CS 43: Computer Networks
Layers all the way down…

Kevin Webb

Swarthmore College

January 27, 2022

Reminder: TODO List

- See EdSTEM: how to choose lab 1 partners (DUE TODAY)
- Complete Lab 0! (DUE TODAY)
- Register your clicker!
- Please let me know about:
  - Your preferred name/pronouns, if different than roster information
  - Academic accommodations

We only need…

- Manage complexity and scale up
  - Decomposing the tasks
  - Standardizing semantics to support interoperability
- Naming and addressing
  - Agreeing on how to describe a host, application, network, etc.
- Moving data to the destination
  - Forwarding messages across multiple physical components
  - Deciding how to get from here to there
- Reliability and fault tolerance
- Resource allocation
  - Figuring out how to share finite bandwidth, memory, etc.

Discussion question

- Green border

- Recall the sequence
  - Answer individually
  - Discuss in your group
  - Answer as a group
  - Class-wide discussion

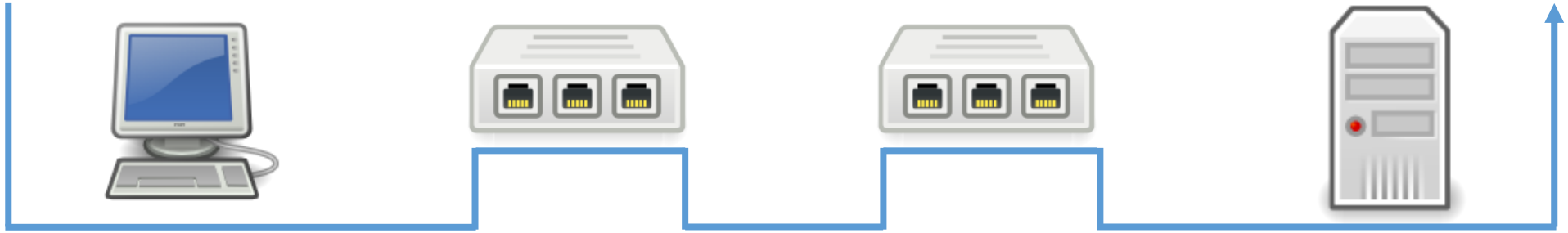
Networks have many concerns, such as reliability, error checking, naming and data ordering. Who/what should be responsible for addressing them? (Why? Which ones belong in which location?)

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



- 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.*
- Examples: error control, quality of service
- Reference: Saltzer, Reed, Clark, “End-To-End Arguments in System Design,” ACM Transactions on Computer Systems, Vol. 2 (4), 1984.

What is a Protocol?

- Goal: get message from sender to receiver, with no ambiguity
- Protocol: message format + transfer procedure

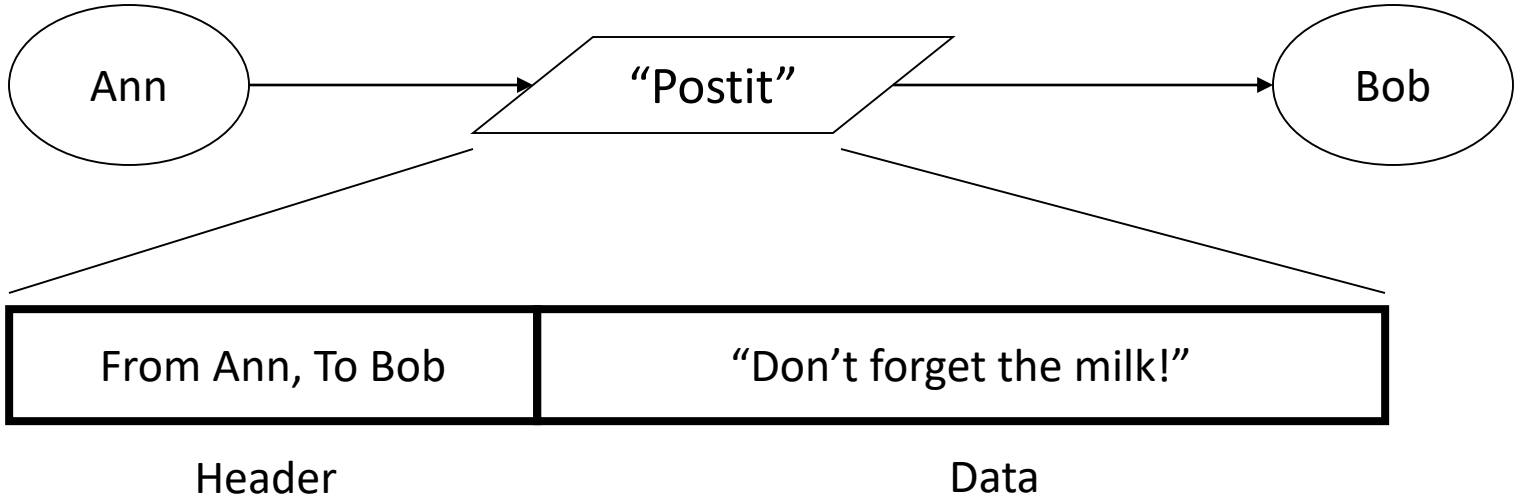
- Multiparty, so no central thread of control
  - sender and receiver are separate processes

- Expectations of operation
  - first you do x, then I do y, then you do z, ...
  - if you do q, I'll do p

Message Format

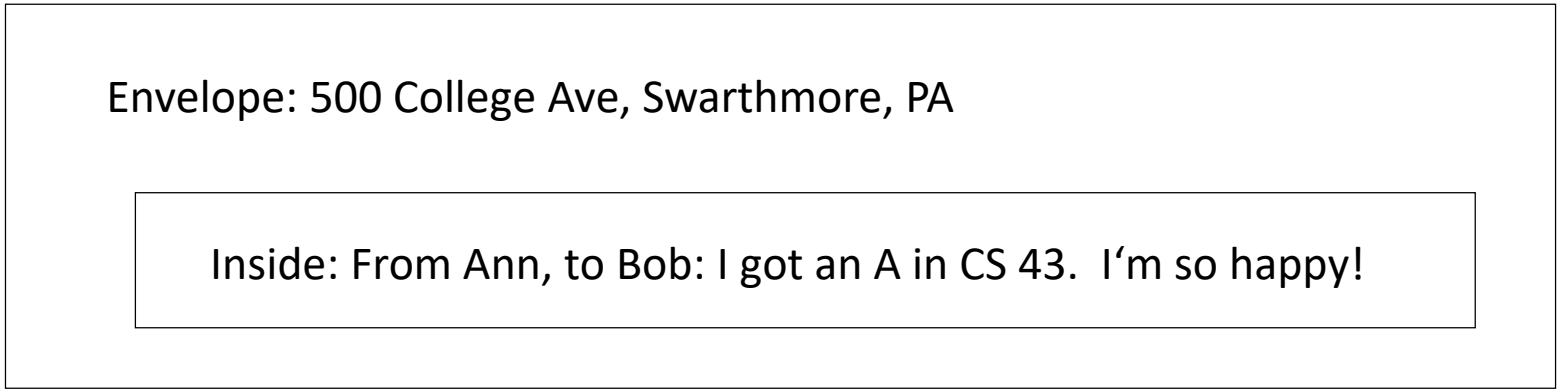
- Message: contains header and data
- Data: what sender wants to receiver to know
- Header: information to support protocol
  - Source and destination addresses
  - State of protocol operation
  - Error control (to check integrity of received data)

Example: Ann Sends Message to Bob

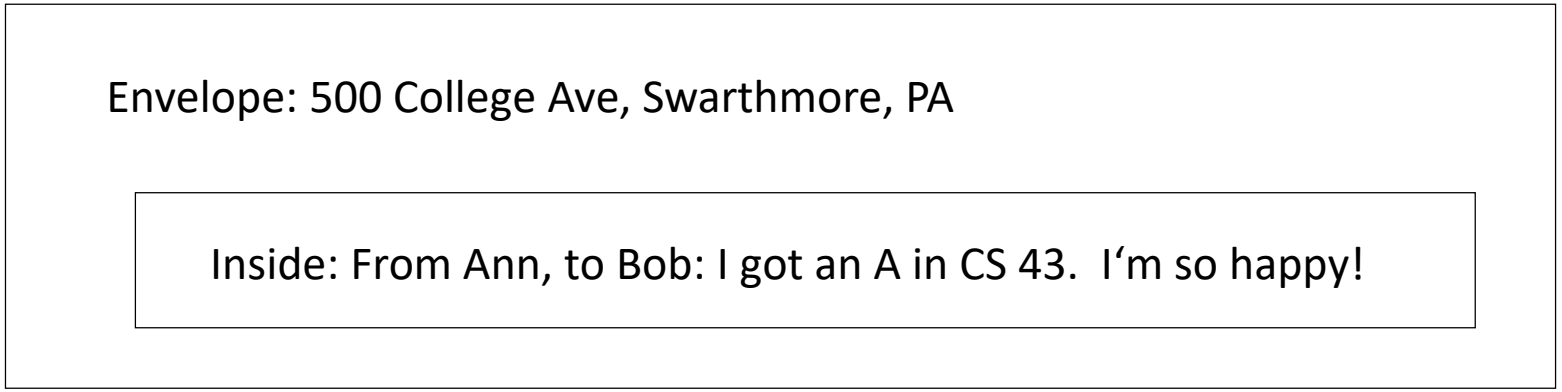


- Protocol
  - Message format: (from, to), message contents
  - Transfer procedure: post on refrigerator

Suppose Ann is mailing the same post-it to Bob via the mail:



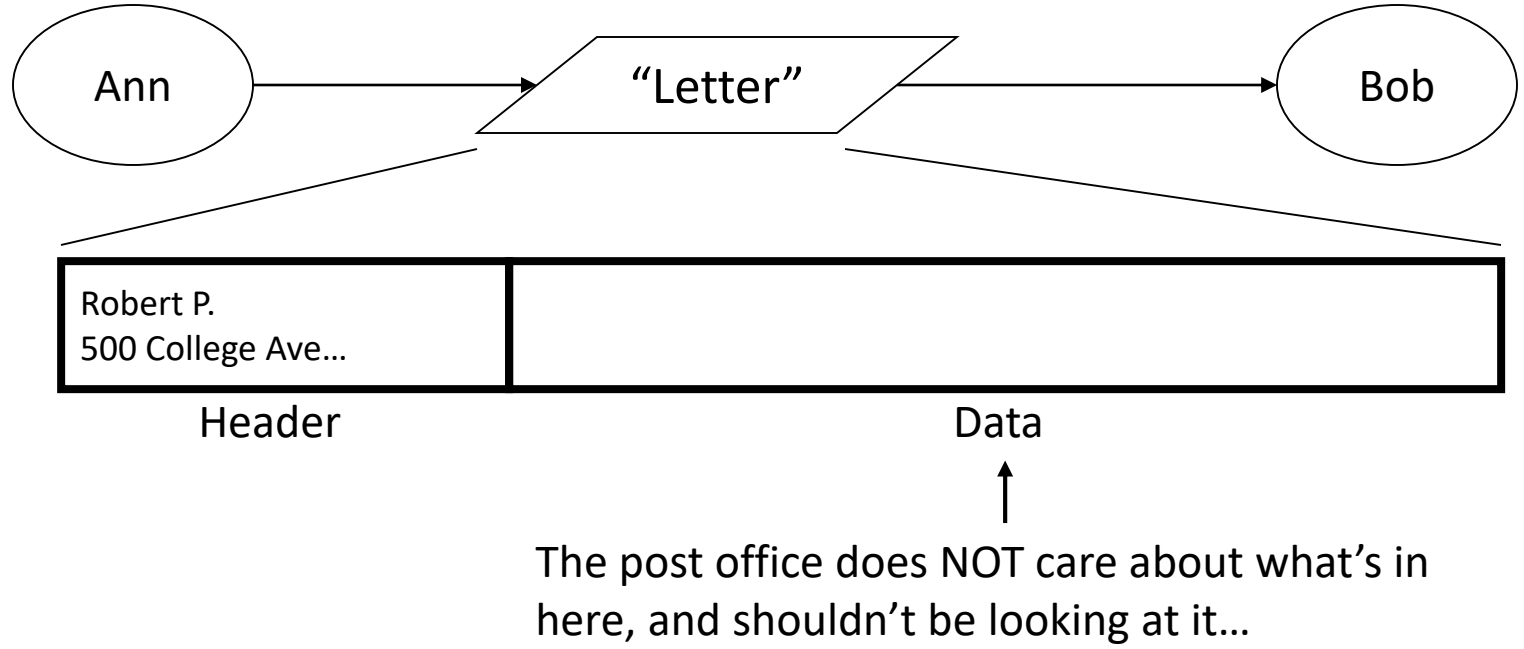
Suppose Ann is mailing the same post-it to Bob via the mail:



Where is the header now?

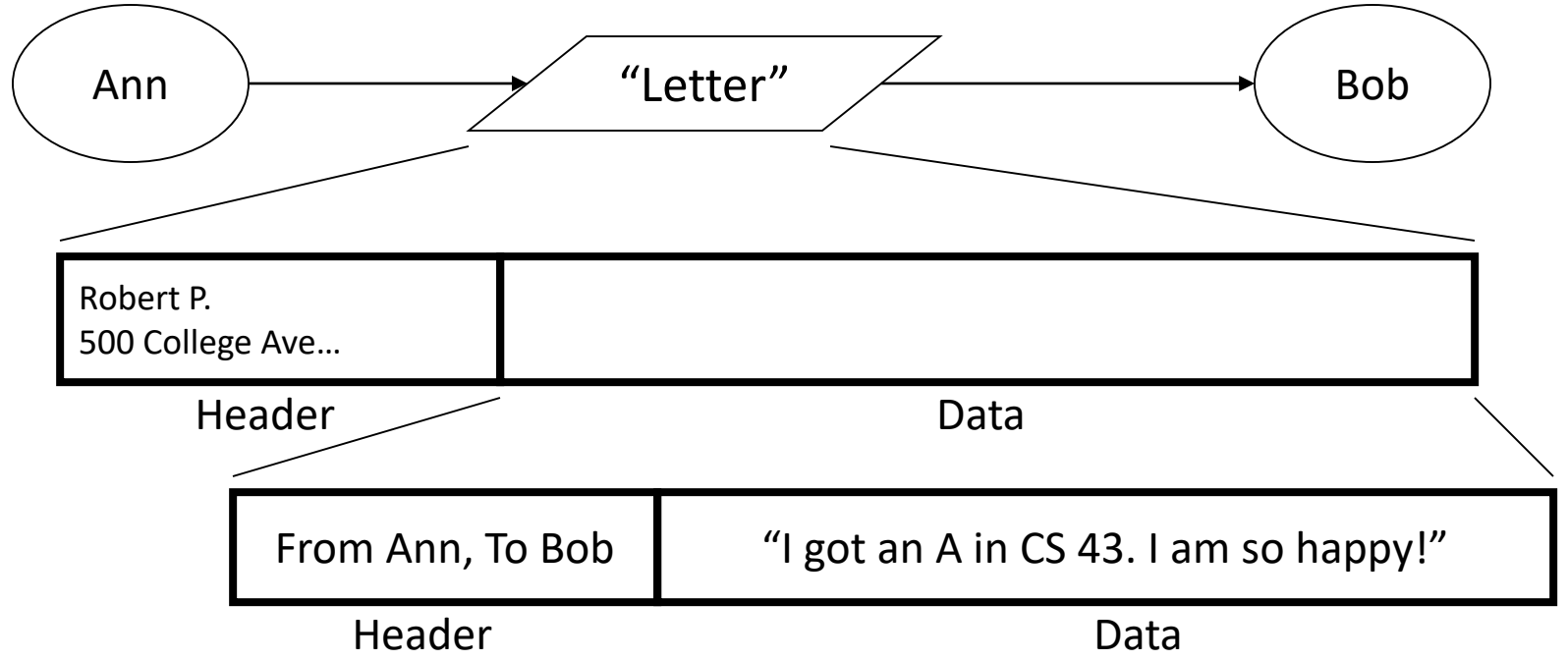
A. The address on the envelope.
B. The “from Ann to Bob”.
C. Somewhere else.

Message Encapsulation



- Protocol
  - Message format: (from, to), message contents
  - Transfer procedure: send in mail, post on refrigerator

Message Encapsulation



- Protocol
  - Message format: (from, to), message contents
  - Transfer procedure: send in mail, post on refrigerator

Layering: Separation of Functions

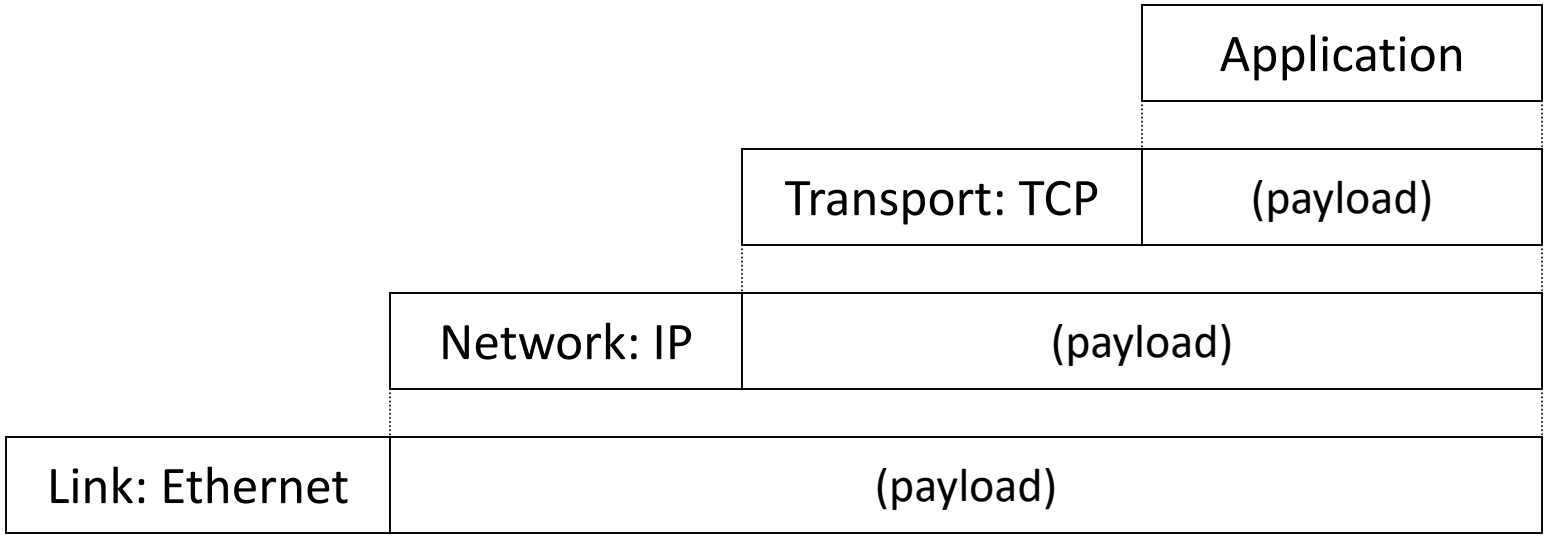
Letter: written/sent by Ann, received/read by Bob
Postal System: Mail delivery of letter in envelope

- Ann 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”: Ann, Bob, letter

Abstraction!

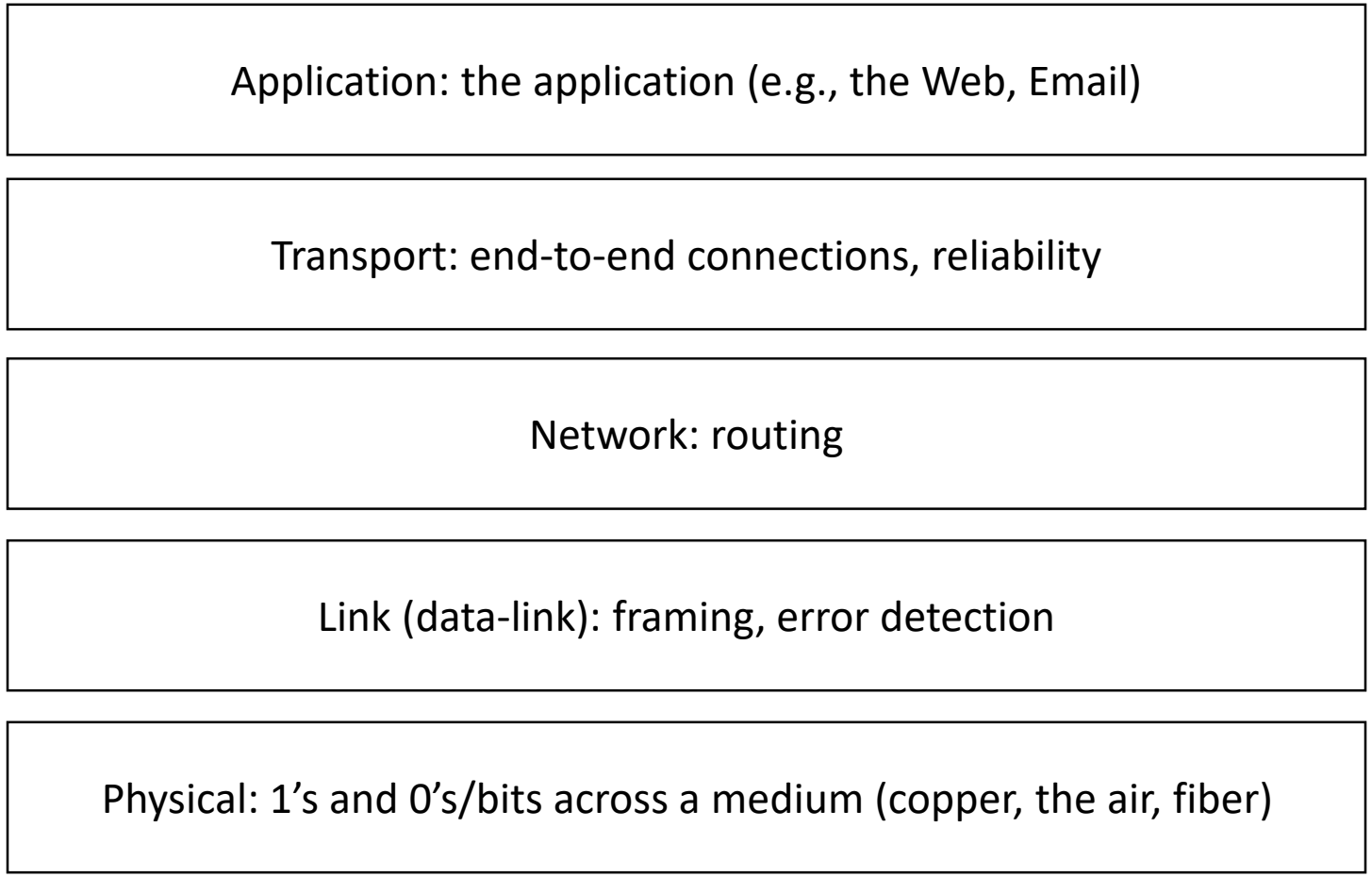
- Hides the complex details of a process behind a simpler interface
- Use abstract representation of relevant properties make reasoning simpler
- Ex: Alice and Bob knowledge of postal system:
  - Letters with addresses go in, come out other side

Encapsulation



- Higher level *n* within lower level *n*-1
- Each level has different concerns, provides abstract services to those above

Five-Layer Internet Model



Analogy

General (Application)

General (Application)

- Wants important message to be delivered.
- Wants the message to be reliable!
- Wants it **NOW**!

- Is he going to deliver it himself?

Analogy

General (Application)

Colonel

General (Application)

Analogy

General (Application)

Colonel

Captain

General (Application)

Analogy

General (Application)

Colonel

Captain

Lieutenant

General (Application)

Analogy

General (Application)

Colonel

Captain

Lieutenant

Private

General (Application)

Analogy

General (Application)

Colonel

Captain

Lieutenant

Private

General (Application)

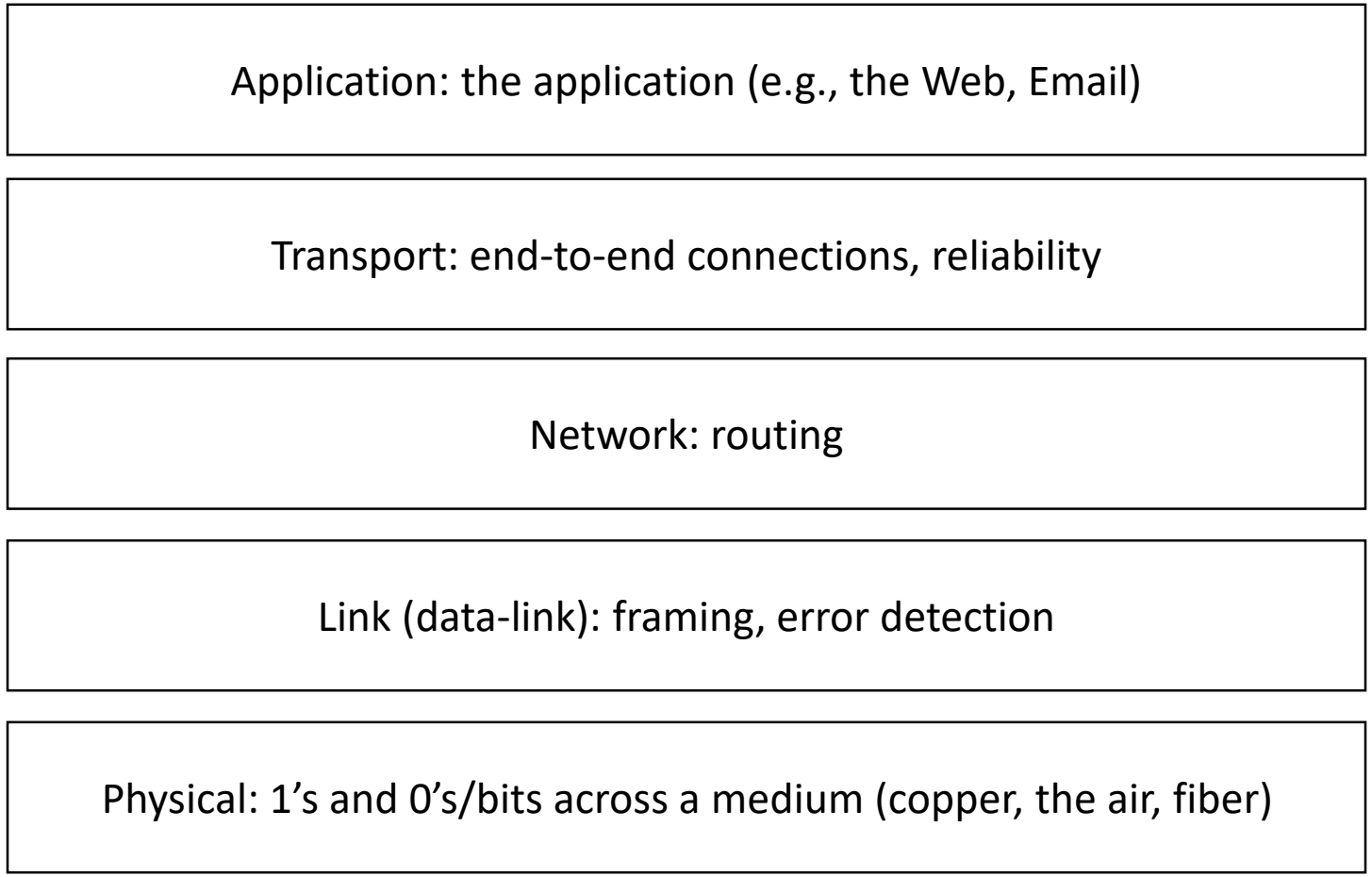
Colonel

Captain

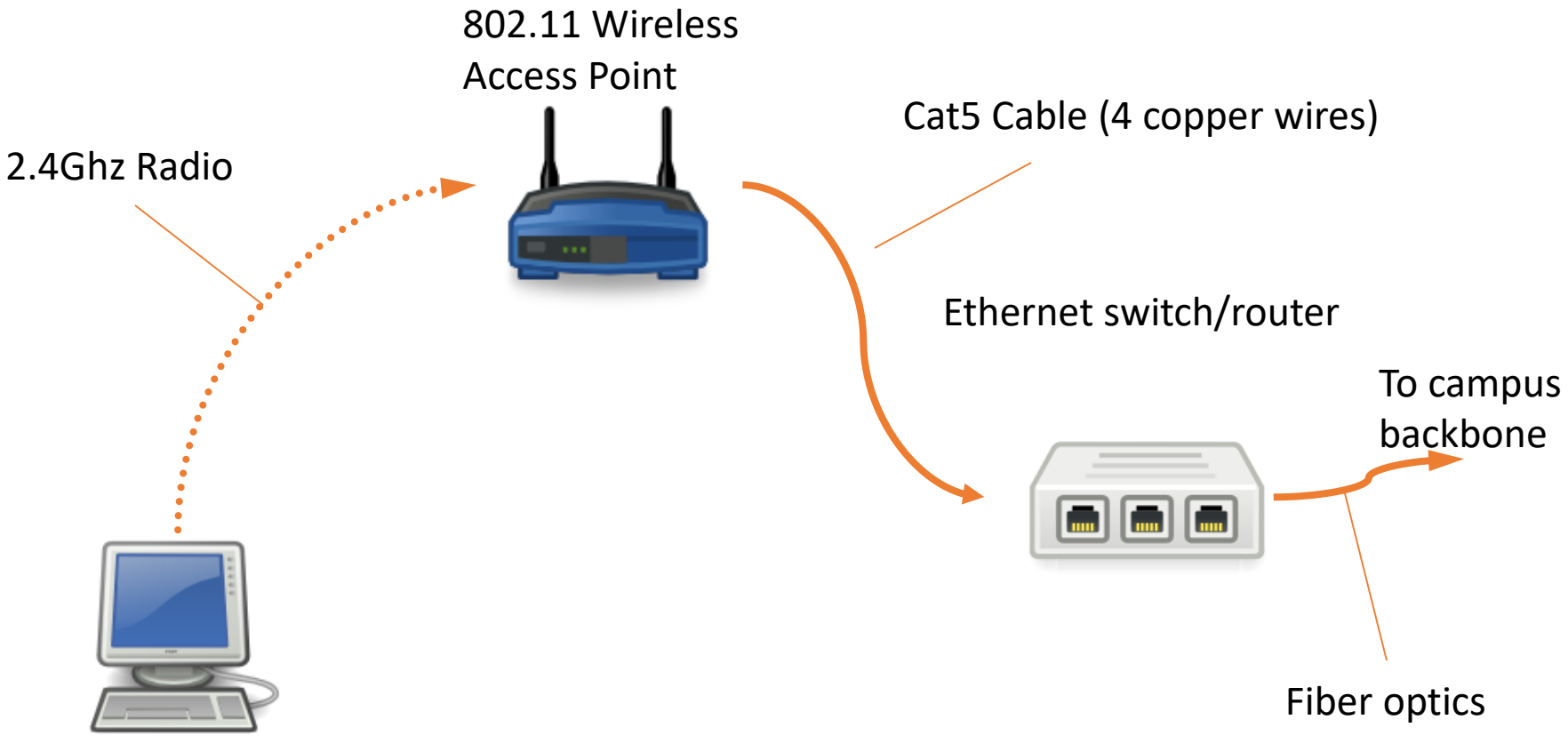
Lieutenant

Private

Five-Layer Internet Model

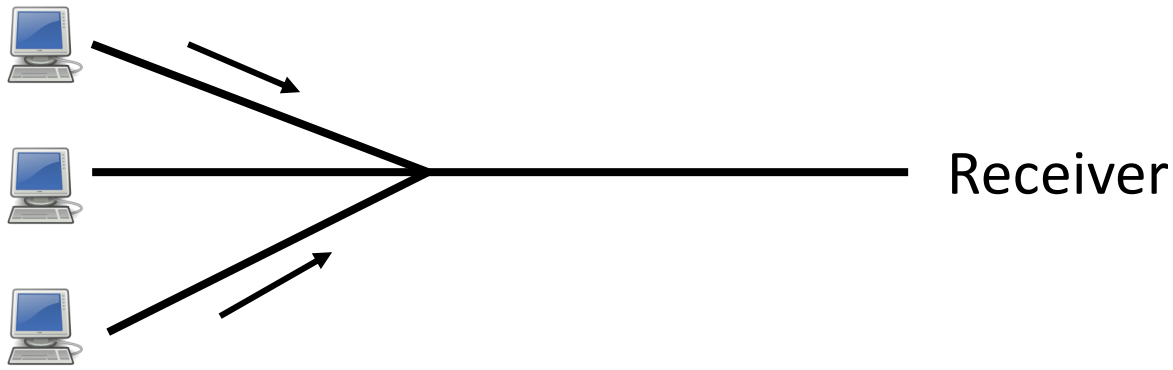


Physical layer
(Copper, Coax, Air, Fiber Optics)



(Data) Link Layer (Ethernet, WiFi, DOCSIS)

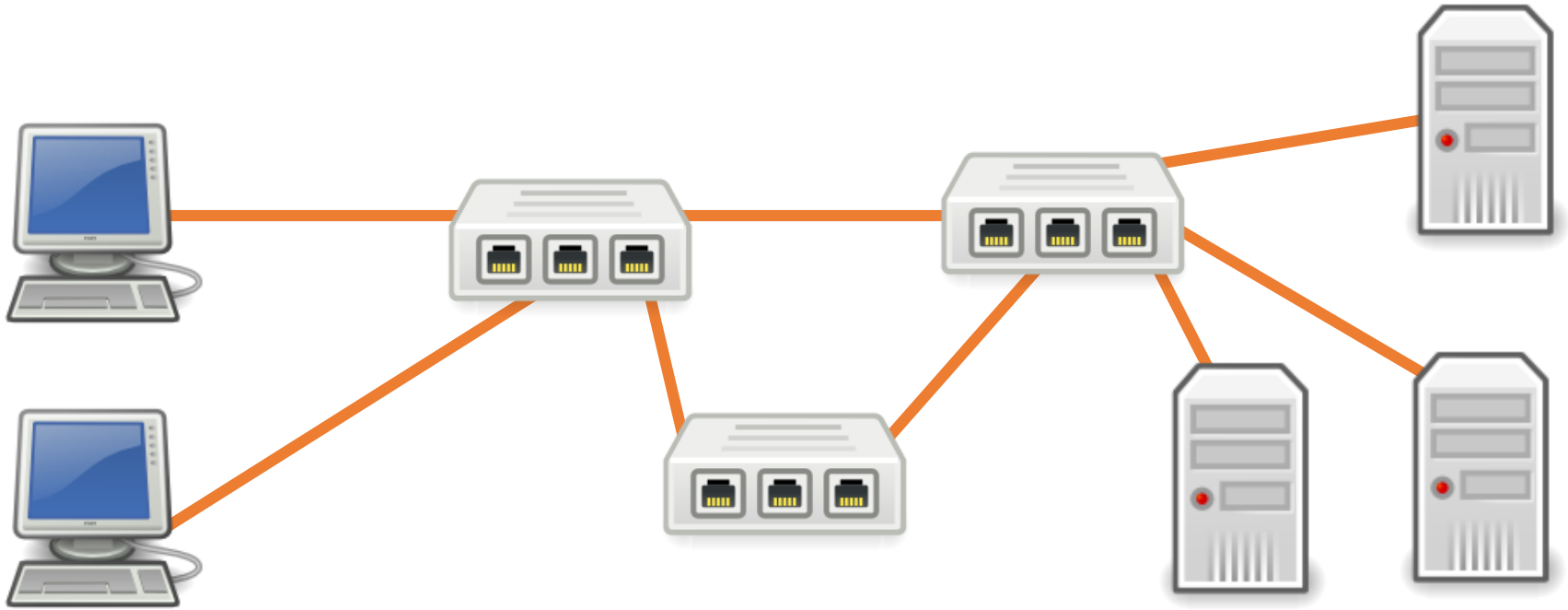
- Break message into chunks (frames) to send over physical medium
- Media access: can it send the frame now?



- Send frame, handle “collisions”

Network Layer (IP)

- **Routers**: choose paths through network



You’re asked to design the Internet. Which do you choose for routing a conversation (“flow”) over the network?

You’re asked to design the Internet. Which do you choose for routing a conversation (“flow”) over the network?

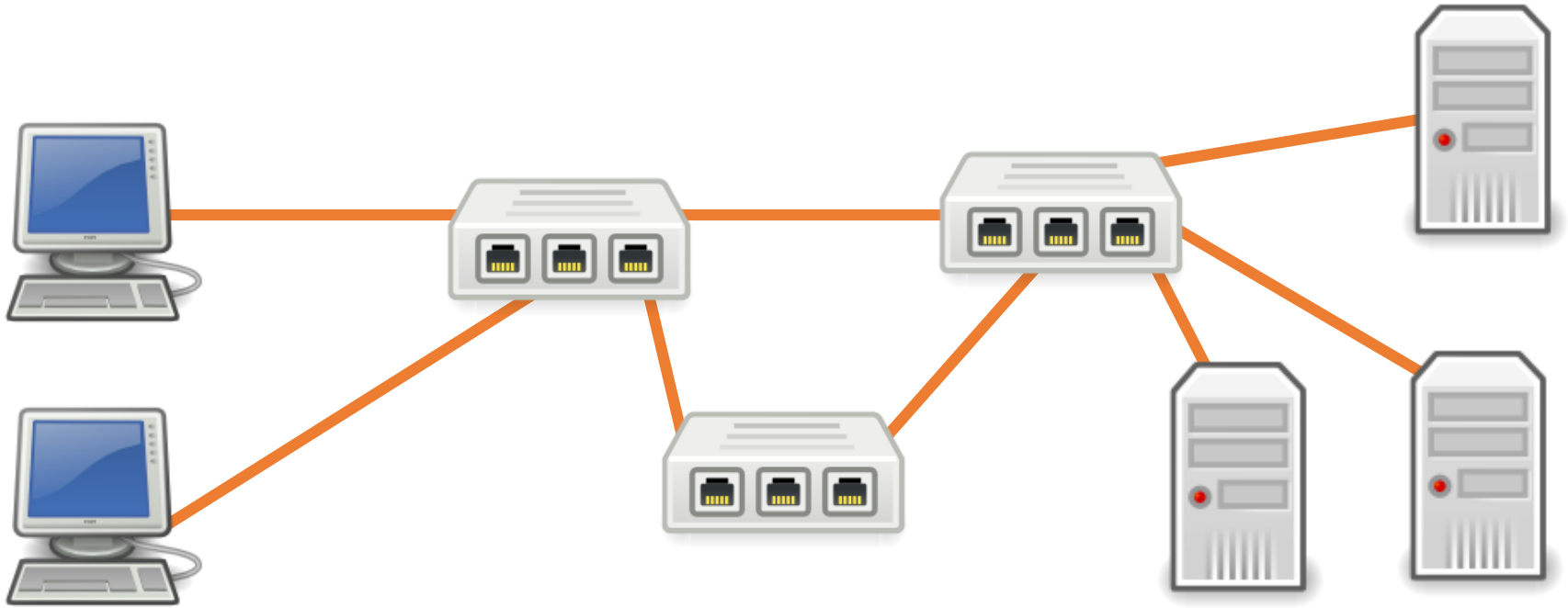
A. I would choose the path for the flow at the beginning and use it for all the flow’s messages.

B. I would reevaluate the path choice for each of the flow’s messages.

C. I would do something else.

Network Layer (IP)

- **Routers**: chooses paths through network
  - *Circuit switching*: guaranteed channel for a session (Telephone system)
  - *Packet switching*: statistical multiplexing of independent pieces of data (Internet)



Transport Layer (TCP, UDP)

- Provides
  - Ordering
  - Error checking
  - Delivery guarantee
  - Congestion control
  - Flow control

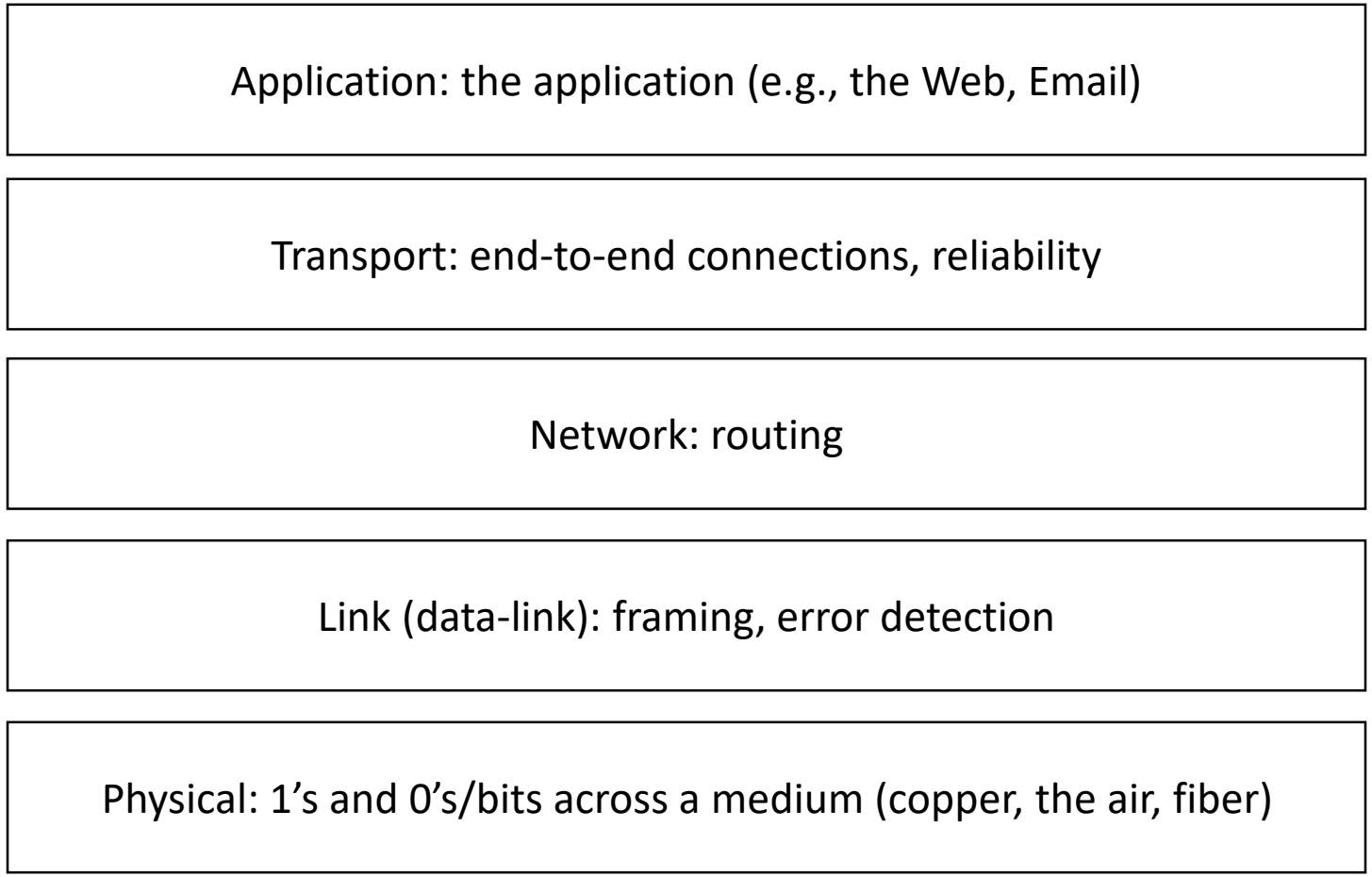
- Or doesn’t!

Application Layer (HTTP, FTP, SMTP, Zoom)

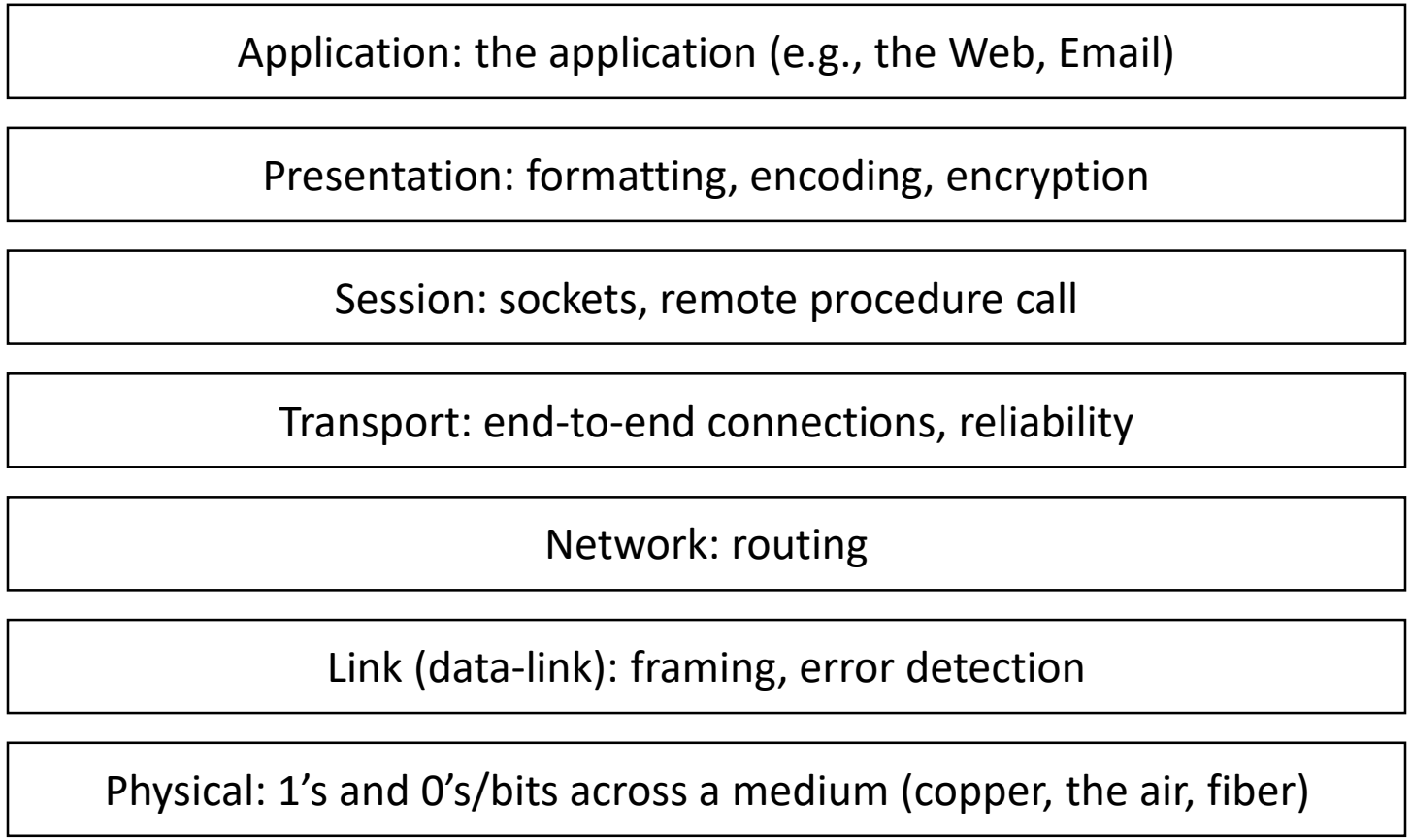
- Does whatever an application does!




Five-Layer Internet Model



OSI Seven-Layer 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?)

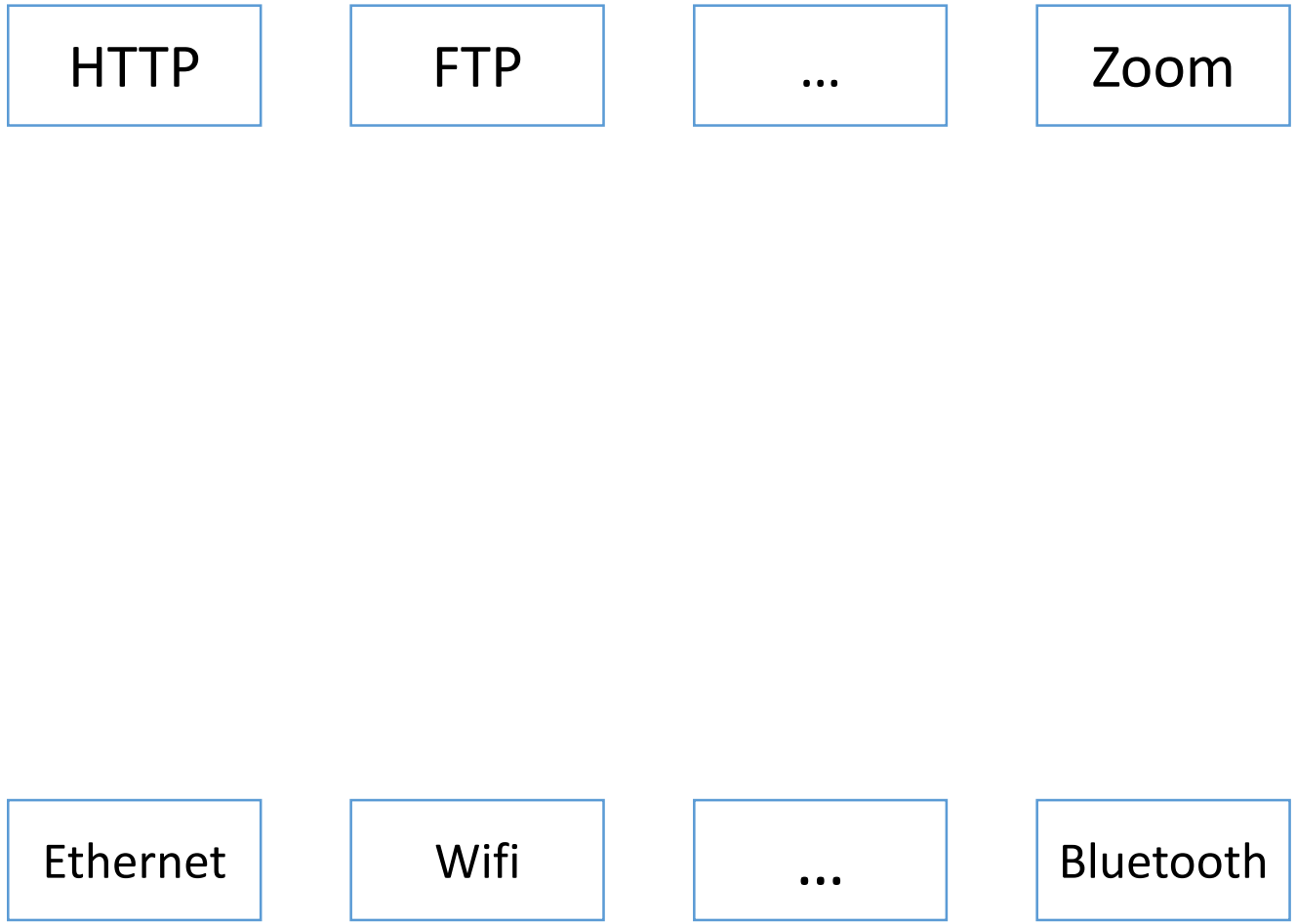
A. Always

B. Usually

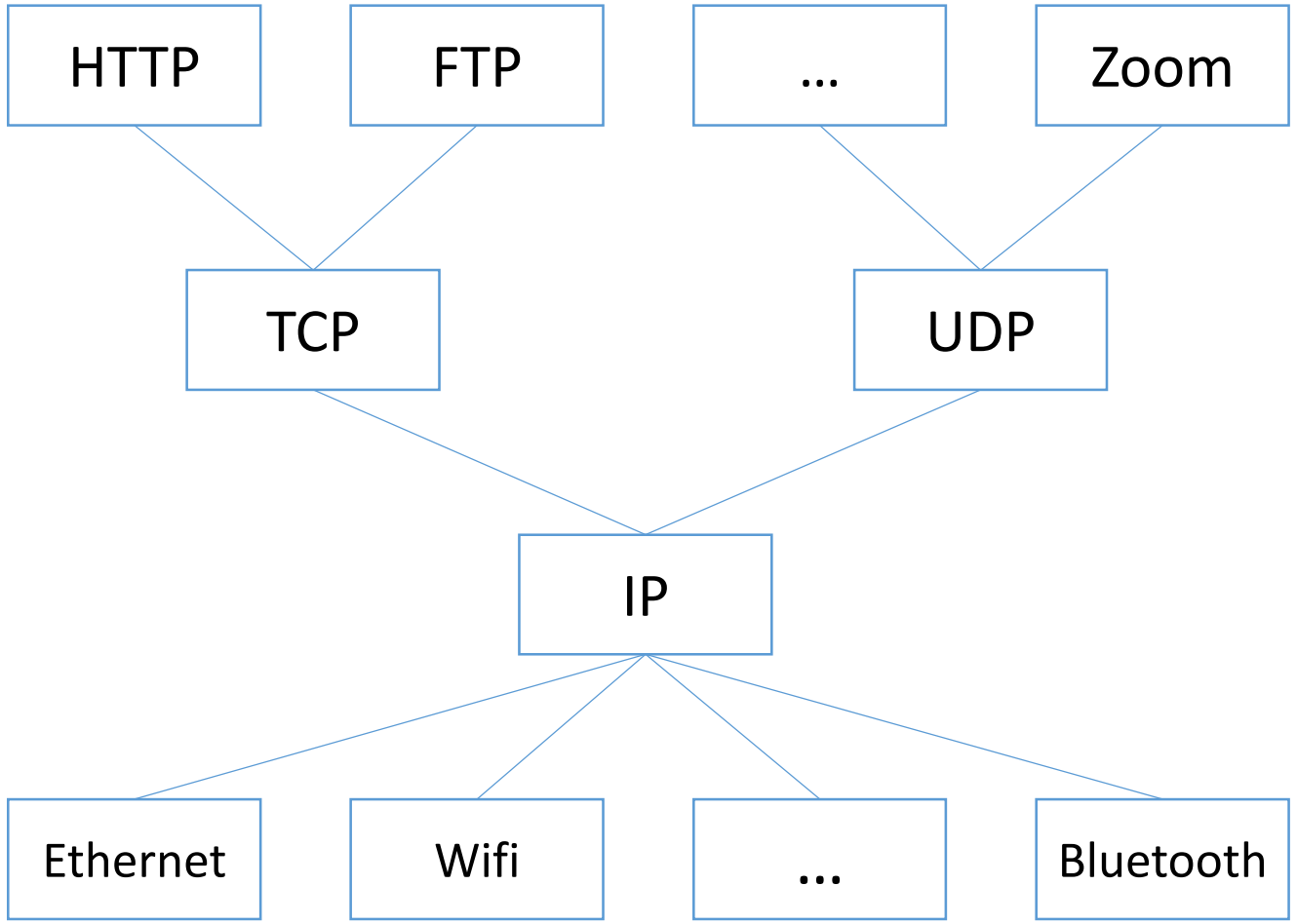
C. Sometimes

D. Never

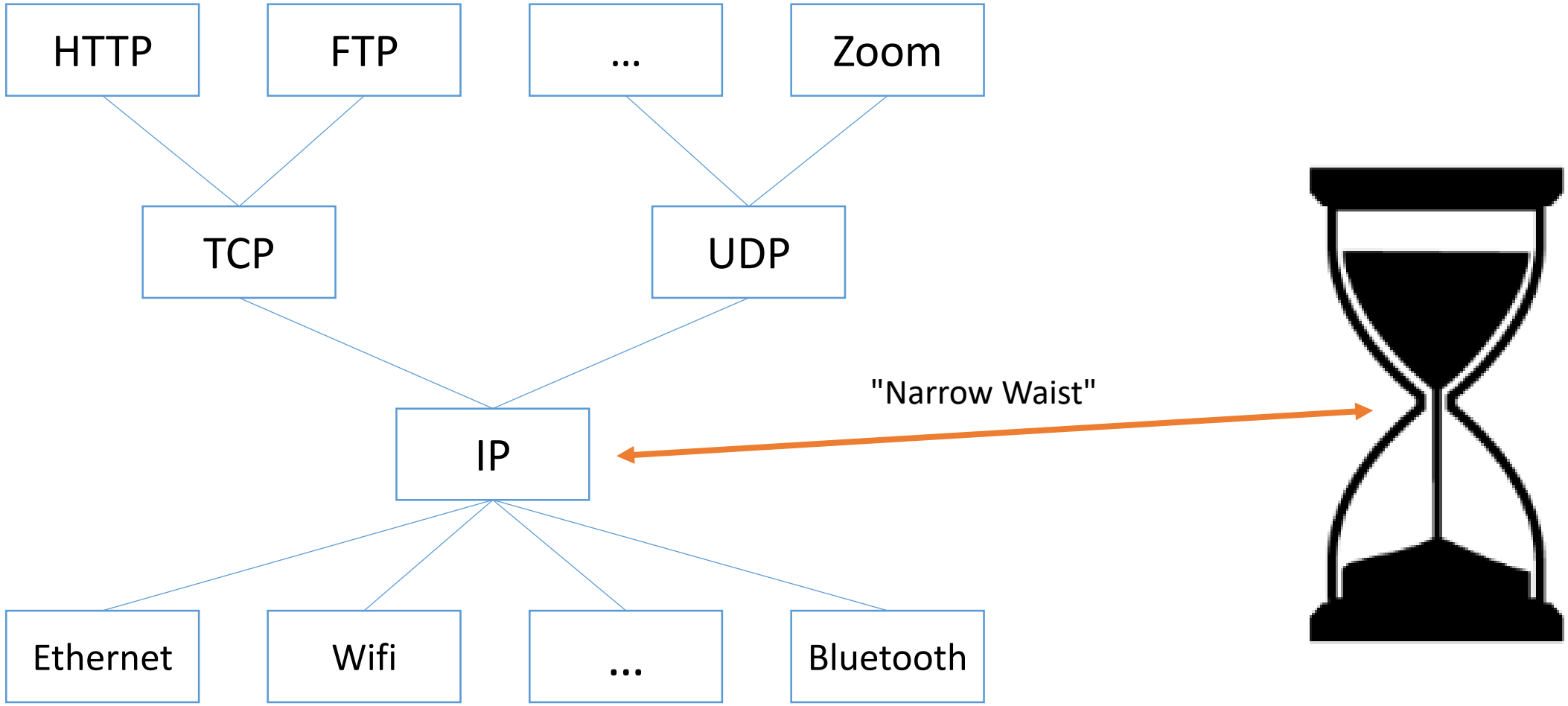
Internet Protocol Suite



Internet Protocol Suite

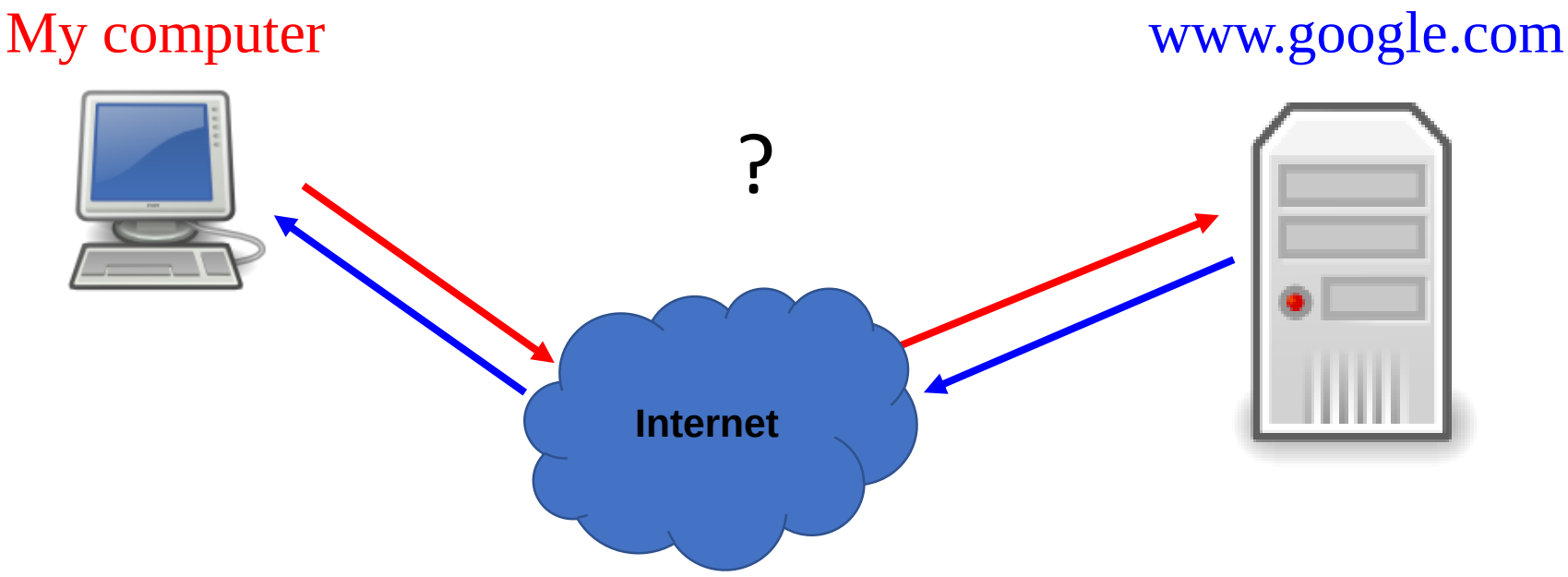


Internet Protocol Suite ("Hourglass model")



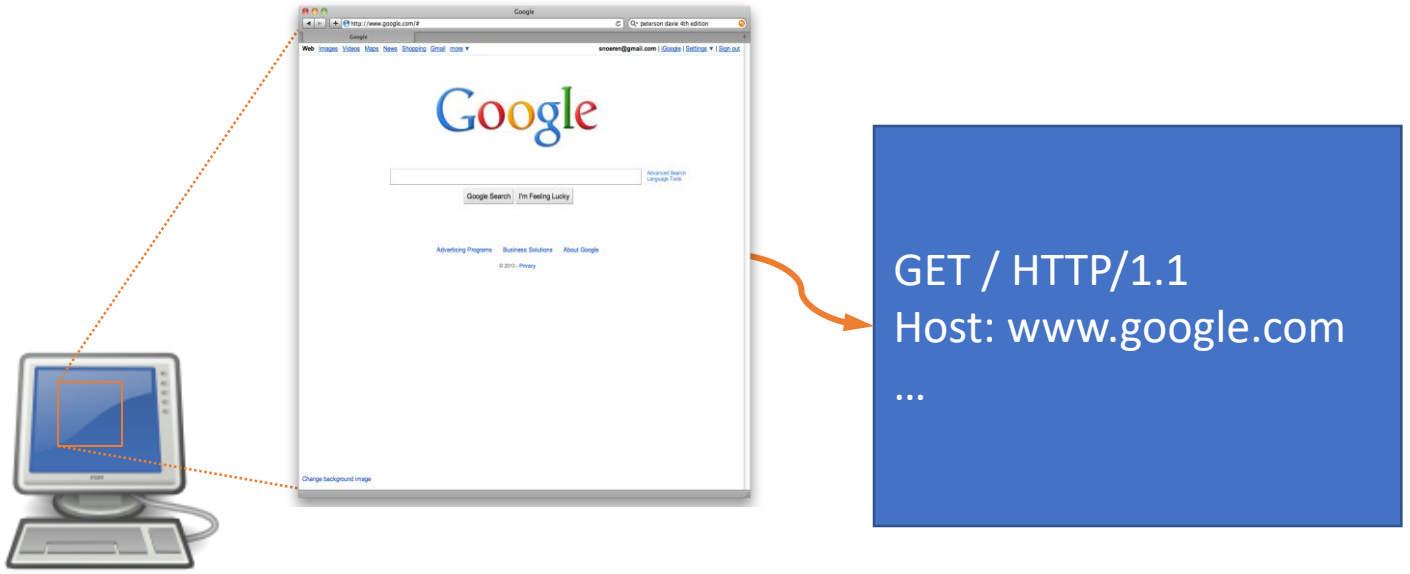
Putting this all together

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



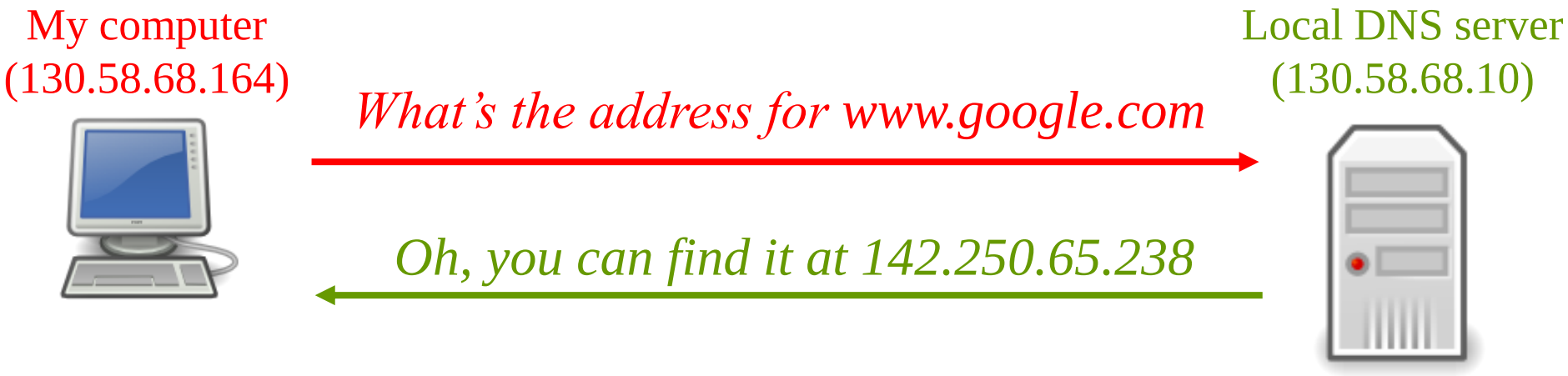
Web request (HTTP)

- Turn click into HTTP request



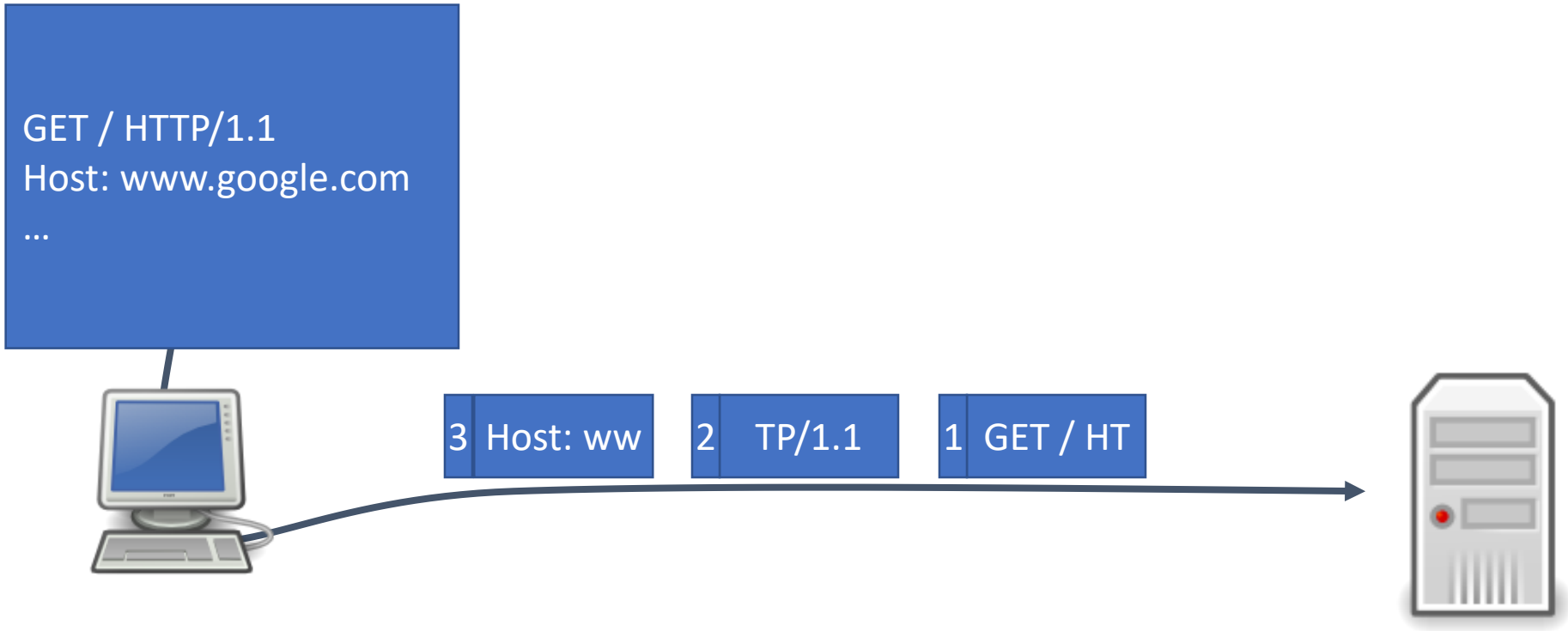
Name resolution (DNS)

- Where is www.google.com?



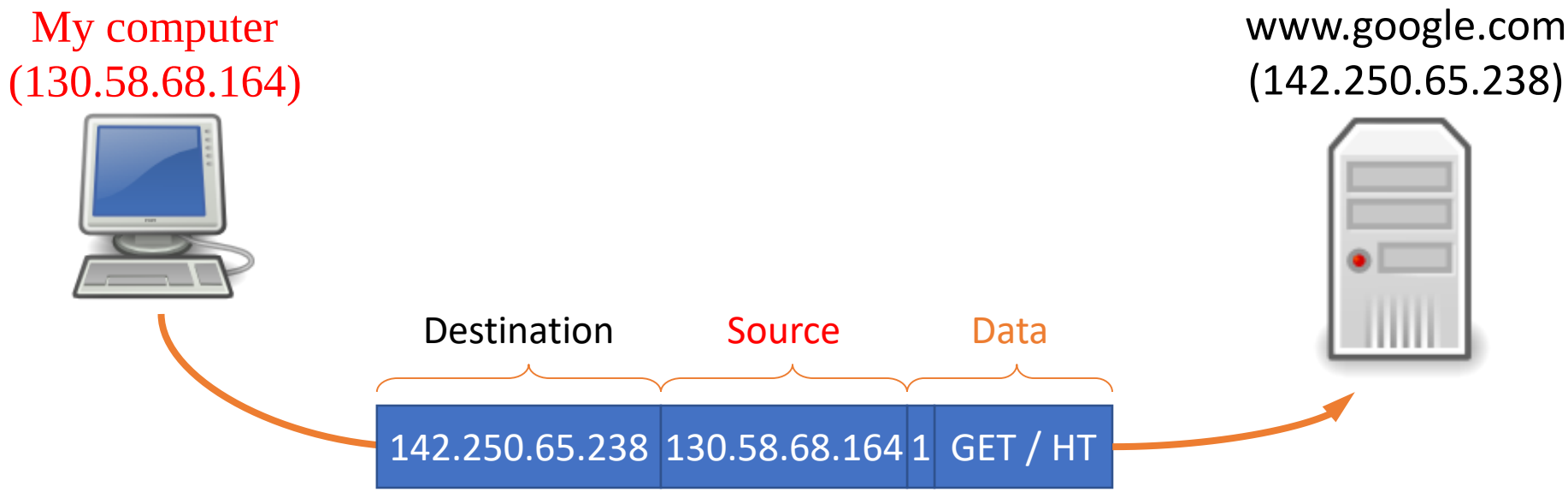
Transport (TCP)

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



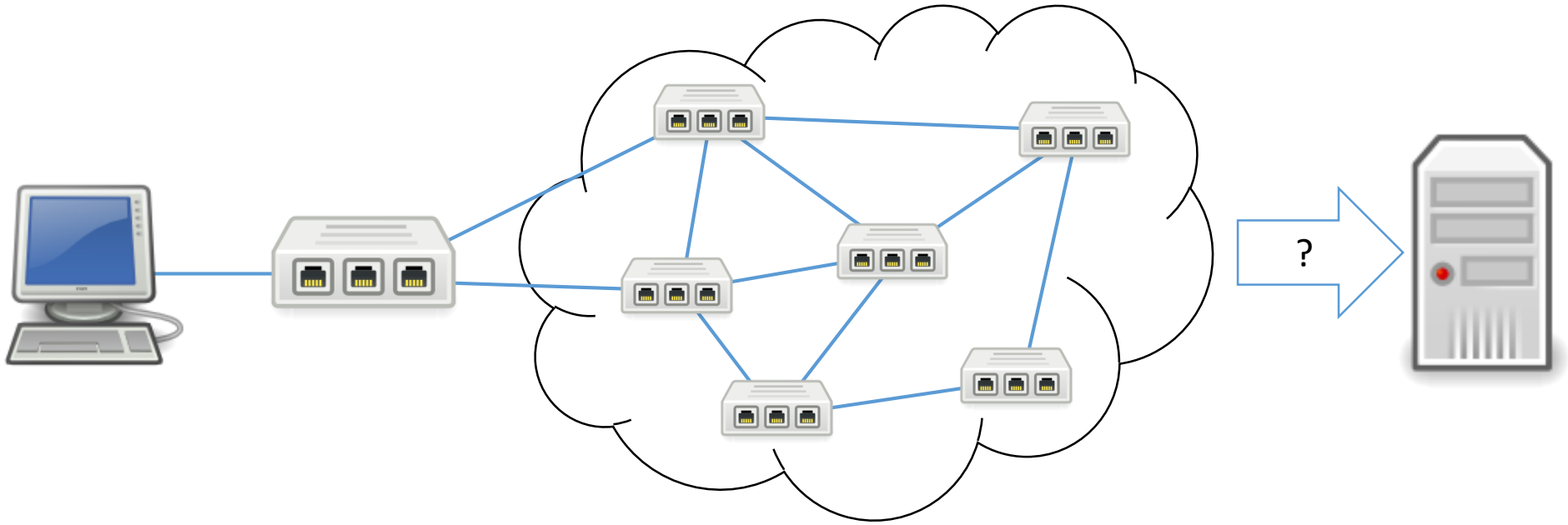
Global Network Addressing

- Add IP header, address each IP packet so it can traverse network and arrive at destination.



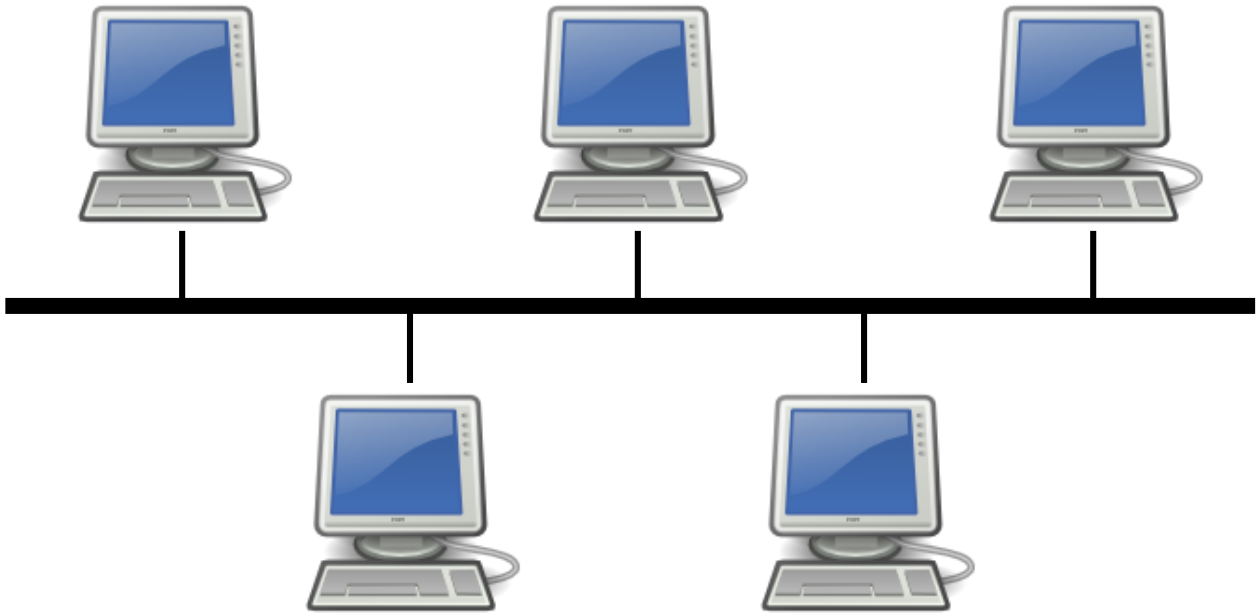
(IP) At Each Router

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

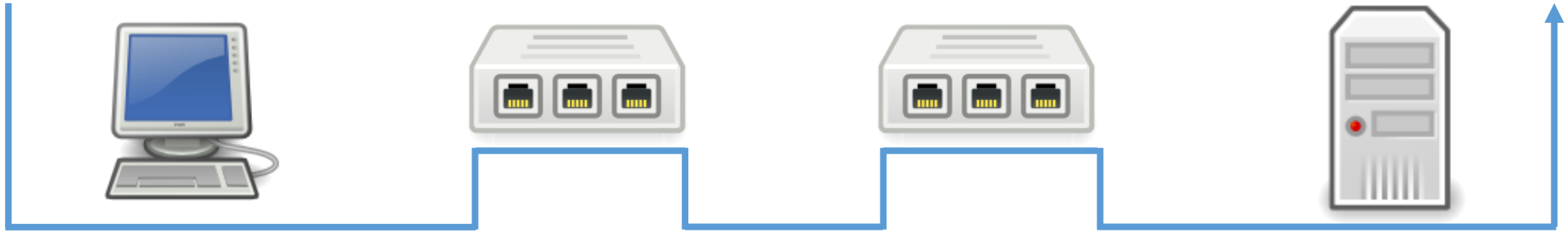


Link & Physical Layers

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



The “End-to-End” Argument



- 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.*
- Examples: error control, quality of service
- Reference: Saltzer, Reed, Clark, “End-To-End Arguments in System Design,” ACM Transactions on Computer Systems, Vol. 2 (4), 1984.

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

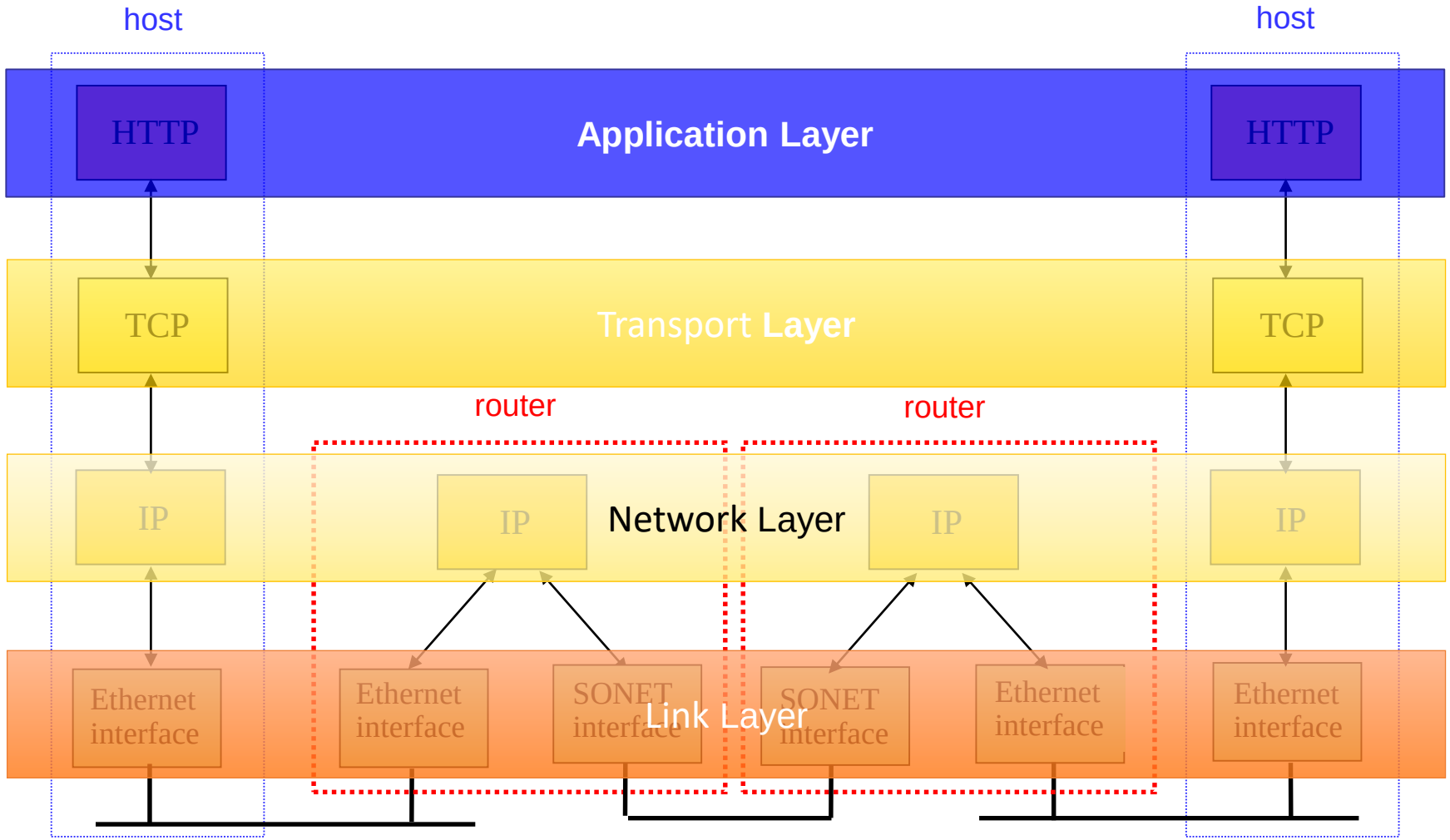
A. All of Them

B. Transport through Physical

C. Network, Link and Physical

D. Link and Physical

TCP/IP Protocol Stack



Summary

- Layers of abstraction divide up responsibility for network functionality
- End-to-end principle: do work at higher layers when possible
- Protocol governs message format and transfer procedure
- Messages encapsulated by protocol headers at each layer