Dr Takfarinas SABER
takfarinas.saber@dcu.ie

CA169
Networks & Internet

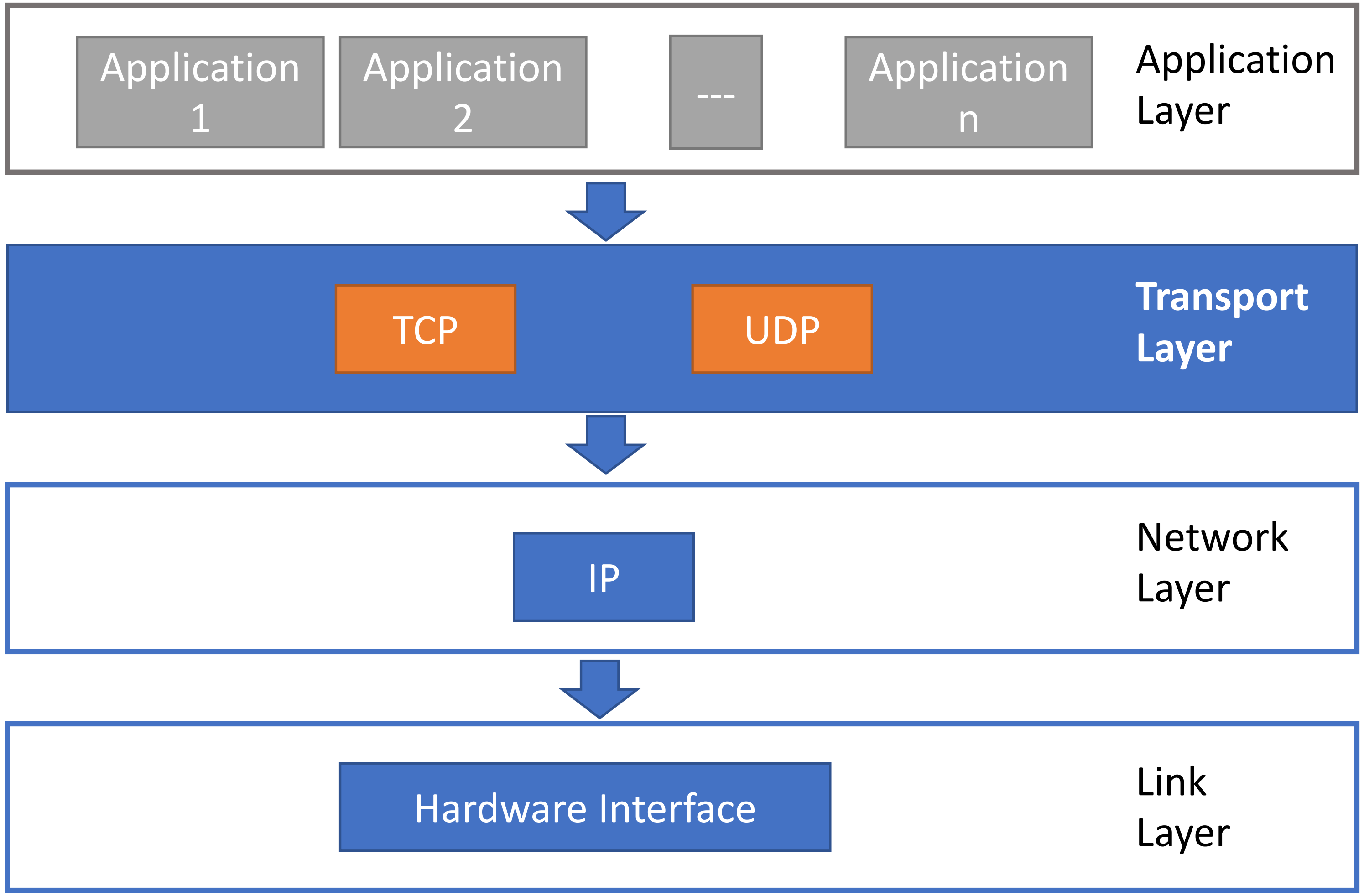
Transport Layer

Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

Last Week

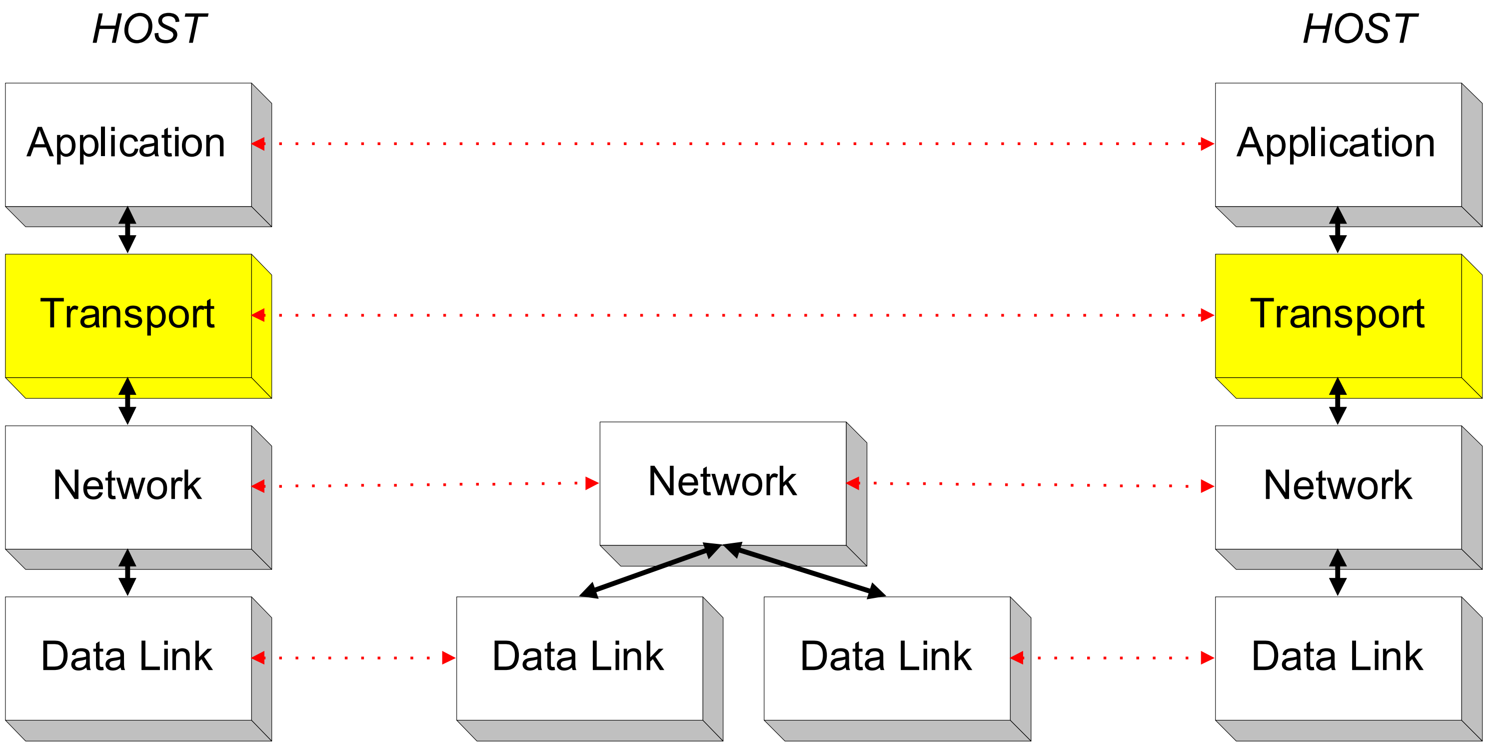
- IP Addressing:
  - an **end to end** addressing
- Two approaches for routing
  1. Static: hand-crafted
  2. Dynamic:
     - Vector Distance
     - Link State

Where do we stand?



Orientation

- Similarly to Network layer protocols, Transport layer protocols are also end-to-end protocols
- They are only implemented at the hosts



Today

- Port Addressing
- UDP: End-to-End Unreliable Transmission
- TCP: End-to-End Reliable Transmission
  - Connection Management
  - Packet Re-transmission
  - Window & Congestion Control
    - Tahoe
    - Reno

Port Addressing

- To address the application where the data is going to

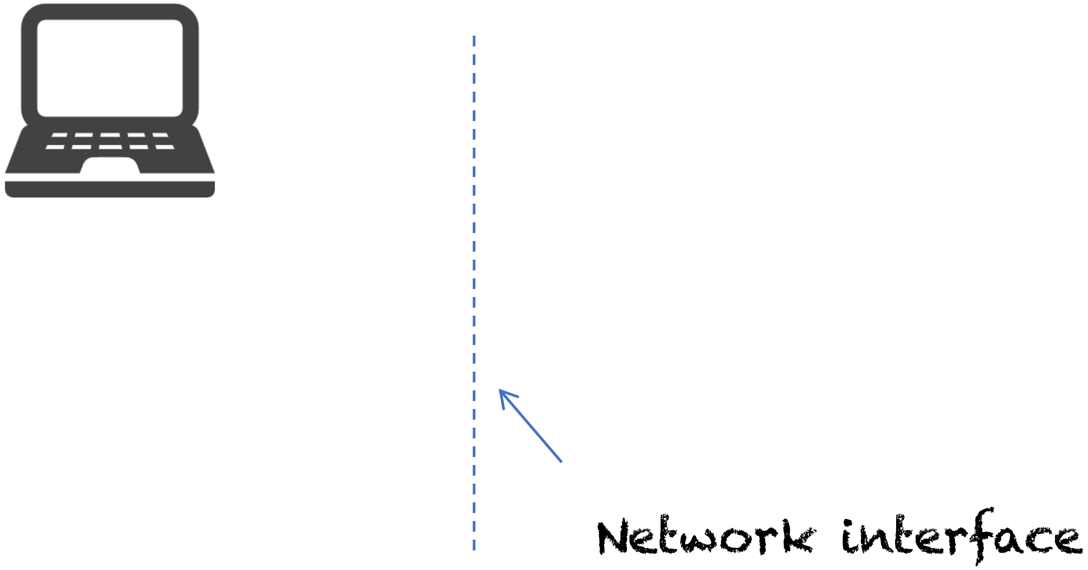
Port Addressing

- To address the application where the data is going to.

Your Computer

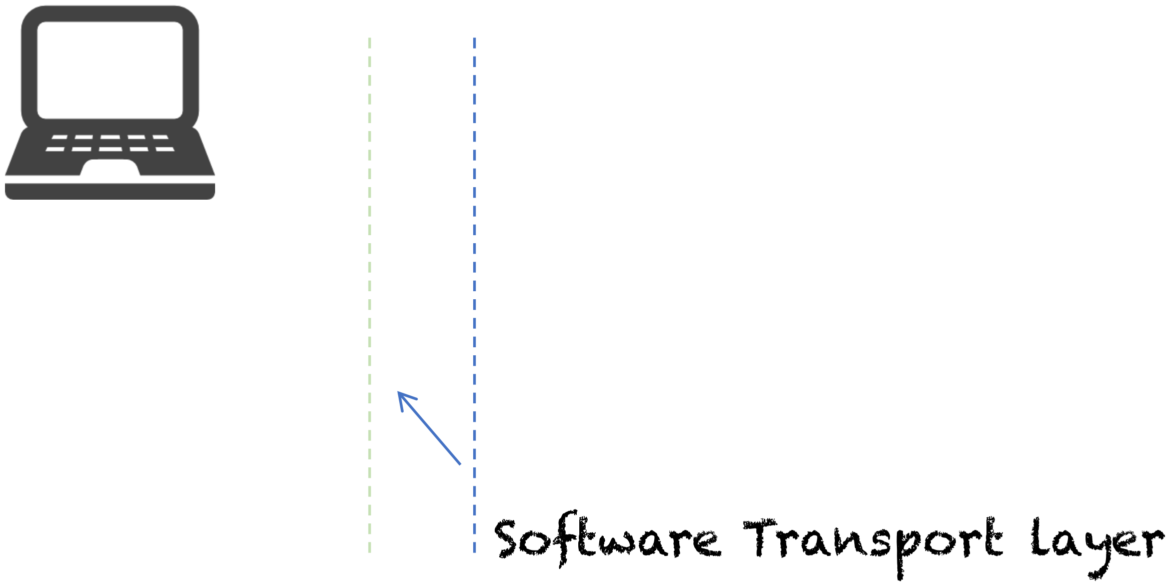
Port Addressing

- To address the application where the data is going to.



Port Addressing

- To address the application where the data is going to.



Port Addressing

- To address the application where the data is going to.

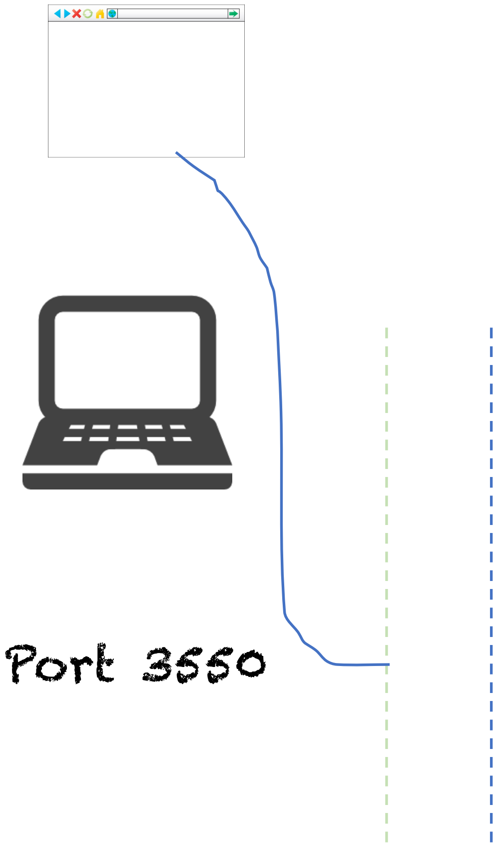
You open an application that requires network access!

Port Addressing

- To address the application where the data is going to.

Port Addressing

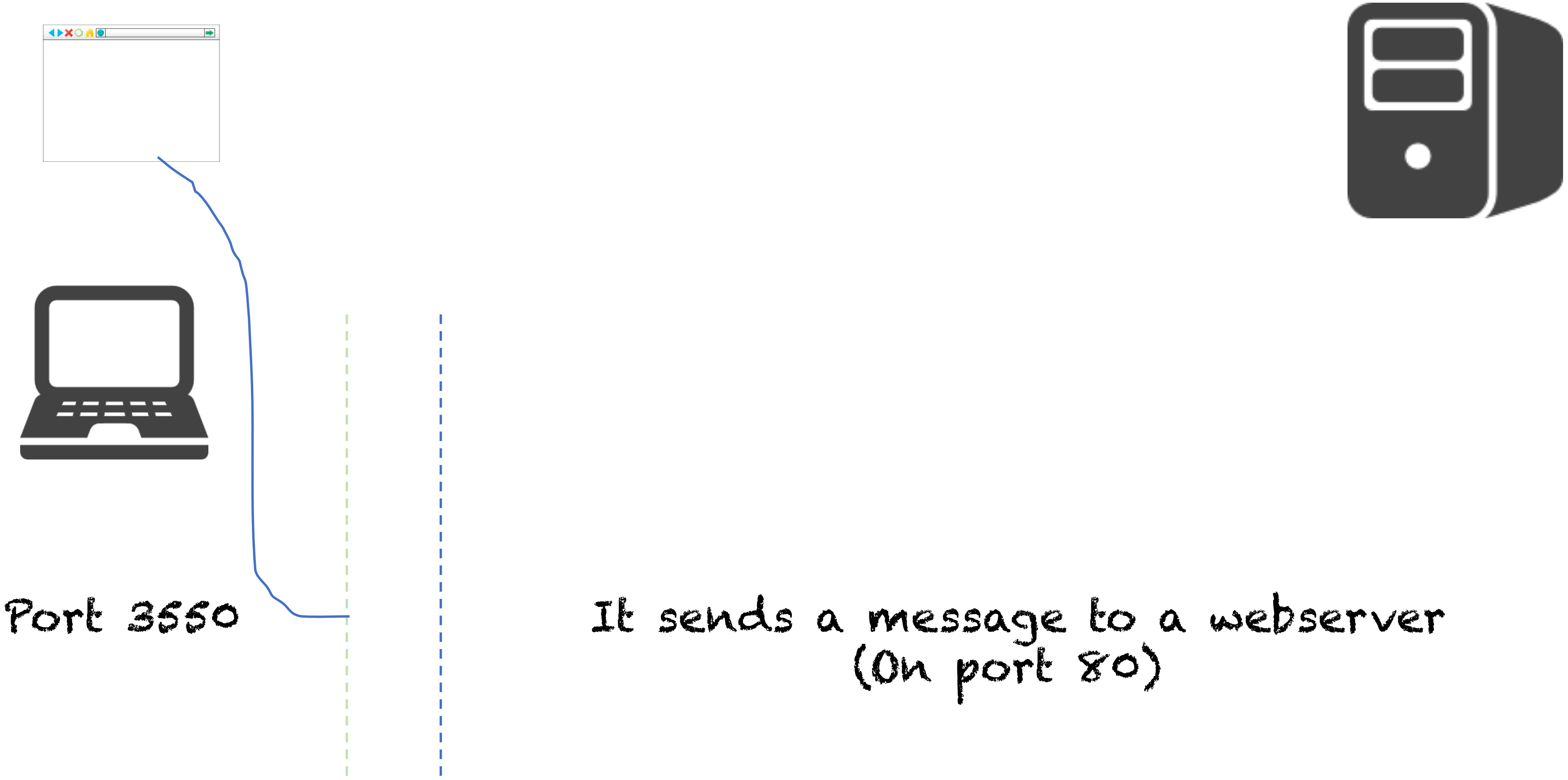
- To address the application where the data is going to.



It chooses a network port to use!
(programed or user defined)

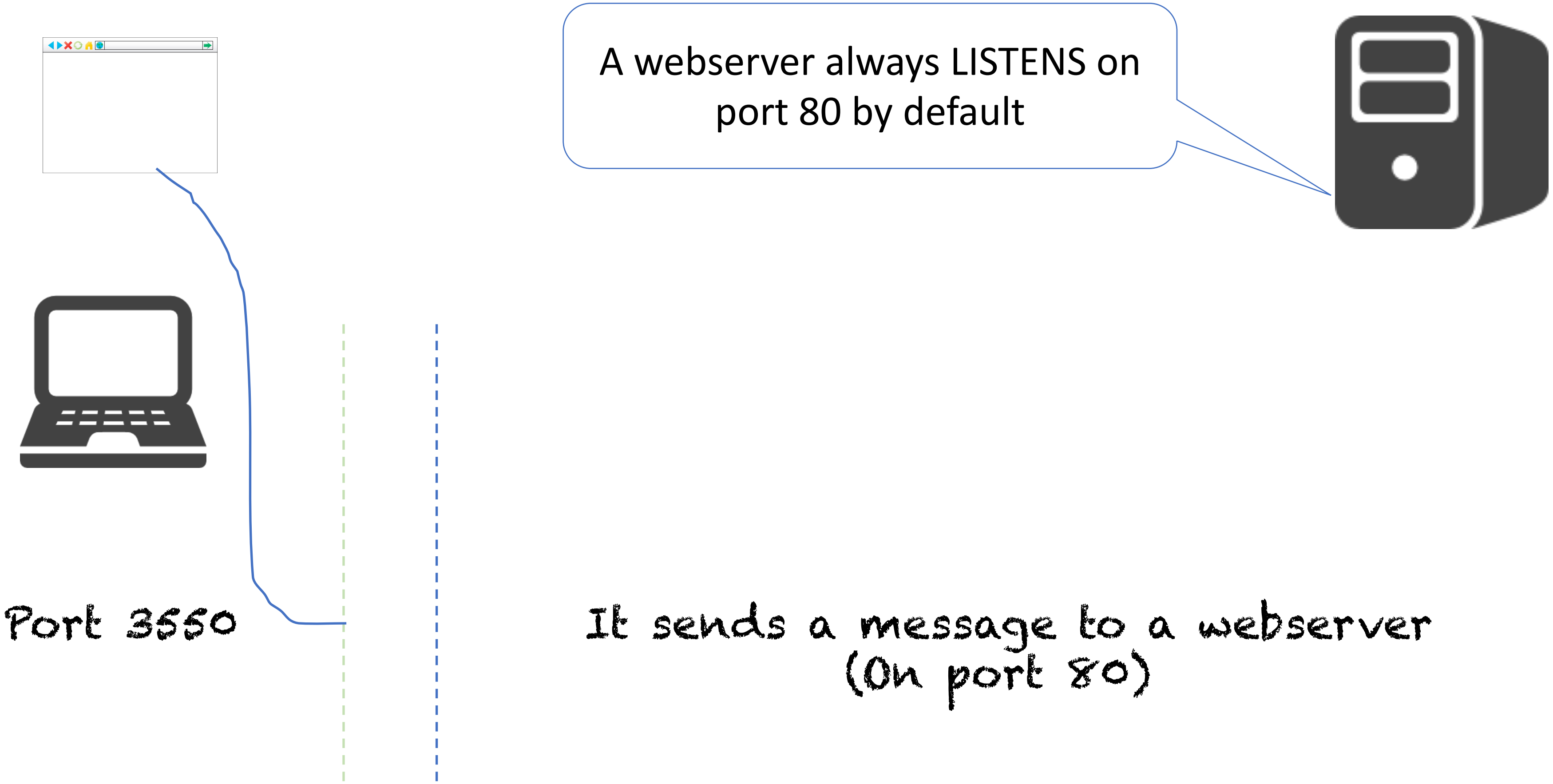
Port Addressing

- To address the application where the data is going to.



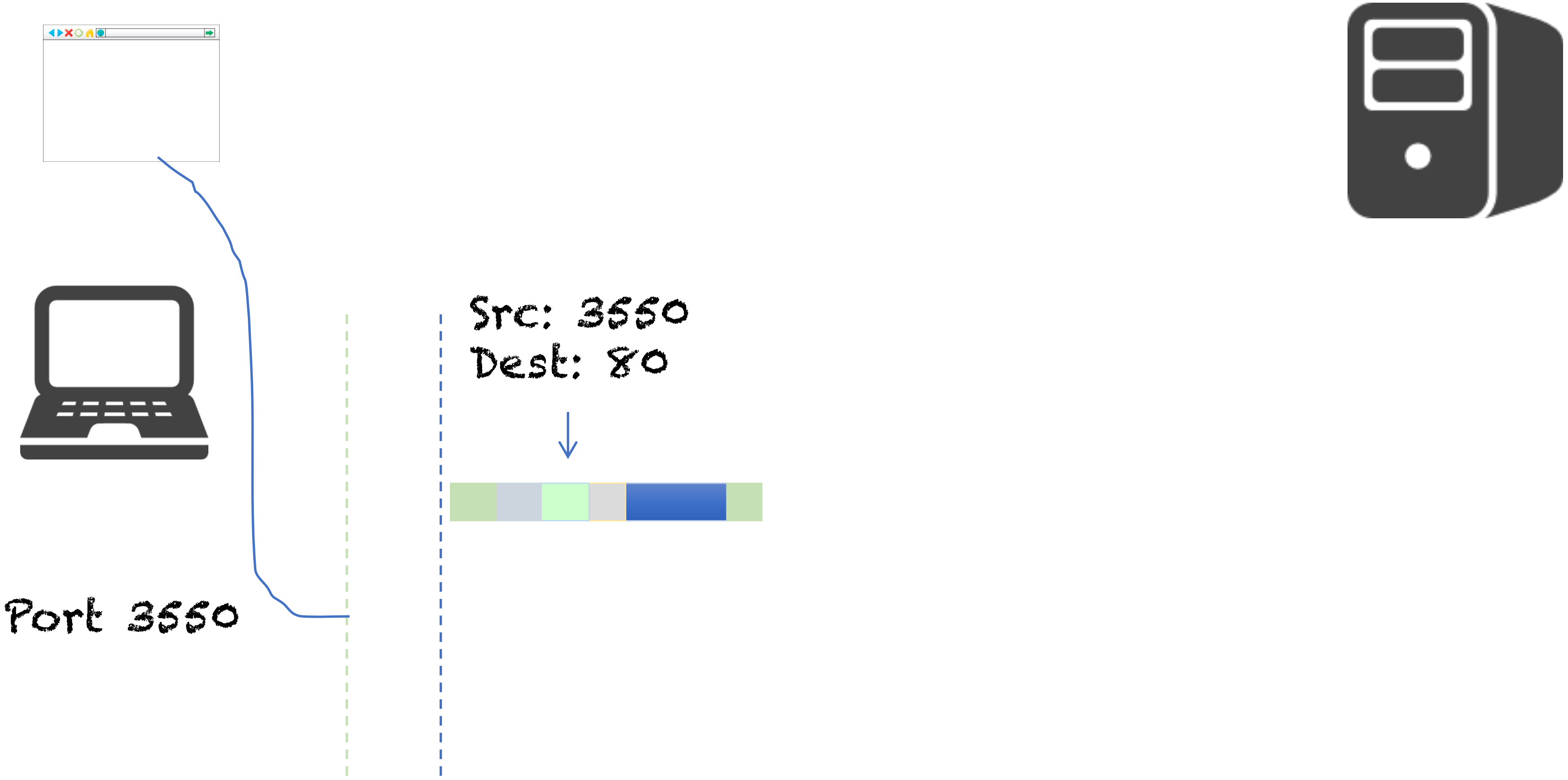
Port Addressing

- To address the application where the data is going to.



Port Addressing

- To address the application where the data is going to.

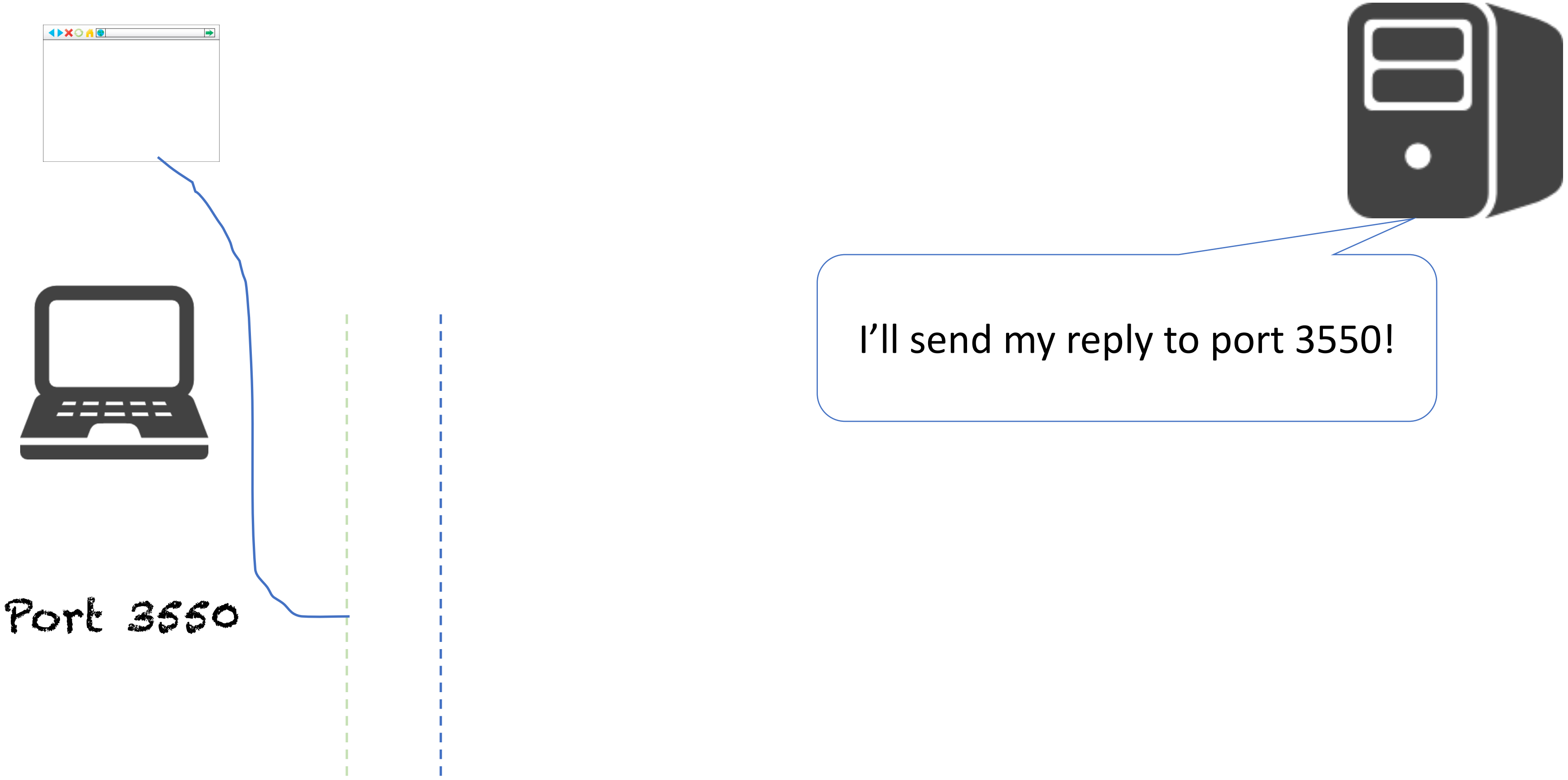


Port Addressing

- To address the application where the data is going to.

Port Addressing

- To address the application where the data is going to.

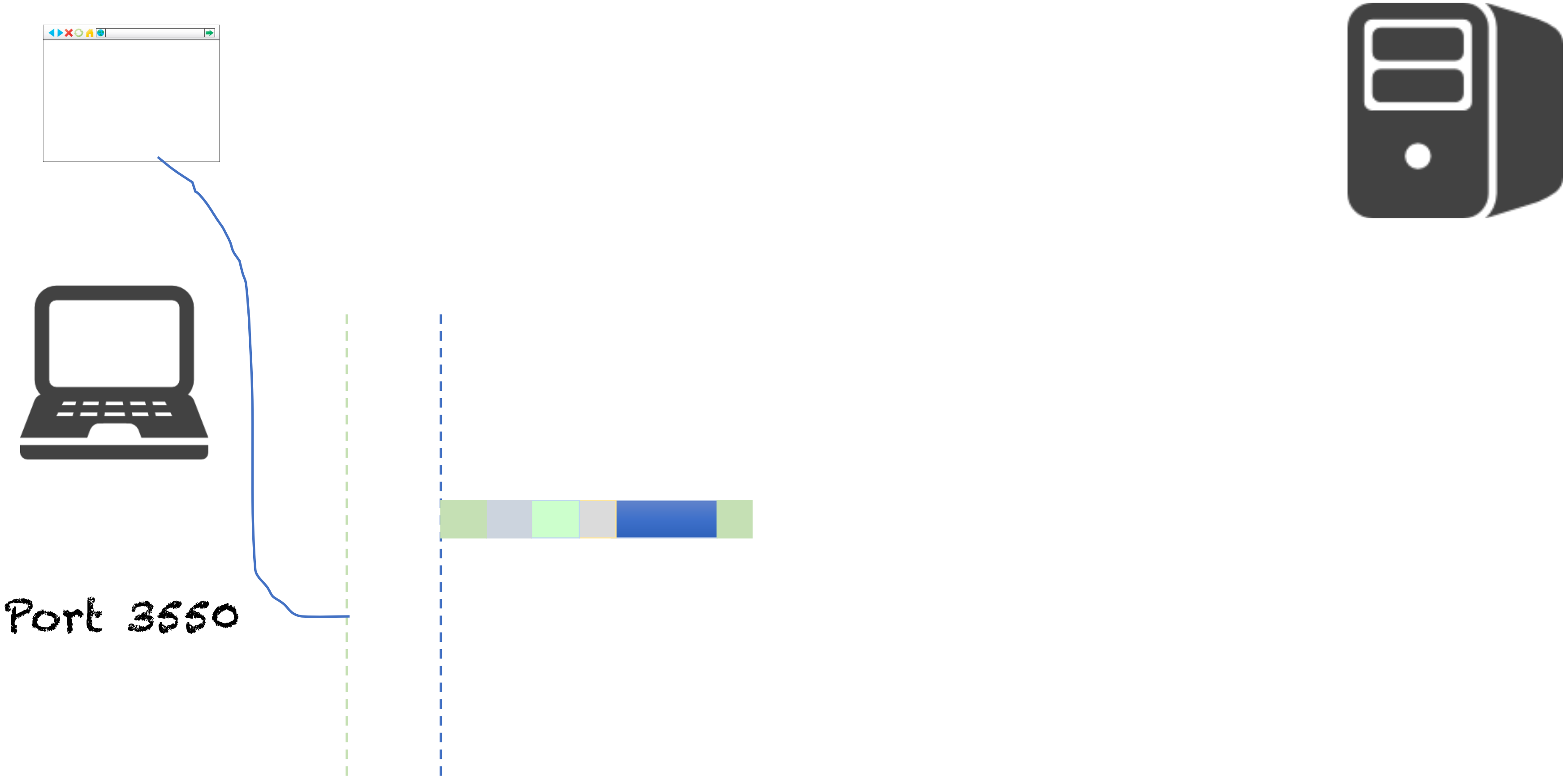


Port Addressing

- To address the application where the data is going to.

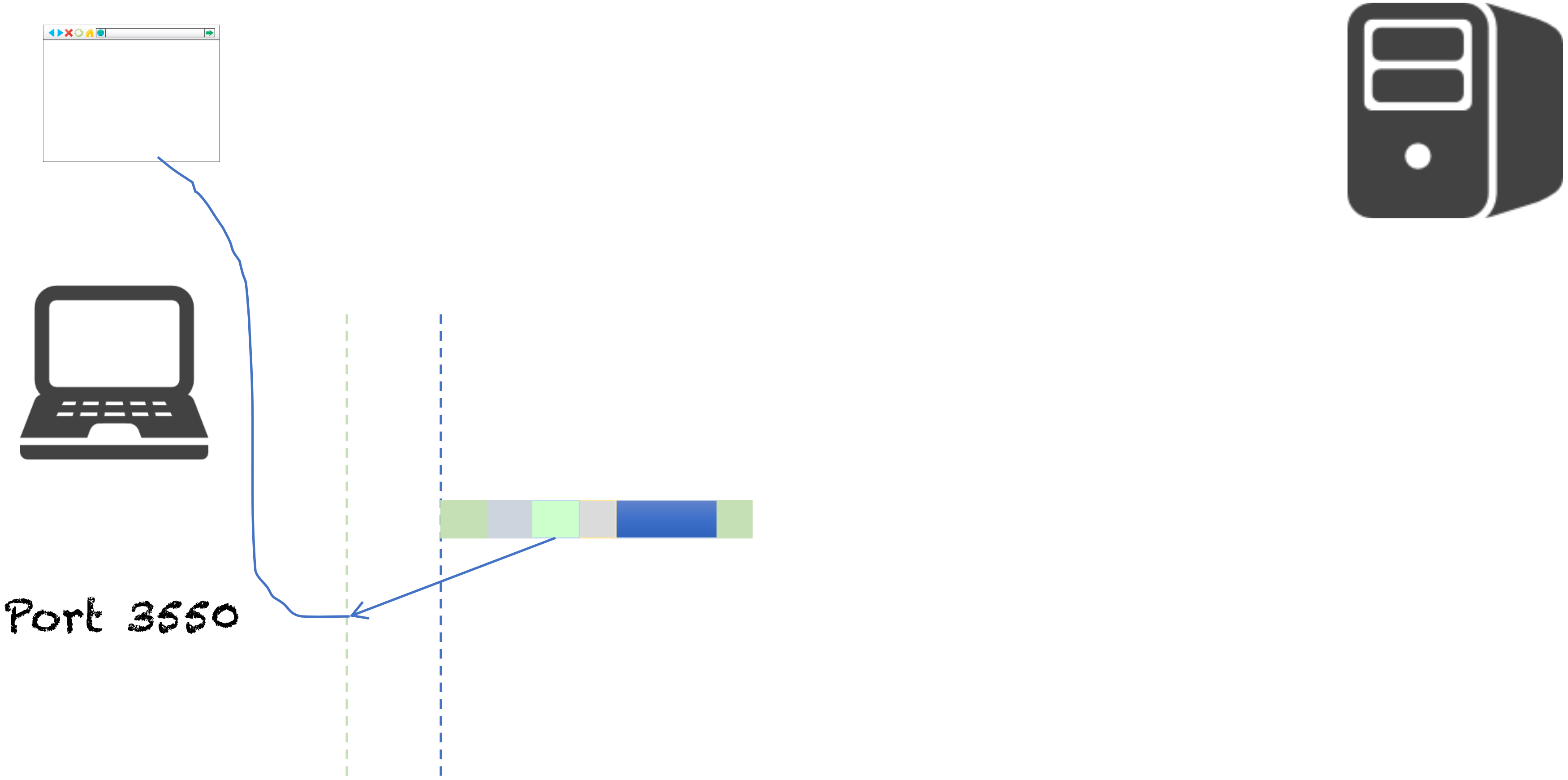
Port Addressing

- To address the application where the data is going to.



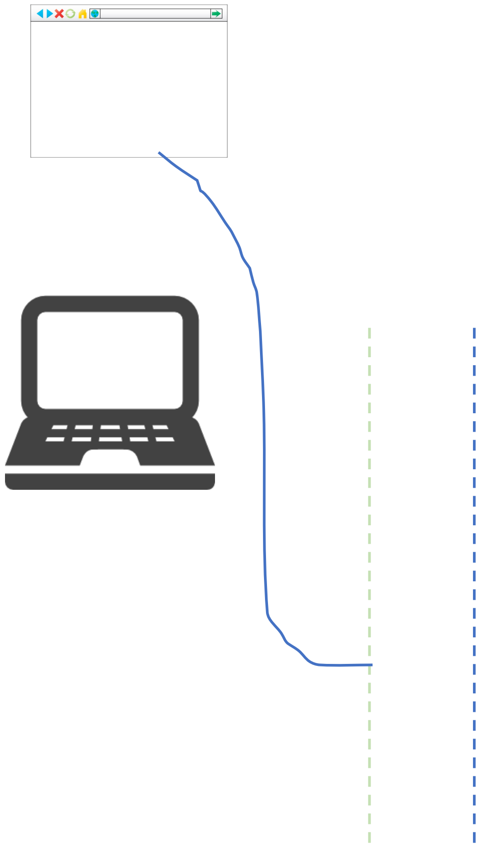
Port Addressing

- To address the application where the data is going to.



Port Addressing

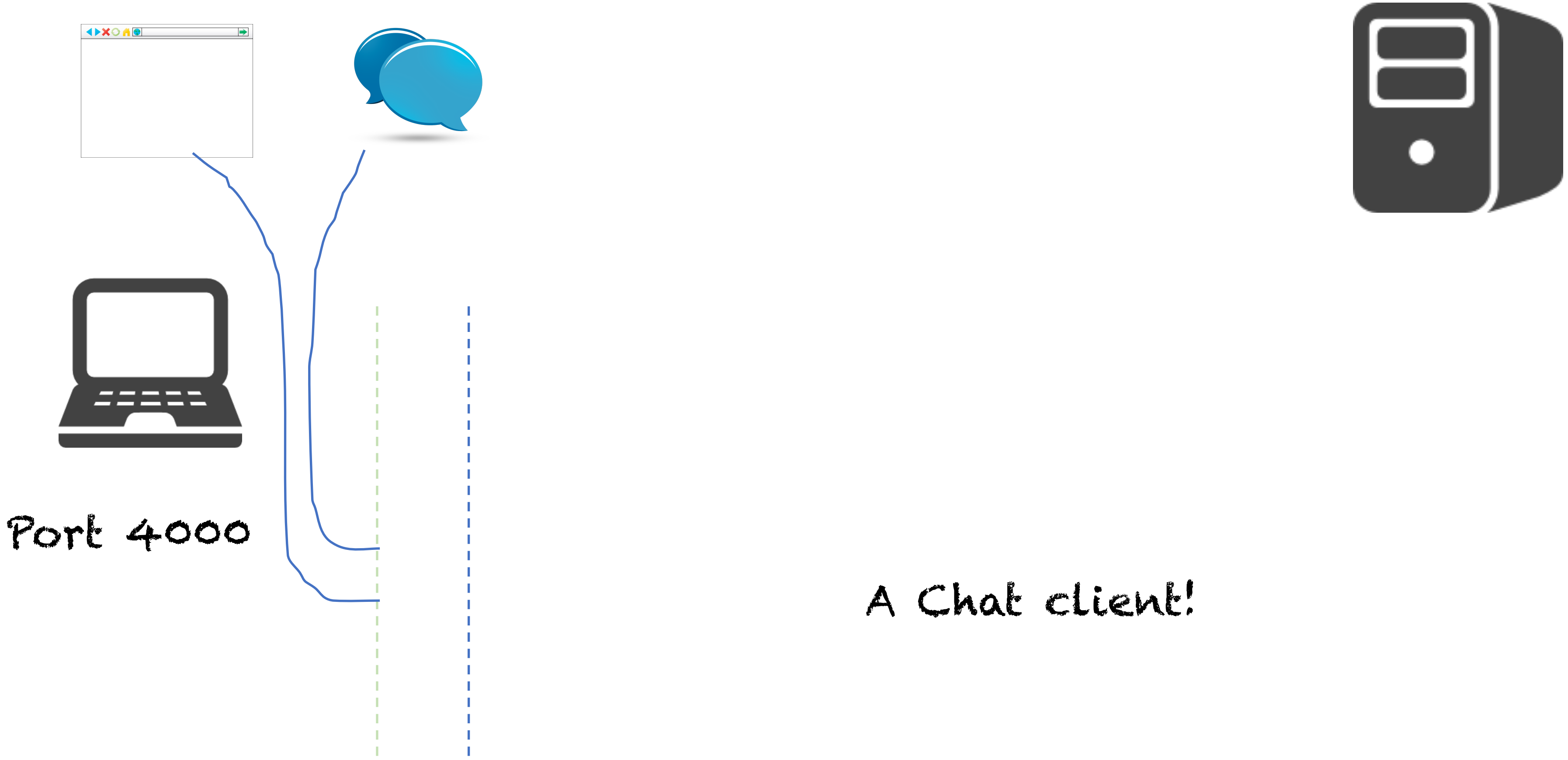
- To address the application where the data is going to.



This is necessary as we have more than
One network app at any time.

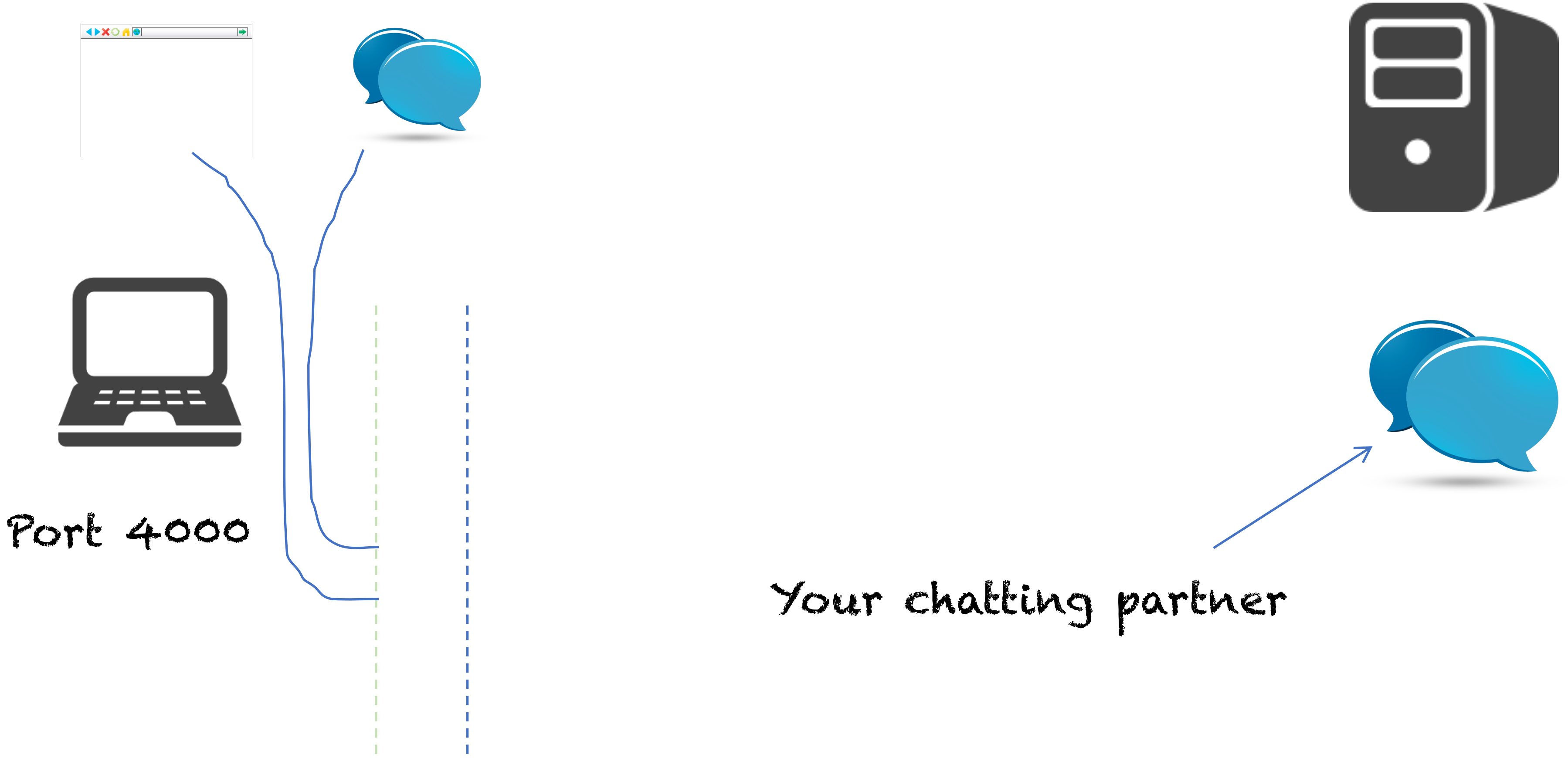
Port Addressing

- To address the application where the data is going to.



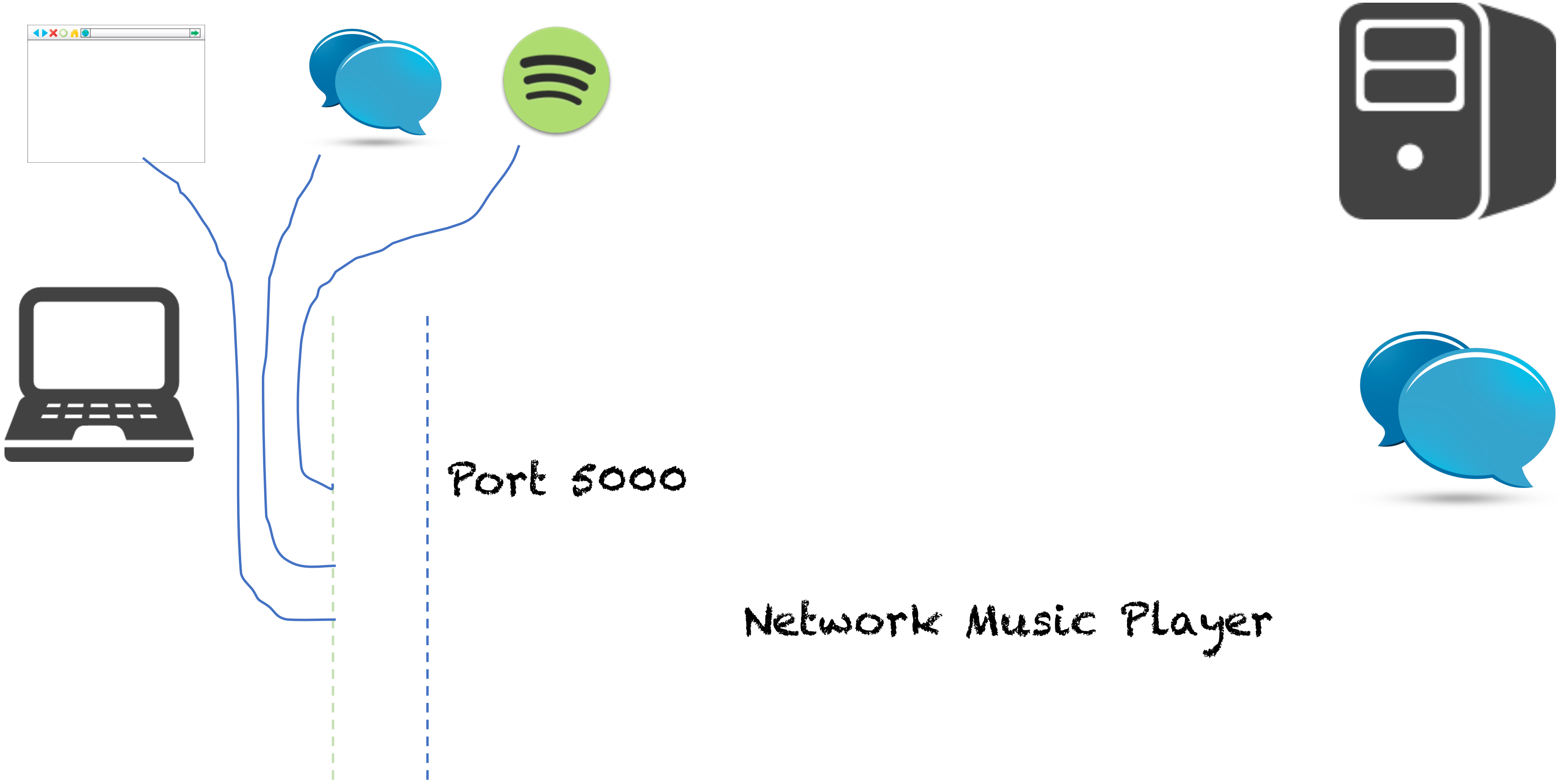
Port Addressing

- To address the application where the data is going to.



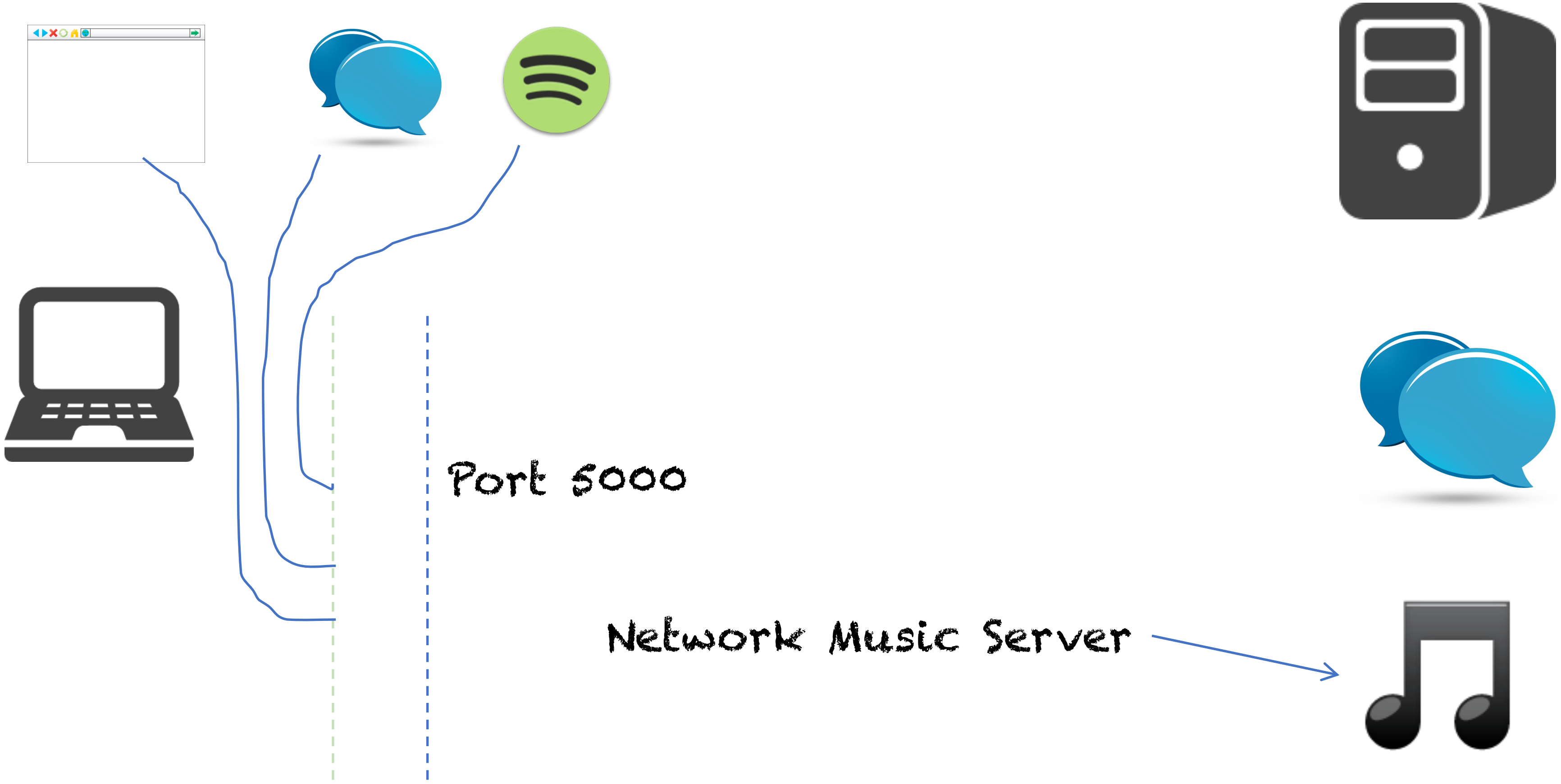
Port Addressing

- To address the application where the data is going to.



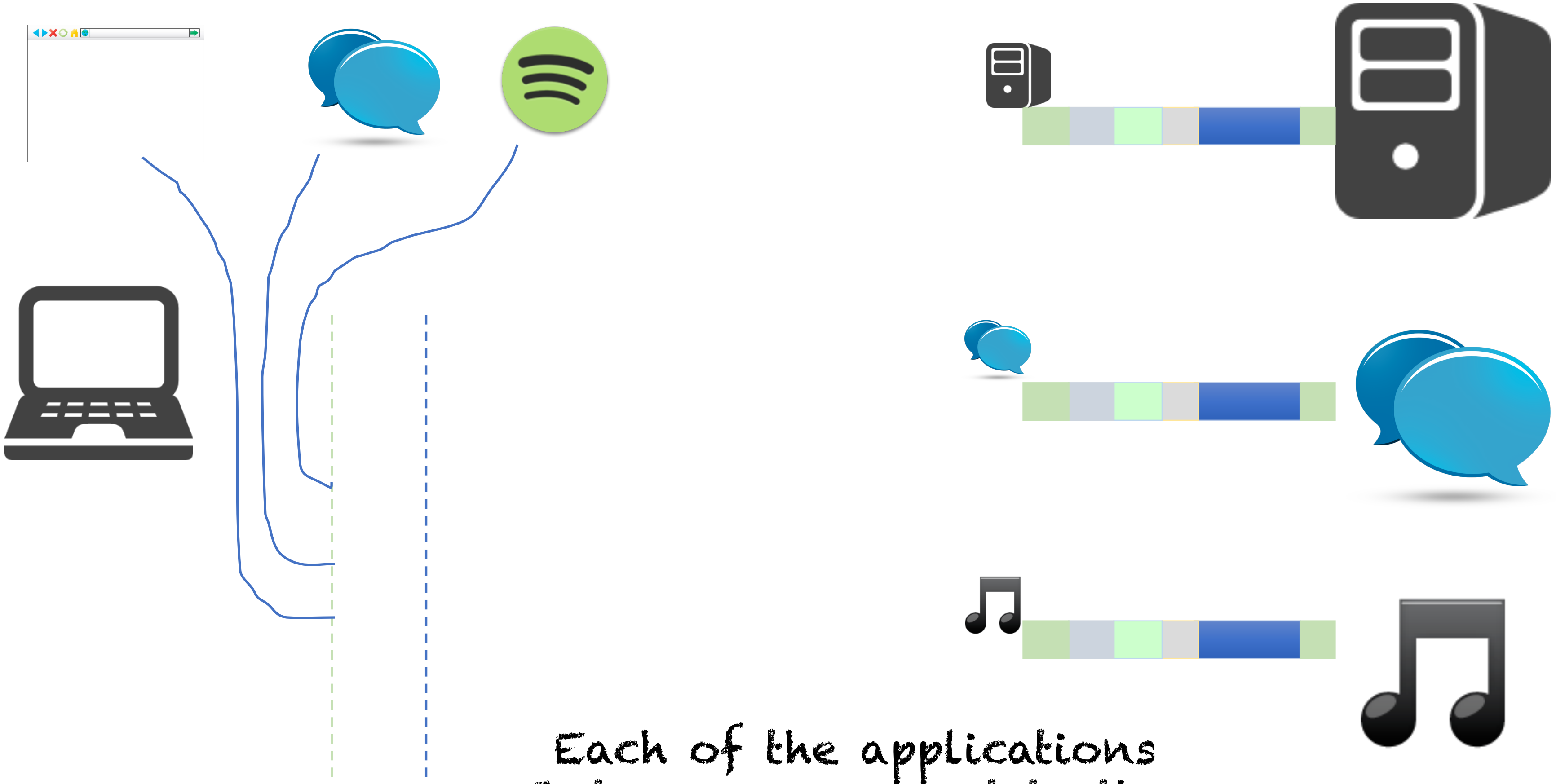
Port Addressing

- To address the application where the data is going to.



Port Addressing

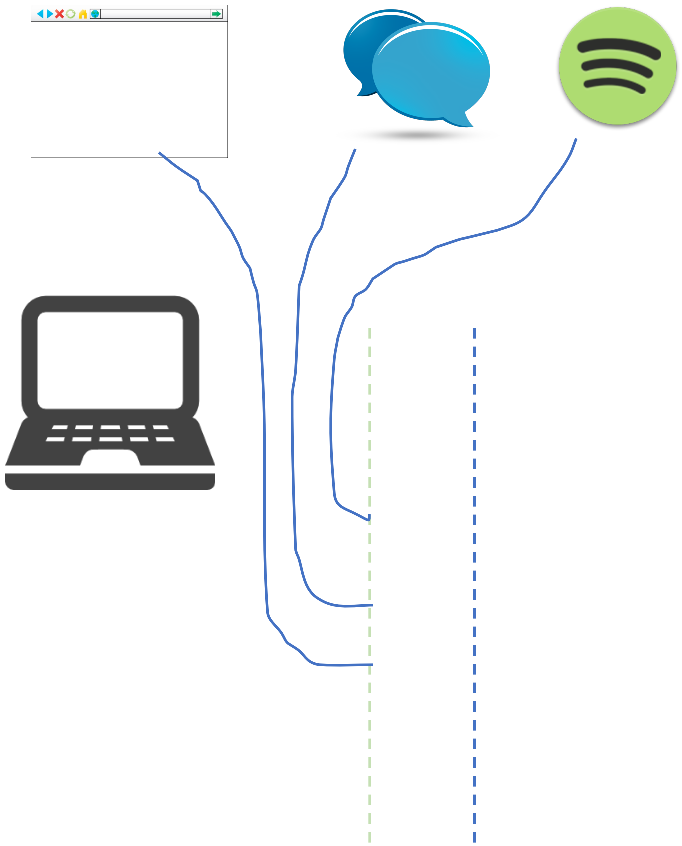
- To address the application where the data is going to.



Each of the applications
Get a message sent to them

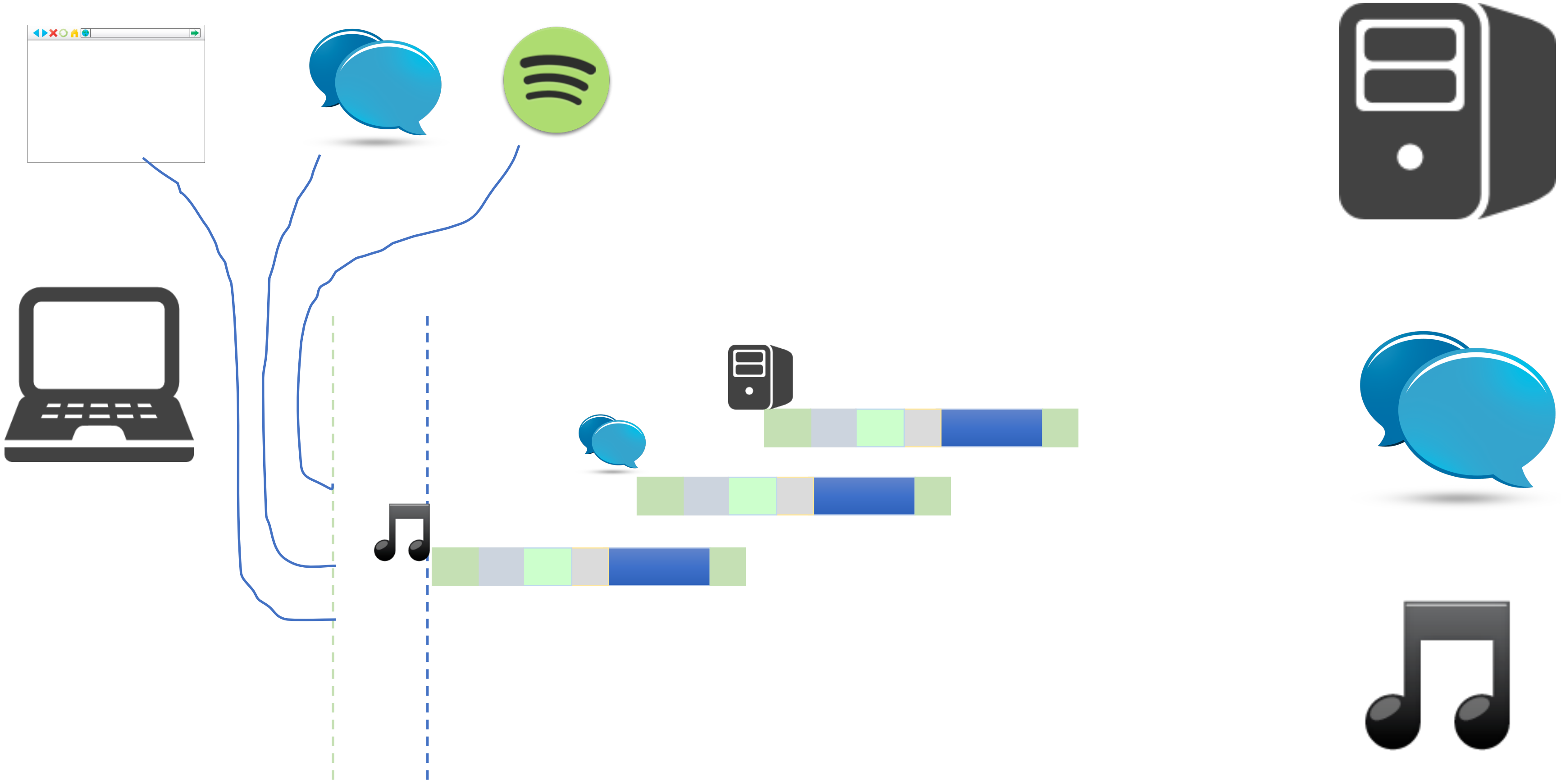
Port Addressing

- To address the application where the data is going to.



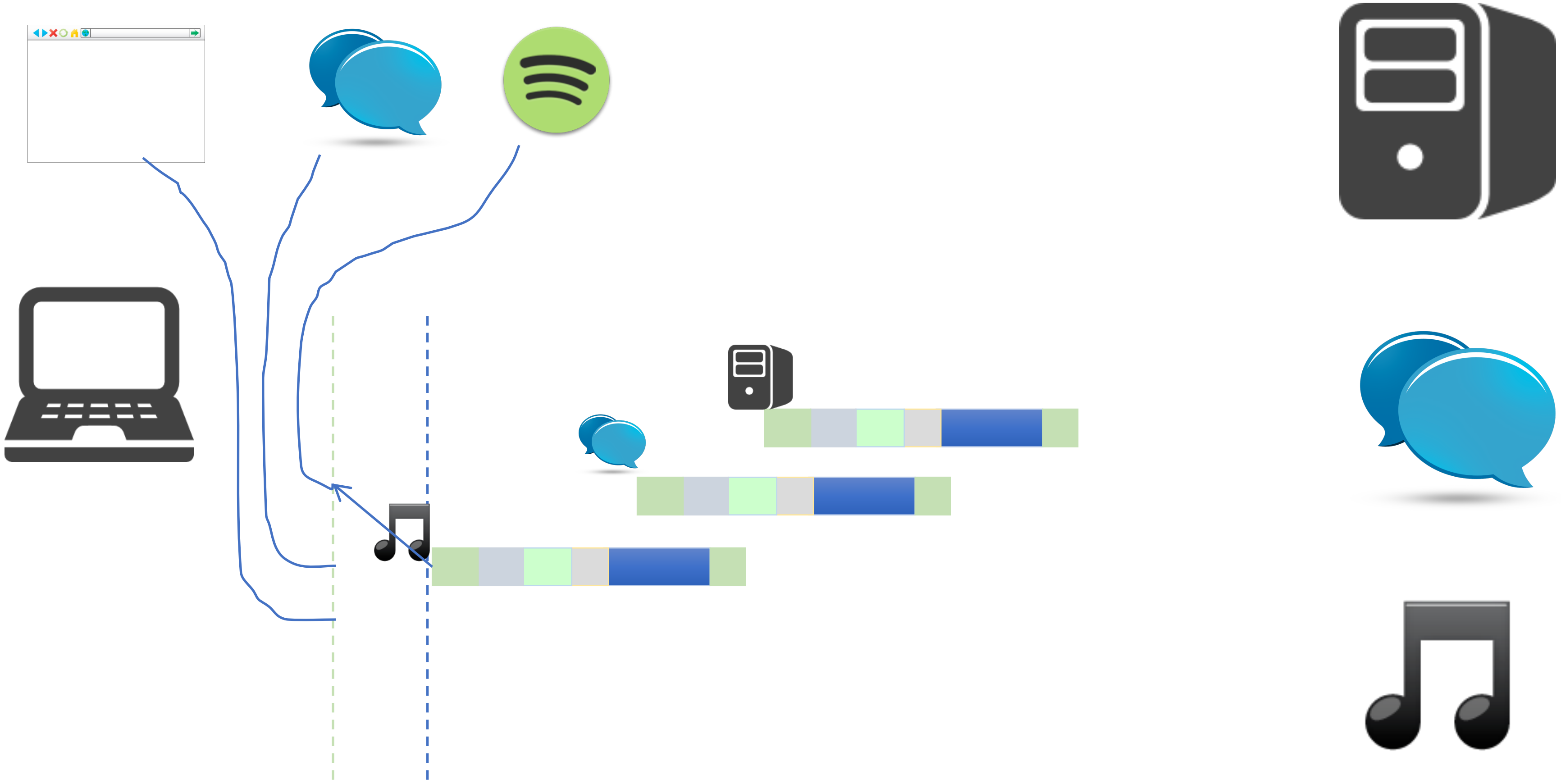
Port Addressing

- To address the application where the data is going to.



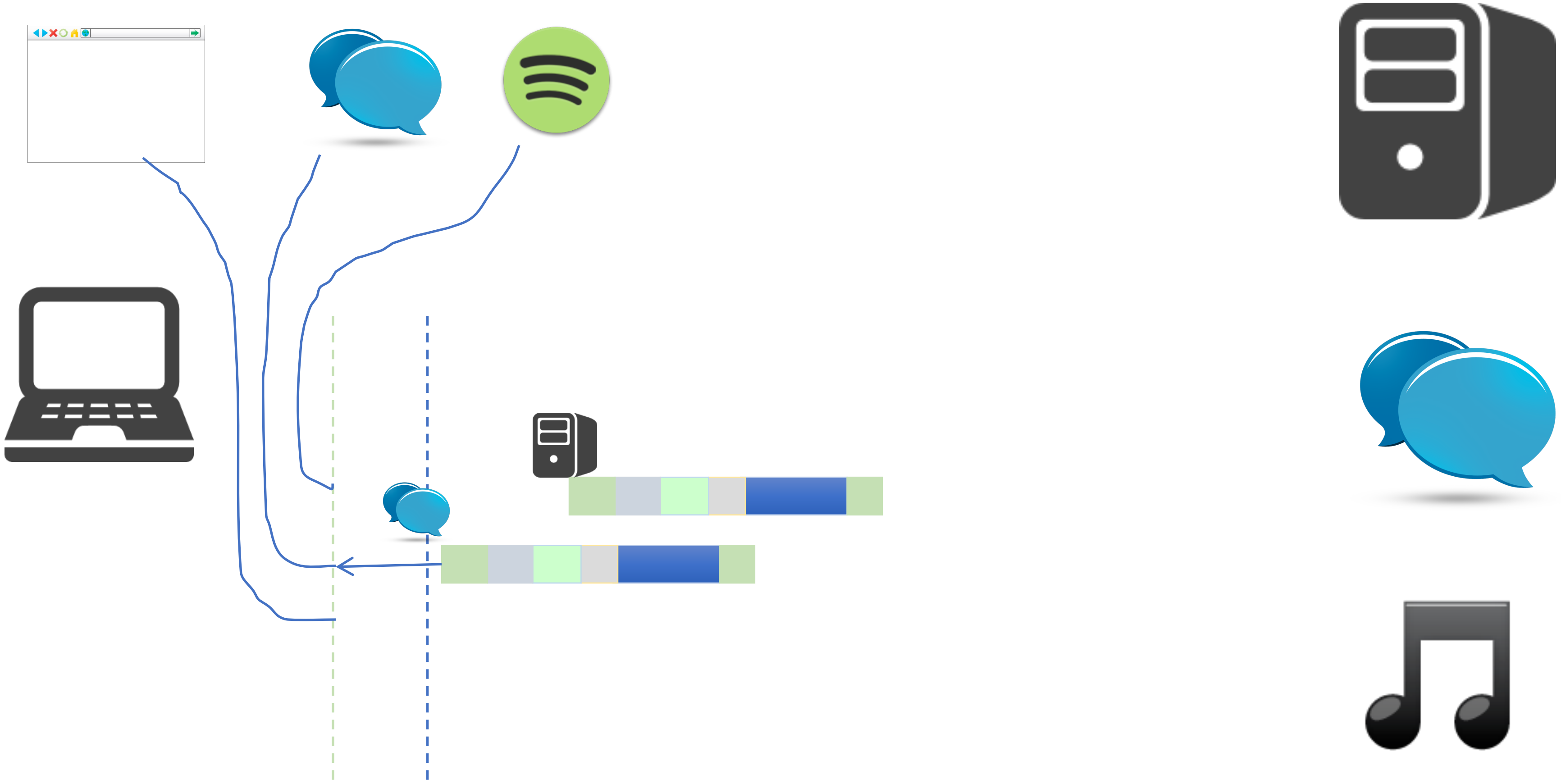
Port Addressing

- To address the application where the data is going to.



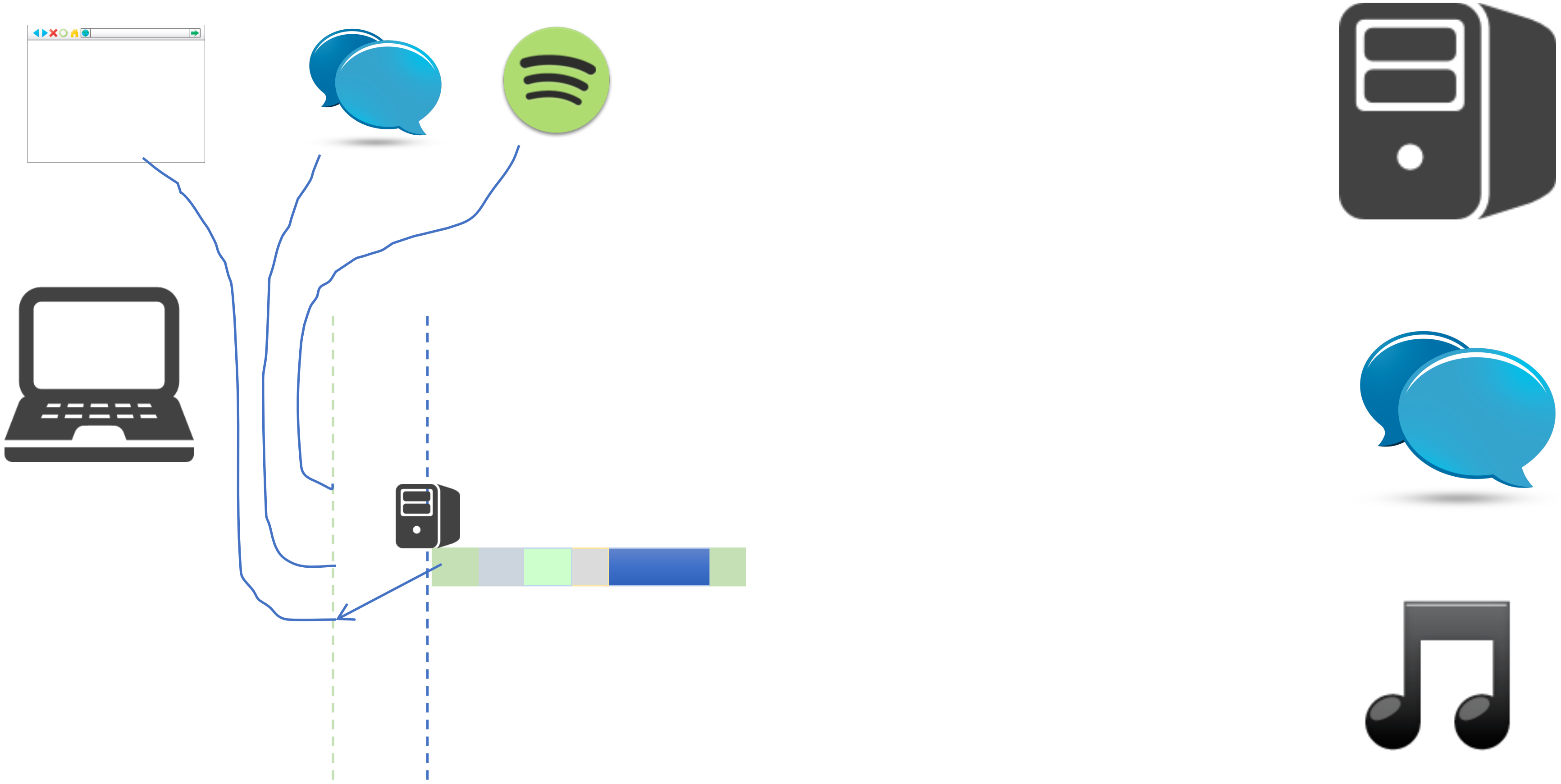
Port Addressing

- To address the application where the data is going to.



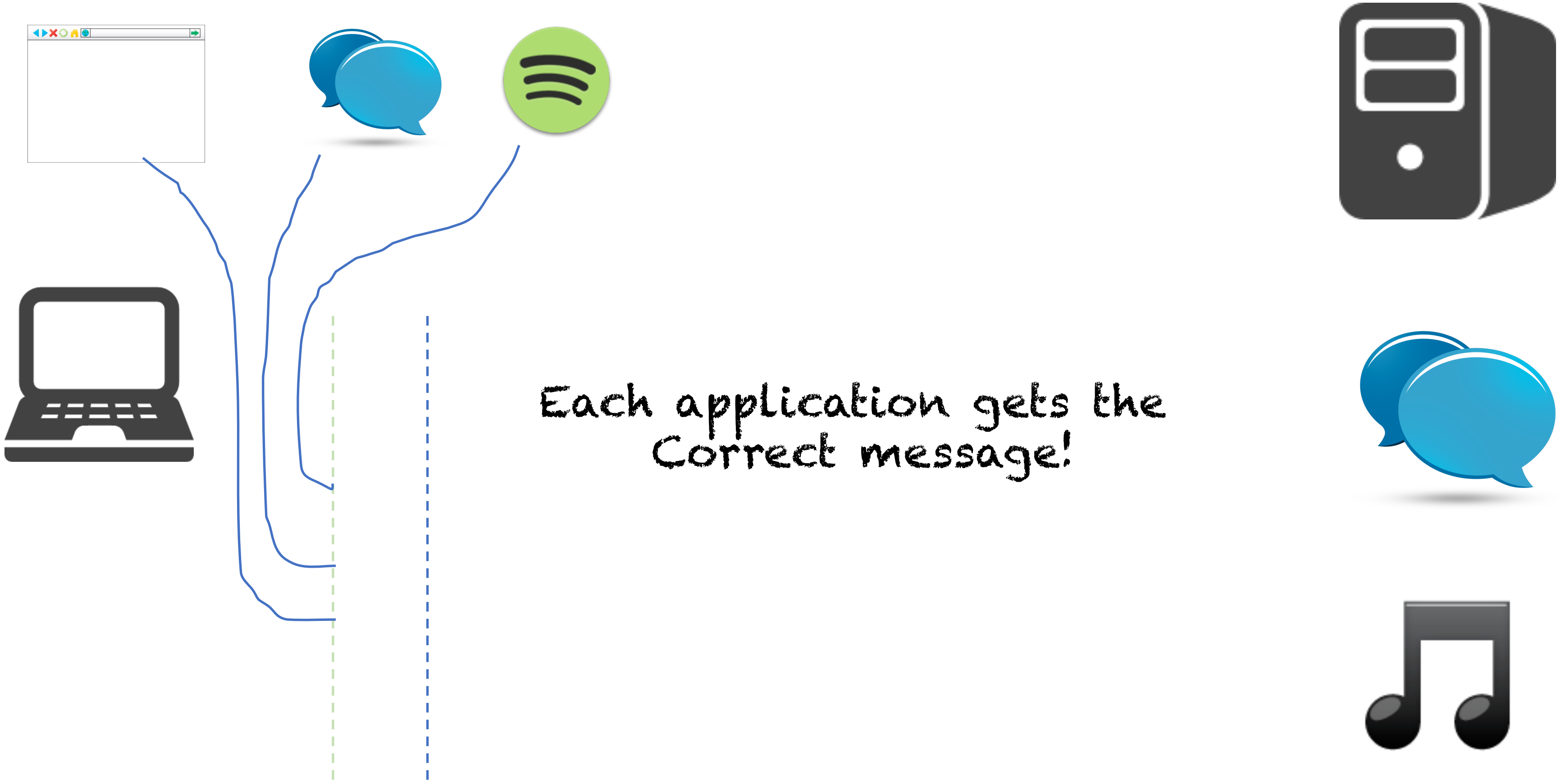
Port Addressing

- To address the application where the data is going to.



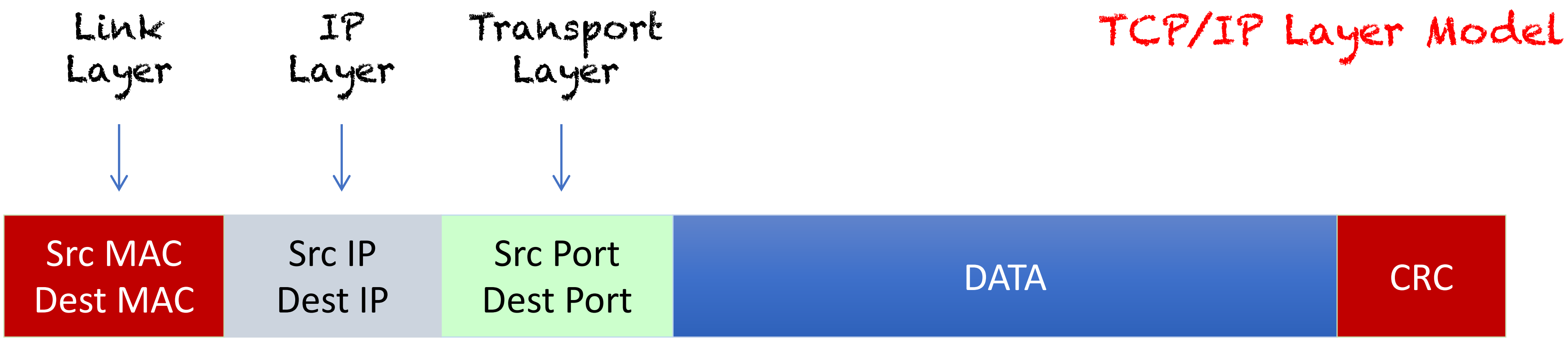
Port Addressing

- To address the application where the data is going to.



Addressing in headers

- The addresses are split over different layers



Transport Protocols in the Internet

	UDP - User Datagram Protocol	TCP - Transmission Control Protocol
Characteristics	• End-To-End Transport • Unreliable	• End-To-End Transport • Reliable
Protocol Details	• Simple • Connectionless • Datagram oriented	• Complex • Connection-oriented • Stream oriented

Use cases	• Unicast and Multicast
	• Useful only for few applications multimedia applications
	  • E.g., Live video streaming
	• Used a lot for network services
	  • E.g., Routing (RIP), Naming (DNS)
		• Unicast only
		• Used for most Internet applications:
		  • E.g., Web (HTTP), Email (SMTP), File Transfer (FTP)

Port Numbers

- UDP and TCP use a tuple **<IP address, port number>**
  - port numbers are used to identify applications
  - There are 65,535 ports per host

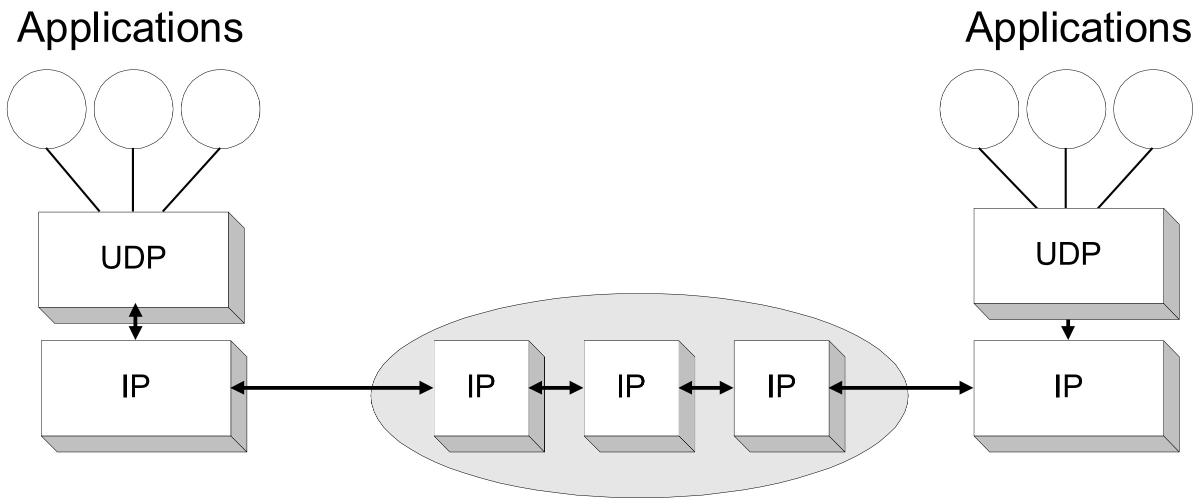
- There are many standard port numbers always used for particular applications (mostly)
  - 80 – HTTP service (web servers)
  - 443 – HTTPS (secure HTTP)
  - 25 – SMTP (Simple Mail Transfer Protocol)
  - 3306 – MySQL Database

- Many other port numbers are free, and allow applications to set their own port numbers

UDP - User Datagram Protocol

- UDP supports **unreliable end-to-end** transmissions of datagrams

- UDP merely extends the host-to-to-host delivery service of IP datagram to an application-to-application service



UDP Format

IP Header	UDP Header	Data
	8 Bytes	

Source Port Number	Destination Port Number
Message Length	Checksum

0 15 31

- **Port numbers** identify sending and receiving applications (processes). Maximum port number is $2^{16}-1 = 65{,}535$

- **Message Length** is at least 8 bytes (I.e., Data field can be empty) and at most 65,535

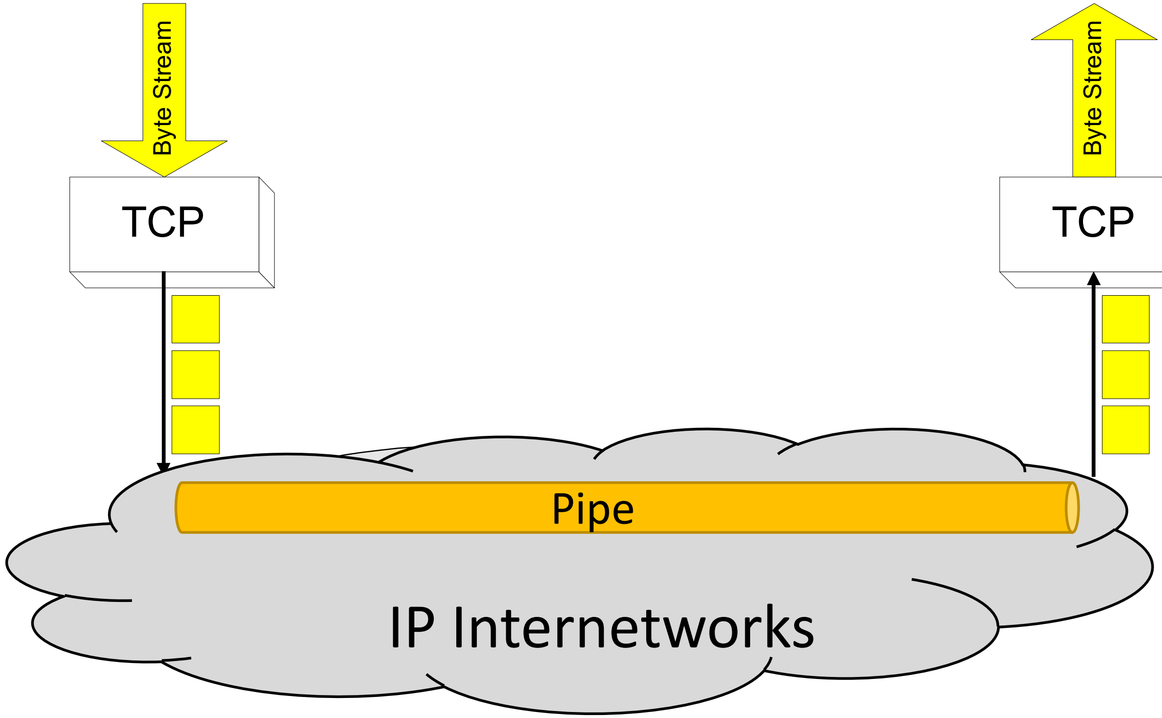
- **Checksum** is for header (of UDP and some of the IP header fields)

Ollscoil Chathair Bhaile Átha Cliath Dublin City University

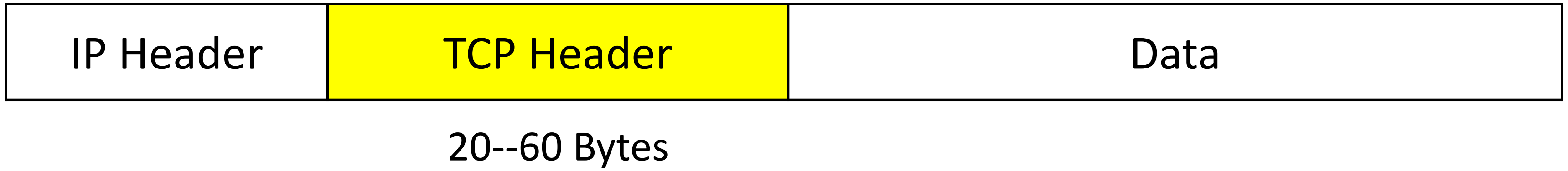
TCP Overview

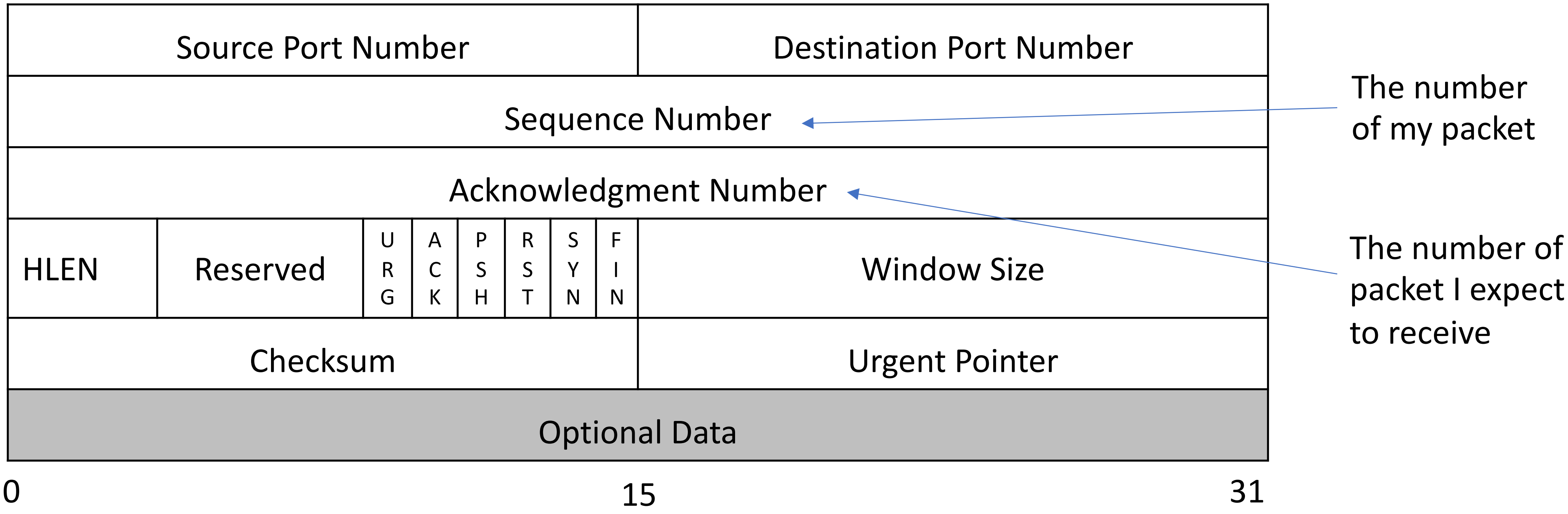
TCP = Transmission Control Protocol

- Connection-oriented protocol
- Provides a reliable, end-to-end byte stream over an unreliable internetwork



TCP Format


IP Header
TCP Header
Data
20--60 Bytes


Source Port Number
Destination Port Number
Sequence Number
Acknowledgment Number
HLEN
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window Size
Checksum
Urgent Pointer
Optional Data
0
15
31
The number of my packet
The number of packet I expect to receive

Challenges for Reliable Transfer

- Over a perfectly reliable channel
  - All the data arrives in order, just as it was sent
  - Simple: Sender sends data, and Receiver receives data
- Over a channel with bit errors
  - All the data arrives in order, but some bits corrupted
  - We need **checksums** so that the Receiver detects errors and asks for a retransmission
- Over a lossy and congested channels
  - Some data are missing, and others arrive with delay
  - Receiver needs to detect a loss or a disorder in packets
  - We need **sequence numbers** to detect missing data and put data back in order

TCP Uses 3 Mechanisms to Achieve a Reliable Transfer

1. Connection: establish and release a connection
2. Re-transmission: retransmit lost or corrupted data
3. Window and Congestion Control: manage the amount of sent data for an efficient resource usage

Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

Set Up/Close a Connection

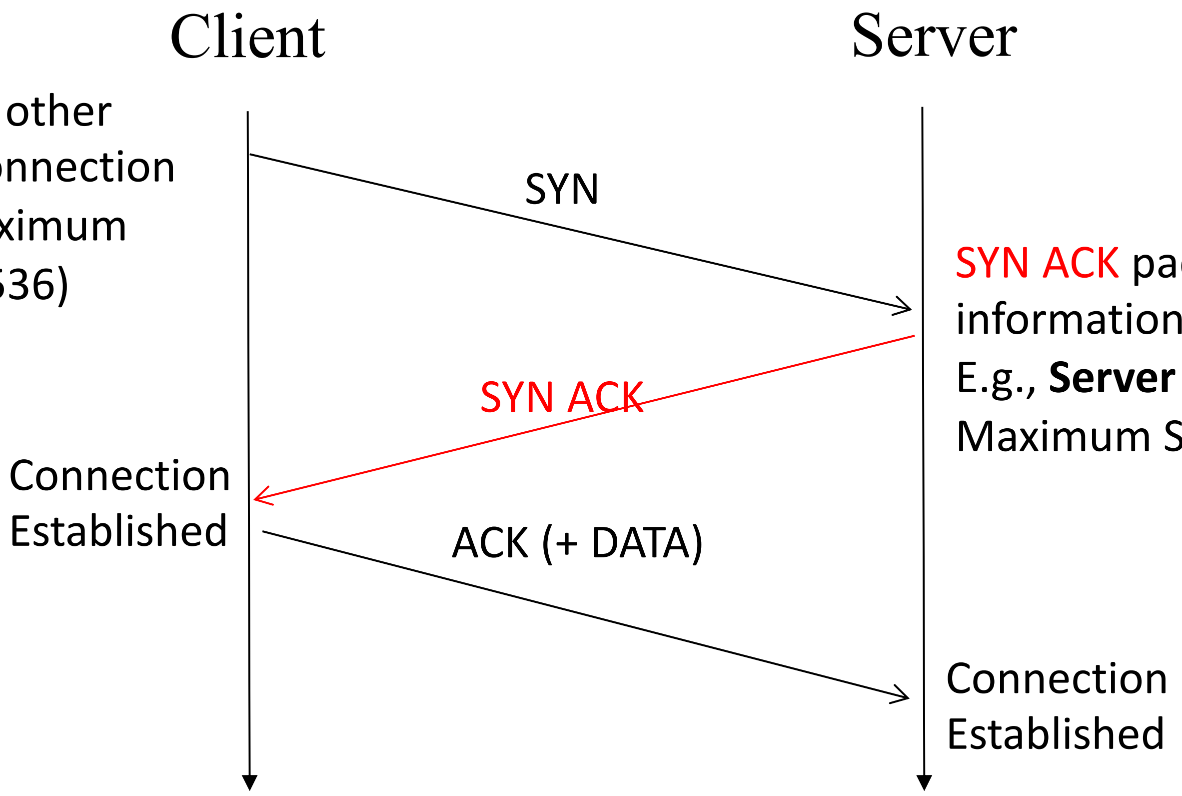
- TCP must first set up a connection with the end point before sending information
  - Make sure the endpoint is accessible/available
  - Make sure the end point has the facilities to handle the data
- TCP must close the connection after finishing the transmission

- TCP uses:
  - 3-Way Handshake to set-up the connection
  - 4-Way Handwave to close the connection

Set Up a Connection: 3-Way Handshake

- We have a Client/Server relationship

SYN packets also contain other information about the connection E.g., Window size, or Maximum Segment Size (default = 536)

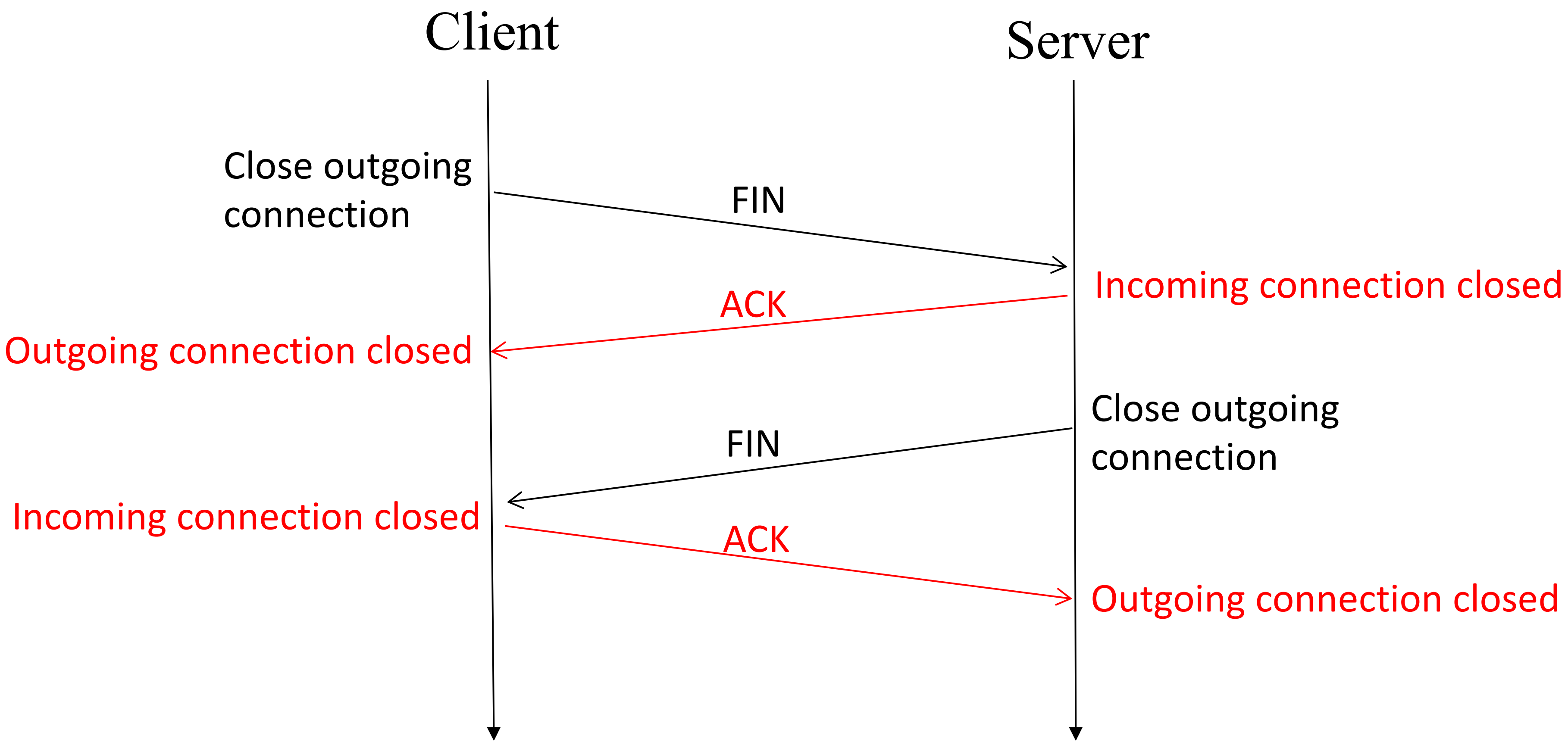


SYN ACK packet also contain other information about the connection E.g., **Server window size**, or Maximum Segment Size

Client and Server are now set up for data transfer

Close a Connection: 4-Way Handwave

- Each of the Client and Server have to close both their outgoing and ingoing connections

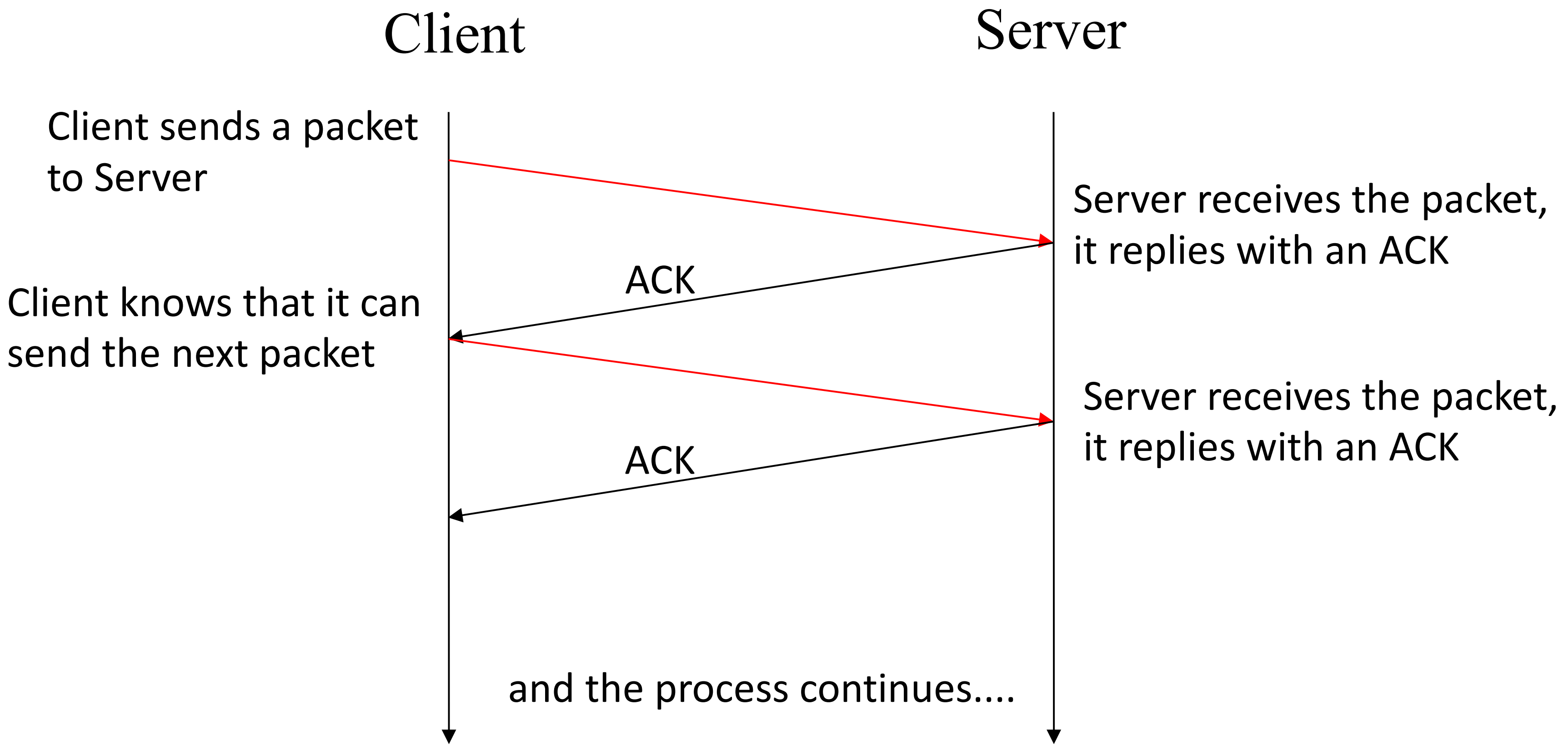


Sequence Numbers

- Once a connection has been established, each subsequent packet contains a sequence number
- This sequence number is used to reorder data in the correct order at the end point. The sequence number is the byte count of the transmission
- It can also be used to determine is packets have been lost over the network
- Received packets are acknowledged with the next expected byte

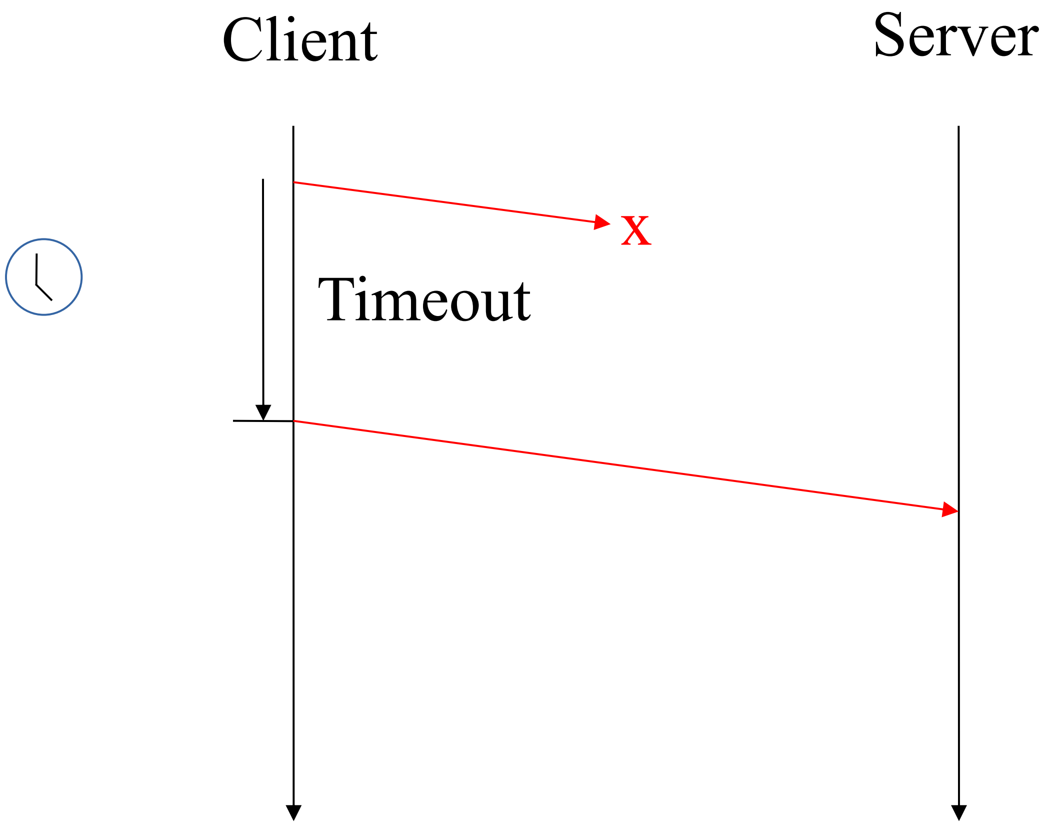
Acknowledging packets

- Acknowledged packets is an important concept for TCP
- It confirms to the Client that his sent data have been received



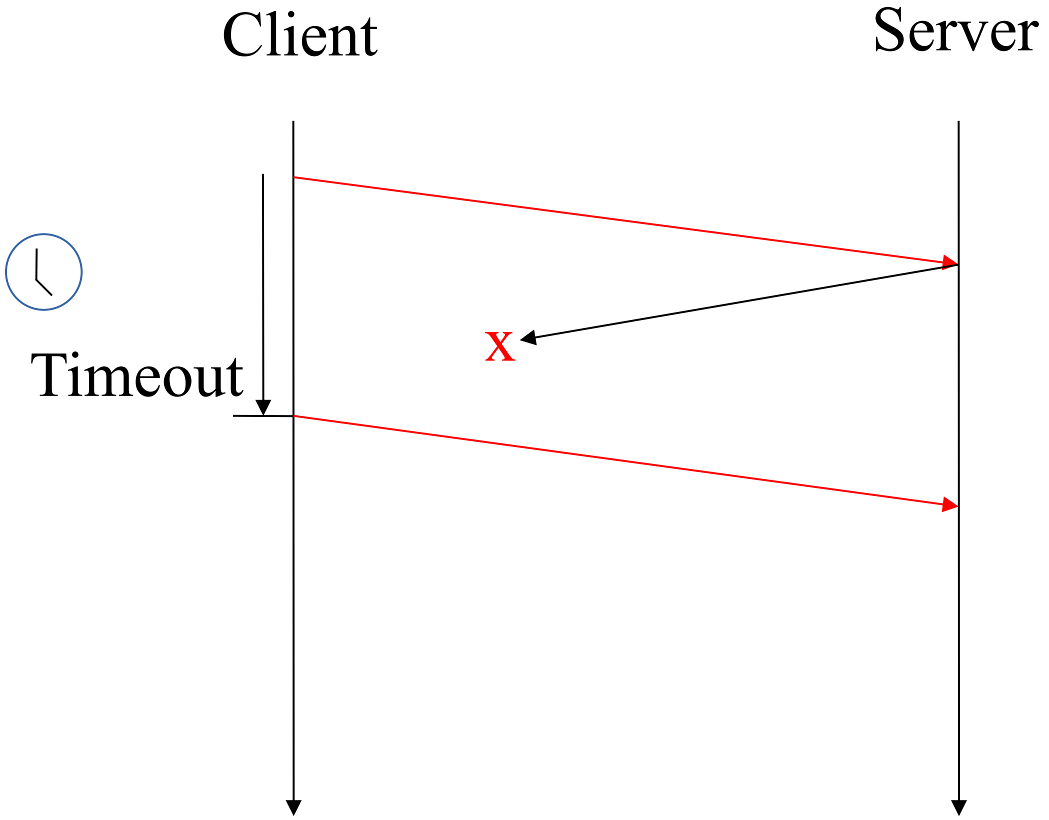
Packet Timeout

- What if the Packet is not received by the Server.
- Client sends a packet, but it is lost



Packet Timeout

- What if the Packet is not received by the Server.
- Server sends an ACK, but the ACK is lost



Acknowledge Schemes

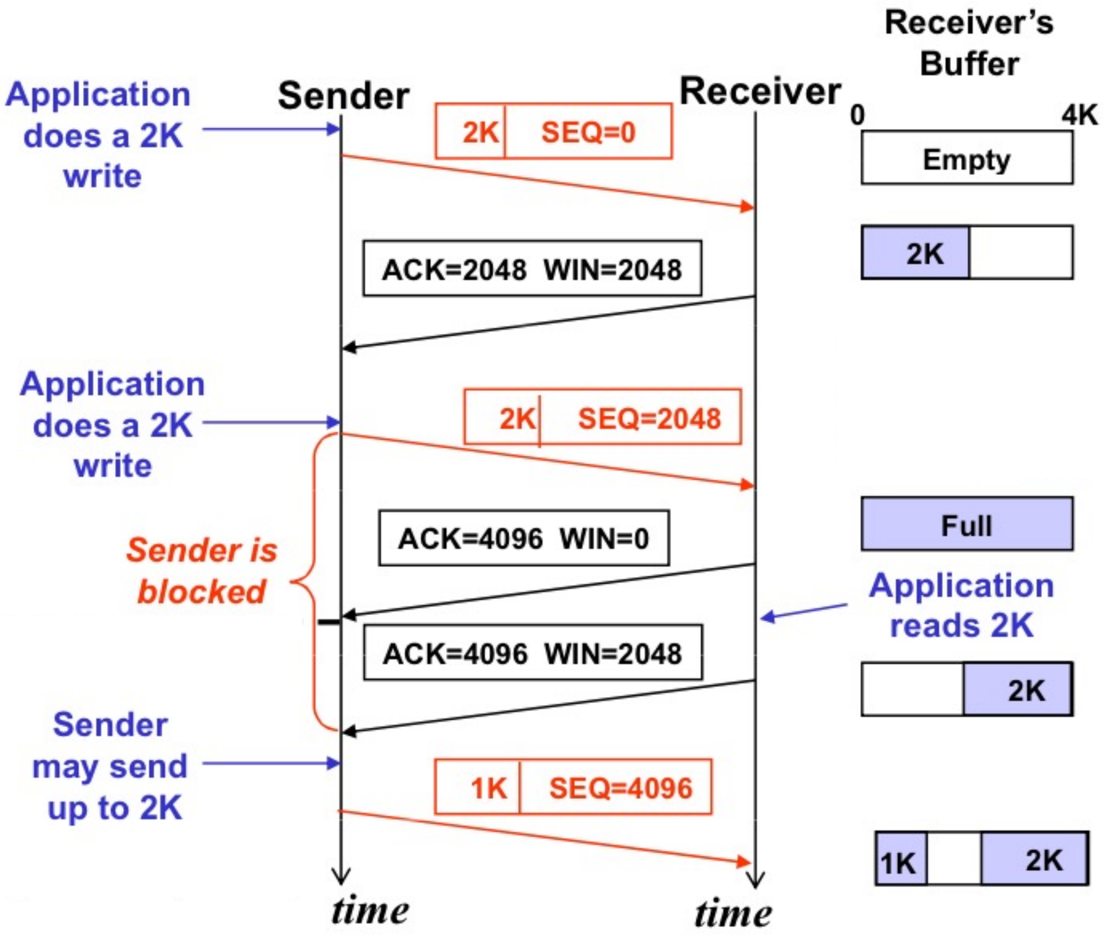
- Stop and Wait
  - Time consuming, does not use the full size of the pipe
- Sliding Window
  - Uses the pipe size (i.e., the minimum bandwidth over the path) to determine how many packets can be sent at a time
  - This is different from Layer 2 where window is fixed
  - Layer 4 windowing starts small then expands to fill the space in the pipe

Why does TCP do this?

TCP Buffering and Windowing

- Let’s assume that an application sends large amounts of data between hosts at once
  - What if one host is not fast enough at handling the data?
  - What if there is a bottleneck at one point in the network?
- The data sent through the network is dropped as the host/network may not have enough resources to deal with the data
- This is a waste of network resources
- Solution: TCP uses **Windowing** and **Congestion control**

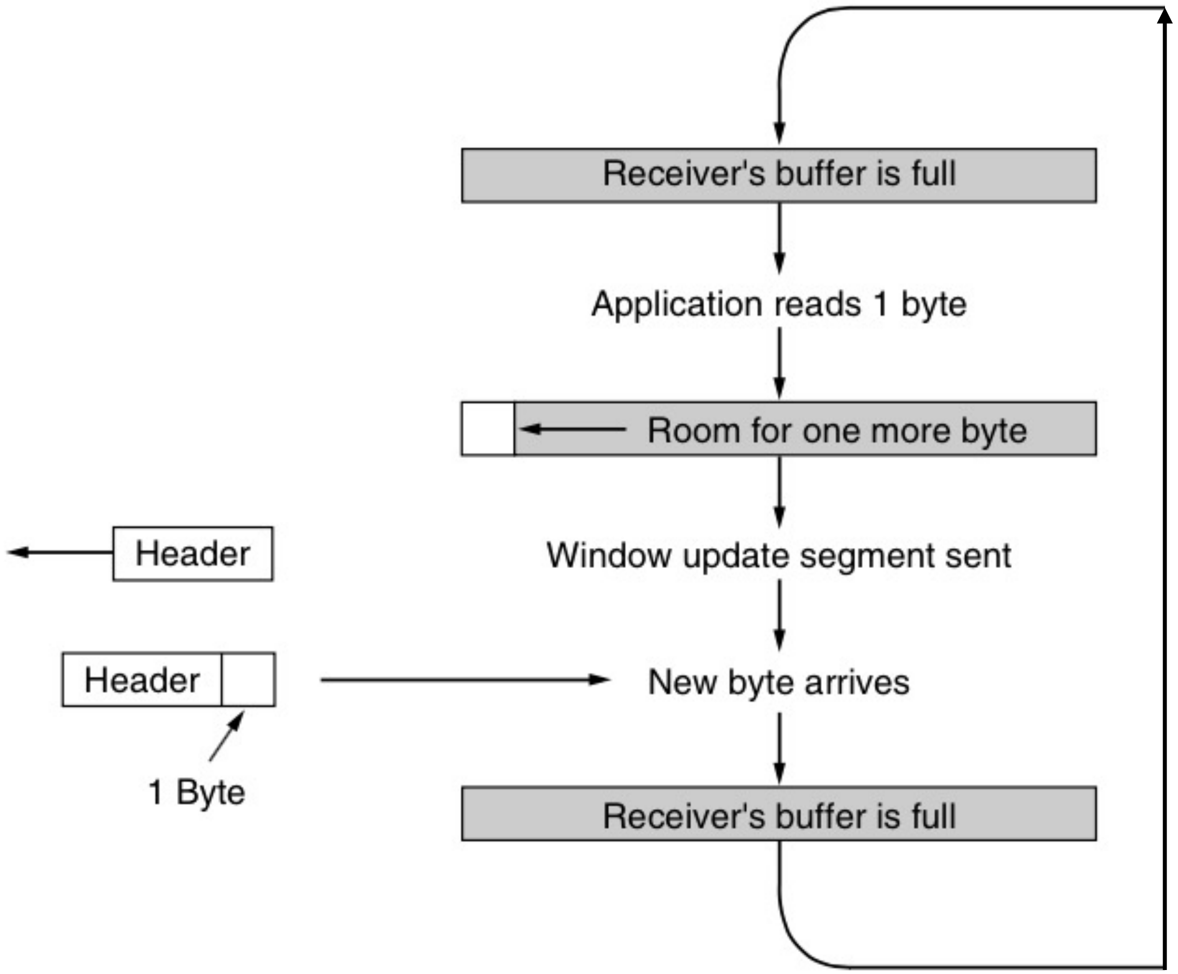
TCP Buffering and Windowing



TCP Buffering and Windowing

- Imagine if a receiver sends a window update every time there is space for one byte in the buffer
  - Silly Window!

- Solution: receiver only sends a window update if:
  - buffer is half empty,
  - or has more space than maximum-size segment

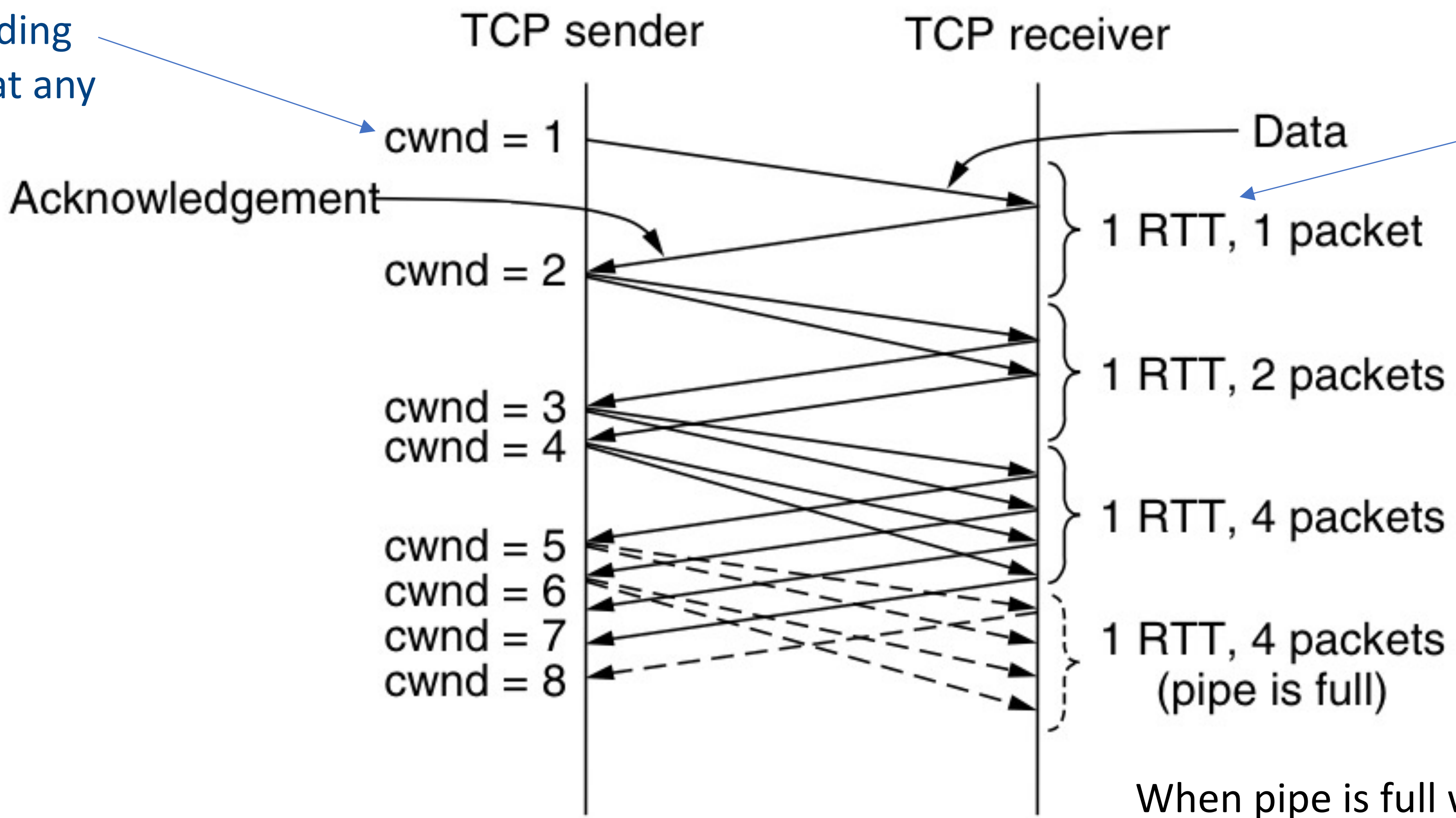


TCP Congestion

- Sending more packets than the network or receiver can handle
- We make an assumption in TCP
  - Timeout = packet loss due to congestion
  - Packets rarely lost through other means
- To avoid congestion, a sender will think:
  - If I receive an ACK for a packet, there is enough space in the network for me to send another!
  - Using ACK's to pace the transmission of new packets
- Sending one packet per acknowledgement may underutilize the network however
- TCP implements some schemes to reduce congestion, while using the network capacity

TCP Congestion - Slow Start

congestion window size: Number of outstanding segments allowed at any given time

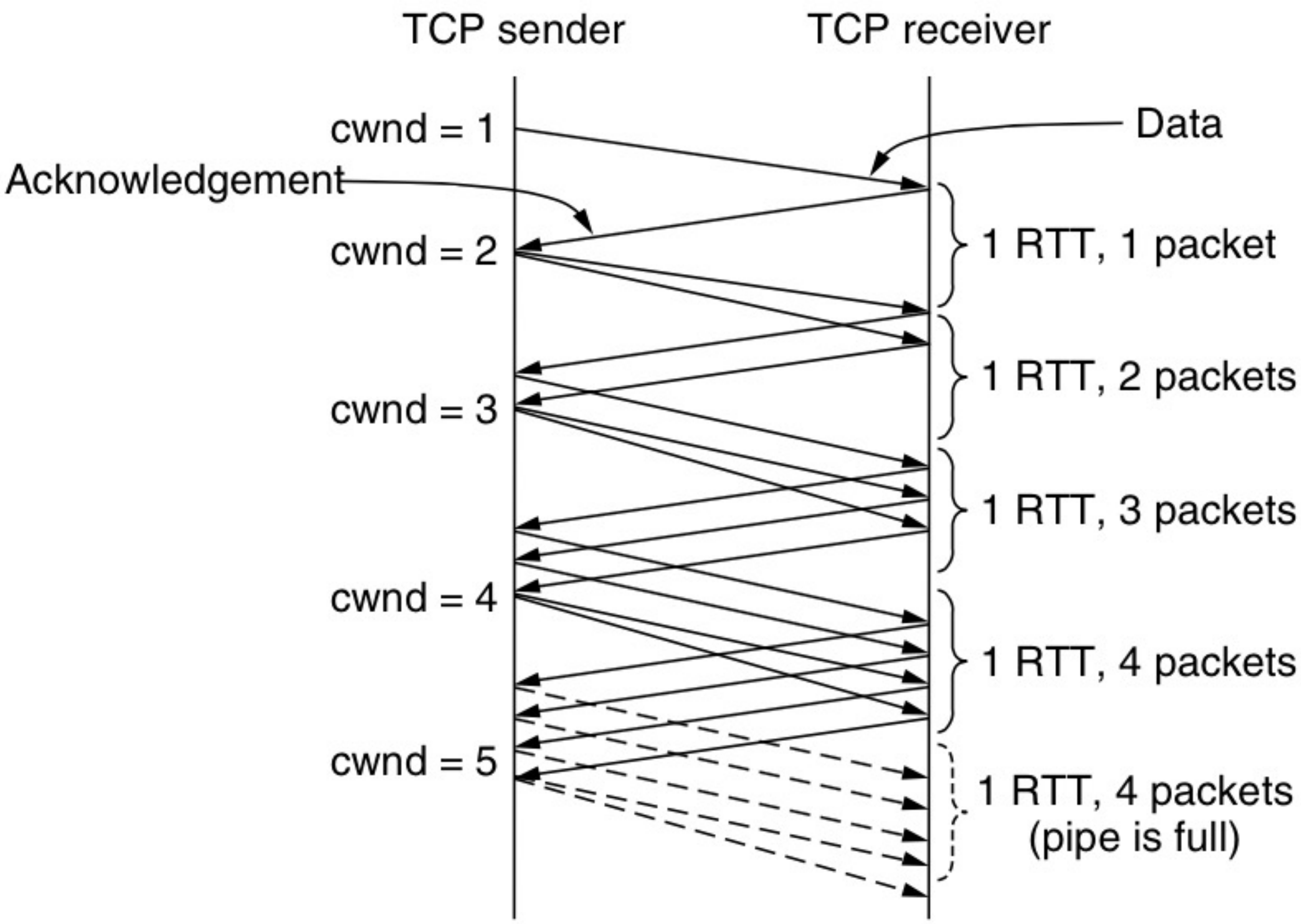


Round Trip Time

Exponential growth of packets

When pipe is full we will get a timeout!

TCP Congestion – Additive Increase



TCP Congestion

- **Slow Start:** Using Slow Start algorithm
- **SS Threshold:** Slow Start threshold

- **Congestion Avoidance:** Using Additive increase algorithm

- **Multiplicative Decrease:** Reducing the window when congestion is noticed

- **Fast Retransmit:** Using repeat ACKs to determine if a packet has been lost (don't wait for Timeout, send lost packet!)
- **Fast Recovery:** on packet loss set the cwnd to ½ the size and continue with additive increase

TCP Congestion

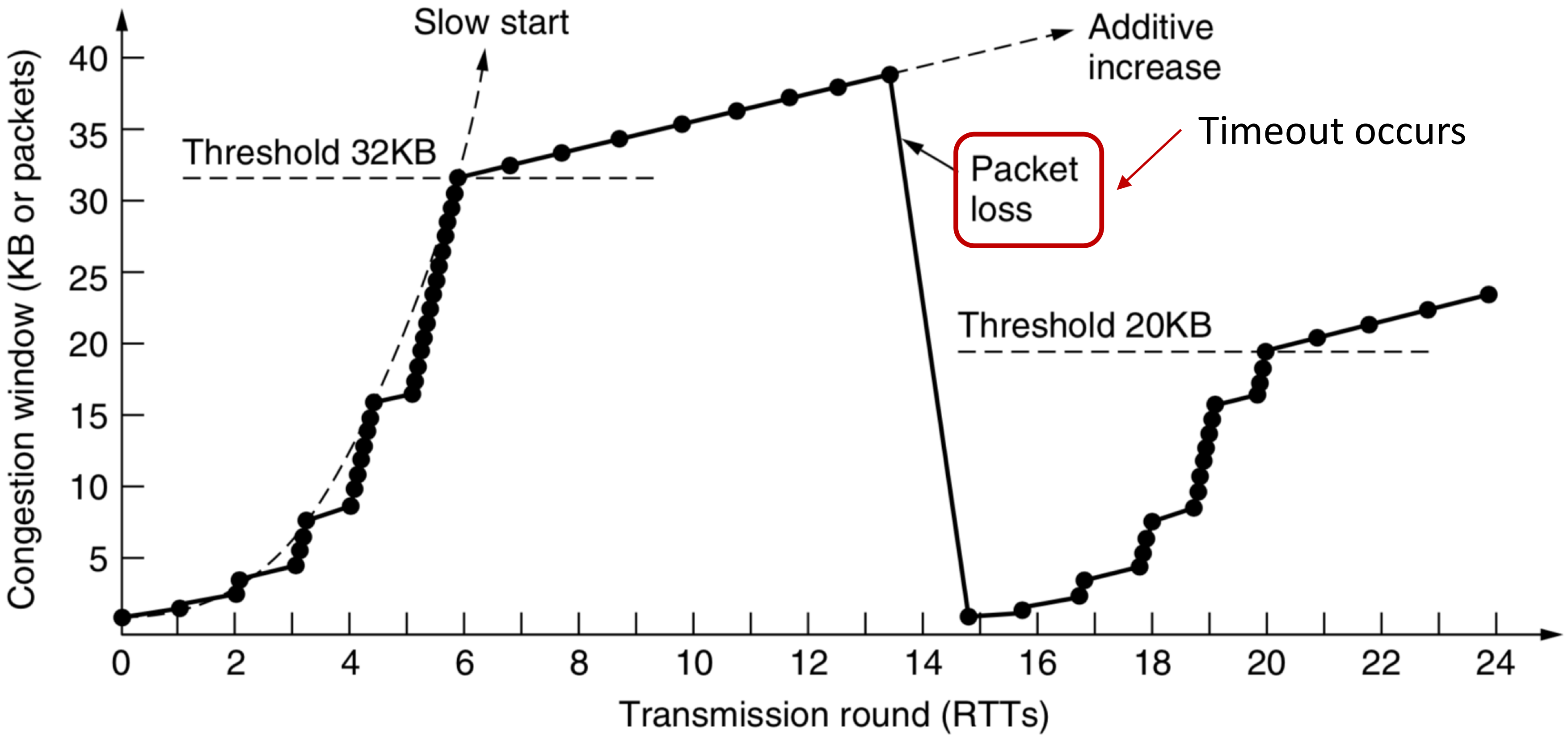
Tahoe:

- Start With Slow Start until:
  - SS Threshold is met
    - ➢ Go to Congenstion Avoidance
  - Packet lost (3 resubmitted Acks)
    - ➢ Go to packet loss!
- Start Congestion Avoidance until:
  - Packet loss (3 resubmitted Acks)
    - ➢ Go to packet loss!
- On Packet Loss:
  - Fast retransmit
  - SSthreshold = ½ CWND
  - CWND = initial state
  - ➢ Go to Slow Start
- Time Out:
  - CWND = 1
  - ➢ Go to Slow Start

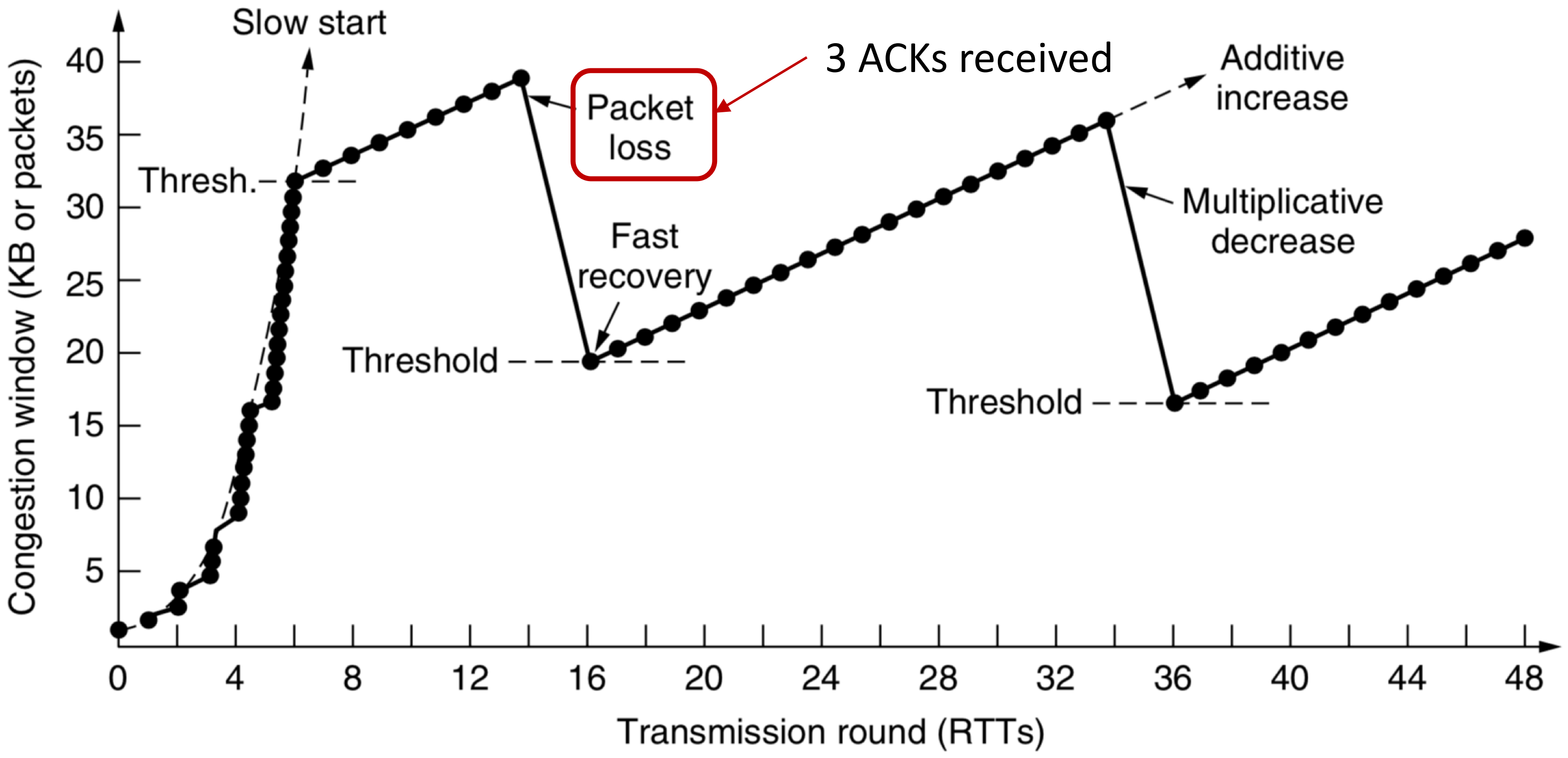
Reno:

- Start With Slow Start until:
  - SS Threshold is met
    - ➢ Go to Congenstion Avoidance
  - Packet lost (3 resubmitted Acks)
    - ➢ Go to packet loss!
- Start Congestion Avoidance until:
  - Packet loss (3 resubmitted Acks)
    - ➢ Go to packet loss!
- On Packet Loss:
  - Fast retransmit
  - CWND = ½ Current CWND
  - ➢ Go to Congestion Avoidance
- Time Out:
  - CWND = 1
  - ➢ Go to Slow Start

TCP - Tahoe


Slow start
Additive increase
Timeout occurs
Packet loss
Threshold 32KB
Threshold 20KB
Congestion window (KB or packets)
Transmission round (RTTs)
40
35
30
25
20
15
10
5
0
2
4
6
8
10
12
14
16
18
20
22
24

TCP - Reno


Slow start
3 ACKs received
Additive increase
Packet loss
Thresh.
Fast recovery
Threshold
Multiplicative decrease
Threshold
Congestion window (KB or packets)
40
35
30
25
20
15
10
5
0
4
8
12
16
20
24
28
32
36
40
44
48
Transmission round (RTTs)