Chapter 17: Networking

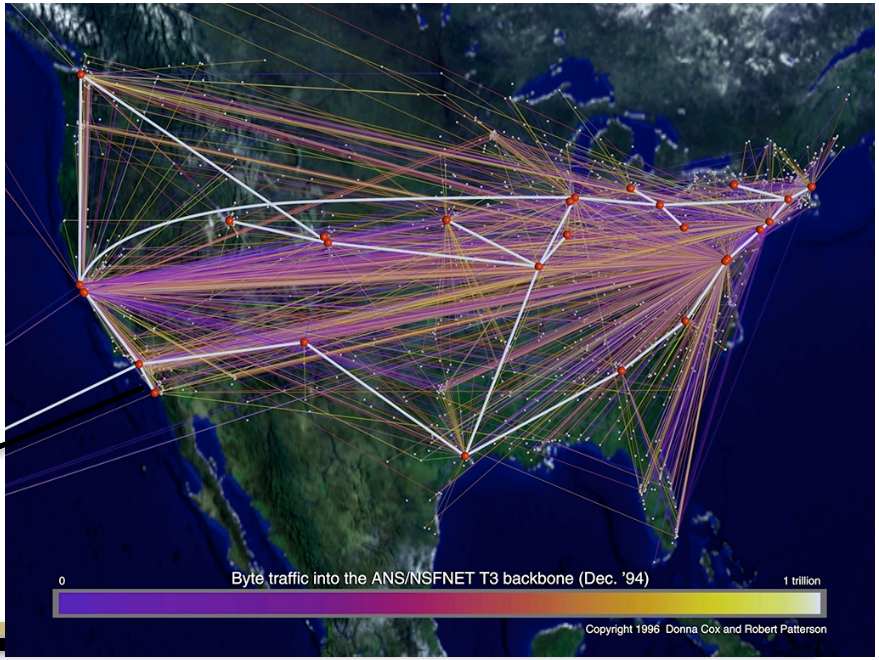
CSCI 3753 Operating Systems

Prof. Rick Han

University of Colorado Boulder

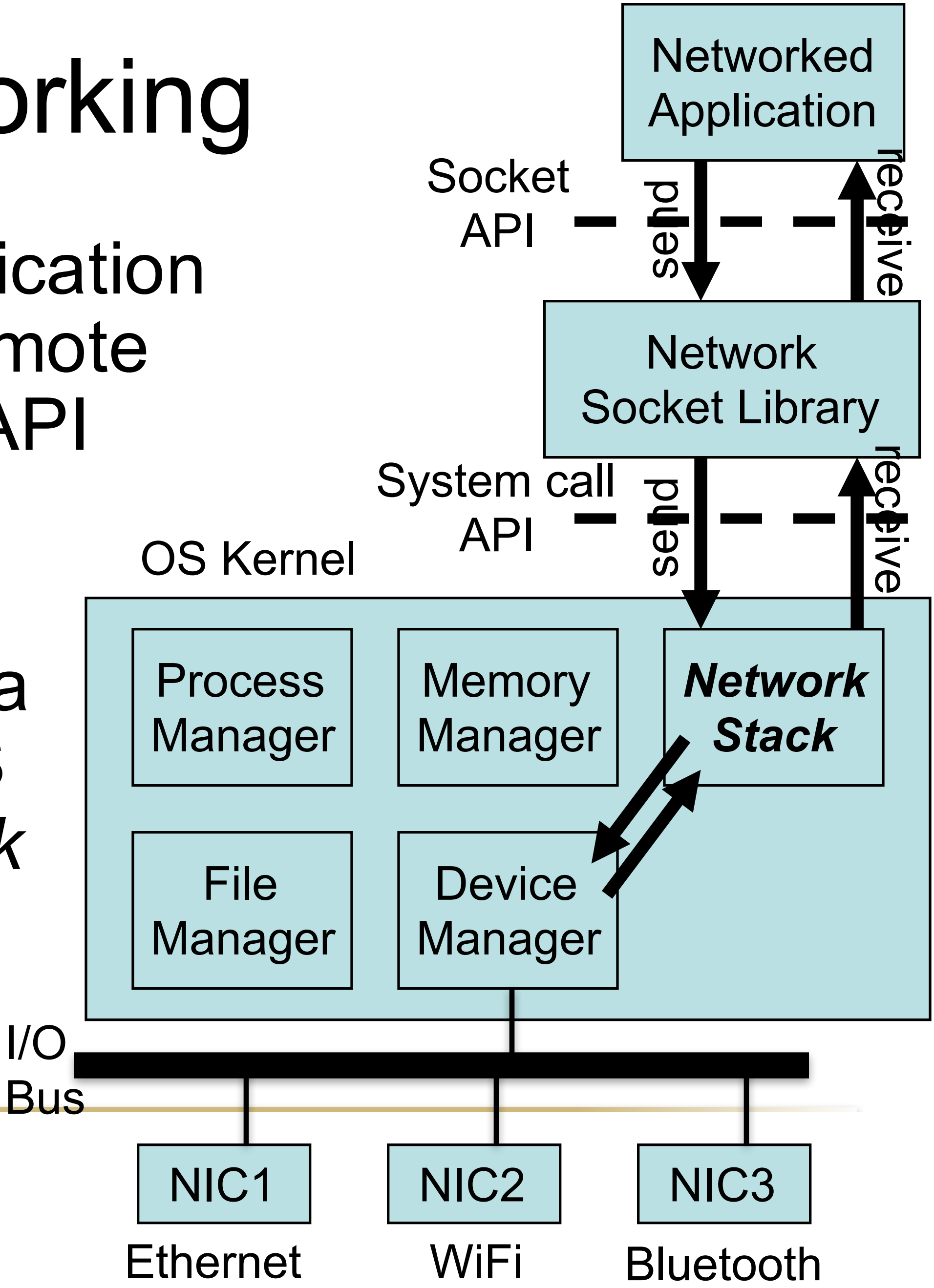
Networking

- Applications today leverage the Internet to send and receive data
  - Web browser requests pages from a Web server, e.g. Web search
  - P2P systems
  - Streaming video
  - Social networks
  - Mobile apps


0
Byte traffic into the ANS/NSFNET T3 backbone (Dec. '94)
1 trillion
Copyright 1996 Donna Cox and Robert Patterson

Networking

- Every networked application communicates to a remote process via a socket API
  - Send(message)
  - Receive(message)
- Socket library talks via system call API to OS kernel's *network stack*
  - Send(message)
  - Receive(message)



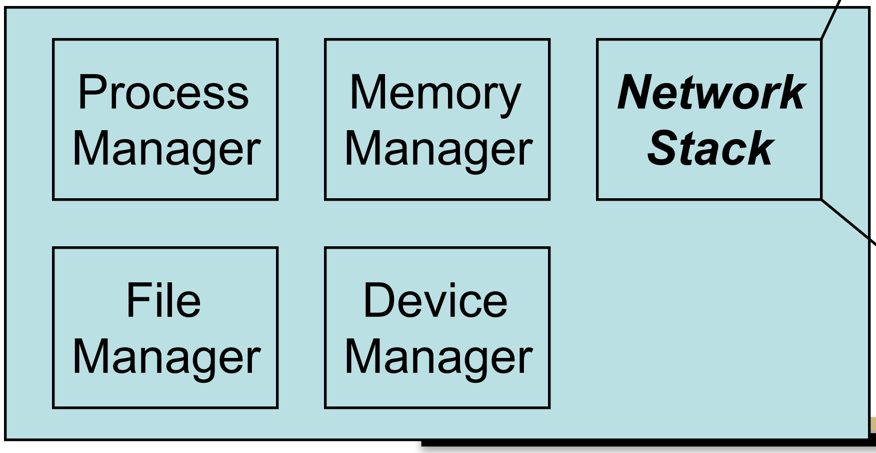
NIC = Network Interface Card

University of Colorado **Boulder**

Networking

- The network stack's architecture is organized into multiple layers of protocols
  - Each protocol performs a specific set of duties

Operating System



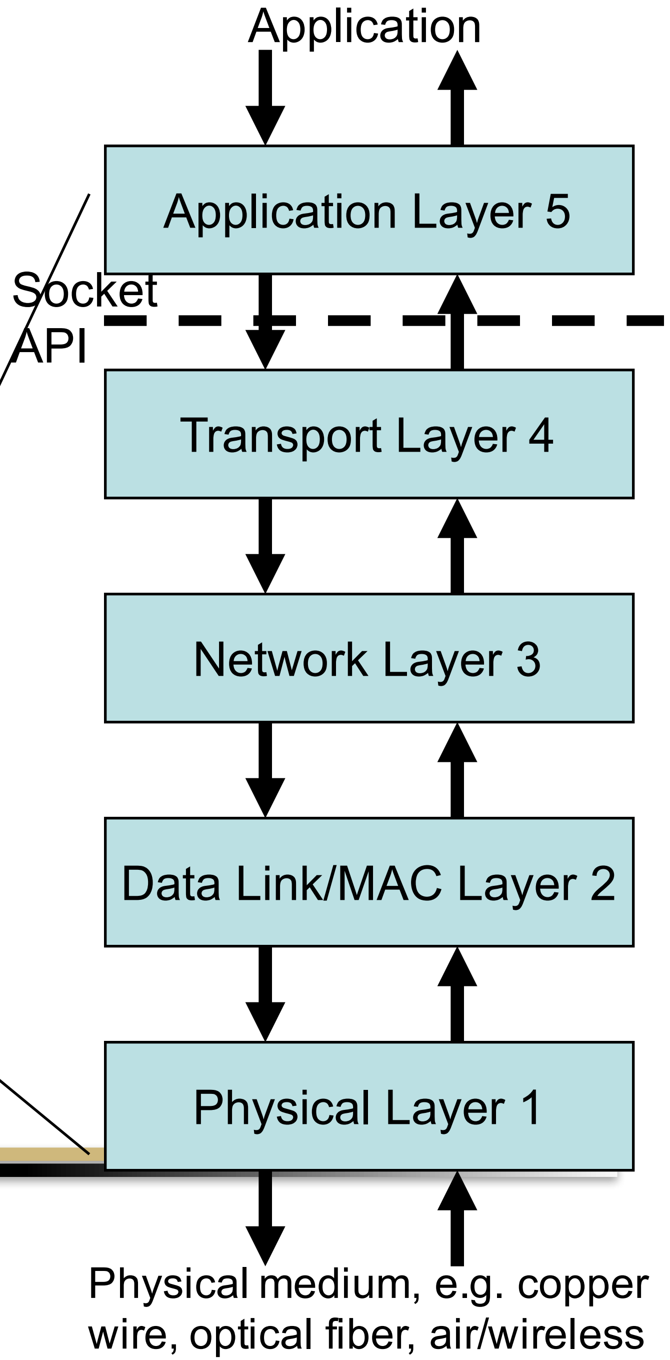
Process Manager

Memory Manager

Network Stack

File Manager

Device Manager



Application

Application Layer 5

Socket API

Transport Layer 4

Network Layer 3

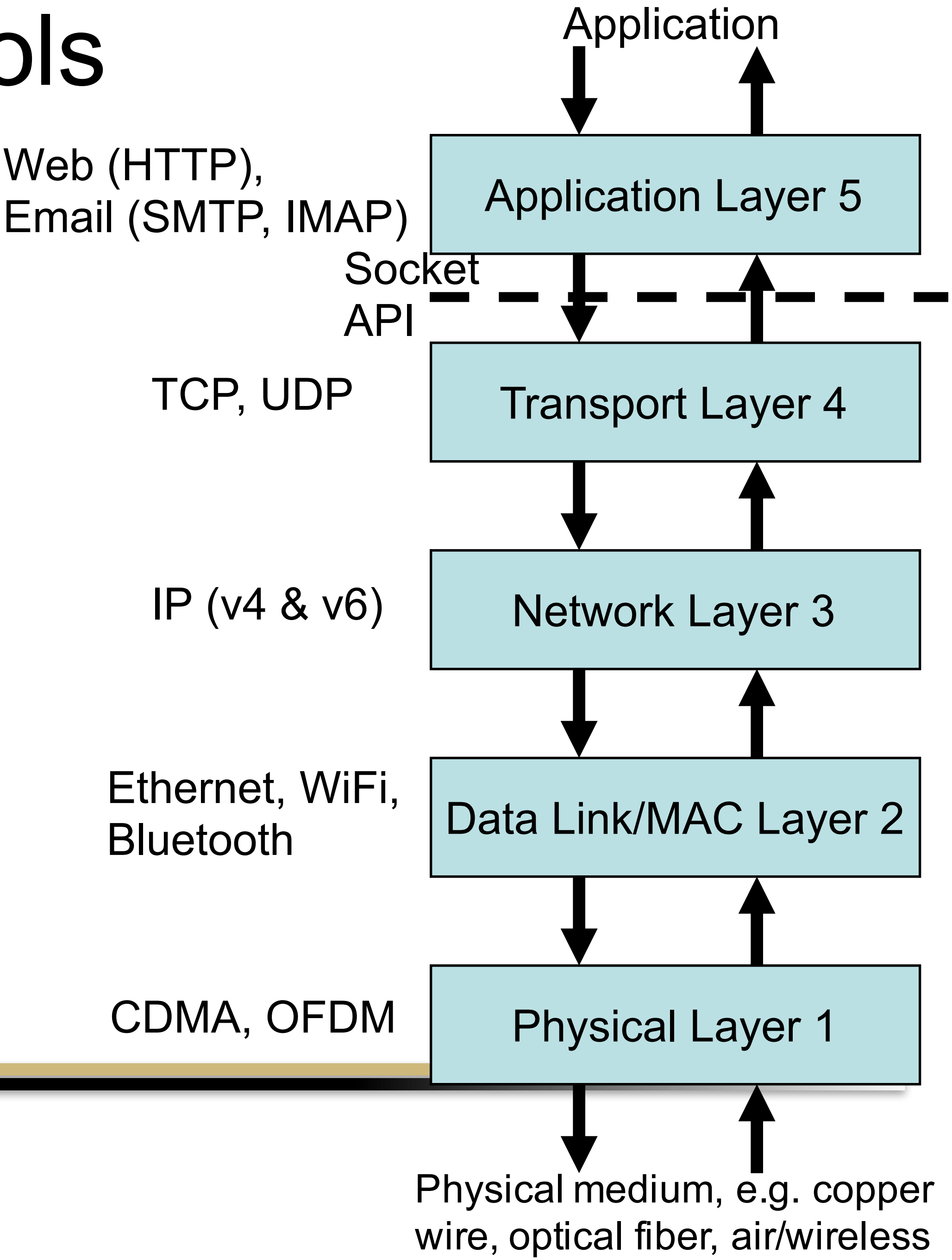
Data Link/MAC Layer 2

Physical Layer 1

Physical medium, e.g. copper wire, optical fiber, air/wireless

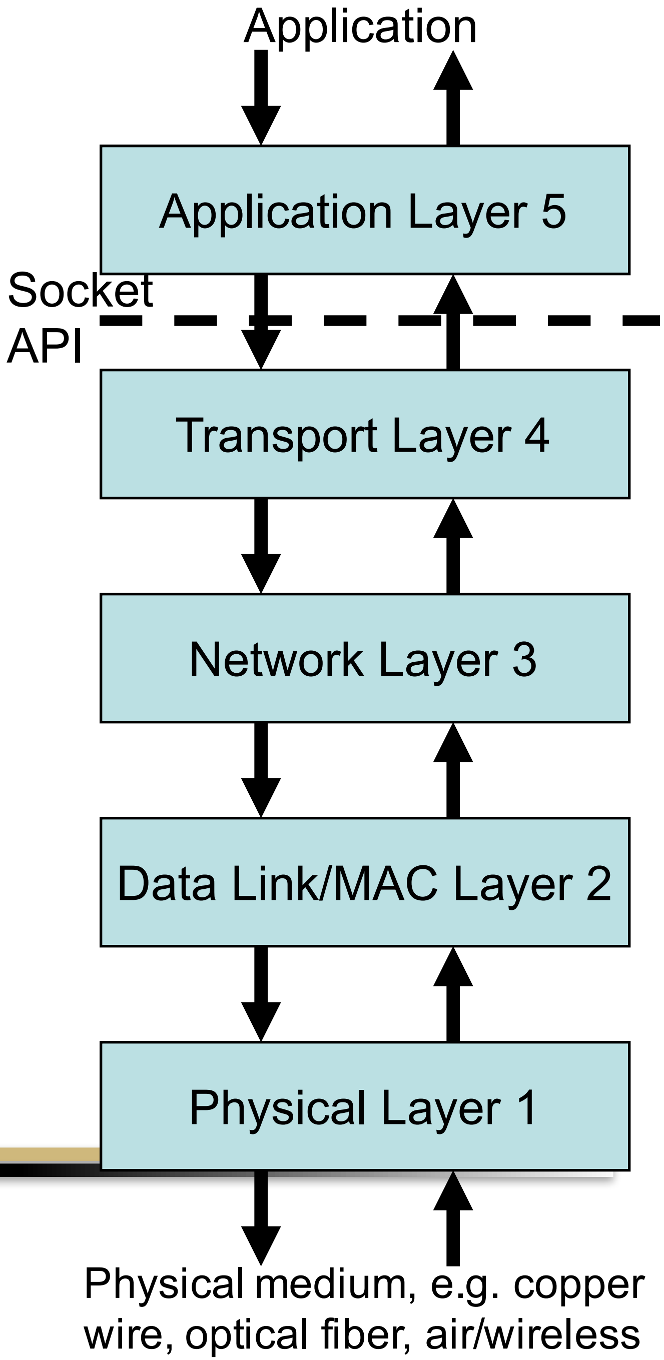
Network Protocols

- Examples of standard network protocols and where they reside in the network stack:



Networking

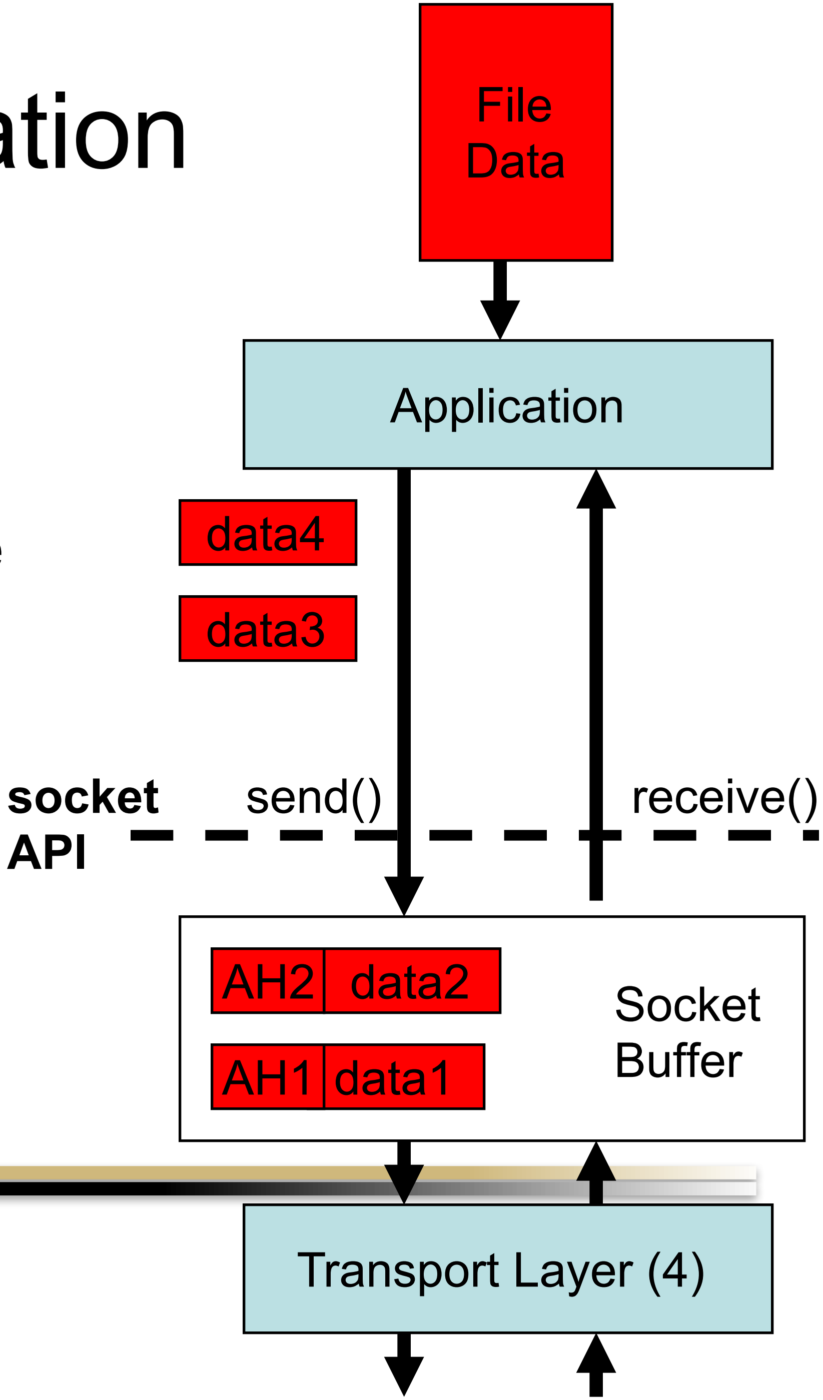
- to send a packet of data to a remote destination,
  - each layer first passes a packet of data down the stack to the next lowest layer
- to receive a packet of data,
  - each layer retrieves a packet of data from the layer below
  - and after processing the packet sends the packet to the layer above



University of Colorado **Boulder**

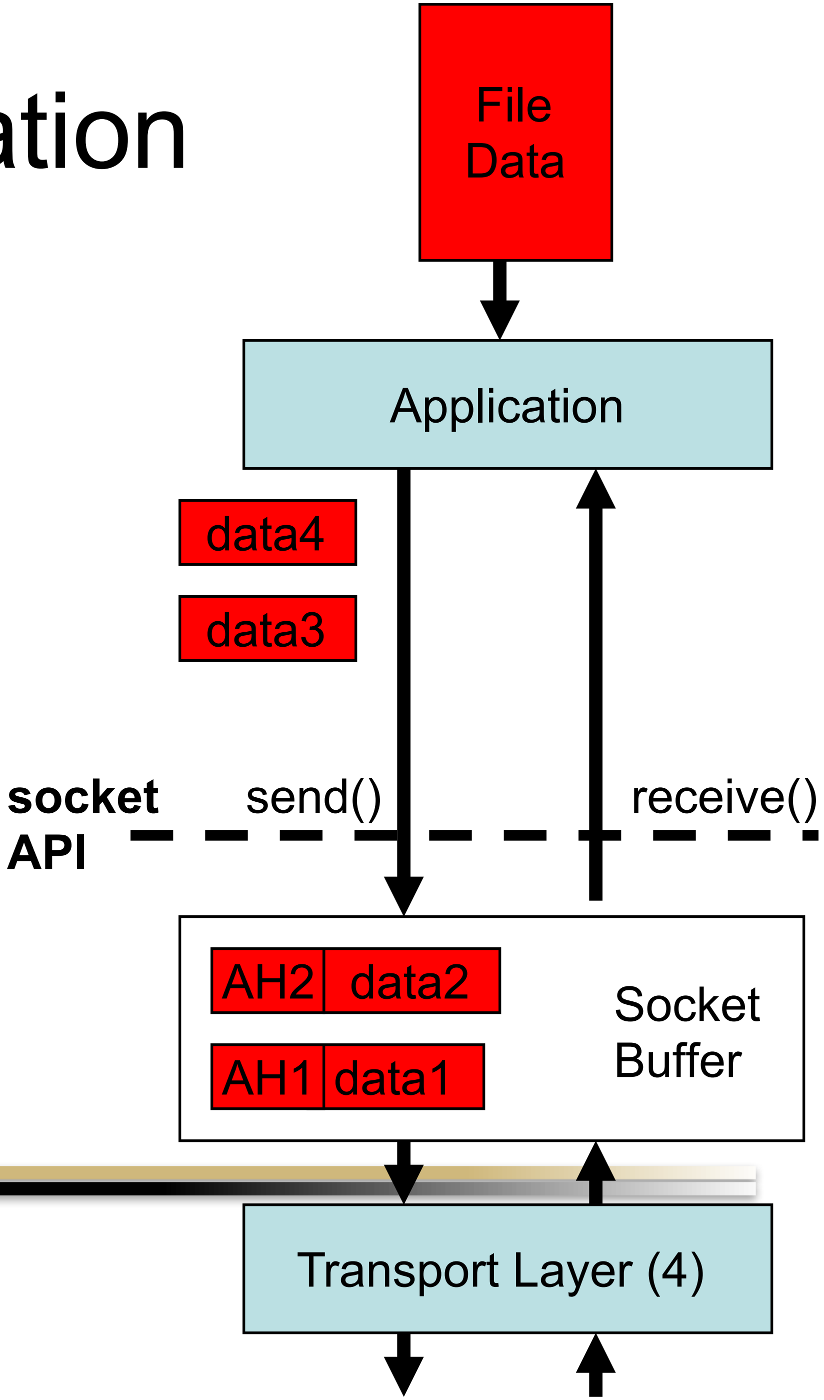
Packetization

- To send, an application calls socket API's send()
  - gives a pointer to the user space buffer containing the data to send
- If the file is large, the application segments the file into smaller packets
  - e.g. a 1 GB file is chopped into 1 KB packets



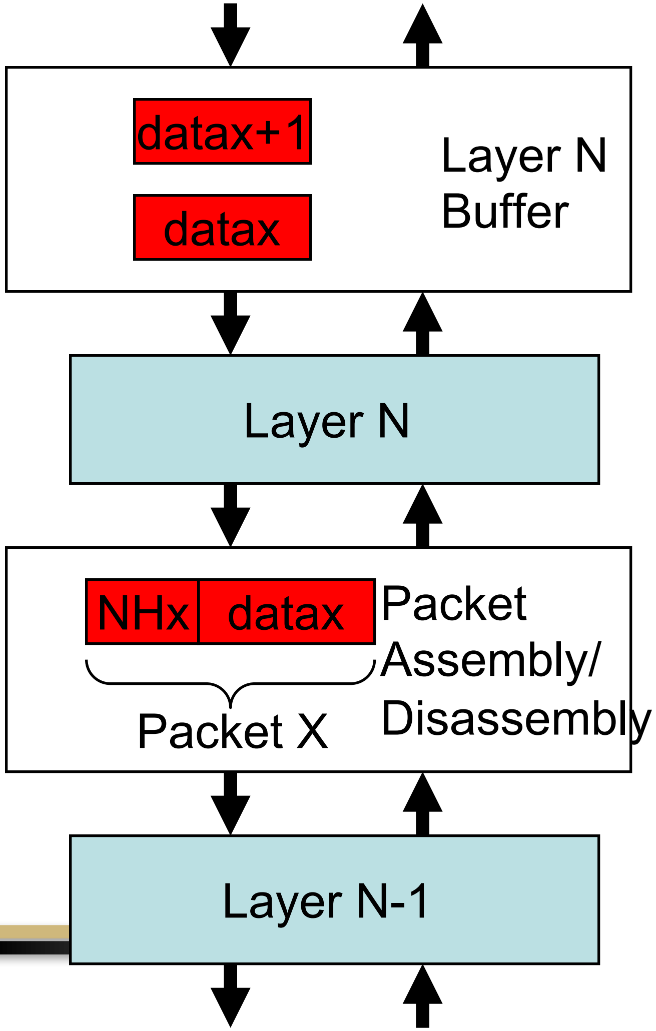
Packetization

- Application layer prepends a layer 5 header to the user data, forming a packet
  - Prepend the header AH1 to data1, forming packet 1
  - Header info is useful at the remote receiver to decode the packet
- Here, packets 1 & 2 are sent down to transport layer 4

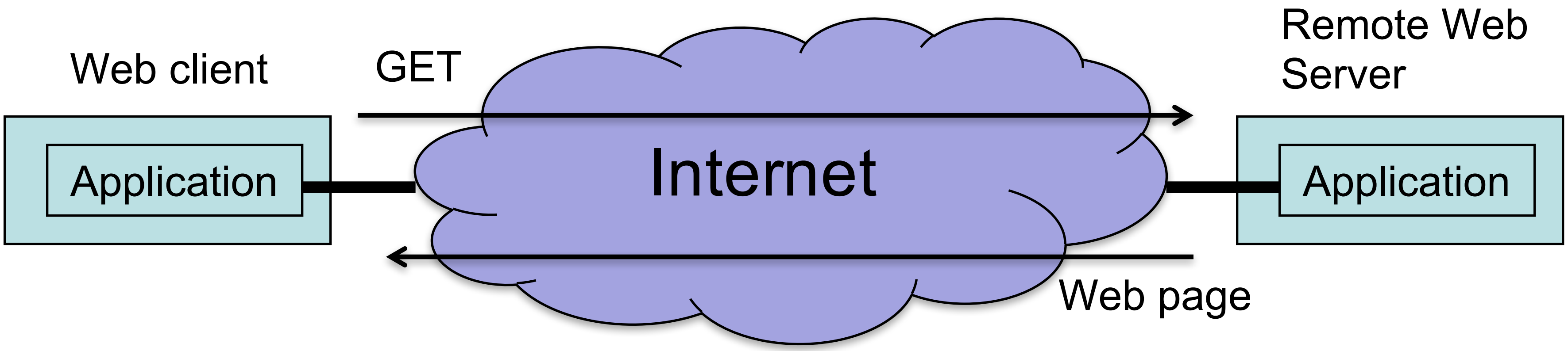


Packetization

- When sending a packet
  - In general, at each layer N, a packet header NH_x is prepended to data x and then sent to a lower layer N-1
  - Packet grows is at descends the network layered stack
- When receiving a packet
  - At each layer N, strip off the layer N header
  - Packet shrinks as it moves up the stack

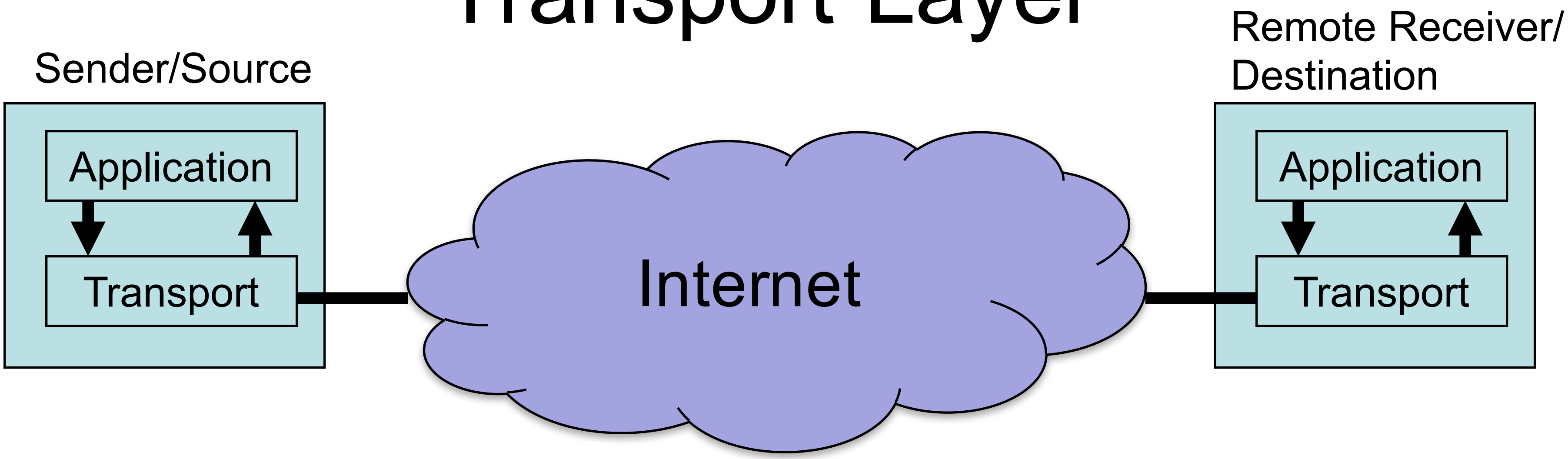


Application Layer



- Let us ignore the lower layers temporarily & focus only on layer 5
- Application layer 5 sender communicates *application-specific* information with its peer layer 5 receiver
  - e.g. Web (HTTP) client sends a GET request (at layer 5) to fetch a Web page from the remote Web server

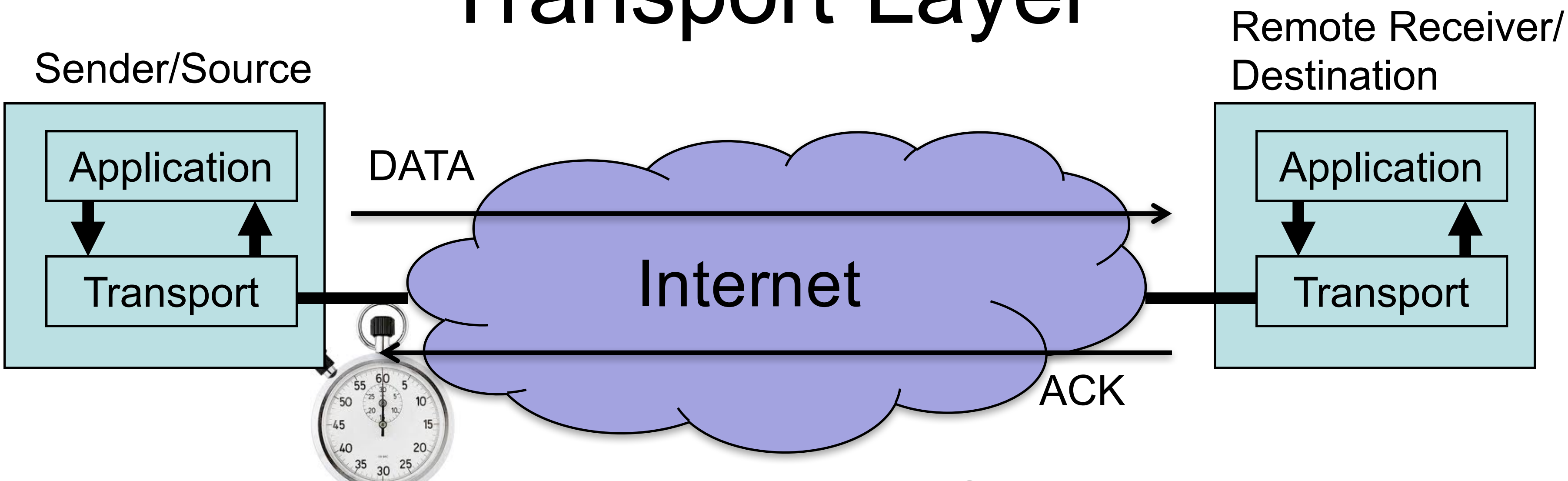
Transport Layer



- The Internet can lose the application’s message!
- The transport layer’s job is end-to-end error recovery, if desired.
- How to recover from a lost packet?
  - *Retransmit* ~~lost packets! This is TCP, the~~ Transmission Control Protocol

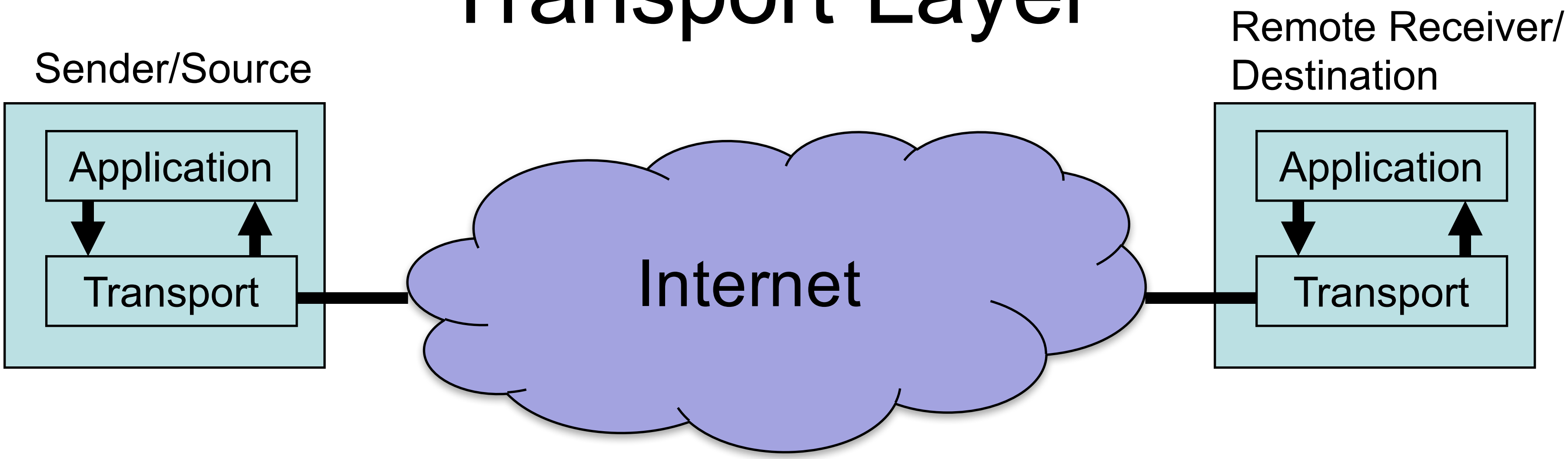
University of Colorado **Boulder**

Transport Layer



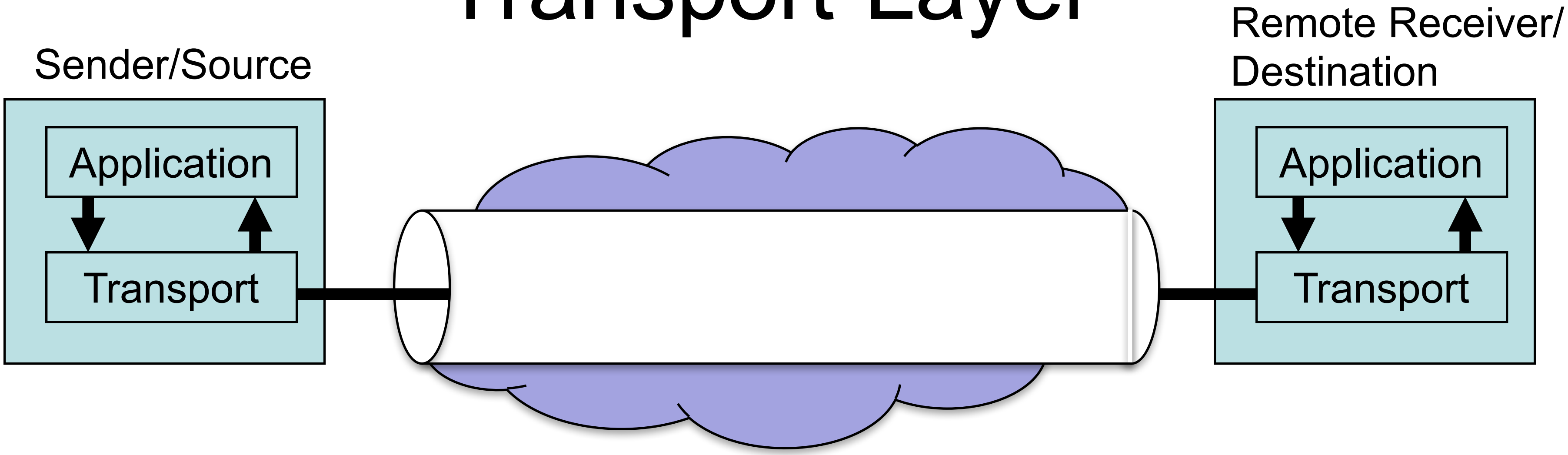
- How does the sender know if a packet was received correctly?
  - Receiver sends an *Acknowledgment* (ACK) packet back to sender
- When does sender know when to retransmit?
  - Sets a *timer*. If it *times out* before ACK received, then retransmit

Transport Layer



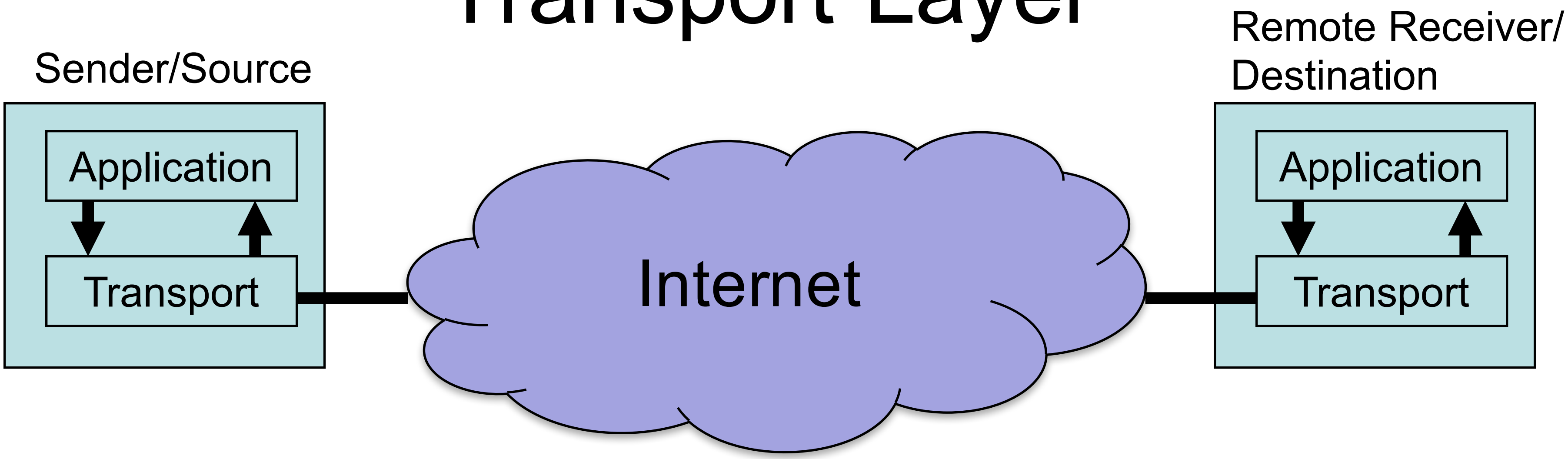
- TCP also ensures in-order delivery
- Many apps require TCP's reliable & in-order packet delivery service
  - Web, email, etc. - can't render a Web page or read email if there are holes in the Web page or email
  - Changing order of Web/email text also makes it unreadable

Transport Layer



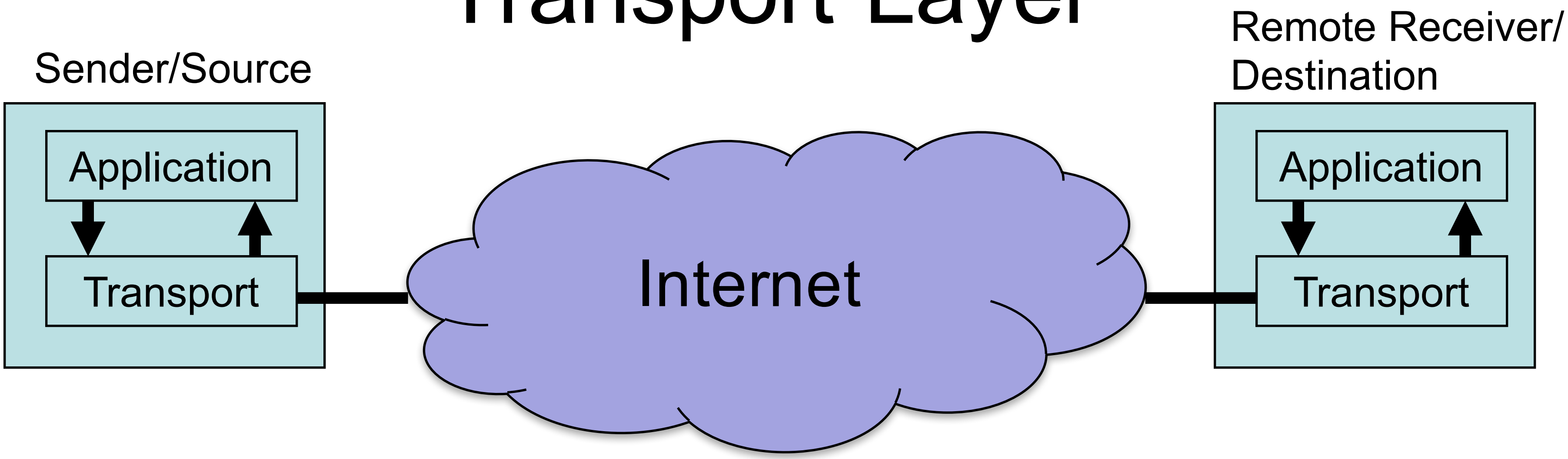
- Apps that use TCP can view the network connection as a pipe abstraction
  - Any data sent into the pipe appears at the other end, hence it is reliable, i.e. pipes don't lose data
  - A pipe preserves the order of the data sent into it at the output of the pipe – no reordering is possible

Transport Layer



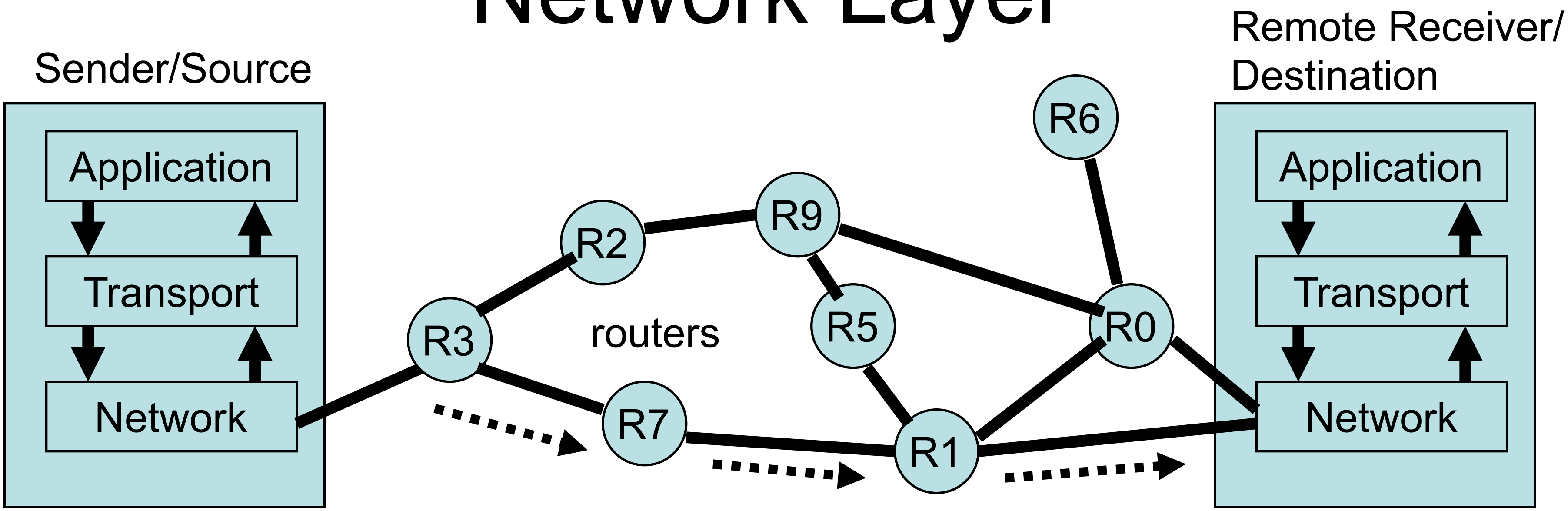
- Reliability comes at the cost of delay due to retransmissions
- Not all apps need/want TCP's reliability
  - Interactive real-time apps like Skype audio/video conferencing can't wait for TCP's retransmissions
  - Must get packet delivered in real time, e.g. within 30 ms

Transport Layer



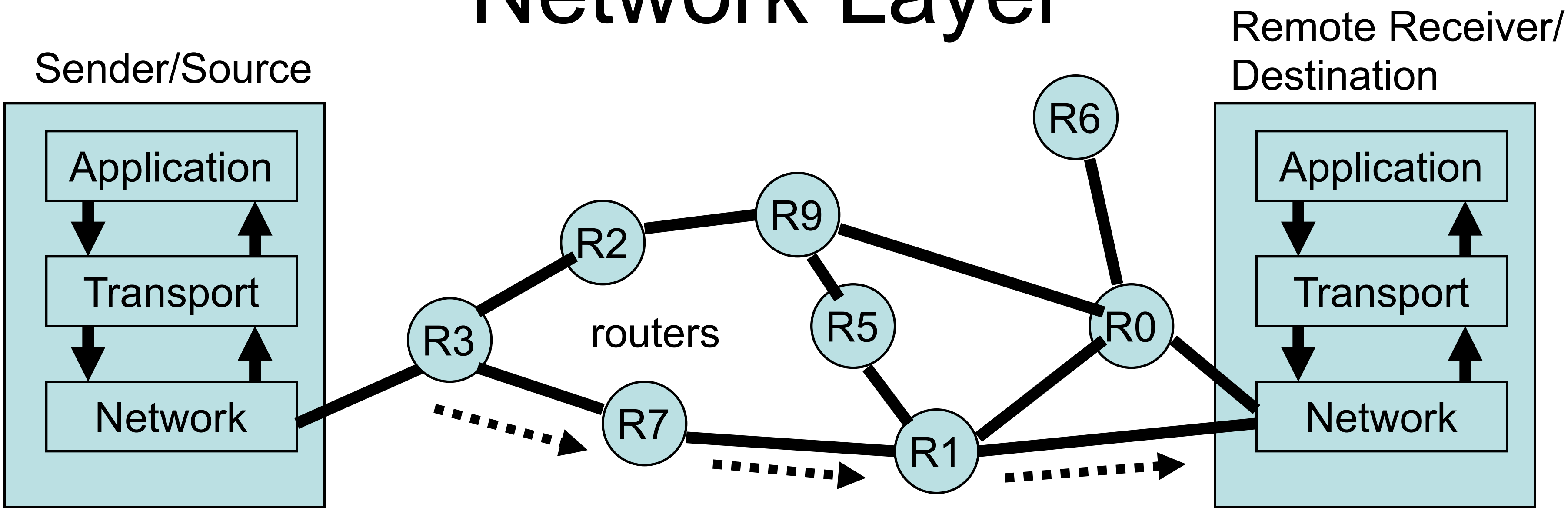
- Real-time Voice-over-IP (VOIP) apps like Skype & FaceTime can tolerate packet loss
  - may lose audio temporarily, but it's OK
- – Such apps are built on top of unreliable UDP (User Datagram Protocol) at layer 4, not TCP

Network Layer



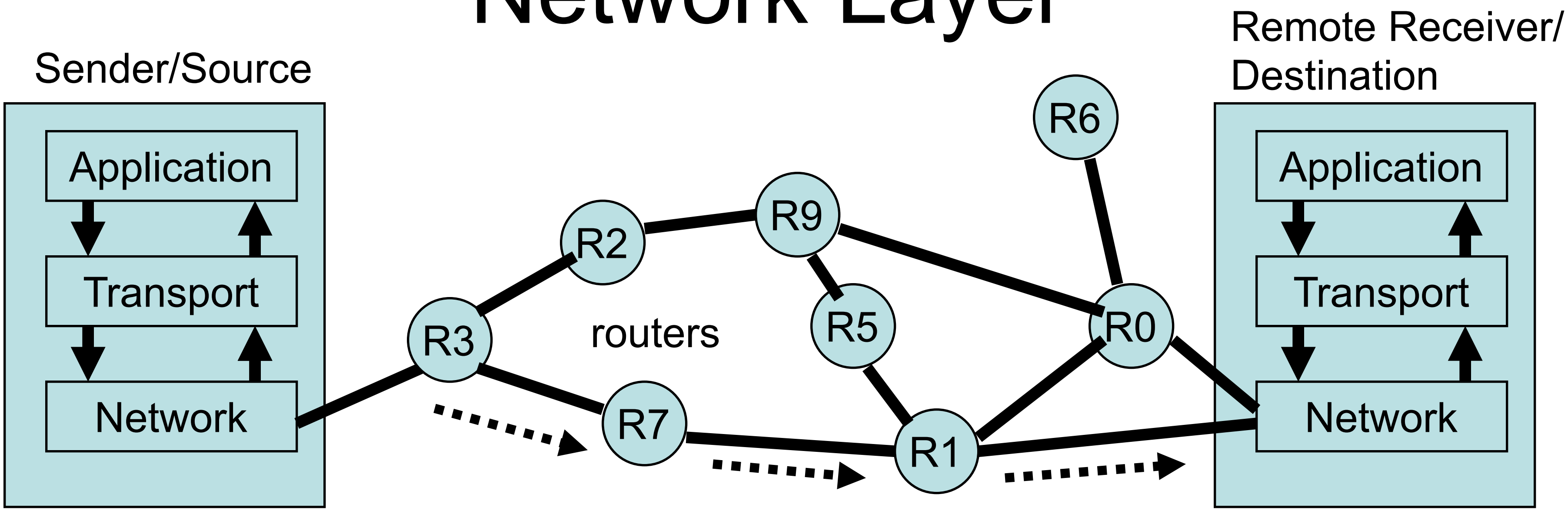
- The Internet consists of many routers that connect together to form a network graph
- The Internet Protocol (IP) network layer must route the IP packet to the correct destination
  - But there are many routes! Which one is best?

Network Layer



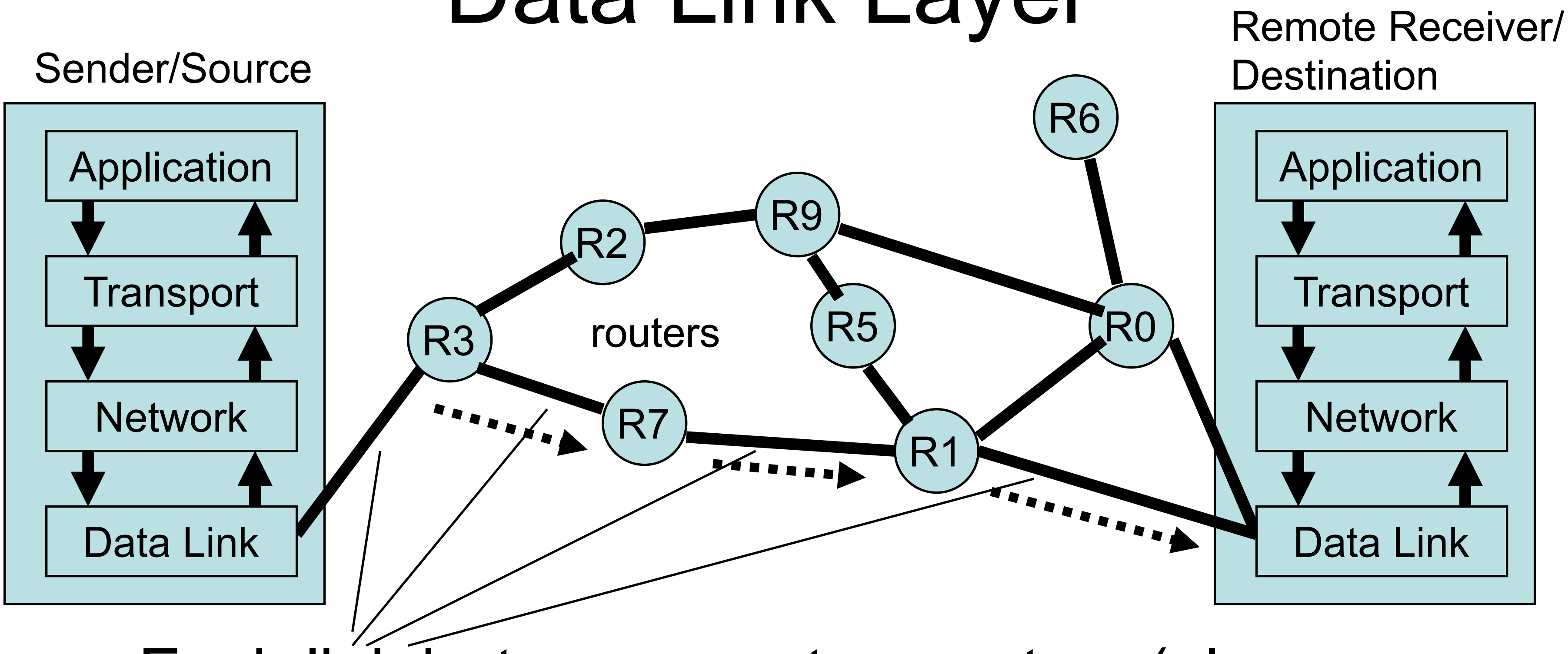
- The network layer tries to find the "shortest path" route, e.g. using Dijkstra's algorithm
  - The metric for shortest path may be minimum # of hops, shortest physical distance, lowest delay, minimum cost, etc.

Network Layer



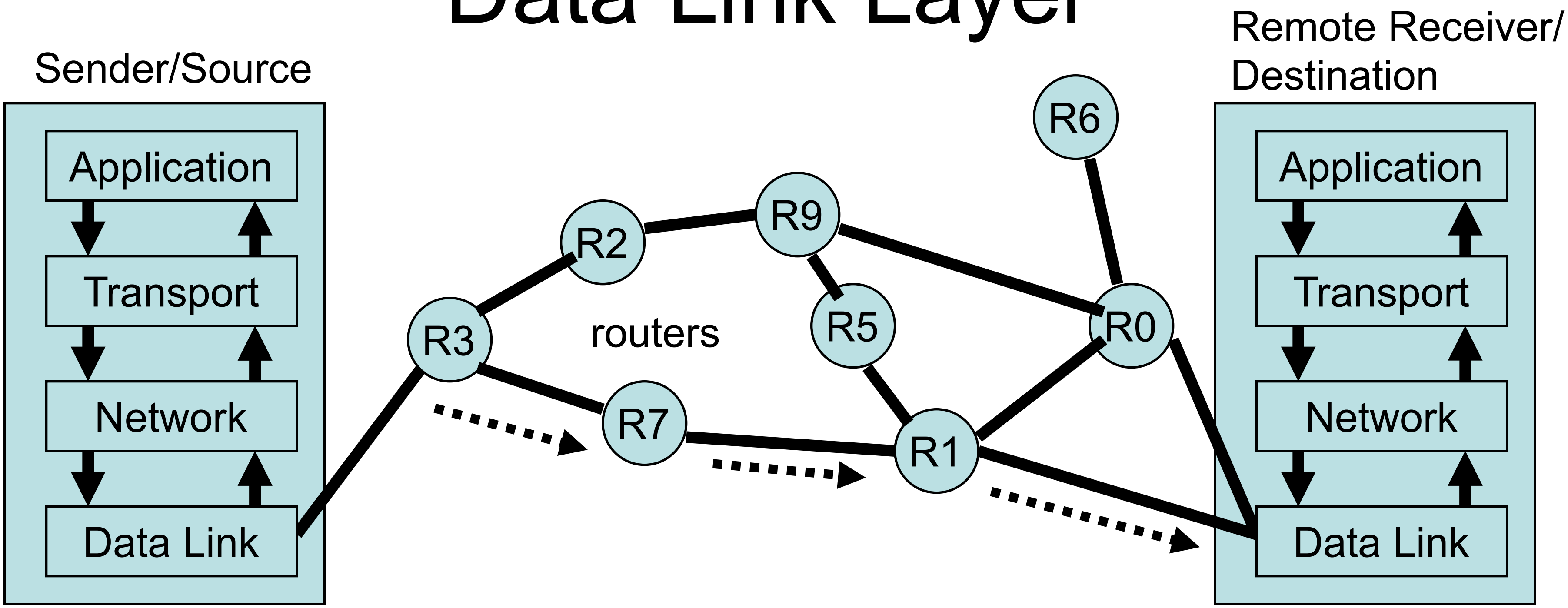
- Each router implements the network layer
- IP routing may lose packets!
  - Any router or link may fail at any time. Also congested router buffers may overflow.
  - That's OK, as long as TCP can retransmit them!

Data Link Layer



- Each link between any two routers (also endpoints) must be able to transmit packets
  - Data link layer is responsible for transmitting packets between any 2 neighboring nodes in the network

Data Link Layer



- This layer must define the beginning and end of packets, i.e. packet framing
- Packets may be lost, so this layer may also retransmit locally
- Examples: Ethernet, WiFi, Bluetooth, …

University of Colorado **Boulder**

Medium Access Control (MAC) Sublayer

- MAC protocols can be considered to be part of data link layer 2
- The previous network graph assumed that there was only 1 sender and 1 receiver on any link, i.e. that it was point-to-point
- In reality, there may be many computers sending and receiving on the same shared link
  - e.g. in WiFi, all nearby laptops share the same wireless link for sending and receiving
  - There may be collisions when 2 transmit at the same time!

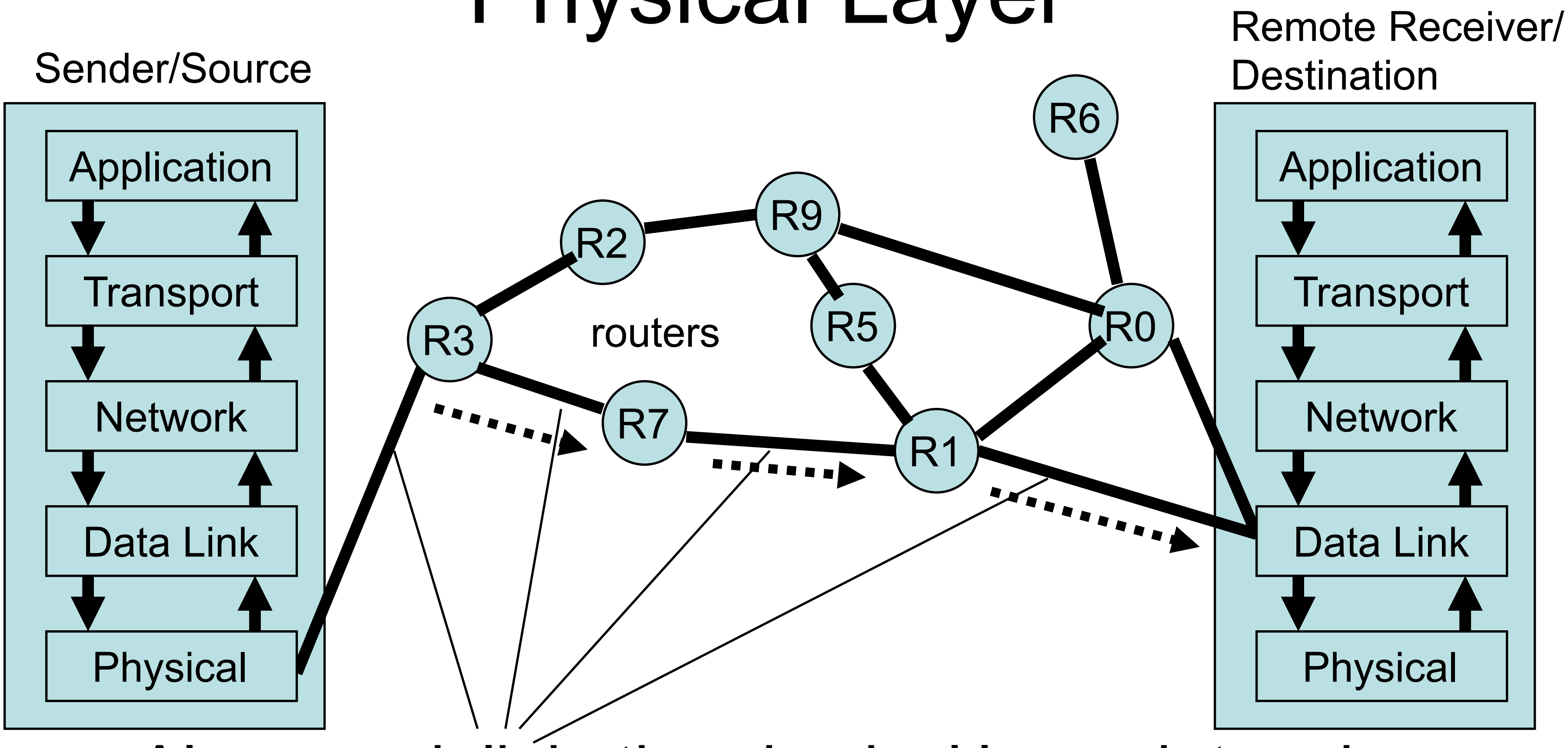
Medium Access Control (MAC) Sublayer

- For shared media, we need a protocol that decides which of the N computers sharing the media gets to send next, to avoid collisions
  - The MAC protocol arbitrates who next gets access to transmit on the shared medium
- Standard MAC protocols include:
  - TDMA: time division multiple access, i.e. each user is assigned a time slot within which to transmit, so there are no collisions
  - FDMA: frequency division multiple access, i.e. each user is assigned a separate frequency, to avoid collisions

Medium Access Control (MAC) Sublayer

- Standard MAC protocols include (continued):
  - CSMA: carrier sense multiple access, i.e. each user senses the medium before transmitting and if the medium is free goes ahead and transmits
    - This approach minimizes collisions in a way that doesn’t require tight synchronization as for TDMA
    - both WiFi/802.11 as well as Ethernet use a form of CSMA

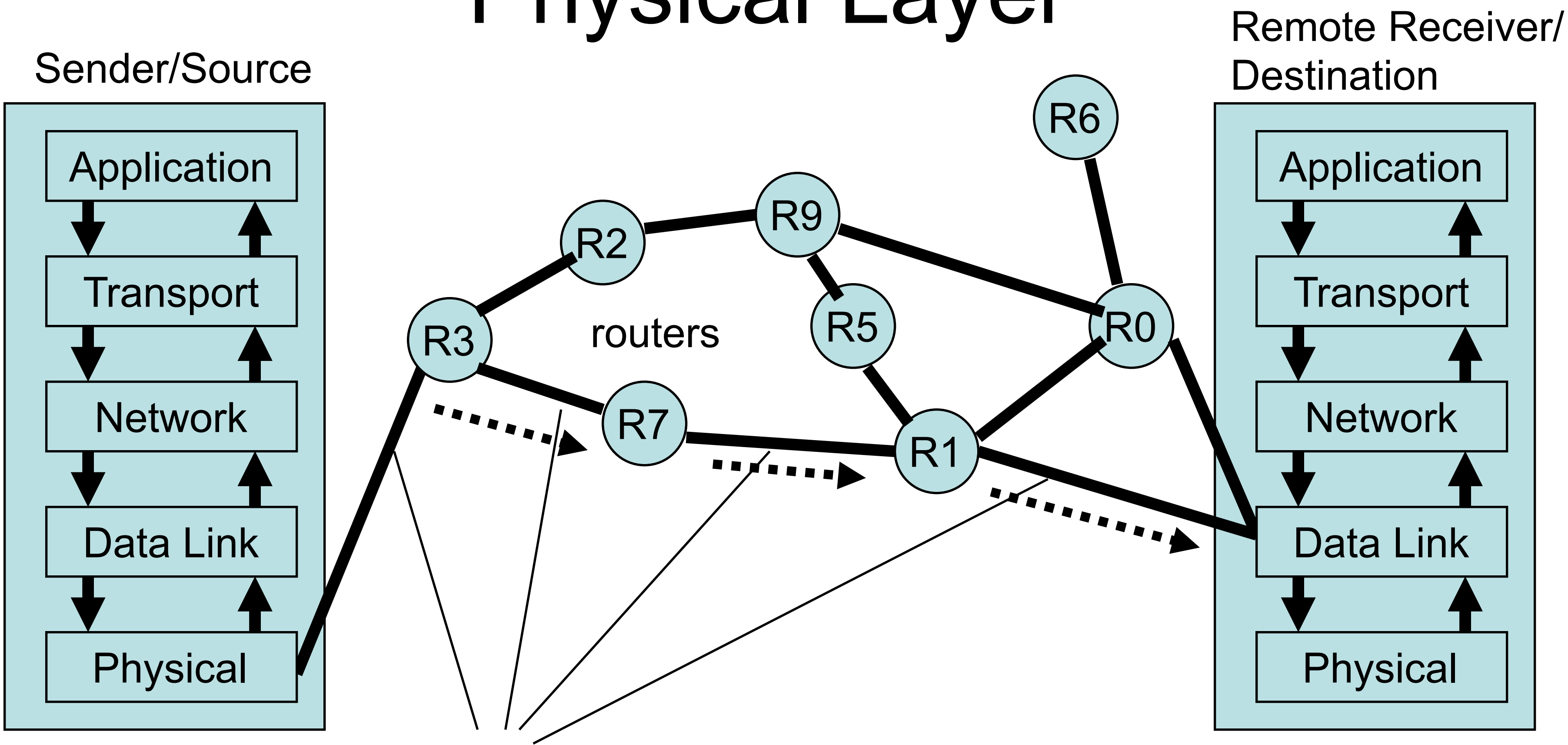
Physical Layer



- Along each link, the physical layer determines how 1's and 0's, i.e. digital bits, are transmitted

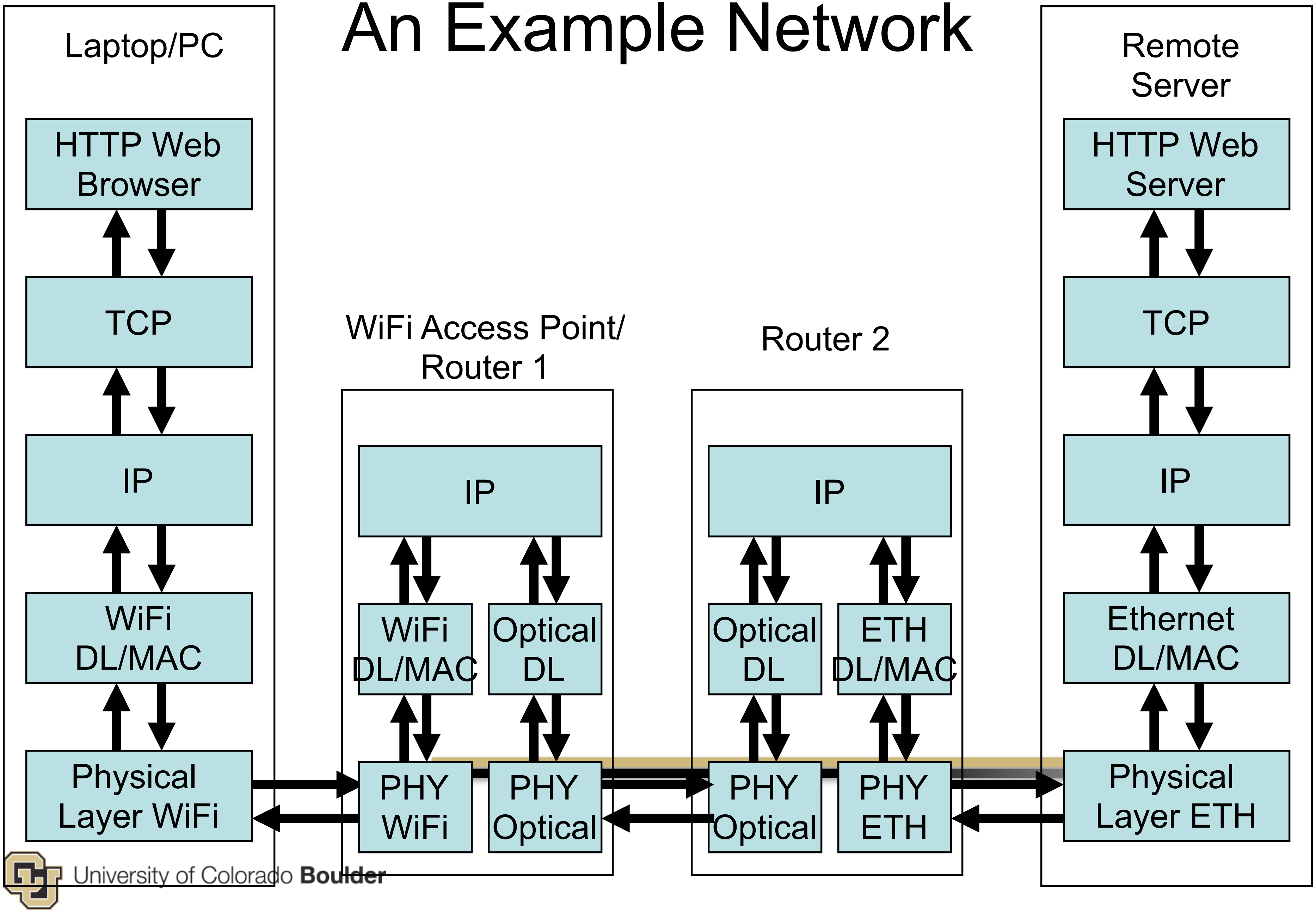
University of Colorado **Boulder**

Physical Layer



- Example: a ‘1’ may be +5 volts, and a ‘0’ may be 0 volts. Or 1s & 0s may correspond to different frequencies.

An Example Network



Laptop/PC

HTTP Web Browser

TCP

IP

WiFi DL/MAC

Physical Layer WiFi

WiFi Access Point/ Router 1

IP

WiFi DL/MAC

Optical DL

PHY WiFi

PHY Optical

Router 2

IP

Optical DL

ETH DL/MAC

PHY Optical

PHY ETH

Remote Server

HTTP Web Server

TCP

IP

Ethernet DL/MAC

Physical Layer ETH