Introduction to Networking

CS 241

April 16, 2014

University of Illinois

Announcements

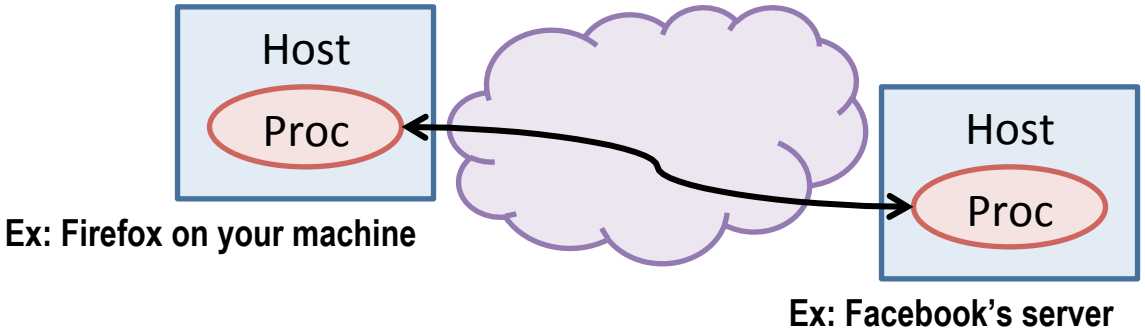
Last call for Pebble pickup

- Right after class today

Networking

What do we expect out of networking?

- An channel between two processes on two remote machines.



Making this happen is complex!

- Hosts
- Routers
- Various Links
- Applications
- Protocols
- Hardware
- Software
- Bit errors
- Packet errors
- Link failures
- Node failures
- Message delays
- Out-of-order delivery
- Eavesdropping

Protocols

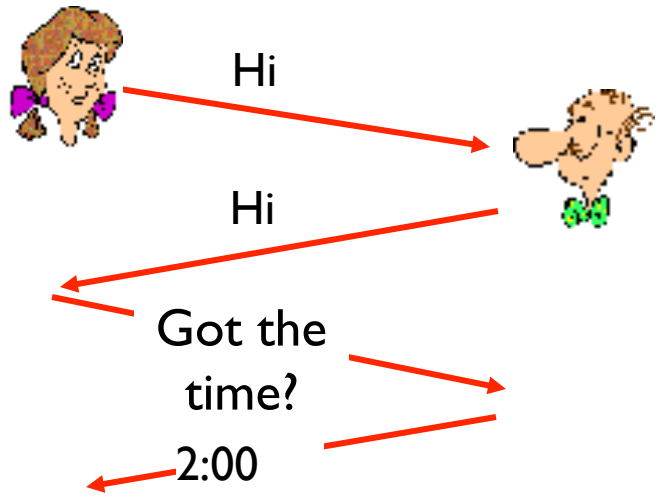
A protocol is a message format and rules for exchanging these messages.

You already use a lot of protocols:

What is a Protocol?

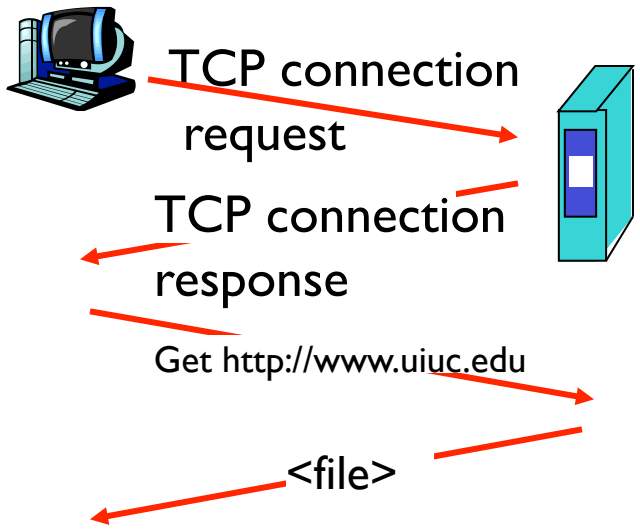
Human protocols

- "what's the time?"
- "I have a question"
- Introductions



Network protocols

- Machines rather than humans
- All Internet communication is governed by protocols



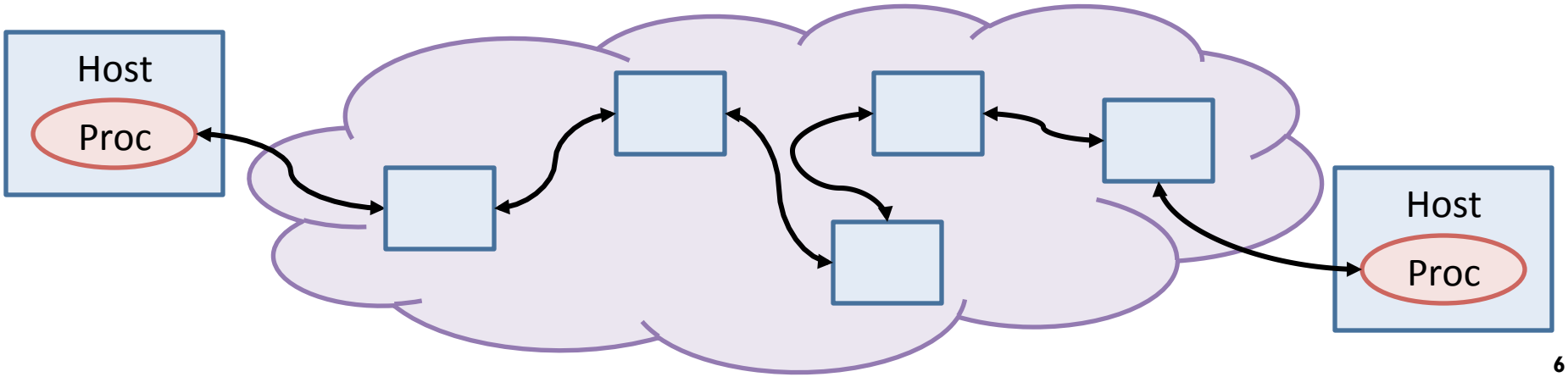
time

Networking Model: Layers of Protocols

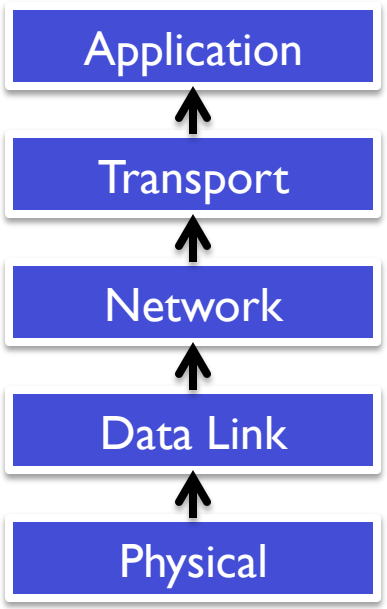
A network channel is effectively only a transmission of 0s and 1s:

```
0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 1 1 1 0 1 0
```

How do we translate these 0s and 1s into HTTP packets?
How do we get those to the right end-user?



The Internet's Protocol Stack

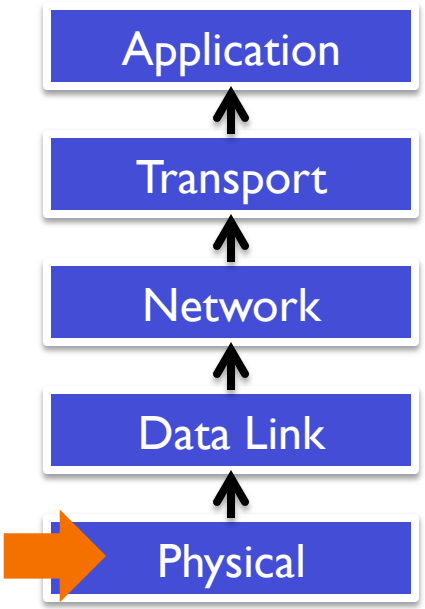


The Internet's core protocols form a "stack".

- Larger seven-layer stack devised by a committee in the 1980s is called the OSI Model
- Here we focus on layers commonly used today

Each layer encapsulates data sent by the layers above it and provides specific features to higher-layer protocols.

Layer 1: Physical Layer



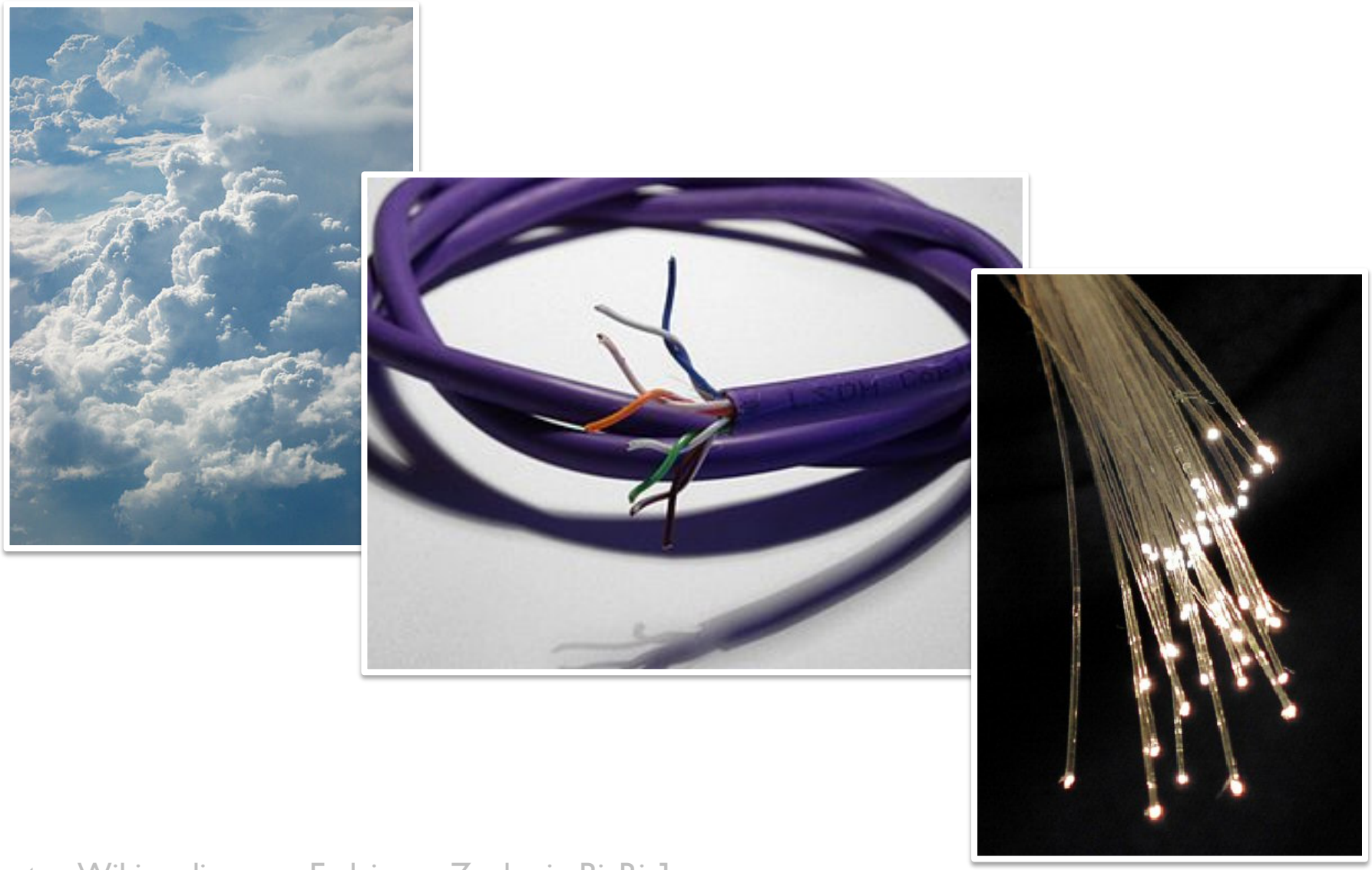
The **Physical Layer** provides hardware-specific details on how to transmit a 0 vs. 1.

- **100BASE-T**: Ethernet
- **GSM "Um Interface"**: Cell phones
- **802.11**: Wi-Fi

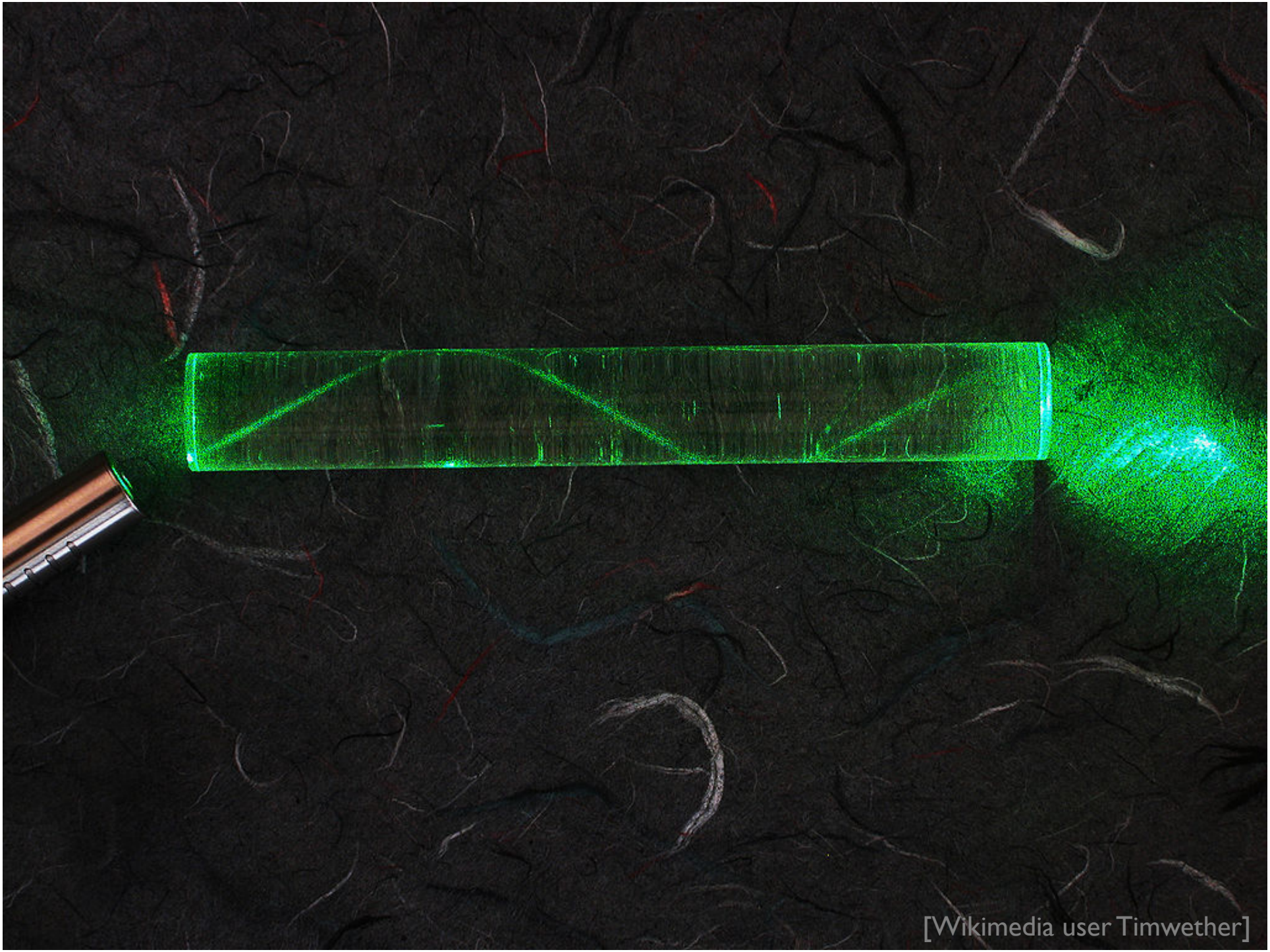
Provides: A digital representation of the underlying signal; *a series of 0s and 1s*.

Key problem: Analog to digital conversion

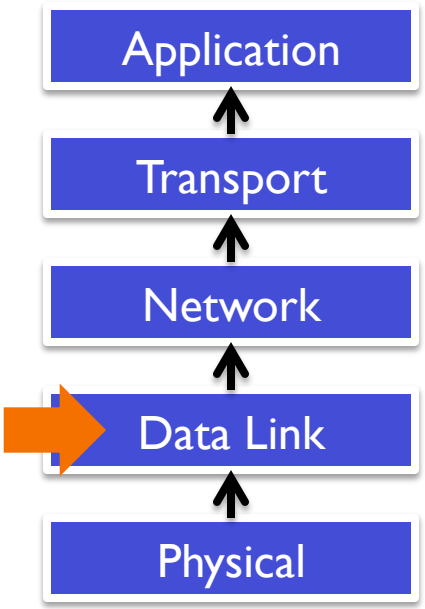
Physical transmission media



[Photos: Wikimedia users Evdaimon, Zephyris, BigRiz]

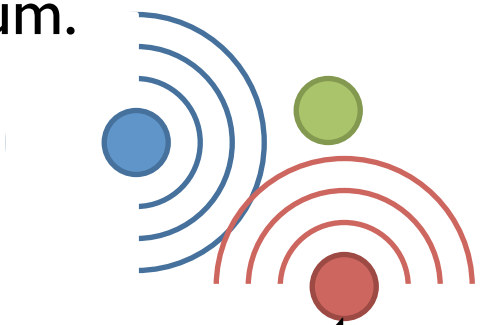

[Wikimedia user Timwether]

Layer 2: Data Link Layer



The **Data Link Layer** sends and receives packets on a transmission medium.

- Consider Wi-Fi:
  - Every computer that is connected to a wi-fi access point uses the same channel: **every computer hears every other computer!**
  - *How do we know the data that is coming in is for us, not for our neighbor?*

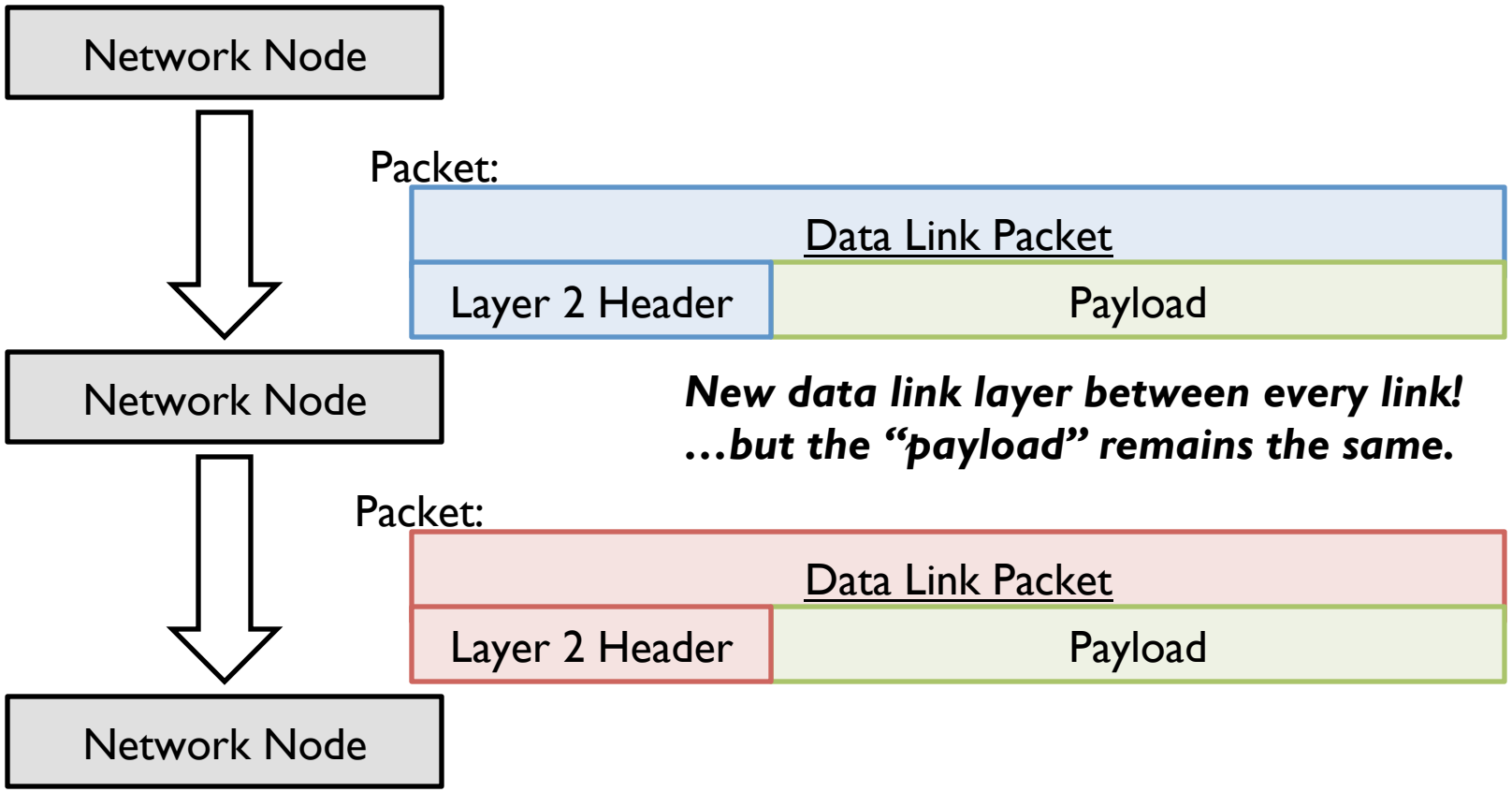


Provides: Send/receive packets on a "wire"

Key problems:

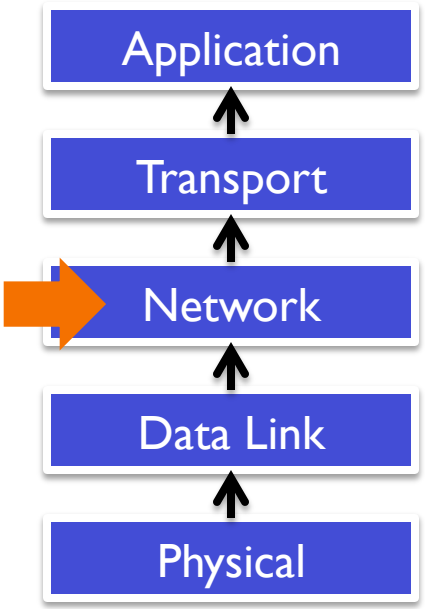
- Framing, errors
- Addressing, resource contention

Layer 2: Data Link Layer



New data link layer between every link!
…but the "payload" remains the same.

Layer 3: Network Layer



The **Network Layer** provides host-to-host communications across many "hops".

- One protocol: IP (IPv4 and IPv6)

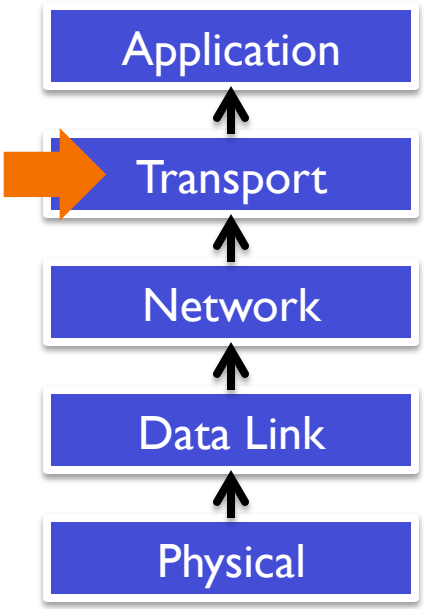
Provides: Information on the source and destination **host**.

- *Where should this packet go?*
- *Who sent this packet in the first place?*

Key problems:

- Addressing
- Heterogeneous transmission media
- Routing

Layer 4: Transport Layer



The **Transport Layer** provides process-to-process communications.

- Two main protocols: TCP and UDP

Provides: Information on the source and destination **process**… *and much more.*

- Uses "network ports", a globally shared resource on a system that associates a **port number** with a process.
- The process making the connection to a remote process needs to know the port number the remote process is listening on.

Key problems:

- Process multiplexing, reliability, congestion...

TCP vs. UDP

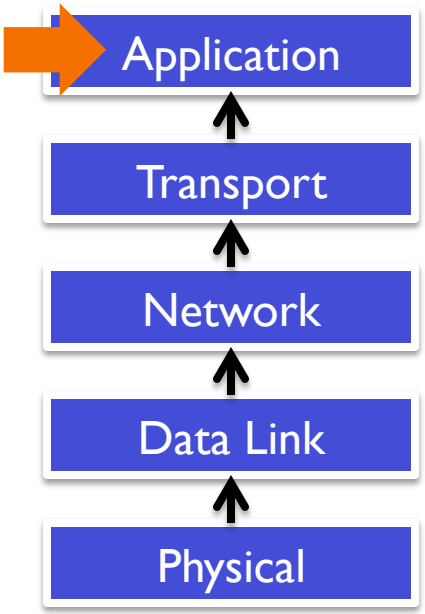
TCP and UDP both provide process-to-process communications via port numbers.

- That is about all UDP does. UDP: fast and cheap!

TCP provides several convenience features:

- Reliable Transmission:
  - Packets will arrive in the order that they were sent
  - All packets will arrive (on an active connection)
  - All packets will be delivered once (no duplicates)
- Flow and Congestion Control:
  - TCP negotiates a rate of transmission between the hosts so the receiver is not overwhelmed with data

Application Layer ("layer 7")

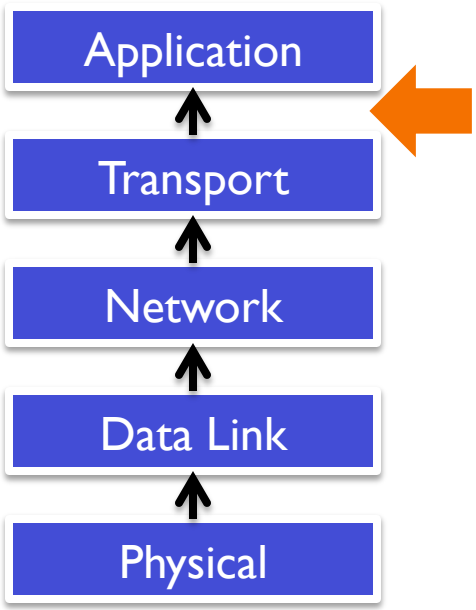


The **Application Layer** refers to any communication software and procedures for particular applications.

- May actually be composed of many layers

Examples?

Where are we as systems programmers?



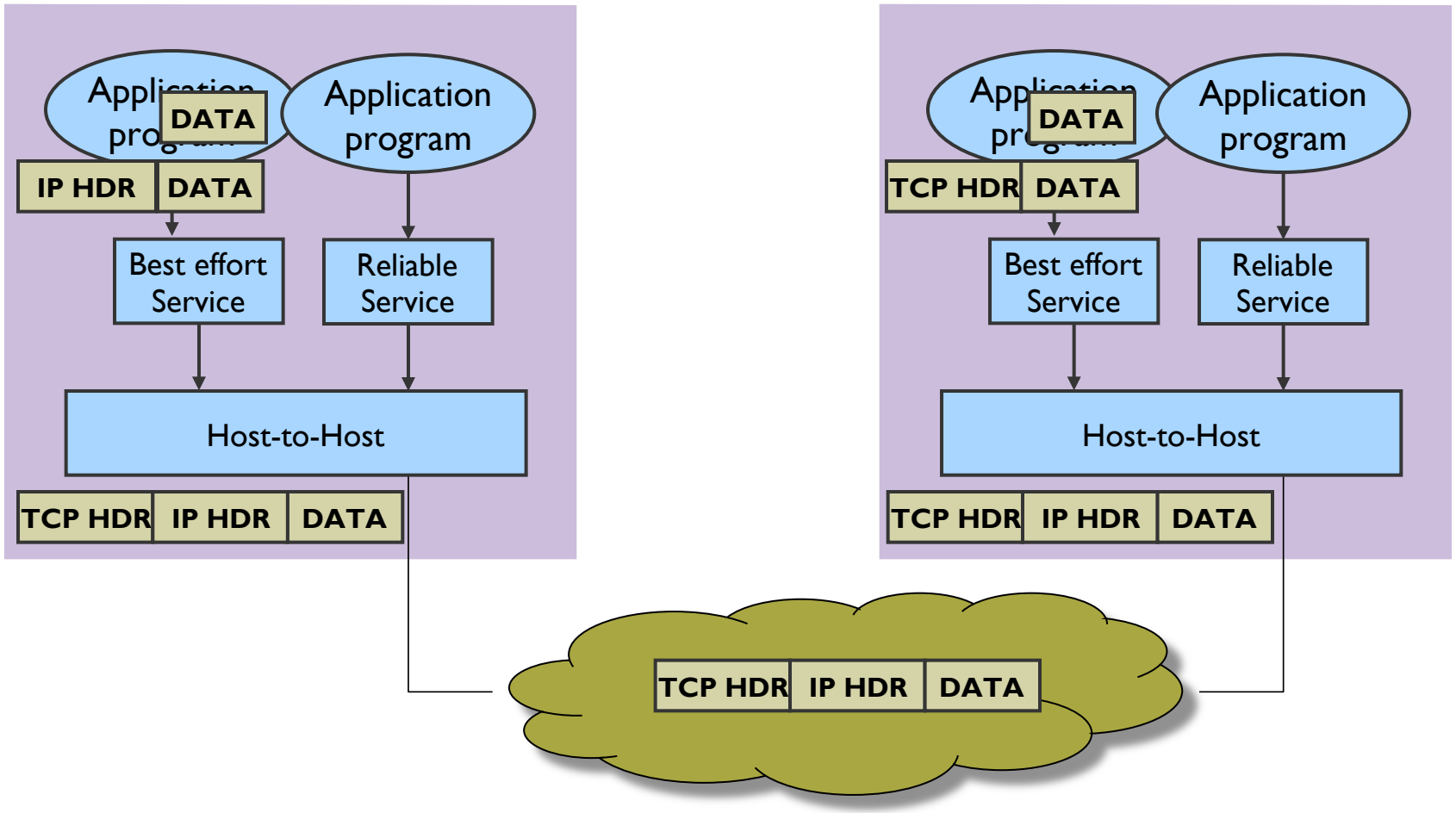
Most commonly, we write applications using an interface to the transport layer

- In this class: the sockets interface
- Others exist

Above sockets interface: User code

Below sockets interface: Kernel

Encapsulation



Why layering?

It's all about modularity

- Eases maintenance, updating of system
- Change of implementation of layer's service transparent to rest of system
- e.g., change in transmission medium (Layer 0) has no effect on network protocol or applications

What other examples of layering have we seen?

Internet Architecture: The "Hourglass" Design