CS 43: Computer Networks

02: Protocols and Layering

September 2, 2026

Slides adapted from Kurose & Ross, Kevin Webb, Vasanta Chaganti, Laurent Vanbever

Announcements

- Register your Clicker: https://join.iclicker.com/DQLN
- No class Monday
- Quiz next Wednesday covering this week's concepts
  - Includes anything in reading, lab, and lecture
  - Study guide goes out by Thursday night
- OH tomorrow at 1pm-2pm in Martin 338 (my office)

Example: Reading Quiz
(this is not graded for points today)

Note: Reading quiz slides will have a red border.

The Internet model discussed in the textbook has ___layers.

A. 2
B. 5
C. 7
D. 10

A protocol defines:

A. The wiring between hosts and routers
B. The format and order of messages, actions taken when they’re sent or received
C. A program that displays web pages
D. The rules for naming websites
E. A type of network link

End of Reading Quiz!

The IP protocol is implemented at which layer of the Internet?

A. Application layer
B. Transport layer
C. Network layer
D. Link layer
E. Physical layer

Note: Peer instruction slide question will have a green border.

The IP protocol is implemented at which layer of the Internet?

A. Application layer
B. Transport layer
C. Network layer
D. Link layer
E. Physical layer

Note: Peer instruction slide question will have a green border.

In today's lecture, we will discuss the **"how" and "why" of protocols**, layering, and encapsulation:

Example questions you should be able to answer:

Every device connected to the Internet has to implement the IP protocol. Why? How?

Why structure Internet protocols into layers?

Write down your answer:

Why does every device connected to the Internet have to implement the IP protocol?

Ok, if you don’t know! Take a guess…

Write down your answer:

Why structure the Internet into layers?

Ok, if you don’t know! Take a guess…

Recall this question from last time…
Which of these are **required** to get your packet from your laptop to a Google server?

A. Manage complexity and scale up

B. Naming and addressing

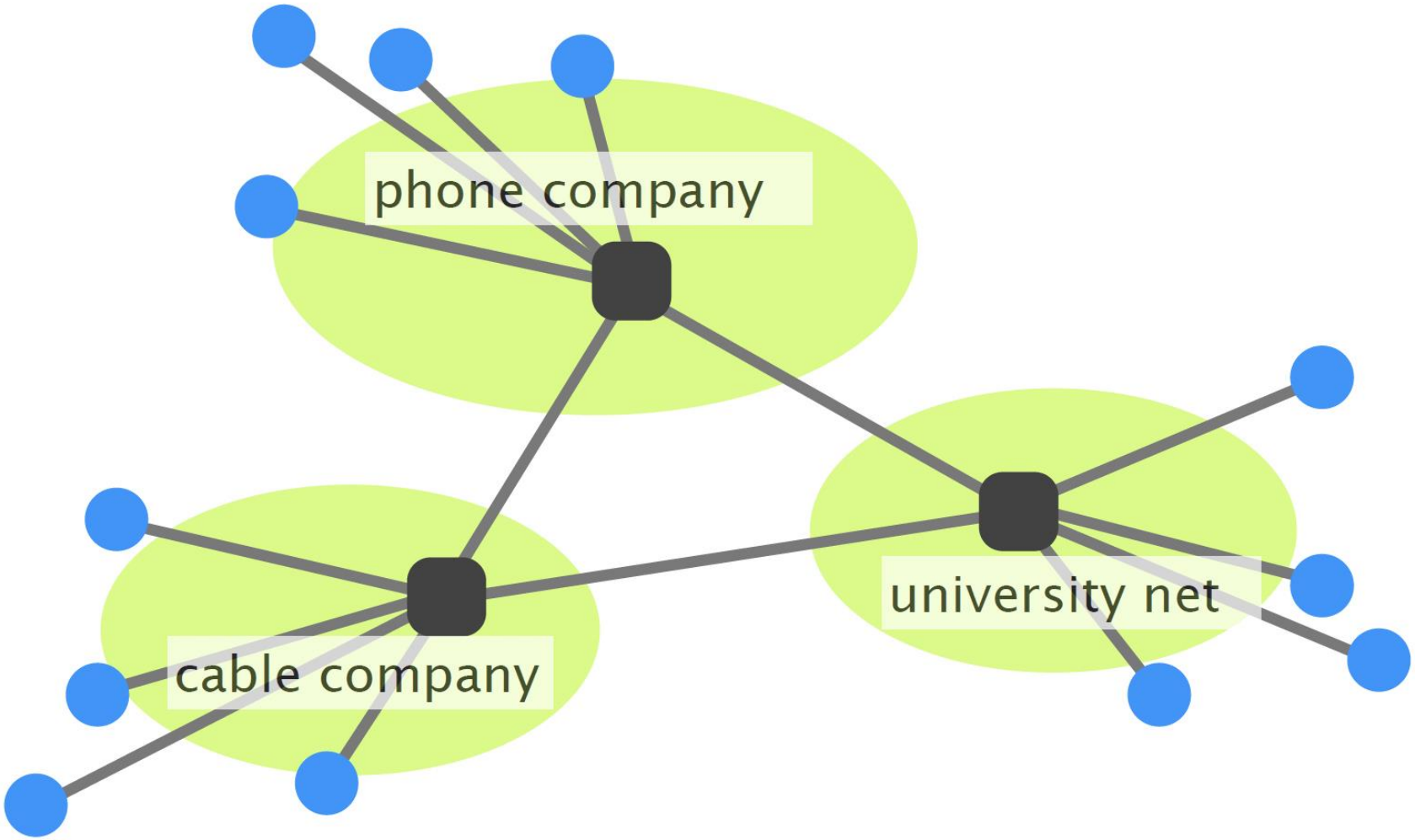
C. Moving data to a destination

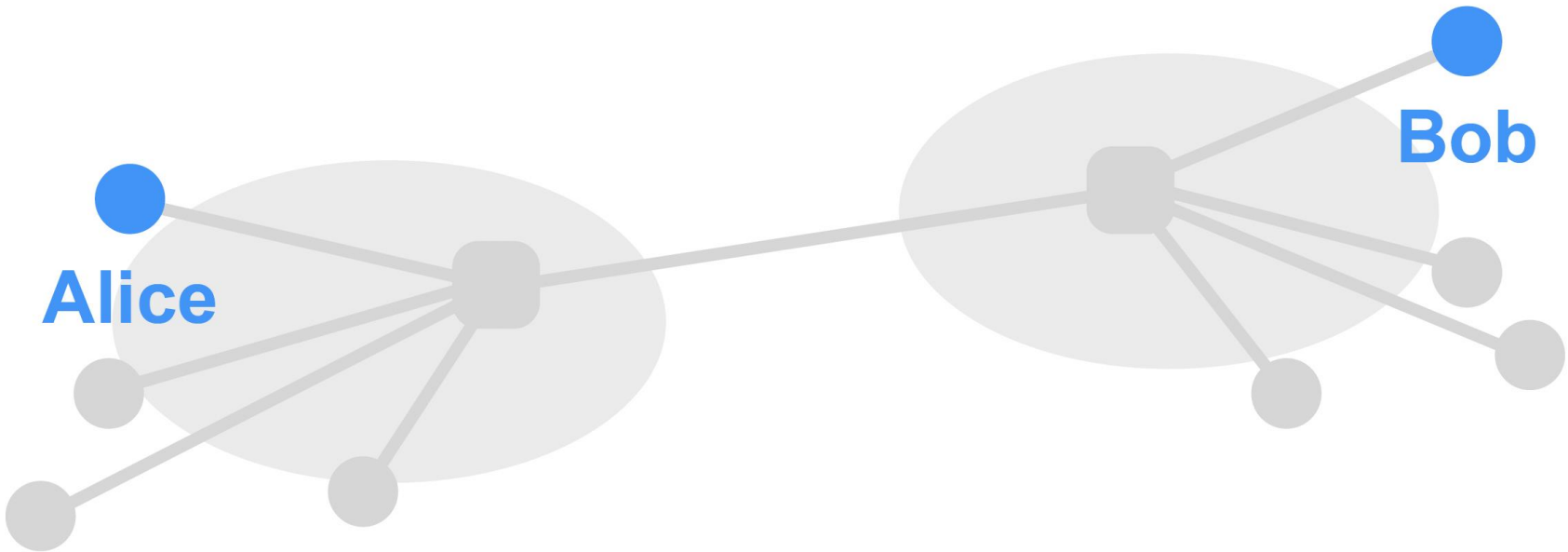
D. Reliability and fault tolerance

E. Security and Privacy

We need everything for modern reliable and secure communication, however, only B & C are strictly required for the network to function so your packet *can* be delivered.

The Internet should allow **processes** on **different hosts** to exchange data, everything else is just complementary…


phone company
cable company
university net

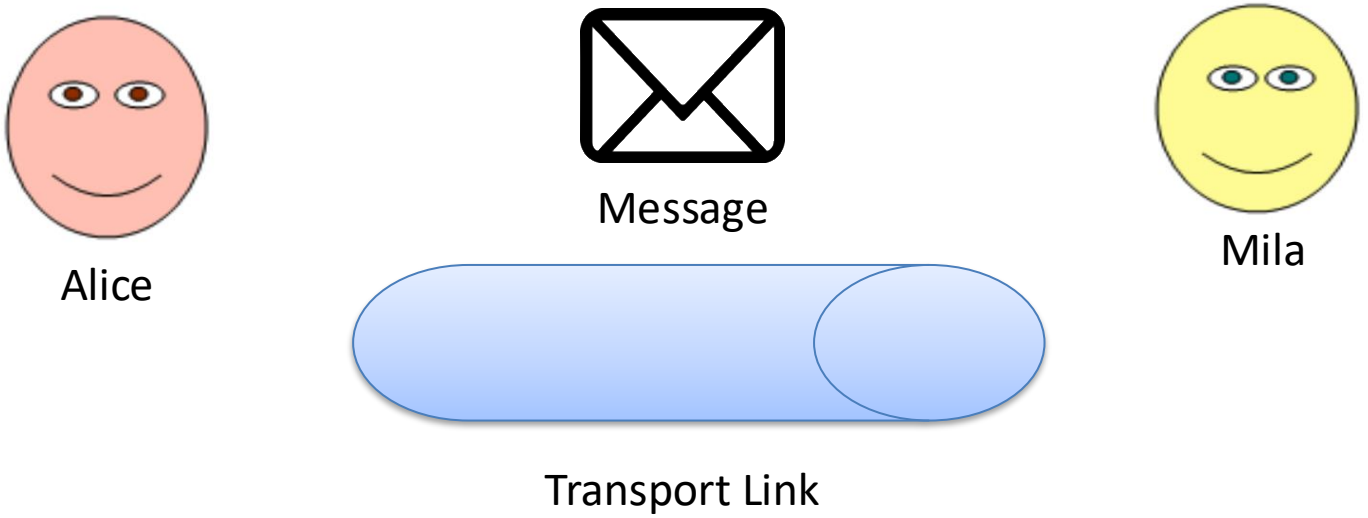

Alice
Bob

A ”Simple” Task: Sending a message from host to destination

But first... let’s try the postal system, something we are all (still!) familiar with and address a couple of key challenges...

A “Simple” analogous task: Post-it Note

Alice and Mila are Swatties starting out their semester and are roommates. Alice wants to leave Mila a written reminder to get milk.

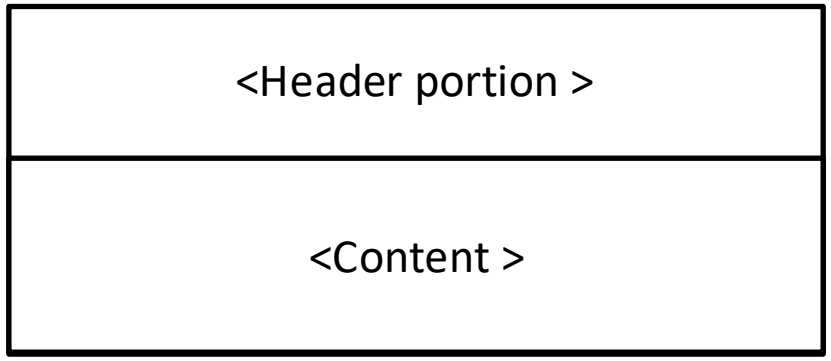


WORKSHEET
A “Simple” analogous task: Post-it Note [4 minutes]

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk. Figure out some key tasks:

1. Construct the message. Write out exactly what Alice posts to Mila. (Example on class slides.)

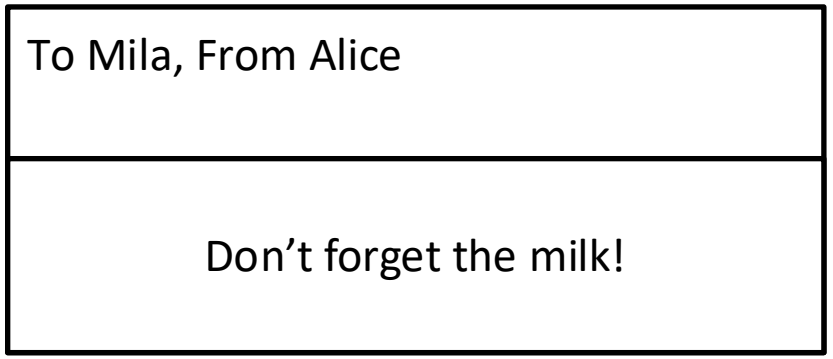
2. Metadata. Besides the sender and receiver, what other information about the message might Alice include so Mila interprets it correctly? List at least two things.



A “Simple” analogous task: Post-it Note

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk.

1. **Structure of the message: (Alice to Mila)**

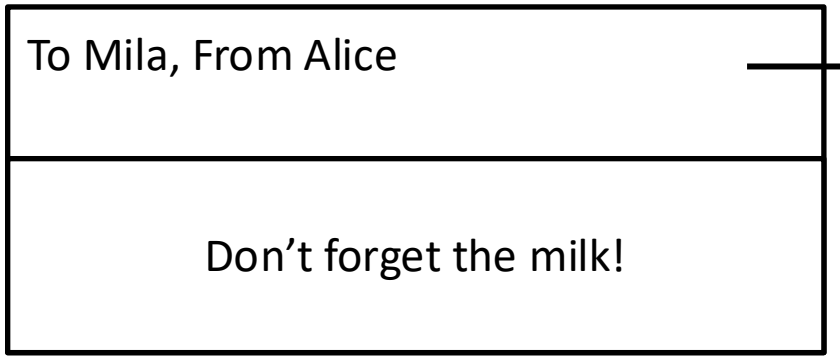


Irrespective of the source and destination, the format of the message stays the same.

A “Simple” analogous task: Post-it Note

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk.

1. **<u>Structure of the message: (Alice to Mila)</u>**



Header: contains sender/receiver information.

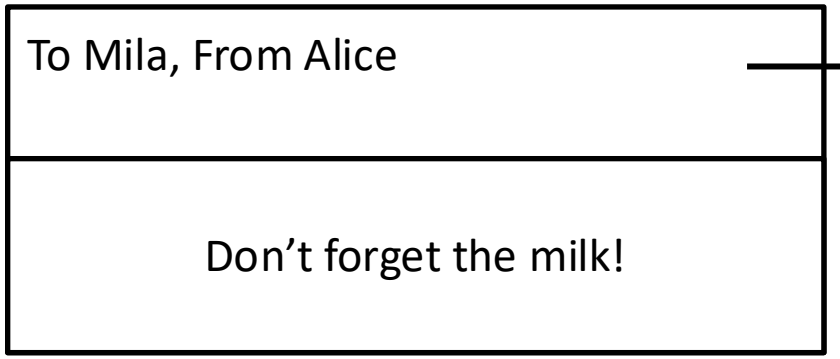
- metadata about the message
- what other metadata might you add?

<u>Irrespective of the source and destination, the format of the message stays the same.</u>

A “Simple” analogous task: Post-it Note

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk.

1. **<u>Structure of the message: (Alice to Mila)</u>**



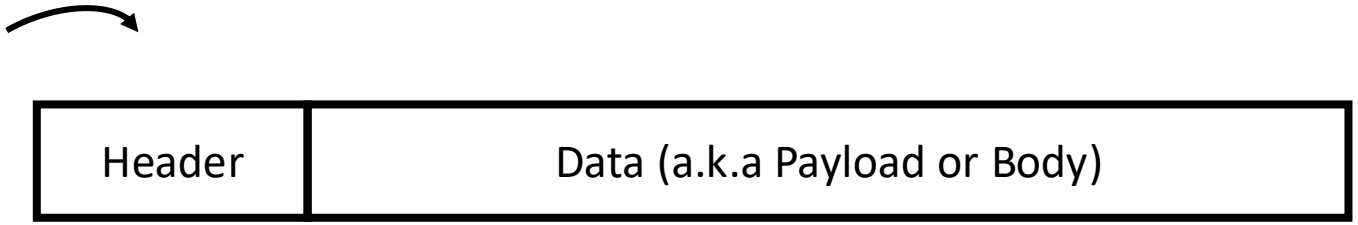
Header: contains sender/receiver information + additional state

- timestamp
- urgent! (priority)
- ordering of messages (1 of 10..)
- error control..

<u>Irrespective of the source and destination, the format of the message stays the same.</u>

Message

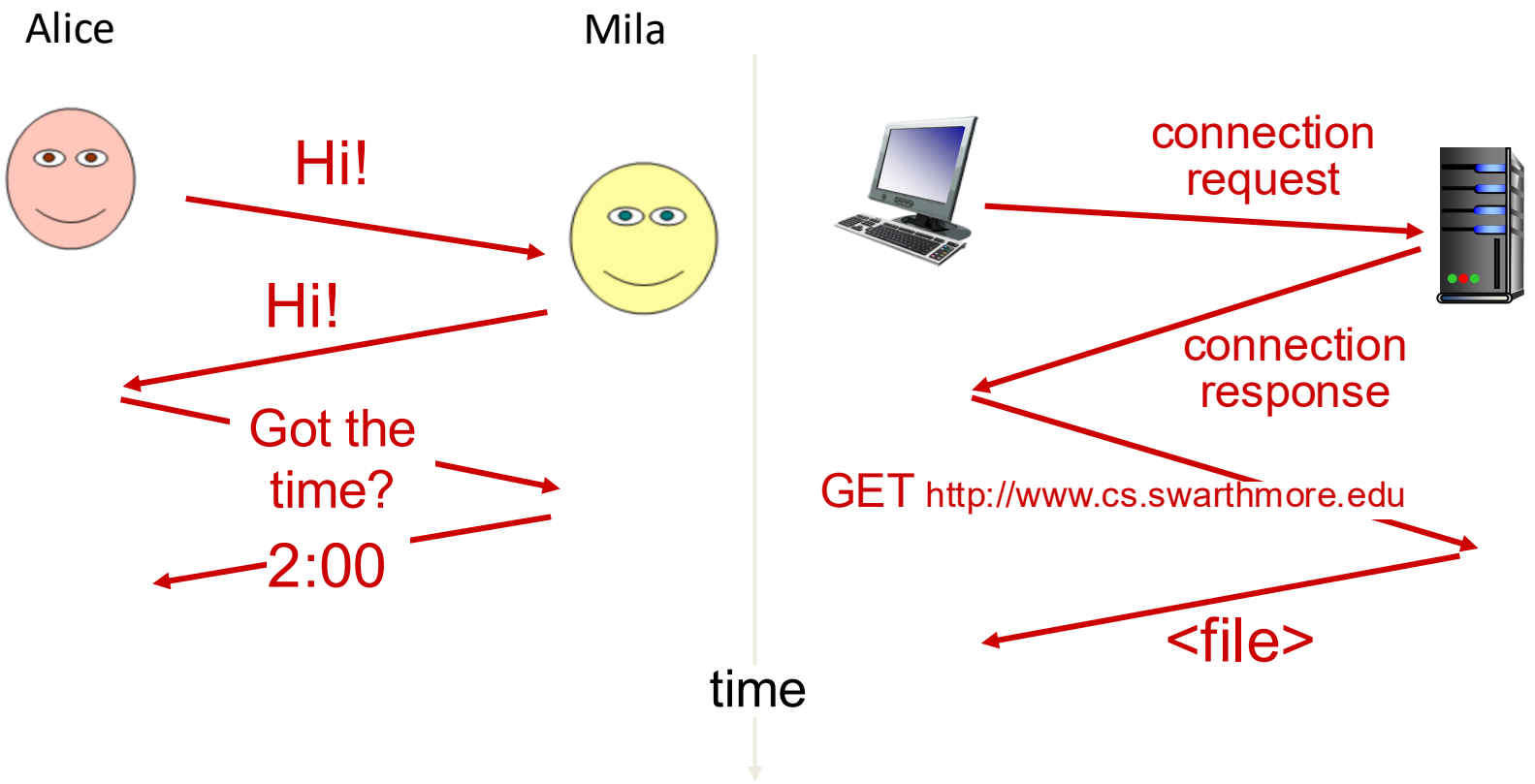
usually very small



- Message: Header + Data
- Data: what sender wants the receiver to know
- Header: information to support protocol
  - Source and destination addresses
  - State of protocol operation
  - Error control (to check integrity of received data)

What is a protocol?

Protocol: message format + transfer procedure



Human Protocol

Network Protocols (defined in RFCs)

What is a protocol?

Goal: get message from sender to receiver

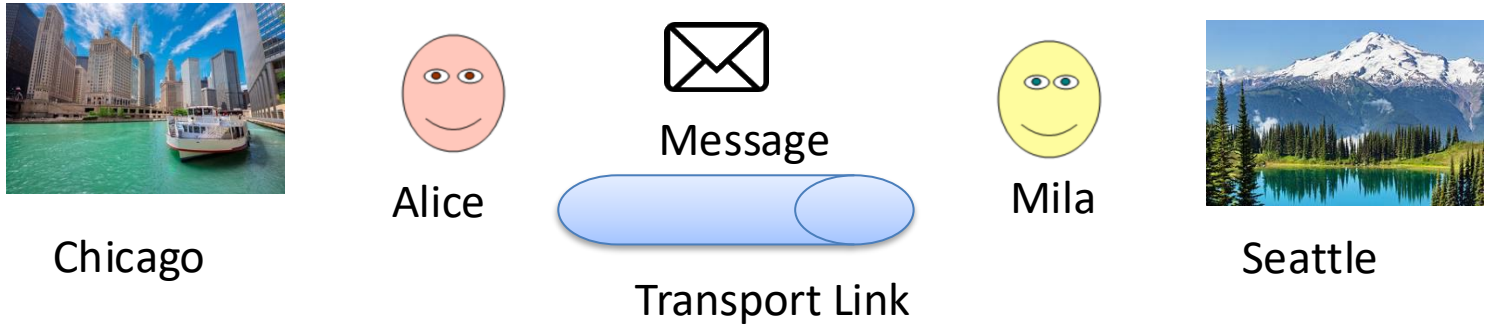
Protocol: message format + transfer procedure

- Expectations of operation
  - first you do x, then I do y, then you do z, ...

- Multiparty! so no central control
  - sender and receiver are separate processes

A “Simple” analogous task: Postal Mail [11 minutes]

Alice moves to Chicago and Mila to Seattle for summer internships. Alice wants to mail Mila a birthday card. Think of this as filling in two different pieces of information: (1) the birthday card itself, and (2) the mailing envelope.



A “Simple” analogous task: Postal Mail

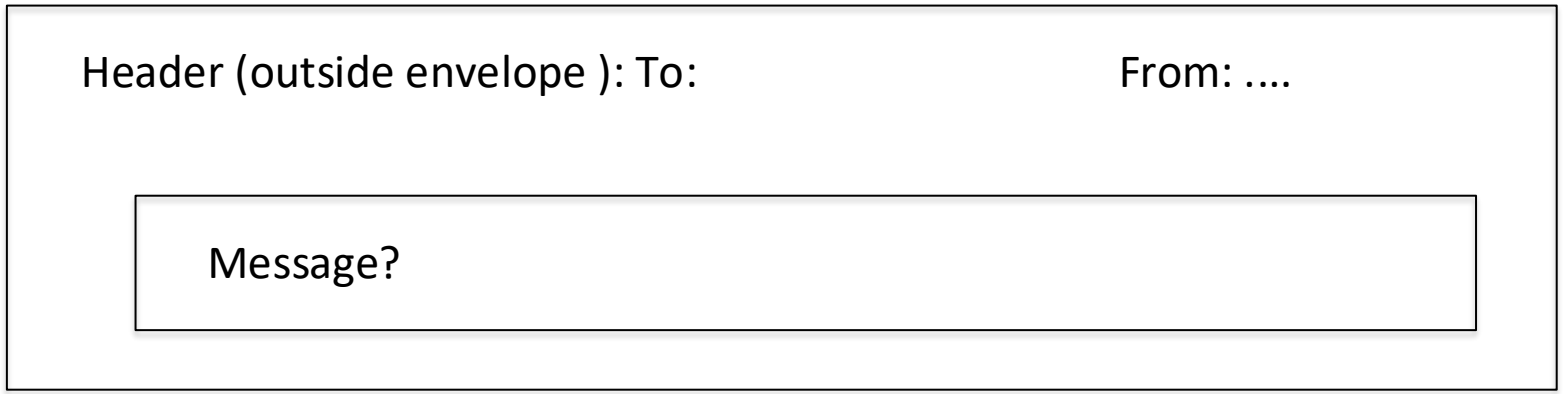
Alice would like to send Mila a birthday card.

1. **Message and header**
2. **The ”mail sending protocol”**
3. **Division of labor**
4. **Wrap-up separation of concerns**

A “Simple” analogous task: Postal Mail

Alice would like to send Mila a birthday card.

1. **<u>Message and header. What goes on the card, and what goes on the envelope? Which parts changed from Part 1, and which stayed the same?</u>**



A “Simple” analogous task: Postal Mail

Alice would like to send Mila a birthday card.

Header portion of the envelope

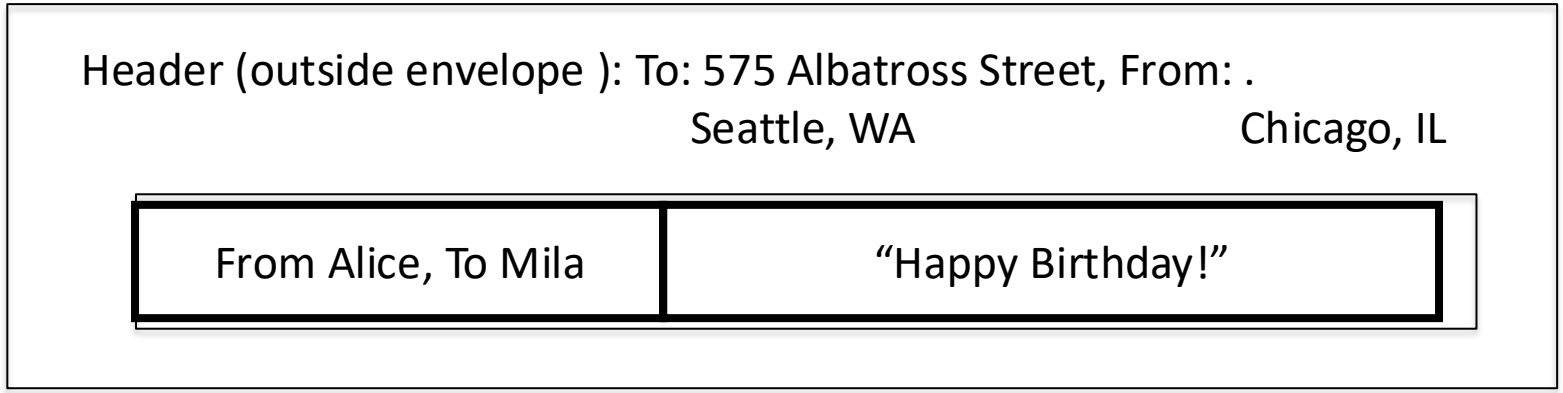
Header (outside envelope): To: 575 Albatross Street, From: .
Seattle, WA Chicago, IL

Message?

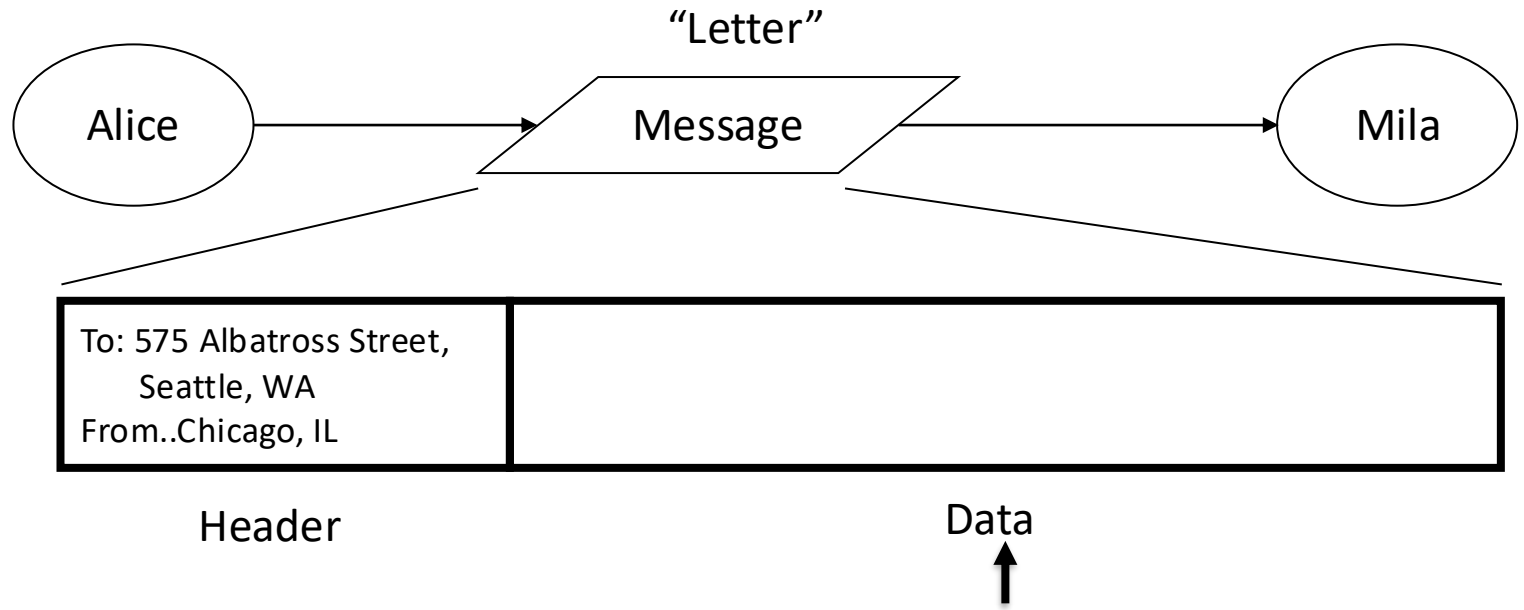
A “Simple” analogous task: Postal Mail

Alice would like to send Mila a birthday card.

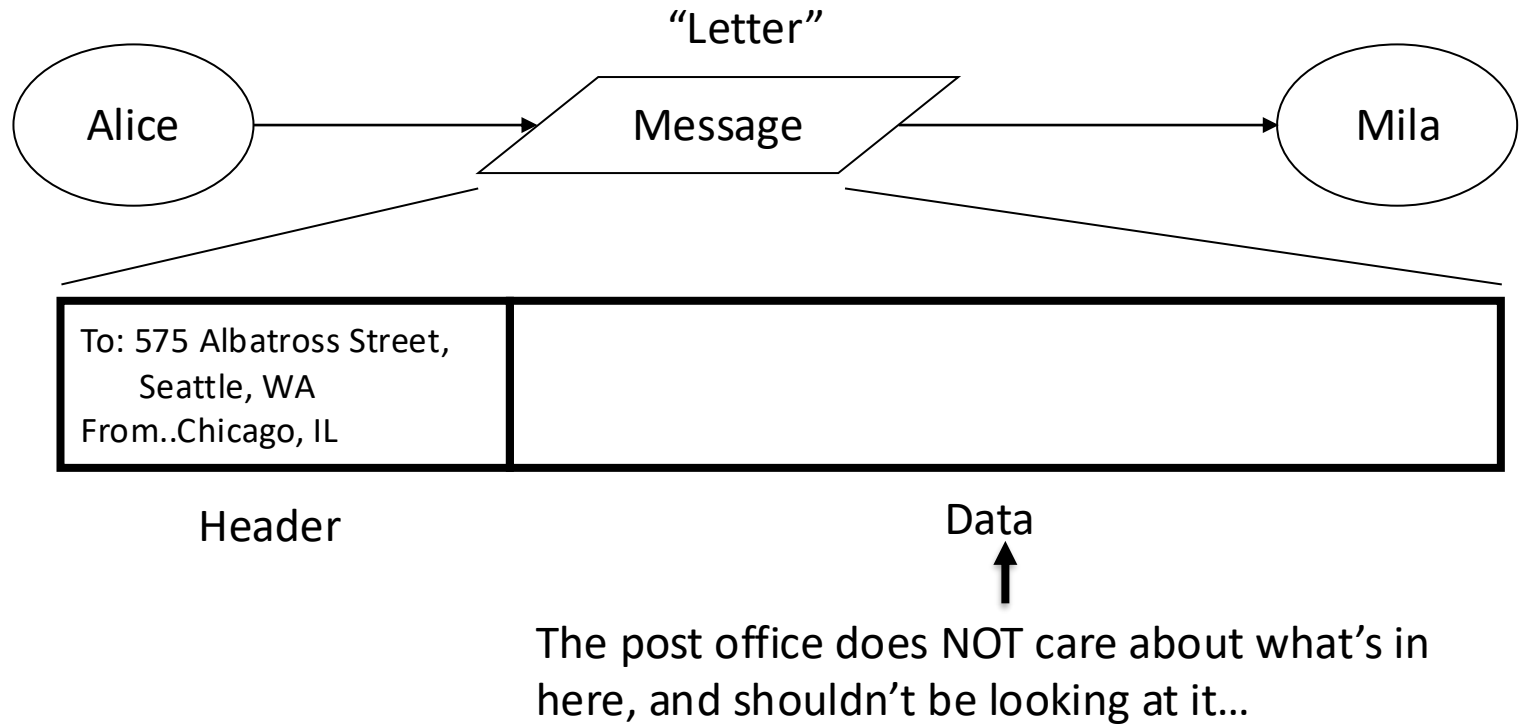
Message portion of the envelope



A “Simple” analogous task: Postal Mail

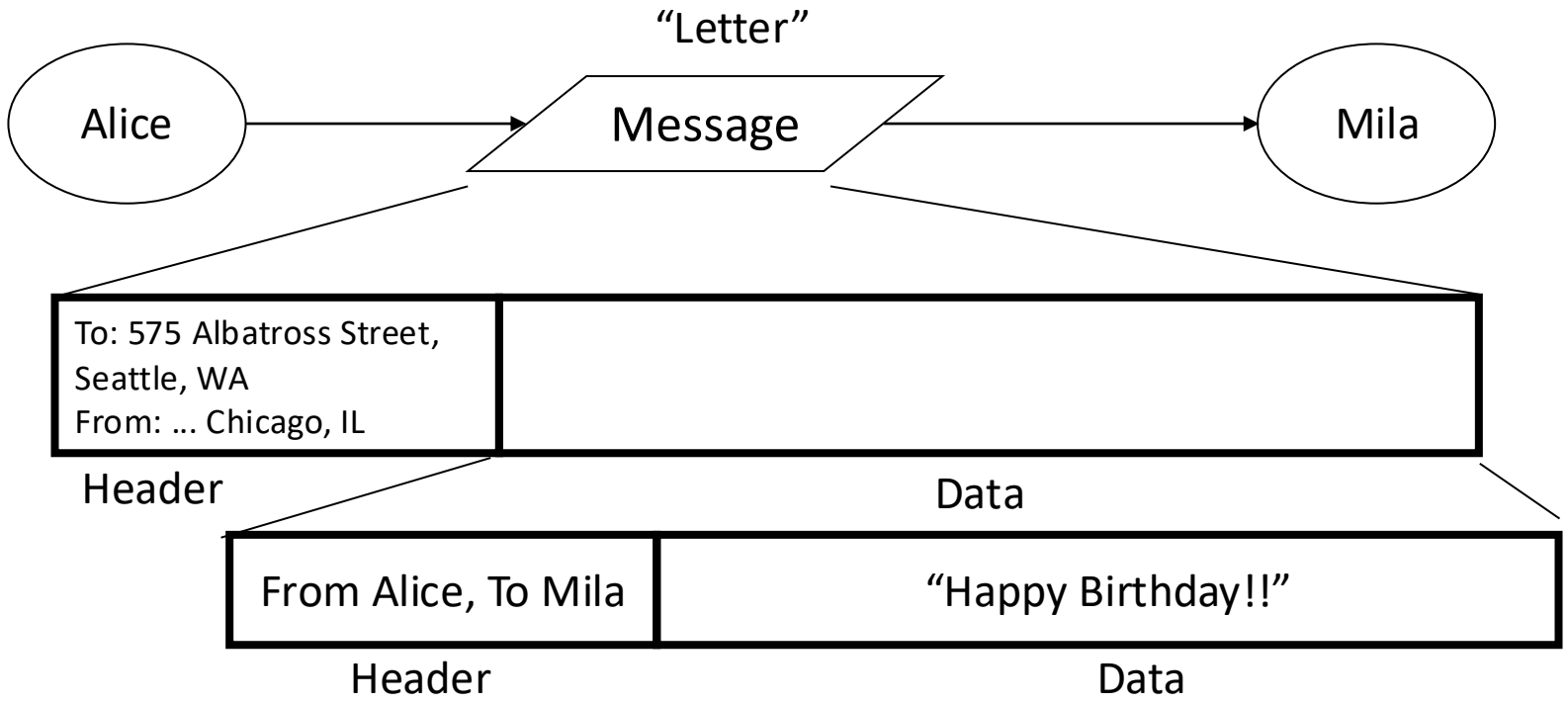


A “Simple” analogous task: Postal Mail



- **Mail Sending Protocol**
  - Message format: (from, to), message contents
  - Transfer procedure: post mail in mailbox (agreed upon convention)

A “Simple” analogous task: Postal Mail: other protocols in use?



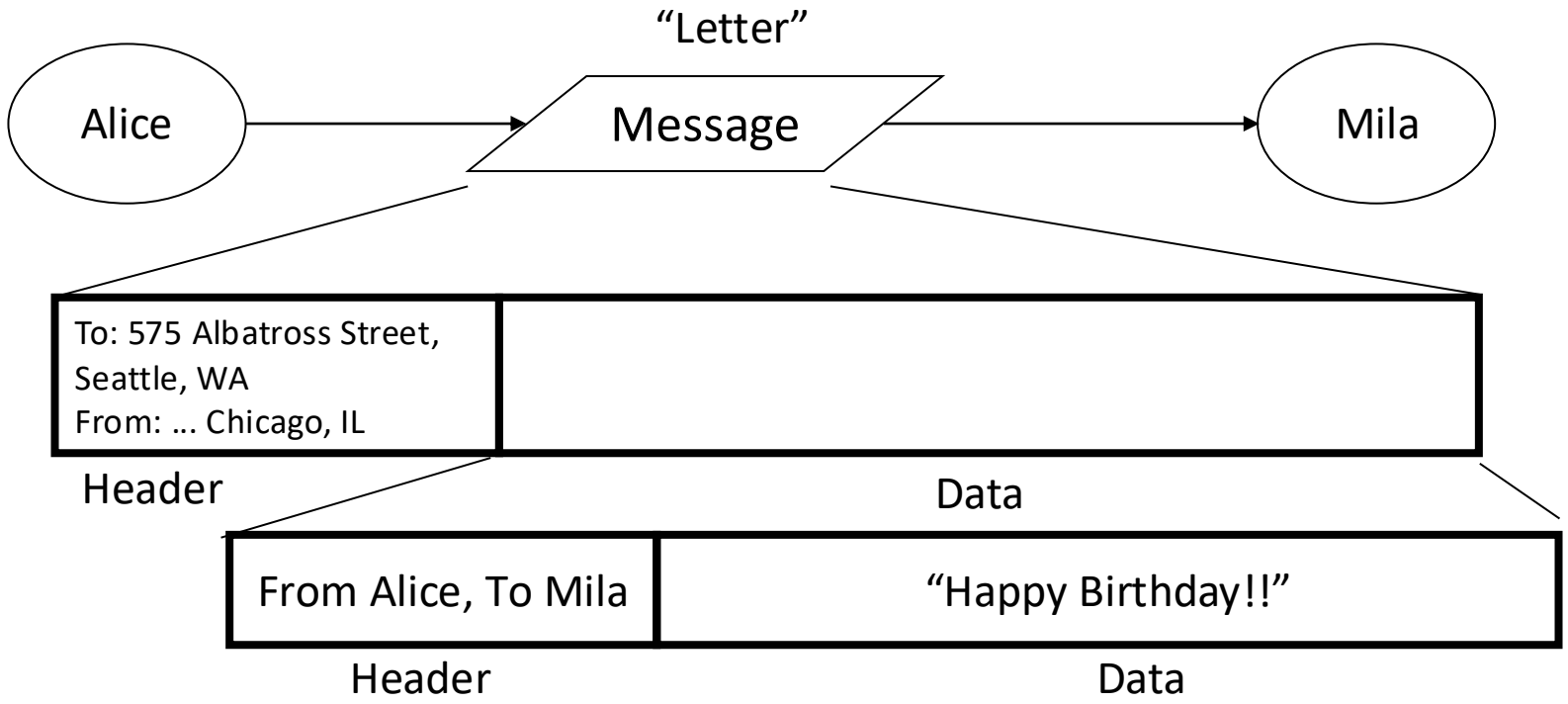
Mail Protocol

- Message format: (from, to), message contents
- Transfer procedure: post mail in mailbox (agreed upon convention)

Card Protocol (within the mail protocol!)

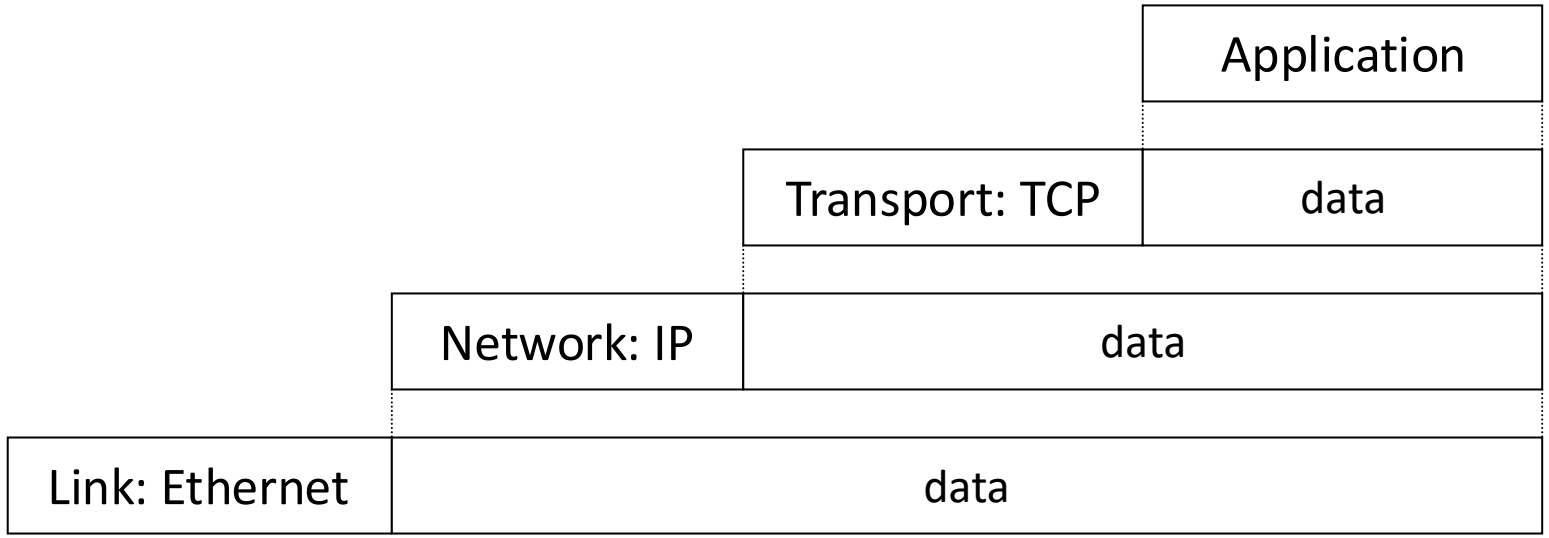
- Message format: (from, to), message contents

Message Encapsulation



- Card protocol: (message + header) treated as payload
- Put it in another protocol: append an additional header

Message Encapsulation



- Higher layer within lower layer
- Each layer has different concerns, provides abstract services to those above

A “Simple” analogous task: Postal Mail

Division of labor. Once the card is in the mailbox, whose job is it to:

a. choose the carrier (USPS, FedEx, ...)?
b. plan the route the card takes across the country?
c. physically move the card from place to place?
d. notice and fix it if the card is lost?

A “Simple” analogous task: Postal Mail

- **Message transportation and delivery**
- Who’s job is it to:

1. provide the sender and receiver addresses?
2. choose the carrier?

(1, 2): Alice decides as the “end host”

3. plan the route?
4. transport vehicles?

(3, 4): Postal Department decides as the service that provides message transfer

5. ensure the message is not lost? (reliability)

Reliability? Open question – stay tuned!

Layering: Separation of Functions

Letter: written/sent by Alice, received/read by Mila
Postal System: Mail delivery of letter in envelope

- Alice and Mila
  - 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, Mila, letter

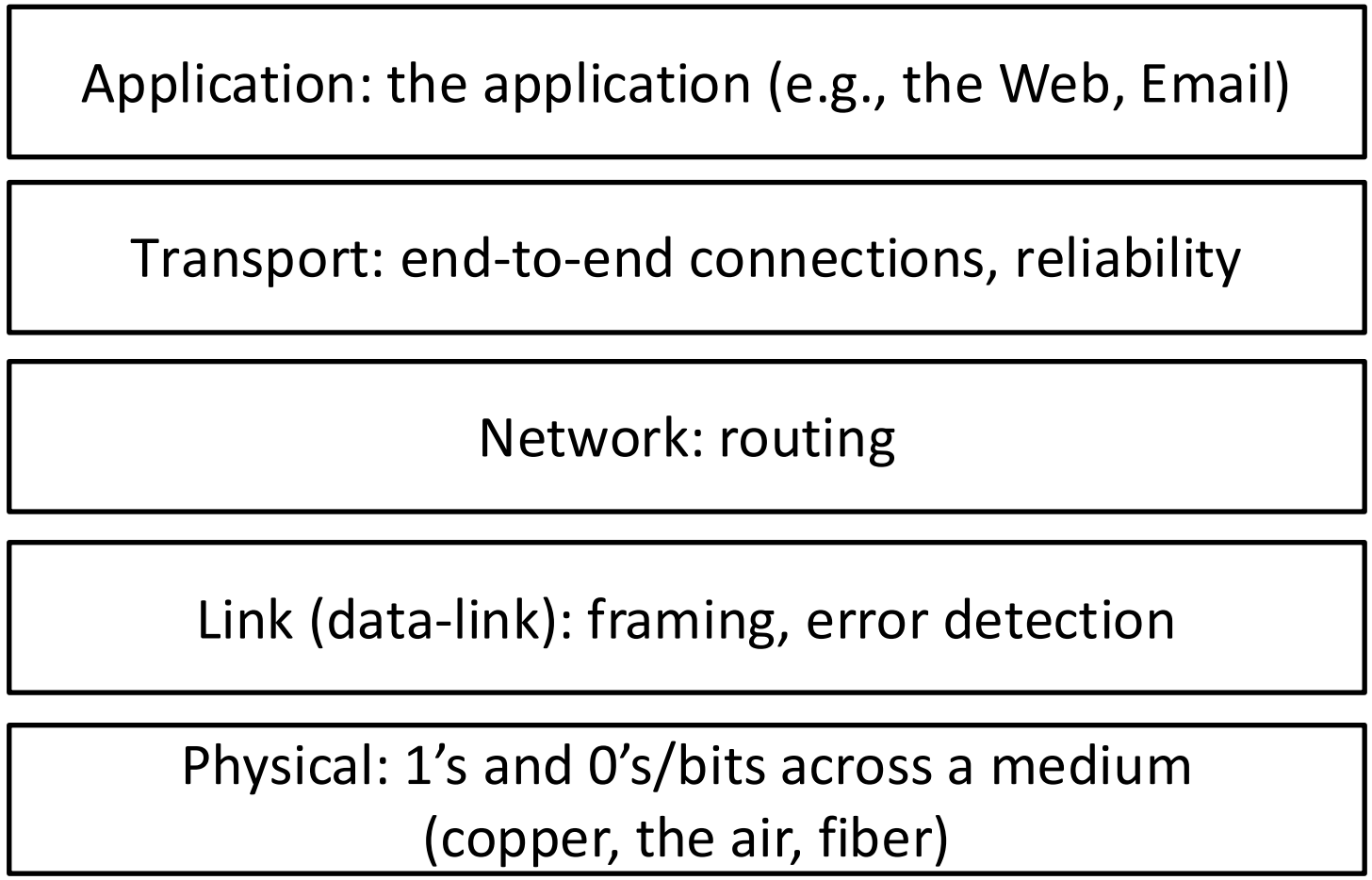
Abstraction!

- Hides the complex details of a process
- Use abstract representation of relevant properties make reasoning simpler
- Ex: Alice and Mila knowledge of postal system:
  - Letters with addresses go in, come out other side

A “Simple” analogous task: Postal Mail

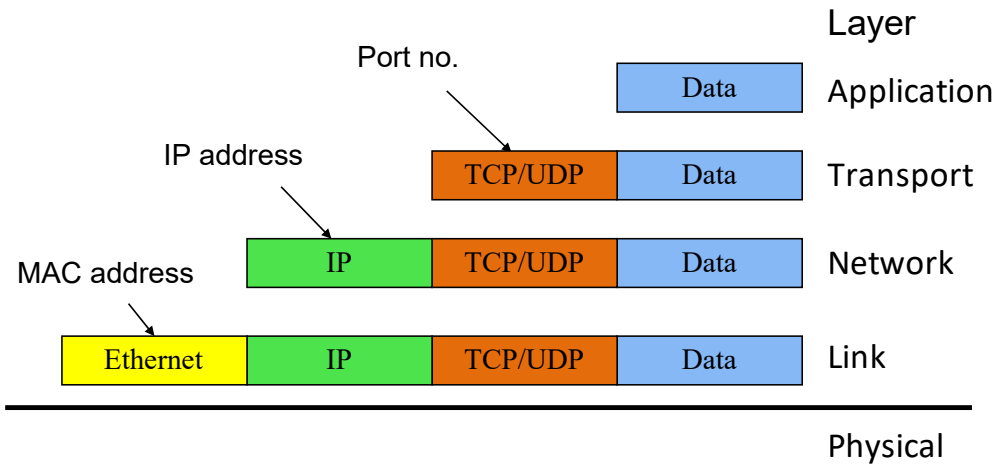
- Many more considerations..
  - Who decides the the sender and receiver addresses? Does someone maintain a mapping peoples’ names to addresses?
  - Can Mila always be guaranteed of this delivery date? What factors influence delivery ?
  - What if the mail gets lost – who’s responsibility is it? Alice, Mila or someone else?
  - What about security? privacy?

Five-Layer Internet Model



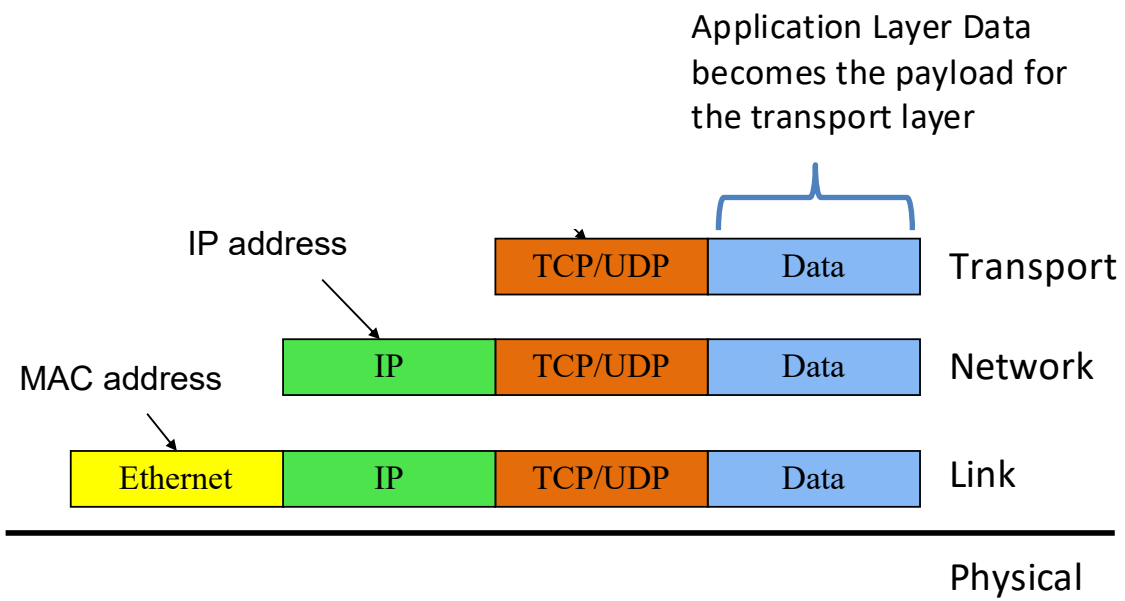
Application Layer
(HTTP, FTP, SMTP, Tiktok)

- Does whatever an application does!



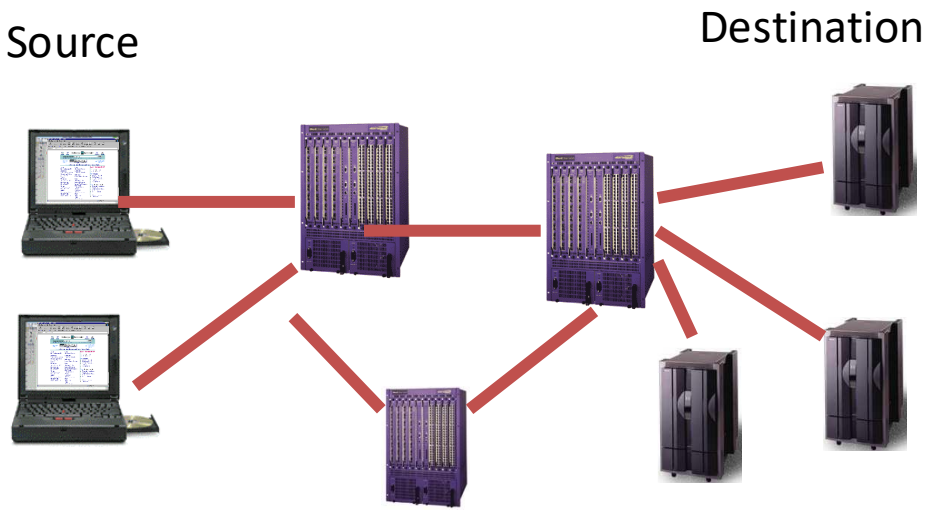
Transport Layer (TCP, UDP)

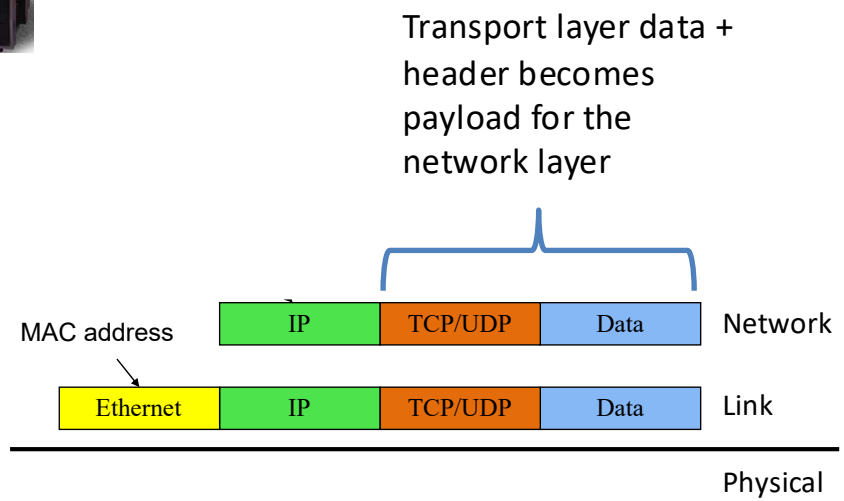
- Provides
  - Ordering
  - Error checking
  - Delivery guarantee
  - Congestion control
  - Flow control
- Or doesn't!



Network Layer (IP)

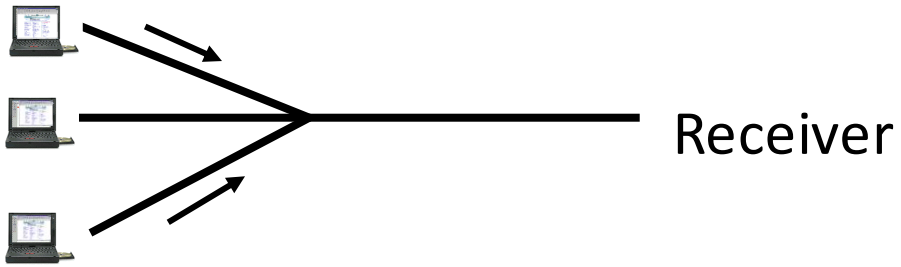
- **Routers**: choose paths through network



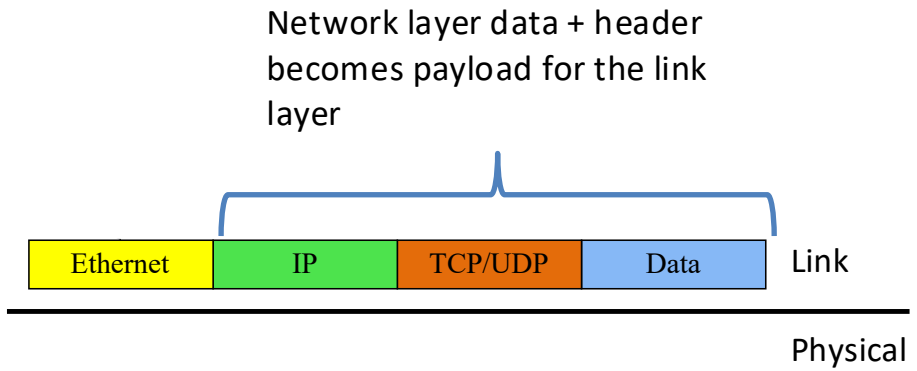


Link Layer (Ethernet, WiFi, Cable)

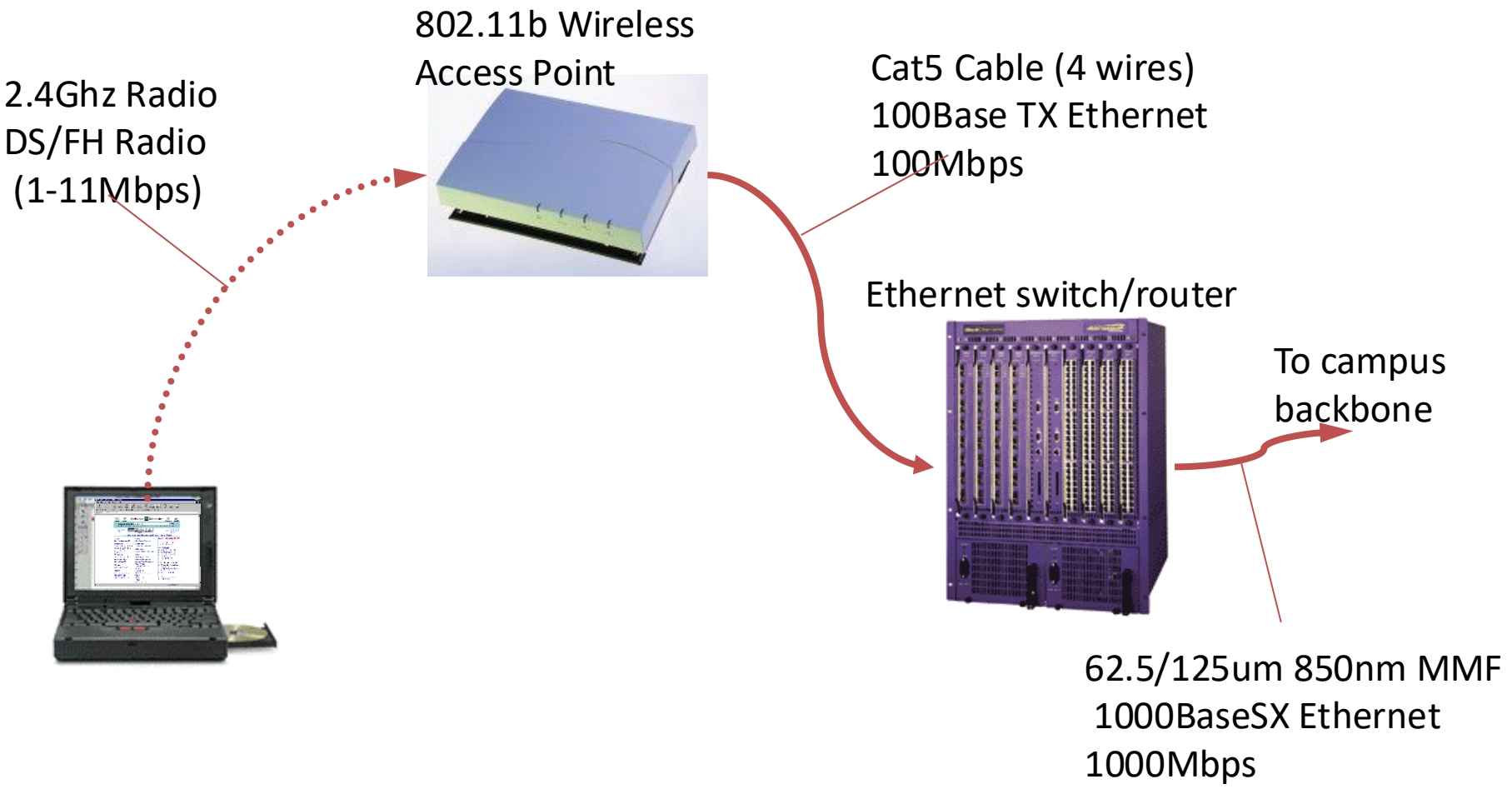
- Who's turn is it to send right now?
- Break message into frames
- Media access: can it send the frame now?



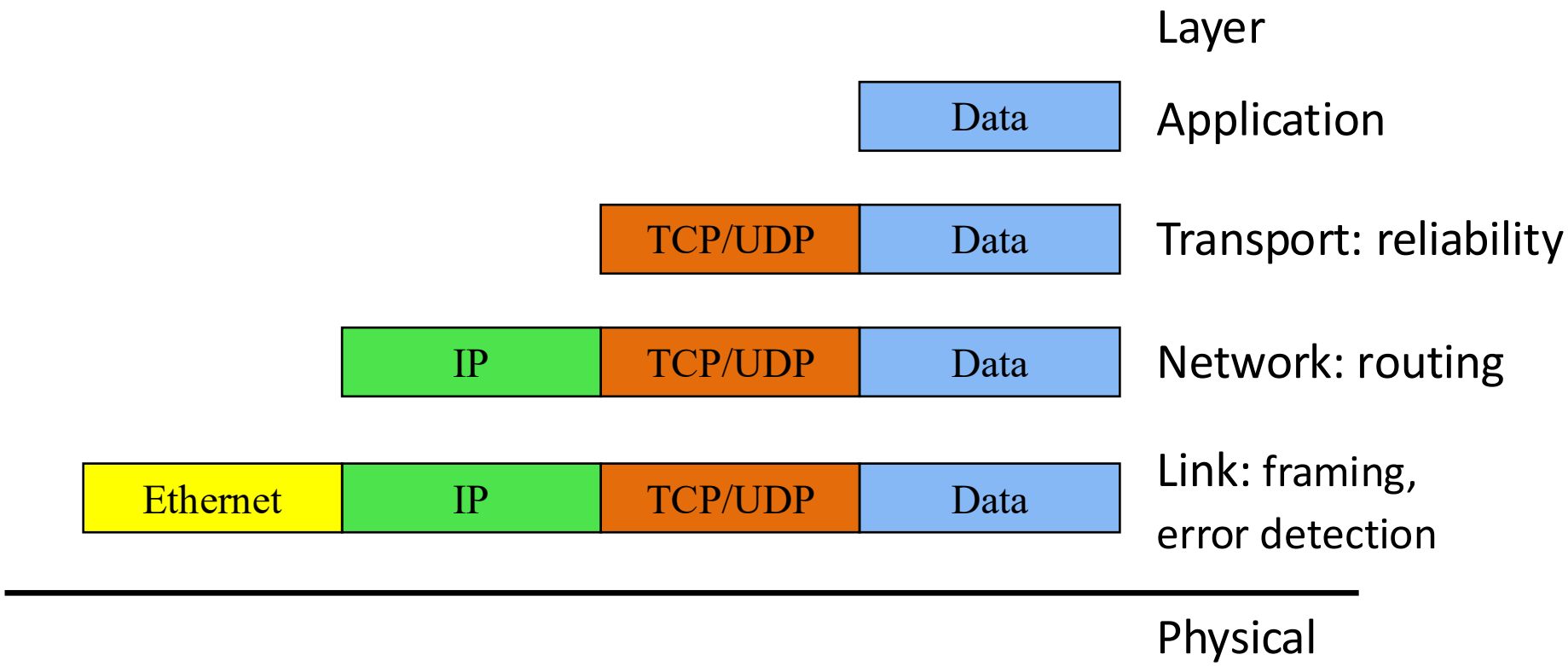
- Send frame, handle "collisions"



Physical layer – move actual bits! (Cat 5, Coax, Air, Fiber Optics)



Layering and encapsulation



Layering: Separation of Functions

- explicit structure allows identification, relationship of complex system's pieces
  - layered reference model for discussion
  - reusable component design
- modularization eases maintenance
  - change of implementation of layer's service transparent to rest of system,
  - e.g., change in postal route doesn't affect delivery of letter

Modularity is a key component of any good system

Problem

Can’t build large systems out of spaghetti code

hard (if not) impossible to understand, debug, update

need to bound the scope of changes

evolve the system without rewriting it from scratch

Solution

Modularity is how we do it

…and understand the system at a higher-level

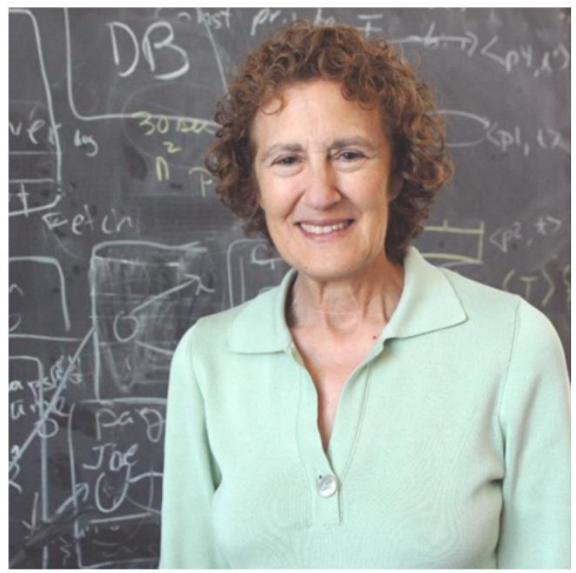
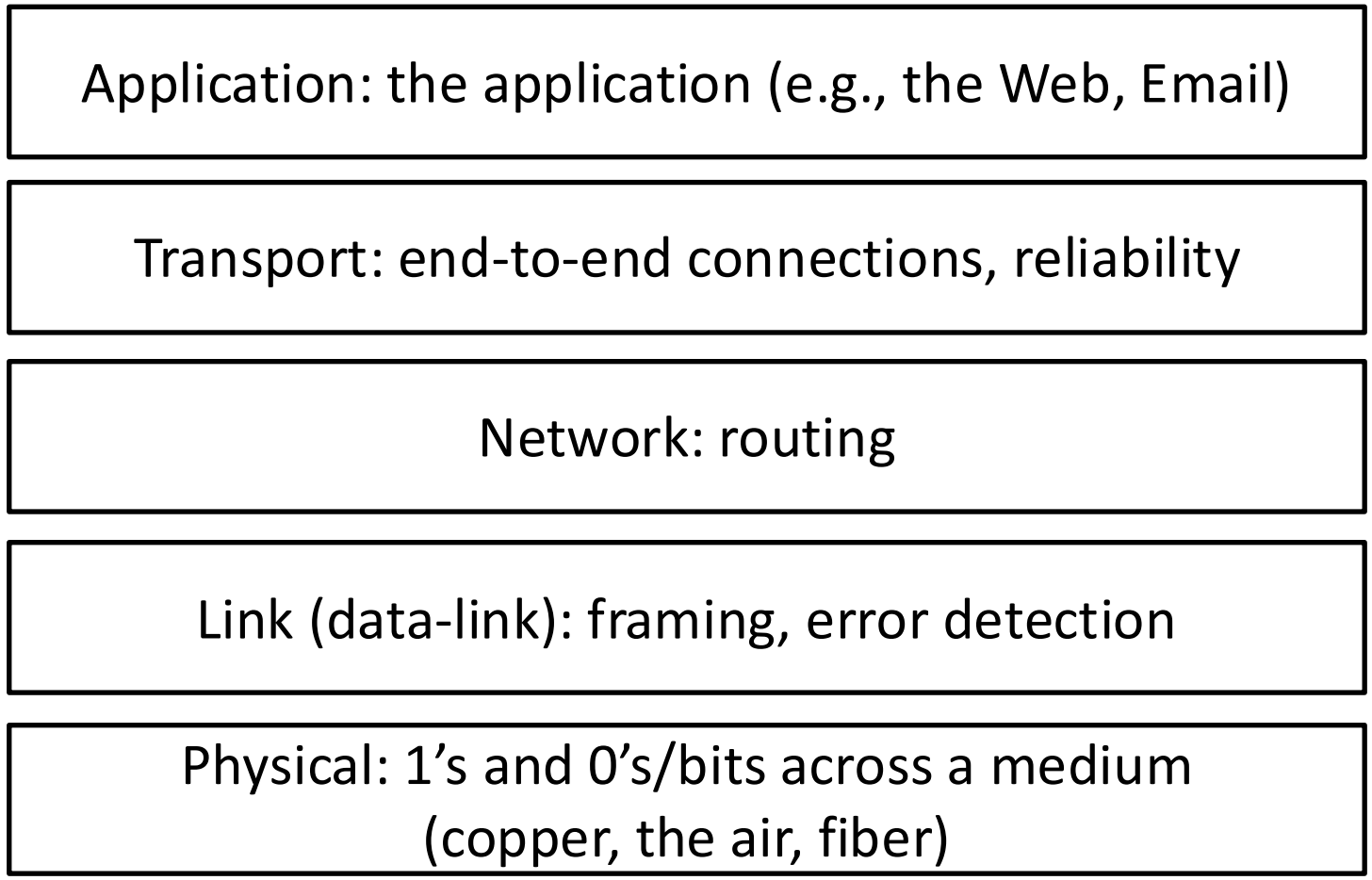


Photo: Donna Coveney

Modularity,
based on abstraction,
is ***the*** way things get done

— *Barbara Liskov*, MIT

Five-Layer Internet Model



Because of our layering abstractions, we can use any technology we want, at any layer (as long as it doesn't interfere with the other layers). (Why or why not?)

A. Always

B. Usually

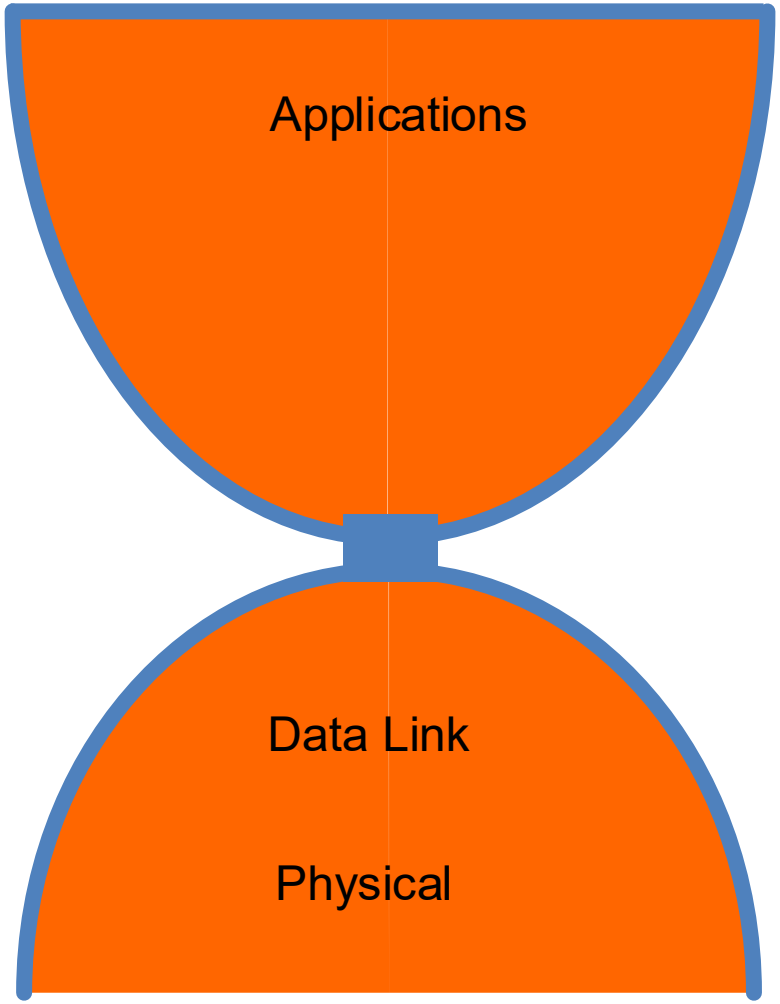
C. Sometimes

D. Never

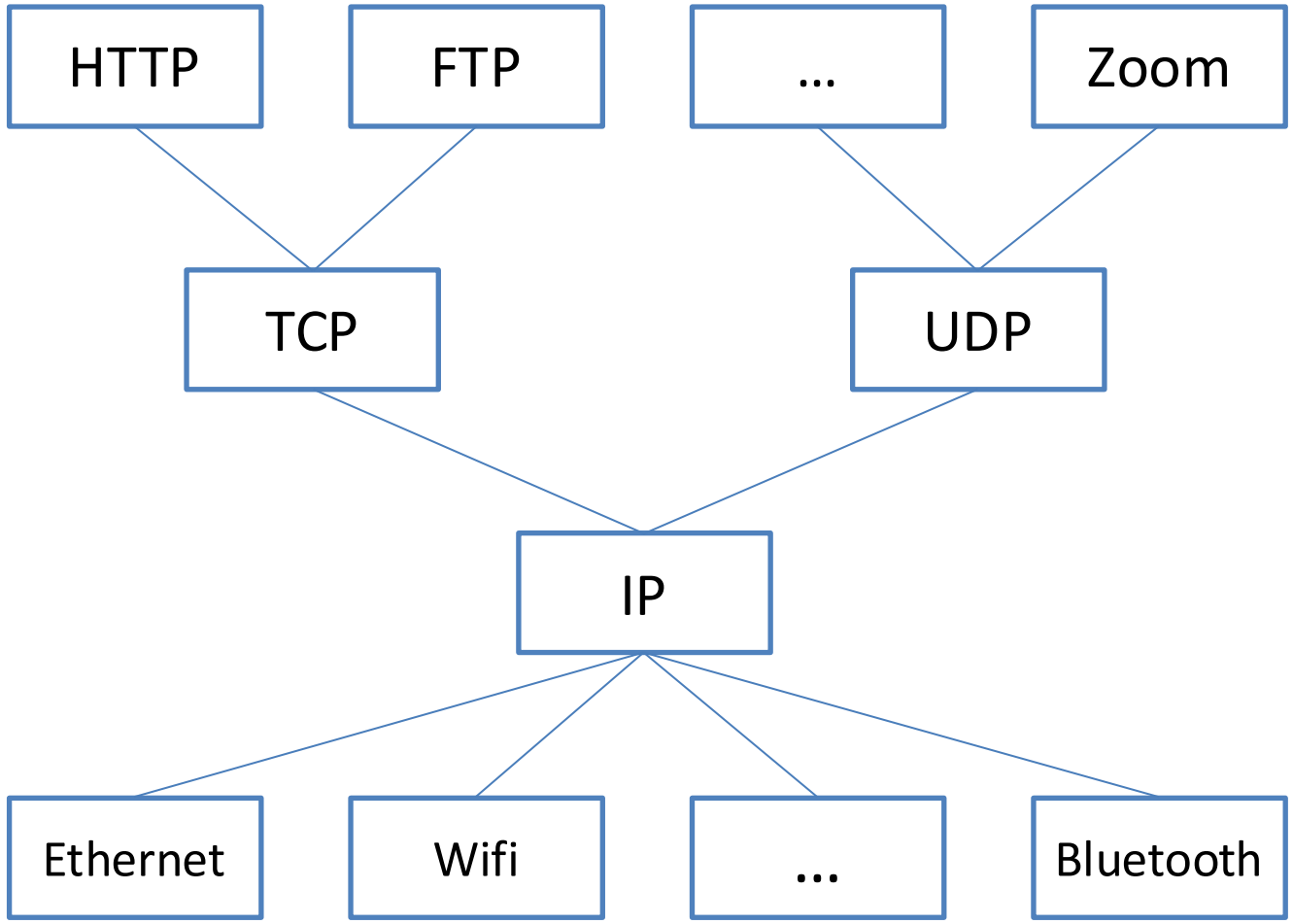
Internet Protocol Suite

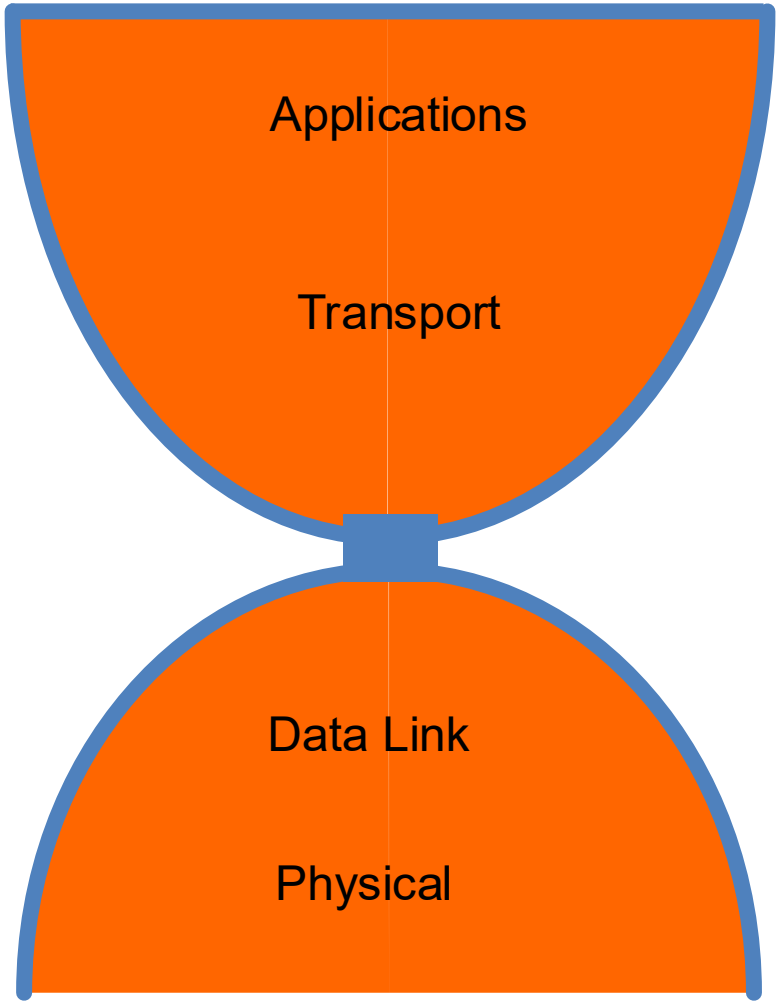
HTTP | FTP | ... | Zoom

Ethernet | Wifi | ... | Bluetooth

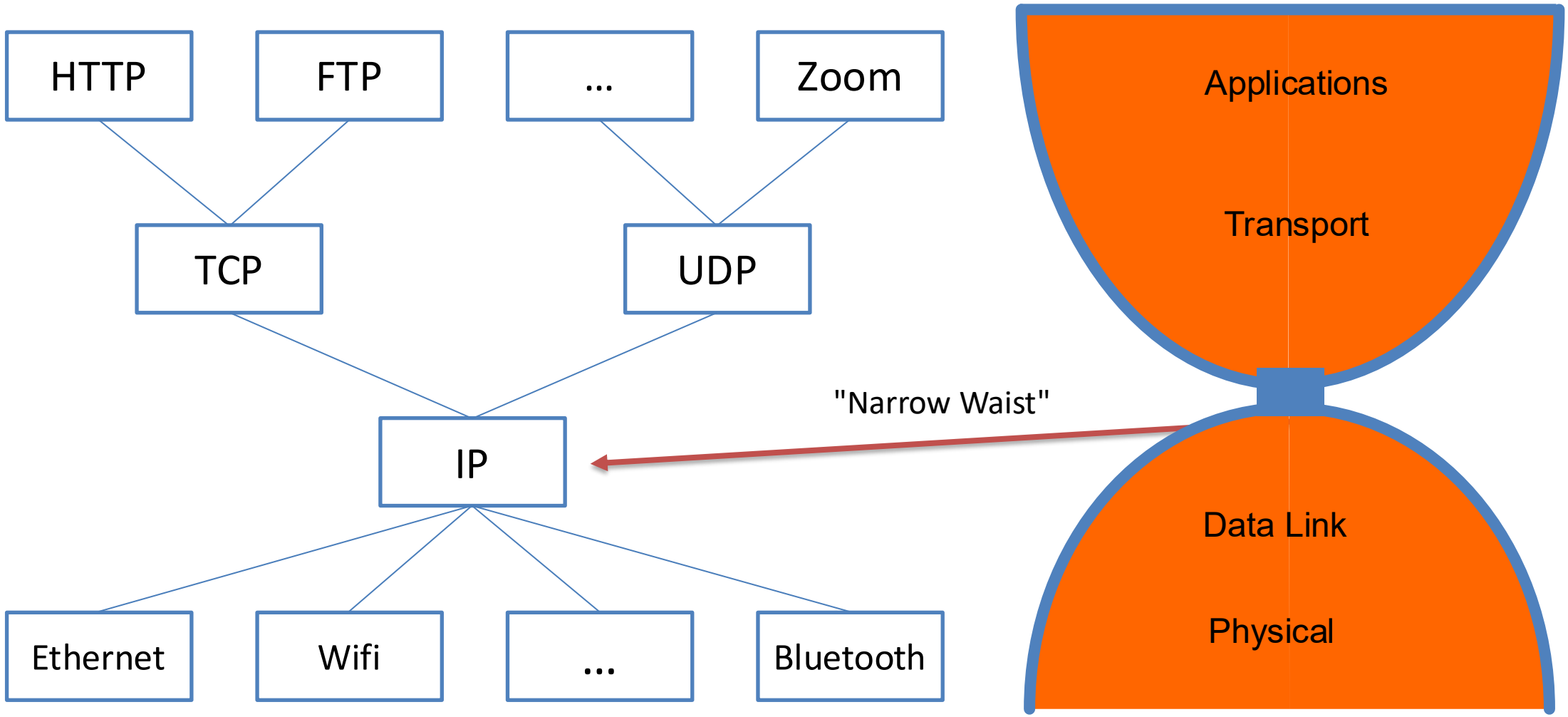


Internet Protocol Suite



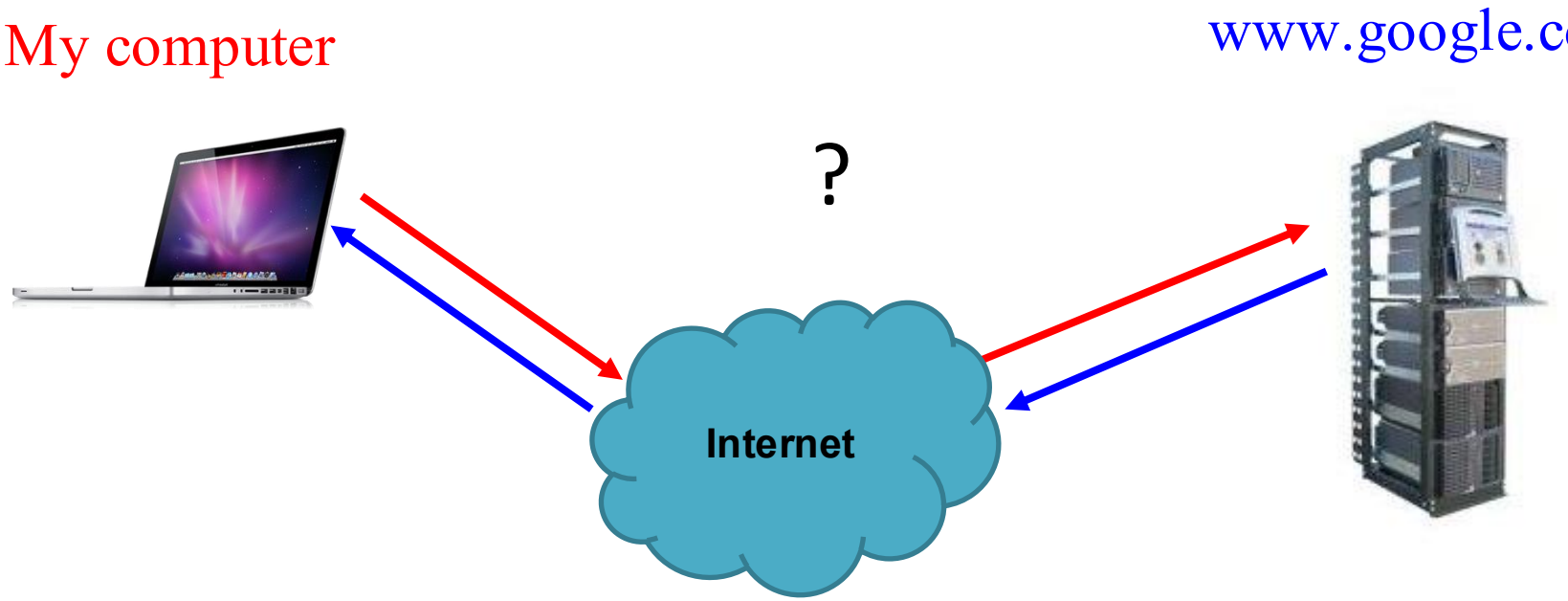


Internet Protocol Suite ("Hourglass model")



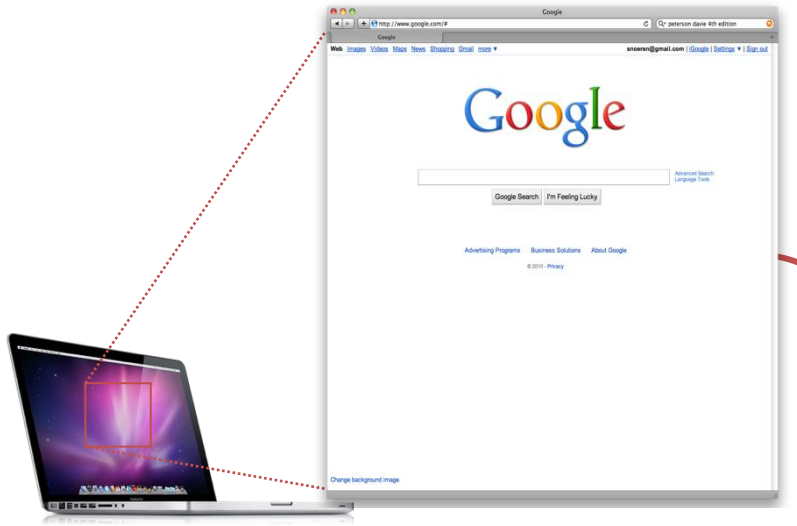
Putting this all together

- **ROUGHLY**, what happens when I click on a Web page from Swarthmore?



Application Layer: Web request (HTTP)

- Turn click into HTTP request



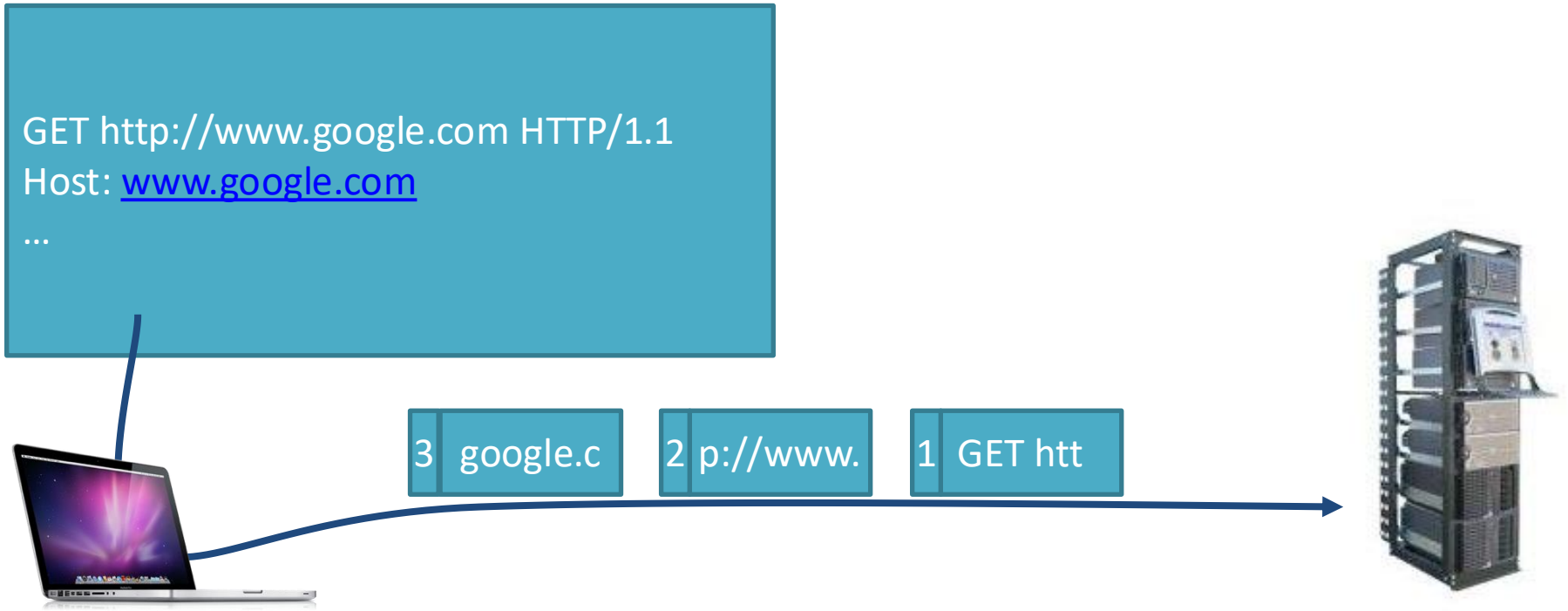
GET http://www.google.com/ HTTP/1.1
Host: www.google.com
…

Application Layer: Name resolution (DNS)

- Where is www.google.com?

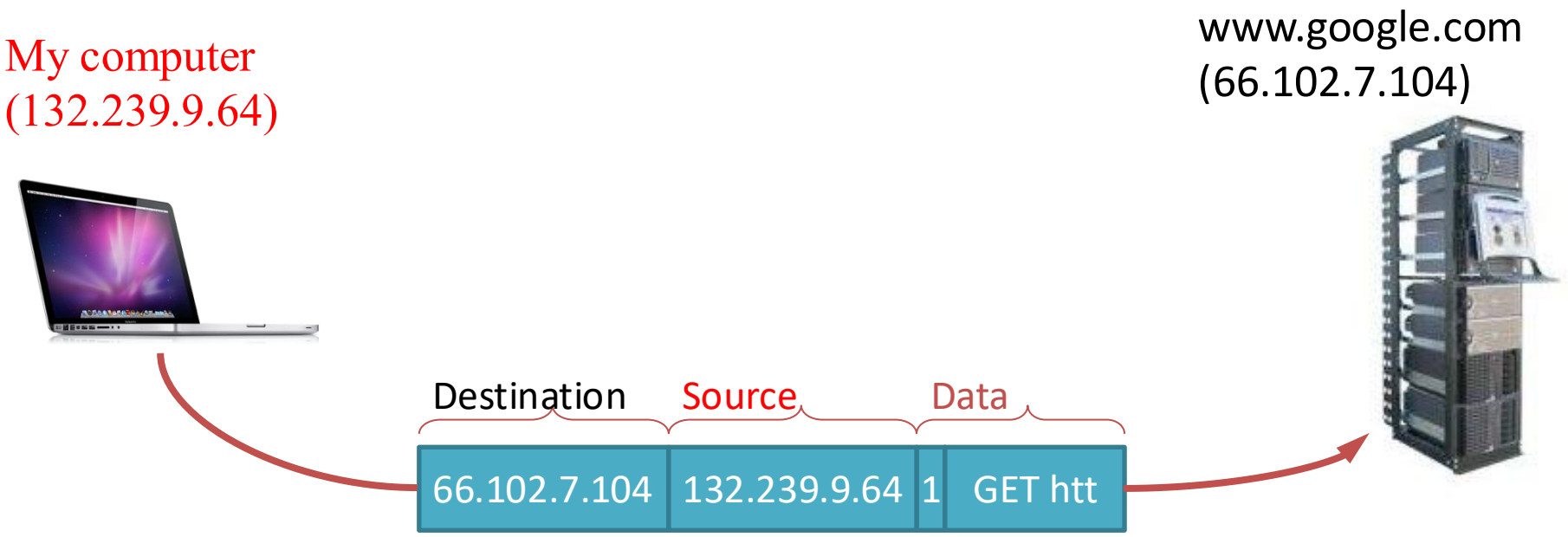
Transport Layer: TCP

- Break message into packets (TCP segments)
- Should be delivered reliably & in-order



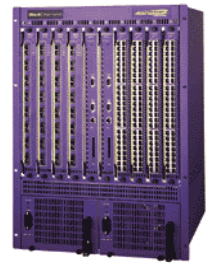
Network Layer: Global Network Addressing

- Address each packet so it can traverse network and arrive at host



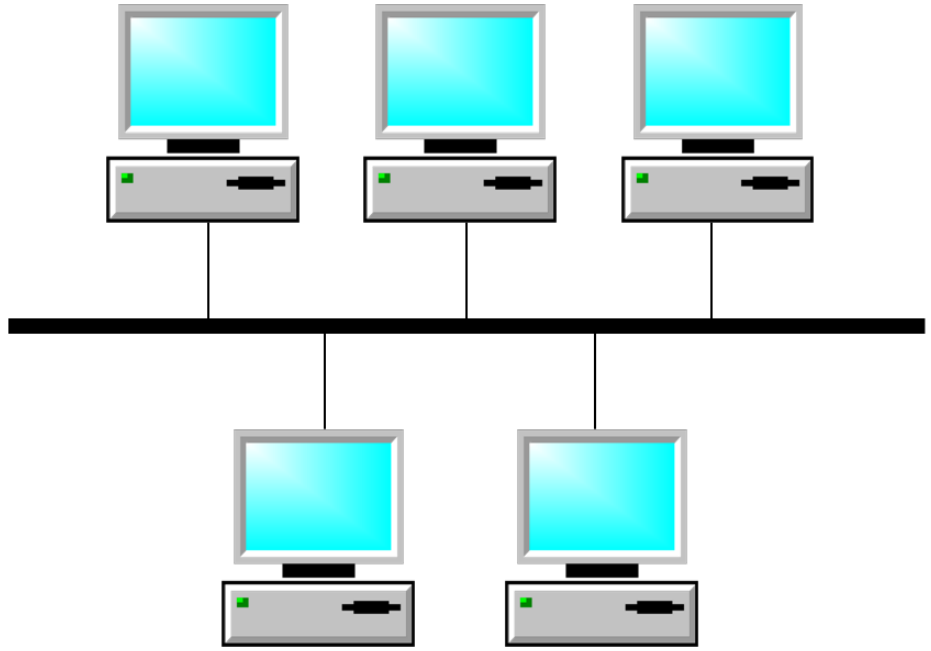
Network Layer: (IP) At Each Router

- Where do I send this to get it closer to Google?
- Which is the best route to take?

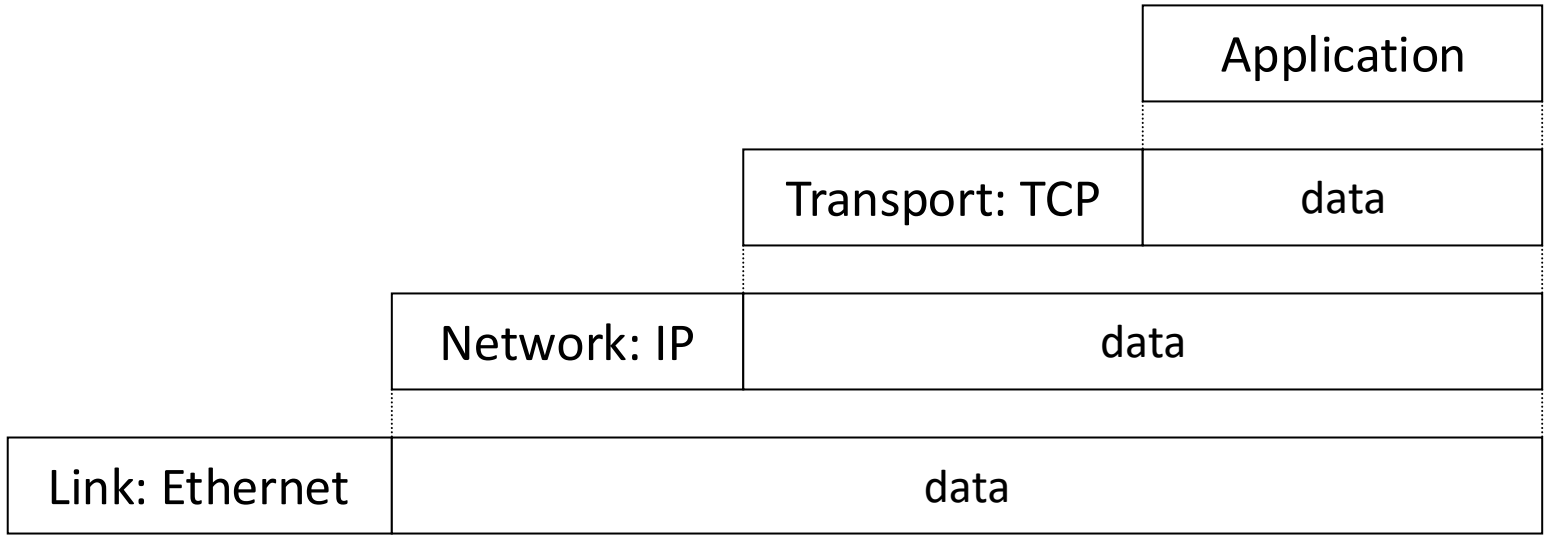


Link & Physical Layers (Ethernet)

- Forward to the next node!
- Share the physical medium.
- Detect errors.

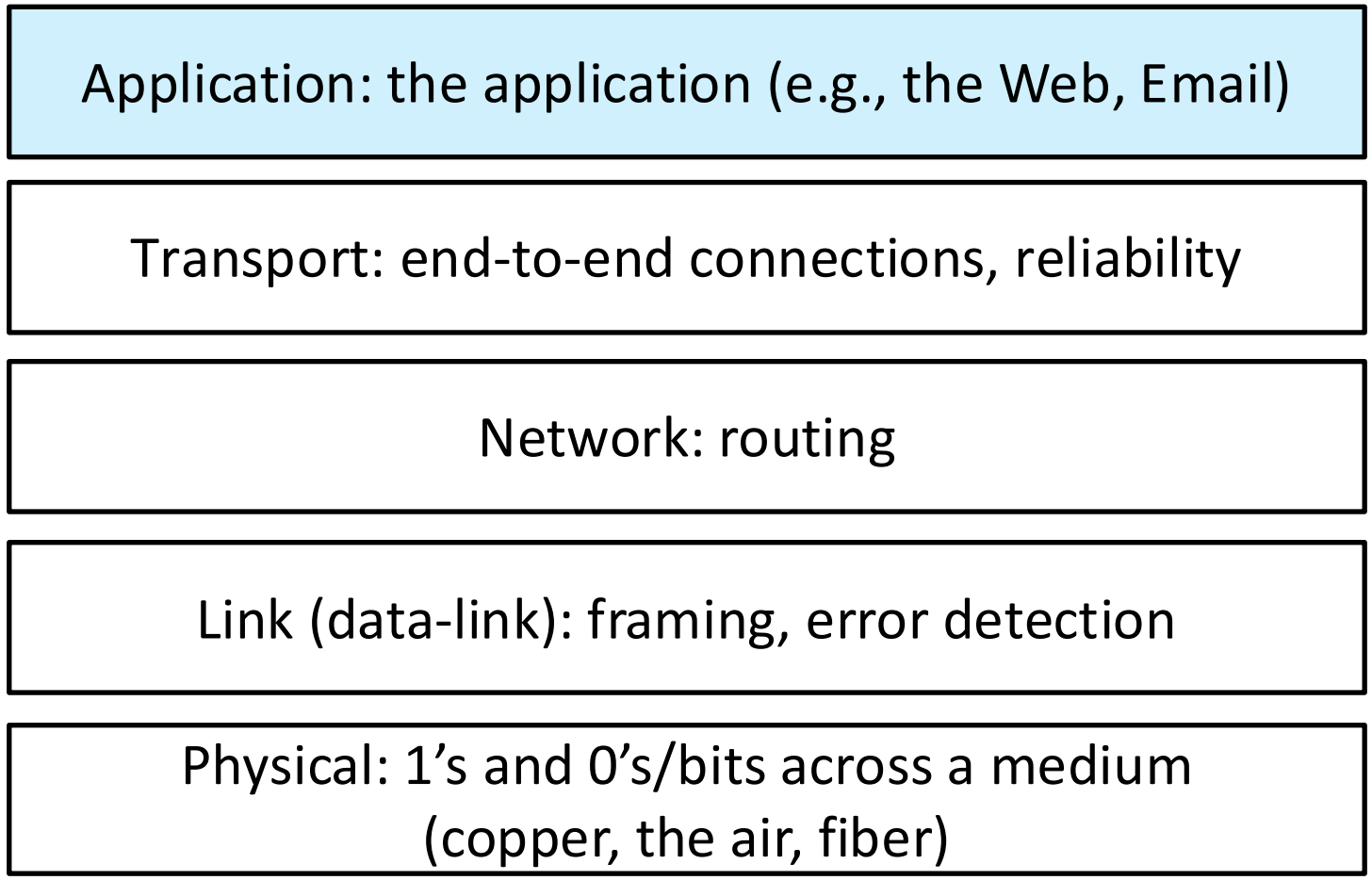


Message Encapsulation



- Higher layer within lower layer
- Each layer has different concerns, provides abstract services to those above

Five-Layer Internet Model



Which layers should routers participate in? (Getting data from host to host.) Why?

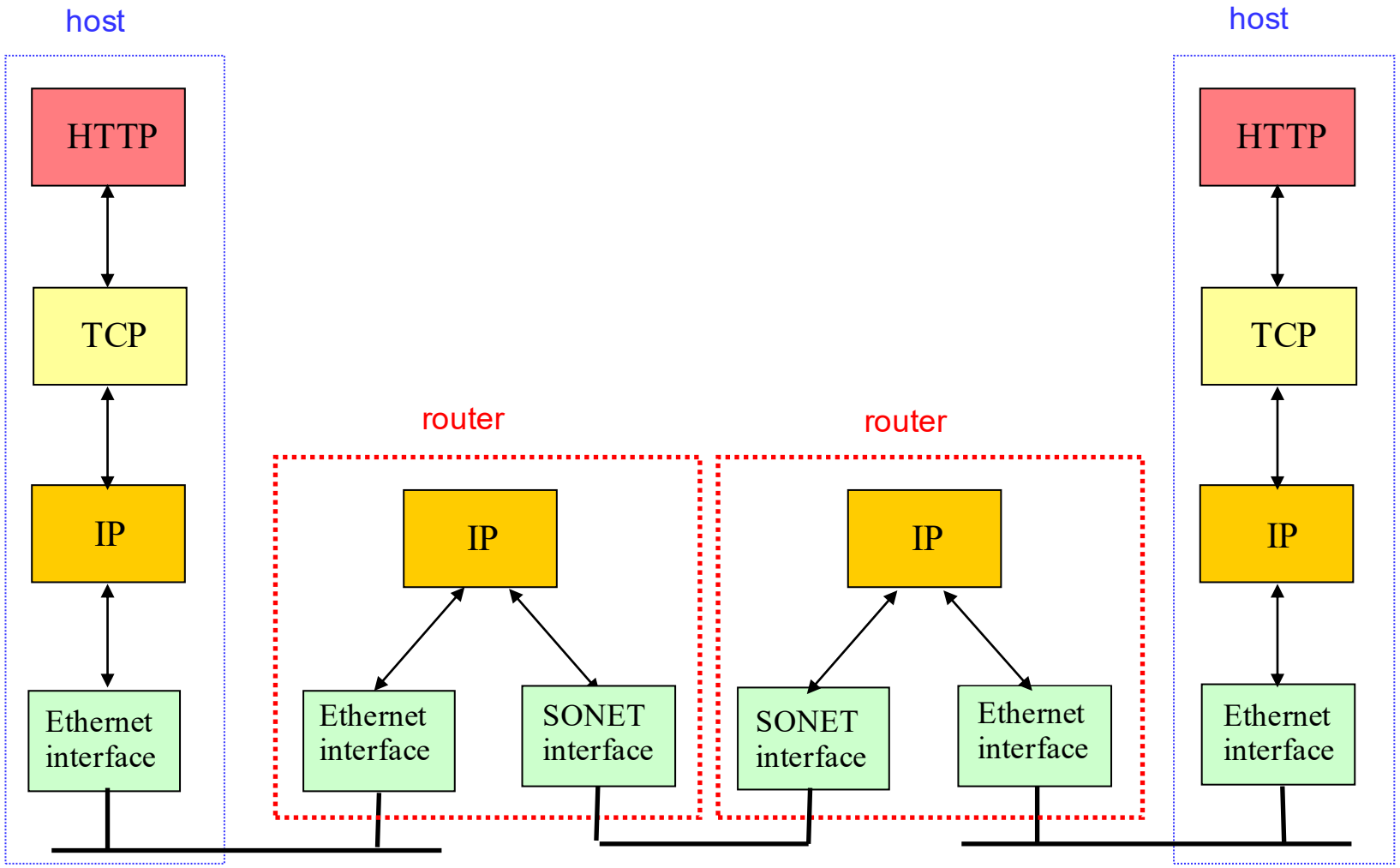
A. All of Them

B. Transport through Physical

C. Network, Link and Physical

D. Link and Physical

TCP/IP Protocol Stack



TCP/IP Protocol Stack