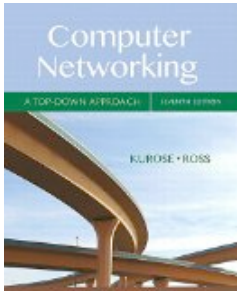
COMP 375: Lecture 02



- **News & Notes:**
  - Project #1: Reverse engineer a network app
    - Available: Before end of Friday
    - Due: Mon, Feb. 12
- **Reading (Due: Fri, Feb. 2)**
  - Section 2.1
  - Complete Reading Quiz #2 by 10am

One Minute Discussion

- What's an example of a protocol you adhere to in your life?

Clicker Test Question

Which of these is **not** an interest of Dr. Sat?

Set Frequency to "BC"

A.	**Gardening**
B.	**Taylor Swift**
C.	**Board Games**
D.	**Broken Bones**

Section 1.5

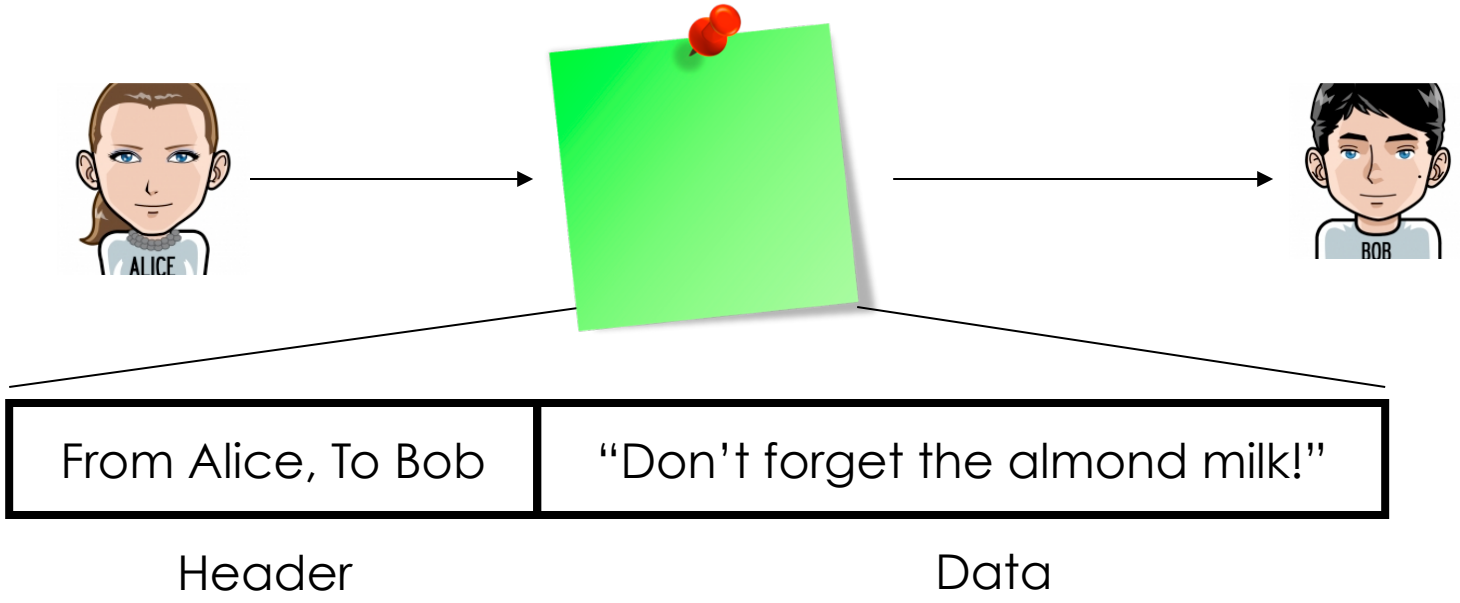
LAYERS AND PROTOCOLS

Protocols help us get a message from a sender to a receiver.

In addition to data, messages need a header to handle protocol info.

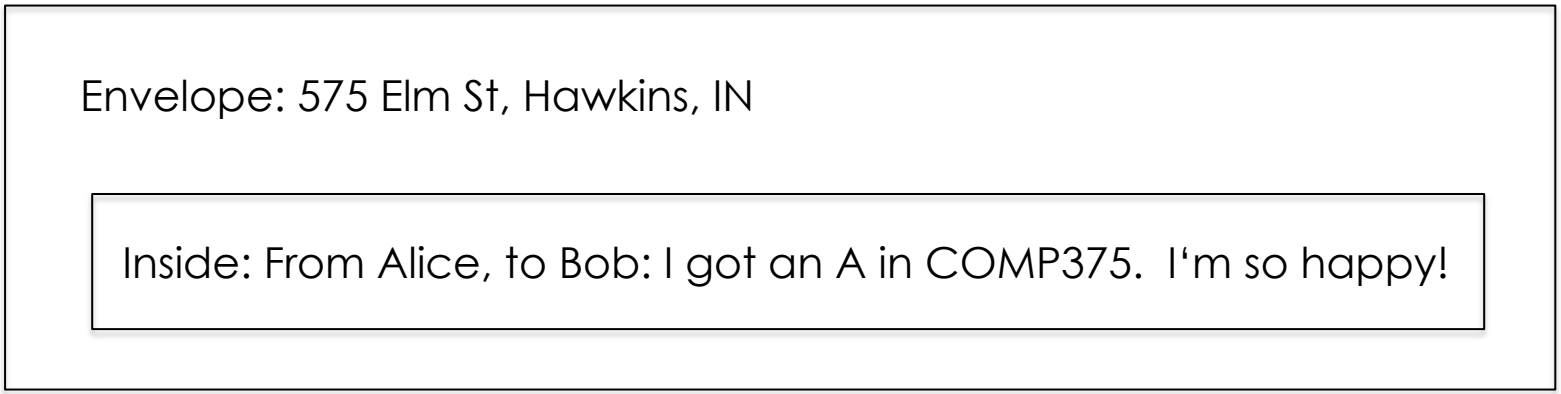
Message:

Alice and Bob, roommates, have agreed to the "Postit Note Protocol" (PNP)



- **Transfer procedure**: Post on refrigerator

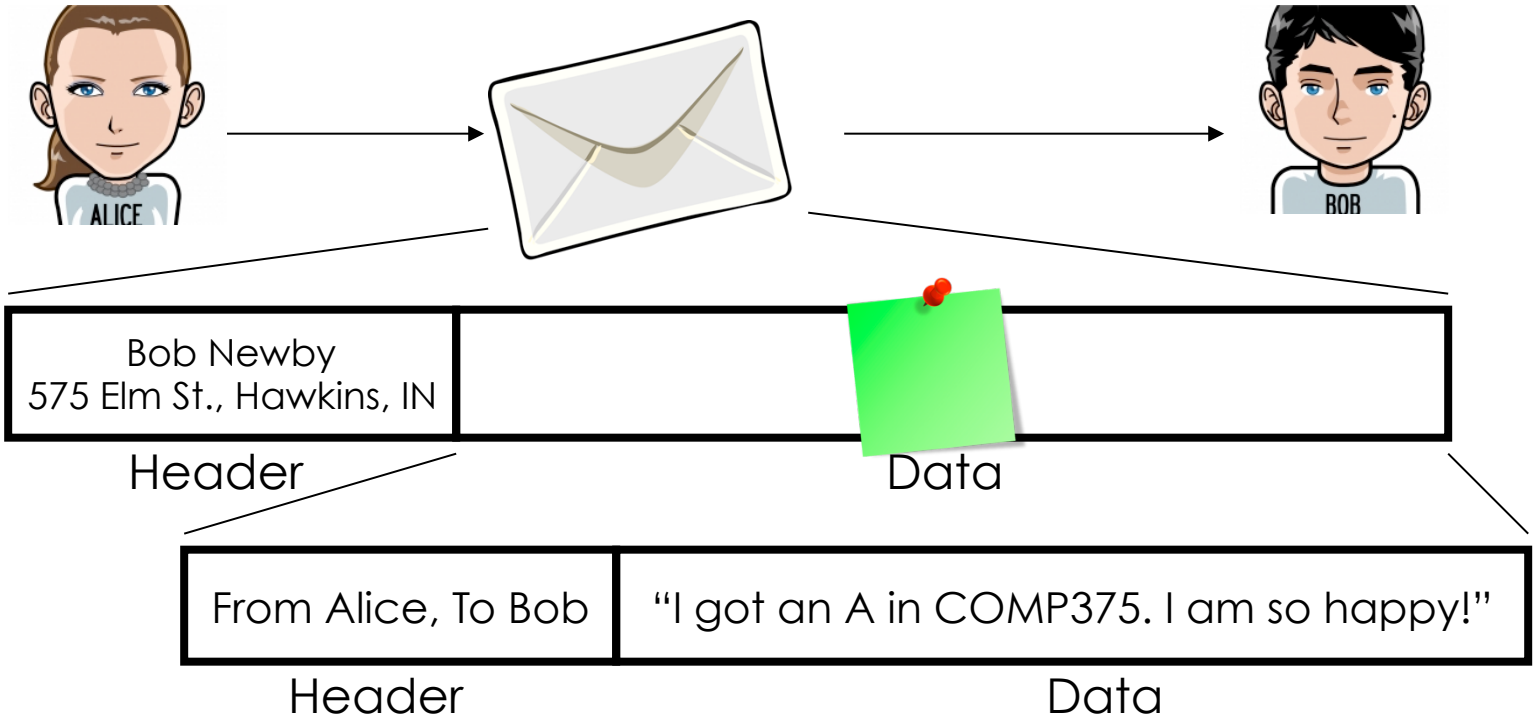
Suppose Alice is mailing a note to Bob via snail mail…



Where is the **header** now?

A.	The address on the envelope.
B.	The "From Alice to Bob".
C.	Somewhere else.

The original message (post-it) gets *encapsulated* in another message.



Layering separates functions, allowing specialization and therefore efficiency.

Letter: written/sent by Alice, received/read by Bob
Postal System: Delivery of letter in envelope

- Alice and Bob
  - Don't have to know about delivery
  - However, aid postal system by providing addresses
- Postal System
  - Only has to know addresses and how to deliver
  - Doesn't care about "data": Alice, Bob, letter

Generals don't want to deal with delivering messages themselves.

General (Application)

General (Application)

Without electronic communication, the military used a layered approach.


General (Application)

Colonel


Captain


Lieutenant


Private

General (Application)

Colonel

Captain

Lieutenant

Private

What's the correct ordering of layers for our Internet stack?

Application

Link

Network

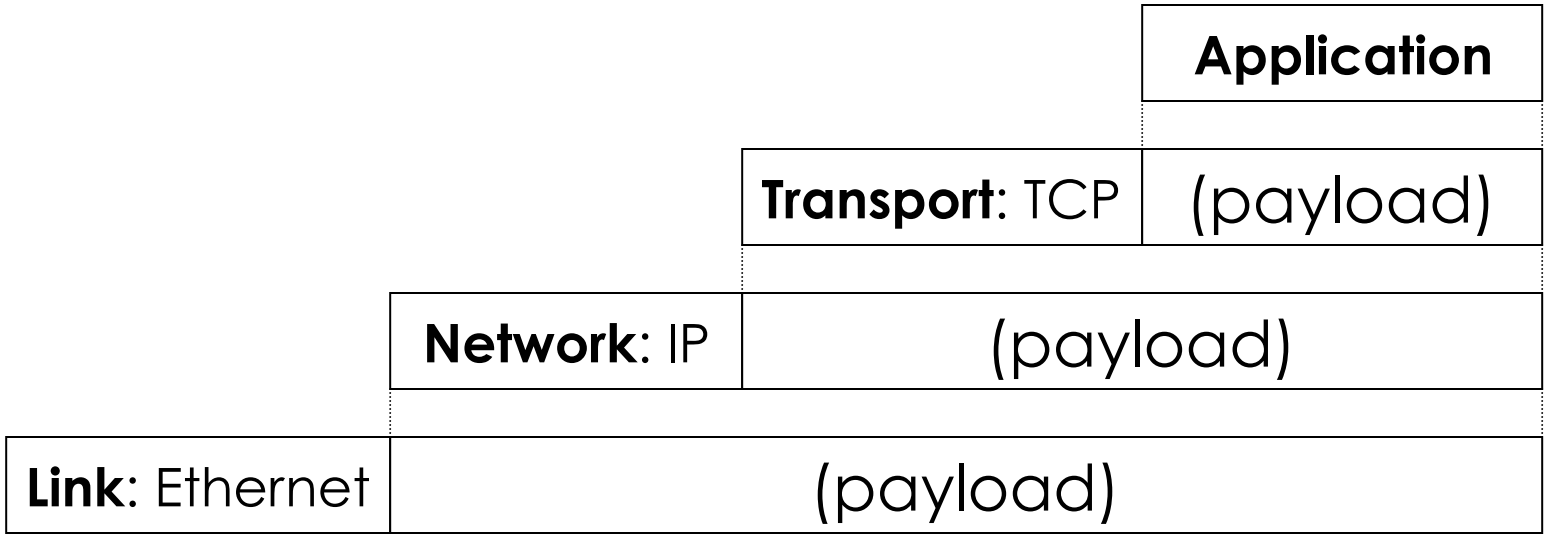
Physical

Transport

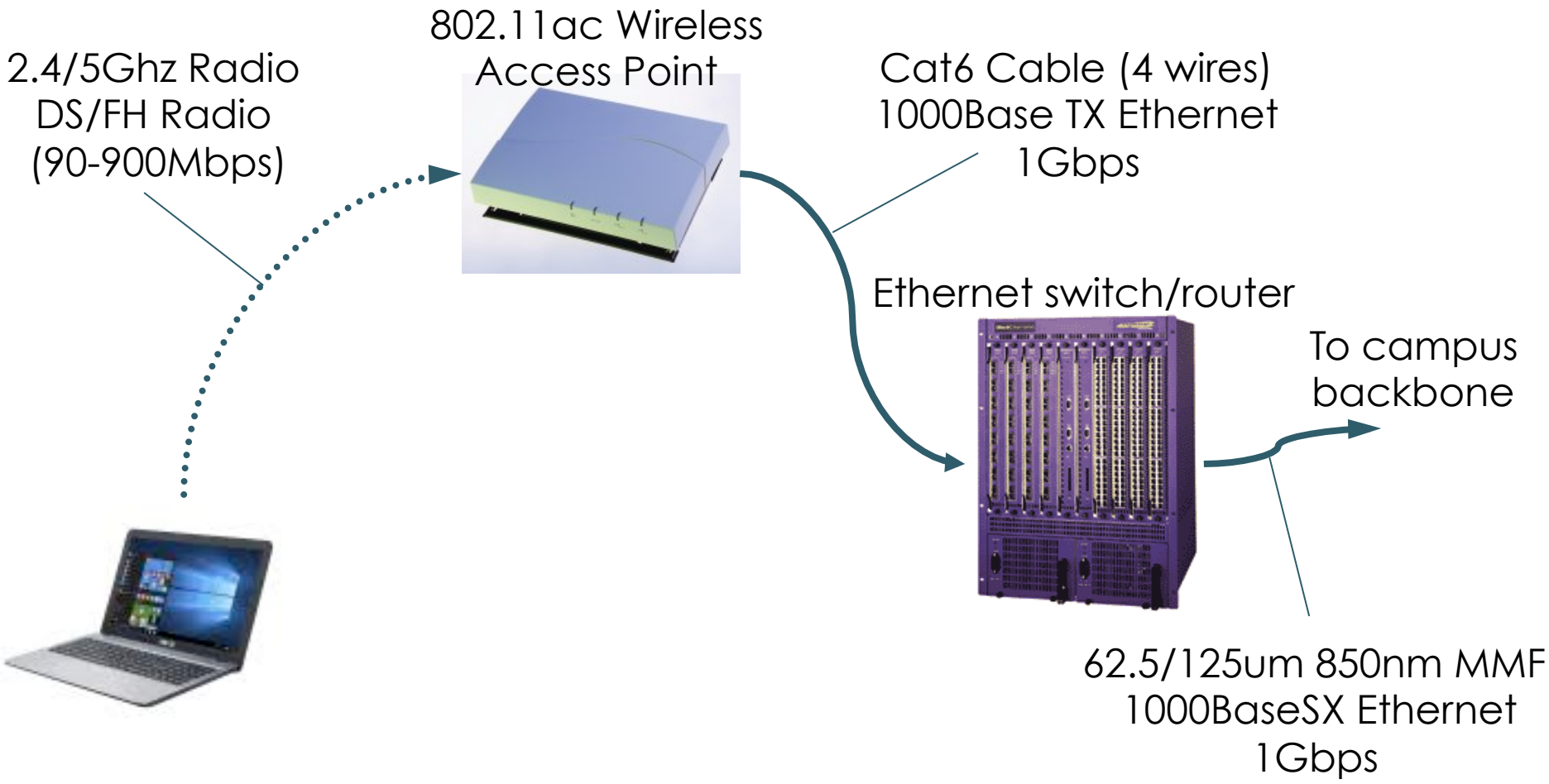
Work with your group to order the following layers from top to bottom…

(You have 1 minute!)

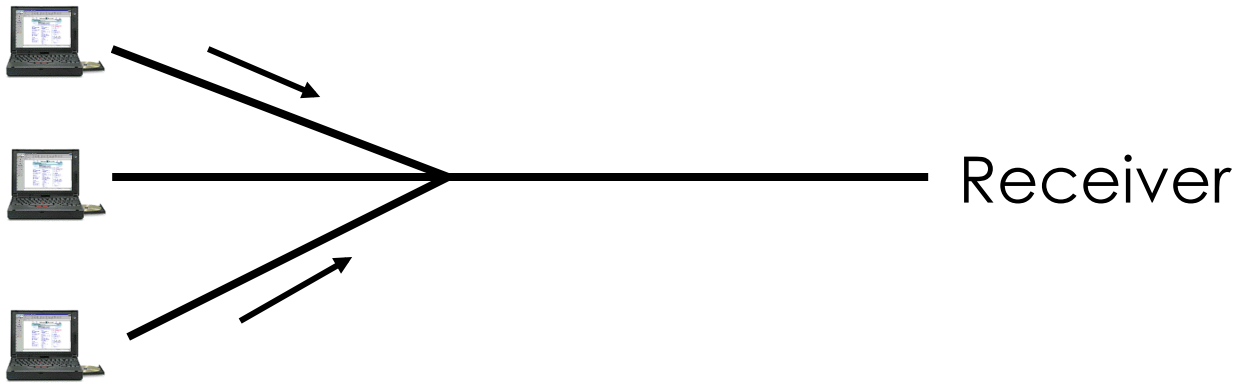
Each layer in the stack encapsulates messages from the layer above it.



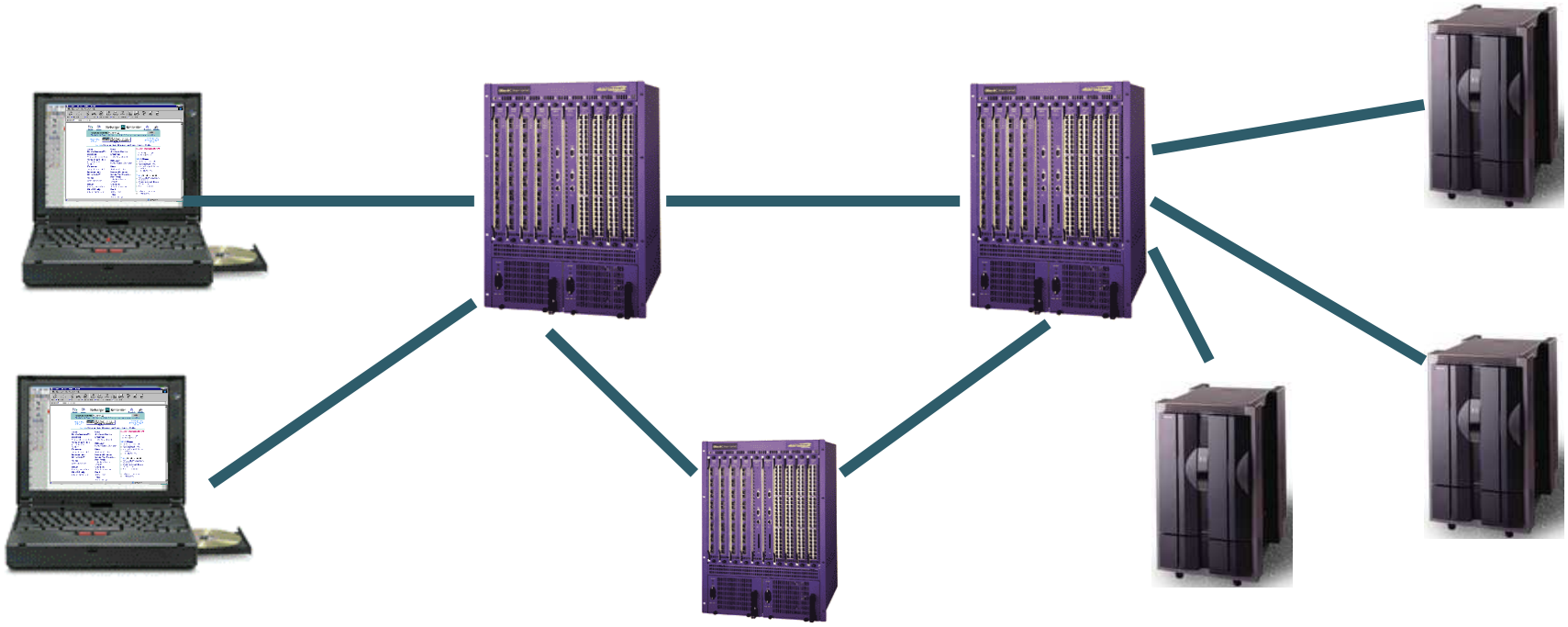
The **physical layer** is in charge of moving bits across wires, air, etc.



The **link layer** can mediate link access and provide reliability services.



The **network layer** is responsible for choosing paths in a network.



Hosts need a path between them in order to communicate.

How would you go about picking a path for a conversation ("flow") through a network?

A.	I would choose the path for the flow at the beginning and use it for all the flow's messages.
B.	I would reevaluate the path choice for each of the flow's messages.
C.	I would do something else.

The **transport layer** may provide a number of services.

- Example services:
  - Ordering
  - Error checking
  - Delivery guarantee
  - Congestion control
  - Flow control

The application layer does whatever the app programmer wants.

Each layer of the Internet model has a set of services it *can* provide.

Application: Whatever the user wants

Transport: End-to-end connections, reliability

Network: Routing

Link: Mediation, reliability

Physical: Bits across a physical medium

Is all this layering **good** *or* **bad**?

A.	**Good**: It lets us subdivide and abstract problems.
B.	**Bad**: It means we sometimes duplicate work on different layers.
C.	**Bad**: We could make better decisions with all information available on all layers.
D.	**Other**: I'll explain!

Who/what should address our concerns?

Networks have many concerns, e.g. reliability, error checking, naming and data ordering. Who/what should be responsible for addressing them?

A.	The network should take care of these for us.
B.	The communicating hosts should handle these.
C.	Some other entity should solve these problems.

The "End-to-End" argument guides many network design decisions.

"Don't provide a function at lower level of abstraction (layer) if you have to do it at higher layer anyway… unless there is a very good performance reason to do so."

Saltzer, Reed, Clark, "End-To-End Arguments in System Design," ACM Transactions on Computer Systems, Vol. 2 (4), pp. 277-288, 1984.

An HTTP (web) request traverses the network stack, crossing multiple devices.