5 TCP and UDP

- When designing a transport layer protocol, the designer must choose how to deal with these limitations. For example, TCP:
  - **Guarantees packet delivery**. Because of that, an application that needs a guaranteed delivery will use TCP as the transport protocol.
  - Is also **connection oriented**. It must establish a connection before transferring data.
- Keep in mind these facts during your study!

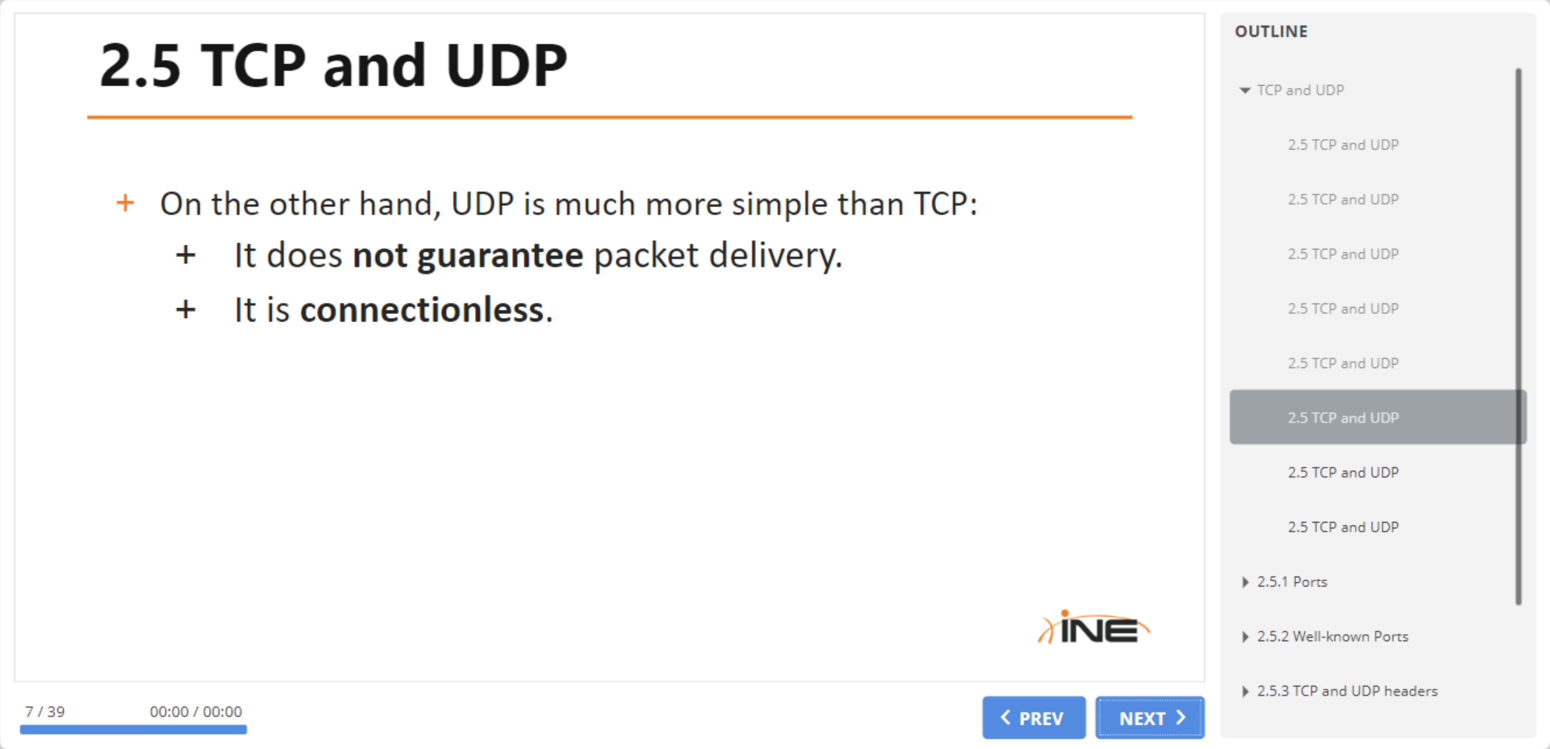


2.5 TCP and UDP

- TCP is the most used transport protocol on the Internet. The vast majority of applications use it, and the IP protocol suite is often called **TCP/IP**.
- Email clients, web browsers and FTP clients are some common applications using TCP.

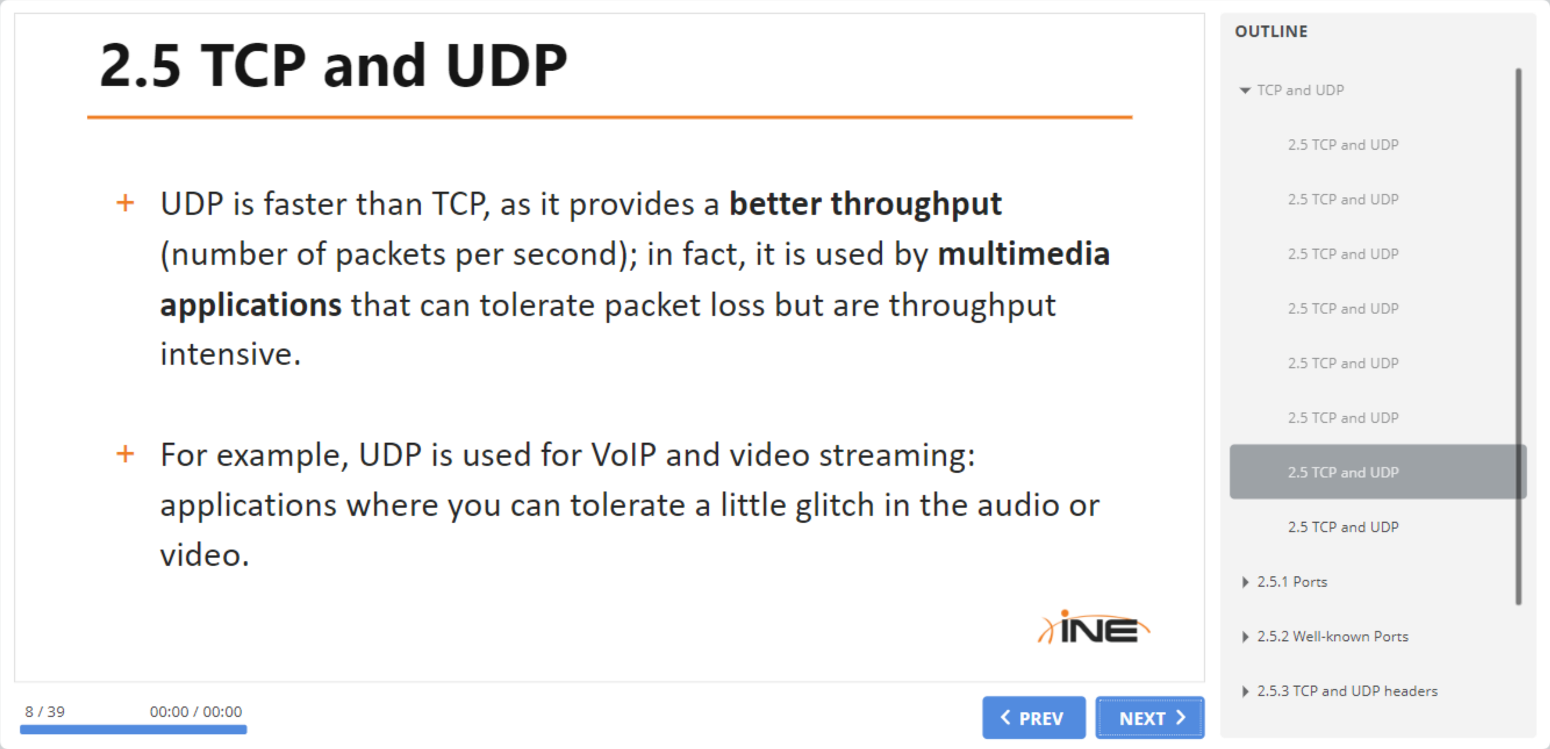
OUTLINE

- TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
  - 2.5 TCP and UDP
- 2.5.1 Ports
- 2.5.2 Well-known Ports
- 2.5.3 TCP and UDP headers



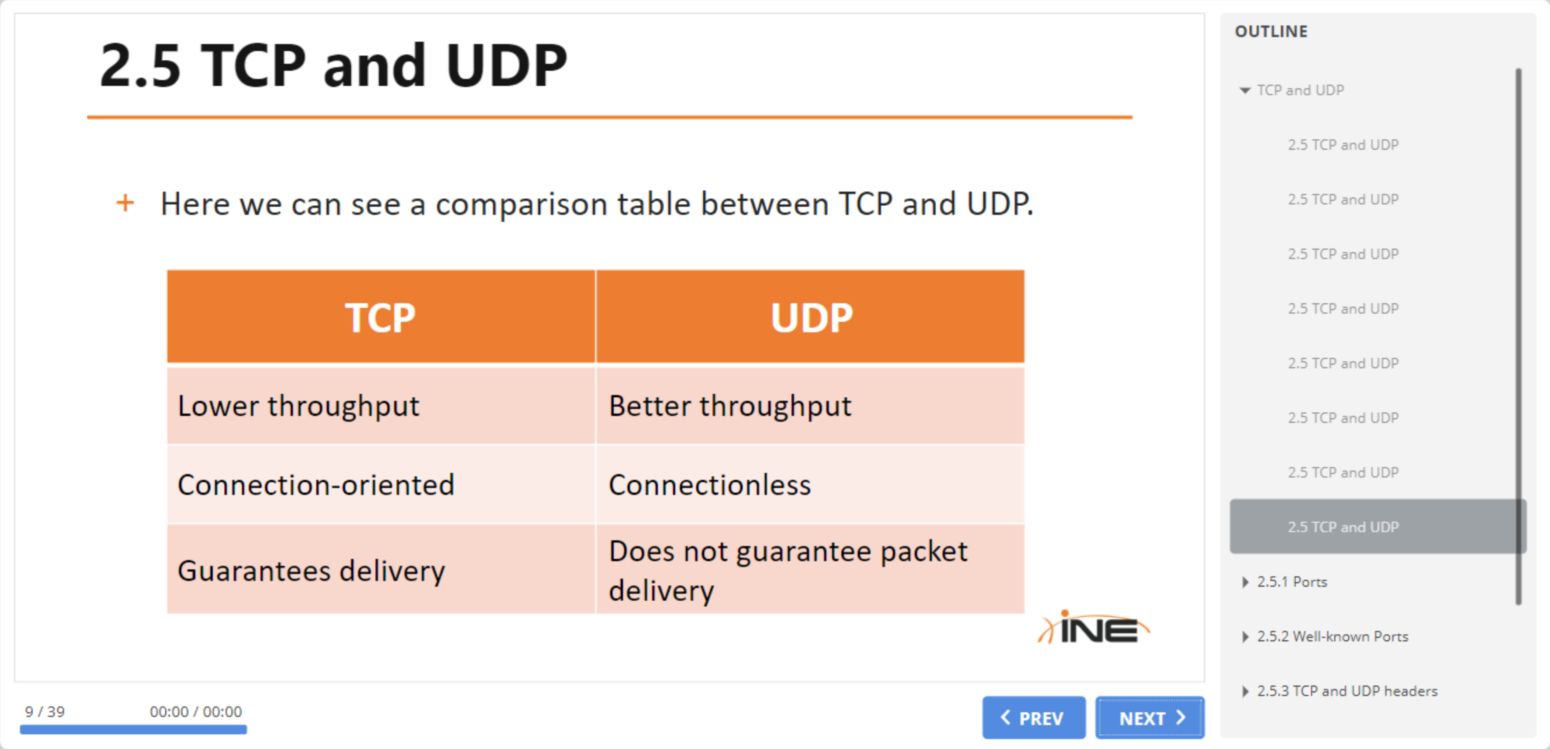
2.5 TCP and UDP

- On the other hand, UDP is much more simple than TCP:
  - It does **not guarantee** packet delivery.
  - It is **connectionless**.



2.5 TCP and UDP

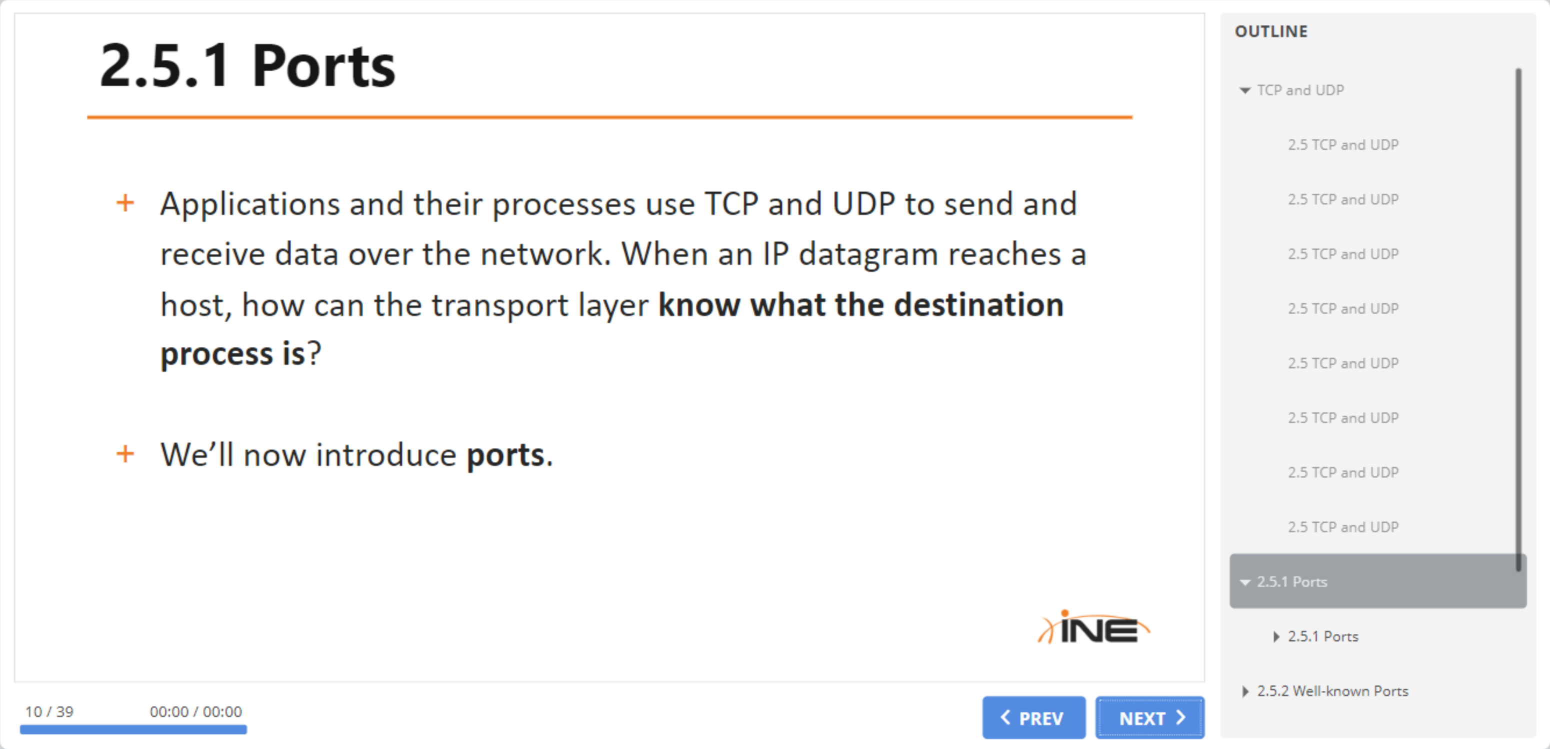
- UDP is faster than TCP, as it provides a **better throughput** (number of packets per second); in fact, it is used by **multimedia applications** that can tolerate packet loss but are throughput intensive.
- For example, UDP is used for VoIP and video streaming: applications where you can tolerate a little glitch in the audio or video.



2.5 TCP and UDP

- Here we can see a comparison table between TCP and UDP.

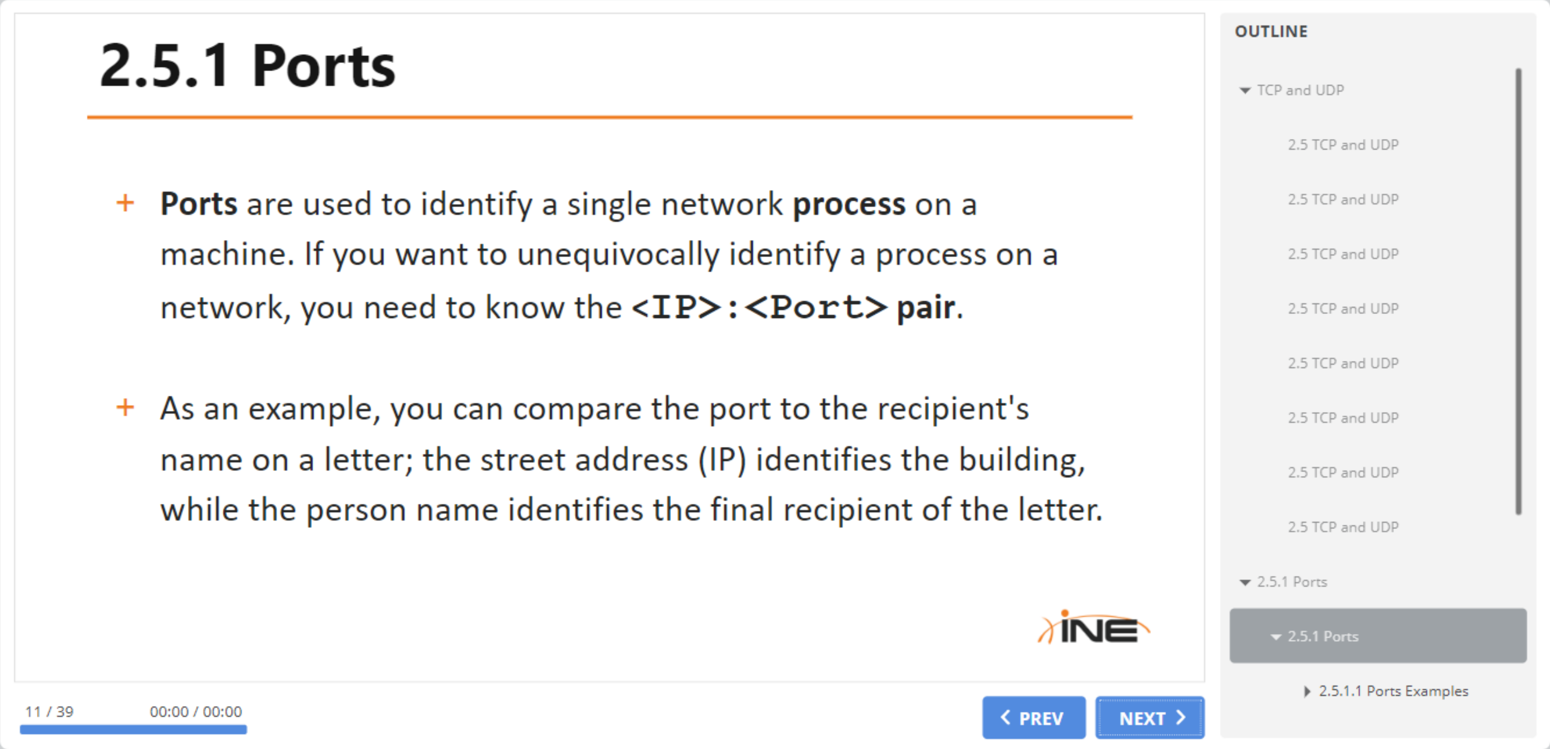
TCP	UDP
Lower throughput	Better throughput
Connection-oriented	Connectionless
Guarantees delivery	Does not guarantee packet delivery



2.5.1 Ports

- Applications and their processes use TCP and UDP to send and receive data over the network. When an IP datagram reaches a host, how can the transport layer **know what the destination process is**?

- We'll now introduce **ports**.



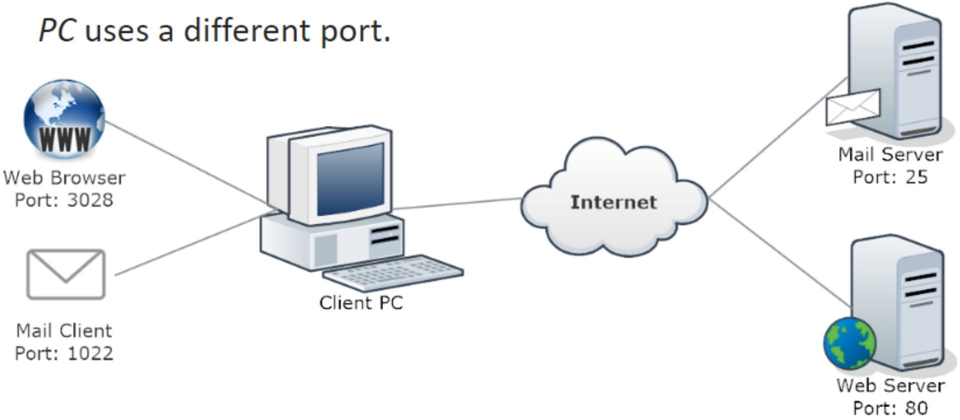
2.5.1 Ports

- **Ports** are used to identify a single network **process** on a machine. If you want to unequivocally identify a process on a network, you need to know the **`<IP>:<Port>` pair**.

- As an example, you can compare the port to the recipient's name on a letter; the street address (IP) identifies the building, while the person name identifies the final recipient of the letter.

2.5.1.1 Ports Examples

+ In this image, you can see how every client application on *Client PC* uses a different port.



INE

12 / 39 00:00 / 00:00

< PREV NEXT >

OUTLINE

▾ TCP and UDP

2.5 TCP and UDP

2.5 TCP and UDP

2.5 TCP and UDP

2.5 TCP and UDP

2.5 TCP and UDP

2.5 TCP and UDP

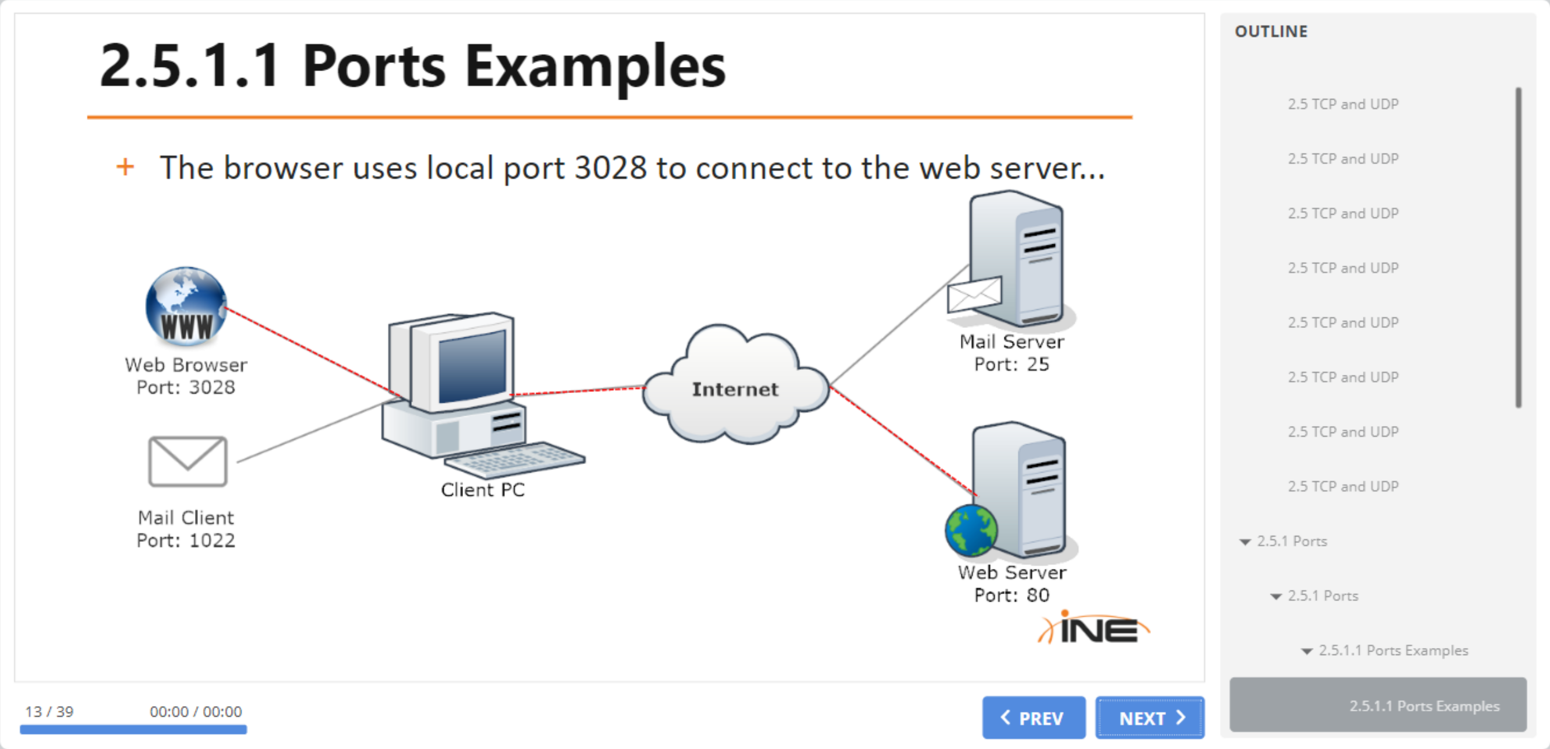
2.5 TCP and UDP

2.5 TCP and UDP

▾ 2.5.1 Ports

▾ 2.5.1 Ports

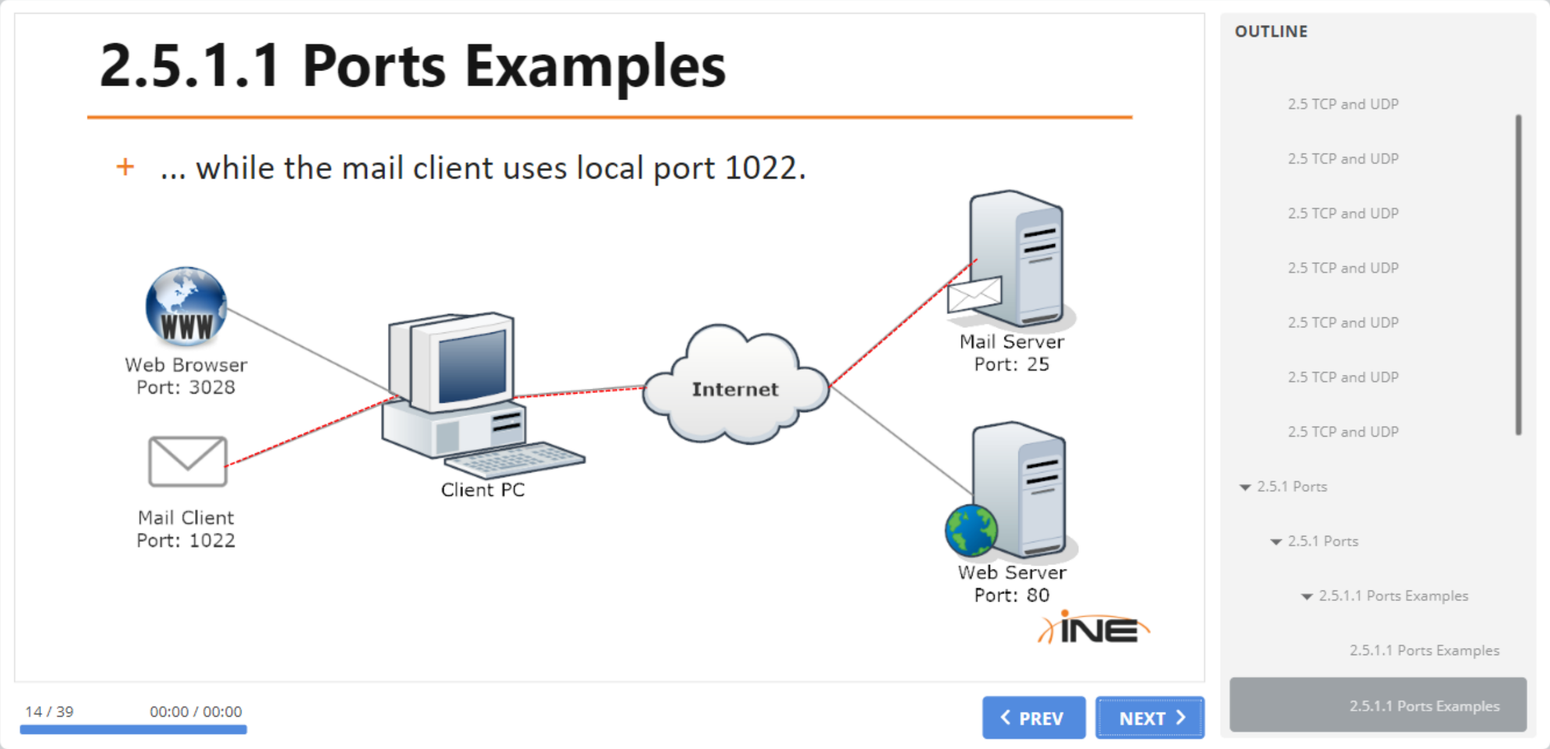
▾ 2.5.1.1 Ports Examples



2.5.1.1 Ports Examples

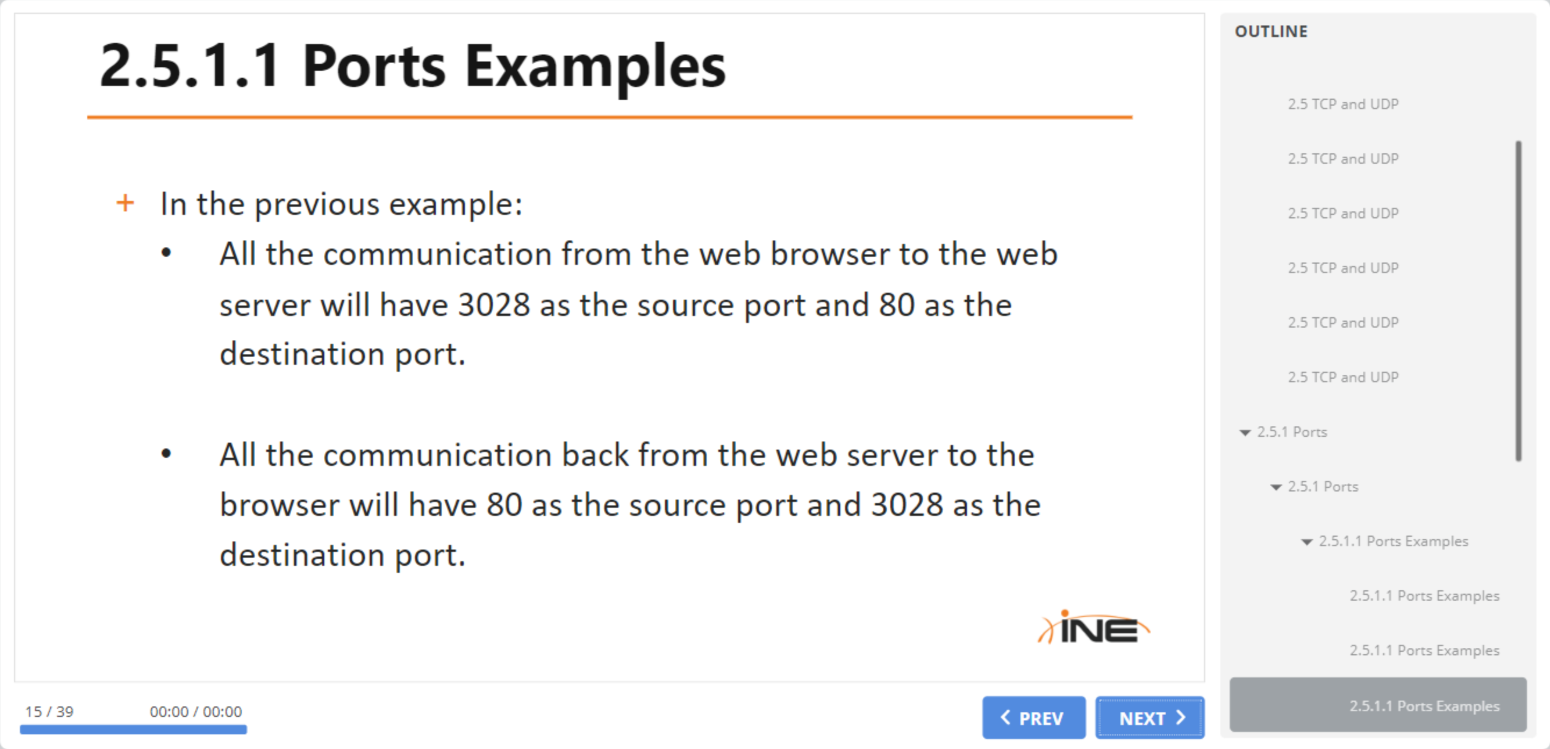
- The browser uses local port 3028 to connect to the web server...

OUTLINE



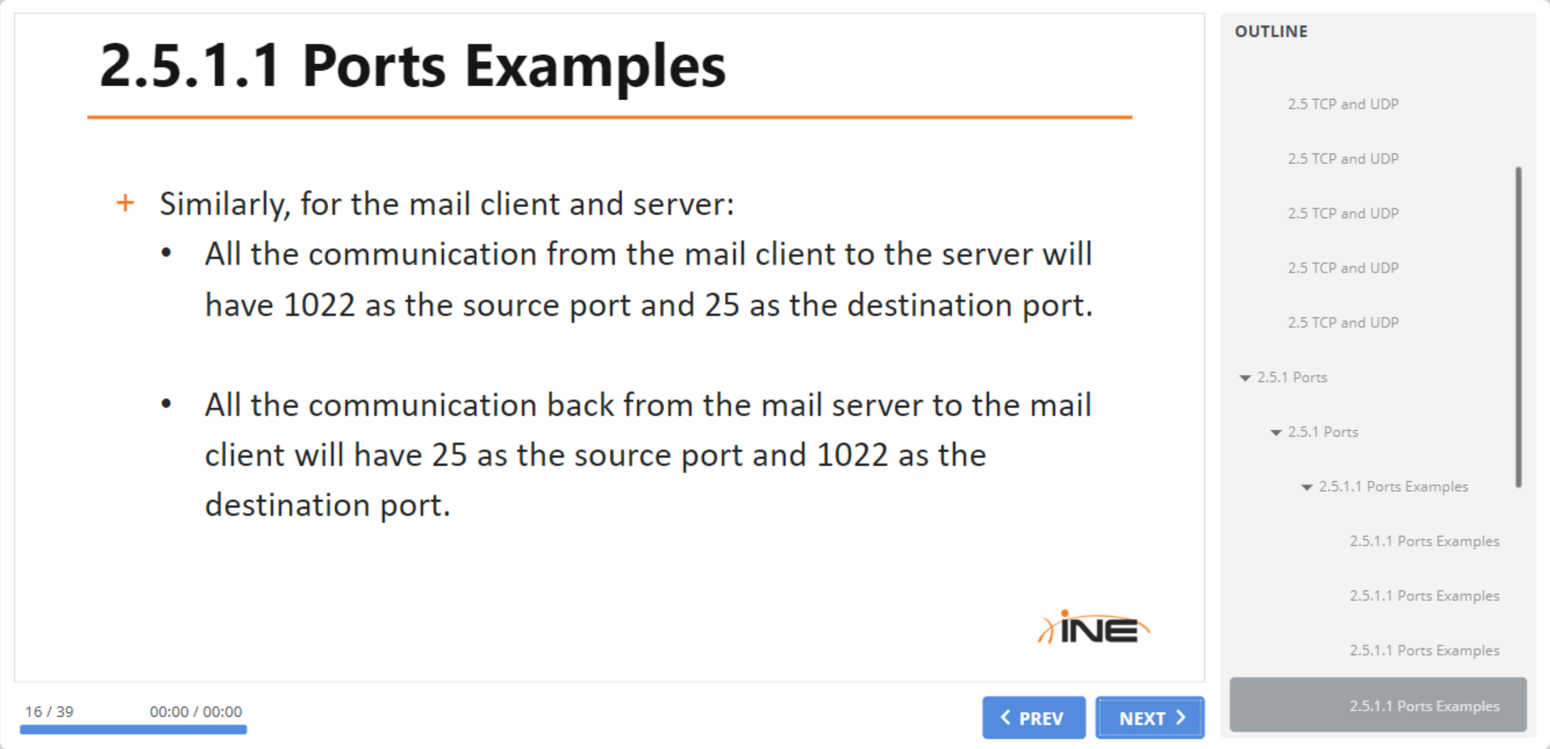
2.5.1.1 Ports Examples

- + ... while the mail client uses local port 1022.



2.5.1.1 Ports Examples

- In the previous example:
  - All the communication from the web browser to the web server will have 3028 as the source port and 80 as the destination port.
  - All the communication back from the web server to the browser will have 80 as the source port and 3028 as the destination port.

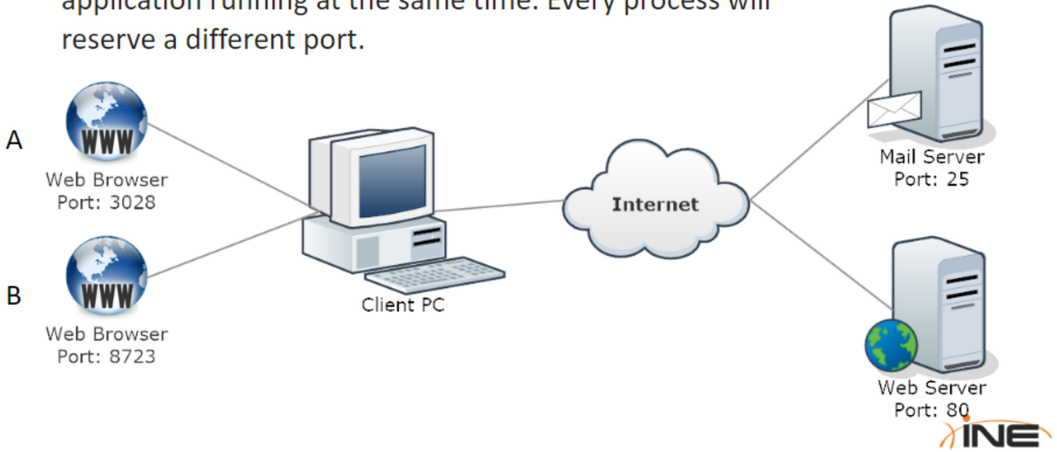


2.5.1.1 Ports Examples

- Similarly, for the mail client and server:
  - All the communication from the mail client to the server will have 1022 as the source port and 25 as the destination port.
  - All the communication back from the mail server to the mail client will have 25 as the source port and 1022 as the destination port.

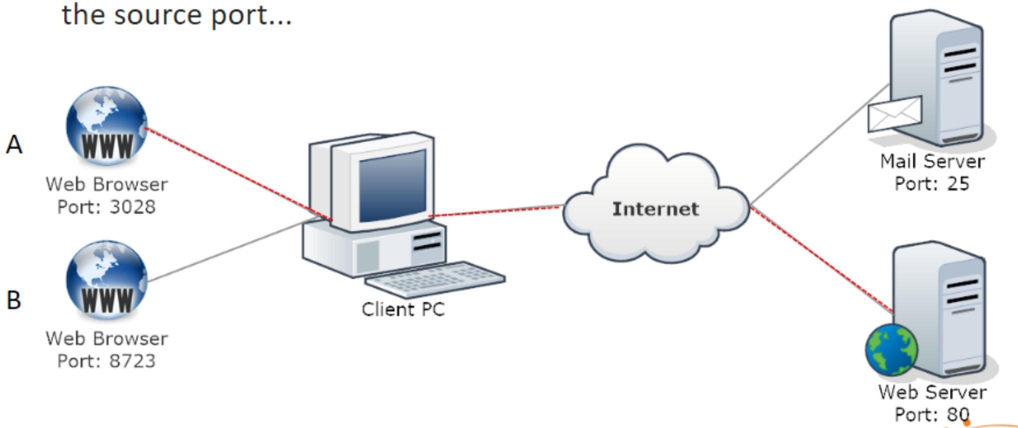
2.5.1.1 Ports Examples

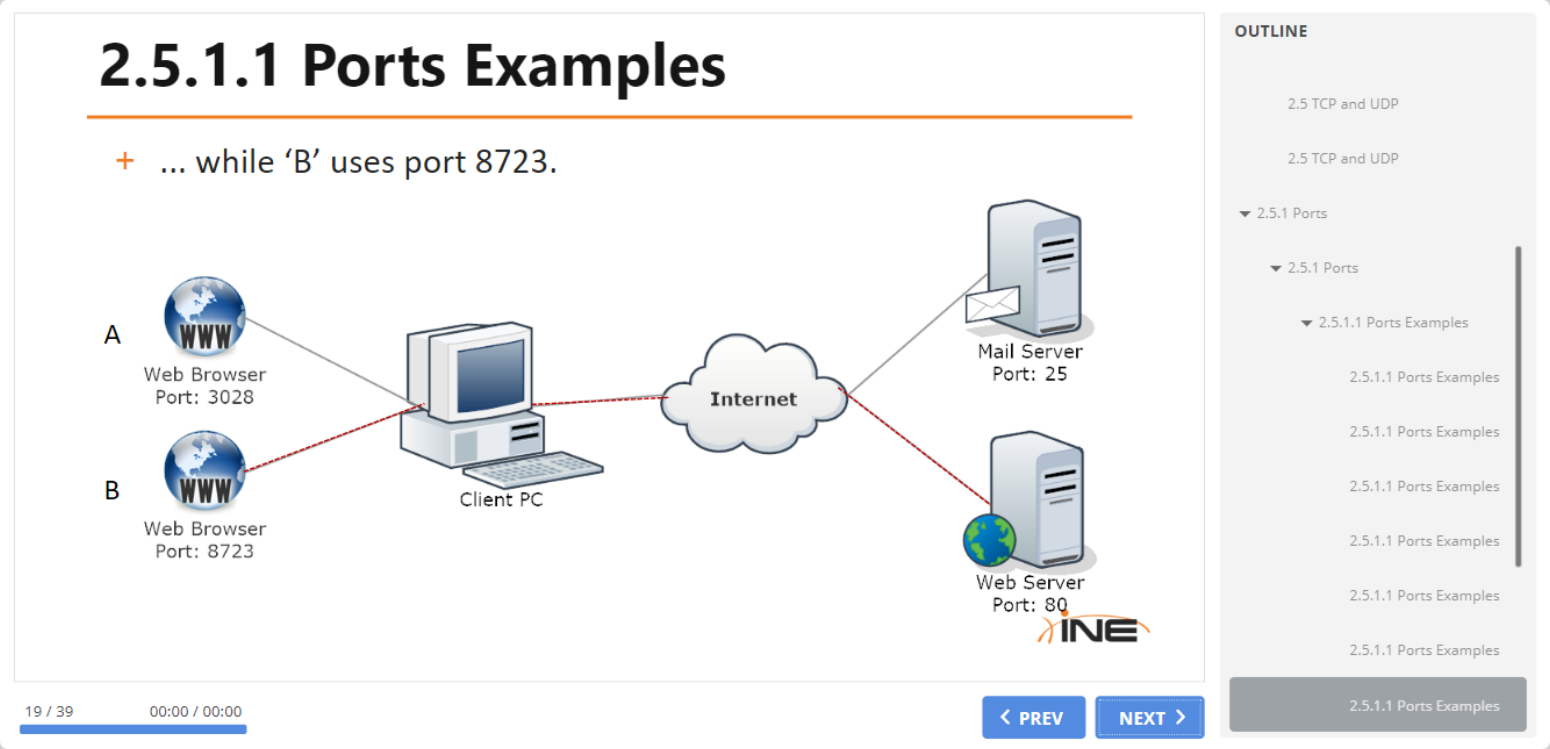
- Furthermore, you may also have multiple instances of the same application running at the same time. Every process will reserve a different port.



2.5.1.1 Ports Examples

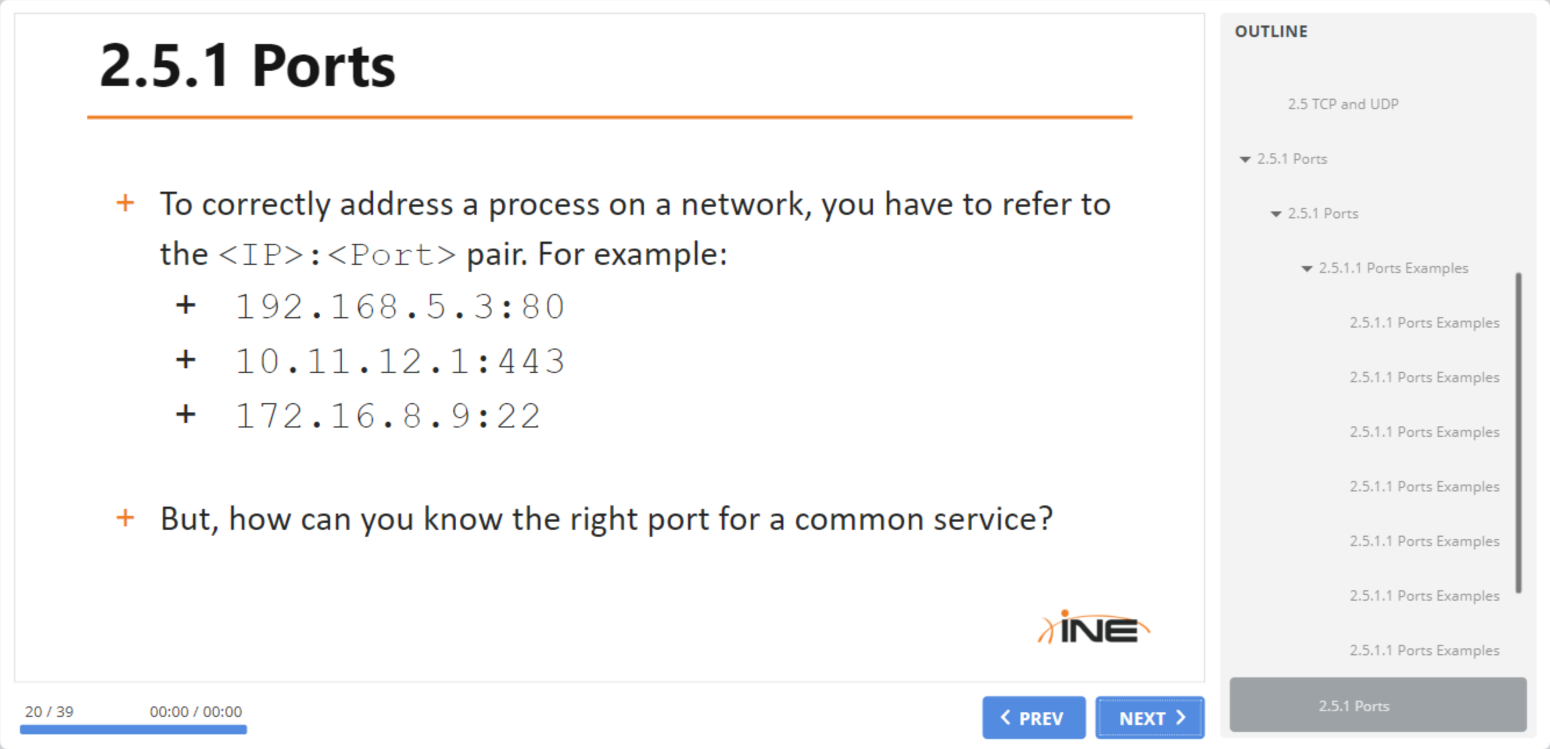
- In this example, ‘A’ communicates with the web server using 3028 as the source port...





2.5.1.1 Ports Examples

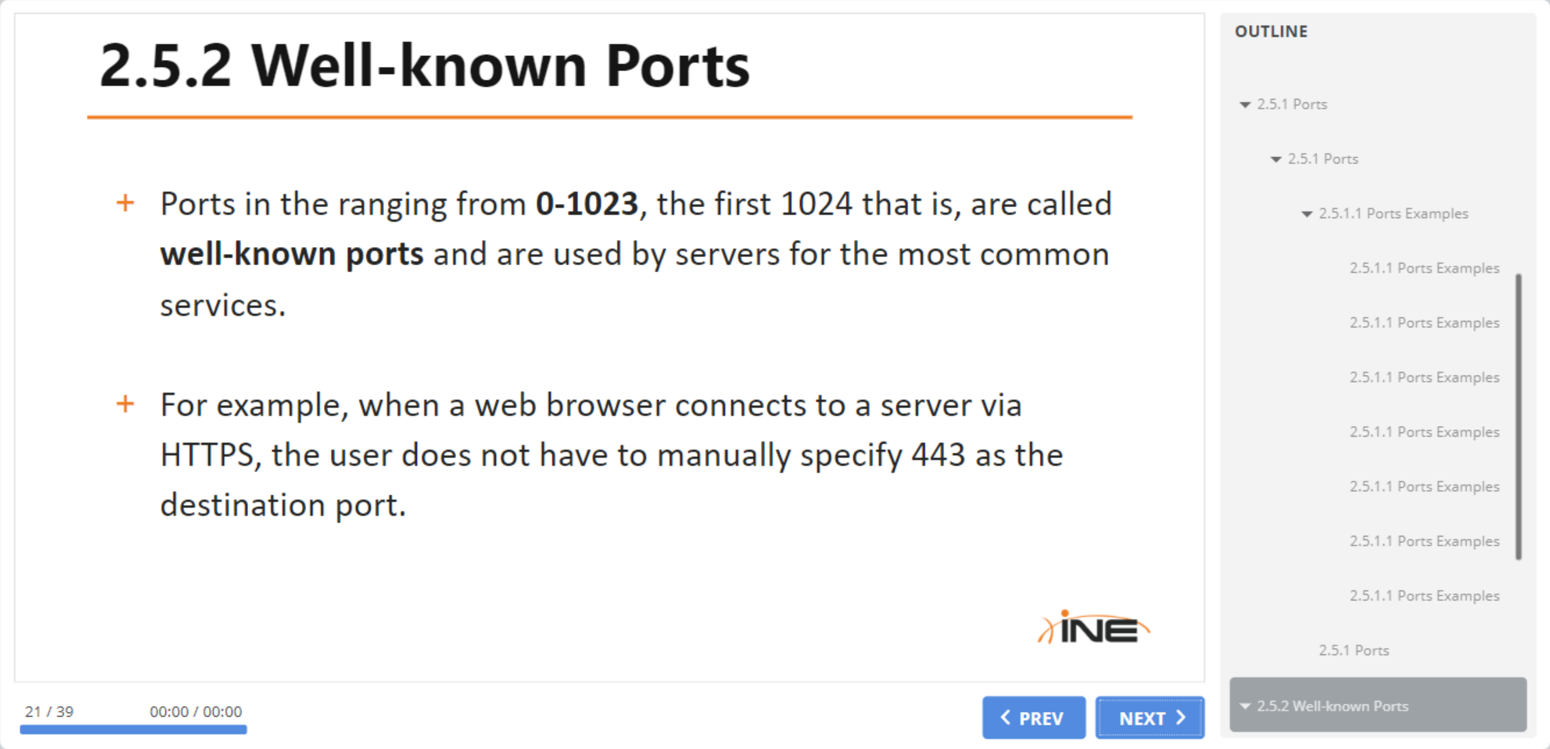
- ... while ‘B’ uses port 8723.



2.5.1 Ports

- To correctly address a process on a network, you have to refer to the `<IP>:<Port>` pair. For example:
  - `192.168.5.3:80`
  - `10.11.12.1:443`
  - `172.16.8.9:22`

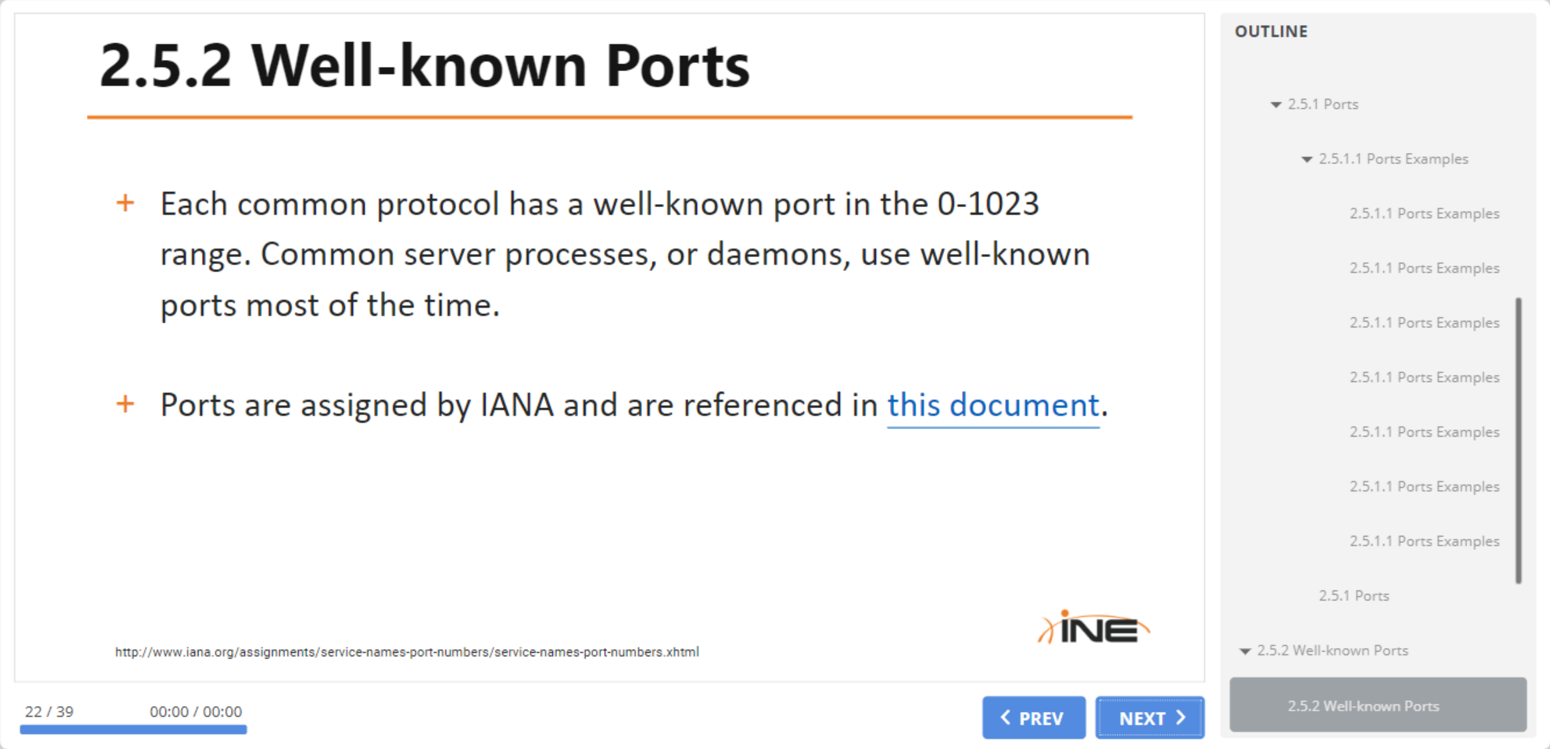
- But, how can you know the right port for a common service?



2.5.2 Well-known Ports

- Ports in the ranging from **0-1023**, the first 1024 that is, are called **well-known ports** and are used by servers for the most common services.

- For example, when a web browser connects to a server via HTTPS, the user does not have to manually specify 443 as the destination port.

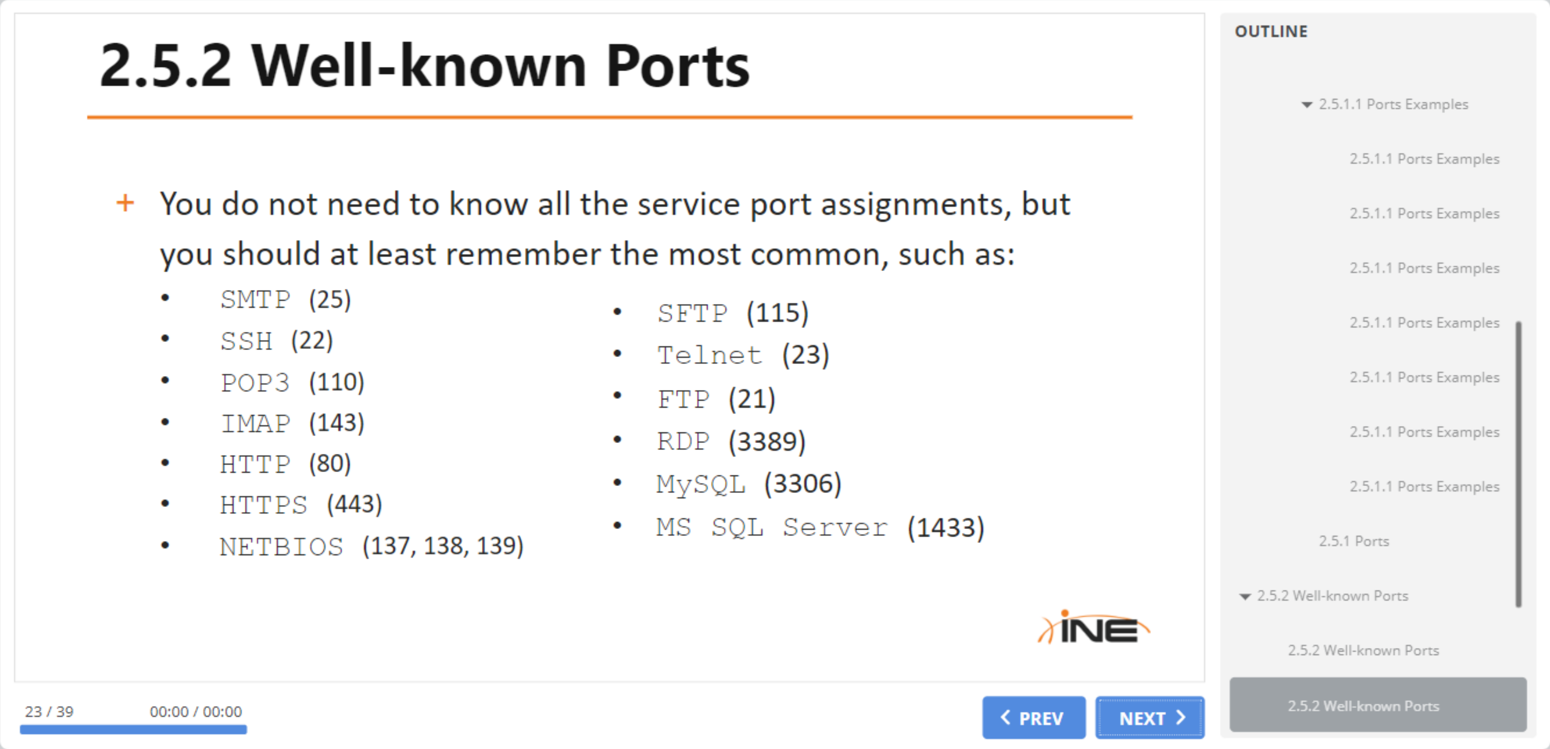


2.5.2 Well-known Ports

- Each common protocol has a well-known port in the 0-1023 range. Common server processes, or daemons, use well-known ports most of the time.

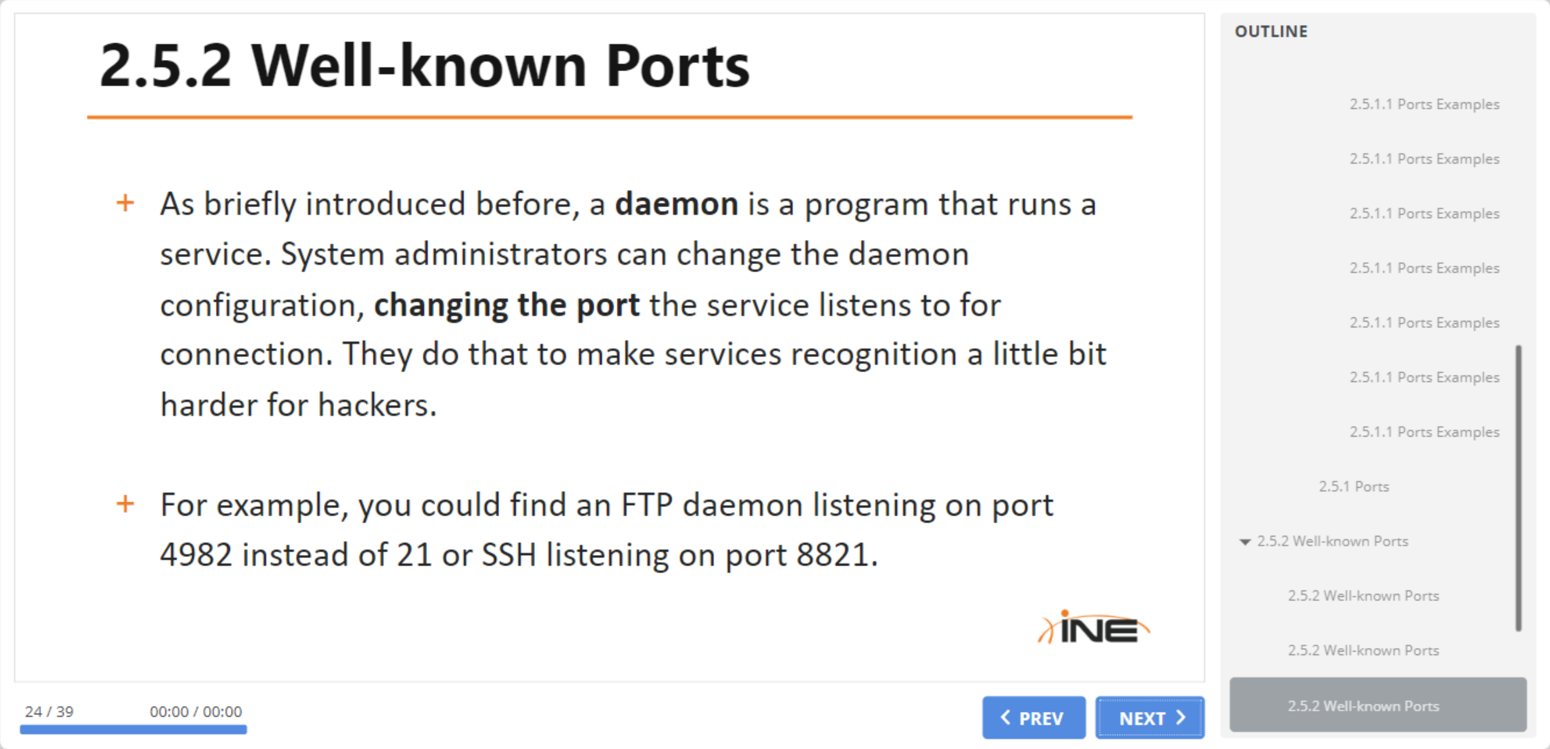
- Ports are assigned by IANA and are referenced in this document.

http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml



2.5.2 Well-known Ports

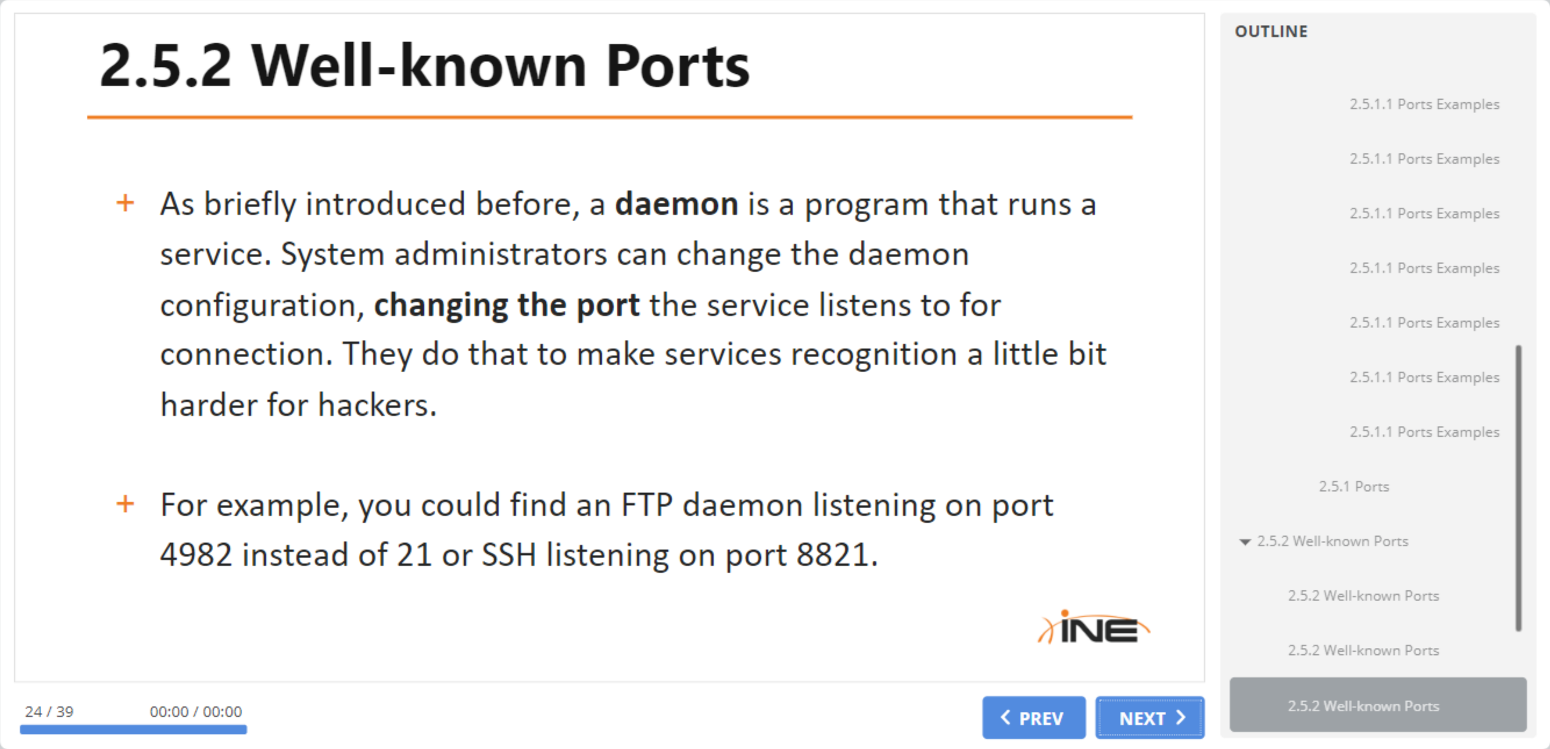
+ You do not need to know all the service port assignments, but you should at least remember the most common, such as:
  - SMTP (25)
  - SSH (22)
  - POP3 (110)
  - IMAP (143)
  - HTTP (80)
  - HTTPS (443)
  - NETBIOS (137, 138, 139)
  - SFTP (115)
  - Telnet (23)
  - FTP (21)
  - RDP (3389)
  - MySQL (3306)
  - MS SQL Server (1433)



2.5.2 Well-known Ports

- As briefly introduced before, a **daemon** is a program that runs a service. System administrators can change the daemon configuration, **changing the port** the service listens to for connection. They do that to make services recognition a little bit harder for hackers.

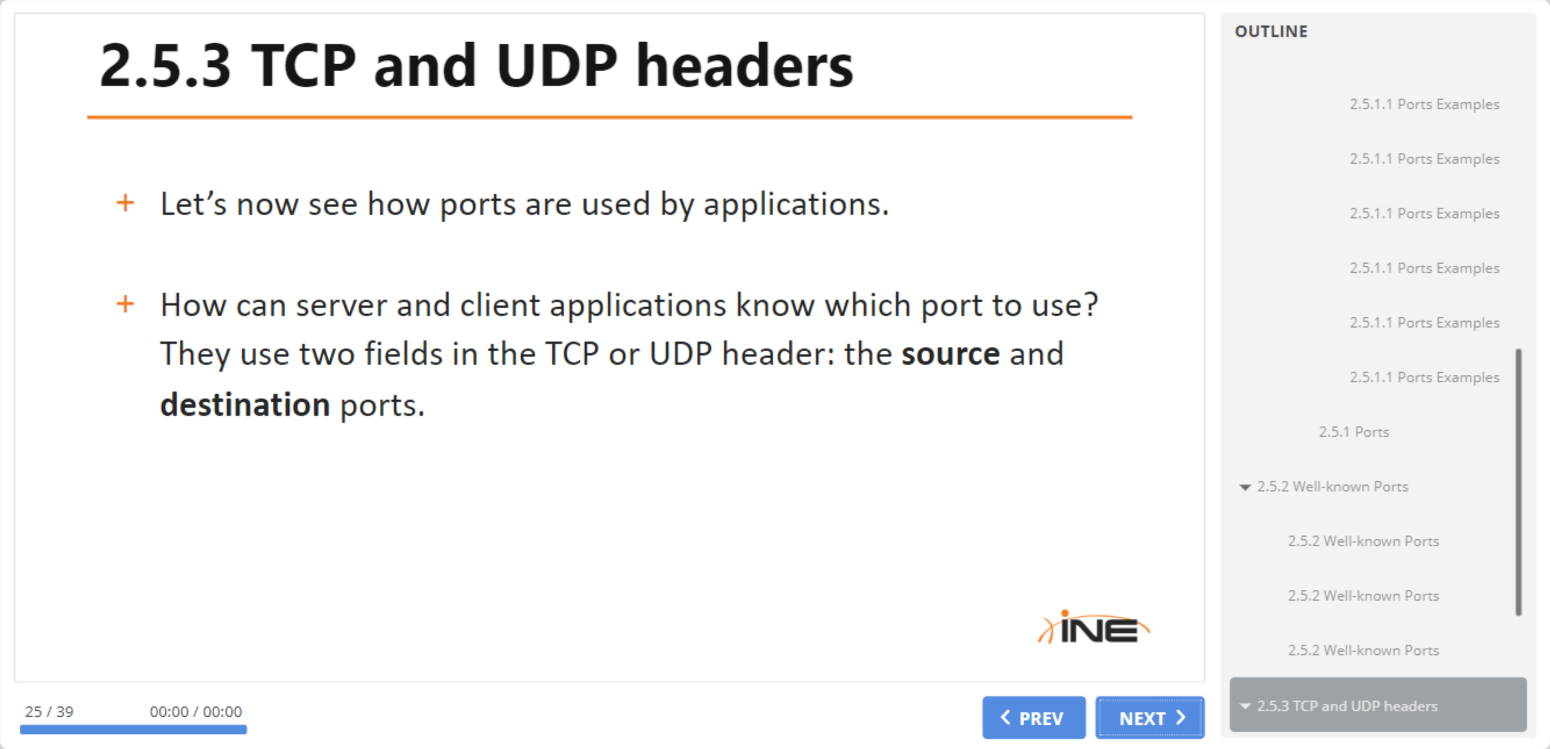
- For example, you could find an FTP daemon listening on port 4982 instead of 21 or SSH listening on port 8821.



2.5.2 Well-known Ports

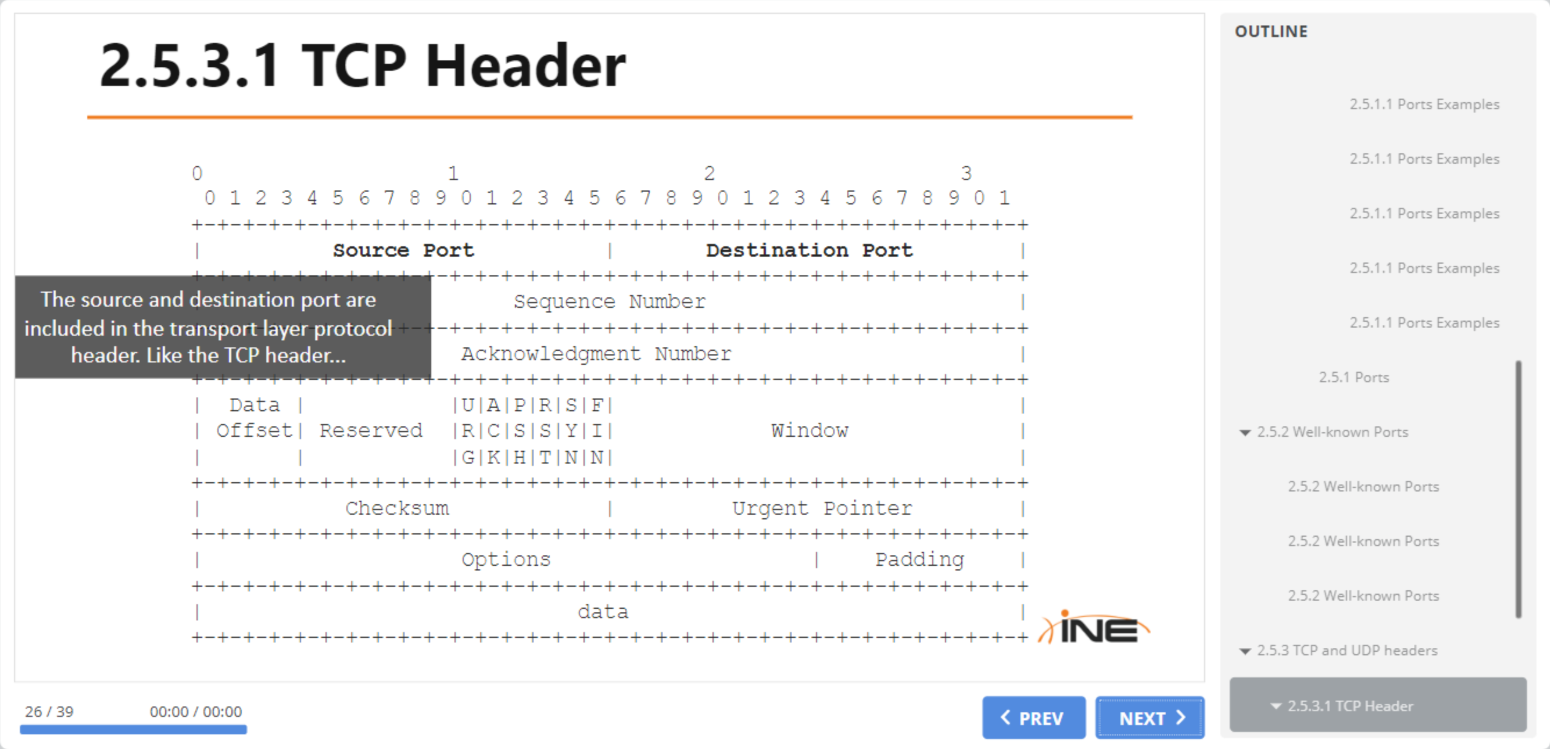
- As briefly introduced before, a **daemon** is a program that runs a service. System administrators can change the daemon configuration, **changing the port** the service listens to for connection. They do that to make services recognition a little bit harder for hackers.

- For example, you could find an FTP daemon listening on port 4982 instead of 21 or SSH listening on port 8821.



2.5.3 TCP and UDP headers

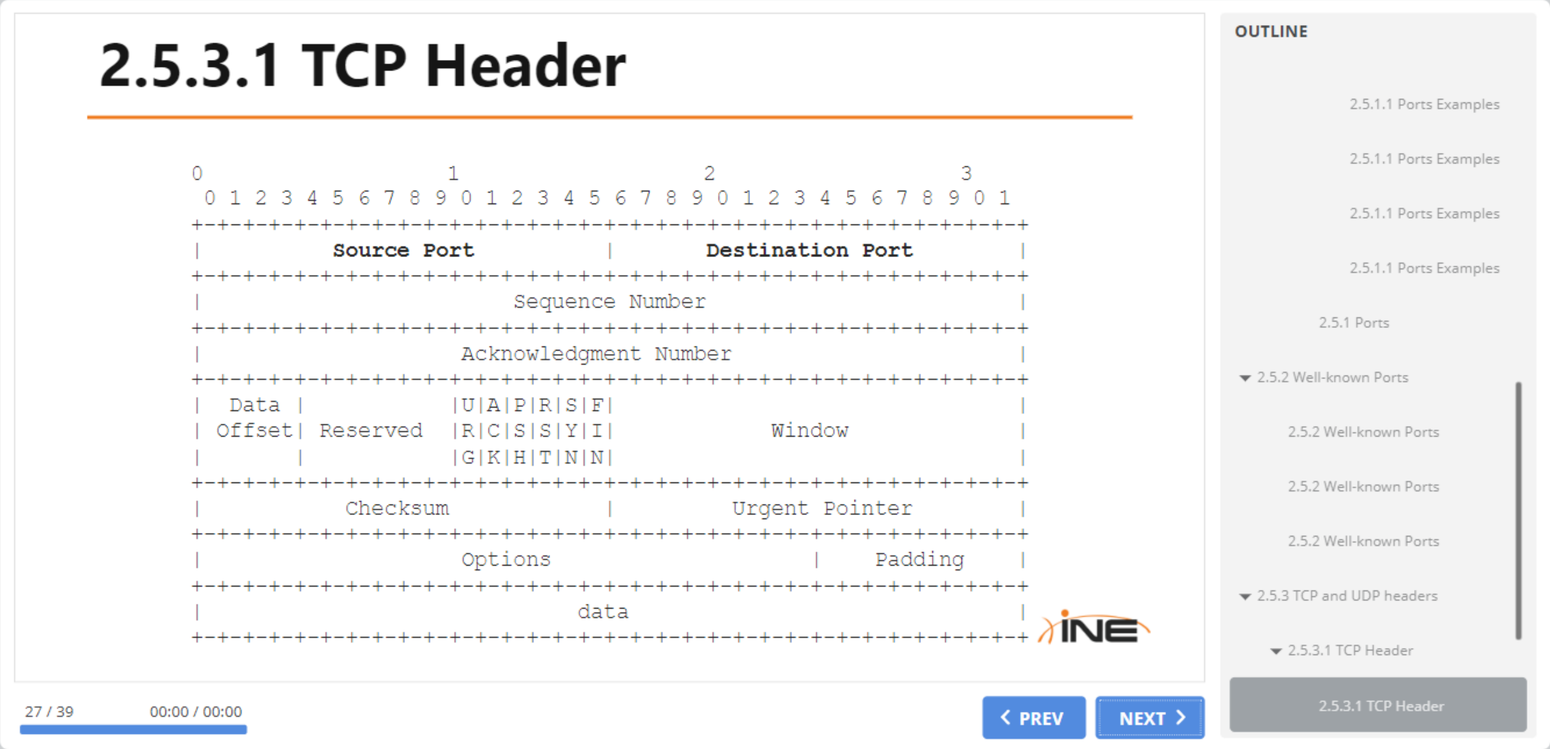
- Let’s now see how ports are used by applications.
- How can server and client applications know which port to use? They use two fields in the TCP or UDP header: the **source** and **destination** ports.



2.5.3.1 TCP Header

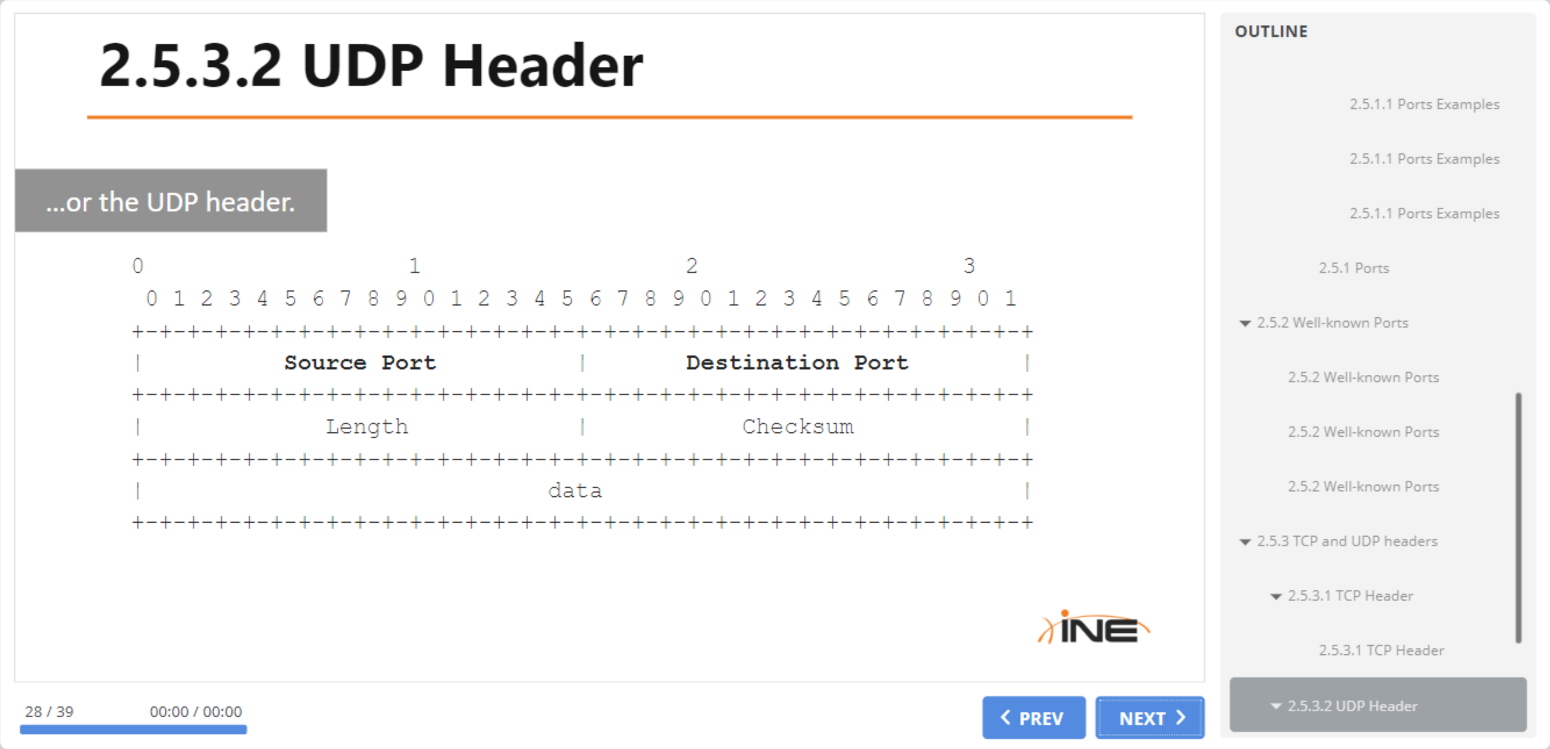
```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                         Sequence Number                        |
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                     Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The source and destination port are included in the transport layer protocol header. Like the TCP header...



2.5.3.1 TCP Header

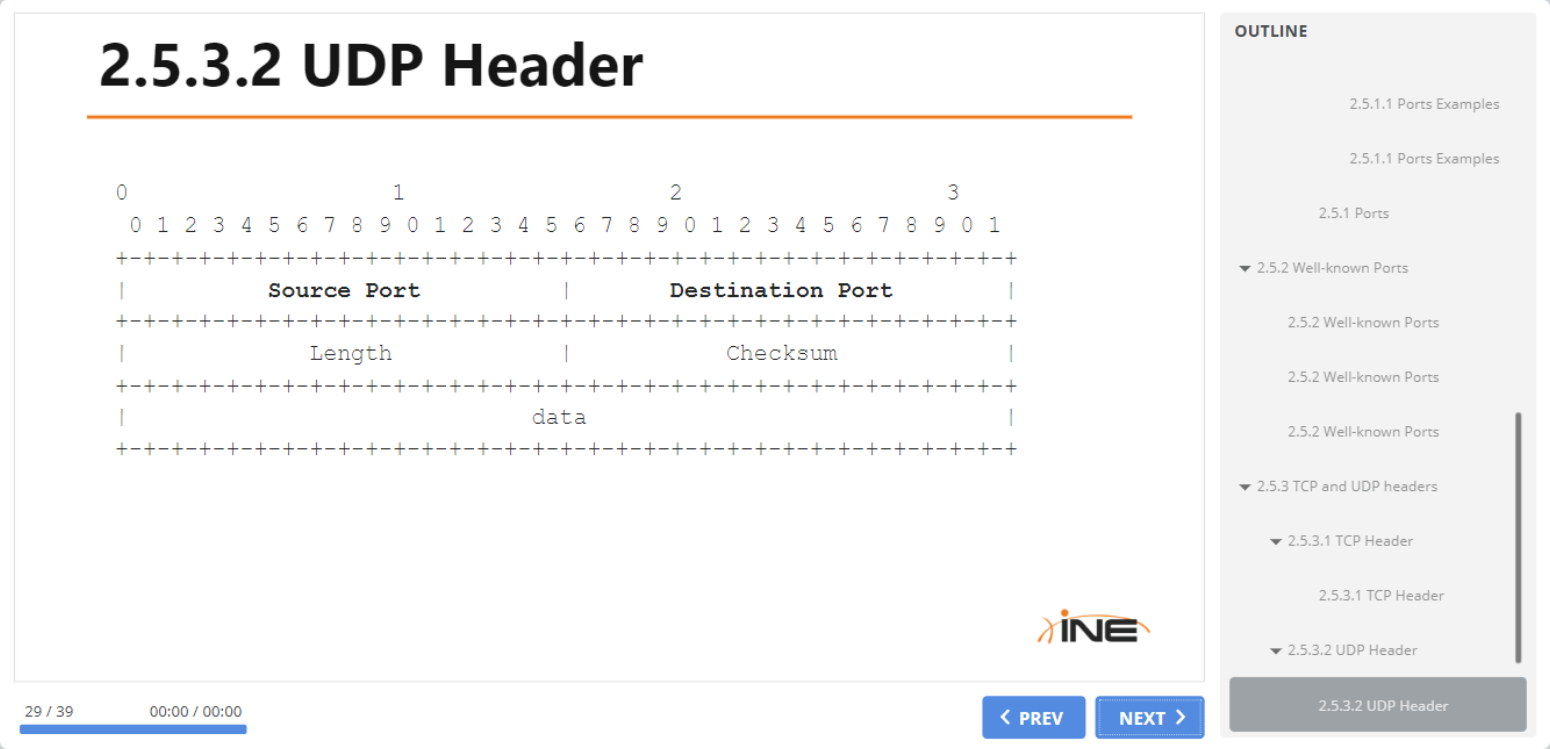
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



2.5.3.2 UDP Header

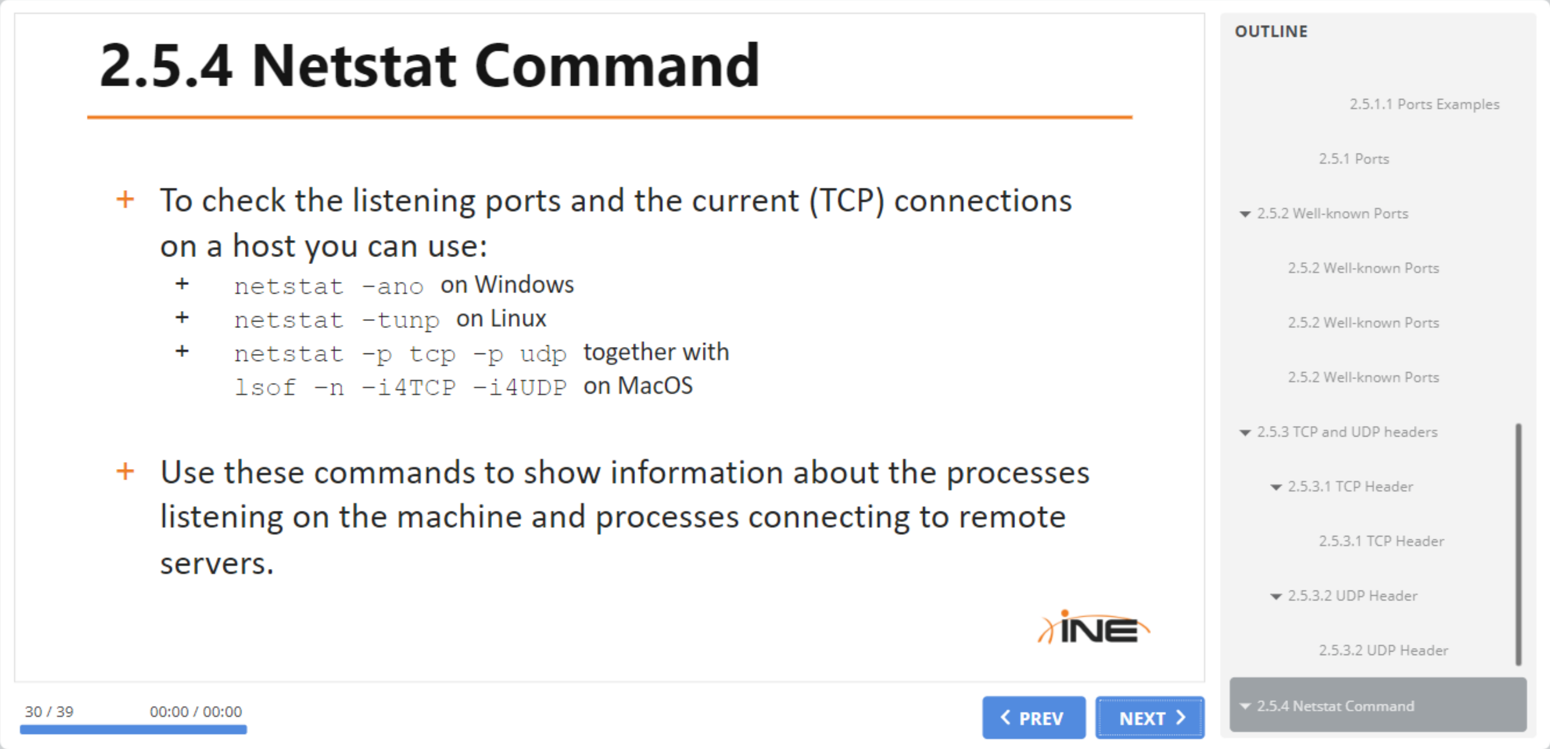
...or the UDP header.

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Source Port          |       Destination Port        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            Length             |           Checksum            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              data                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



2.5.3.2 UDP Header

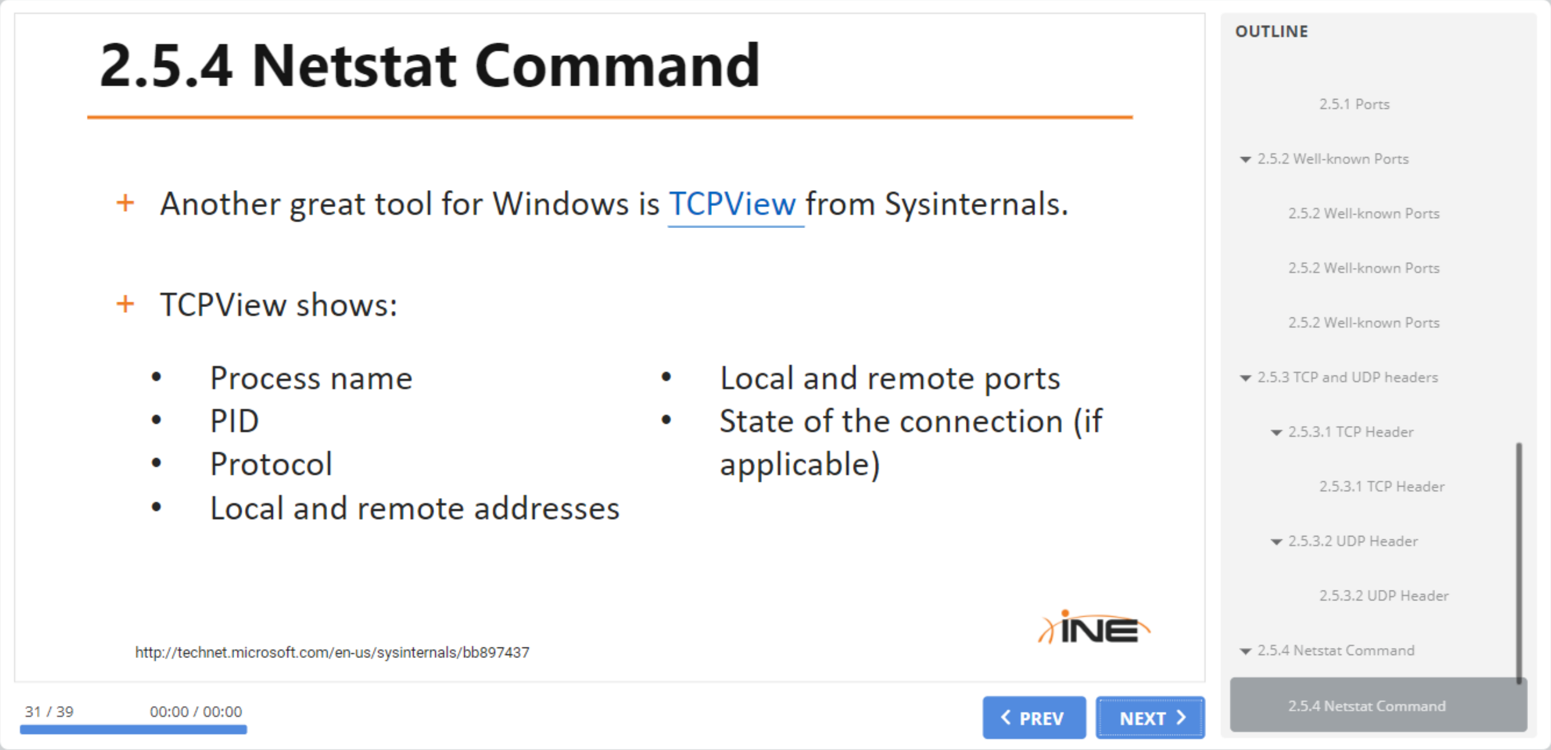
```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Source Port          |       Destination Port        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            Length             |           Checksum            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                             data                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



2.5.4 Netstat Command

- To check the listening ports and the current (TCP) connections on a host you can use:
  - `netstat -ano` on Windows
  - `netstat -tunp` on Linux
  - `netstat -p tcp -p udp` together with `lsof -n -i4TCP -i4UDP` on MacOS

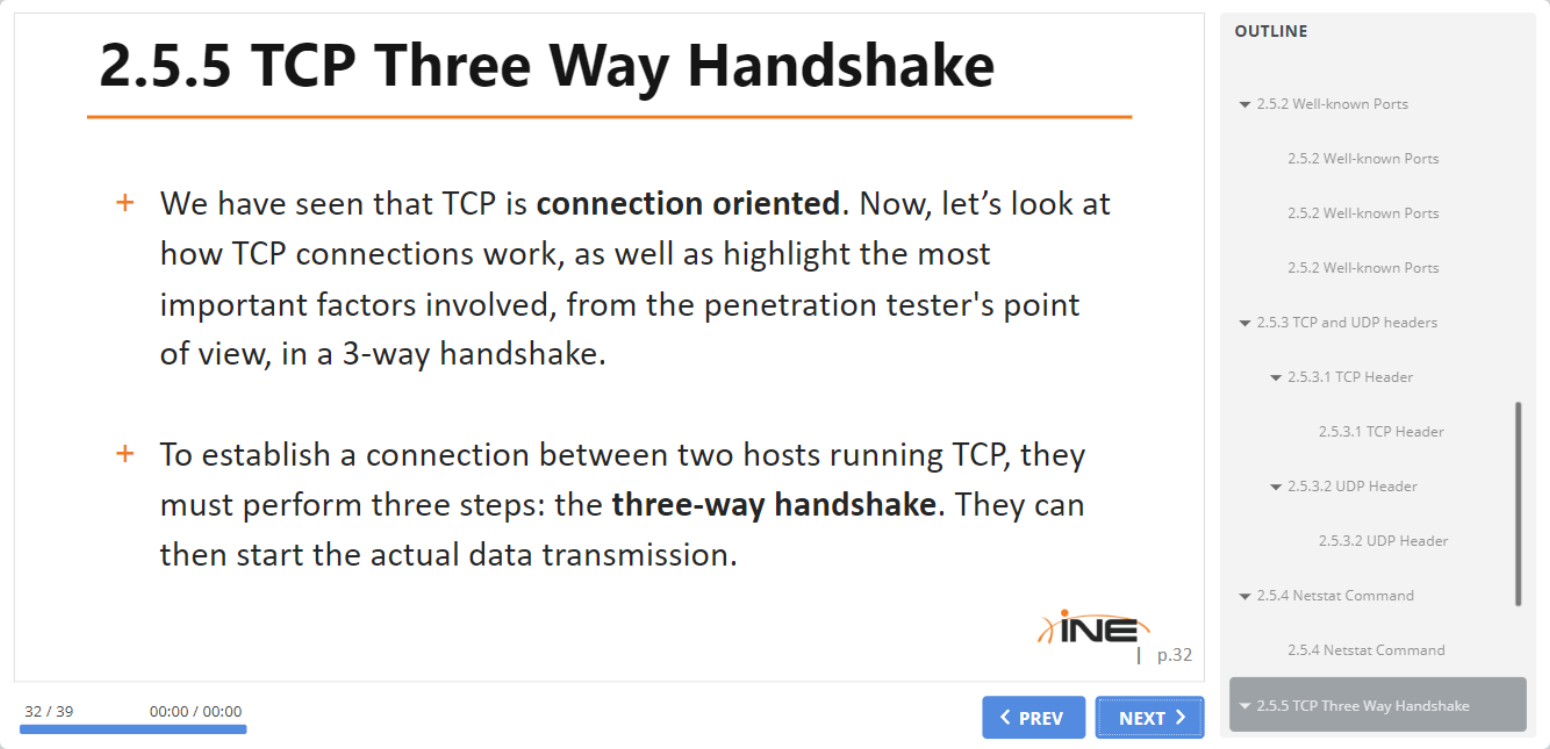
- Use these commands to show information about the processes listening on the machine and processes connecting to remote servers.



2.5.4 Netstat Command

- Another great tool for Windows is TCPView from Sysinternals.
- TCPView shows:
  - Process name
  - PID
  - Protocol
  - Local and remote addresses
  - Local and remote ports
  - State of the connection (if applicable)

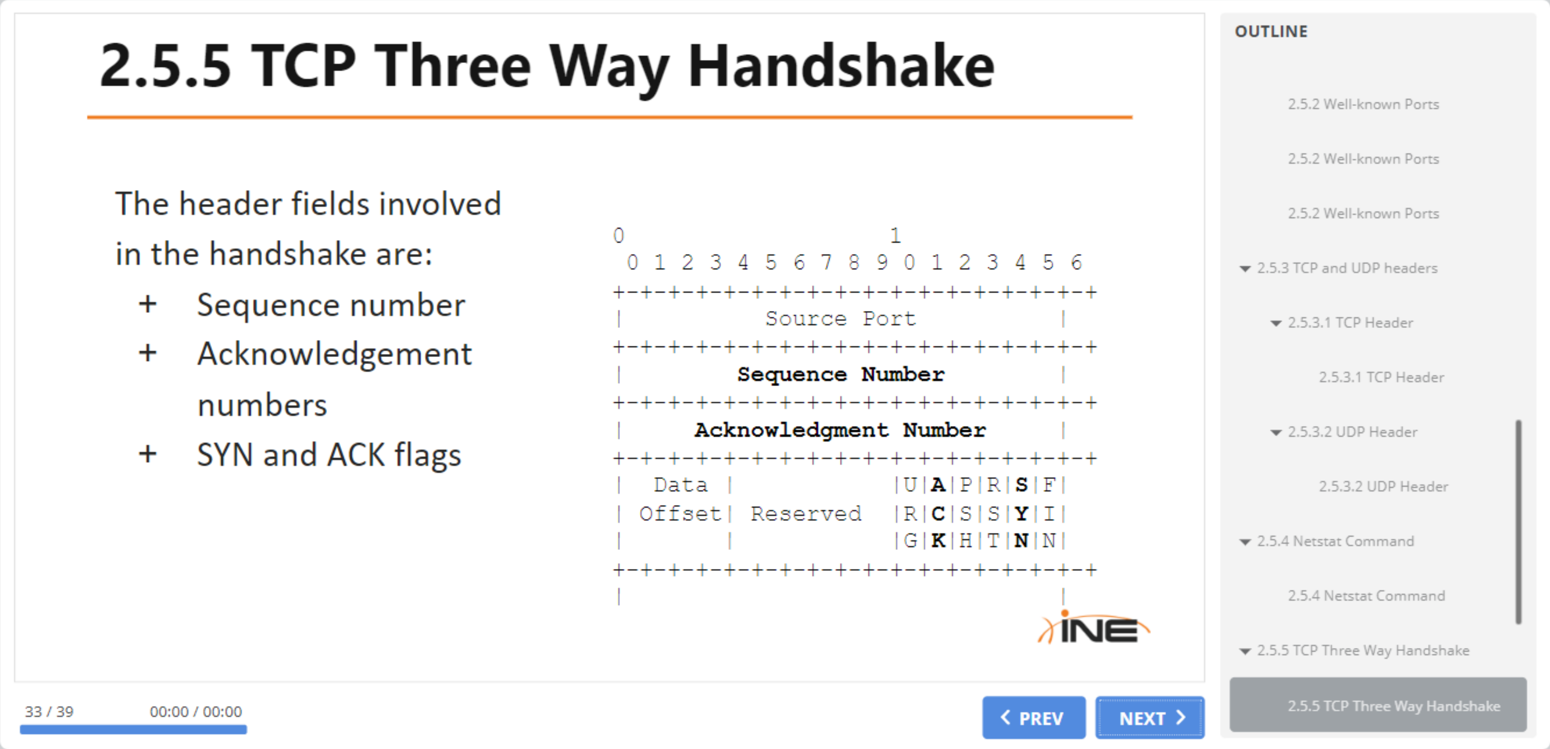
http://technet.microsoft.com/en-us/sysinternals/bb897437



2.5.5 TCP Three Way Handshake

- We have seen that TCP is **connection oriented**. Now, let’s look at how TCP connections work, as well as highlight the most important factors involved, from the penetration tester's point of view, in a 3-way handshake.

- To establish a connection between two hosts running TCP, they must perform three steps: the **three-way handshake**. They can then start the actual data transmission.



2.5.5 TCP Three Way Handshake

The header fields involved in the handshake are:

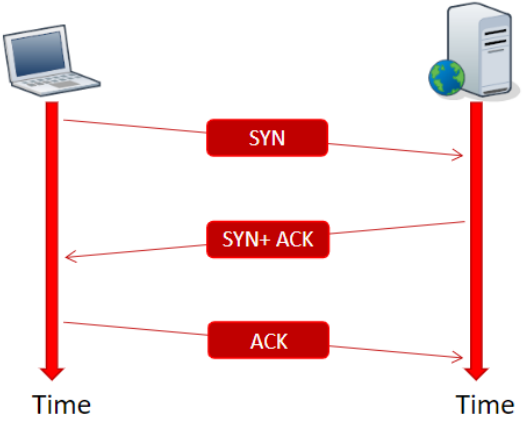
- Sequence number
- Acknowledgement numbers
- SYN and ACK flags

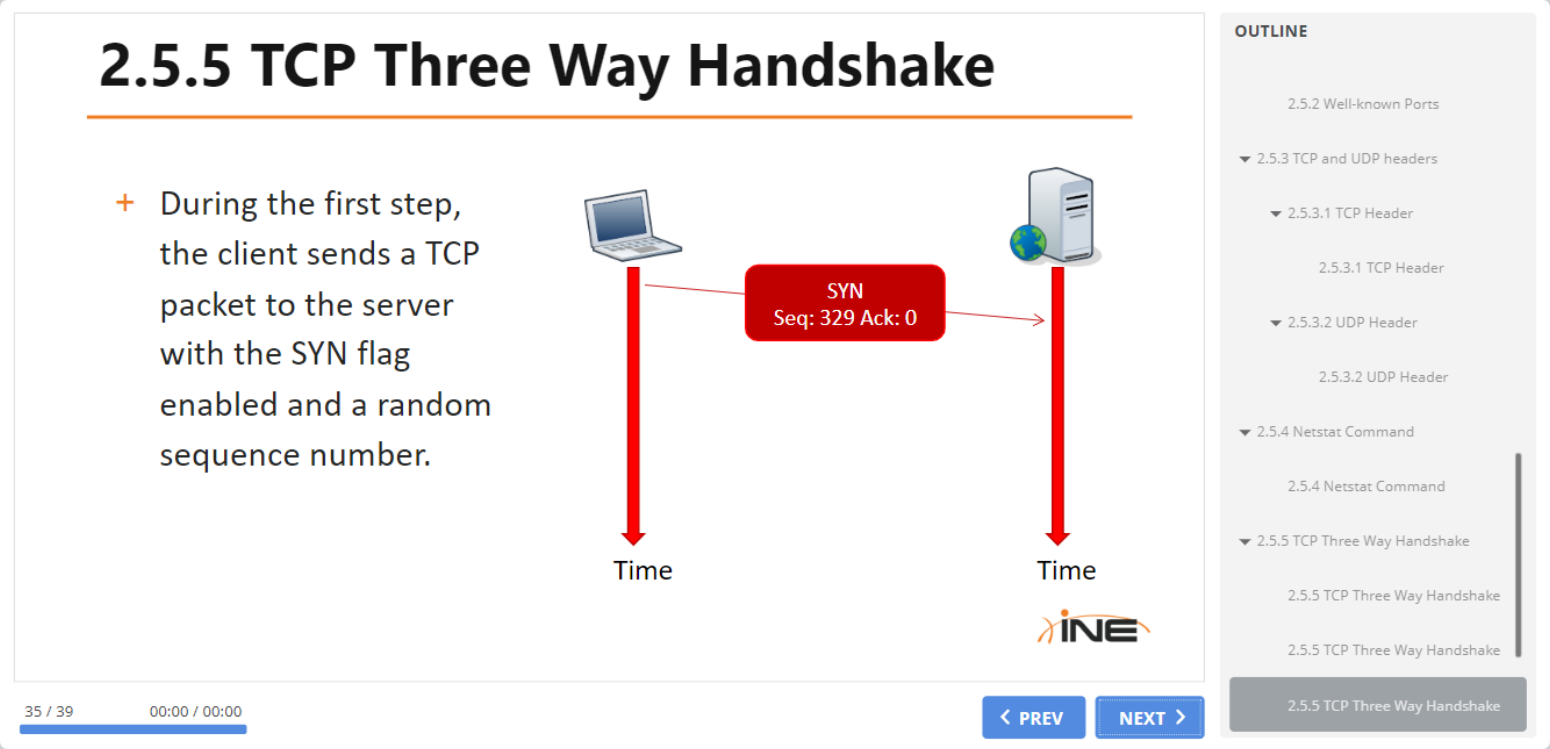
```
 0                   1
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Source Port           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        Sequence Number         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Acknowledgment Number      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Data |          |U|A|P|R|S|F|
 | Offset|  Reserved  |R|C|S|S|Y|I|
 |       |          |G|K|H|T|N|N|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                |
```

OUTLINE

2.5.5 TCP Three Way Handshake

+ The steps in the handshake are used to synchronize the sequence and acknowledgment numbers between the server and the client.



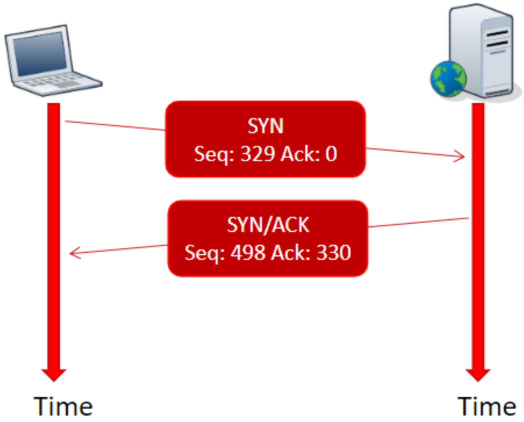


2.5.5 TCP Three Way Handshake

- During the first step, the client sends a TCP packet to the server with the SYN flag enabled and a random sequence number.

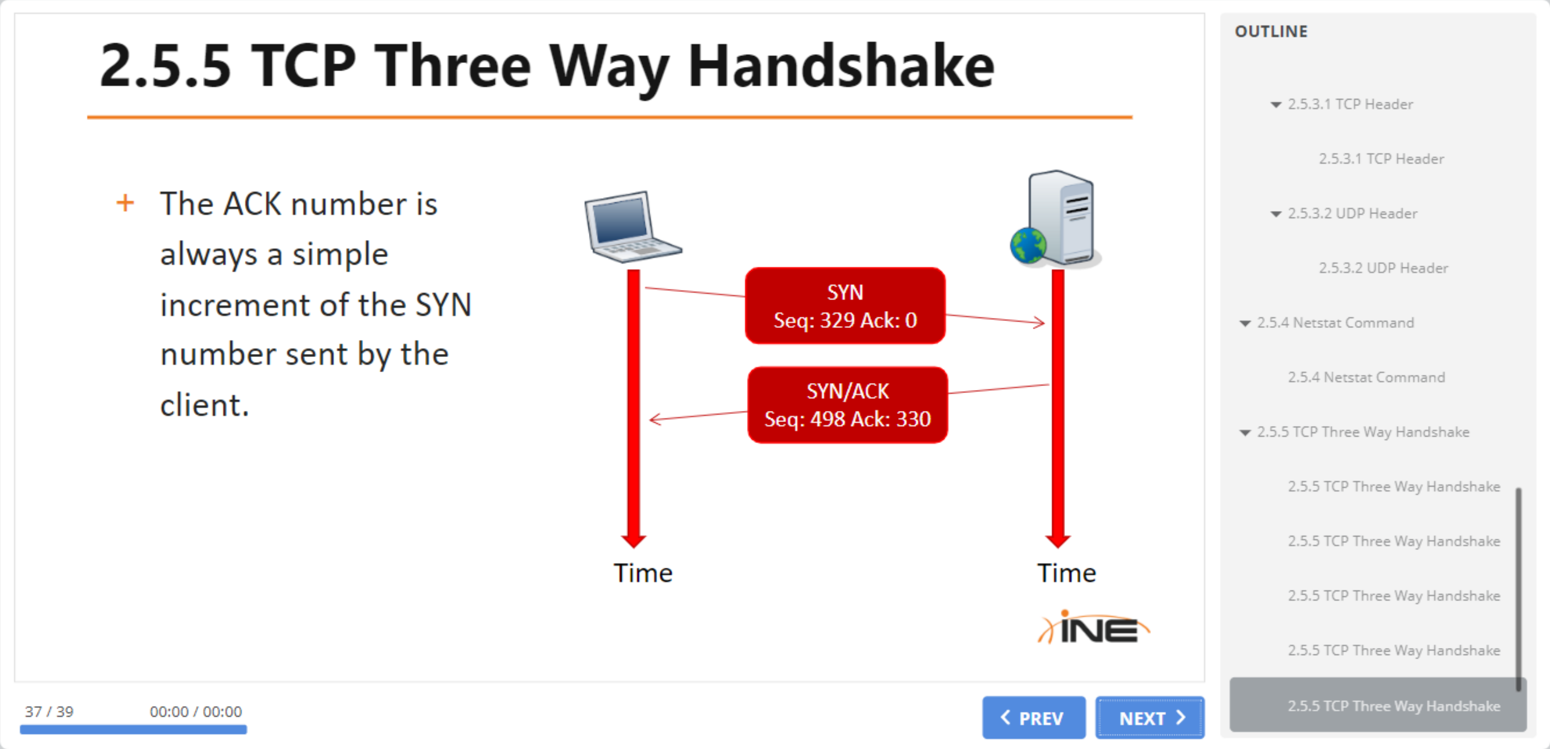
2.5.5 TCP Three Way Handshake

- In the second step, the server replies by sending a packet with both the SYN and ACK flag set and another random sequence number.



36 / 39 00:00 / 00:00

‹ PREV NEXT ›



2.5.5 TCP Three Way Handshake

- The ACK number is always a simple increment of the SYN number sent by the client.

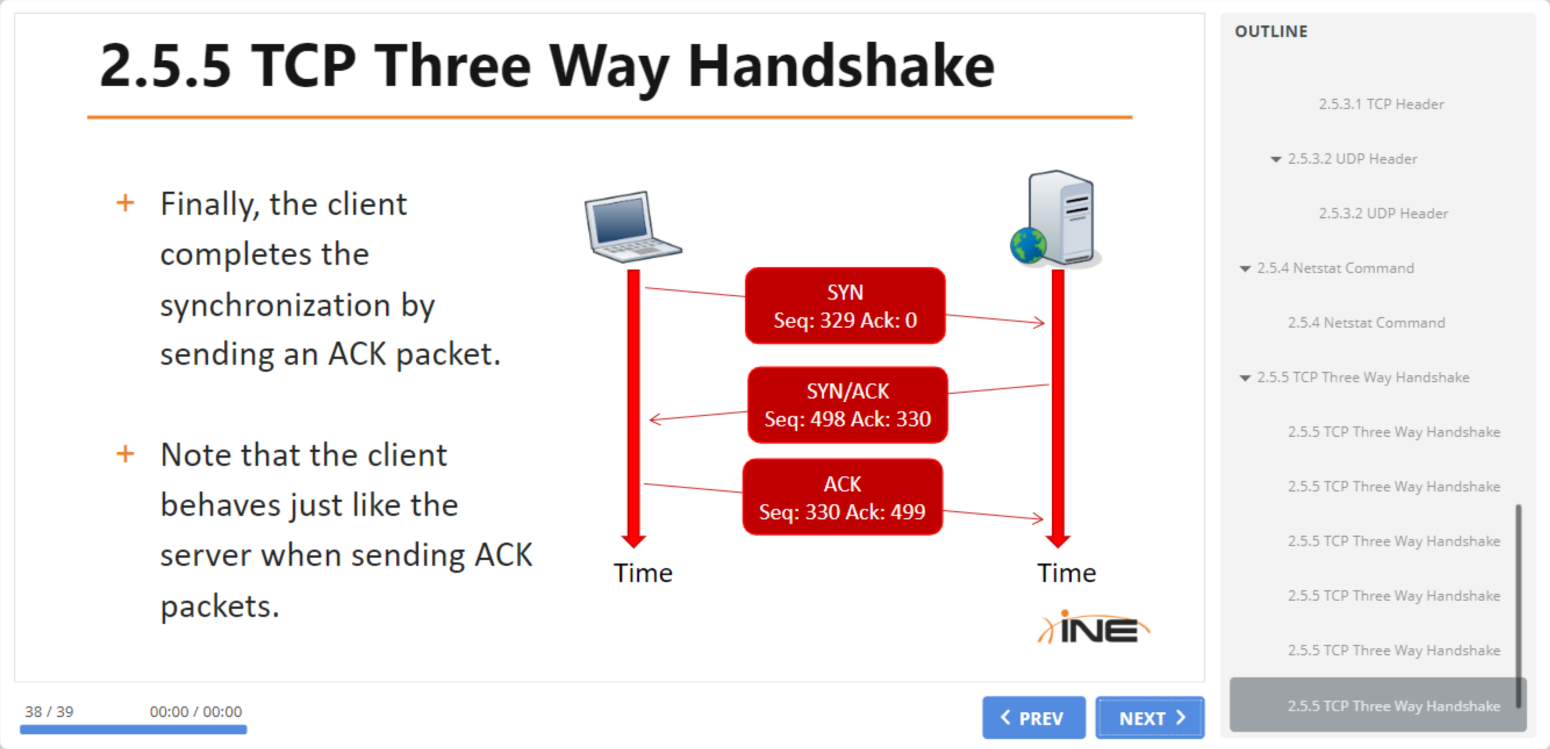
SYN
Seq: 329 Ack: 0

SYN/ACK
Seq: 498 Ack: 330

Time

Time

INE
OUTLINE
2.5.3.1 TCP Header
2.5.3.1 TCP Header
2.5.3.2 UDP Header
2.5.3.2 UDP Header
2.5.4 Netstat Command
2.5.4 Netstat Command
2.5.5 TCP Three Way Handshake
2.5.5 TCP Three Way Handshake
2.5.5 TCP Three Way Handshake
2.5.5 TCP Three Way Handshake
2.5.5 TCP Three Way Handshake
2.5.5 TCP Three Way Handshake
37 / 39
00:00 / 00:00
PREV
NEXT



2.5.5 TCP Three Way Handshake

- Finally, the client completes the synchronization by sending an ACK packet.
- Note that the client behaves just like the server when sending ACK packets.

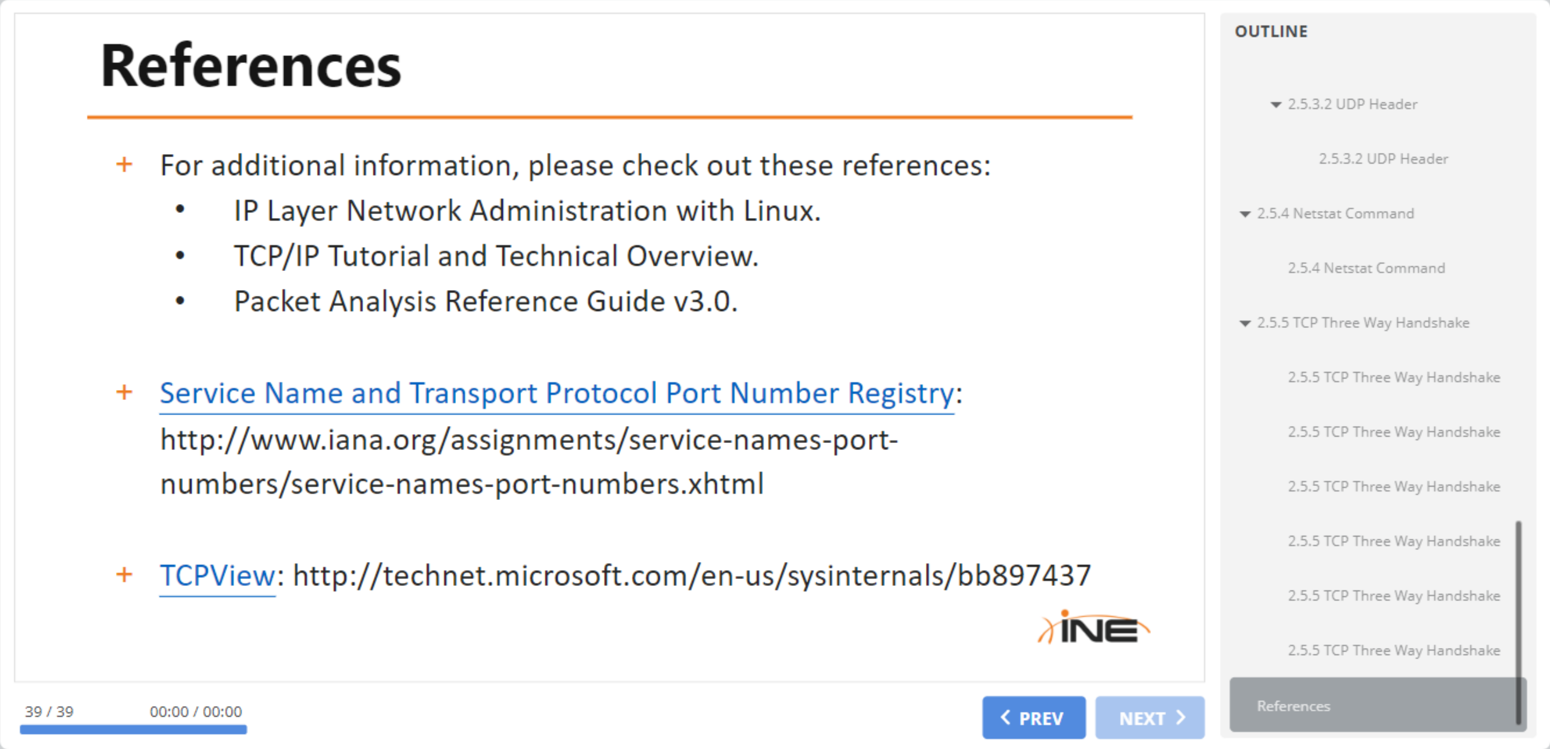
SYN
Seq: 329 Ack: 0

SYN/ACK
Seq: 498 Ack: 330

ACK
Seq: 330 Ack: 499

Time

Time



References

- For additional information, please check out these references:
  - IP Layer Network Administration with Linux.
  - TCP/IP Tutorial and Technical Overview.
  - Packet Analysis Reference Guide v3.0.

- Service Name and Transport Protocol Port Number Registry: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

- TCPView: http://technet.microsoft.com/en-us/sysinternals/bb897437