ENEE 457: Computer Systems Security

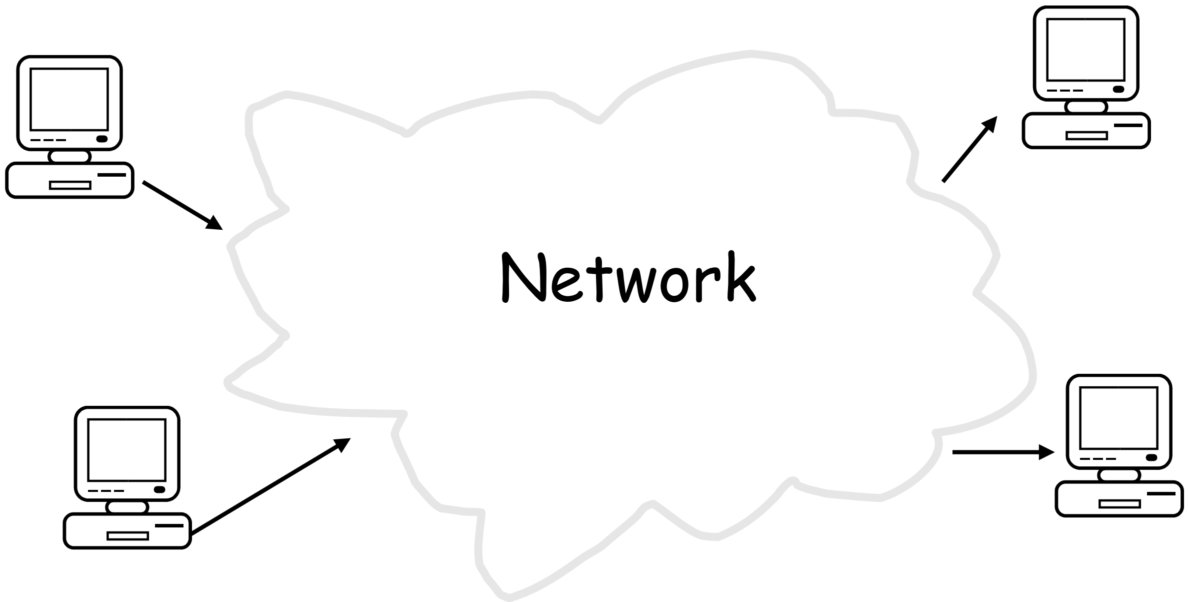
Lecture 18
Computer Networking Basics

Charalampos (Babis) Papamanthou

Department of Electrical and Computer Engineering
University of Maryland, College Park

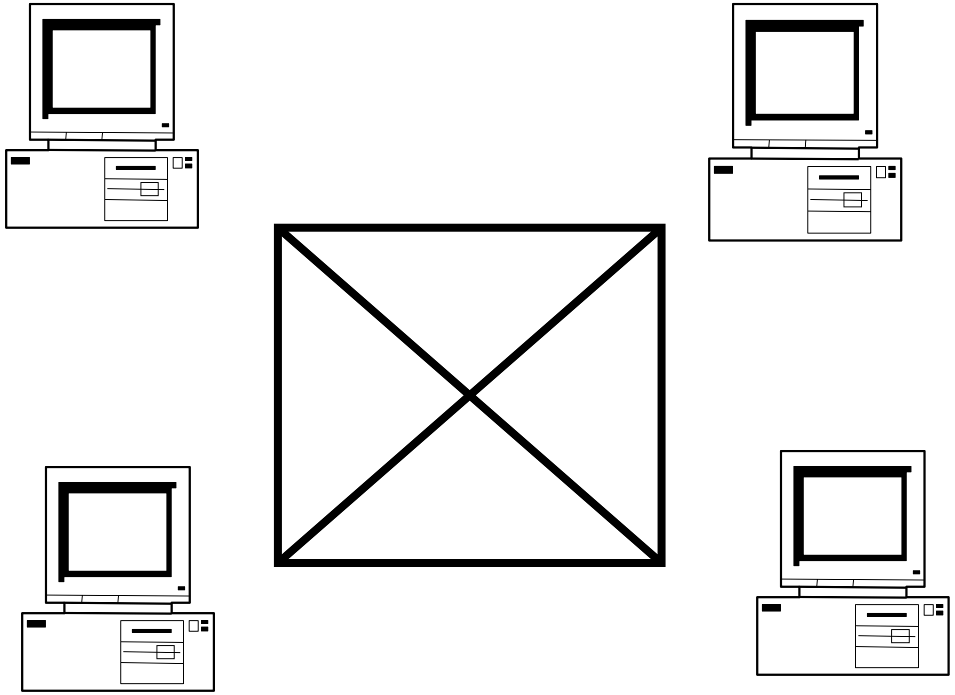
Computer Networking: The general problem

- Lots of devices, hosts want to communicate with each other
    - To exchange data
    - To access remote services
- Goal: Universal Communication (from any host to any host)
- How do we design the cloud?



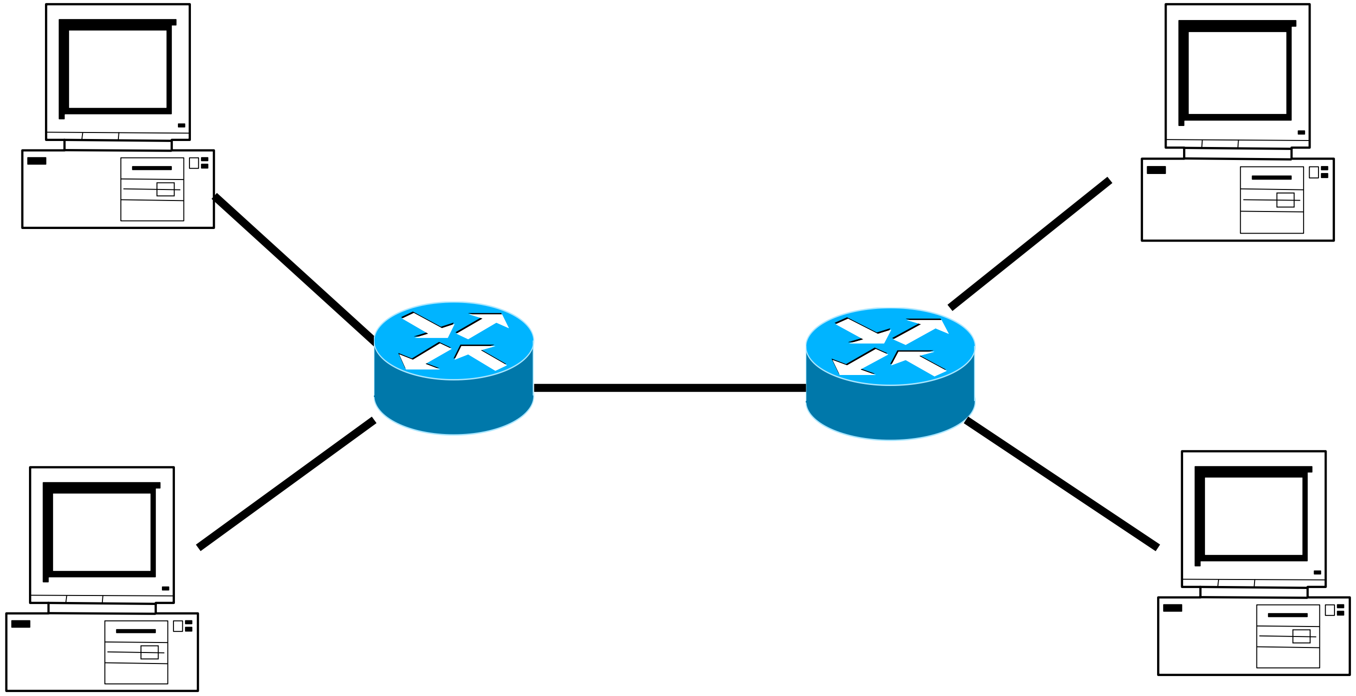
First approach: N^2 links

- Connect every host with each other
  - Does not work, too costly, plus hosts come and go, too difficult to manage
  - In particular, the complexity is quadratic



Second approach: Share resources

- Share resources through intermediate routers and switches (that is how internet works)

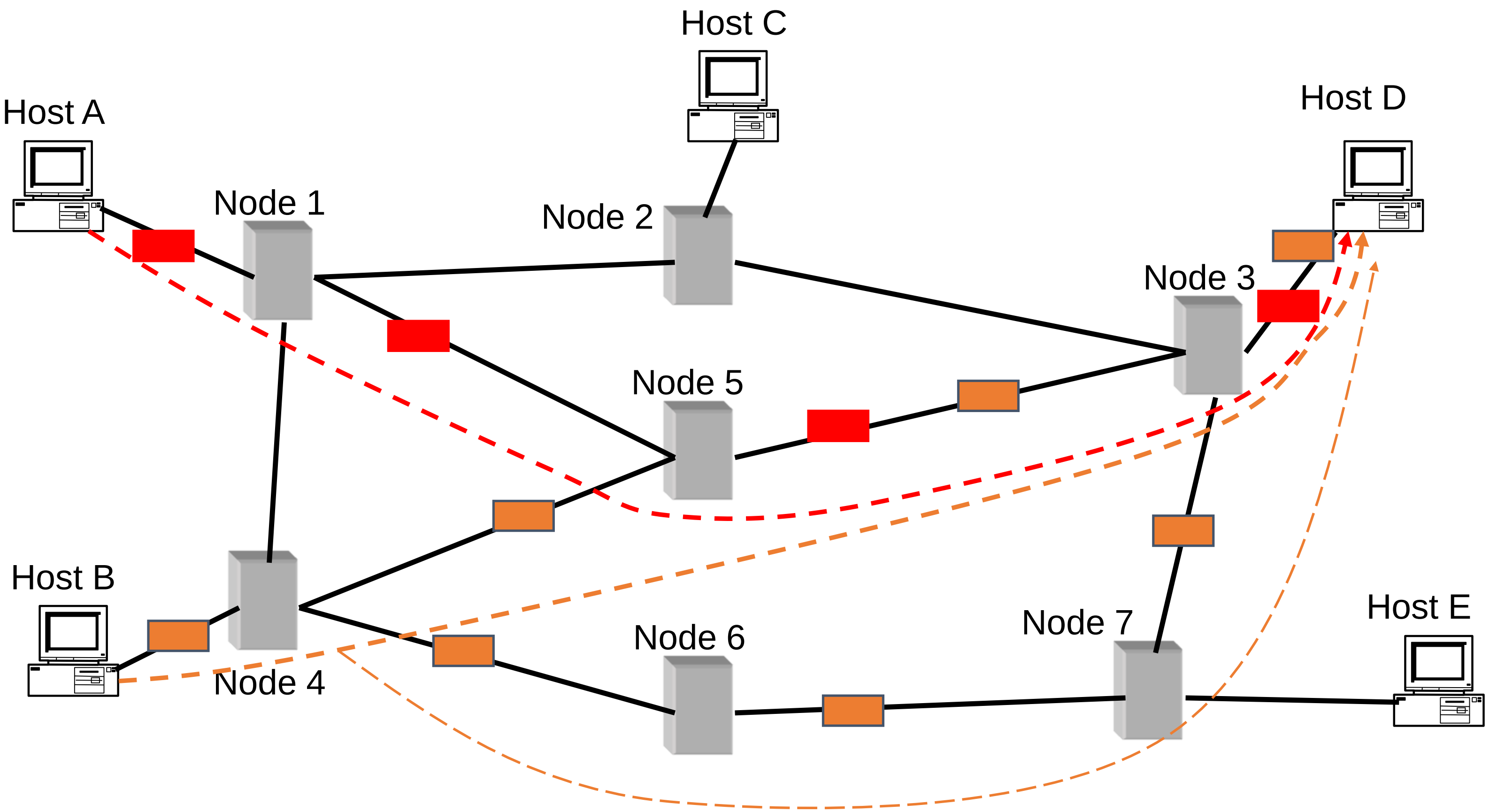


Two ways for that: Circuit and Packet switching

- **Circuit switching**
  - Legacy phone network
  - Single route through sequence of hardware devices established when two nodes start communication
  - Data sent along route
  - Route maintained until communication ends
- **Packet switching**
  - Internet
  - Data split into packets
  - Packets transported independently through network
  - Each packet handled on a best efforts basis
  - Packets may follow different routes

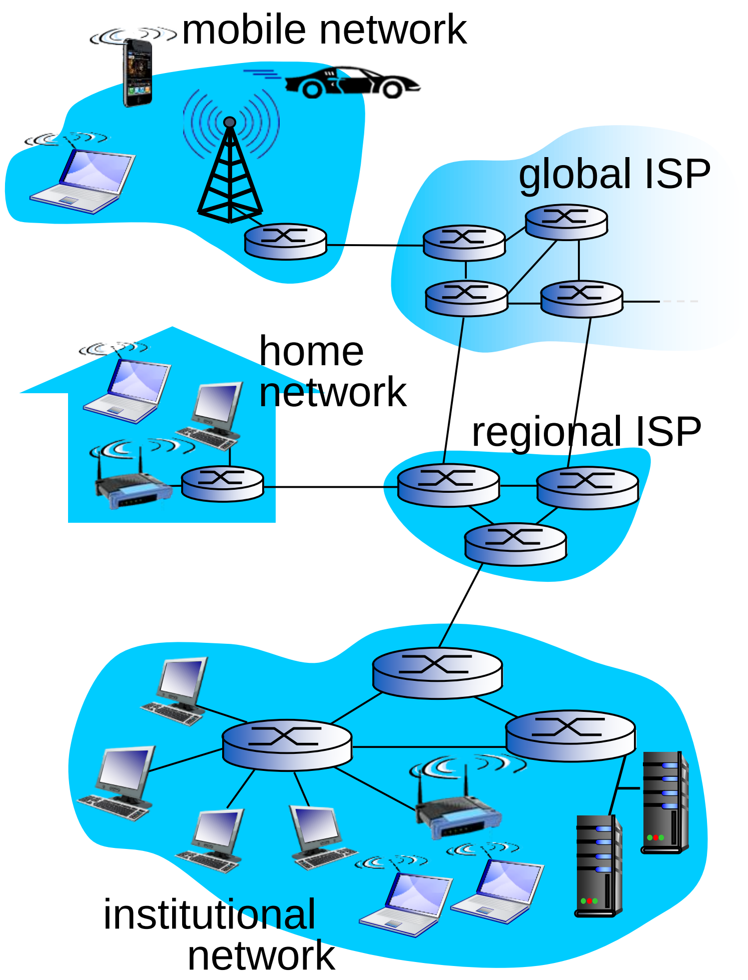
Packet switching

- Packets in a flow may not follow the same path (depends on routing as we will see later) → packets may be reordered



What is the internet?

- Network of Networks based on packet switching
- Millions of connected computing devices that can communicate with each other
    - Desktops, laptops, smartphones, servers
- Communication links
    - fiber, copper, radio, satellite
- Packet switches
    - forward packets (chunks of data) between ISPs
- Protocols
    - HTTP, TCP, IP



Layered organization of Internet

- Internet is complex, with many pieces
  - hosts
  - routers
  - links of various media
  - applications
  - protocols
  - hardware, software

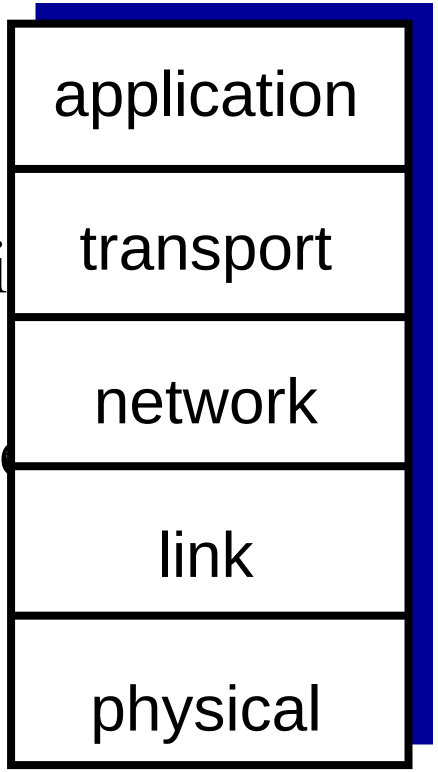
Question:

is there any hope of having an *organizing* structure of the internet?

…. or at least our discussion of networks

Internet Protocol Stack

- **Application layer**: supporting network applications
  - FTP, SMTP, HTTP
- **Transport layer**: process-process data transfer
  - TCP, UDP
- **Network layer**: routing of datagrams from source to destinati
  - IP, routing protocols
- **Link layer**: data transfer between neighboring network eleme
  - Ethernet, 802.111 (WiFi),
- **Physical layer**: bits "on the wire"



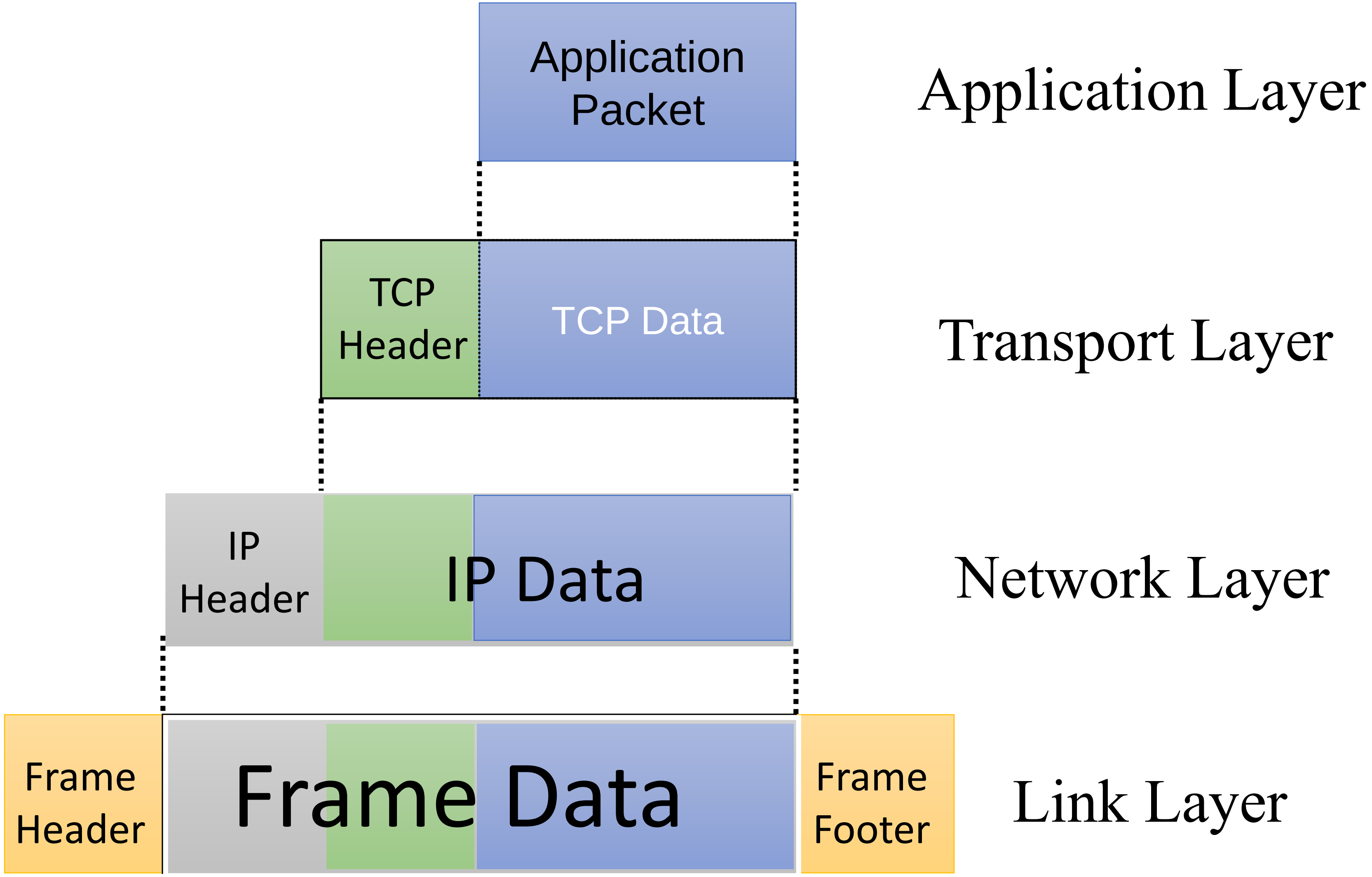
Examples

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

More examples

- Application layer
  - E.g., contains all useful network applications
  - e.g., FTP, HTTP, SSH, telnet
- Transport layer
  - E.g., contains protocols defining the properties of the connection (e.g., connection oriented/connectionless)
  - uses 16-bit addresses (ports)
  - e.g., TCP, UDP, DSCP (implements congestion control)
- Network layer
  - E.g., contains protocols defining how to route between logical addresses (e.g., IPs)
  - uses 32-bit internet protocol (IP) addresses in IPv4
  - 128-bit IP addresses in IPv6
  - Best efforts
  - e.g., IPv4, IPv6, IPsec (providing security)
- Link layer
  - E.g., contains protocols defining how to route between physical addresses (e.g., MAC addresses) depending on the physical medium (Ethernet, WiFi, optical fiber)
  - uses 48-bit media access control (MAC) addresses
  - Local area network: Packets called frames
  - e.g., IEEE 802.11 (for wireless)
- Physical layer: How to physically send the information

Internet Packet Encapsulation



Travelling of an email message

- Given the email address and the email program, the application layer will figure out
  - The destination IP (through a DNS query)
  - The destination port (e.g., for HTTP it is 80)
- Then various TCP or UDP segments will be created. If it is TCP, then ordering should be included. If it is UDP, losses can be tolerated
- Then various IP segments will be created, which will be sent at best-effort basis. This segment will indicate what kind of routing we are doing

Routers and Switches

- Routers are used to route across networks, where I do not know the MAC address of the machine I am trying to send
- Switches are used to route within LANs where I do know the MAC of the machine it should go
- E.g.: When I am sending a an email from **ece.umd.edu** to **eecs.mit.edu**
  - My message will be routed through local switches in my building/office to the router of my building/office.
  - At that point, the router will see that there is no local machine that it knows that can handle this message, but, based on the IP, it will find another router that can handle this message
  - Eventually, the message will get to the router of mit.edu
  - And through local switches and other routers will reach the destination

DNS protocol

- It runs at the application layer
- It maps http addresses to IP addresses

ARP (address resolution protocol)

- The address resolution protocol (ARP) connects the network layer to the data layer by converting IP addresses to MAC addresses
- ARP works by broadcasting requests and caching responses for future use
- The protocol begins with a computer broadcasting a message of the form

  who has <IP address1> tell <IP address2>

- When the machine with <IP address1> or an ARP server receives this message, its broadcasts the response

  <IP address1> is <MAC address>

- The requestor’s IP address <IP address2> is contained in the link header
- The Linux and Windows command arp - a displays the ARP table

Internet Address	Physical Address	Type
128.148.31.1	00-00-0c-07-ac-00	dynamic
128.148.31.15	00-0c-76-b2-d7-1d	dynamic
128.148.31.71	00-0c-76-b2-d0-d2	dynamic
128.148.31.75	00-0c-76-b2-d7-1d	dynamic
128.148.31.102	00-22-0c-a3-e4-00	dynamic
128.148.31.137	00-1d-92-b6-f1-a9	dynamic

ARP Spoofing

- The ARP table is updated whenever an ARP response is received
- Requests are not tracked
- ARP announcements are not authenticated
- Machines trust each other
- A rogue machine can spoof other machines

ARP Poisoning (ARP Spoofing)

- According to the standard, almost all ARP implementations are stateless
- An arp cache updates every time that it receives an arp reply… even if it did not send any arp request!
- It is possible to "poison" an arp cache by sending gratuitous arp replies
- Using static entries solves the problem but it is almost impossible to manage!

Transport layer protocols

TCP :

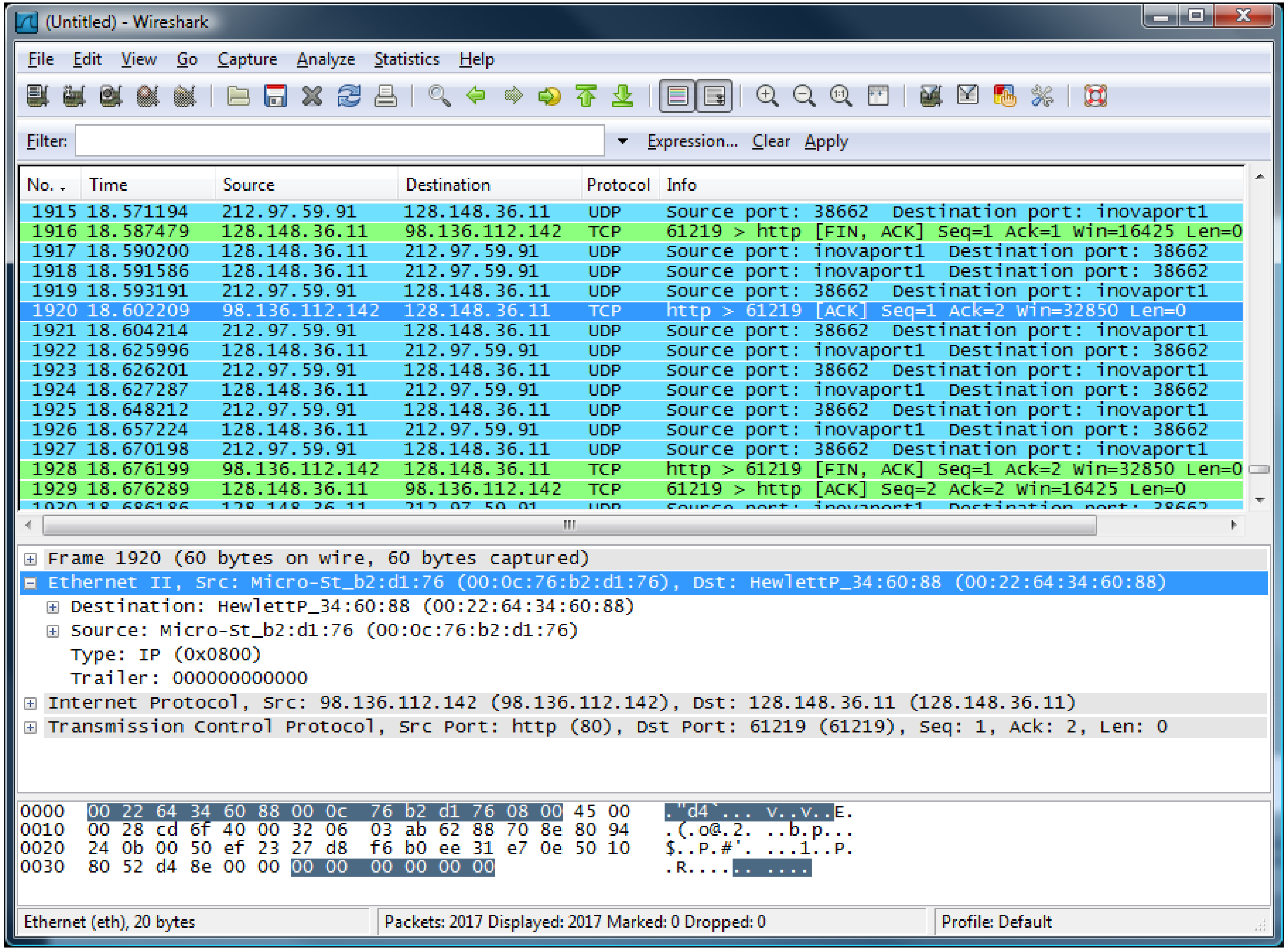
- reliable transport between sending and receiving process
- flow control: sender won't overwhelm receiver
- congestion control: throttle sender when network overloaded
- does not provide: timing, minimum throughput guarantee, security
- connection-oriented: setup required between client and server processes

UDP :

- unreliable data transfer between sending and receiving process
- does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security, orconnection setup,

Q: why bother? Why is there a UDP?

Sniffing Network Packets with Wireshark



Network Security Problems

- The internet was not designed with security in mind
- Bad guys can
  - Launch DOS attacks
  - Can sniff packets
  - Can change the origin destination addresses
- To have security when you use networking, you need to use TLS at the application layer
  - Confidentiality
  - Integrity
  - Authentication

Transport Vs Network Layer

- Network Layer: Communication between computers (e.g., from IP1 to IP2)
- Transport Layer: Communication between processes (e.g., serving an HTTP request)
  - UDP (unreliable, best-effort)
  - TCP (reliable)

TCP protocol

- TCP creates reliable transportation over an unreliable network

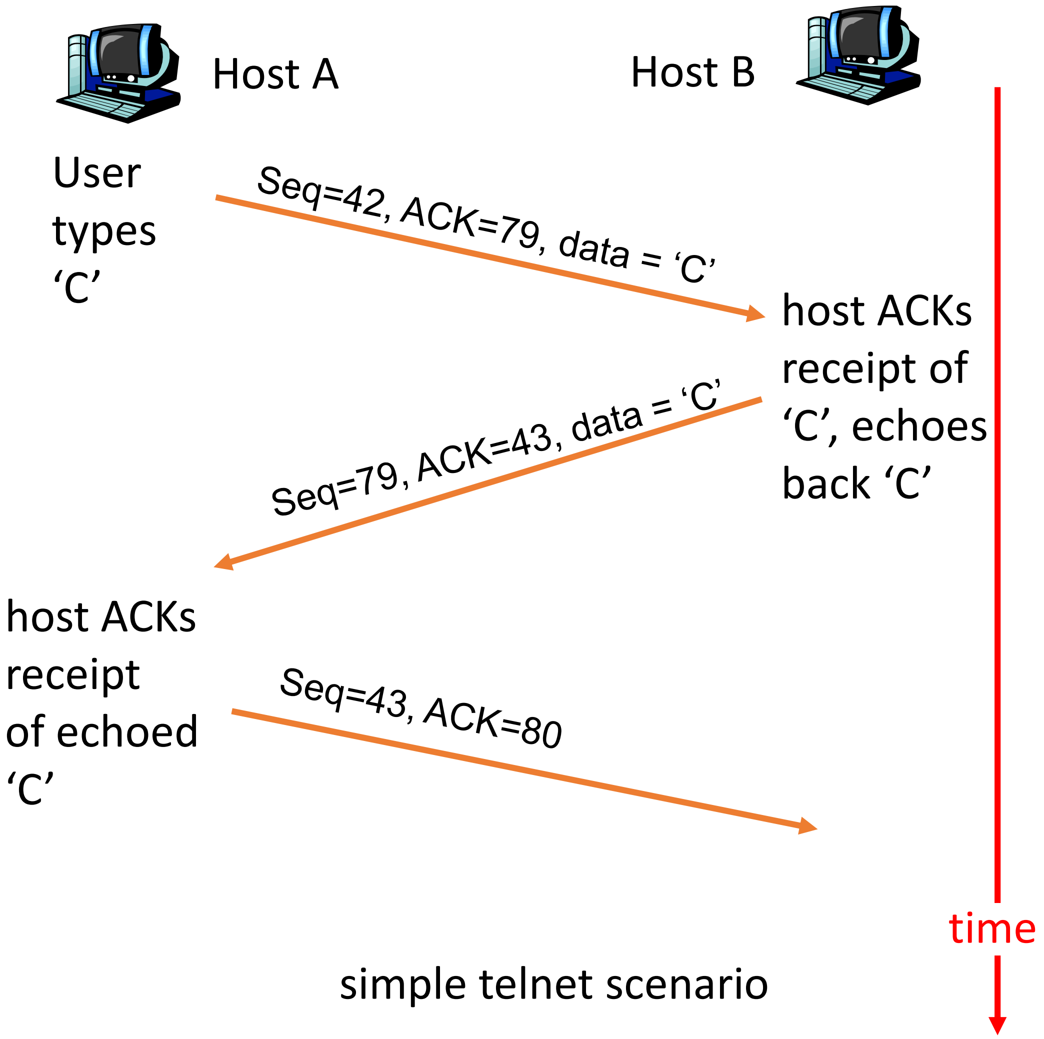
TCP seq. #'s and ACKs

Seq. #'s:

- byte stream "number" of first byte in segment's data
- It can be used as a pointer for placing the received data in the receiver buffer

ACKs:

- seq # of next byte expected from other side



simple telnet scenario

Example: Unidirectional Communication

Byte numbers

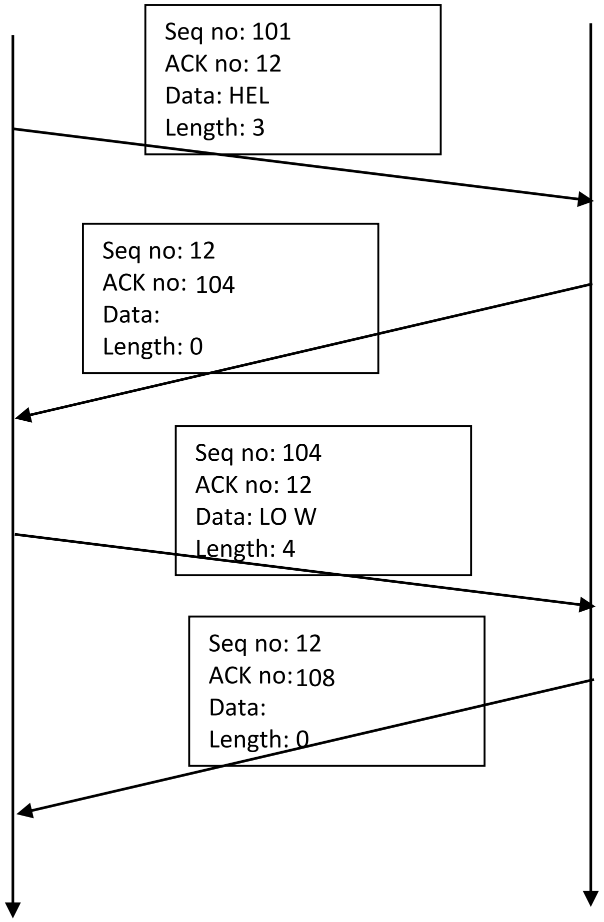
101	102	103	104	105	106	107	108	109	110	111
H	E	L	L	O		W	O	R	L	D

Seq. #'s:

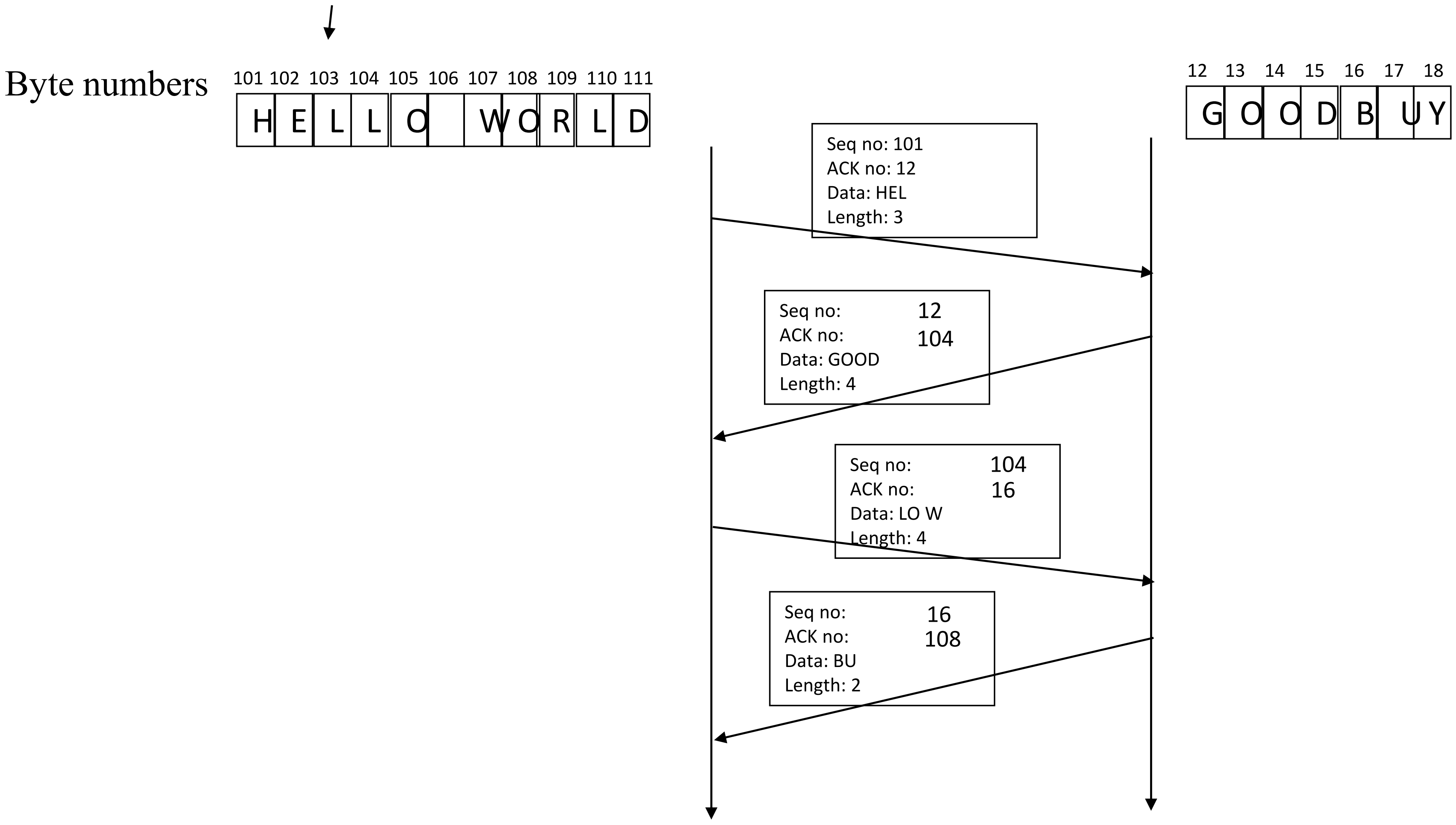
- byte stream "number" of first byte in segment's data
- It can be used as a pointer for placing the received data in the receiver buffer

ACKs:

- seq # of next byte expected from other side

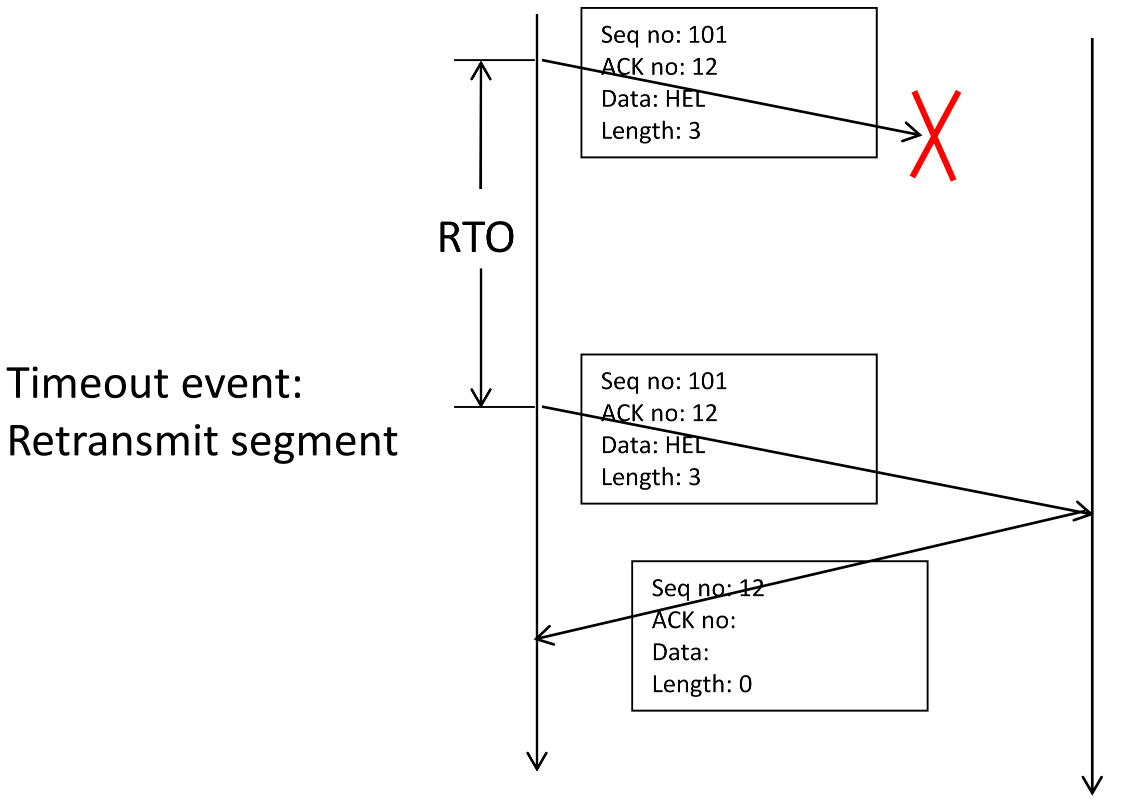


Bidirectional communication



Timeout

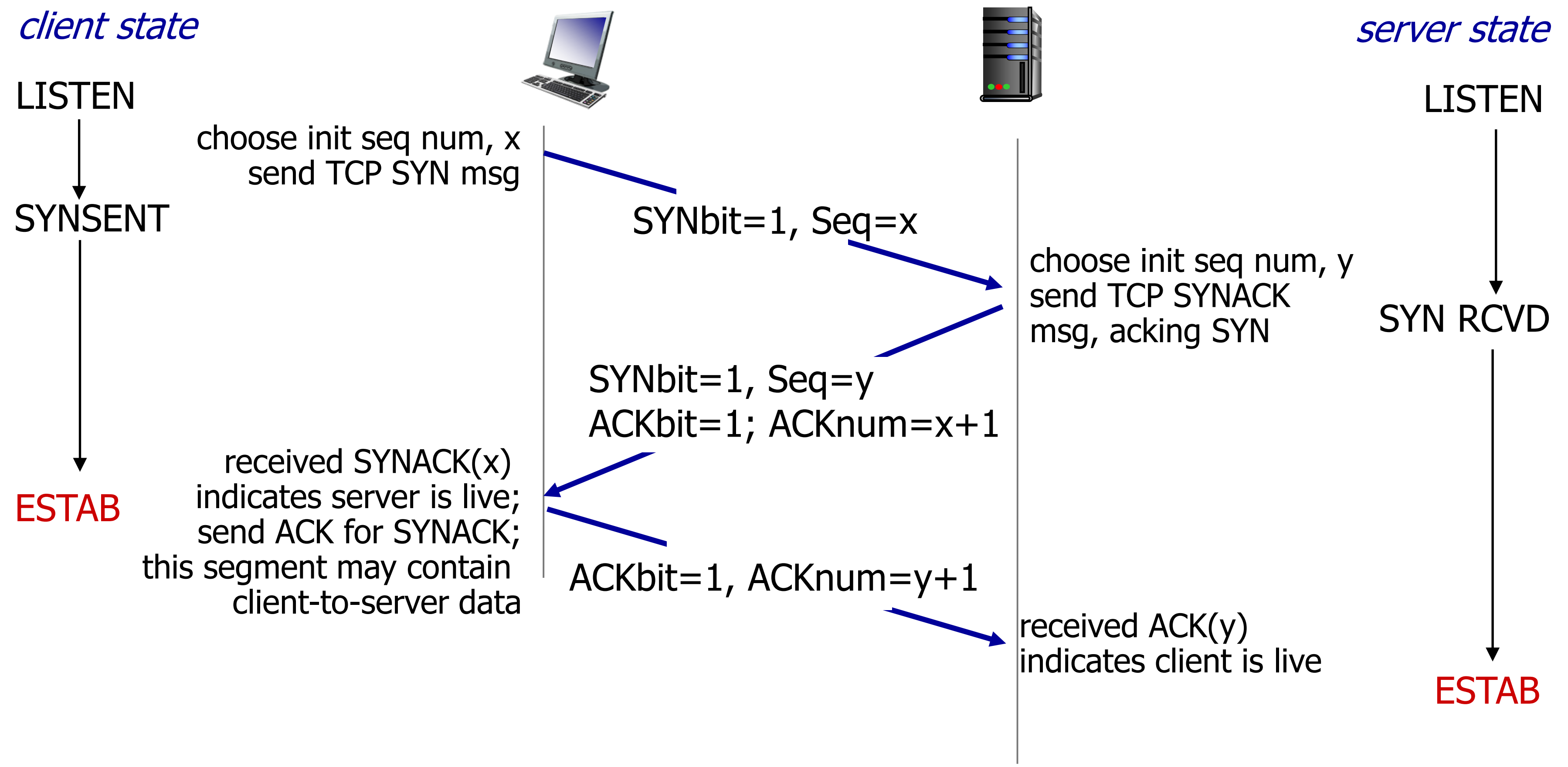
If an ACK is not received before RTO (retransmission timeout), a timeout is declared



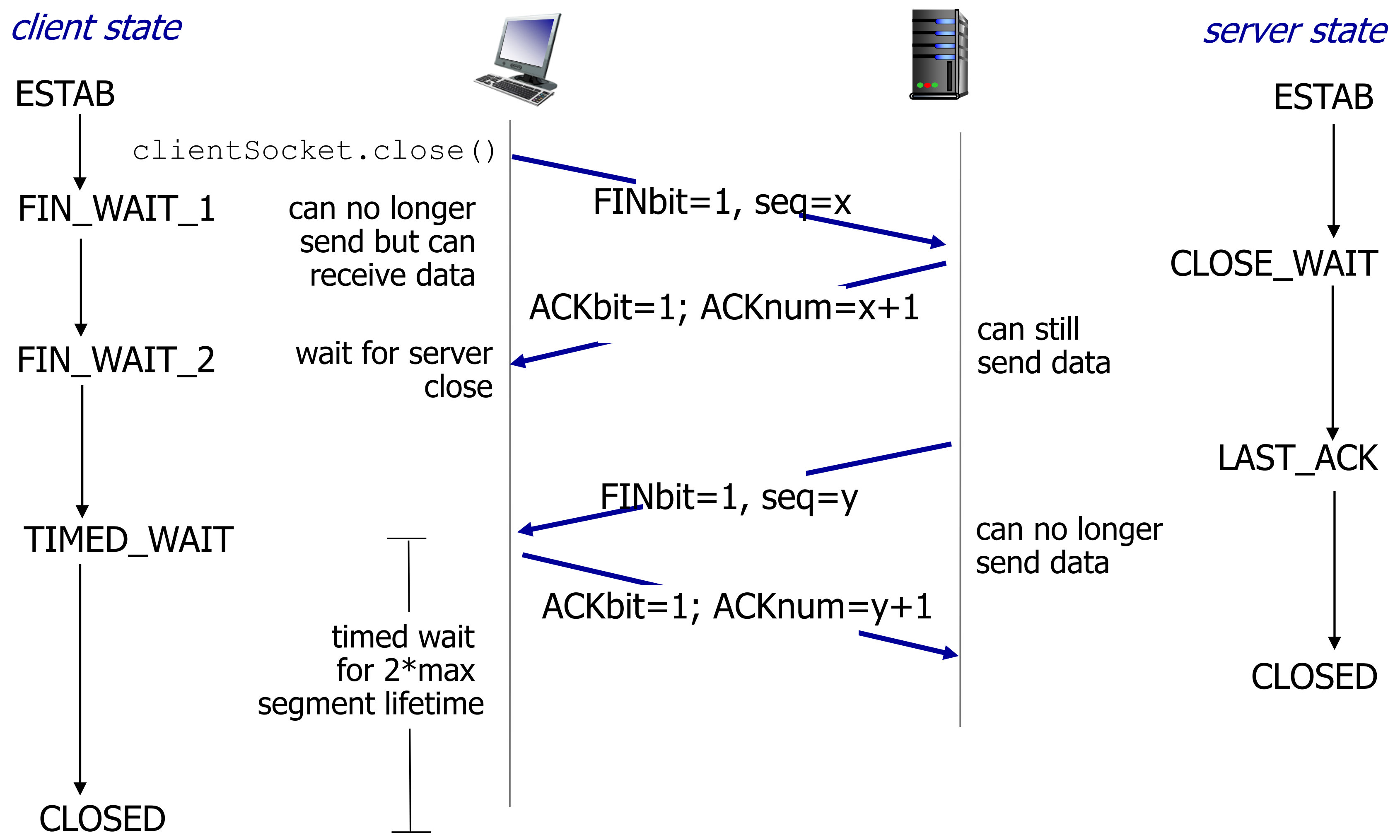
What if timeout is too big?
What is timeout is too small?

Ideally, a timeout should occur right after an ACK is received…

TCP 3-way handshake



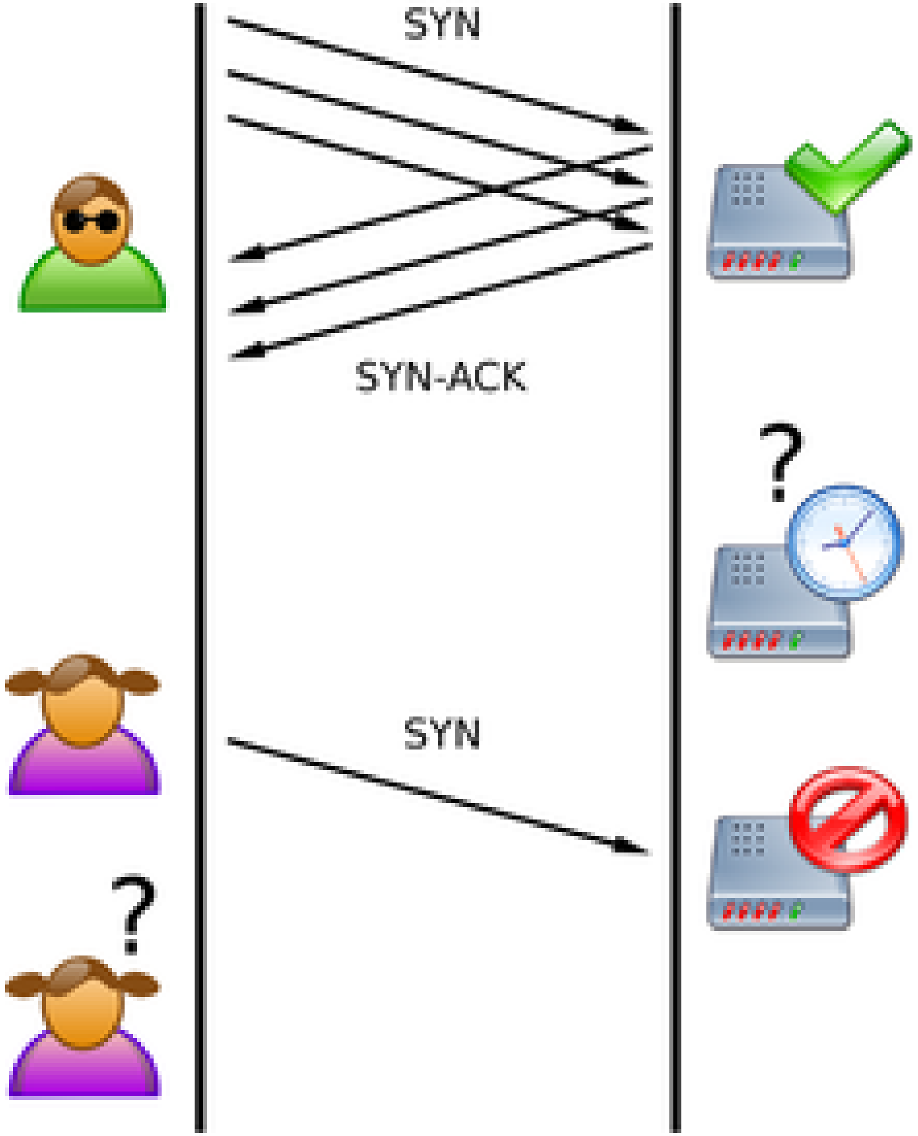
Closing a TCP connection



SYN-Flooding attack

- After the received receives a SYN message, it allocates space for the connection
- In particular it stores
  - IP of sender
  - Port of the application
  - Maximum segment size for this connection
  - Outgoing sequence number (note that I need this sequence number so that when I receive an ack I know that it refers to a specific TCP segment)
- The attack goes as follows:
  - The attacker spoofs a bunch of IPs sends a lot of SYNs
  - I reply with a lot of SYN-ACKs (hence I need to store all this information for each connection)
  - But no ACKs will never come back (since the IPs are random)
  - Hence I am keeping all these connections open for no reason exhausting my memory

Syn-Flooding attack



How to solve this problem: Syn Cookies

- Do not assign state until you get back the ACK
- But how to you store your generated sequence number?
- Just outsource it on the SYN-ACK message by using a MAC