Transport Layer Protocols

Applied Network security (comp420 –001)

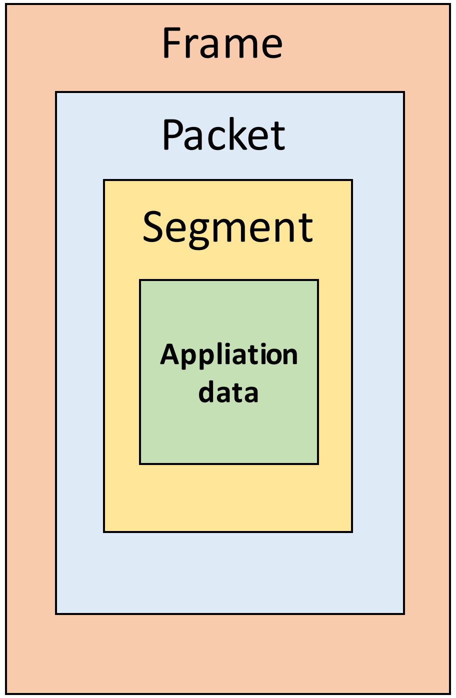
Department of Computer science

Fall 2020

Transport Layer Protocols

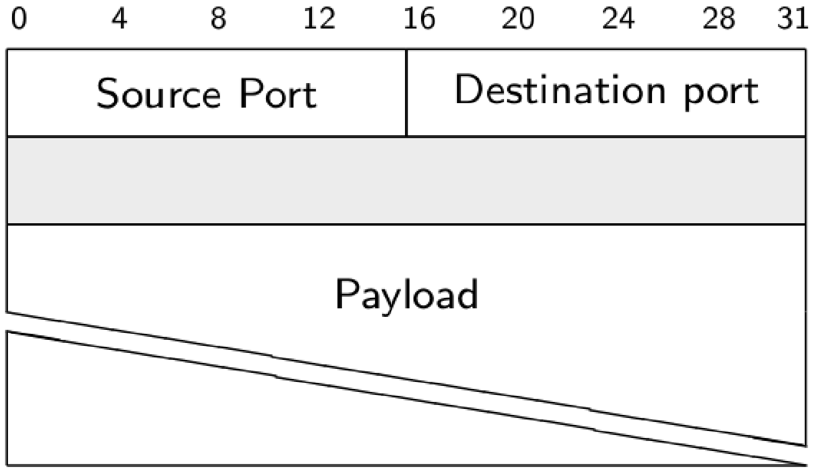
- Transport layer protocols pass data between the application and network layers through ports.
- User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are two famous transport protocols.

Layer	Protocol	Addressing	Devices
Application	HTTP, DNS, …	----	----
Transport	TCP, UDP	Sockets	----
Internetworking (IP)	IPv4, IPv6	IP	
Link (Ethernet)	Ethernet	MAC	



User Datagram Protocol (UDP)

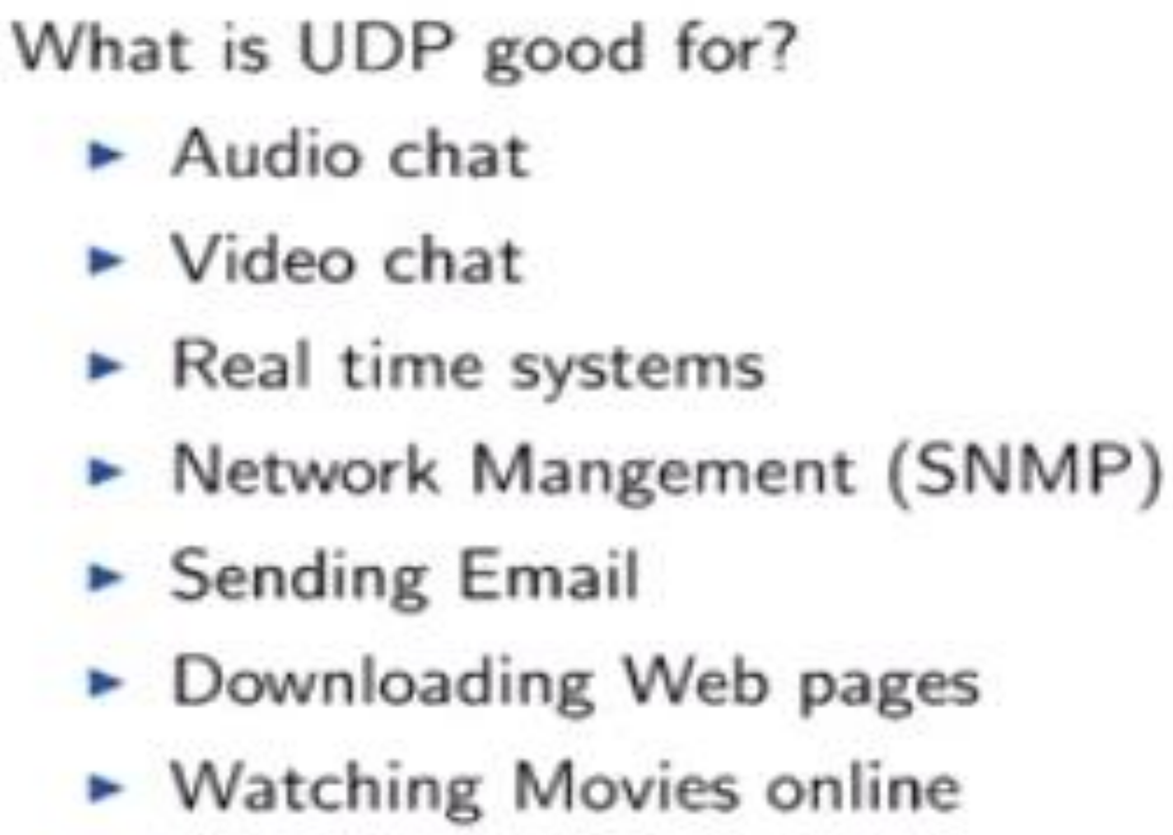
- UDP communicates with processes at the application layer through ports and at the network layer through IP.
- UDP is what we call connectionless. That means it does not establish a connection between the source and the destination.
- UDP is about as barebones of a transport layer protocol as you can get, to the point where you're almost working directly with IP. This simplicity is what makes UDP attractive for some applications, such as audio streaming or domai n name services.



UDP (Cont.)

- We'll be focusing on the source port and the destination port, and the payload.
- The UDP packet also includes the length and the checksum, which are grayed out, and we won't be focusing on those.
- The size of the UDP segment may vary from one segment to the next.
- The checksum is used to determine that the application data is correct and if any errors have been introduced into the segment. The checksum is also run over some bits of the IP header as well.
- The payload is where the magic happens. That's the application data.
- UDP segment contains two fields that describe the source and the destination of communication.

Your Turn 1

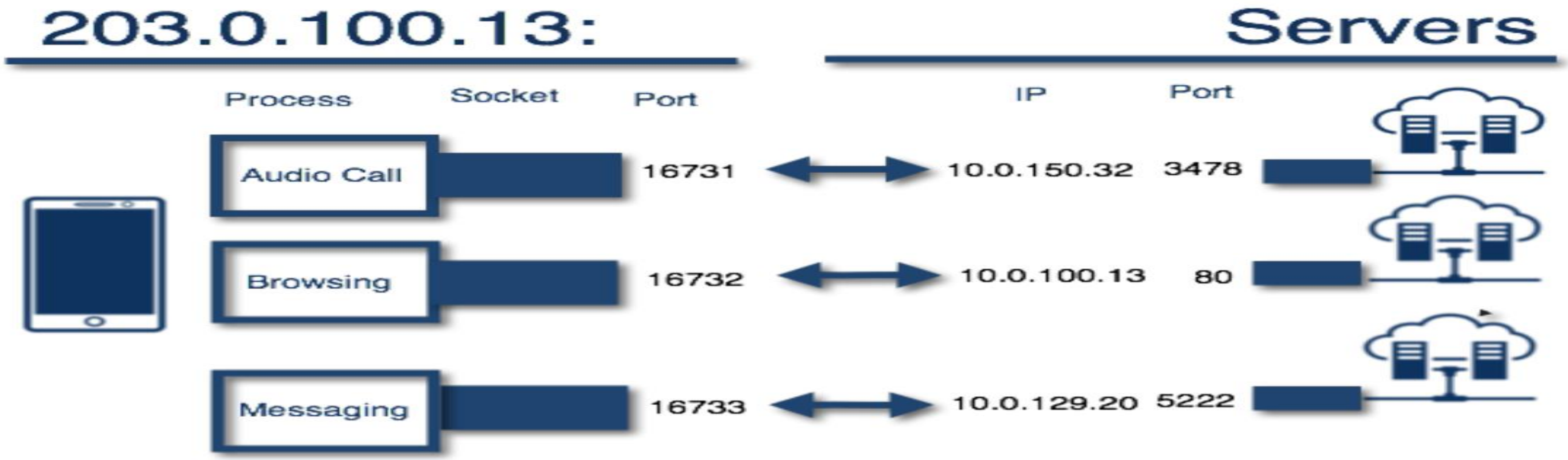


What is UDP good for?

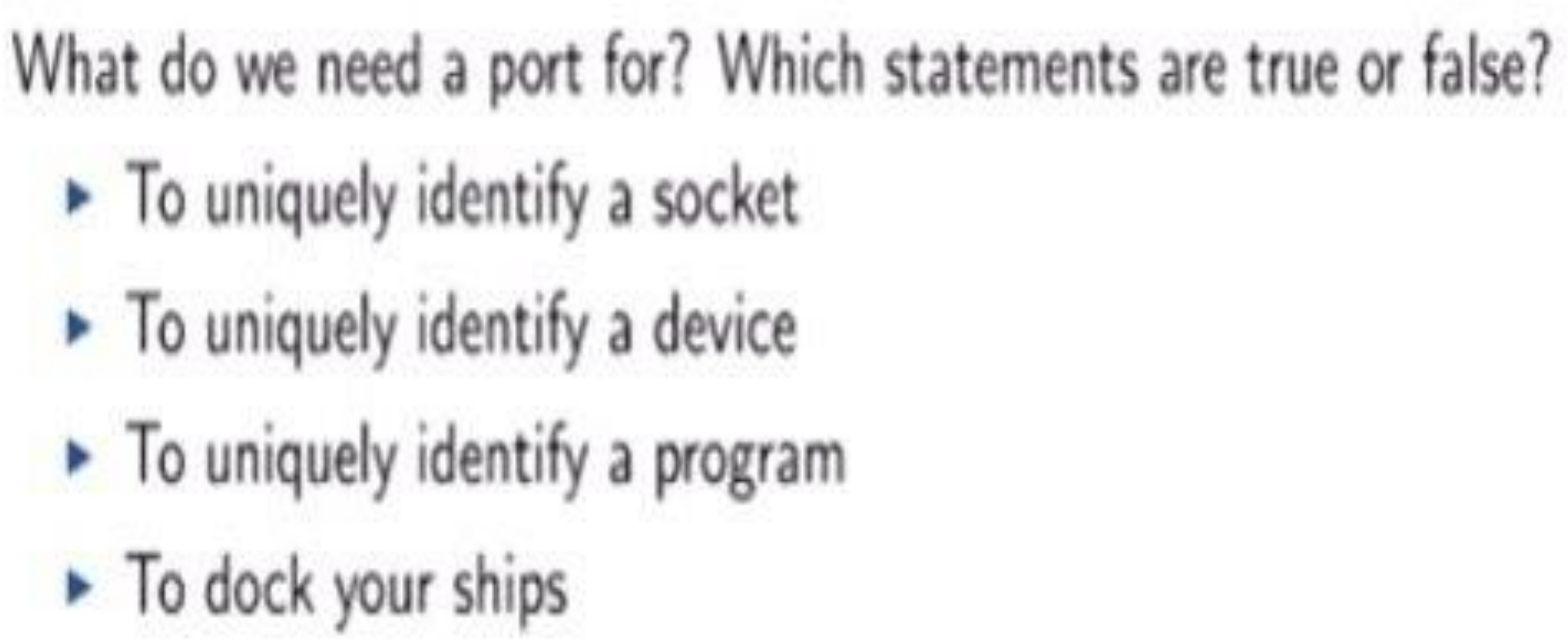
- Audio chat
- Video chat
- Real time systems
- Network Mangement (SNMP)
- Sending Email
- Downloading Web pages
- Watching Movies online

Ports

- Ports answer this fundamental question of how do we connect programs on our device to IP traffic from the network.
- By combining an IP address with a port (referred to as a socket), the device can determine what processes to deliver the payload to.
- The processes do not talk directly to the network. They instead communicate through the sockets. You can consider sockets as doors through which data passes from the network to the process and from the process to the network.



Your Turn 2



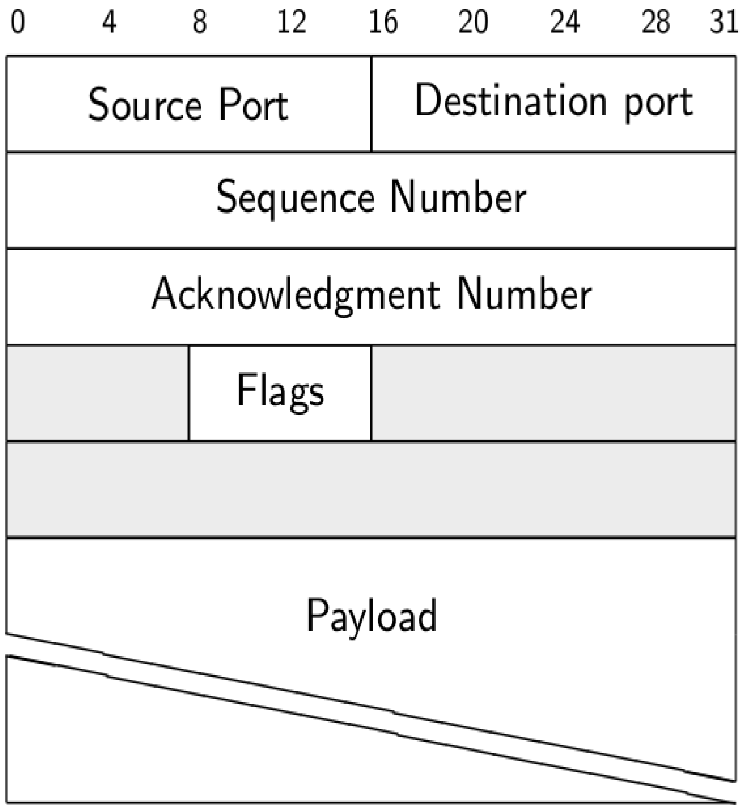
What do we need a port for? Which statements are true or false?

- To uniquely identify a socket
- To uniquely identify a device
- To uniquely identify a program
- To dock your ships

Transmission Control Protocol (TCP)

- TCP is quite different from UDP in several dimensions.
- **First,** TCP is connection oriented, which means that a connection must be established before any data is sent.
- **Second,** TCP is streaming oriented, which means that an application can write data in very small or very large amounts, and the TCP layer will take care of putting them into packets. In contrast to UDP, TCP transmits a continuous stream of bytes not individual messages or records.
  - TCP connection consists of two continuous streams of bytes, one for receiving and one for sending. TCP does not have a notion of discrete messag es with a defined beginning and end within a stream.
- **Third,** TCP provides reliability. TCP makes sure that data delivered and is also delivered in the right order and will be retransmitted if it is lost.

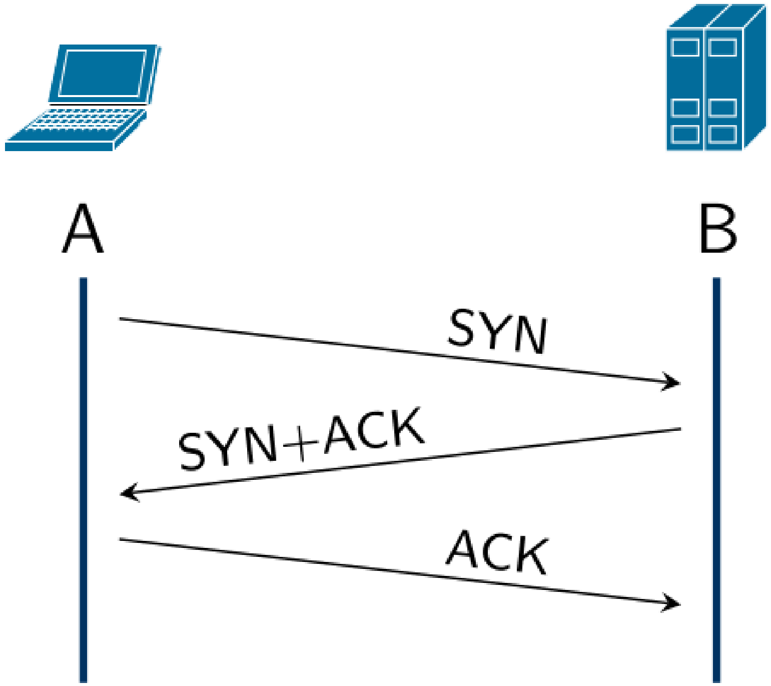
Transmission Control Protocol (TCP)



- Similar to UDP, TCP packets also contain a source and destination port and of course, there is always a payload.
- **Sequence Number:** Position of the packet contents in the overall stream.
- **Acknowledgment Number:** Position up to which the stream has been completely received + 1; i.e., the next expected sequence number.
- **Flags:**
    - **SYN:** Synchronize, i.e., initiate a new connection.
    - **ACK:** Acknowledge receipt of previous packets. Set for all but the first packet.
    - **FIN:** Finish, indicate no more data from sender.
    - **RST:** Reset the connection.

TCP Handshaking

- Let's say we have two parties, A and B, and an application on A's machine wants to talk to B. So it tells the TCP layer to open a connection with B on a certain port.
- A first sends the packet with a SYN flag set. We call this the SYN packet.
- Next, B responds with a SYN packet of its own. The ACK bit is now also sent, so B acknowledges the receipt of SYN packet, and we call this the SYN ACK packet.
- Last but not least, A still needs to confirm that it received B's SYN packet, so A needs to send a final ACK packet.
- An important aspect here is that both parties can only send data after the handshake is complete. For a normal TCP, the initial SYN packet must not carry any data. Here you see why UDP might be preferential for performance-critical applications.



Closing a TCP Connection

- Closing, in fact, works similar to connection establishment.
- If we are done sending data, we send a packet with the FIN flag set. The other side replies with a FIN ACK, and we finally acknowledge that with a final ACK packet.
- The important difference to connection establishment

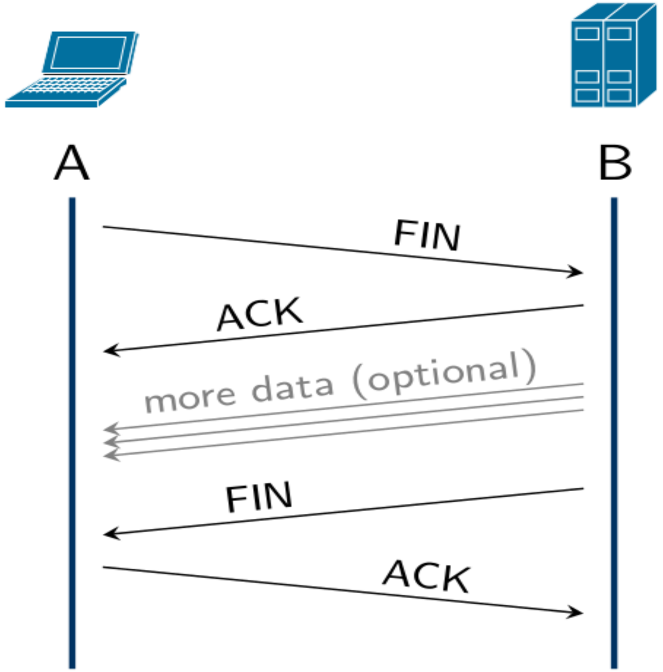
is that connections can be half closed where only one

end has indicated that it is not sending any data anymore.

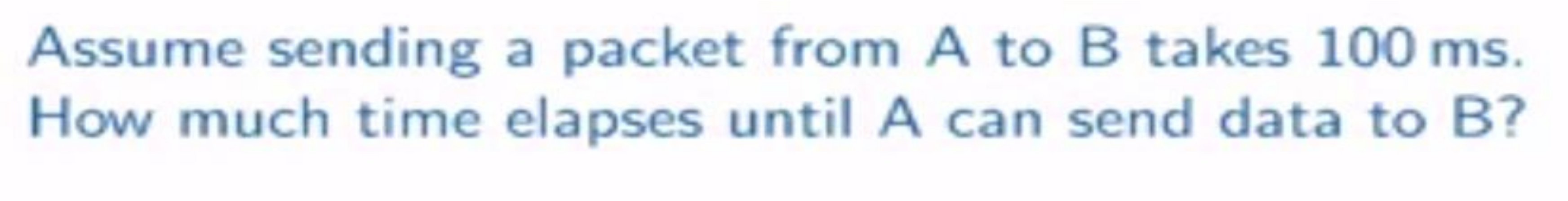
If we close the connection, the other party can

acknowledge that, and then still continue sending data

until it itself sends a FIN packet.



Your Turn 3



Assume sending a packet from A to B takes 100 ms.
How much time elapses until A can send data to B?

TCP Vs. UDP

TCP	UDP
TCP supports host-to-host communication. It is a connection-oriented protocol.	UDP enables process-to-process communication and is a connectionless protocol.
TCP sends individual packets	UDP sends messages, called datagrams
TCP is considered a reliable transport protocol	UDP is considered a best-effort protocol
TCP provides error and flow control. It is meant to provide error-free data transmission through handling retransmission of dropped or garbled packets as well as acknowledgement of all packets that arrived	No such mechanisms are supported in UDP. The packets don't necessarily arrive in order and they may get lost or dropped. No acknowledgement on packet arrival.
TCP communication entails overhead in terms of delays and bandwidth	UDP has much lower bandwidth overhead and latency
TCP is good when we need reliability and is used in HTTP, and Simple Mail Transfer Protocol (SMTP)	UDP is faster and is used for VoIP, video streaming, gaming and live broadcasts