

CSI 62
Operating Systems and
Systems Programming
Lecture 24

Networking and TCP/IP (Con't), RPC,
Distributed File Systems

April 21st, 2022

Prof. Anthony Joseph and John Kubiawicz

<http://cs162.eecs.Berkeley.edu>

Recall: Distributed Consensus Making

- Consensus problem
 - All nodes propose a value
 - Some nodes might crash and stop responding
 - Eventually, all remaining nodes decide on the same value from set of proposed values
- Distributed Decision Making
 - Choose between “true” and “false”
 - Or Choose between “commit” and “abort”
- Equally important (but often forgotten!): make it durable!
 - How do we make sure that decisions cannot be forgotten?
 - » This is the “D” of “ACID” in a regular database
 - In a global-scale system?
 - » What about erasure coding or massive replication?
 - » Like **BlockChain** applications!

Recall: Two-Phase Commit Protocol

- **Persistent stable log on each machine:** keep track of whether commit has happened
 - If a machine crashes, when it wakes up it first checks its log to recover state of world at time of crash
- **Prepare Phase:**
 - The global coordinator requests that all participants will promise to commit or **rollback** the **transaction**
 - Participants record promise in log, then acknowledge
 - If anyone votes to abort, coordinator writes **"Abort"** in its log and tells everyone to abort; each records **"Abort"** in log
- **Commit Phase:**
 - After all participants respond that they are prepared, then the coordinator writes **"Commit"** to its log
 - Then asks all nodes to commit; they respond with ACK
 - After receive ACKs, coordinator writes **"Got Commit"** to log
- Log used to guarantee that all machines either commit or don't

Distributed Decision Making Discussion (1/2)

- Why is distributed decision making desirable?
 - Fault Tolerance!
 - A group of machines can come to a decision even if one or more of them fail during the process
 - » Simple failure mode called “failstop” (different modes later)
 - After decision made, result recorded in multiple places
- Why is 2PC not subject to the General’s paradox?
 - Because 2PC is about *all nodes eventually coming to the same decision – not necessarily at the same time!*
 - Allowing us to reboot and continue allows time for collecting and collating decisions

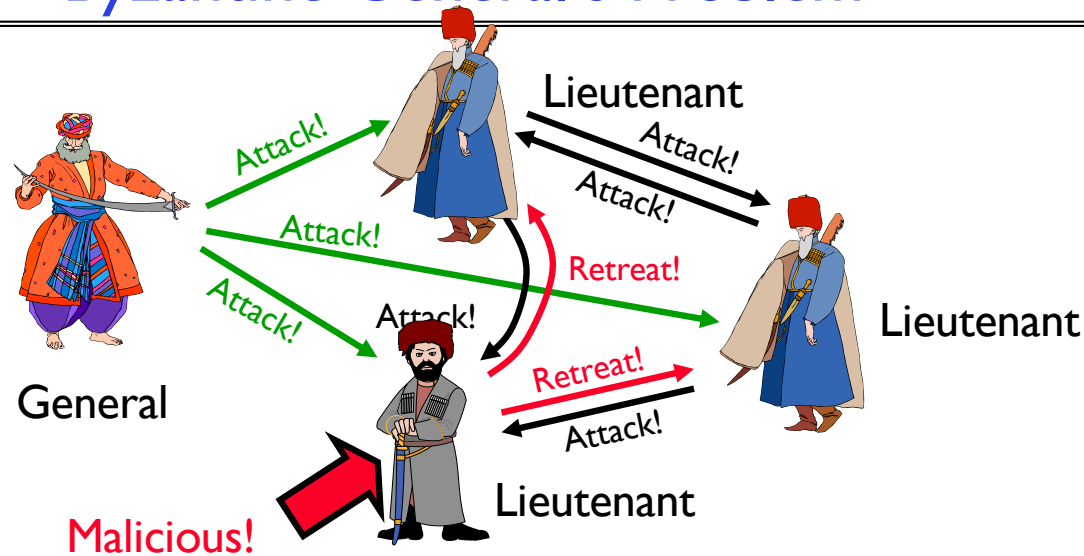
Distributed Decision Making Discussion (2/2)

- Undesirable feature of Two-Phase Commit: Blocking
 - One machine can be stalled until another site recovers:
 - » Site B writes "prepared to commit" record to its log, sends a "yes" vote to the coordinator (site A) and crashes
 - » Site A crashes
 - » Site B wakes up, check its log, and realizes that it has voted "yes" on the update. It sends a message to site A asking what happened. At this point, B cannot decide to abort, because update may have committed
 - » B is blocked until A comes back
 - A blocked site holds resources (locks on updated items, pages pinned in memory, etc) until learns fate of update

Alternatives to 2PC

- **Three-Phase Commit:** One more phase, allows nodes to fail or block and still make progress.
- **PAXOS:** An alternative used by Google and others that does not have 2PC blocking problem
 - Develop by Leslie Lamport (Turing Award Winner)
 - No fixed leader, can choose new leader on fly, deal with failure
 - Some think this is extremely complex!
- **RAFT:** PAXOS alternative from John Ousterhout (Stanford)
 - Simpler to describe complete protocol
- What happens if one or more of the nodes is malicious?
 - **Malicious:** attempting to compromise the decision making
 - Use a more hardened decision making process:
Byzantine Agreement and **Block Chains**

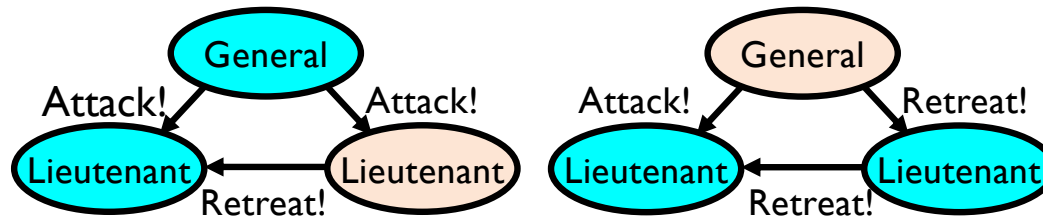
Byzantine General's Problem



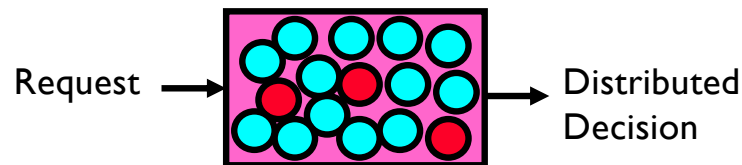
- Byzantine General's Problem (n players):
 - One General and $n-1$ Lieutenants
 - Some number of these (f) can be insane or malicious
- The commanding general must send an order to his $n-1$ lieutenants such that the following Integrity Constraints apply:
 - IC1: All loyal lieutenants obey the same order
 - IC2: If the commanding general is loyal, then all loyal lieutenants obey the order he sends

Byzantine General's Problem (con't)

- Impossibility Results:
 - Cannot solve Byzantine General's Problem with $n=3$ because one malicious player can mess up things

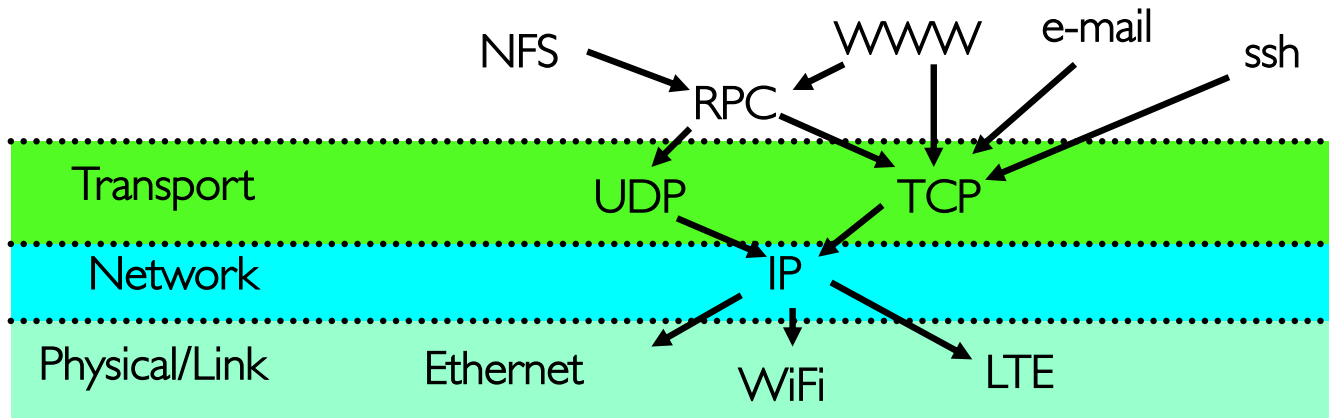


- With f faults, need $n > 3f$ to solve problem
- Various algorithms exist to solve problem
 - Original algorithm has #messages exponential in n
 - Newer algorithms have message complexity $O(n^2)$
 - » One from MIT, for instance (Castro and Liskov, 1999)
- Use of BFT (Byzantine Fault Tolerance) algorithm
 - Allow multiple machines to make a coordinated decision even if some subset of them ($< n/3$) are malicious



Network Protocols

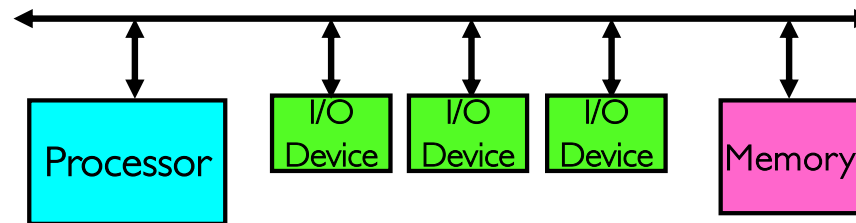
- Networking protocols: many levels
 - Physical level: mechanical and electrical network (e.g., how are 0 and 1 represented)
 - Link level: packet formats/error control (for instance, the CSMA/CD protocol)
 - Network level: network routing, addressing
 - Transport Level: reliable message delivery
- Protocols on today's Internet:



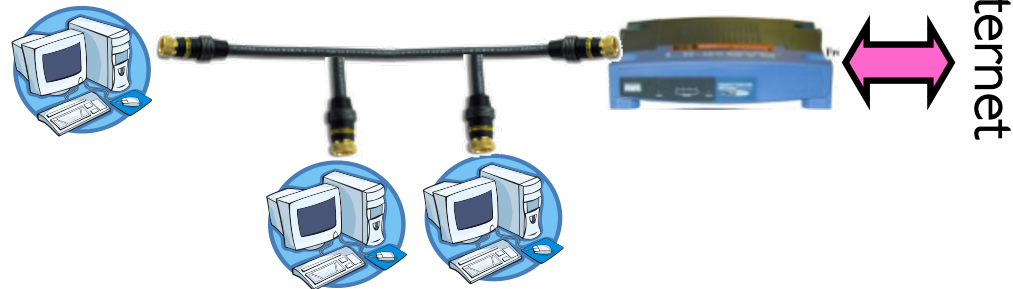
Broadcast Networks



- **Broadcast Network:** Shared Communication Medium

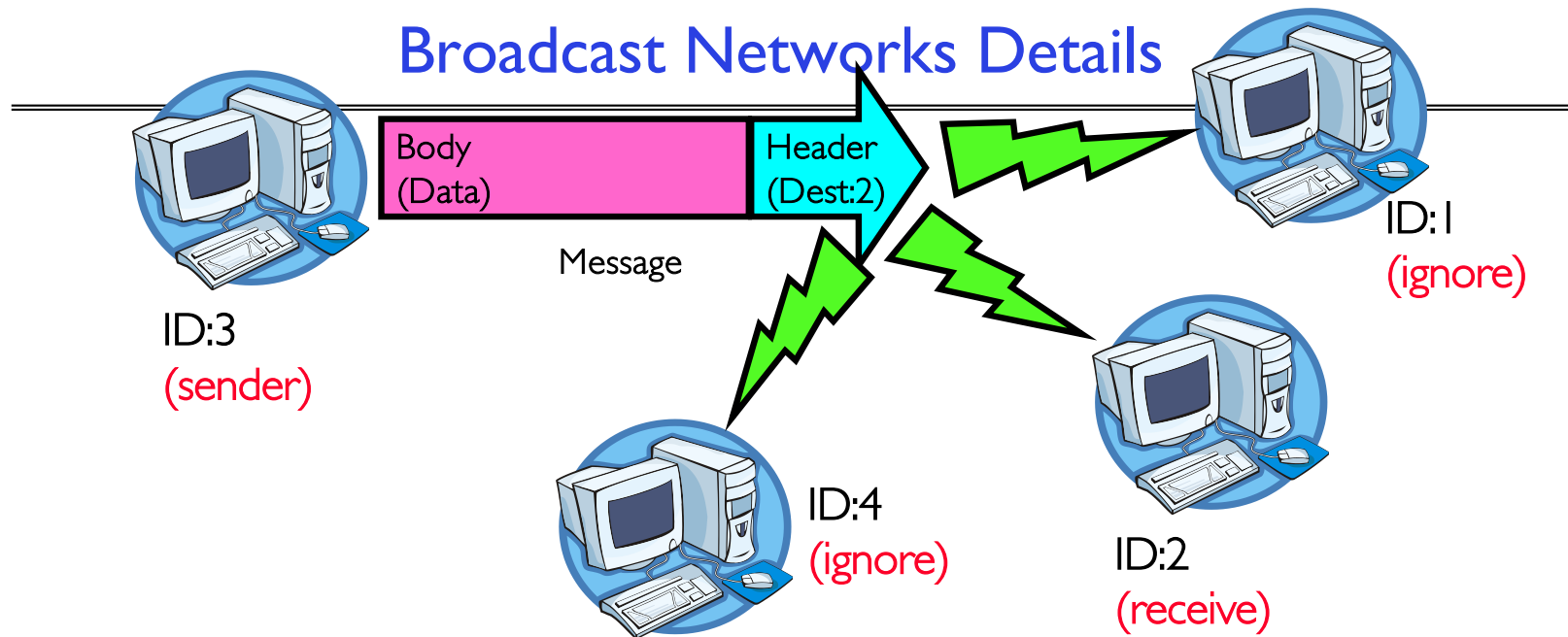


- Shared Medium can be a set of wires
 - » Inside a computer, this is called a bus
 - » All devices simultaneously connected to devices



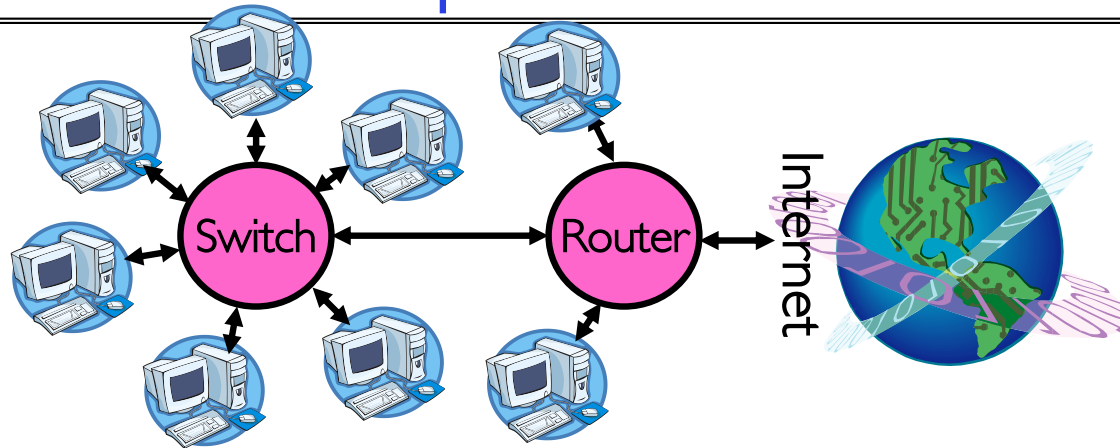
- Originally, Ethernet was a broadcast network
 - » All computers on local subnet connected to one another
- More examples (wireless: medium is air): cellular phones (GSM, CDMA, and LTE), WiFi

Broadcast Networks Details



- **Media Access Control (MAC) Address:**
 - 48-bit physical address for hardware interface
 - Every device (in the world!?) has a unique address
- **Delivery:** When you broadcast a packet, how does a receiver know who it is for? (packet goes to everyone!)
 - Put header on front of packet: [Destination MAC Addr | Packet]
 - Everyone gets packet, discards if not the target
 - In Ethernet, this check is done in hardware
 - » No OS interrupt if not for particular destination

Point-to-point networks

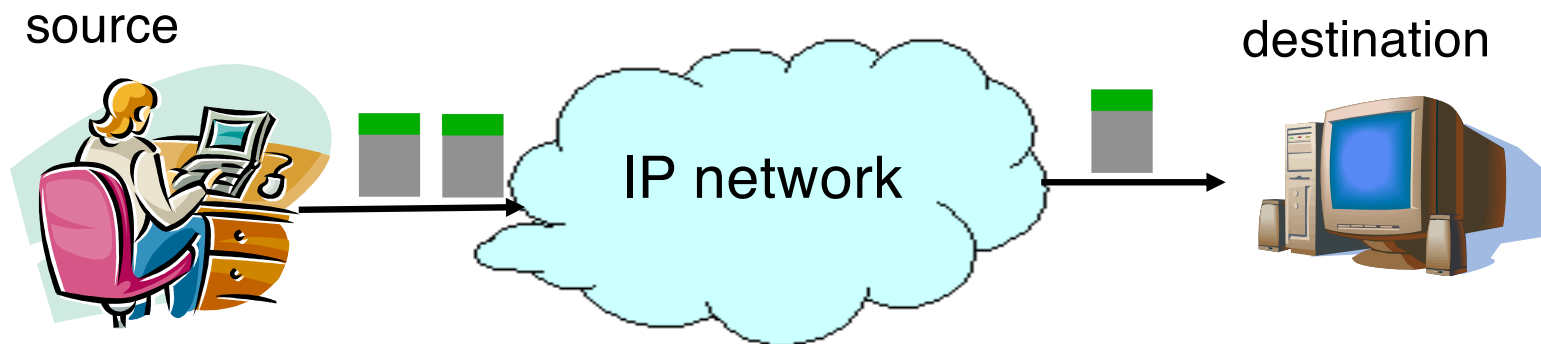


- Why have a shared bus at all? Why not simplify and only have point-to-point links + routers/switches?
 - Originally wasn't cost-effective, now hardware is cheap!
- **Point-to-point network:** a network in which every physical wire is connected to only two computers
- **Switch:** a bridge that transforms a shared-bus (broadcast) configuration into a point-to-point network
 - Adaptively figures out which ports have which MAC addresses
- **Router:** a device that acts as a junction between physical networks to transfer data packets among them
 - Routes between switching domains using (for instance) IP addresses

The Internet Protocol (IP)

Application
Present. Session
Transport
Network
Datalink
Physical

- Internet Protocol: Internet's network layer
- Service it provides: “Best-Effort” Packet Delivery
 - Tries it's “best” to deliver packet to its destination
 - Packets may be lost
 - Packets may be corrupted
 - Packets may be delivered out of order
- IP Is a Datagram service!
 - Routes across many physical switching domains (subnets)

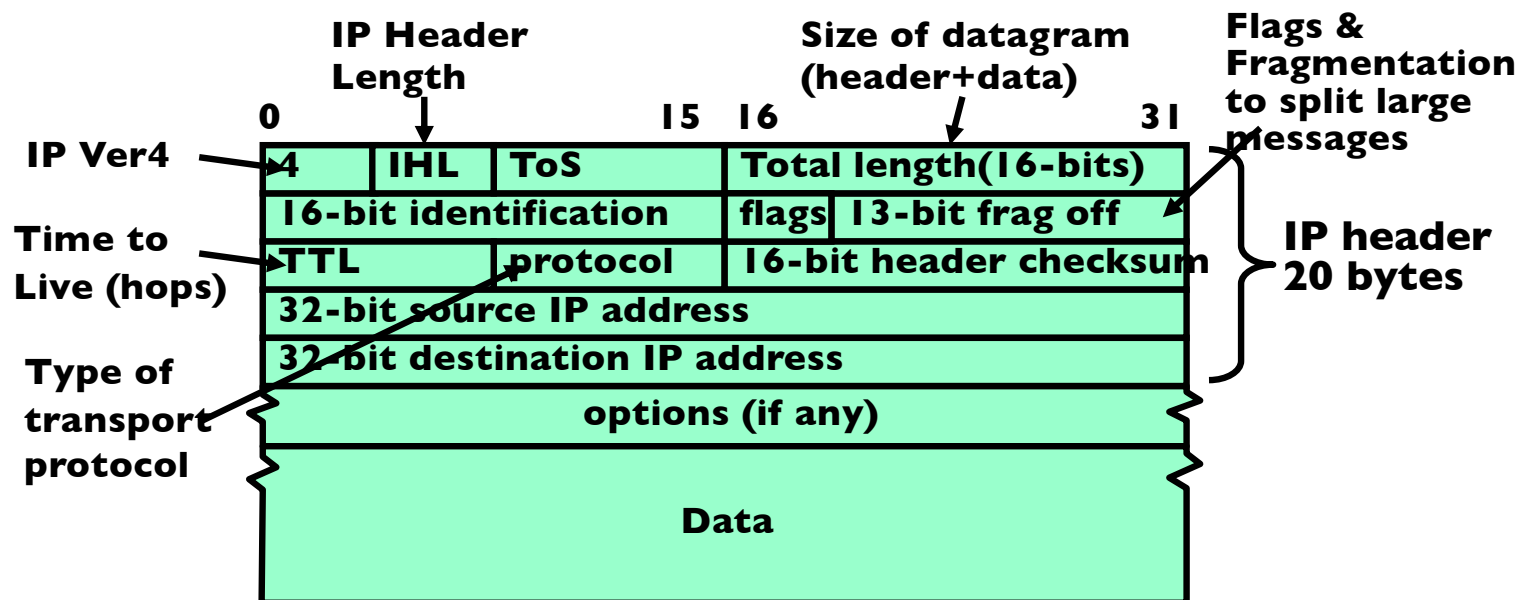


IPv4 Address Space

- **IP Address:** a 32-bit integer used as destination of IP packet
 - Often written as four dot-separated integers, with each integer from 0—255 (thus representing $8 \times 4 = 32$ bits)
 - Example CS file server is: 169.229.60.83 \equiv 0xA9E53C53
- **Internet Host:** a computer connected to the Internet
 - Host has one or more IP addresses used for routing
 - » Some of these may be private and unavailable for routing
 - Not every computer has a unique IP address
 - » Groups of machines may share a single IP address
 - » In this case, machines have private addresses behind a “Network Address Translation” (NAT) gateway
- **Subnet:** network connecting hosts with related IP addresses
 - A subnet is identified by 32-bit value, with the bits which differ set to zero, followed by a slash and a mask
 - » Example: 128.32.131.0/24 designates a subnet in which all the addresses look like 128.32.131.XX
 - » Same subnet: 128.32.131.0/255.255.255.0
 - **Mask:** The number of matching prefix bits
 - » Expressed as a single value (e.g., 24) or a set of ones in a 32-bit value (e.g., 255.255.255.0)
 - Often routing *within* subnet is by MAC address (smart switches)

IPv4 Packet Format

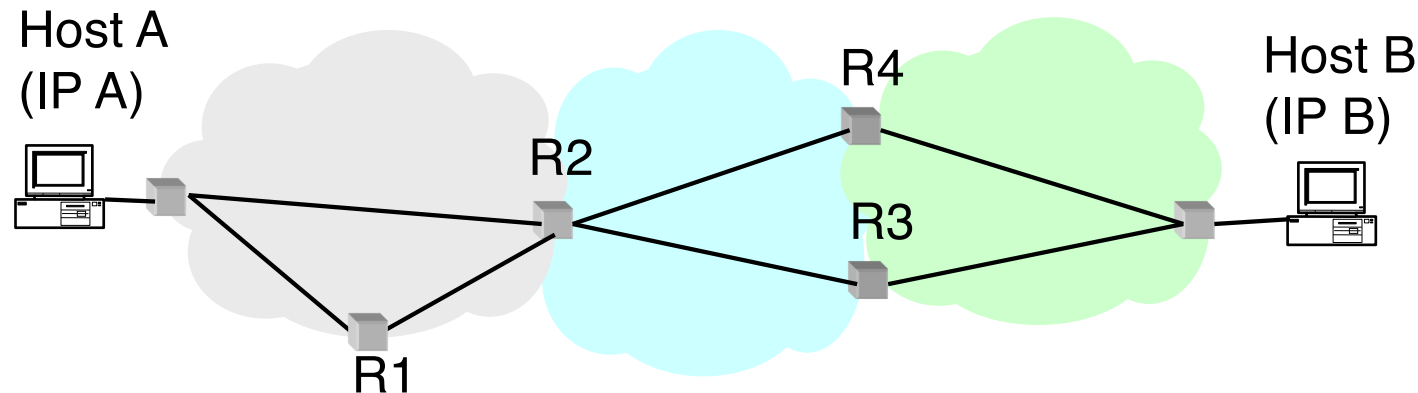
- IP Packet Format:



- IP Datagram**: an unreliable, unordered, packet sent from source to destination
 - Function of network – deliver datagrams!

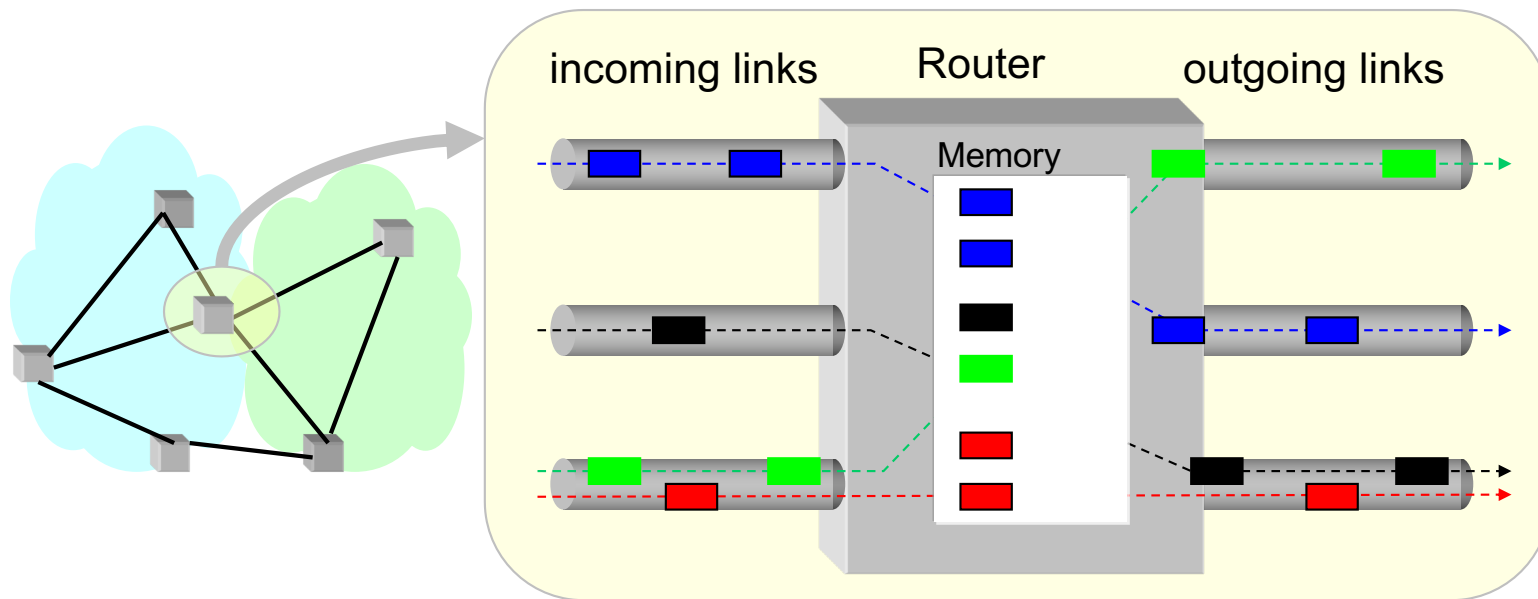
Wide Area Network

- **Wide Area Network (WAN)**: network that covers a broad area (e.g., city, state, country, entire world)
 - E.g., Internet is a WAN
- WAN connects multiple physical (datalink) layer networks (LANs)
- Datalink layer networks are connected by **routers**
 - Different LANs can use different communication technology (e.g., wireless, cellular, optics, wired)



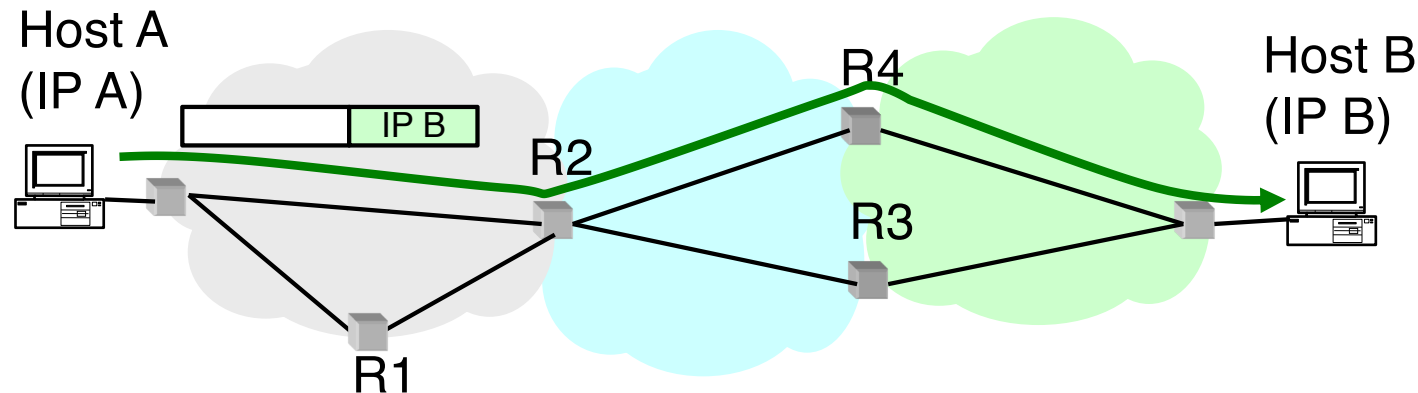
Routers

- **Forward** each packet received on an **incoming link** to an **outgoing link** based on packet's destination IP address (towards its destination)
- **Store & forward**: packets are buffered before being forwarded
- **Forwarding table**: mapping between IP address and the output link



Packet Forwarding

- Upon receiving a packet, a router
 - read the IP destination address of the packet
 - consults its forwarding table → output port
 - forwards packet to corresponding output port
- Default route (for subnets without explicit entries)
 - Forward to more authoritative router



IP Addresses vs. MAC Addresses

- Why not use MAC addresses for routing?
 - Doesn't scale
- Analogy
 - MAC address → SSN
 - IP address → (unreadable) home address
- MAC address: uniquely associated with device for the entire lifetime of the device
- IP address: changes as the device location changes
 - Your notebook IP address at school is different from home



IP Addresses vs. MAC Addresses

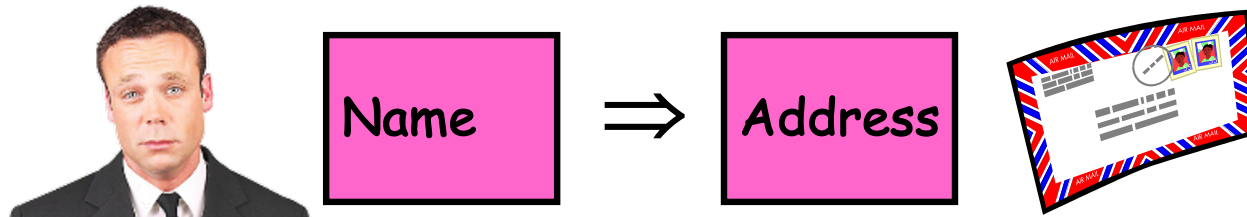
- Why does packet forwarding using IP addr. scale?
- Because IP addresses can be aggregated
 - E.g., all IP addresses at UC Berkeley start with **0xA9E5**, i.e., any address of form 0xA9E5**** belongs to Berkeley
 - Thus, a router in NY needs to keep a **single** entry for **all** hosts at Berkeley
 - If we were using MAC addresses the NY router would need to maintain **an entry for every** Berkeley host!!
- Analogy:
 - Give this letter to person with SSN: 123-45-6789 vs.
 - Give this letter to “John Smith, 123 First Street, LA, US”



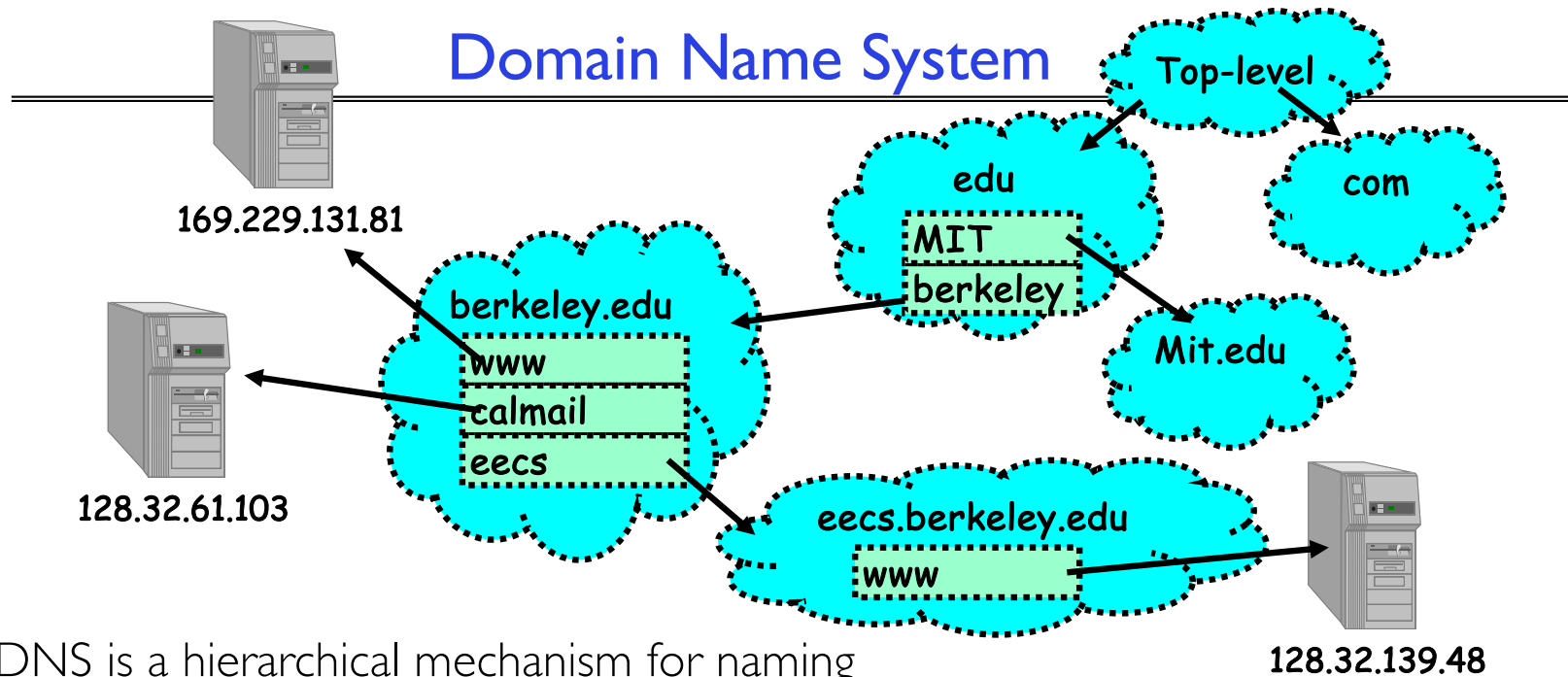
Administrivia

- Midterm 3: Thursday 4/28: 7-9PM
 - All course material
 - Review session Monday 4/25 1-3PM

Naming in the Internet



- How to map human-readable names to IP addresses?
 - E.g. `www.berkeley.edu` \Rightarrow `128.32.139.48`
 - E.g. `www.google.com` \Rightarrow different addresses depending on location, and load
- Why is this necessary?
 - IP addresses are hard to remember
 - IP addresses change:
 - » Say, Server 1 crashes gets replaced by Server 2
 - » Or – `google.com` handled by different servers
- Mechanism: Domain Naming System (DNS)



- DNS is a hierarchical mechanism for naming
 - Name divided in domains, right to left: `www.eecs.berkeley.edu`
- Each domain owned by a particular organization
 - Top level handled by ICANN (Internet Corporation for Assigned Numbers and Names)
 - Subsequent levels owned by organizations
- Resolution: series of queries to successive servers
- Caching: queries take time, so results cached for period of time

How Important is Correct Resolution?

- If attacker manages to give incorrect mapping:
 - Can get someone to route to server, thinking that they are routing to a different server
 - » Get them to log into “bank” – give up username and password
- Is DNS Secure?
 - Definitely a weak link
 - » What if “response” returned from different server than original query?
 - » Get person to use incorrect IP address!
 - Attempt to avoid substitution attacks:
 - » Query includes random number which must be returned
- In July 2008, hole in DNS security located!
 - Dan Kaminsky (security researcher) discovered an attack that broke DNS globally
 - » One person in an ISP convinced to load particular web page, then all users of that ISP end up pointing at wrong address
 - High profile, highly advertised need for patching DNS
 - » Big press release, lots of mystery
 - » Security researchers told no speculation until patches applied

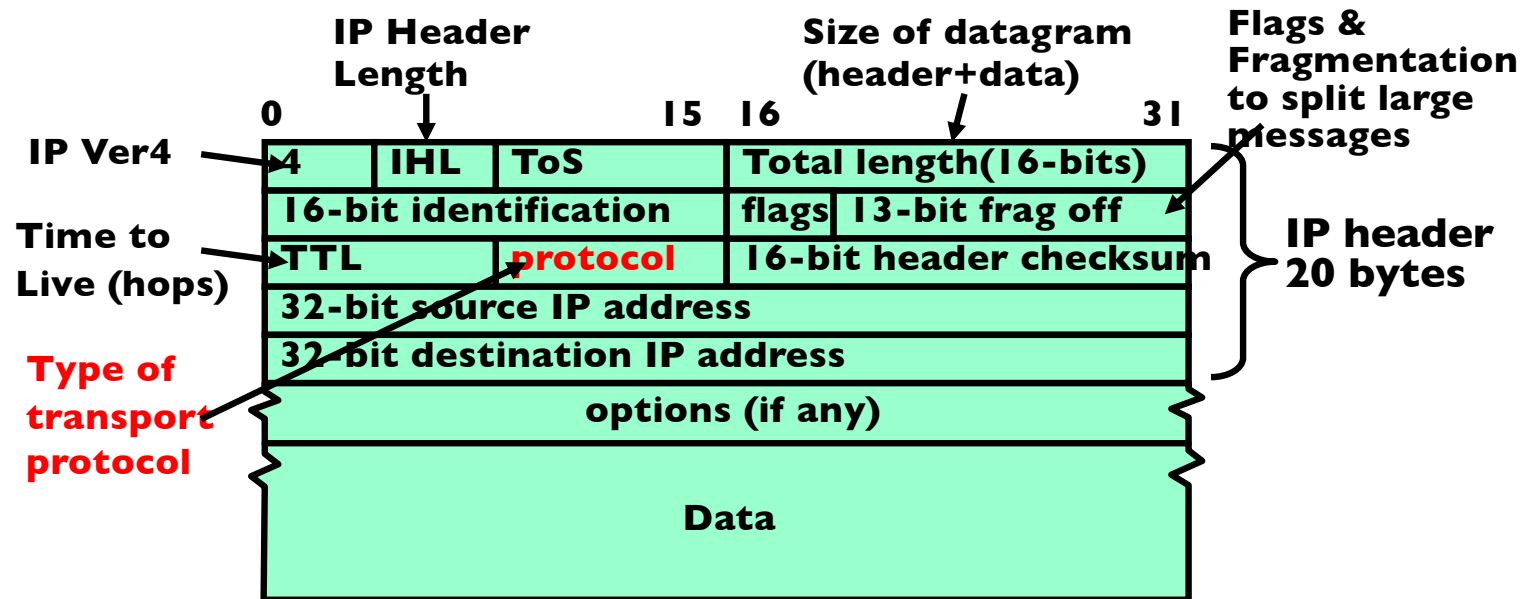
Network Layering

- **Layering**: building complex services from simpler ones
 - Each layer provides services needed by higher layers by utilizing services provided by lower layers
- The physical/link layer is pretty limited
 - Packets are of limited size (called the “**Maximum Transfer Unit** or MTU: often 200-1500 bytes in size)
 - Routing is limited to within a physical link (wire) or perhaps through a switch
- Our goal in the following is to show how to construct a secure, ordered, message service routed to anywhere:

Physical Reality: Packets	Abstraction: Messages
Limited Size (MTU)	Arbitrary Size
Unordered (sometimes)	Ordered
Unreliable	Reliable
Machine-to-machine	Process-to-process
Only on local area net	Routed anywhere
Asynchronous	Synchronous
Insecure	Secure

Recall: IPv4 Packet Format

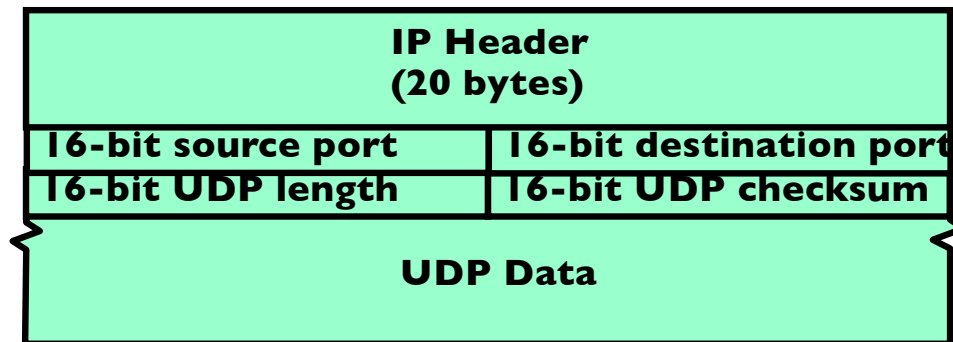
- IP Packet Format:



- IP Datagram**: an unreliable, unordered, packet sent from source to destination
 - Function of network – deliver datagrams!

Building a messaging service on IP

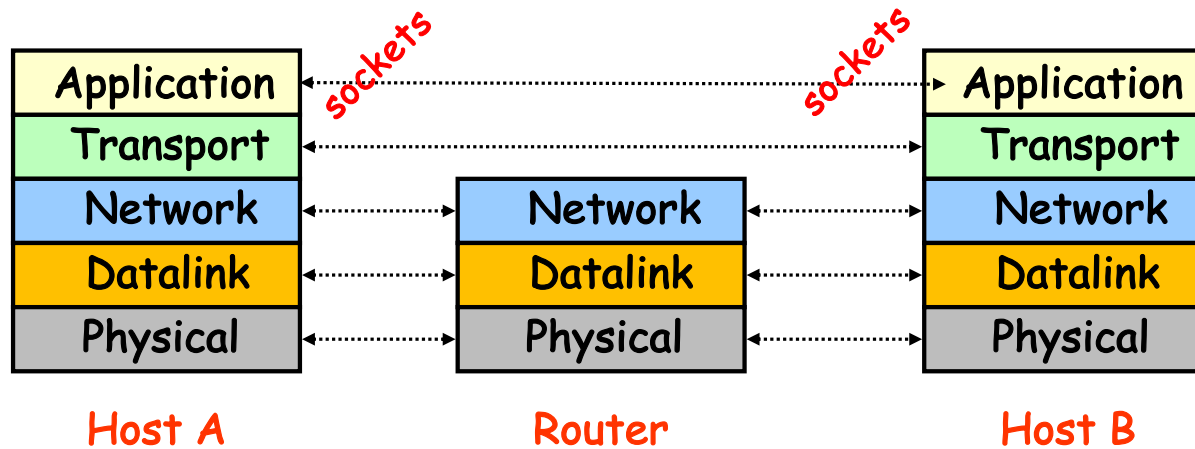
- Process to process communication
 - Basic routing gets packets from machine→machine
 - What we really want is routing from process→process
 - » Add “**ports**”, which are 16-bit identifiers
 - » A communication channel (**connection**) defined by 5 items:
[source addr, source port, dest addr, dest port, protocol]
- For example: The Unreliable Datagram Protocol (UDP)
 - Layered on top of basic IP (**IP Protocol 17**)
 - » **Datagram**: an unreliable, unordered, packet sent from source user → dest user (Call it UDP/IP)



- Important aspect: low overhead!
 - » Often used for high-bandwidth video streams
 - » Many uses of UDP considered “anti-social” – none of the “well-behaved” aspects of (say) TCP/IP

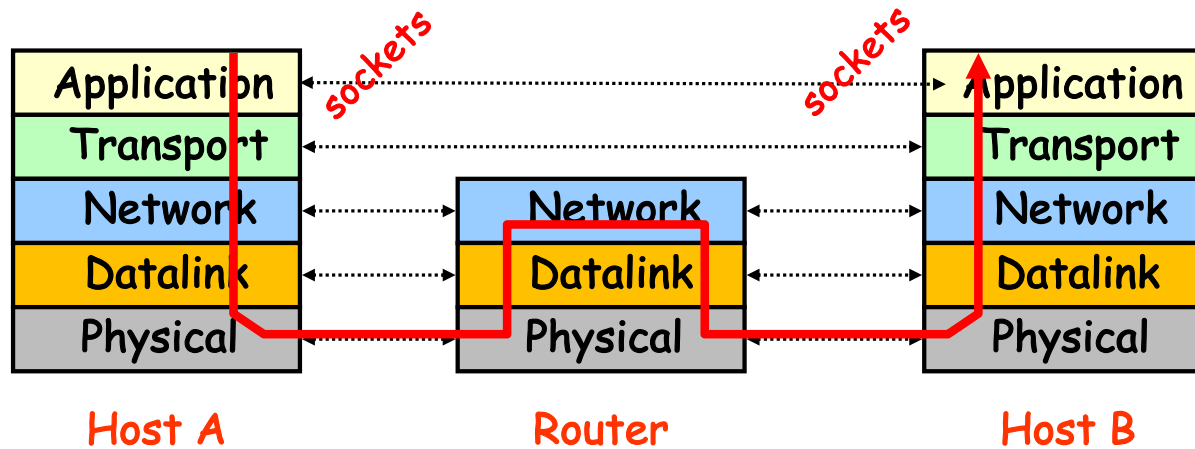
Internet Architecture: Five Layers

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts

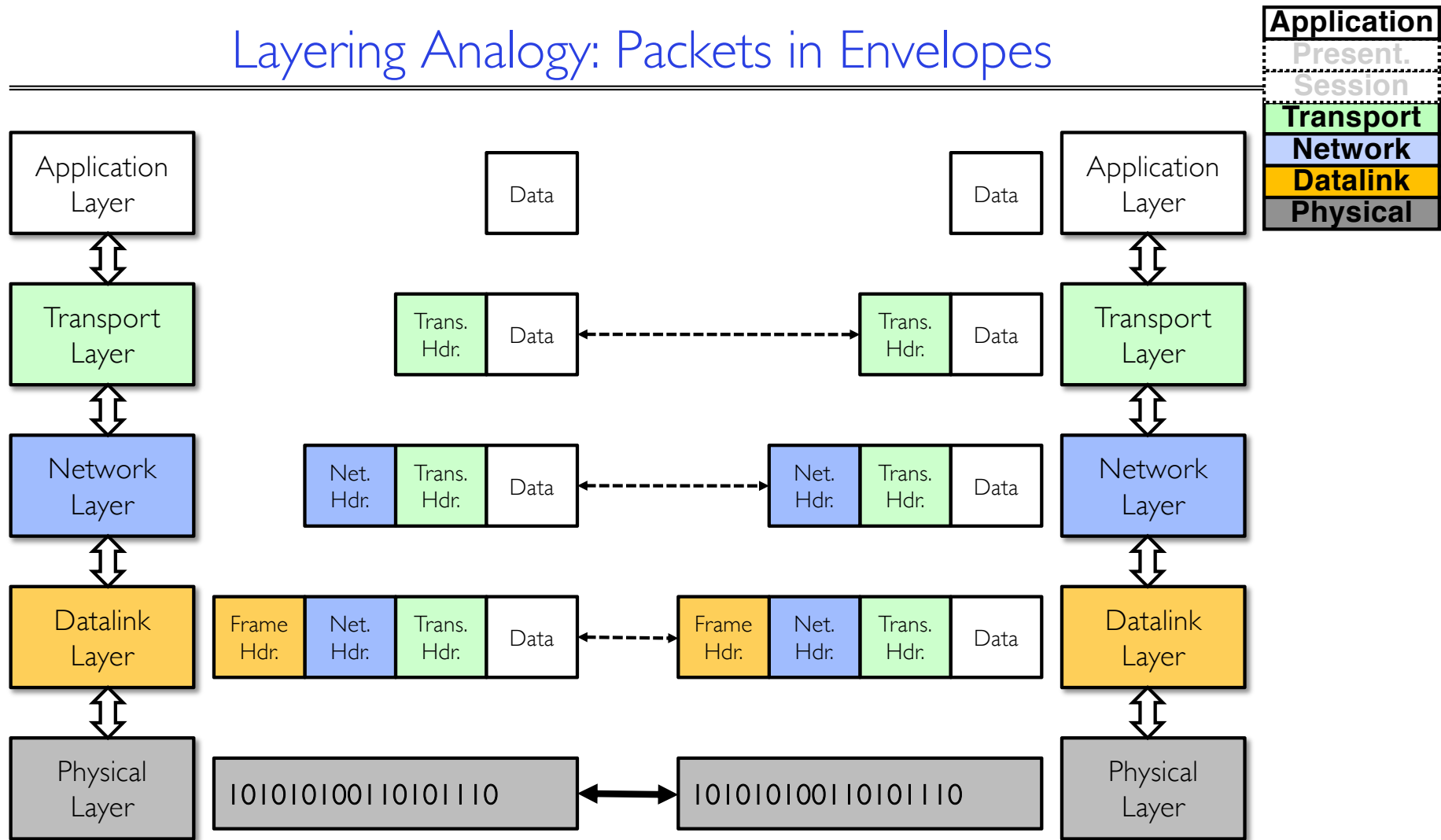


Internet Architecture: Five Layers

- Communication goes down to physical network
- Then from network peer to peer
- Then up to relevant layer



Layering Analogy: Packets in Envelopes

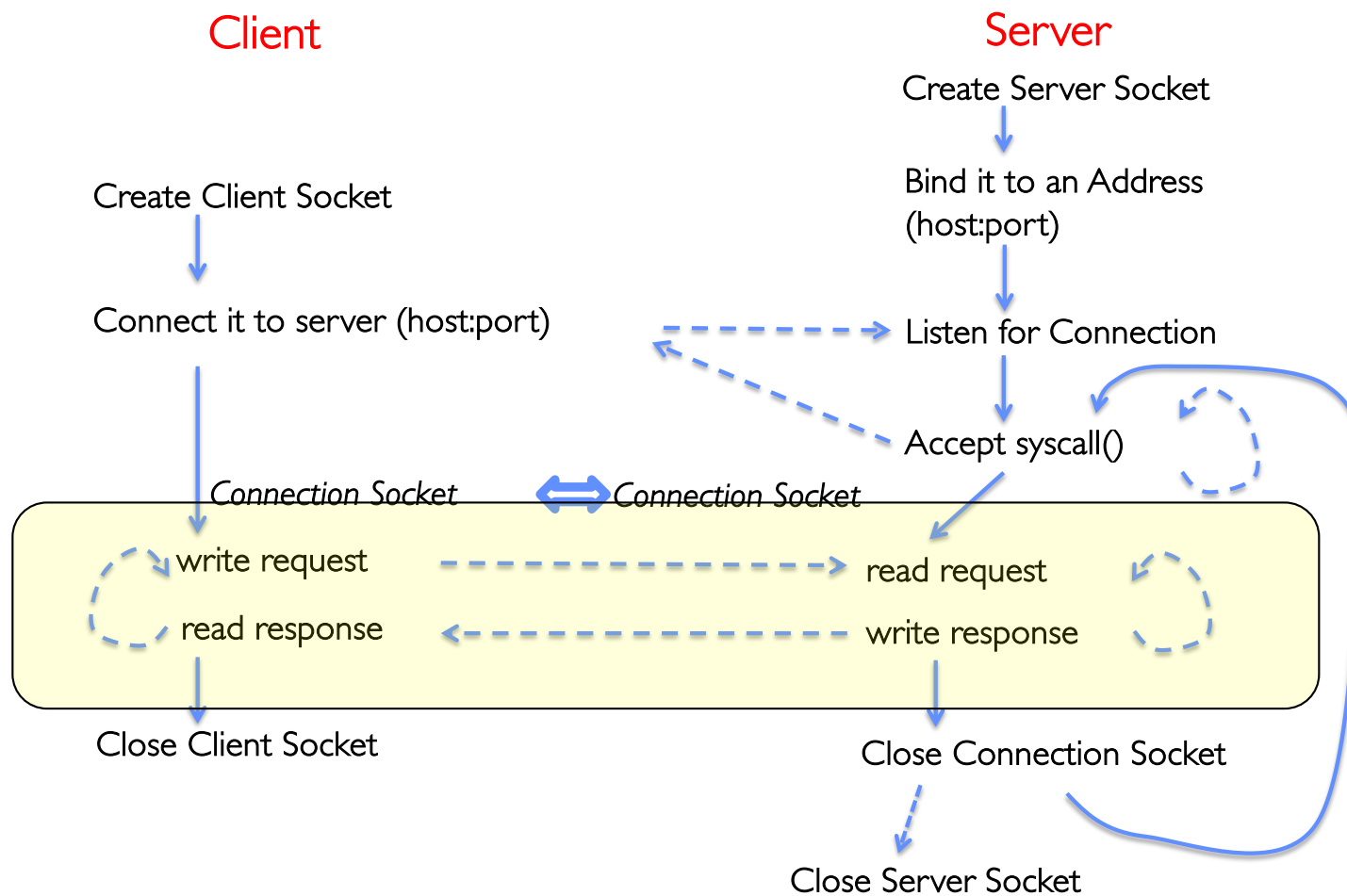


Internet Transport Protocols

Application
Present. Session
Transport
Network
Datalink
Physical

- Datagram service (**UDP**): IP Protocol 17
 - No-frills extension of “best-effort” IP
 - Multiplexing/Demultiplexing among processes
- Reliable, in-order delivery (**TCP**): IP Protocol 6
 - Connection set-up & tear-down
 - Discarding corrupted packets (segments)
 - Retransmission of lost packets (segments)
 - Flow control
 - Congestion control
- Other examples:
 - DCCP (33), Datagram Congestion Control Protocol
 - RDP (26), Reliable Data Protocol
 - SCTP (132), Stream Control Transmission Protocol

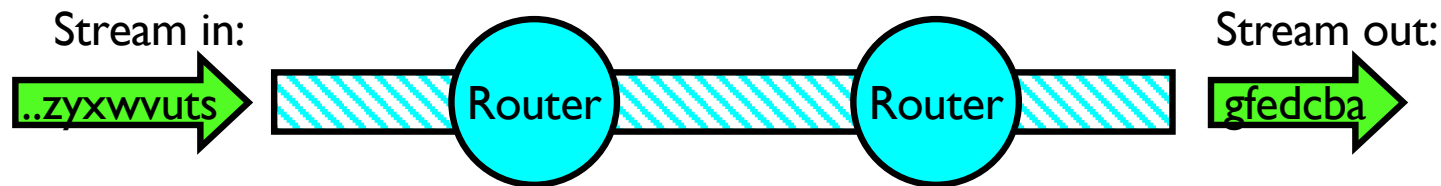
Recall: Sockets in concept



Reliable Message Delivery: the Problem

- All physical networks can garble and/or drop packets
 - Physical media: packet not transmitted/received
 - » If transmit close to maximum rate, get more throughput – even if some packets get lost
 - » If transmit at lowest voltage such that error correction just starts correcting errors, get best power/bit
 - Congestion: no place to put incoming packet
 - » Point-to-point network: insufficient queue at switch/router
 - » Broadcast link: two host try to use same link
 - » In any network: insufficient buffer space at destination
 - » Rate mismatch: what if sender send faster than receiver can process?
- Reliable Message Delivery on top of Unreliable Packets
 - Need some way to make sure that packets actually make it to receiver
 - » Every packet received at least once
 - » Every packet received at most once
 - Can combine with ordering: every packet received by process at destination exactly once and in order

Transmission Control Protocol (TCP)

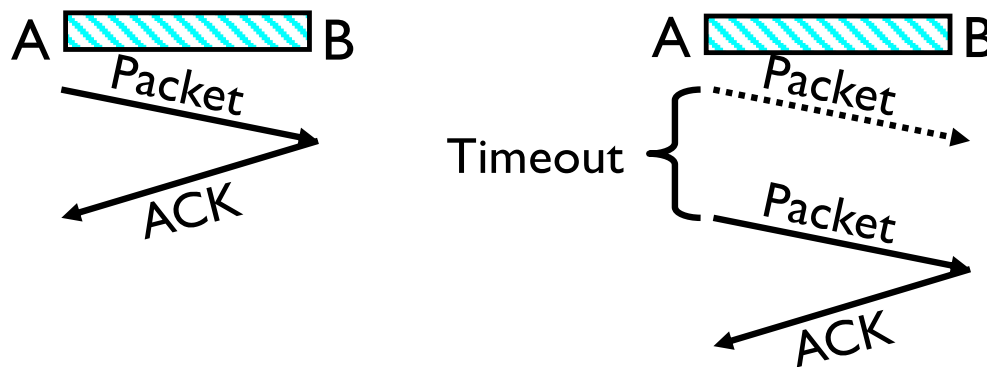


- Transmission Control Protocol (TCP)
 - TCP (IP Protocol 6) layered on top of IP
 - Reliable byte stream between two processes on different machines over Internet (read, write, flush)
- TCP Details
 - Fragments byte stream into packets, hands packets to IP
 - » IP may also fragment by itself
 - Uses window-based acknowledgement protocol (to minimize state at sender and receiver)
 - » “Window” reflects storage at receiver – sender shouldn’t overrun receiver’s buffer space
 - » Also, window should reflect speed/capacity of network – sender shouldn’t overload network
 - Automatically retransmits lost packets
 - Adjusts rate of transmission to avoid congestion
 - » A “good citizen”

Problem: Dropped Packets

- All physical networks can garble or drop packets
 - Physical hardware problems (bad wire, bad signal)
- Therefore, IP can garble or drop packets
 - It doesn't repair this itself (end-to-end principle!)
- Building reliable message delivery
 - Confirm that packets aren't garbled
 - Confirm that packets arrive **exactly once**

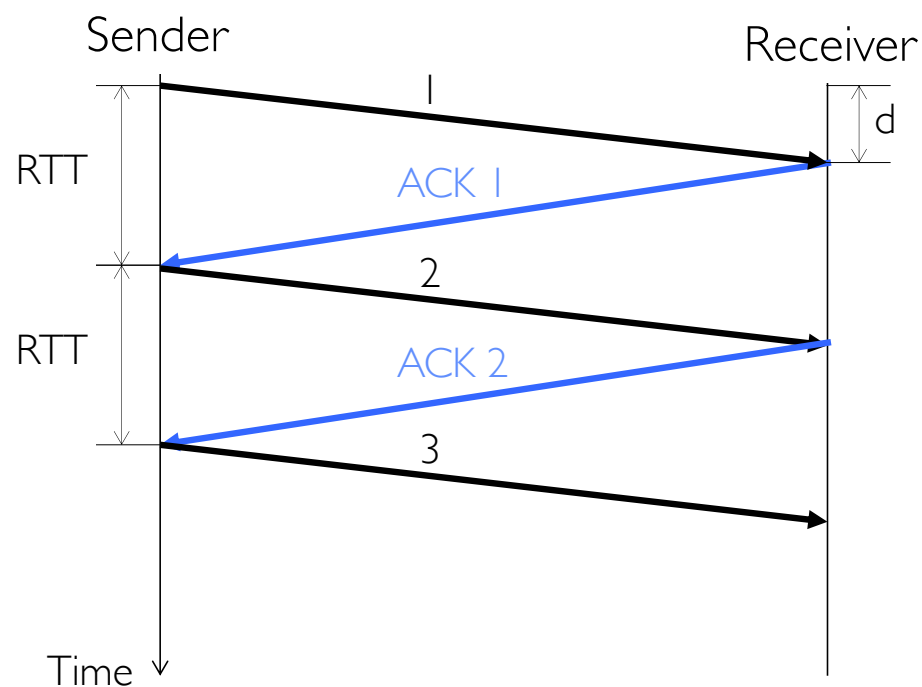
Using Acknowledgements



- How to ensure transmission of packets?
 - Detect garbling at receiver via checksum, discard if bad
 - Receiver acknowledges (by sending “ACK”) when packet received properly at destination
 - Timeout at sender: if no ACK, retransmit
- Some questions:
 - If the sender doesn't get an ACK, does that mean the receiver didn't get the original message?
 - » No
 - What if ACK gets dropped? Or if message gets delayed?
 - » Sender doesn't get ACK, retransmits, Receiver gets message twice, ACK each

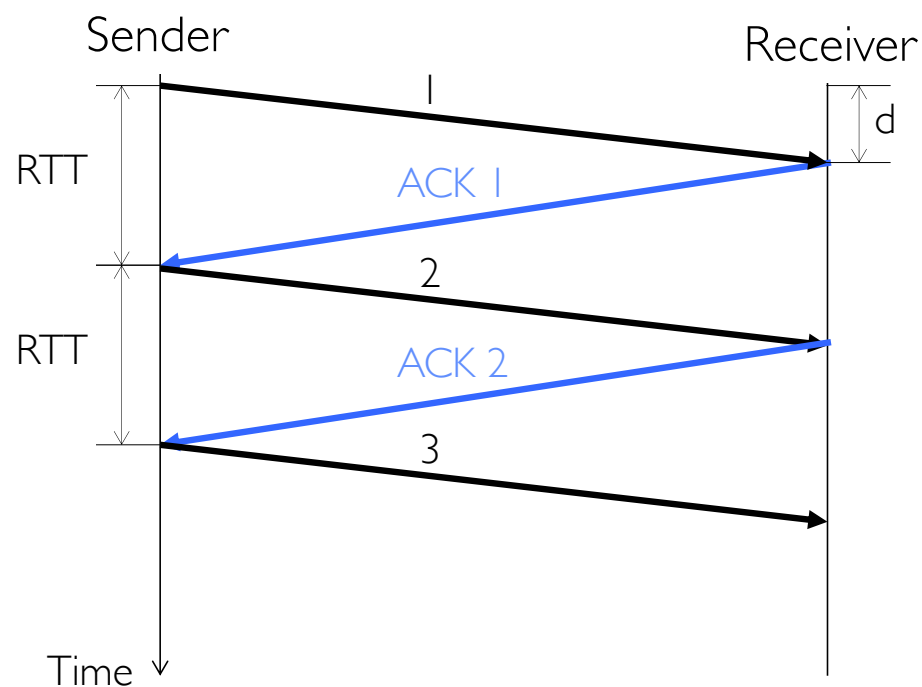
Stop-and-Wait (No Packet Loss)

- Send; wait for ACK; repeat
- Round Trip Time (RTT): time it takes a packet to travel from sender to receiver and back
 - One-way latency (d): one way delay from sender and receiver
- For symmetric latency,
 $RTT = 2d$



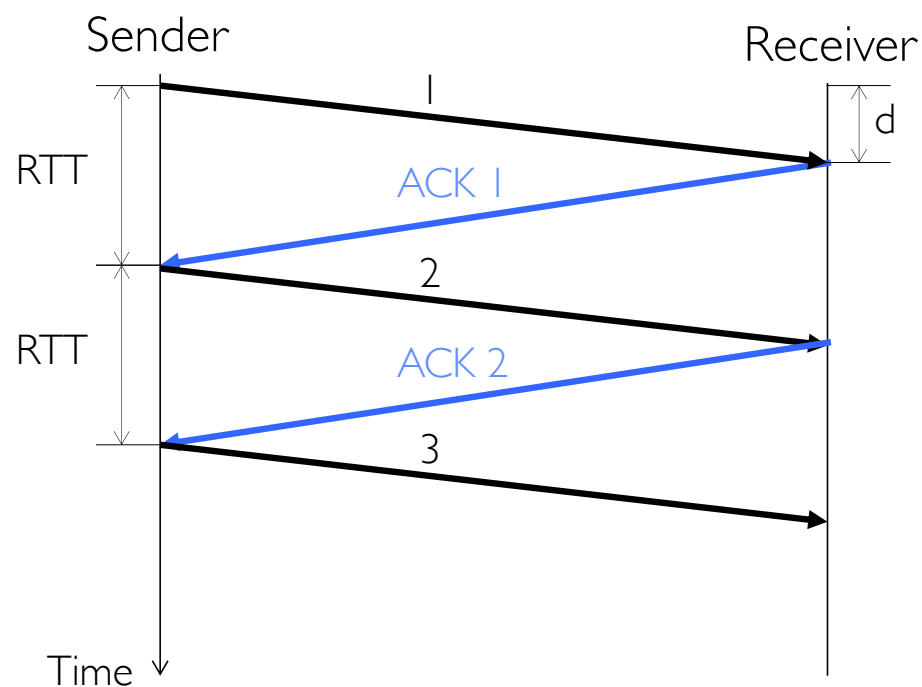
Stop-and-Wait (No Packet Loss)

- How fast can you send data?
- Little's Law applied to the network:
$$n = B \cdot \text{RTT}$$
- For Stop-and-Wait, $n = 1$ packet
- So bandwidth is 1 packet per RTT
 - Depends only on latency, not network capacity (!)



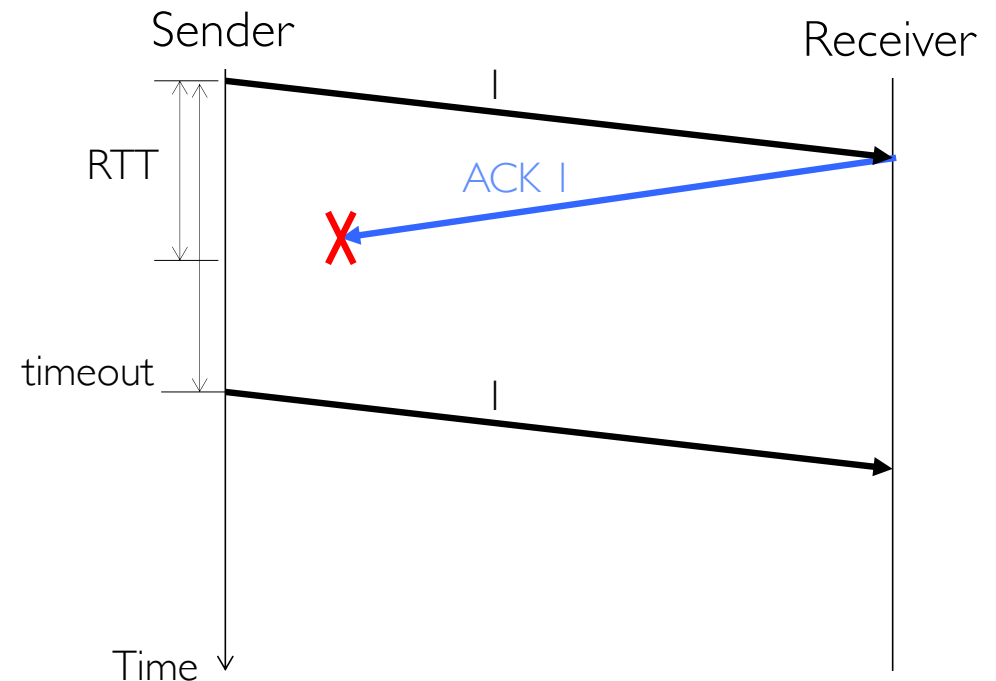
Stop-and-Wait (No Packet Loss)

- So bandwidth is 1 packet per RTT
 - Depends only on latency, not network capacity (!)
- Suppose $RTT = 100$ ms and 1 packet is 1500 bytes
- Throughput = $\frac{1500 \cdot 8}{0.1} = 120$ Kbps
- Very inefficient if we have a 100 Mbps link!



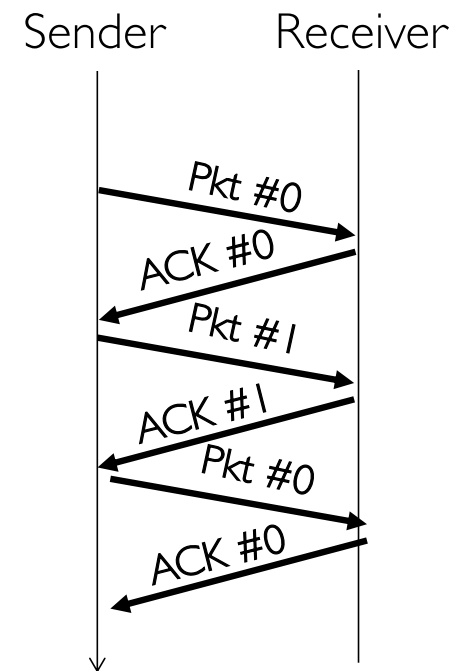
Stop-and-Wait with Packet Loss

- Loss recovery relies on timeouts
- How to choose a good timeout?
 - Too short – lots of duplication
 - Too long – packet loss is really disruptive!
- How to deal with duplication?
 - Retransmission certainly opens up the possibility for



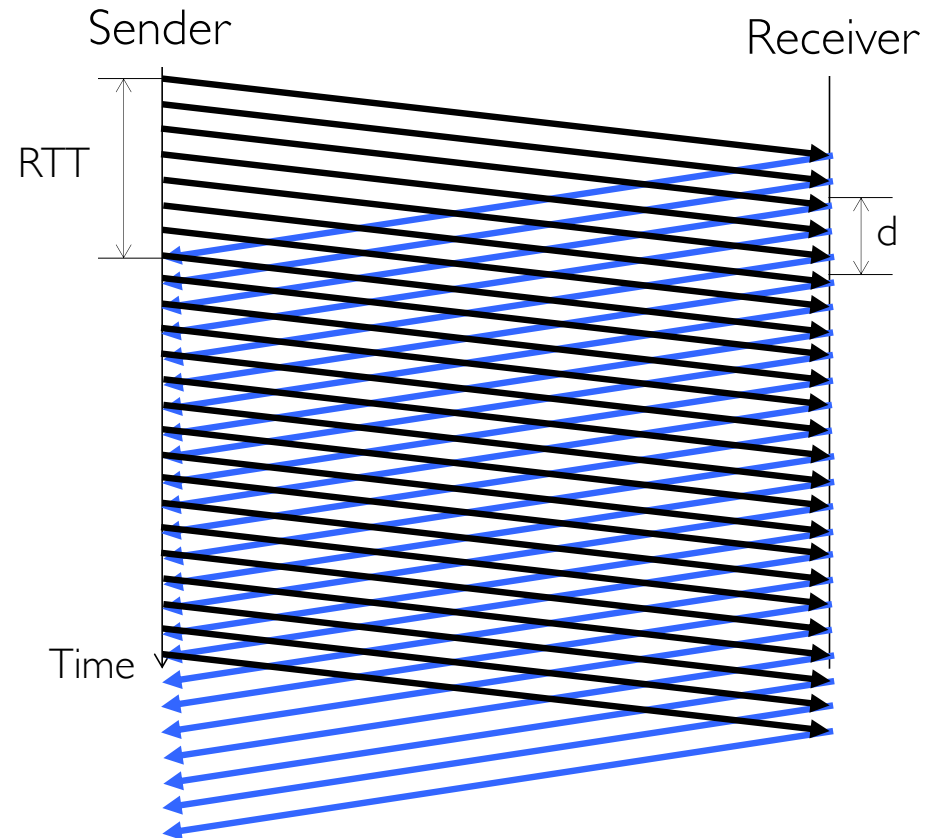
How to Deal with Message Duplication?

- Solution: put sequence number in message to identify re-transmitted packets
 - Receiver checks for duplicate number's; Discard if detected
- Requirements:
 - Sender keeps copy of unACK'd messages
 - » Easy: only need to buffer messages
 - Receiver tracks possible duplicate messages
 - » Hard: when ok to forget about received message?
- **Alternating-bit protocol:**
 - Send one message at a time; don't send next message until ACK received
 - Sender keeps last message; receiver tracks sequence number of last message received
- Pros: simple, small overhead
- Con: doesn't work if network can delay or duplicate messages arbitrarily



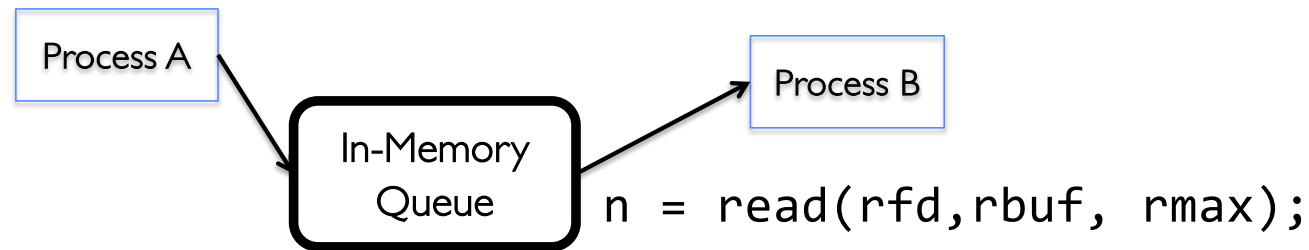
Advantages of Moving Away From Stop-and-Wait

- Larger space of acknowledgements
 - Pipelining: don't wait for ACK before sending next packet
- ACKs serve dual purpose:
 - Reliability: Confirming packet received
 - Ordering: Packets can be reordered at destination
- How much data is in flight now?
 - Bytes in-flight: $W_{\text{send}} = \text{RTT} \times B$
 - Here B is in “bytes/second”
 - $W_{\text{send}} \equiv$ Sender's “Window Size”
 - Packets in flight = $(W_{\text{send}} / \text{packet size})$
- How long does the sender have to keep the packets around?
- How long does the receiver have to keep the packets' data?
- What if sender is sending packets faster than the receiver can process the data?



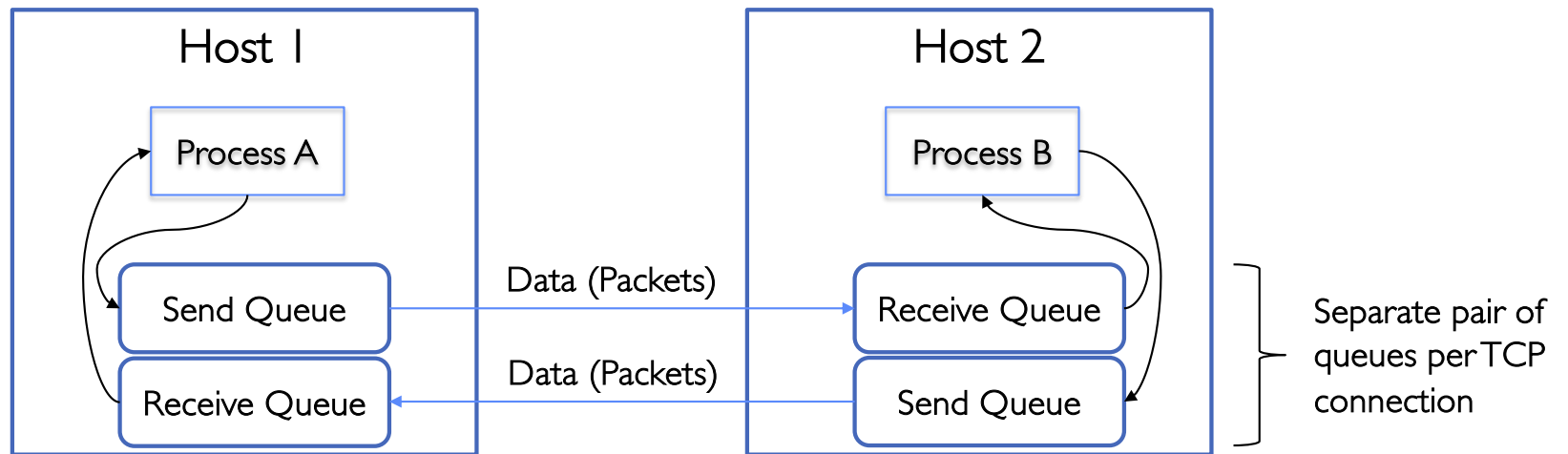
Recall: Communication Between Processes

```
write(wfd, wbuf, wlen);
```



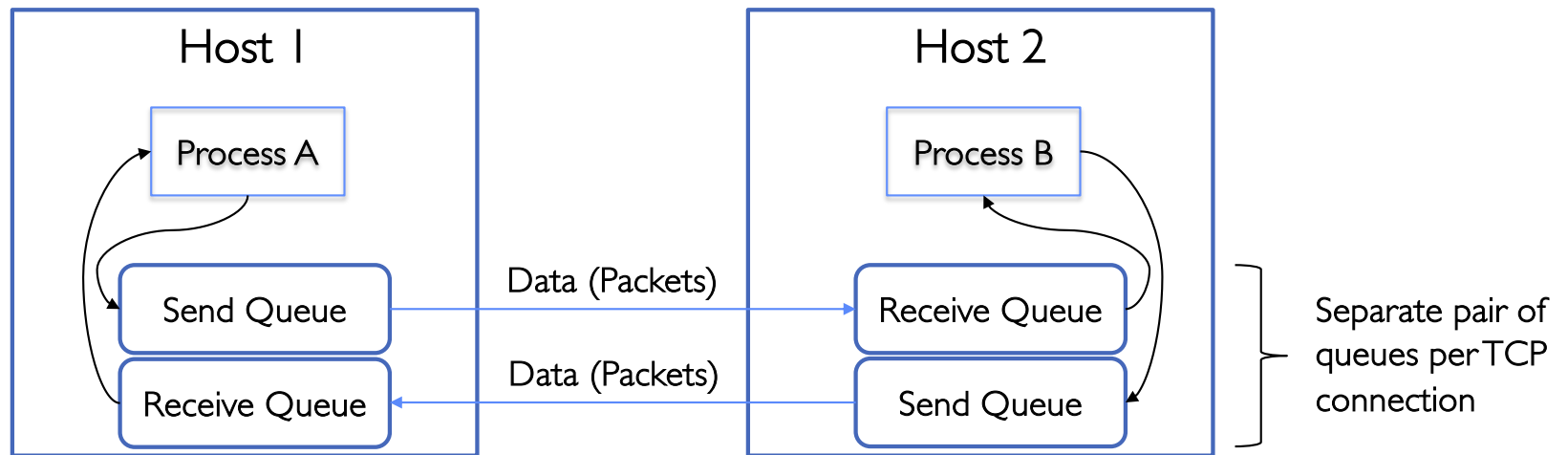
- Data written by A is held in memory until B reads it
- Queue has a fixed capacity
 - Writing to the queue blocks if the queue is full
 - Reading from the queue blocks if the queue is empty
- POSIX provides this abstraction in the form of *pipes*

Buffering in a TCP Connection



- A single TCP connection needs *four* in-memory queues:
 - Send buffer: add data on write syscall, remove data when ACK received
 - Receive buffer: add data when packets received, remove data on read syscall

Window Size: Space in Receive Queue



- A host's *window size* for a TCP connection is how much remaining space it has in its receive queue
- A host advertises its window size in every TCP packet it sends!
- Sender never sends more than receiver's advertised window size

Sliding Window Protocol

- TCP sender knows receiver's window size, and aims never to exceed it
- But packets that it previously send may arrive, filling the window size!

Rule: TCP sender ensures that:

Number of Sent but UnACKed Bytes < Receiver's Advertised Window Size

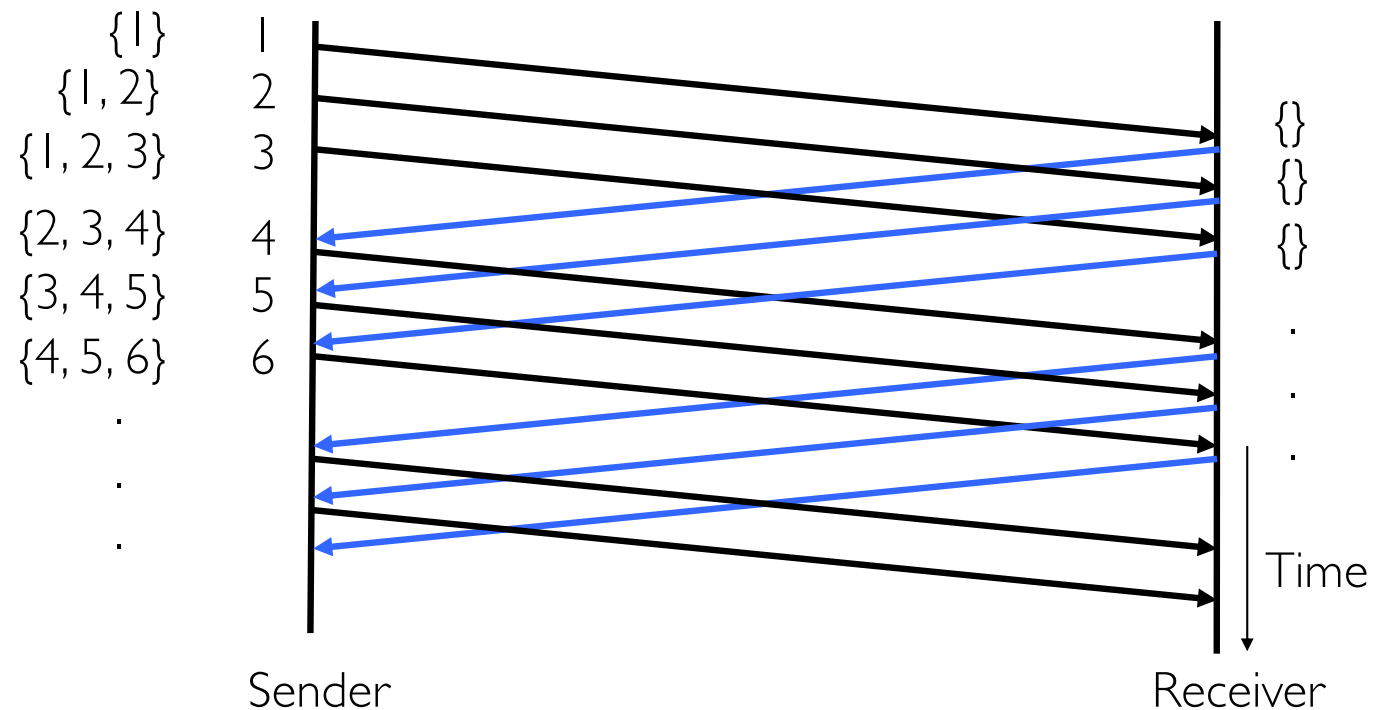
- Can send new packets as long as sent-but-unacked packets haven't already filled the advertised window size

Sliding Window (No Packet Loss)

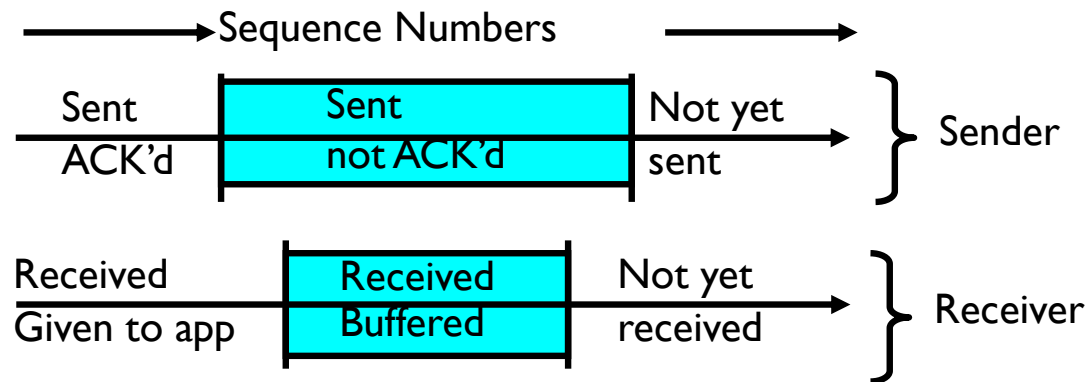
- Example: Window size (w) = 3 packets
- Window size to fill link is given by:
 $w = B_{pkt} \cdot RTT$
- $B_{pkt} \equiv \text{Packets/sec}$
- Little's Law once again!
- For TCP, window is in bytes, not packets

Unacked packets
that sender sent

Out-of-seq packets
in receiver's window

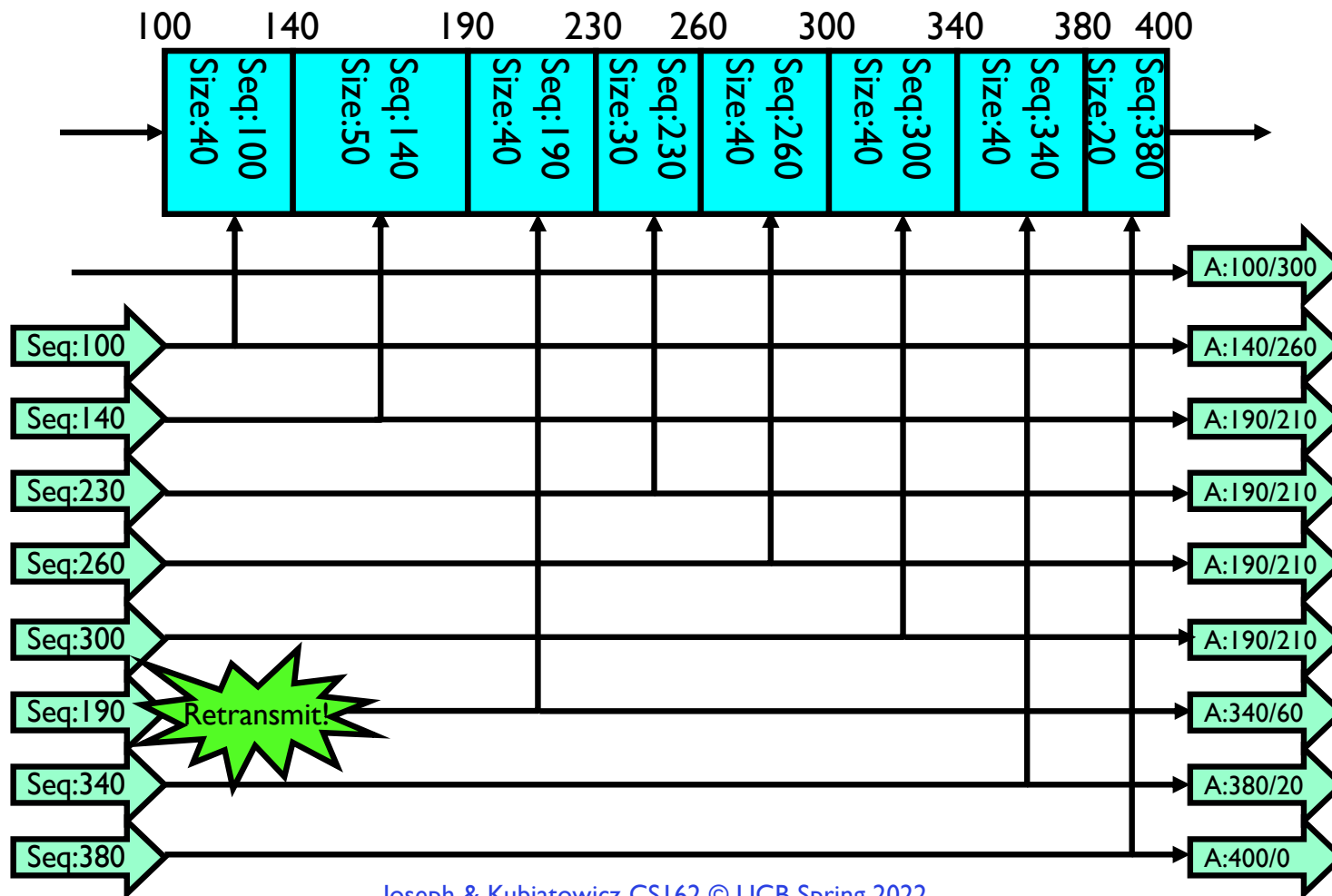


TCP Windows and Sequence Numbers: PER BYTE!



- Sender has three regions:
 - Sequence regions
 - » sent and ACK'd
 - » sent and not ACK'd
 - » not yet sent
 - Window (colored region) adjusted by sender
- Receiver has three regions:
 - Sequence regions
 - » received and ACK'd (given to application)
 - » received and buffered
 - » not yet received (or discarded because out of order)

Window-Based Acknowledgements (TCP)



Summary

- **TCP**: Reliable byte stream between two processes on different machines over Internet (read, write, flush)
 - Uses window-based acknowledgement protocol
 - Congestion-avoidance dynamically adapts sender window to account for congestion in network