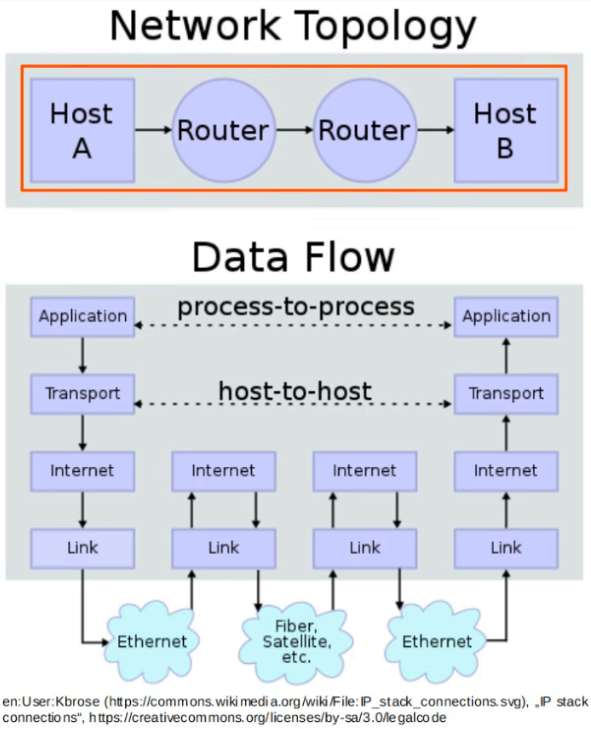
TCP and UDP Overview (Layer 4 Protocols)

- **Lecture Focus**: Comparison of TCP and UDP (CCNA Exam Topic 1.5)
- **Course Context**: Covers Layer 4 protocols.
  - Previous coverage: Layer 1 (cables), Layer 2 (MAC addresses, switching, STP), Layer 3 (IP addresses, routing).
- **Lecture Agenda**:
  1. Basics of Layer 4
  2. TCP (Transmission Control Protocol)
  3. UDP (User Datagram Protocol)
  4. Comparison of TCP vs. UDP

Exam Goal: High-level understanding of basic characteristics and differences (comparing the two).

Basic Functions of Layer 4 Protocols

- **Transparent Transfer of Data Between End Hosts**



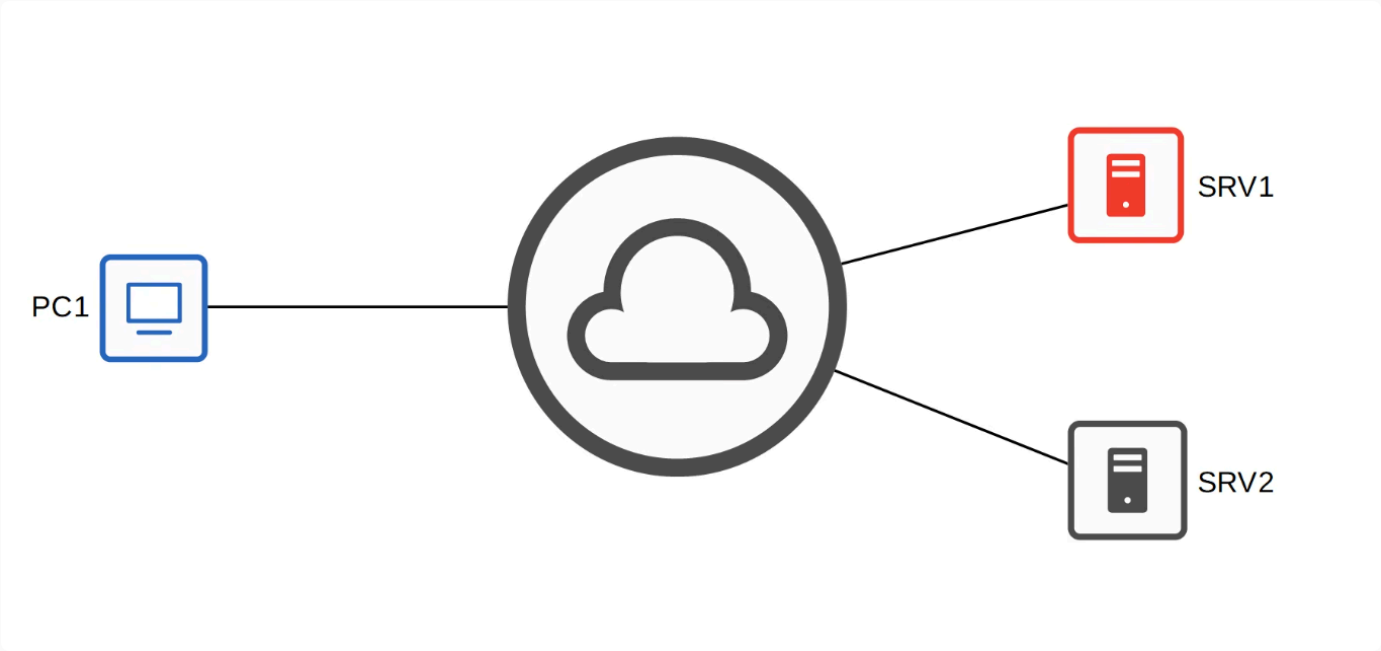
en:User:Kbrose (https://commons.wikimedia.org/wiki/File:IP_stack_connections.svg), „IP stack connections", https://creativecommons.org/licenses/by-sa/3.0/legalcode

- The Transport Layer encapsulates data with a Layer 4 header.
- It then uses the services of lower layers (Layers 3, 2, and 1) to deliver the data unchanged to the destination host.
- The hosts are not aware of the details of the underlying network; the transfer is "transparent" to them.
- *Example:* Host A sending data to Host B across a network topology.

- **Providing Services to Applications**
  - TCP provides these services, UDP does not.
  - **Reliable data transfer**: Making sure the destination host received every bit of data it is supposed to.
  - **Error recovery**: If an error occurs in transmission, Layer 4 can ensure the data is sent again.
  - **Data sequencing**: Making sure that even if data arrives out of order, the end host can sequence it in the correct order.
  - **Flow control**: Making sure the source host doesn't send traffic faster than the destination host can handle.
- **Layer 4 Addressing (Port Numbers)**
  - Layer 4 addresses are called "port numbers."
  - Note: The word "port" can also refer to physical interfaces on network devices, but the Layer 4 port is a different meaning.
  - **Functions of port numbers**:
    - Identifying the Application Layer protocol being used.
    - Providing "session multiplexing."

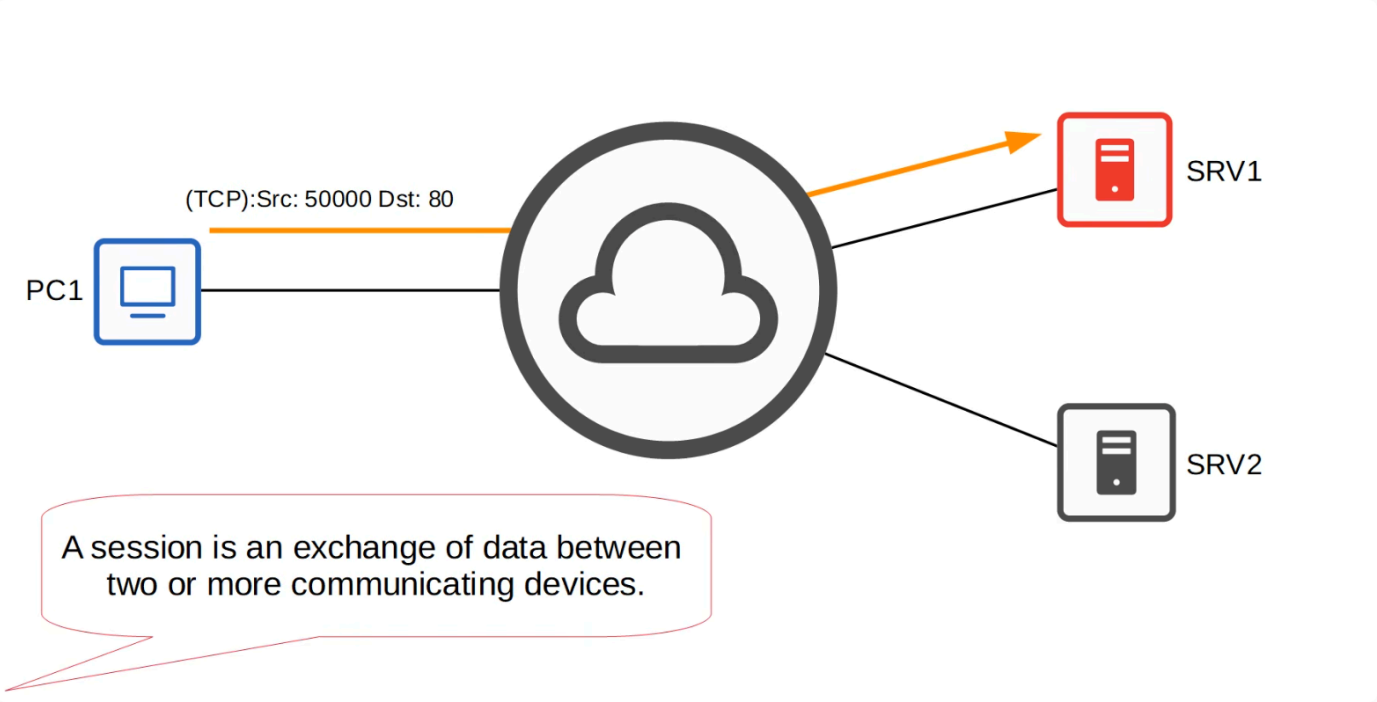
Port Numbers and Session Multiplexing Explained

- **Definition of a Session**

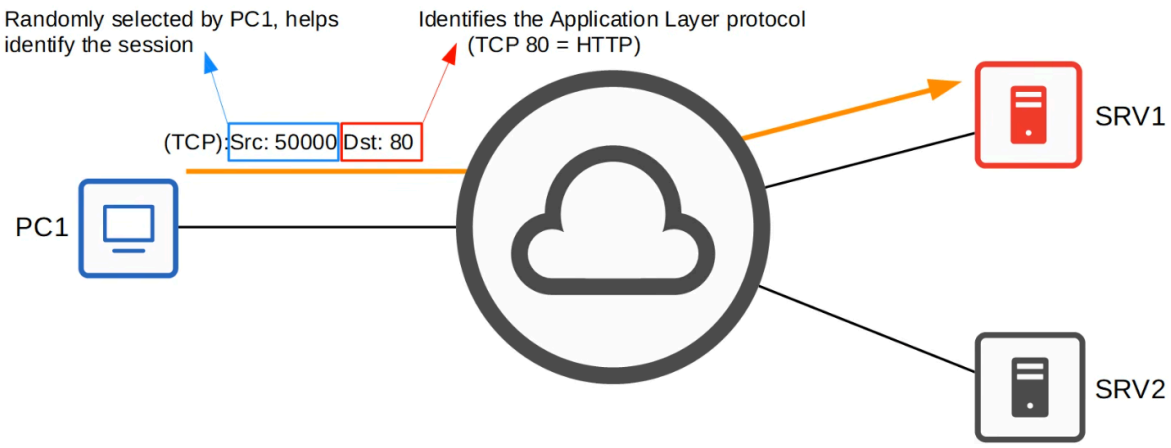


  - A session is an exchange of data between two or more communicating devices.
  - A personal computer needs to handle multiple communication sessions at once (e.g., multiple internet tabs open).

- **Identifying Services and Sessions (Example)**

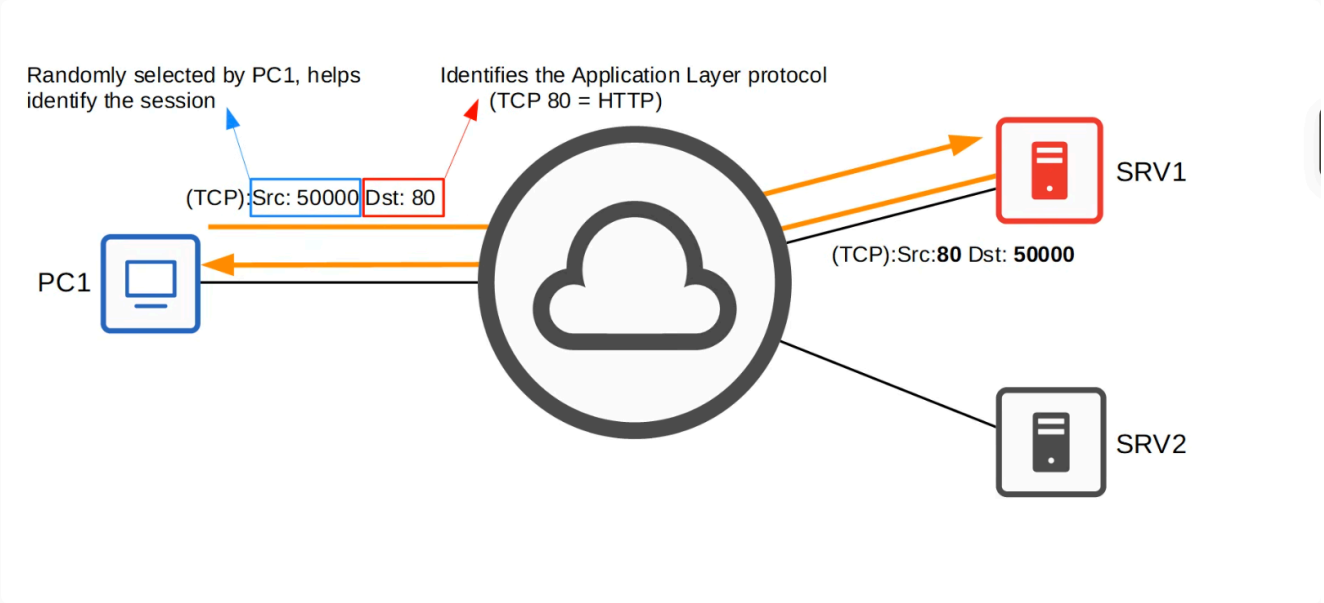


- *Example:* PC1 communicates with SRV1 using TCP.



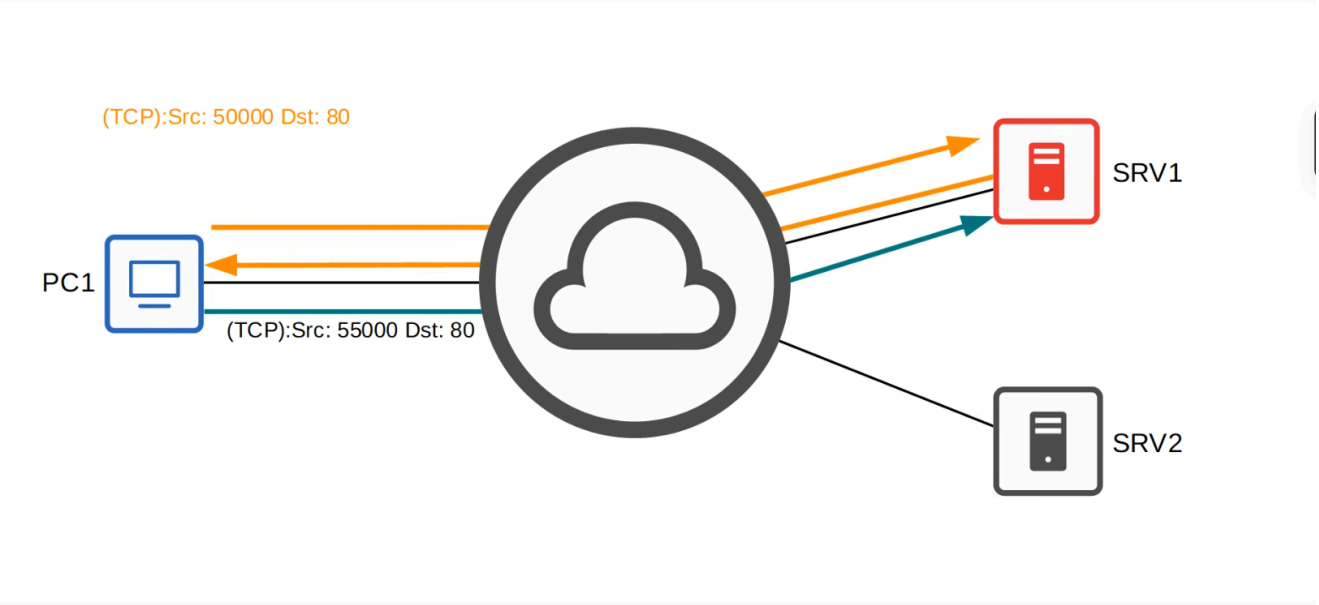
- **Source Port**: 50000 (randomly selected by PC1).
- **Destination Port**: 80.
- The destination port identifies the Application Layer protocol (e.g., TCP port 80 is used for HTTP to access websites).
- The source port, in combination with the destination port, helps identify the specific session.

- *Example:* SRV1 replies to PC1.

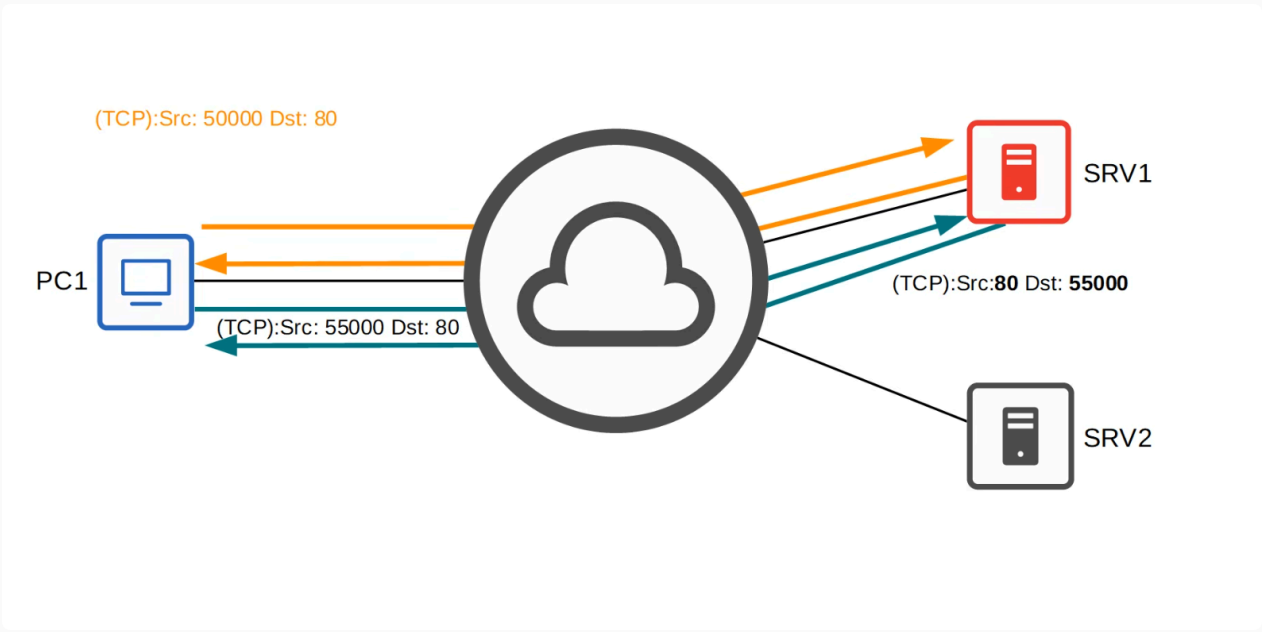


  - **Source Port**: 80.
  - **Destination Port**: 50000.
  - The reversal of port numbers tells PC1 the reply belongs to the same session.

- *Example: What if* PC1 opens a separate connection to SRV1 for HTTP.

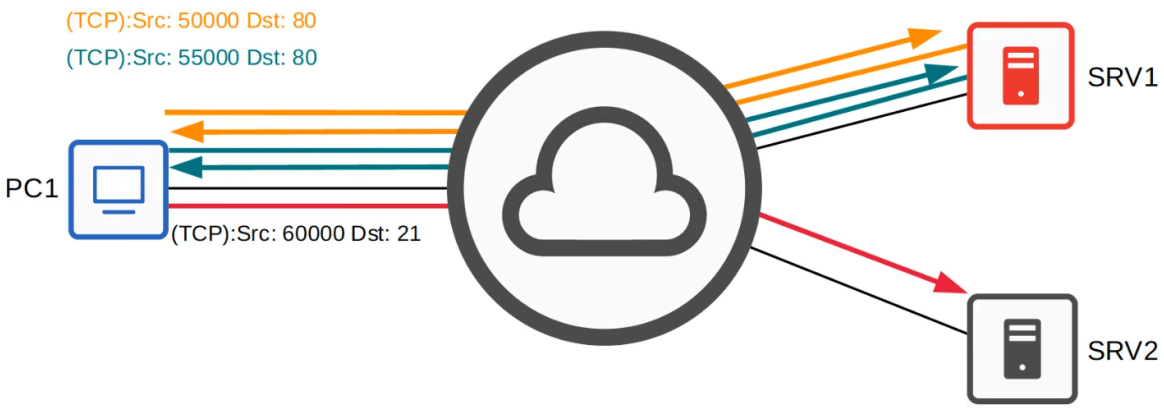


- It would use destination port 80 again. But it is using a different source port.

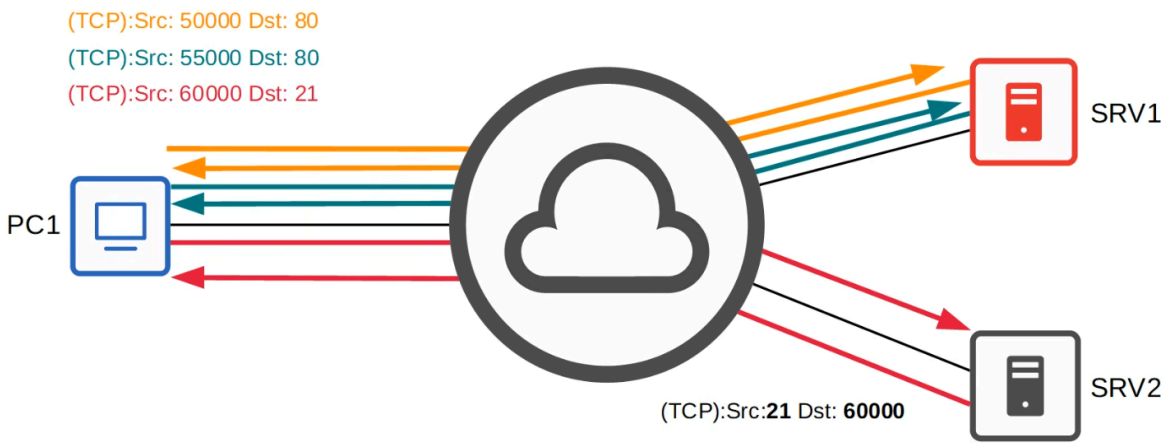


- SRV1 reponse will use that source port as the DST port for its response.
- So PC1 knows it is part of that session.

- *Example:* PC1 communicates with SRV2 for a file transfer using TCP.



  - **Source Port**: 60000.
  - **Destination Port**: 21 (used for FTP, the File Transfer Protocol).
- *Example:* SRV2 replies to PC1 will reverse the port number.



  - **Source Port**: 21.
  - **Destination Port**: 60000.
  - This identifies the communication as part of the same file transfer session.

Port Number Ranges (IANA)

- Port numbers used by Application Layer protocols are registered with the **IANA (Internet Assigned Numbers Authority)**.
- **Well-known ports**: 0–1023
  - Used for major protocols (e.g., HTTP, FTP).
  - Very strictly regulated.
- **Registered ports**: 1024–49151
  - Registration is required to use these.
  - Not as strict as the well-known port range.
- **Ephemeral ports** (also known as private or dynamic ports): 49152–65535
  - Hosts use this range when selecting the random source port.
  - Note: In the previous examples, all randomly selected source port numbers came from this range.
- Port numbers are a function of both main Layer 4 protocols, TCP and UDP.

Transmission Control Protocol (TCP) Overview

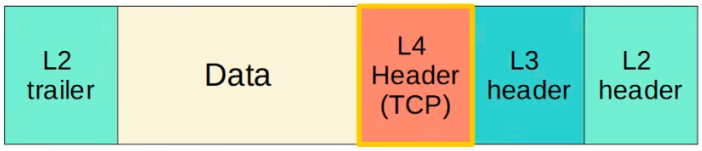
- **Connection-Oriented**
  - Before sending data, hosts communicate to establish a connection.
  - Once the connection is established, the data exchange begins.
- **Reliable Communication**
  - The destination host must acknowledge (Ack) that it received each TCP segment.
  - Note: "Segment" is the Layer 4 PDU (Protocol Data Unit), similar to "packet" at Layer 3 and "frame" at Layer 2.
  - If the source host doesn't receive an acknowledgment for a segment, it is sent again.

- **Sequencing**
  - There is a sequence field in the TCP header.
  - Sequence numbers allow the destination host to put segments in the correct order, even if they arrive out of order.
- **Flow Control**
  - The destination host can tell the source host to increase or decrease the rate that data is sent.
  - This prevents the destination host from being overwhelmed by traffic arriving faster than it can process.

TCP Header Overview

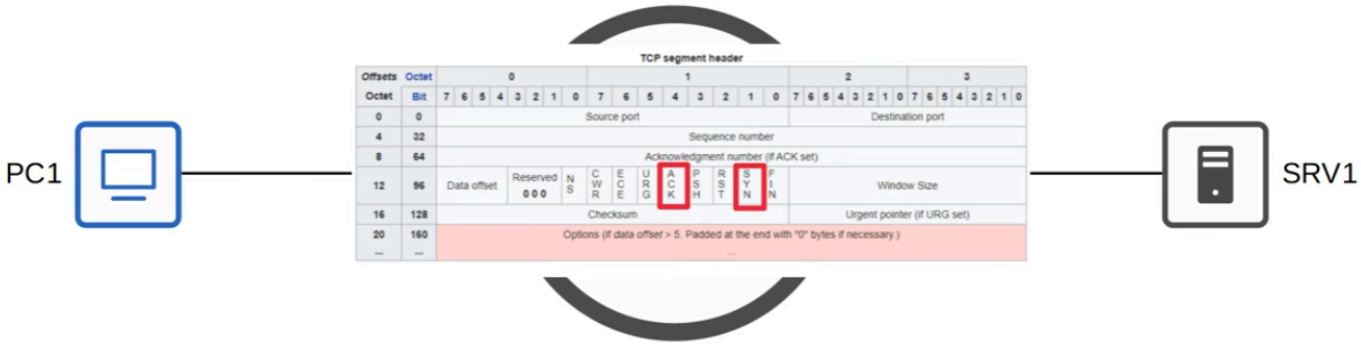
- The TCP header contains many fields used to provide services like sequencing, reliable communication, and flow control.
- Note: For the CCNA, you do not need to memorize the entire header, but should be aware of key fields.

Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															



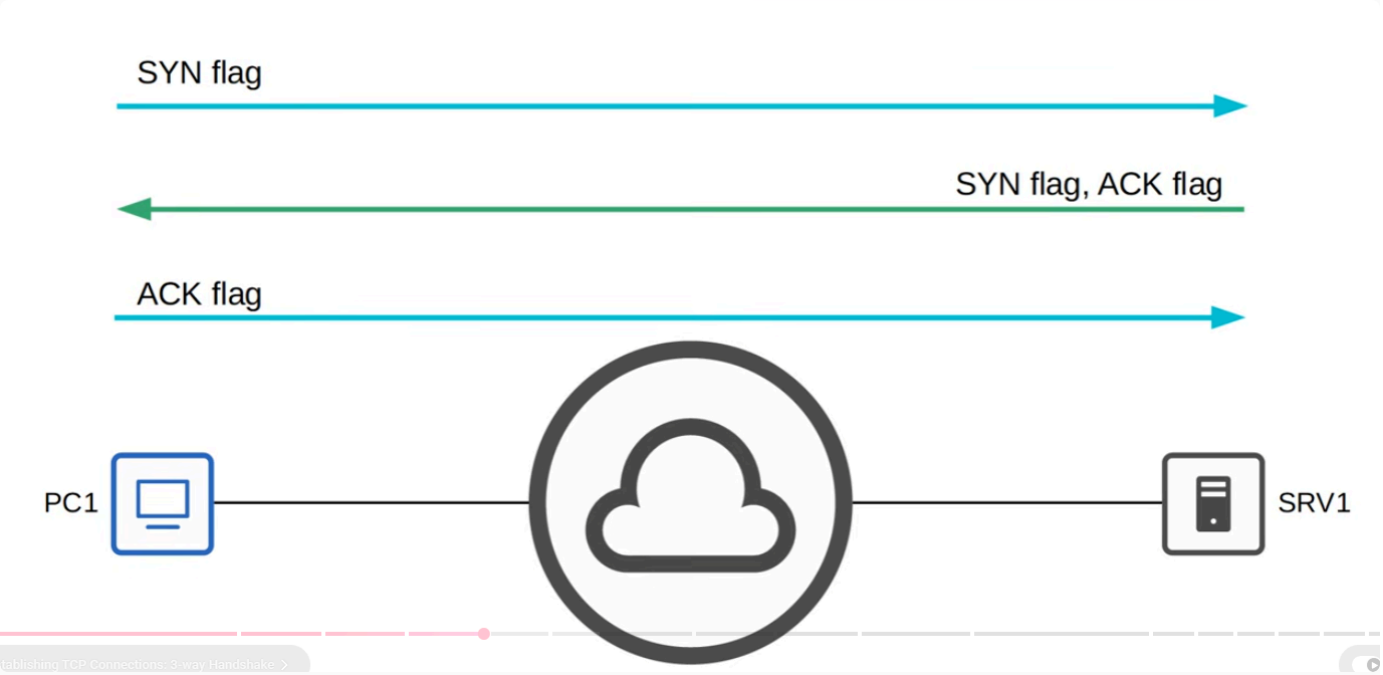
- **Key Fields**:
  - **Source and Destination Port**:
    - Each field is 16 bits (2 bytes) in length.
    - This allows for a total of 65,536 (216216) available port numbers.
  - **Sequence Number and Acknowledgment Number**:
    - Used to provide sequencing and reliable communication.
  - **Flag Bits**:
    - TCP has various flags, but three important ones are **ACK**, **SYN**, and **FIN**.
    - These flags are used to establish and terminate connections.
  - **Window Size**:
    - Used for flow control to adjust the rate at which data is sent.

TCP Three-Way Handshake (Connection Establishment)



- TCP is connection-oriented; hosts establish a connection before sending data.
- Uses TCP flags: **SYN** (Synchronization) and **ACK** (Acknowledgment).

- **Process**:



1. **SYN**: PC1 sends a segment to SRV1 with the **SYN** flag set.
2. **SYN-ACK**: SRV1 replies with a segment with both **SYN** and **ACK** flags set.
3. **ACK**: PC1 sends a final segment with the **ACK** bit set.

- **Result**: The connection is established. The real data exchange begins only after these first three messages.

TCP Four-Way Handshake (Connection Termination)

- Used when a host decides it no longer needs the connection.
- Uses TCP flags: **FIN** (Finish) and **ACK**.

- **Process**:

1. **FIN**: PC1 sends a segment to SRV1 with the **FIN** flag set.
2. **ACK**: SRV1 responds with an **ACK**.
3. **FIN**: SRV1 then sends its own **FIN**.
4. **ACK**: PC1 sends an **ACK** in response to SRV1's FIN.

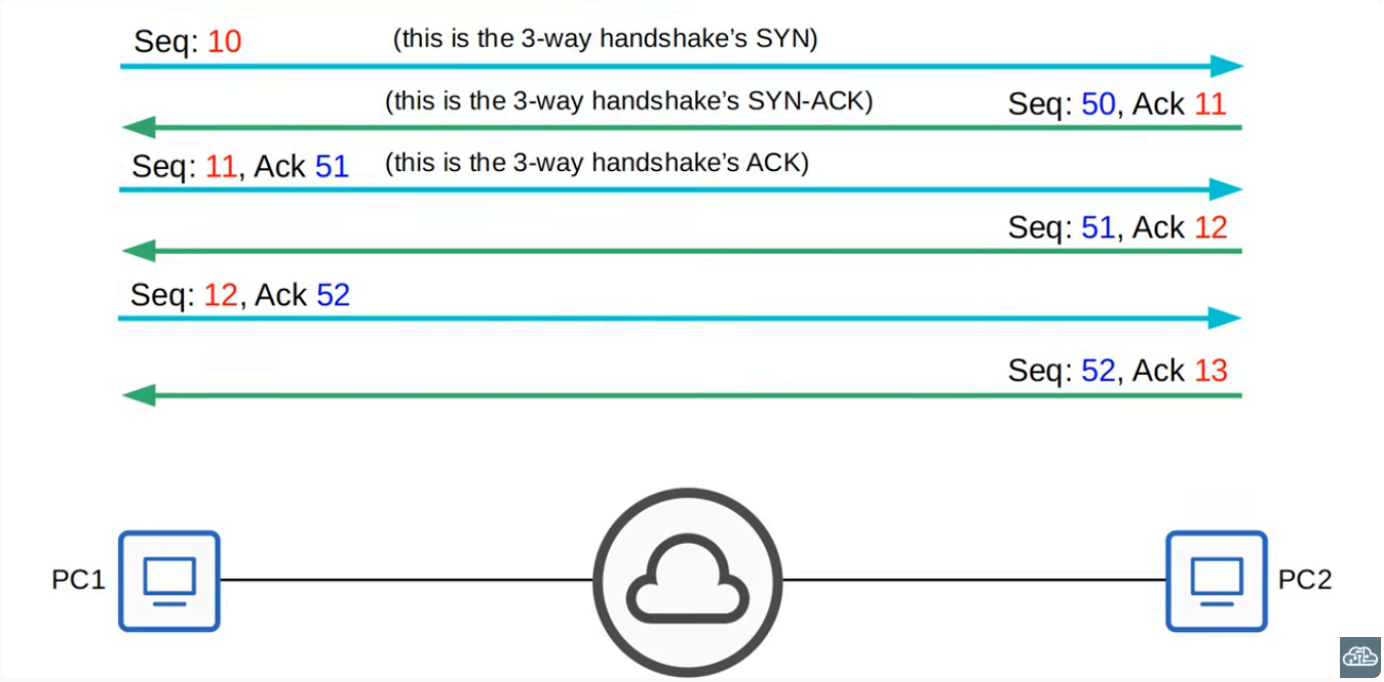
TCP Connection Oriented Summary

- Before actually sending data to the destination host, the two hosts communicate to establish a connection.
- Once the connection is established, the data exchange begins.
- As demonstrated, PC1 and SRV1 established a connection using the three-way handshake before exchanging data.

TCP Sequencing and Reliable Communication

- **Forward Acknowledgment**: TCP acknowledges data by stating the sequence number of the **next** expected segment, rather than the one just received.

- **Three-Way Handshake Sequence Example (PC1 and PC2)**:

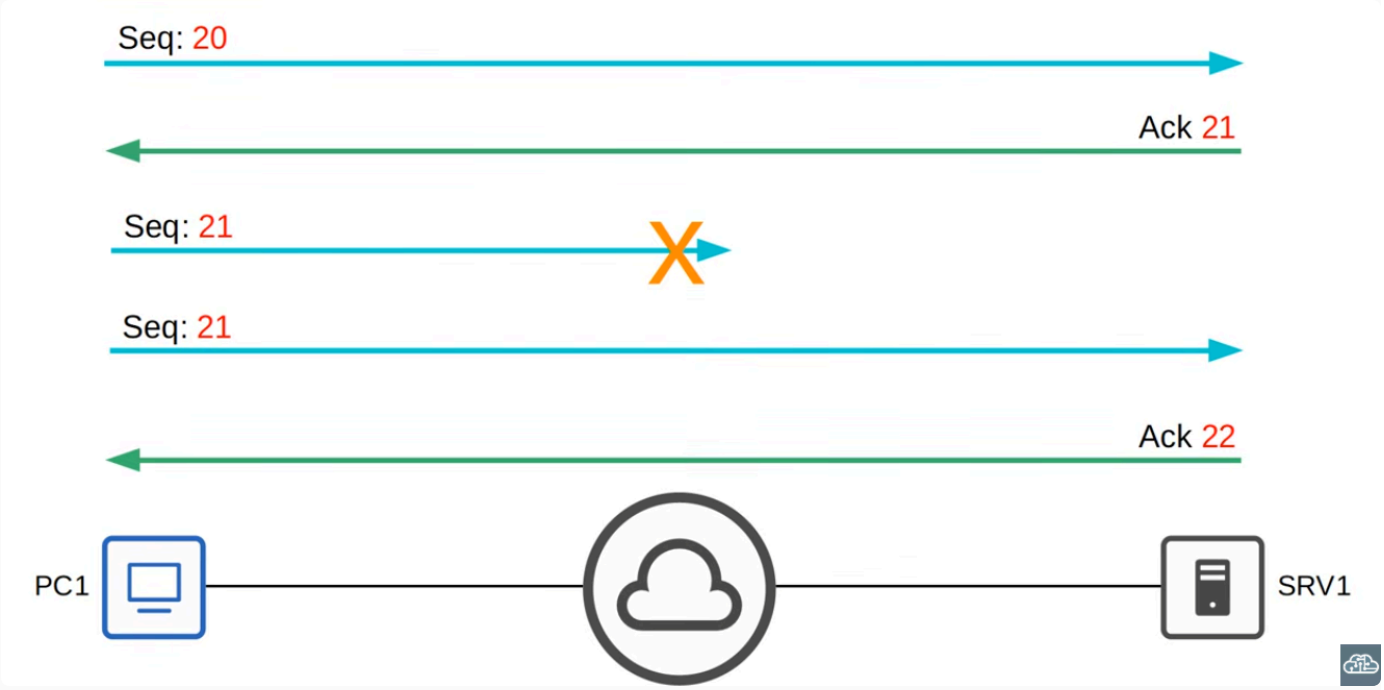


1. **SYN**: PC1 sends a SYN segment to PC2.
    - Sets a random initial sequence number (e.g., 10).
2. **SYN-ACK**: PC2 replies with a SYN-ACK segment.
    - Sets its own random initial sequence number (e.g., 50).
    - Sets the acknowledgment field to **11** (next expected sequence number from PC1).
3. **ACK**: PC1 sends the final ACK segment.
    - Sequence number is **11**.
    - Sets the acknowledgment field to **51** (next expected sequence number from PC2).
4. **Data Exchange**: PC2 sends a segment.
    - Sequence number is **51**.
    - Sets the acknowledgment field to **12** (next expected sequence number from PC1).

- **Summary**:
  - Hosts set a random initial sequence number.
  - Sequence numbers allow hosts to put segments in the correct order even if they arrive out of sequence.

TCP Retransmission

- If a segment isn't acknowledged, the source host sends it again.
- **Example**:



  - PC1 sends SRV1 a segment with sequence number 20.
  - Using forward acknowledgment, SRV1 sends Ack 21 to PC1.
  - PC1 then sends Sequence number 21, but it doesn't reach SRV1.
  - After waiting a certain amount of time with no Ack, PC1 resends the segment.
  - This is called **TCP retransmission**.
  - This time SRV1 receives it, and sends Ack 22 to tell PC1 that it was received.

TCP Flow Control

- Acknowledging every single segment, no matter what size, is inefficient.
- The TCP header's **window size field** allows more data to be sent before an acknowledgment is required.
- **Example**:

  - A host could send three segments, with sequence numbers 20, 21, and 22, and then an Ack is sent with sequence number 23.
- **Sliding Window**:
  - A "sliding window" is used to dynamically adjust how large the window size is.
  - The window size is increased as much as possible until a segment is dropped, then the window size backs down to a more reasonable level, and slowly increases again.
- **Note**:
  - In all of these examples, simple sequence numbers were used.
  - In real situations, the sequence numbers get much larger and do not increase by 1 with each message, especially when the sliding window size gets very large.
  - For the CCNA, just understand the concepts and don't worry about the exact numbers.

User Datagram Protocol (UDP) Overview

- **Connectionless**
  - UDP is not connection-oriented; the sending host does not establish a connection with the destination host before sending data.
  - Data is simply sent.
- **No Reliable Communication**
  - Acknowledgments are not sent for received segments.
  - If a segment is lost, UDP has no mechanism to re-transmit it.
  - Segments are sent "best-effort" (provides no guarantee of delivery; makes the effort but without guarantees).
- **No Sequencing**
  - Unlike TCP, UDP has no sequence field in its header.
  - If segments arrive out of order, UDP has no mechanism to put them back in order.
- **No Flow Control**
  - UDP has no mechanism like TCP's window size to control the flow of data.

UDP Header Overview

- **Fields**:

UDP datagram header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

  - Source port number
  - Destination port number
  - Length field (indicating the length of the segment)
  - Checksum (so the receiving host can check for errors)

Comparing TCP and UDP

- **Header Comparison**

TCP segment header

Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

UDP datagram header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

  - TCP has a larger header with more fields, allowing it to provide additional functions like sequencing and error recovery.
  - UDP has a smaller, simpler header.
- **When to Use TCP**
  - TCP provides more features but at the cost of additional overhead (larger header).
  - Acknowledgments and retransmissions can slow down data transfer.
  - Preferred for applications that require reliable communications.
  - *Example:* Downloading a file (e.g., a PDF). You want to make sure you get the whole file without a page missing.
- **When to Use UDP**
  - Preferred for real-time voice and video applications.
  - These applications are very delay-sensitive; you don't want the overhead of TCP slowing them down.
  - *Example:* VoIP phone calls, Zoom, Skype.

- **Special Cases**
  - Some applications use UDP but provide reliability within the application itself.
  - *Example:* TFTP (Trivial File Transfer Protocol).
  - *Example:* During a Skype call, if audio cuts out, you can ask the other person to repeat what they said (effectively asking for a "retransmission").
  - Some applications use both TCP and UDP depending on the situation.
  - *Example:* DNS (Domain Name System).

TCP vs UDP Comparison Summary

TCP	UDP
Connection-oriented	Connectionless
Reliable	Unreliable
Sequencing	No sequencing
Flow control	No flow control
Use for downloads, file sharing, etc	Used for VoIP, live video, etc

- Both TCP and UDP provide Layer 4 addressing in the form of port numbers.
- These port numbers identify Application Layer protocols and allow for session multiplexing.
- These are essential functions provided by both Layer 4 protocols.

Important Port Numbers (Application Layer Protocols)

- **Protocols Using TCP**
  - **FTP** (File Transfer Protocol): Uses TCP ports 20 and 21.
  - **SSH** (Secure Shell): Uses TCP port 22. Commonly used to connect to the CLI of routers and switches.
  - **Telnet**: Uses TCP port 23. Can also be used to connect to the CLI of devices.
  - **SMTP** (Simple Mail Transfer Protocol): Uses TCP port 25. Used for sending email.
  - **HTTP** (Hypertext Transfer Protocol): Uses TCP port 80. Commonly used for accessing web pages.
  - **POP3** (Post Office Protocol 3): Uses TCP port 110. Used for retrieving emails.
  - **HTTPS** (Hypertext Transfer Protocol Secure): Uses TCP port 443.
- **Protocols Using UDP**
  - **DHCP** (Dynamic Host Configuration Protocol): Uses UDP ports 67 and 68. Allows hosts to automatically set their IP address.
  - **TFTP** (Trivial File Transfer Protocol): Uses UDP port 69.
  - **SNMP** (Simple Network Management Protocol): Uses UDP ports 161 and 162.
  - **Syslog**: Uses UDP port 514.
- **Protocols Using Both TCP and UDP**
  - **DNS** (Domain Name System): Usually uses UDP, but uses TCP in some situations.

Lecture Review Summary

- **Layer 4 Basics**: Covered the basics of Layer 4, including Layer 4 addressing in the form of port numbers.
- **TCP**: Looked at TCP, a Layer 4 protocol that provides various services to applications, such as reliable communication and flow control.
- **UDP**: Looked at UDP, which does not provide the various services that TCP does, but uses a smaller header with less overhead.

- **Comparison**: Spent time comparing the two protocols.
- **Exam Focus**: Remember the exam topics list expects you to be able to compare the two for the exam, so focus on that.
