CS 43: Computer Networks

03: Protocols, Layering and (some) HTTP

September 3, 2025

SWARTHMORE COLLEGE

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

Announcements

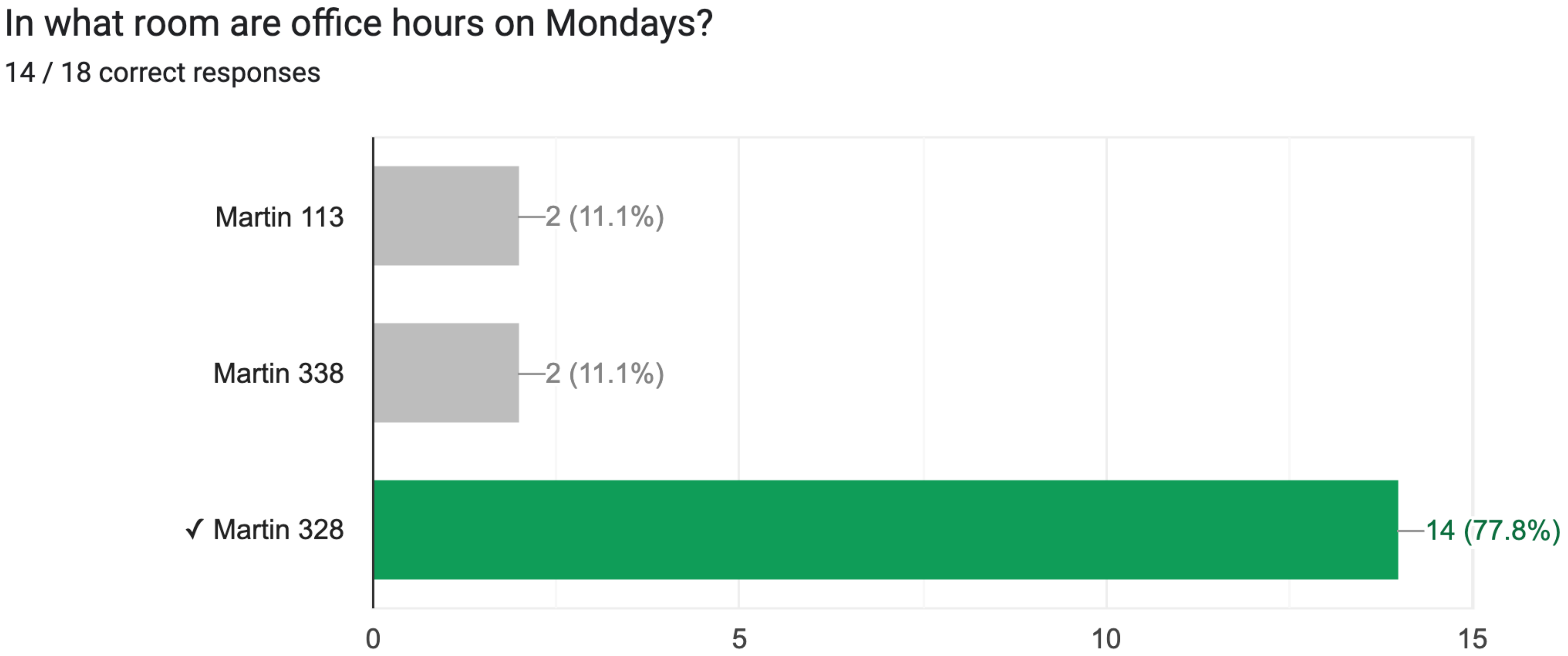
- Register your Clicker: https://forms.gle/89eA9682c6wU57Qb6

- I am out of town all of next week, Prof. Kevin Webb will give lecture and lab

Today

- Policy quiz results
- Protocols and encapsulation
- Layering
- HTTP [if time]

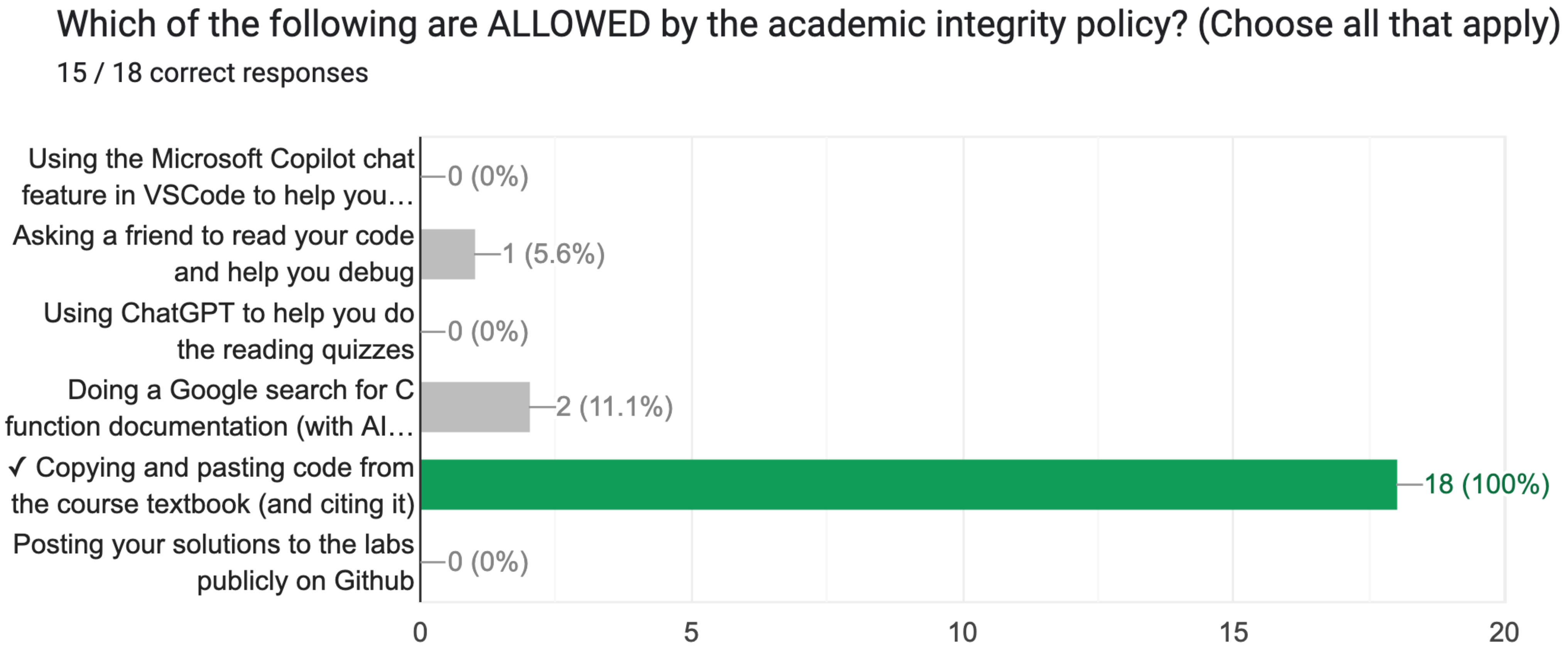
Policy Quiz Results



In what room are office hours on Mondays?

14 / 18 correct responses

Policy Quiz Results



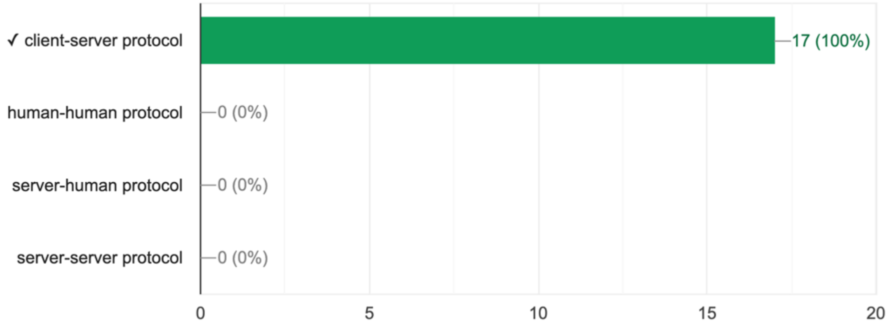
Which of the following are ALLOWED by the academic integrity policy? (Choose all that apply)

15 / 18 correct responses

Reading Quiz Results

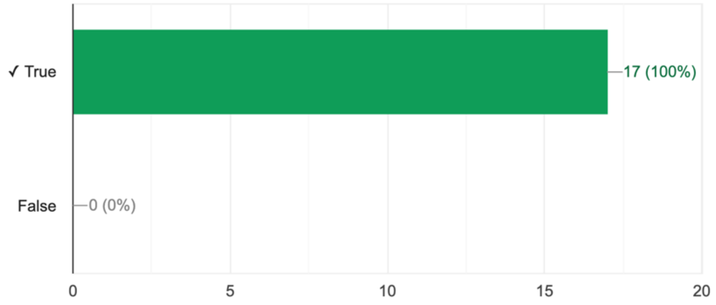
HTTP is a...

17 / 17 correct responses



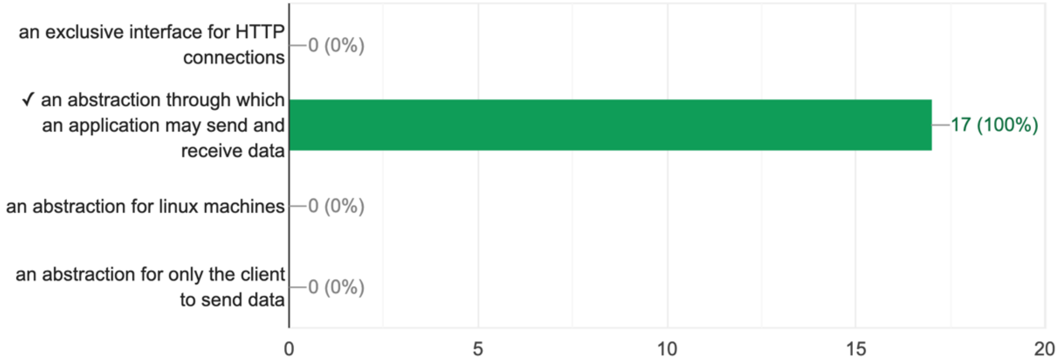
A server passively waits for an incoming connection, and a client is responsible for actively connecting to a server.

17 / 17 correct responses



A socket is

17 / 17 correct responses



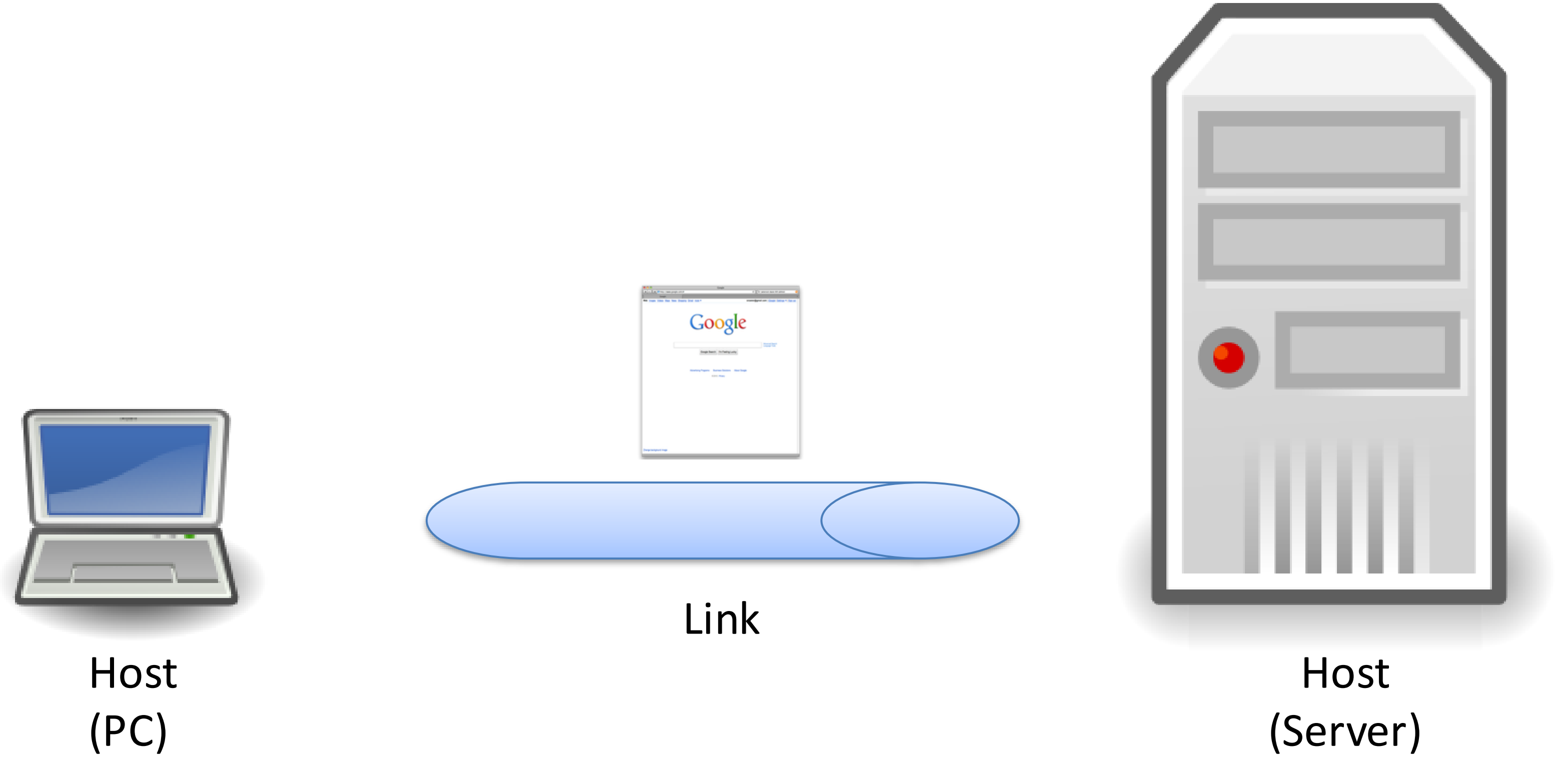
What is the goal of a network?

- Allow devices communicate with one another and coordinate their actions to work together.

- Piece of cake, right?

A “Simple” Task

<u>Send information from one computer to another</u>



Link

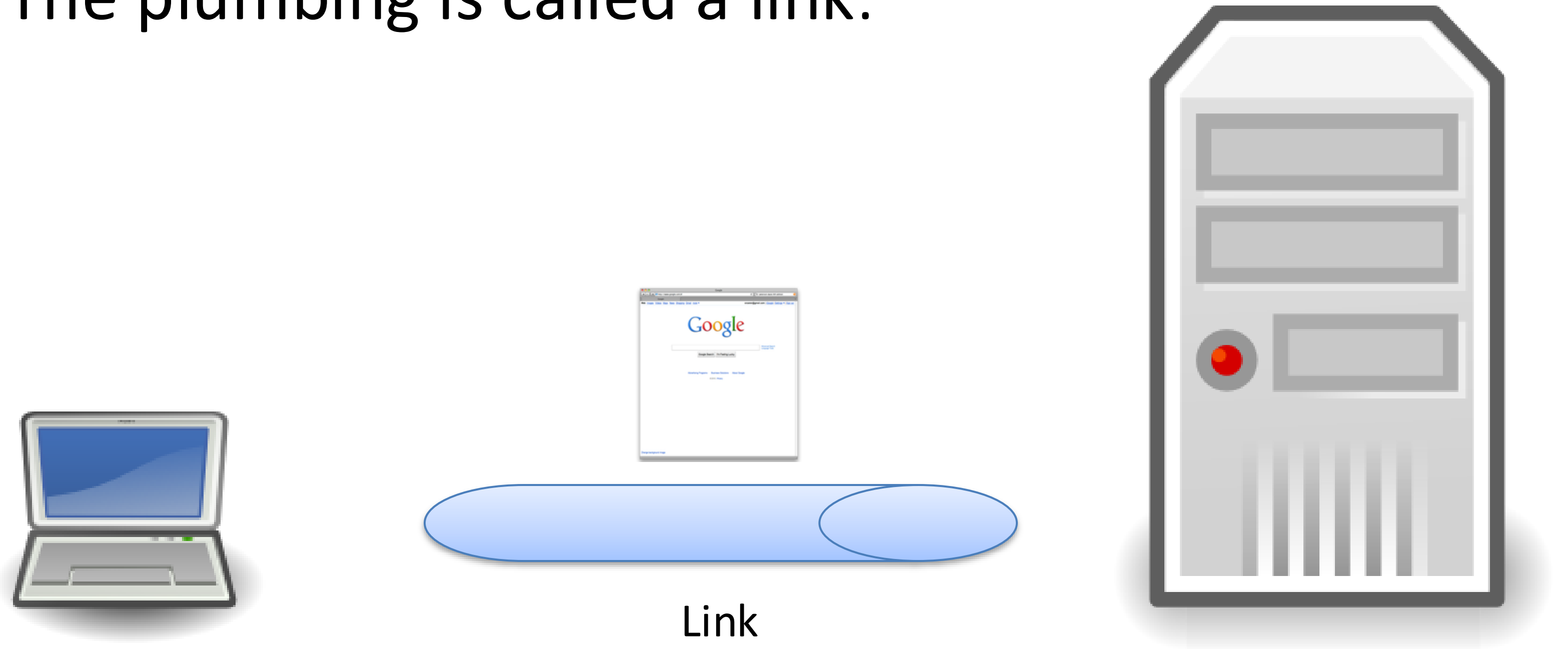
Host
(PC)

Host
(Server)

A “Simple” Task

Send information from one computer to another

- hosts: endpoints of a network
- The plumbing is called a link.



Host
(PC)

Link

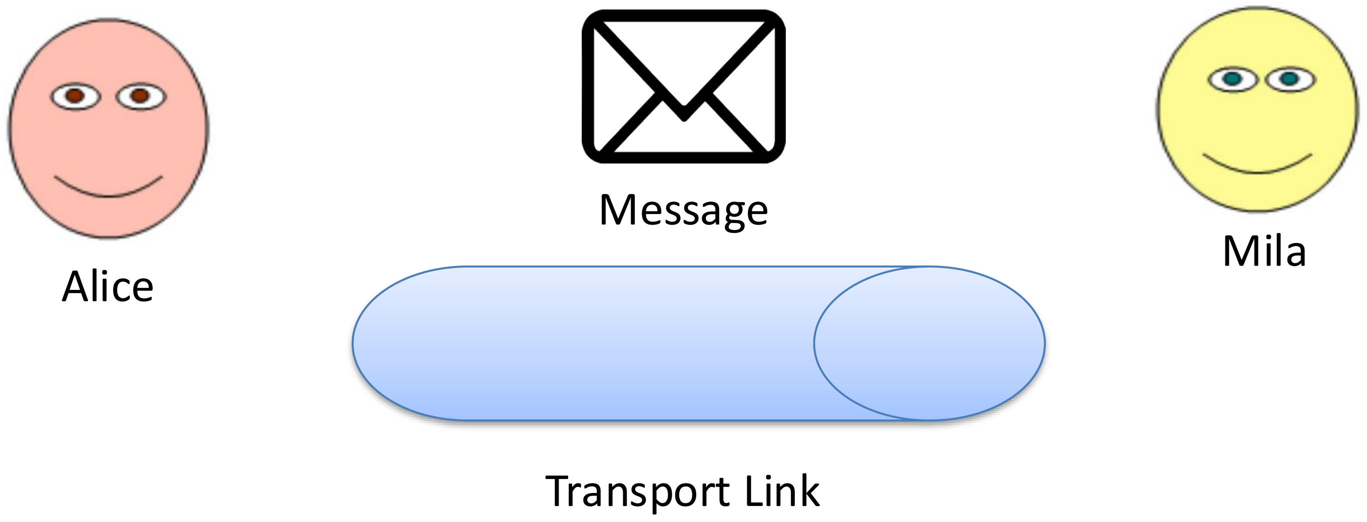
Host
(Server)

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

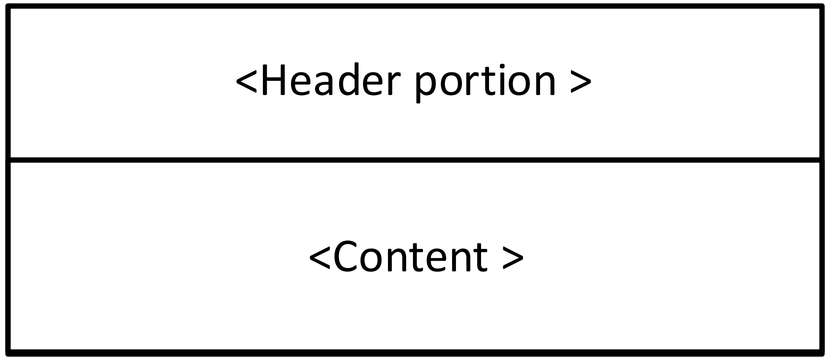
Alice and Mila are Swatties starting out their semester and are roommates. Alice wants to give Mila a reminder to get milk.



WORKSHEET
A “Simple” analogous task: Post-it Note

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

1. **Structure of the message:**
   - Construct the message that Alice posts to Mila.
2. **Organizing a drop-off point.**
   - Who chooses the drop-off point?
3. **Write a protocol to write a note /post—it to your housemate**



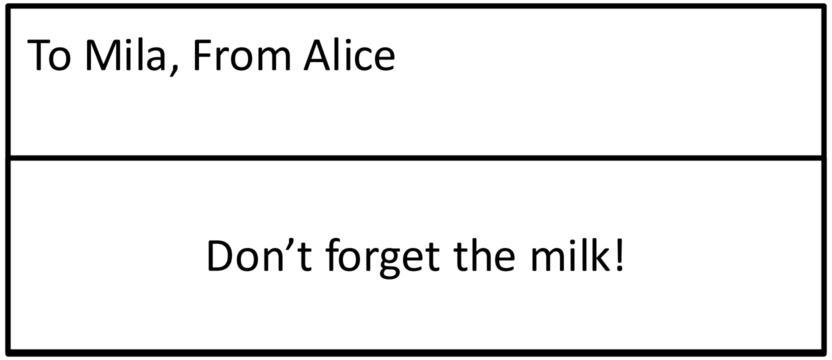
<Header portion >

<Content >

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)**



To Mila, From Alice

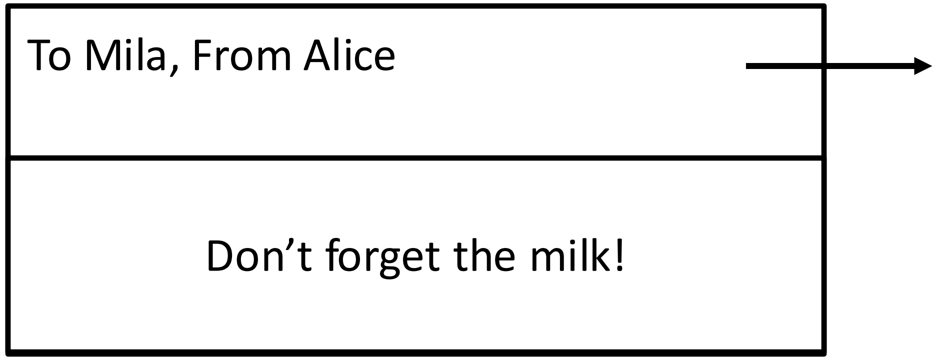
Don’t forget the milk!

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>**



To Mila, From Alice

Don’t forget the milk!

Header: contains sender/receiver information.

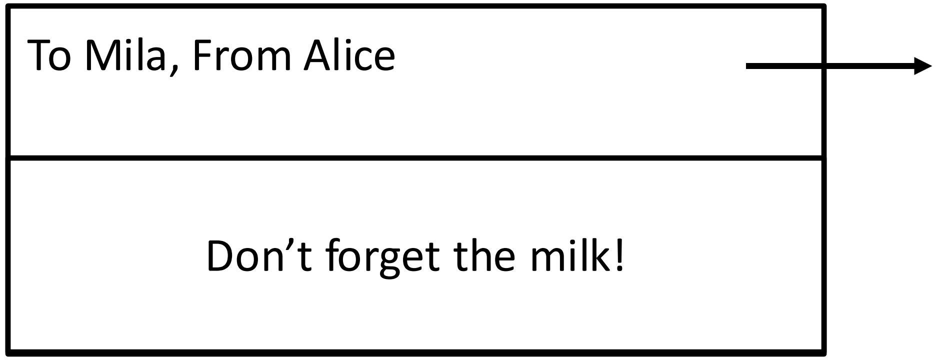
- metadata about the message
- what other metadata might you add?

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



To Mila, From Alice

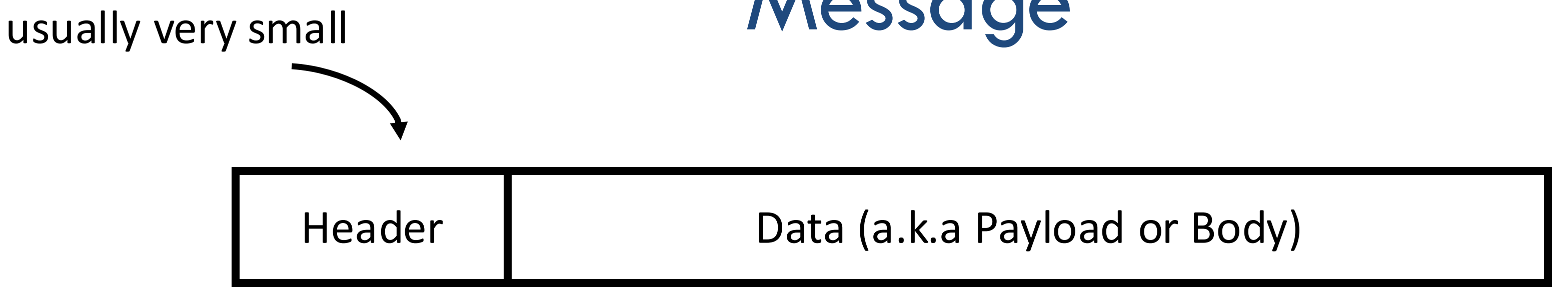
Don’t forget the milk!

Header: contains sender/receiver information + additional state

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

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

Message



usually very small

Header

Data (a.k.a Payload or Body)

- 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)

A “Simple” analogous task: Post-it Note

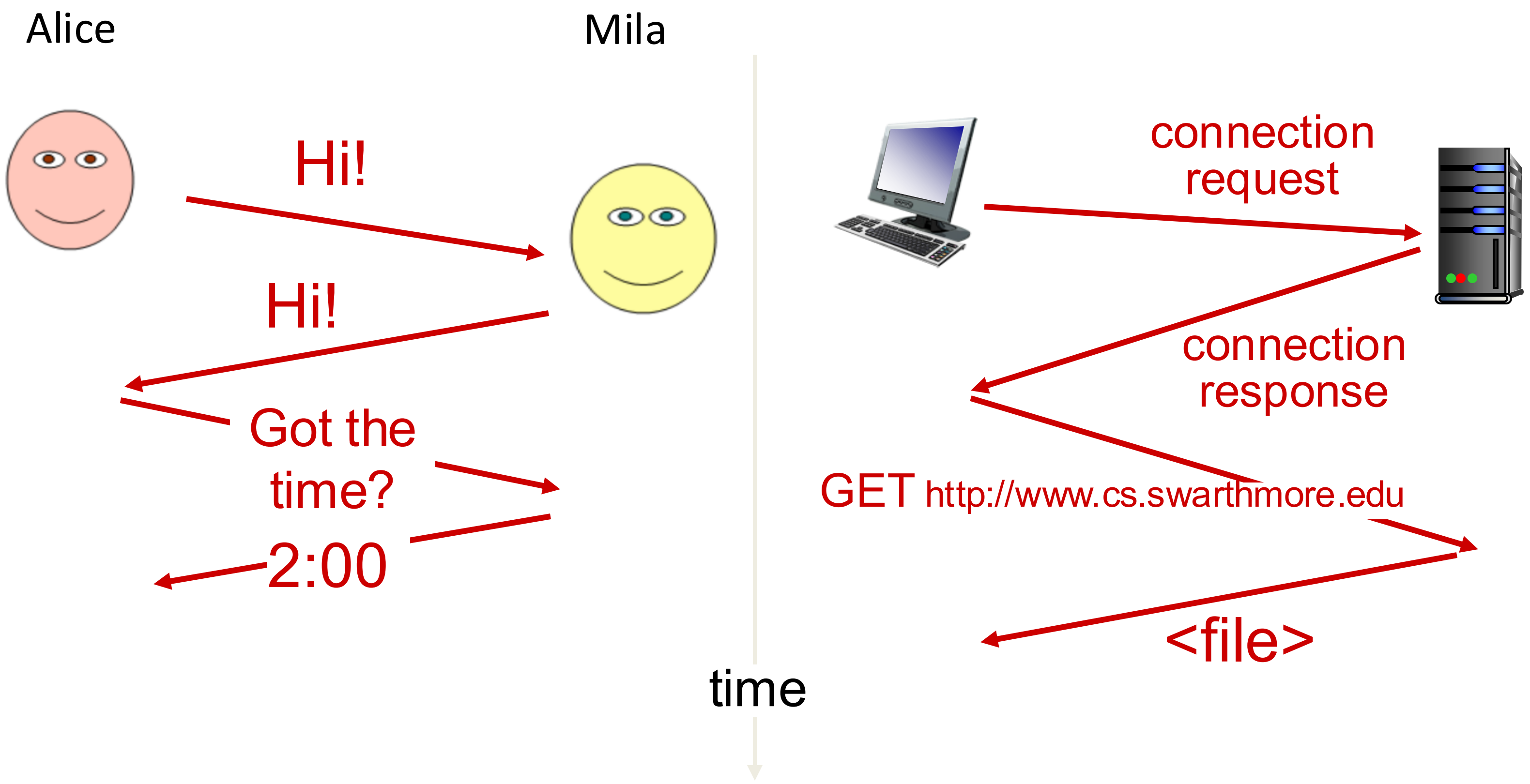
Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk.

2. **<u>Organizing a drop-off point.</u>**
    - Who decides?
    - Generally by mutual consensus – previously agreed upon location.

<u>Everyone agrees to place messages on refrigerator to relay messages to housemates</u>

What is a protocol?

Protocol: message format + transfer procedure



Alice

Mila

Hi!

Hi!

Got the time?

2:00

time

connection request

connection response

GET http://www.cs.swarthmore.edu

<file>

Human Protocol

Network Protocols (defined in RFCs)

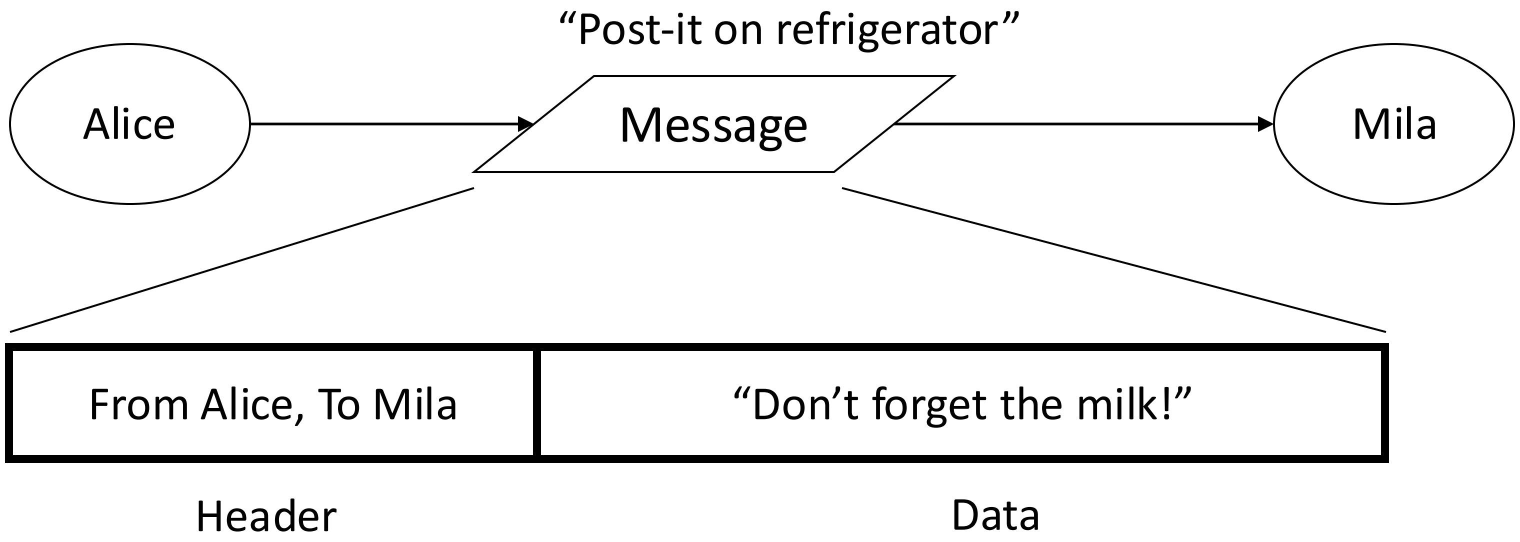
What is a protocol?

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: Post-it Note



“Post-it on refrigerator”

Alice

Message

Mila

From Alice, To Mila

“Don’t forget the milk!”

Header

Data

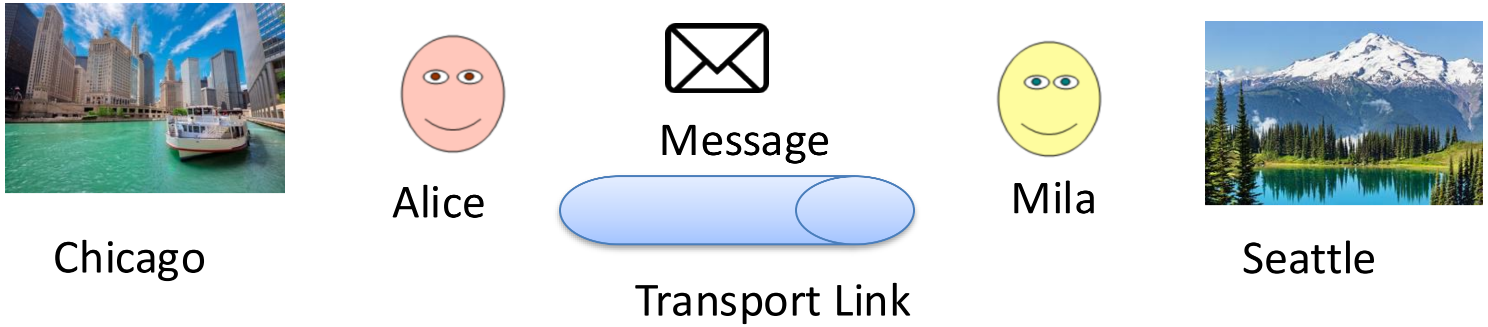
Write a protocol to write a note /post—it to your housemate

Protocol: message format + transfer procedure

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

A “Simple” analogous task: Postal Mail

Alice moves to Chicago and Mila to Seattle for summer internships. Alice would like to send Mila a birthday card. Think of this as filling two different pieces of information (1. the birthday card, 2. the mailing envelope).



Chicago

Alice

Message

Transport Link

Mila

Seattle

A “Simple” analogous task: Postal Mail

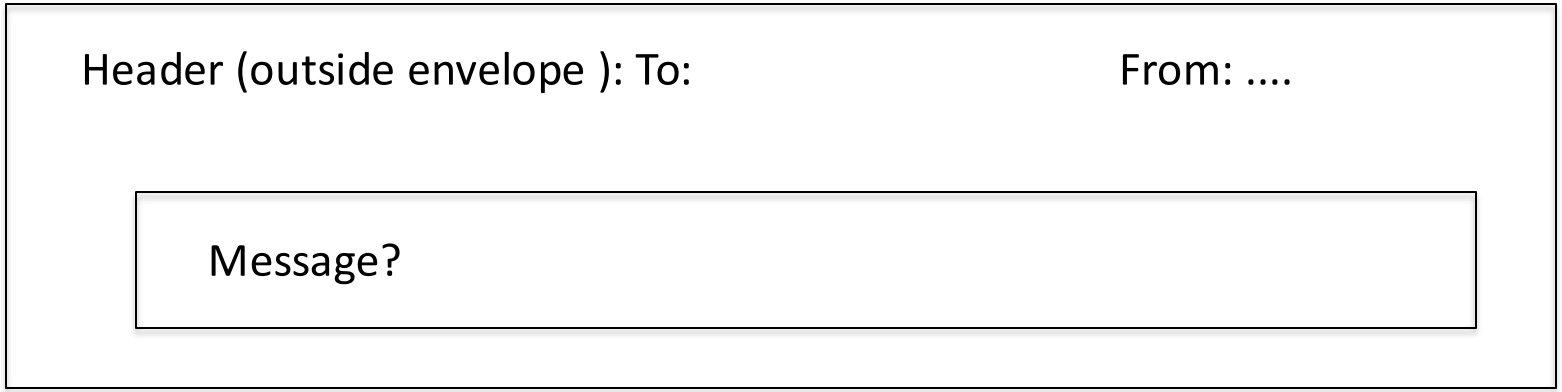
Alice would like to send Mila a birthday card.

1. **Construct the message and header. Have the header and message portions changed from the previous scenario?**
2. **List the message format and transfer procedure of the “mail sending protocol” that Alice uses.**
   - Who chooses the drop-off point?
   - Is this the only protocol in use?
3. **Message transportation and delivery**
   - Whose job is it to:
     - choose the carrier?
     - plan the route?
     - deliver the message?
     - ensure the message is not lost?

A “Simple” analogous task: Postal Mail

Alice would like to send Mila a birthday card.

1. **<u>Construct the message and the header. Have the header and message portions changed from the previous scenario?</u>**



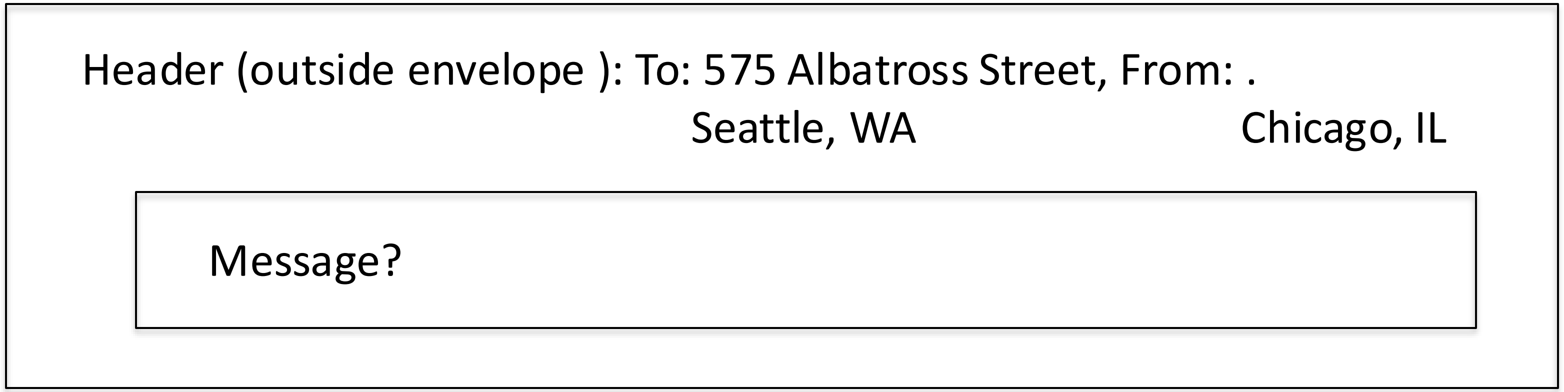
Header (outside envelope): To: From: ....

Message?

A “Simple” analogous task: Postal Mail

Alice would like to send Mila a birthday card.

Header portion of the envelope



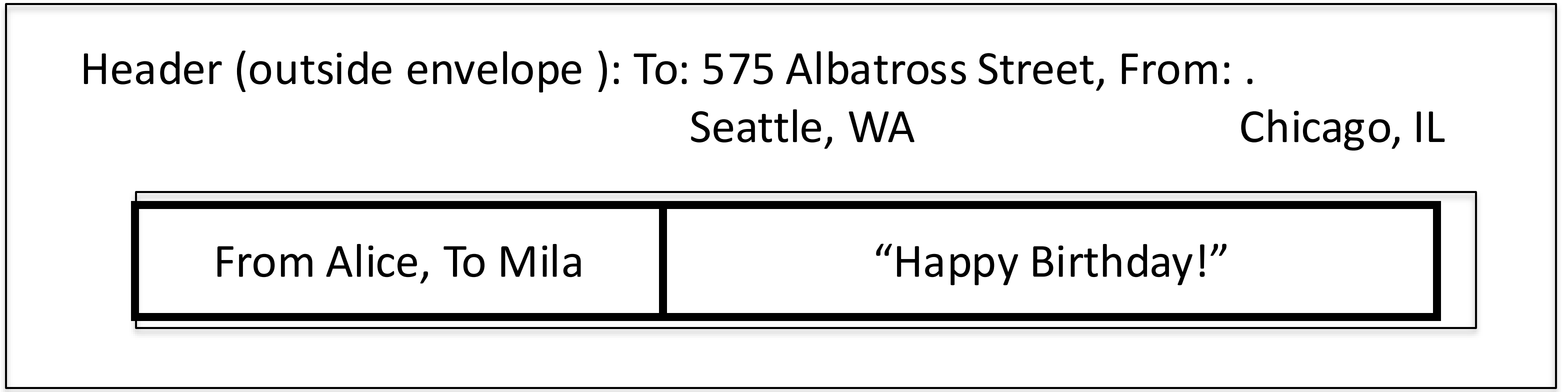
Header (outside envelope): To: 575 Albatross Street, From: .
Seattle, WA Chicago, IL

Message?

A “Simple” analogous task: Postal Mail

Alice would like to send Mila a birthday card.

Message portion of the envelope

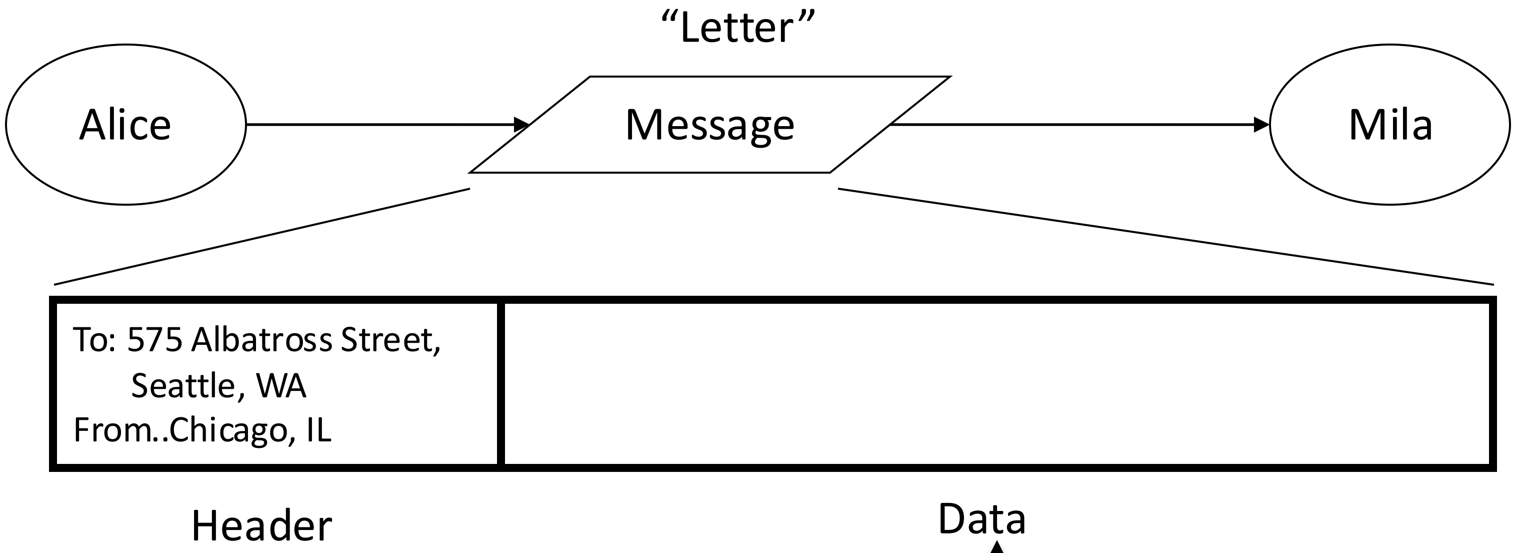


Header (outside envelope): To: 575 Albatross Street, From: .
Seattle, WA Chicago, IL

From Alice, To Mila

“Happy Birthday!”

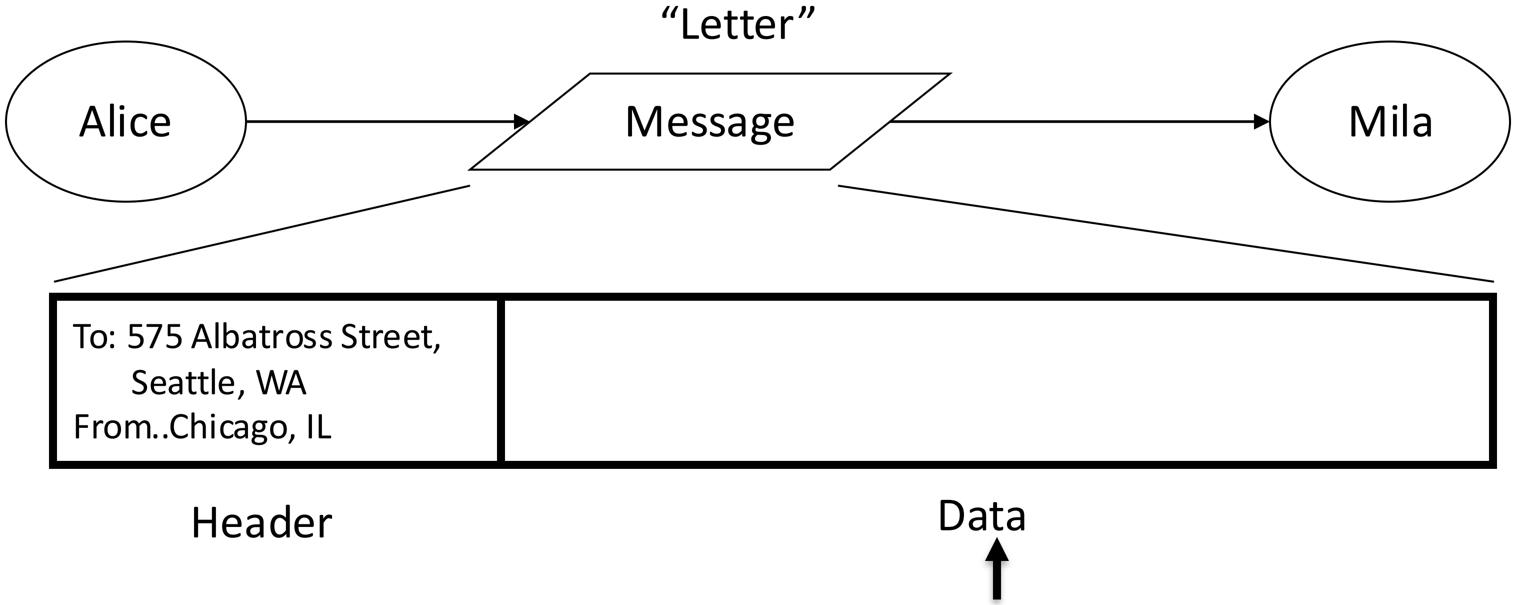
A “Simple” analogous task: Postal Mail



To: 575 Albatross Street,
Seattle, WA
From..Chicago, IL

The post office does NOT care about what’s in here, and shouldn’t be looking at it…

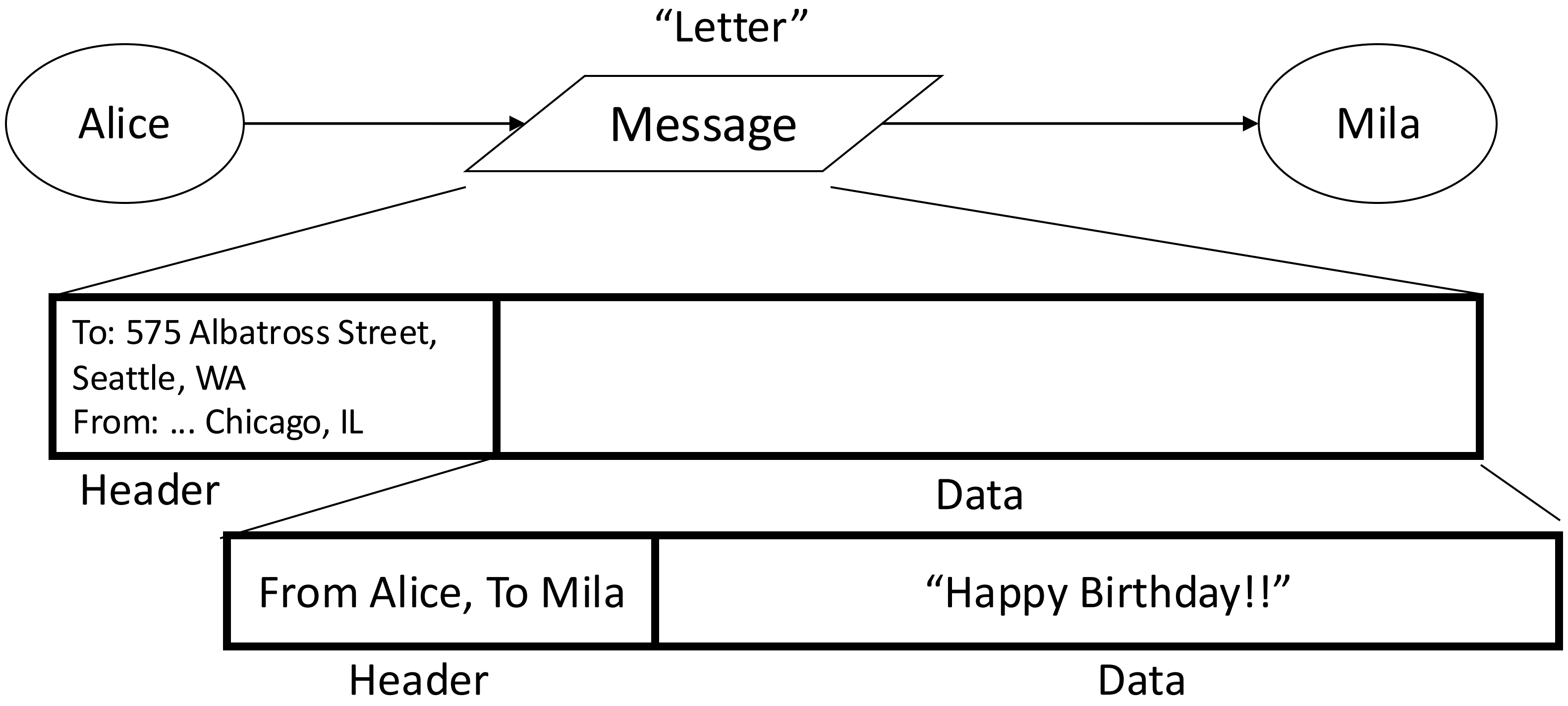
A “Simple” analogous task: Postal Mail



The post office does NOT care about what’s in here, and shouldn’t be looking at it…

- **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?



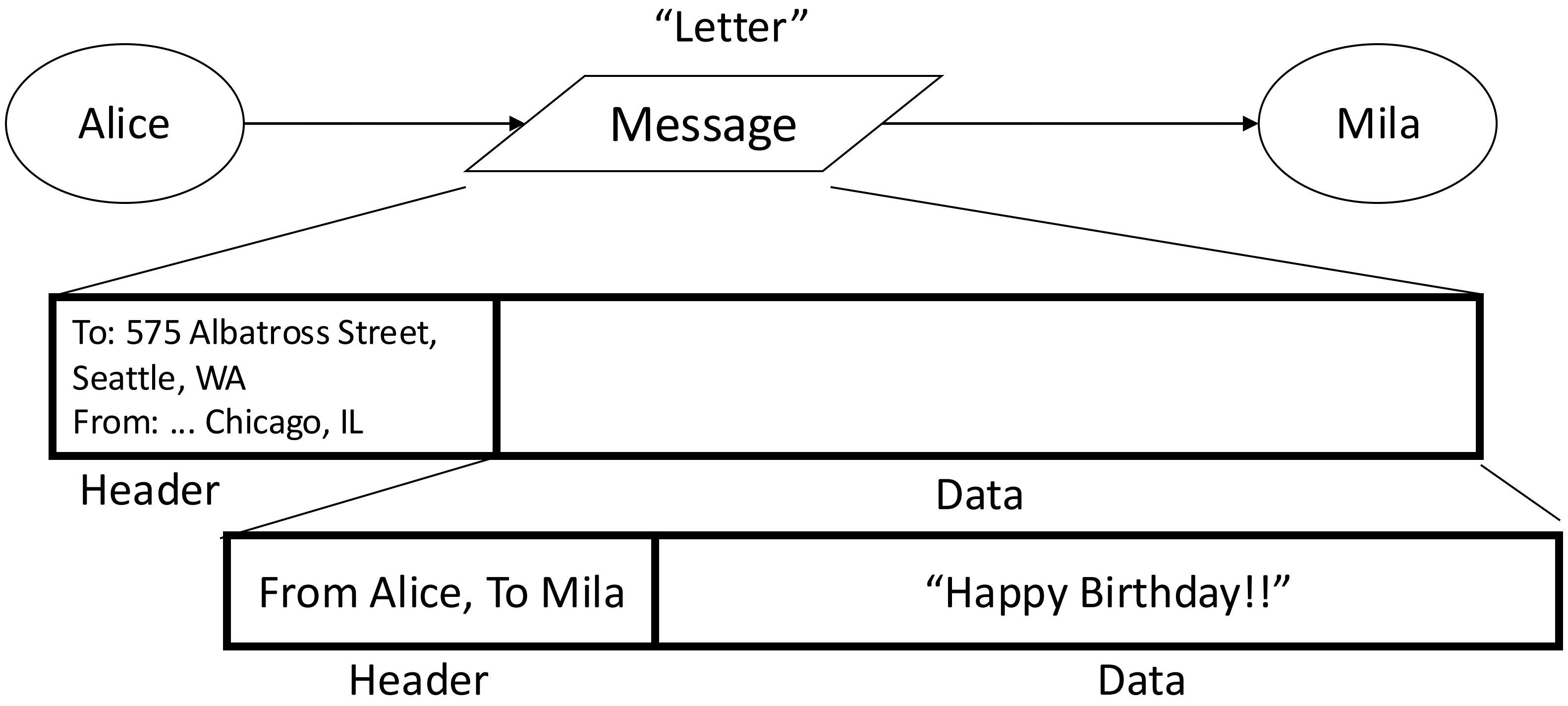
Mail Protocol

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

Card Protocol (within the mail protocol!)

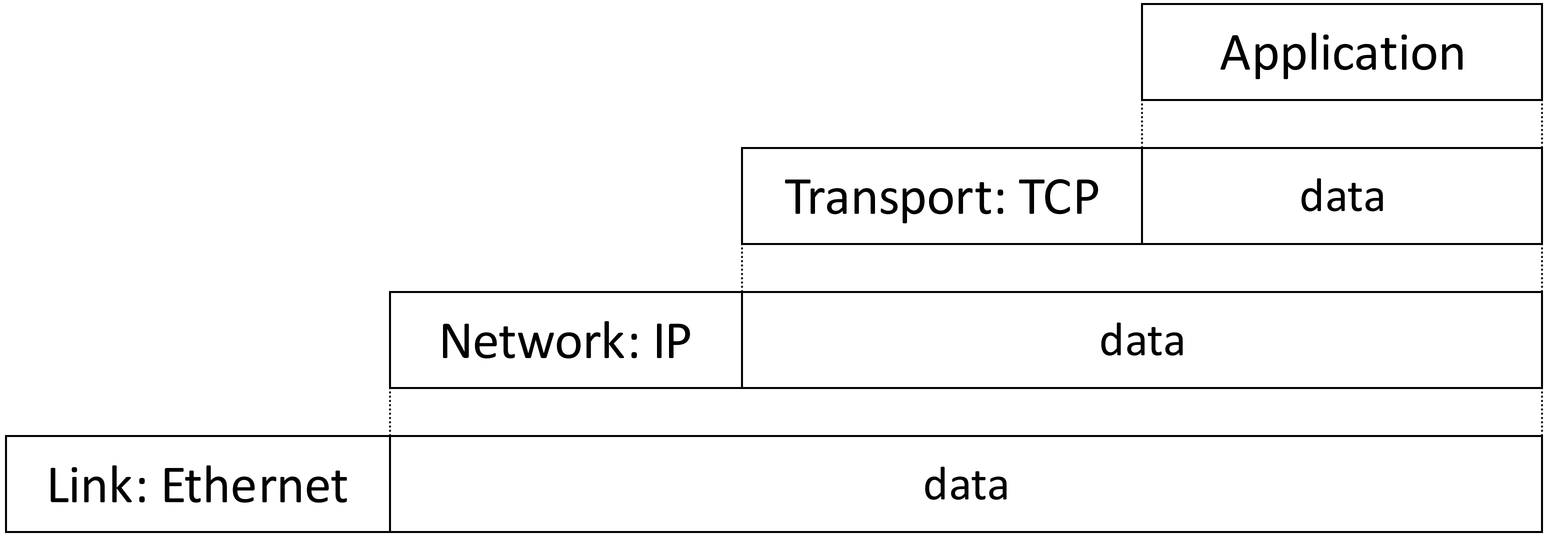
- Message format: (from, to), message contents

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

- **<u>Message transportation and delivery</u>**
- Who’s job is it to:
  1. provide the sender and receiver addresses?
  2. choose the carrier?
  3. plan the route?
  4. deliver the message?
  5. ensure the message is not 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”

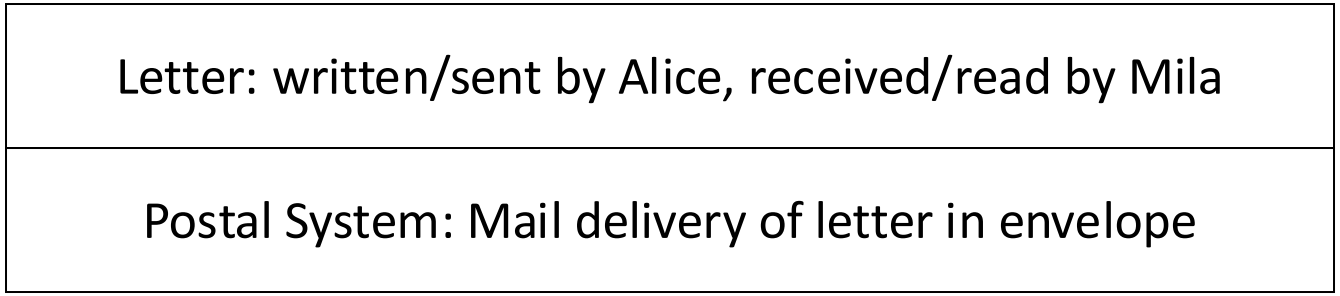
3. plan the route?
4. transport vehicles?

(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

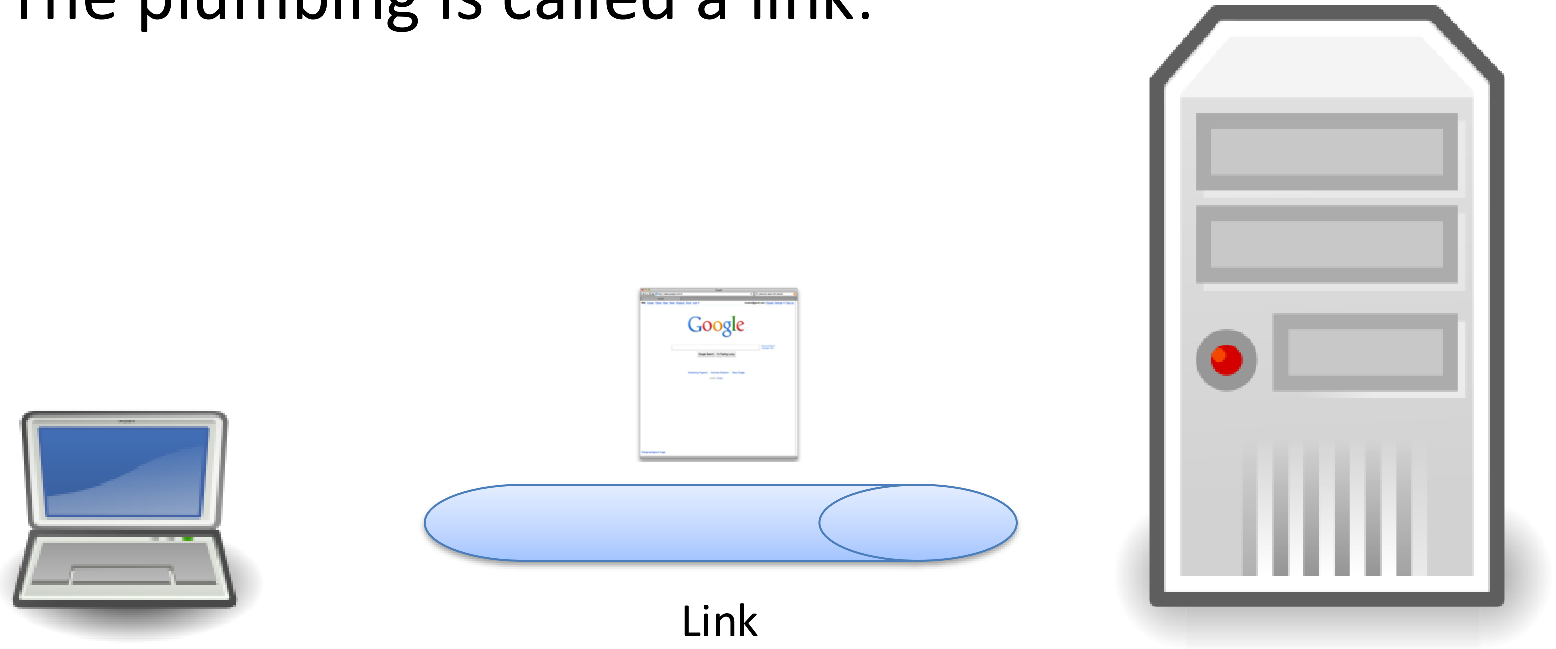
A “Simple” analogous task: Postal Mail

- 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?

A “Simple” Task

Send information from one computer to another

- hosts: endpoints of a network
- The plumbing is called a link.

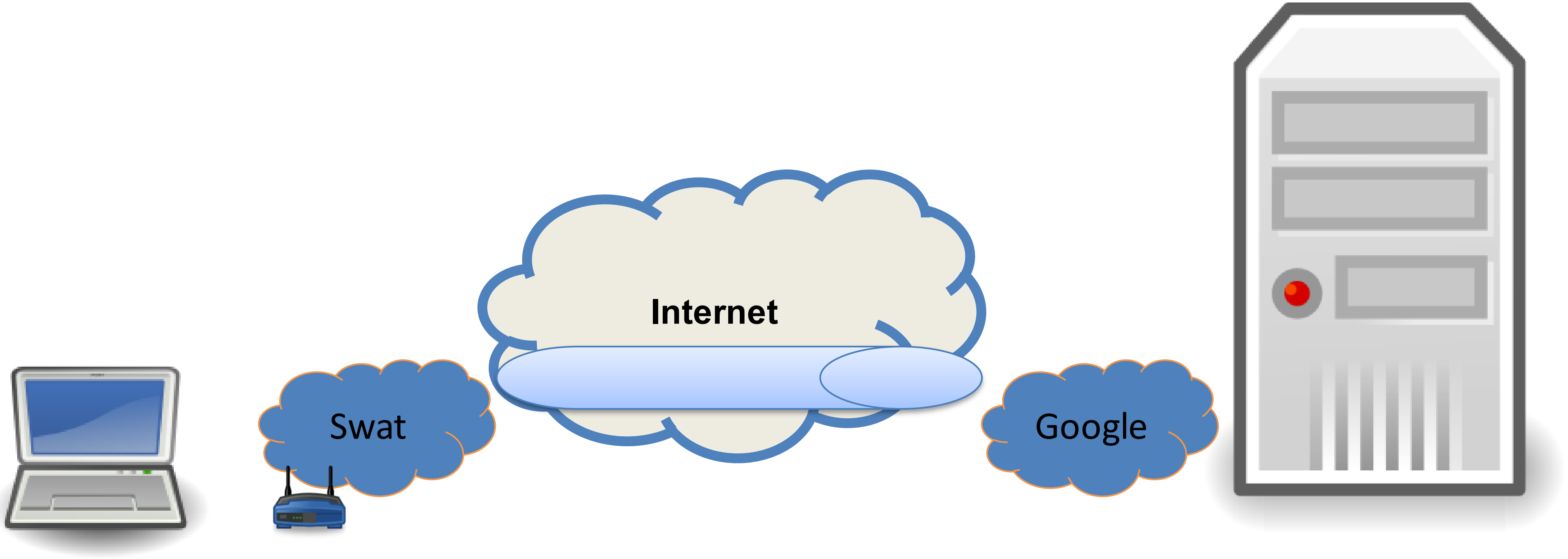


Host
(PC)

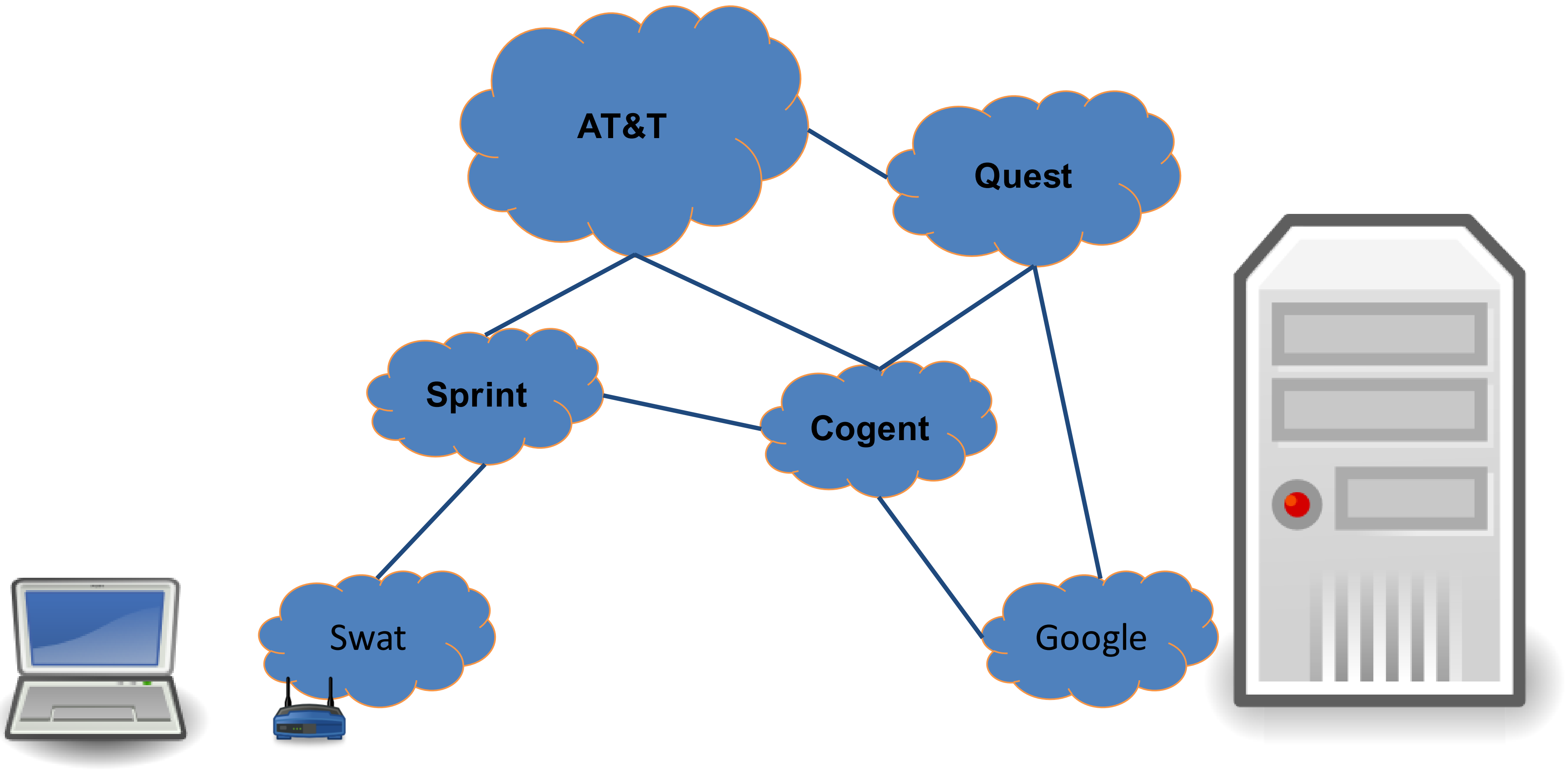
Link

Host
(Server)

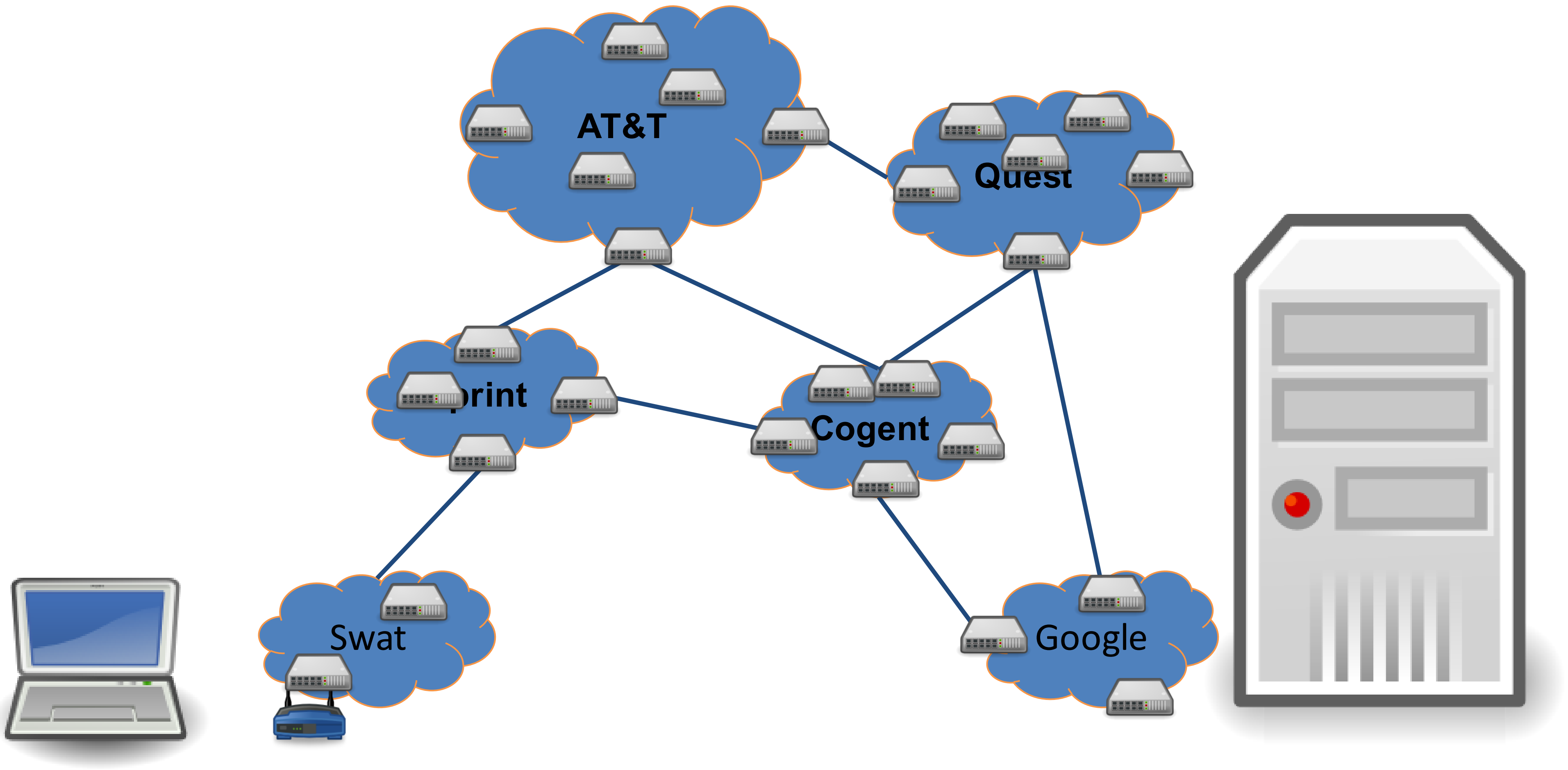
Not Really So Simple…



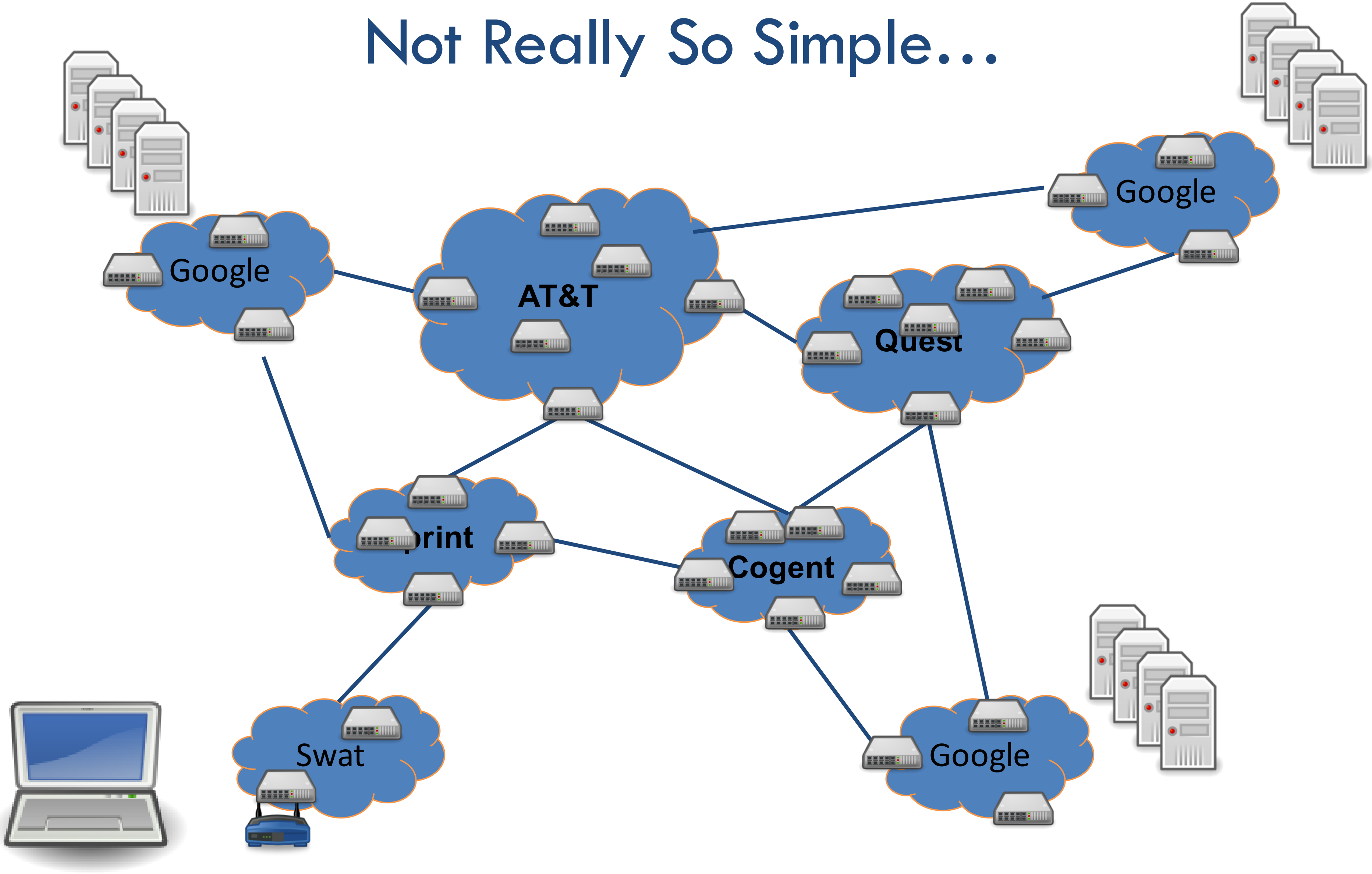
Not Really So Simple…



Not Really So Simple…



Not Really So Simple…



We only need…

- Manage complexity and scale up
  - Layering abstraction: divide responsibility
  - Protocols: standardize behavior for interoperability

We only need…

- Manage complexity and scale up
- Naming and addressing
  - Agreeing on how to describe/express a host, application, network, etc.

We only need…

- Manage complexity and scale up
- Naming and addressing
- Moving data to the destination
  - Routing: deciding how to get it there
  - Forwarding: copying data across devices/links

We only need…

- Manage complexity and scale up
- Naming and addressing
- Moving data to the destination
- Reliability and fault tolerance
  - How can we guarantee that the data arrives?
  - How do we handle link or device failures?

We only need…

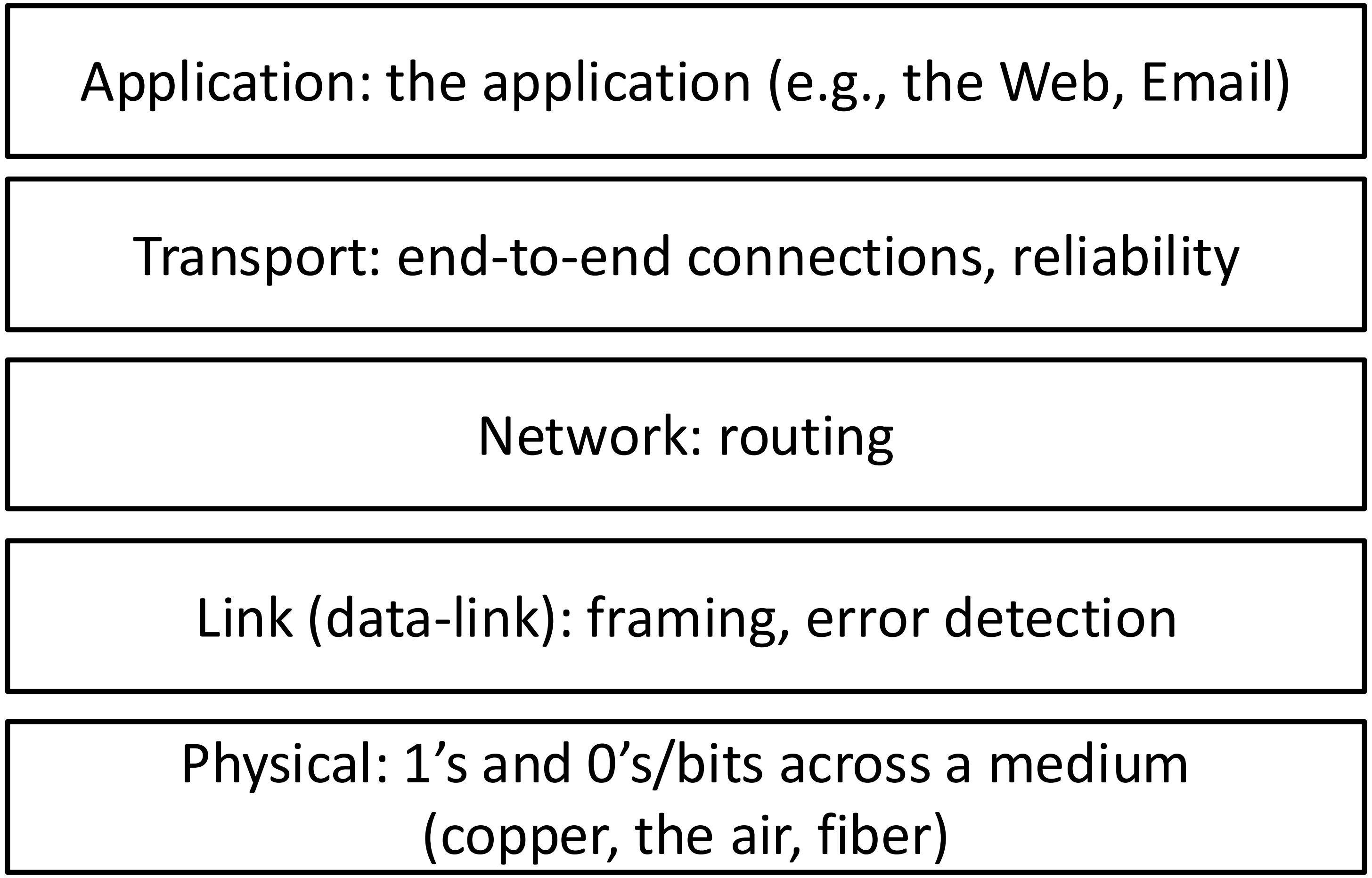
- Manage complexity and scale up
- Naming and addressing
- Moving data to the destination
- Reliability and fault tolerance
- Resource allocation, Security, Privacy..

We only need…

- Manage complexity and scale up
- Naming and addressing
- Moving data to the destination
- Reliability and fault tolerance
- Resource allocation, Security, Privacy..

(Lots of others too.)

Five-Layer Internet Model



Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

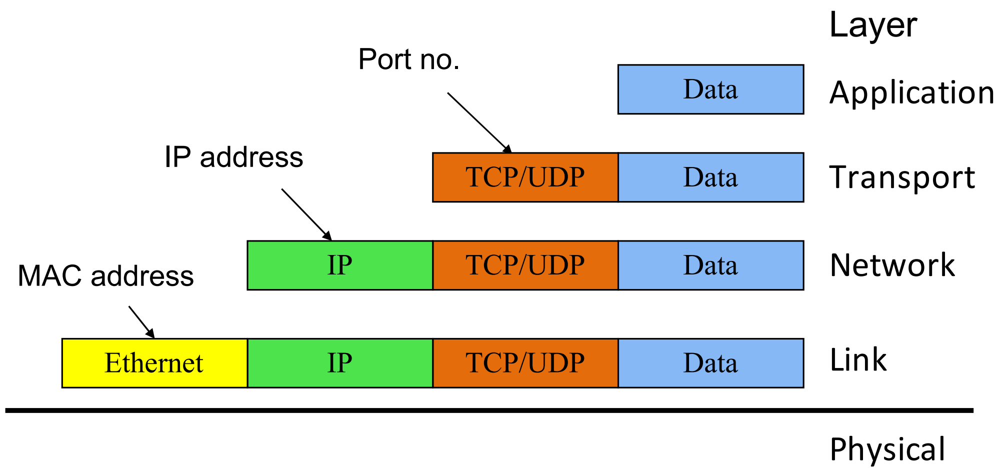
Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium
(copper, the air, fiber)

Application Layer
(HTTP, FTP, SMTP, Tiktok)

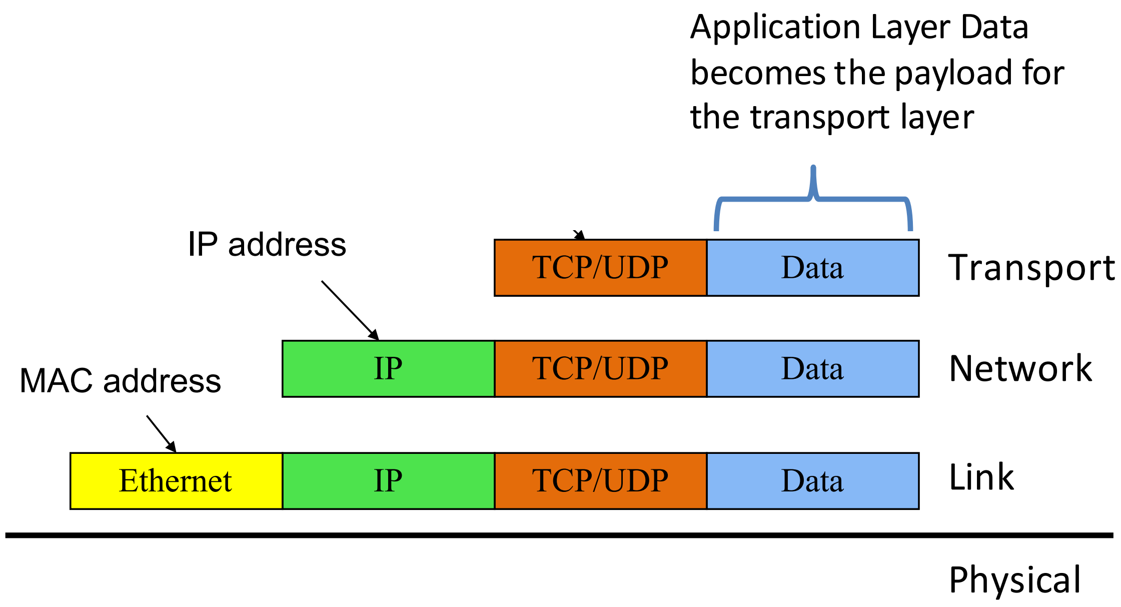
- Does whatever an application does!



				Layer
			Data	Application
		TCP/UDP	Data	Transport
	IP	TCP/UDP	Data	Network
Ethernet	IP	TCP/UDP	Data	Link
				Physical

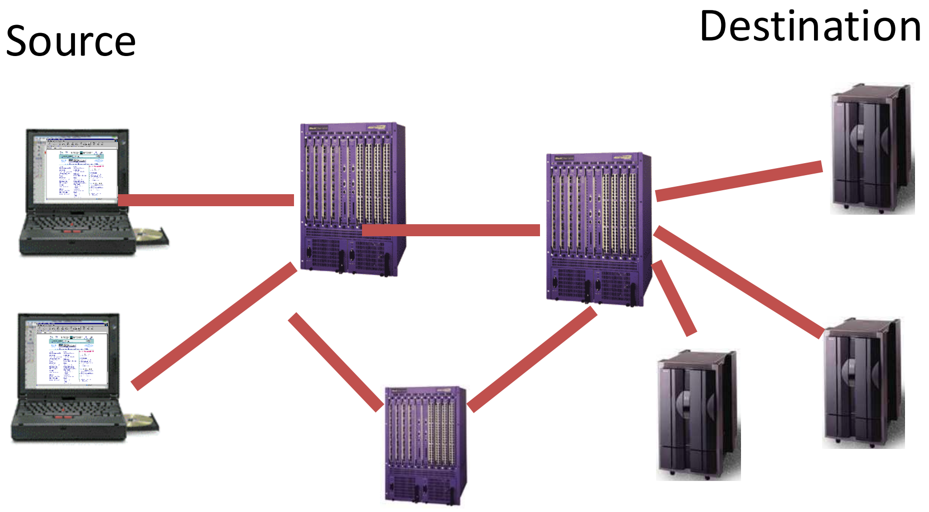
Transport Layer (TCP, UDP)

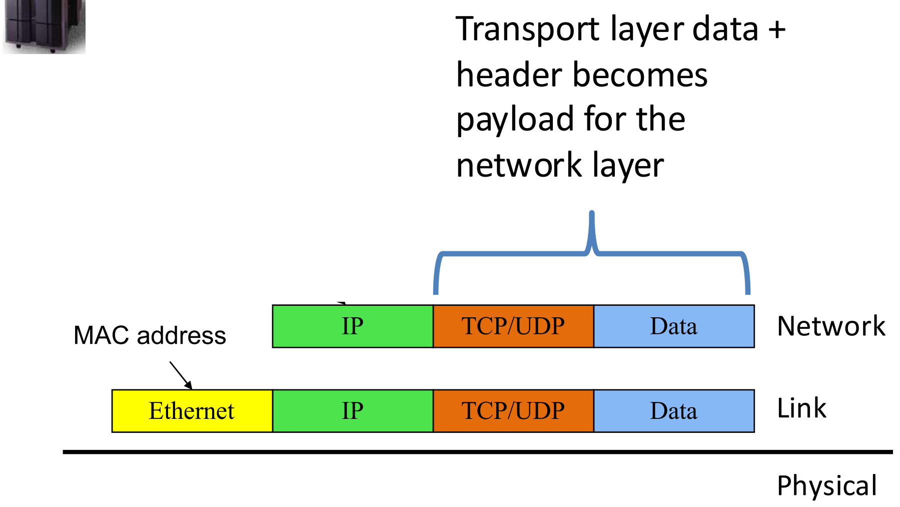
- Provides
  - Ordering
  - Error checking
  - Delivery guarantee
  - Congestion control
  - Flow control
- Or doesn’t!



Network Layer (IP)

- **Routers**: choose paths through network





Transport layer data + header becomes payload for the network layer

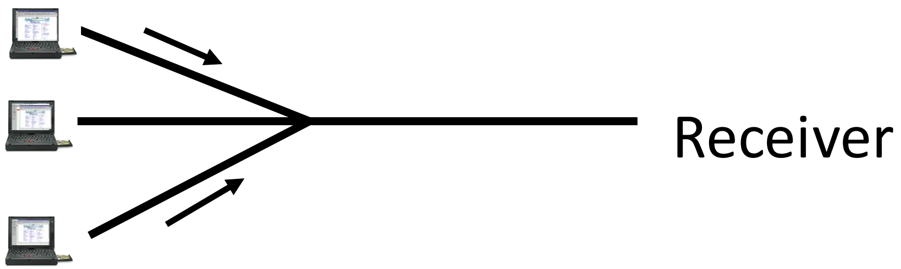
MAC address

	IP	TCP/UDP	Data	Network
Ethernet	IP	TCP/UDP	Data	Link

Physical

Link Layer (Ethernet, WiFi, Cable)

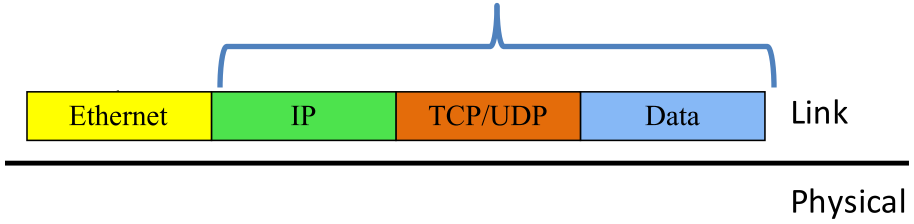
- Who’s turn is it to send right now?
- Break message into frames
- Media access: can it send the frame now?



Receiver

- Send frame, handle “collisions”

Network layer data + header becomes payload for the link layer



Ethernet

IP

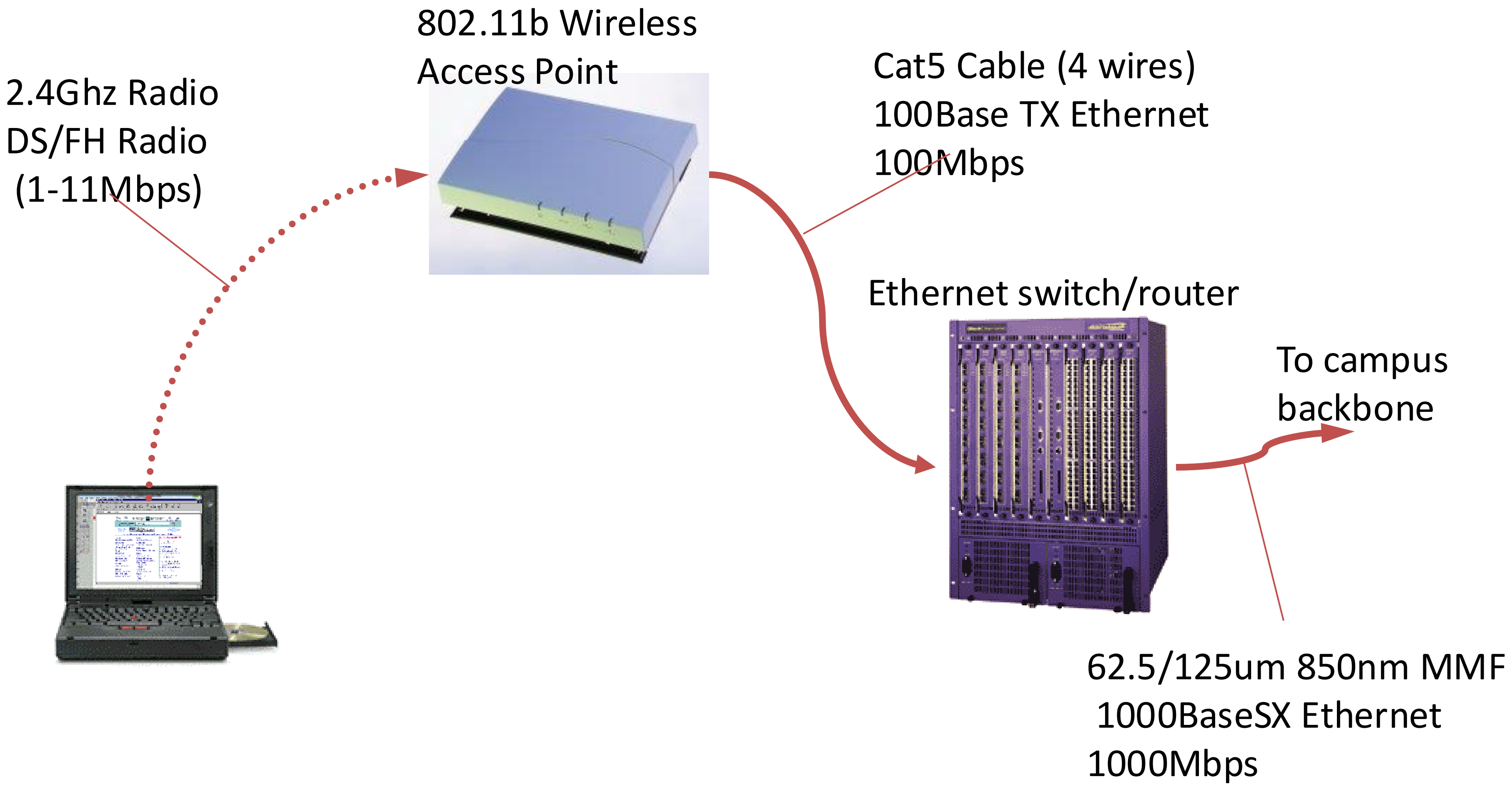
TCP/UDP

Data

Link

Physical

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



802.11b Wireless
Access Point

2.4Ghz Radio
DS/FH Radio
(1-11Mbps)

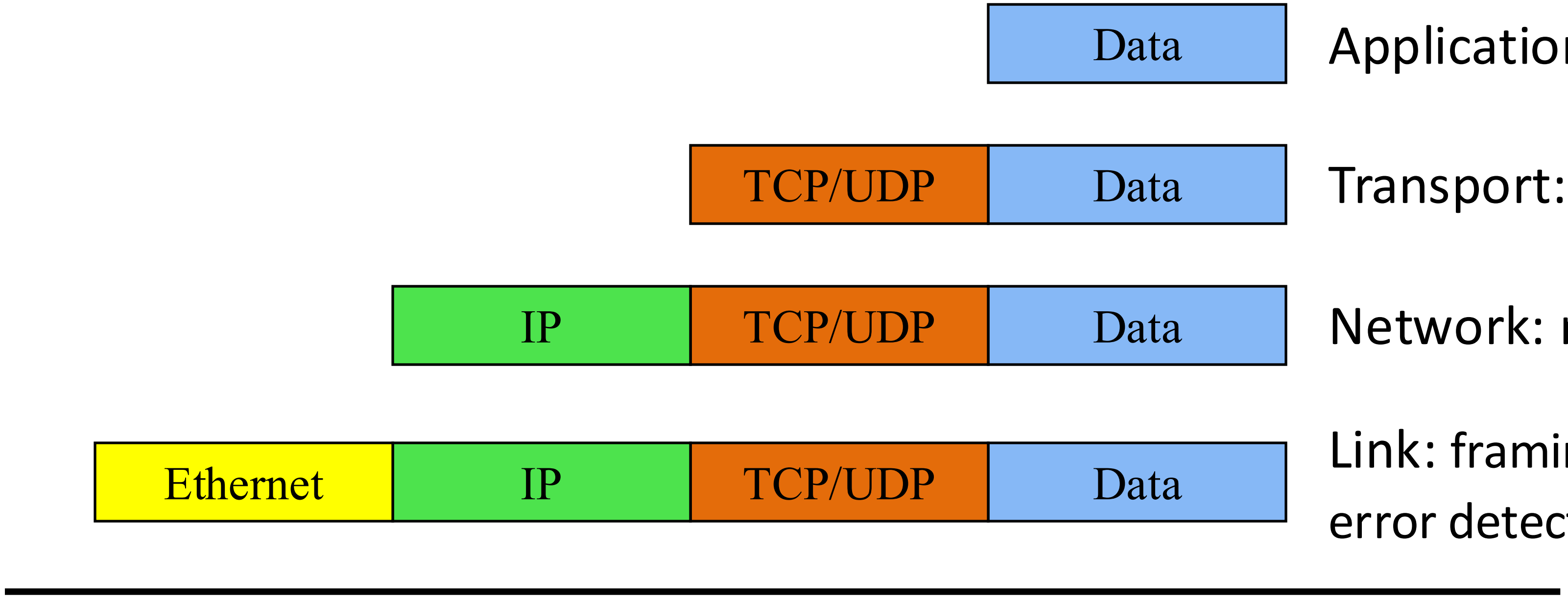
Cat5 Cable (4 wires)
100Base TX Ethernet
100Mbps

Ethernet switch/router

To campus
backbone

62.5/125um 850nm MMF
1000BaseSX Ethernet
1000Mbps

Layering and encapsulation



Layer

Data — Application

TCP/UDP | Data — Transport: reliability

IP | TCP/UDP | Data — Network: routing

Ethernet | IP | TCP/UDP | Data — Link: framing, error detection

Physical

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

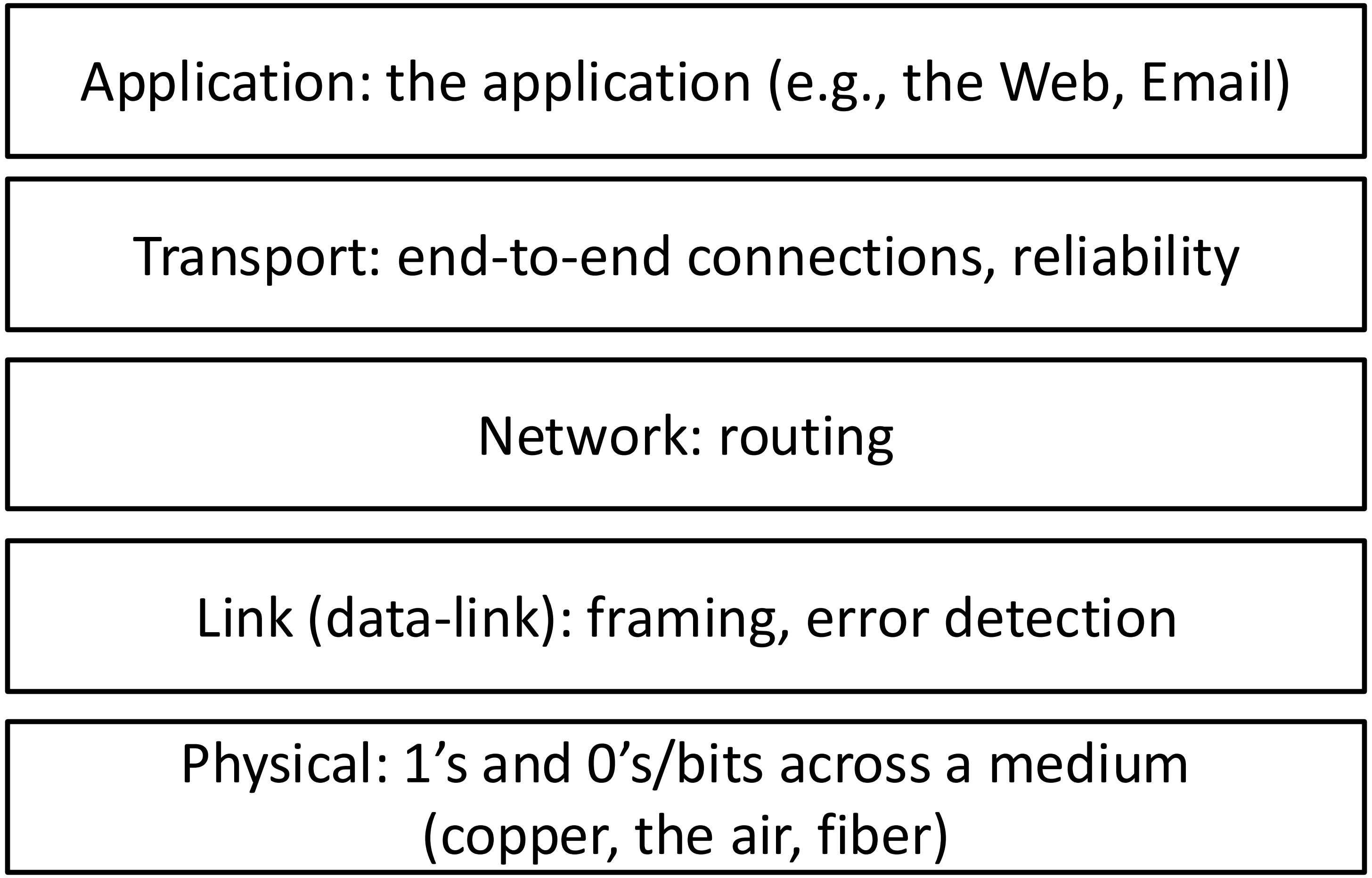
Abstraction!

- Hides the complex details of a process

- Use abstract representation of relevant properties make reasoning simpler

- Ex: Your knowledge of postal system:
  - Letters with addresses go in, come out other side

Five-Layer Internet Model



Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium
(copper, the air, fiber)

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?)

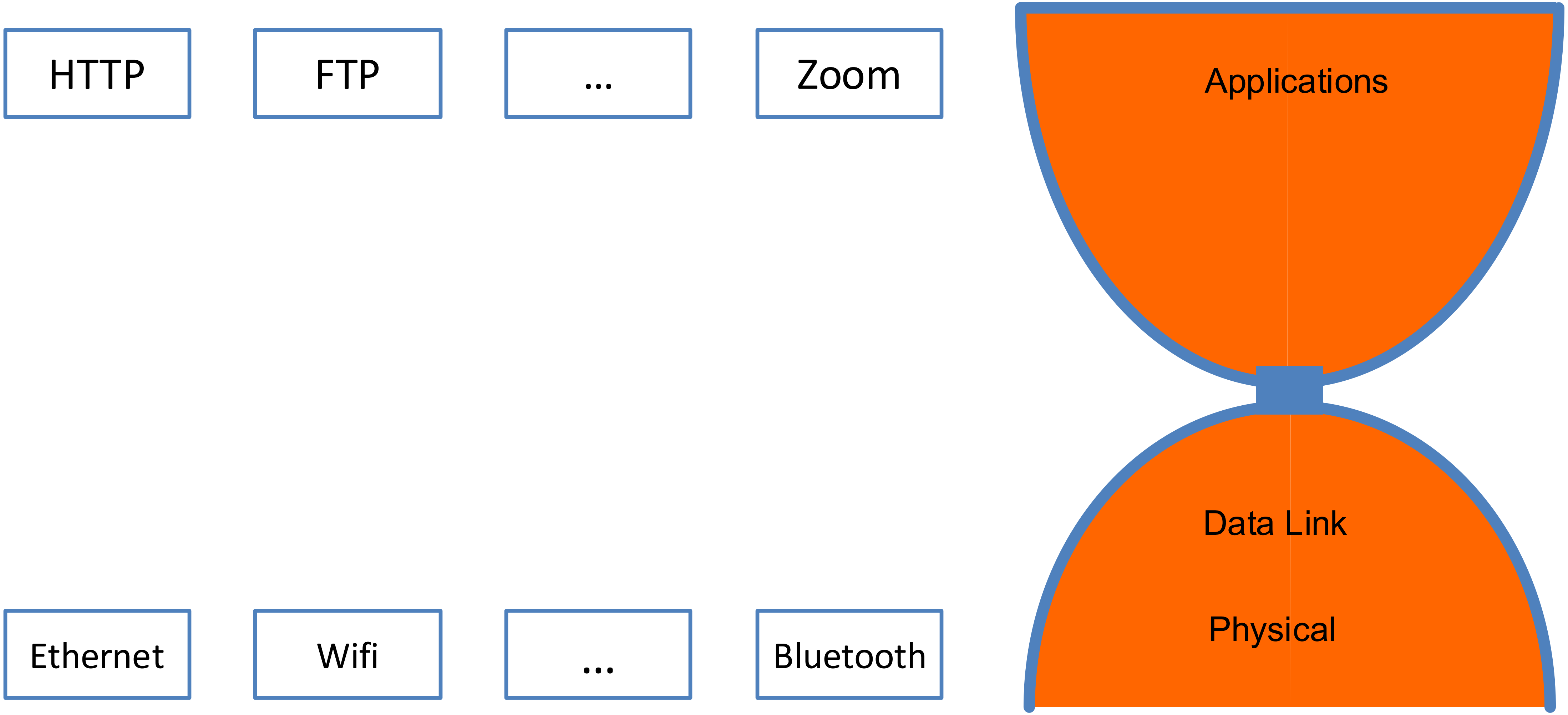
A. Always

B. Usually

C. Sometimes

D. Never

Internet Protocol Suite



HTTP

FTP

...

Zoom

Applications

Data Link

Physical

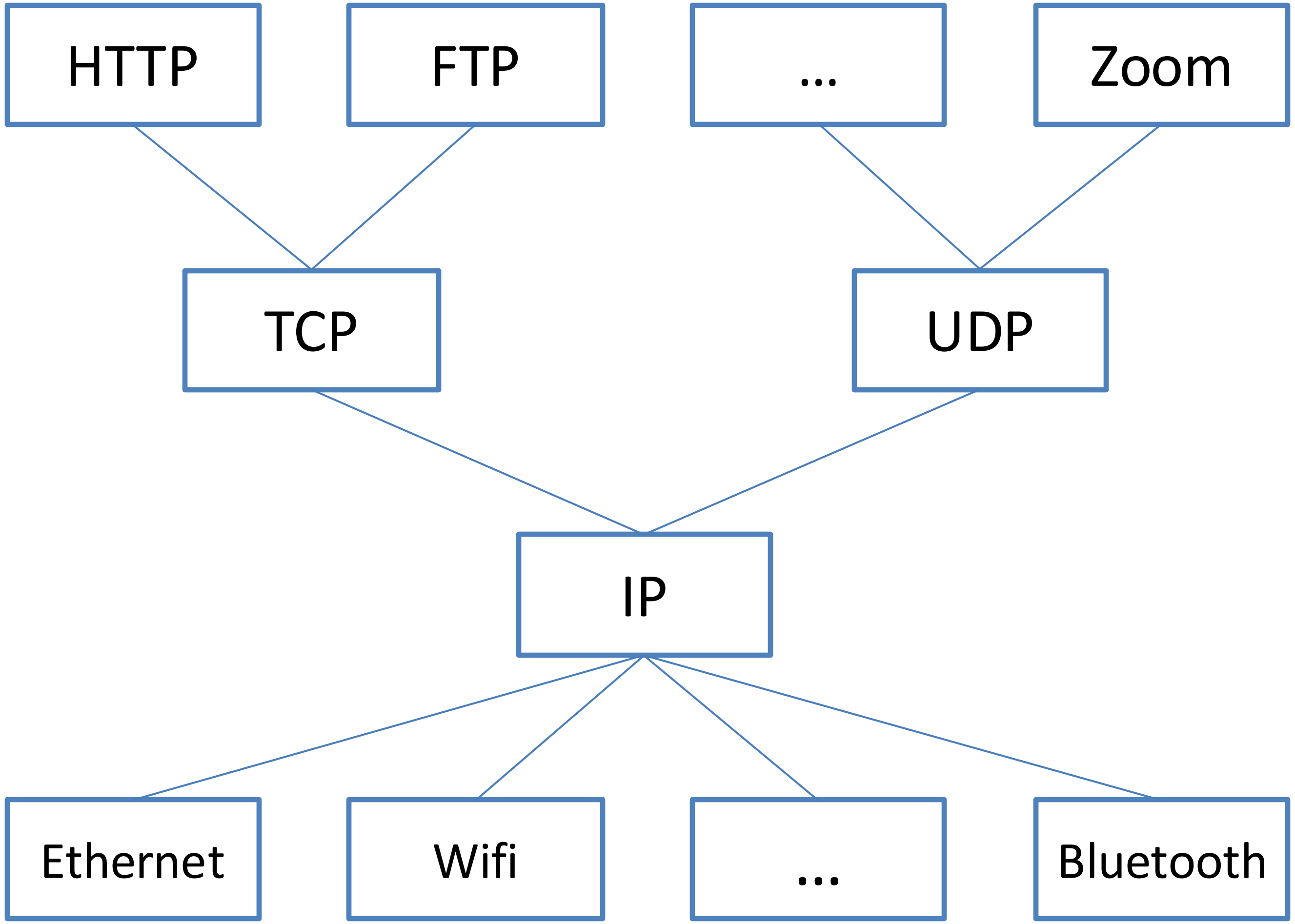
Ethernet

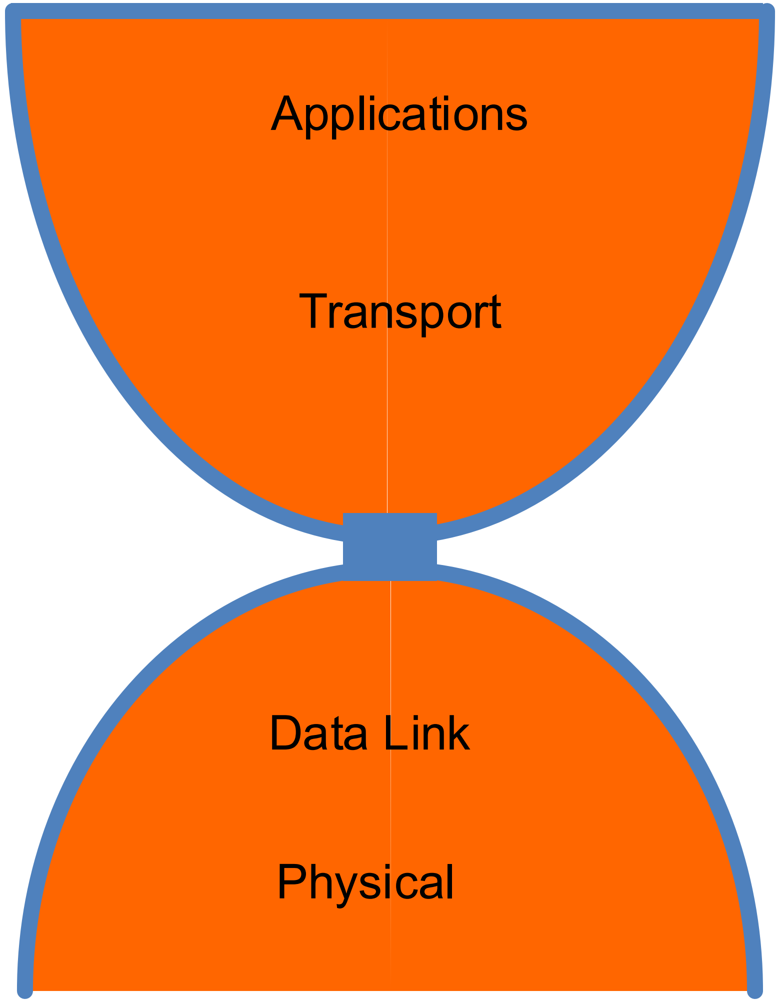
Wifi

...

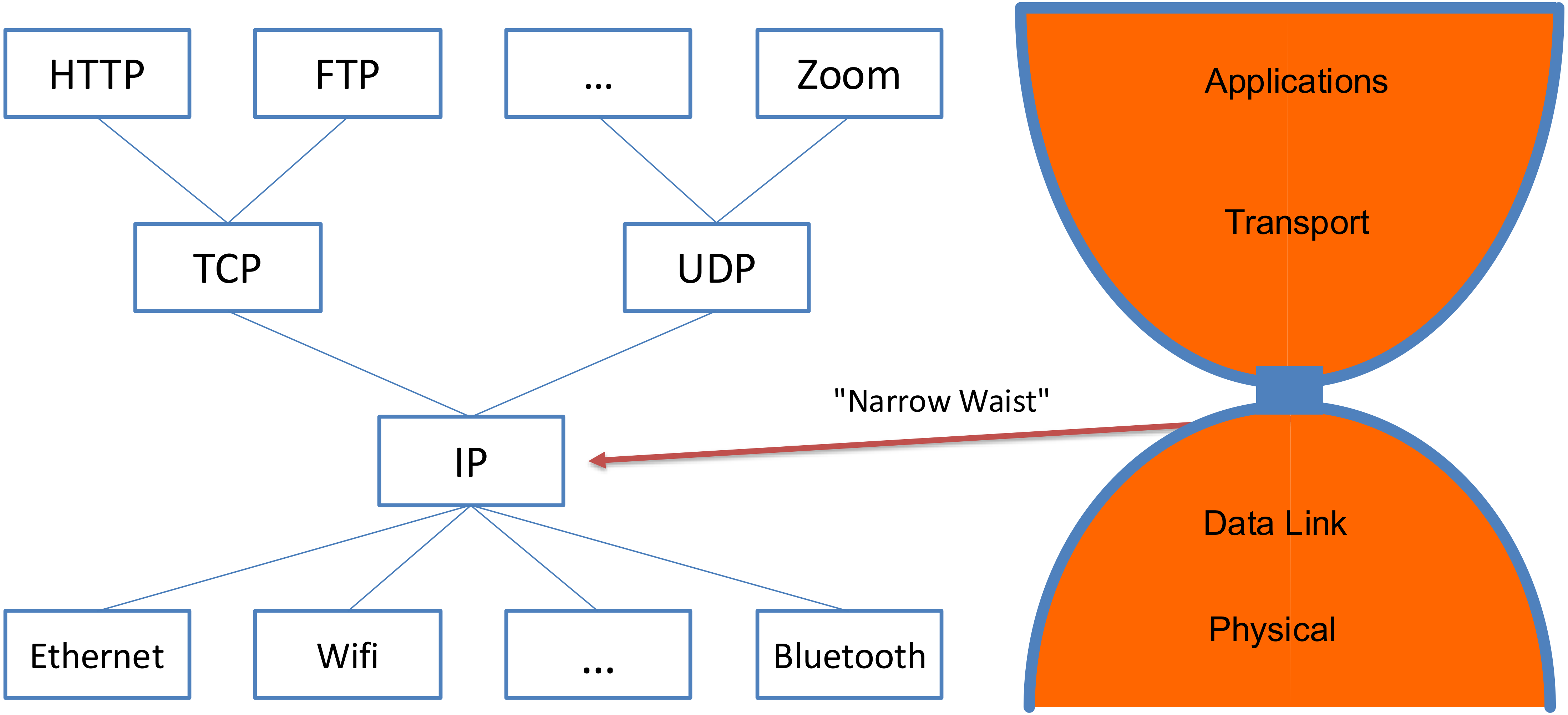
Bluetooth

Internet Protocol Suite



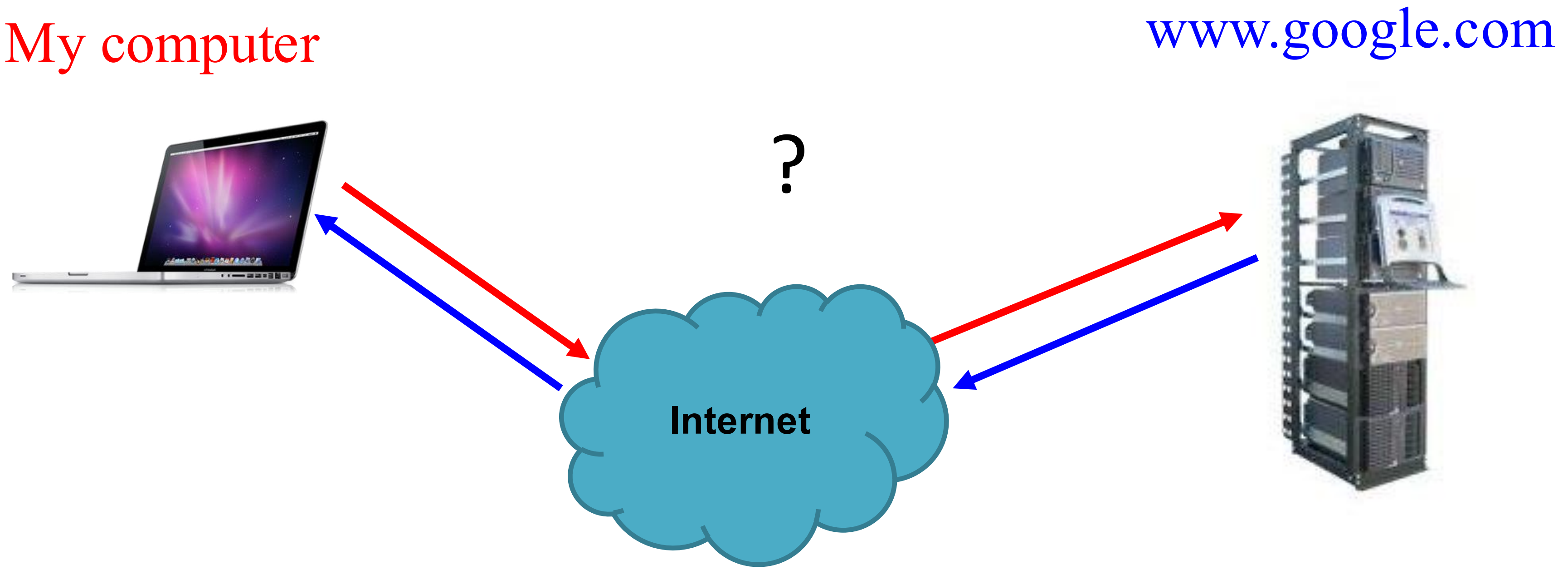


Internet Protocol Suite ("Hourglass model")



Putting this all together

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



My computer

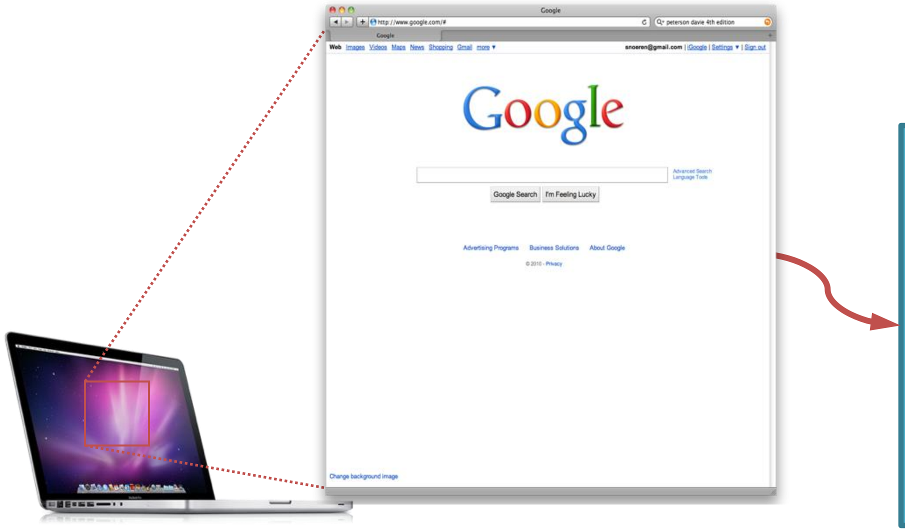
www.google.com

?

Internet

Application Layer: Web request (HTTP)

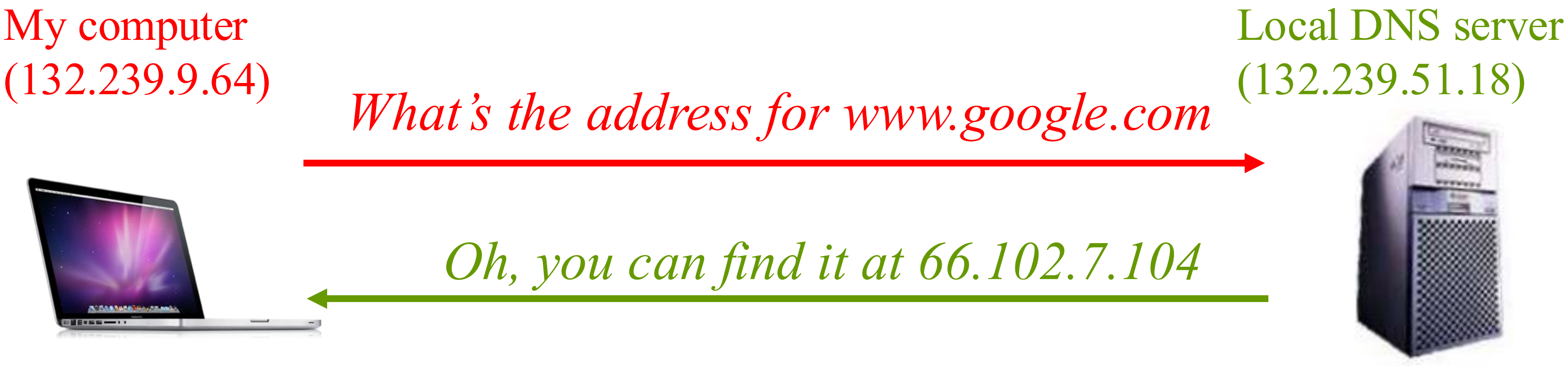
- Turn click into HTTP request



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

Application Layer: Name resolution (DNS)

- Where is www.google.com?



My computer
(132.239.9.64)

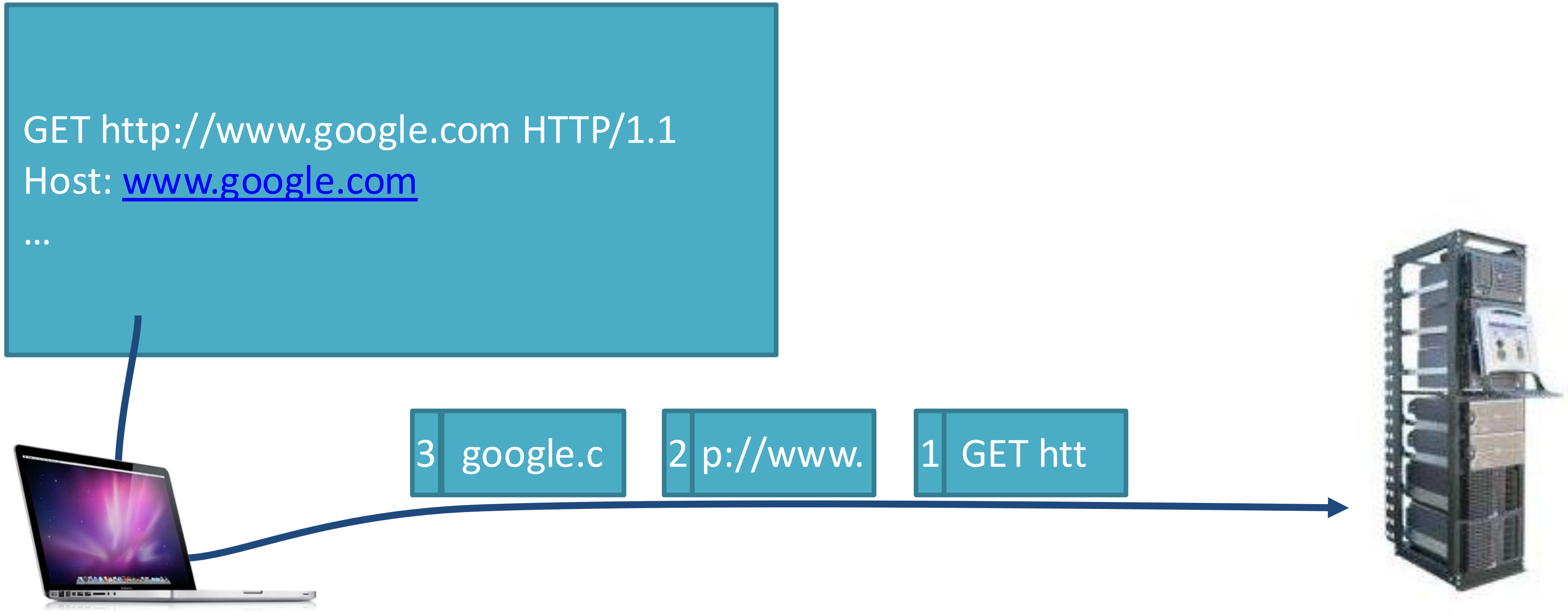
Local DNS server
(132.239.51.18)

What's the address for www.google.com

Oh, you can find it at 66.102.7.104

Transport Layer: TCP

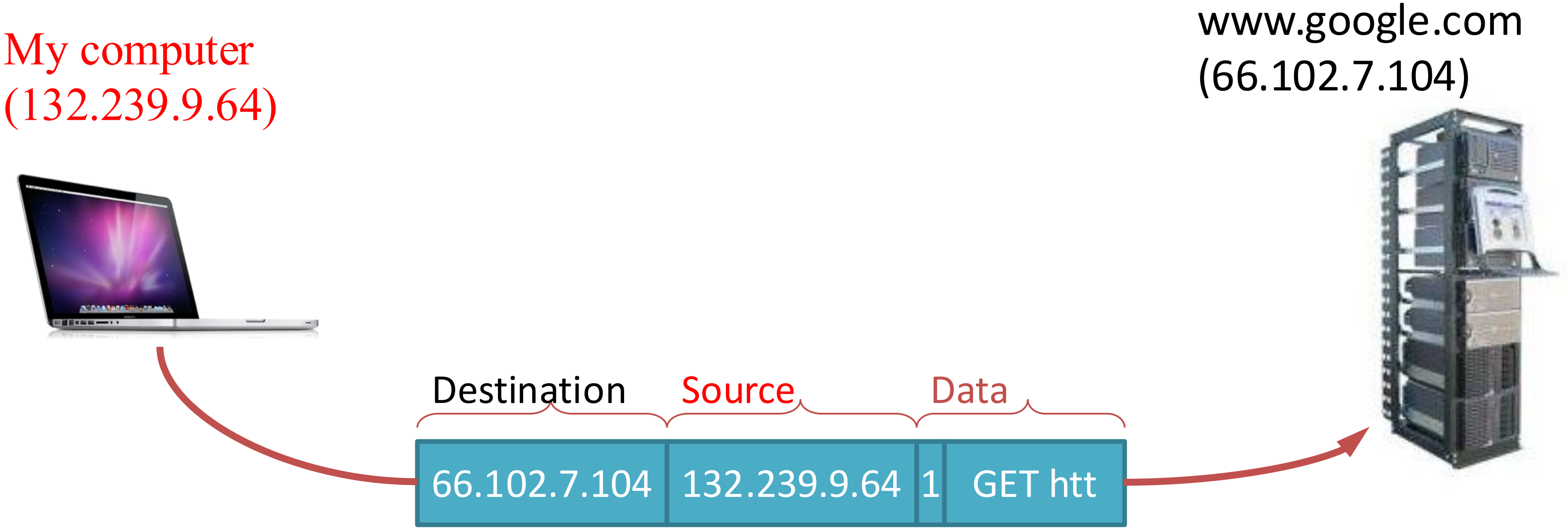
- Break message into packets (TCP segments)
- Should be delivered reliably & in-order



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

Network Layer: Global Network Addressing

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



My computer
(132.239.9.64)

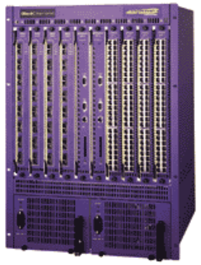
www.google.com
(66.102.7.104)

Destination Source Data

66.102.7.104 | 132.239.9.64 | 1 | GET htt

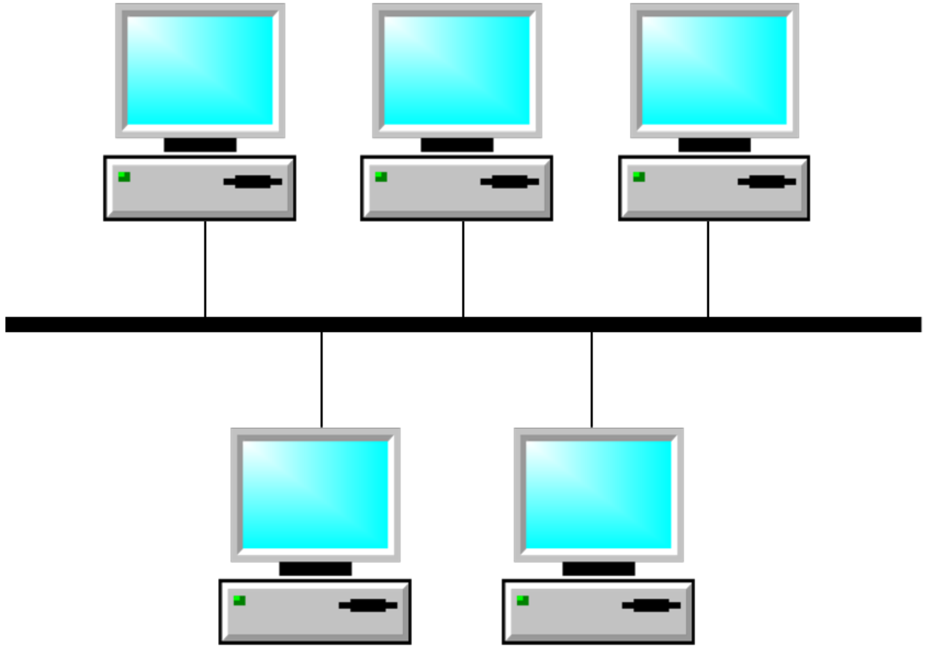
Network Layer: (IP) At Each Router

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

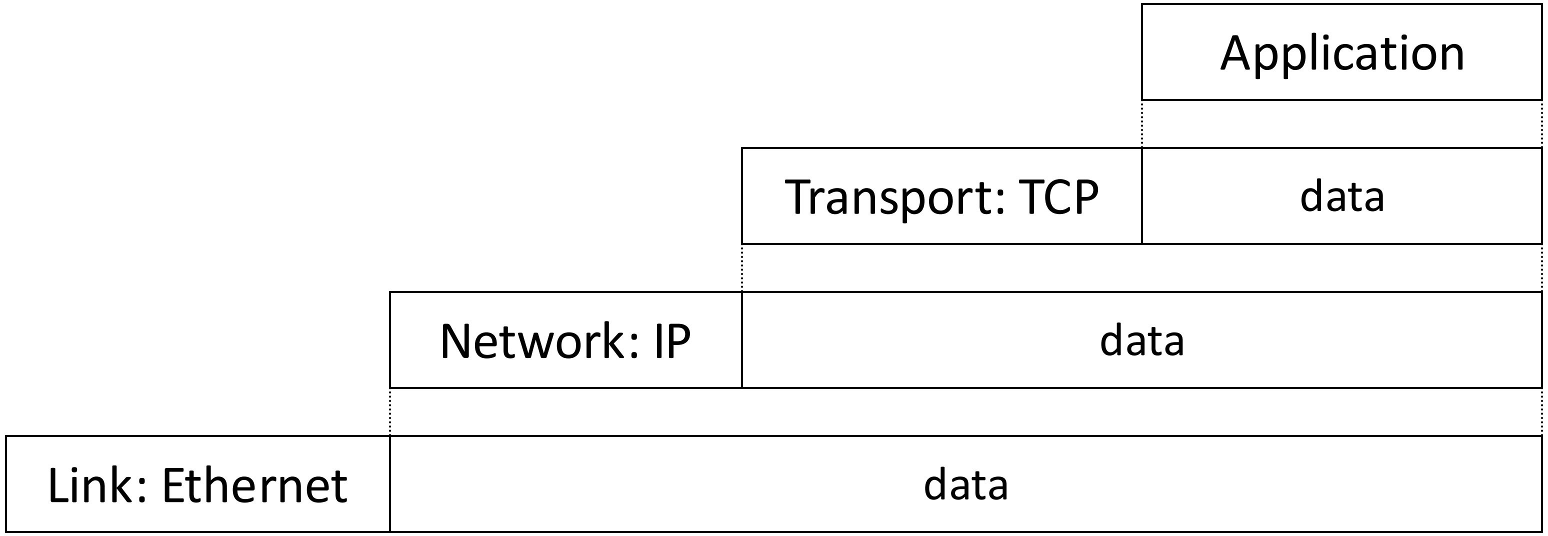


Link & Physical Layers (Ethernet)

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



Message Encapsulation



Application

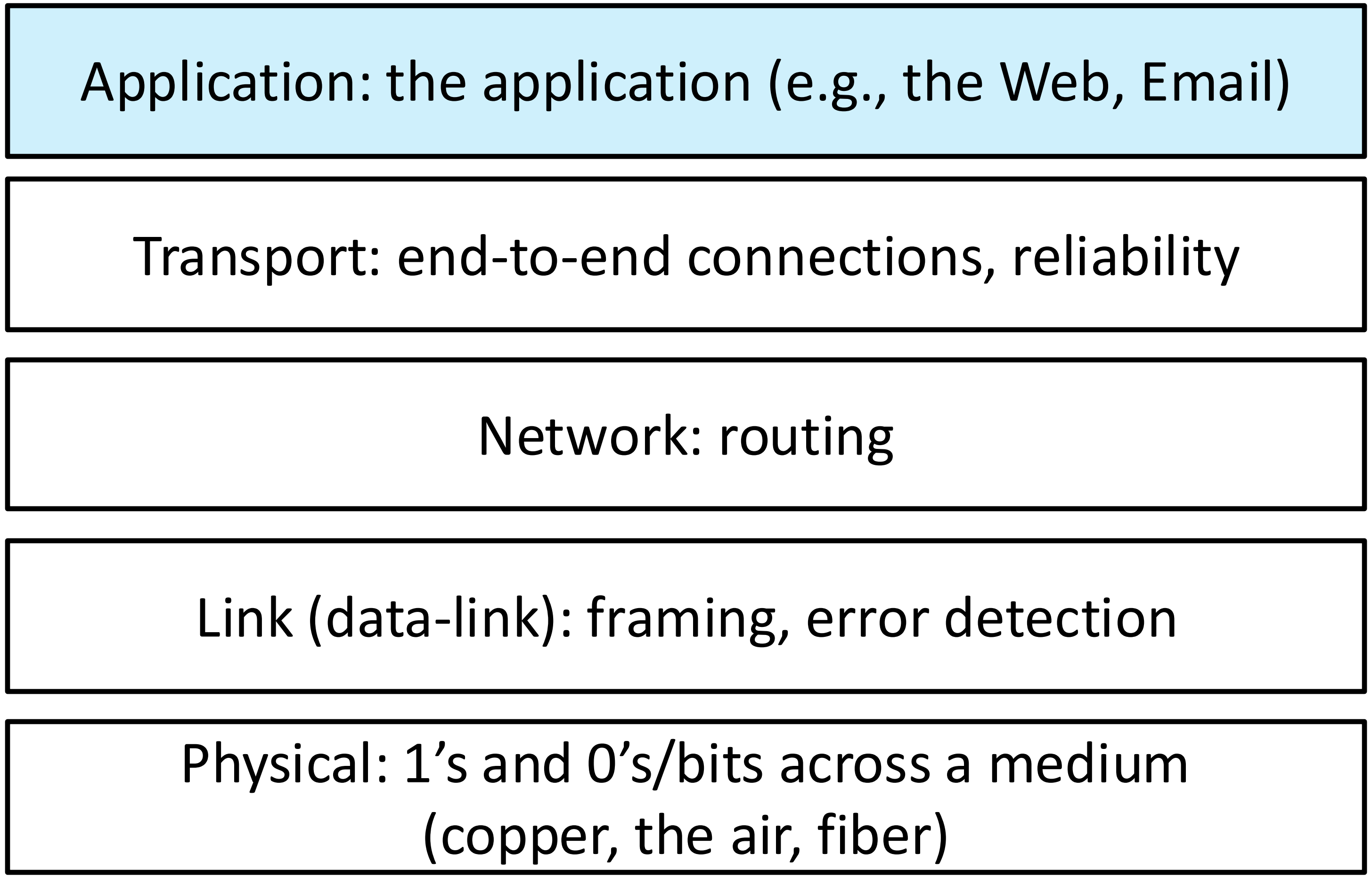
Transport: TCP | data

Network: IP | data

Link: Ethernet | data

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

Five-Layer Internet Model



Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium
(copper, the air, fiber)

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

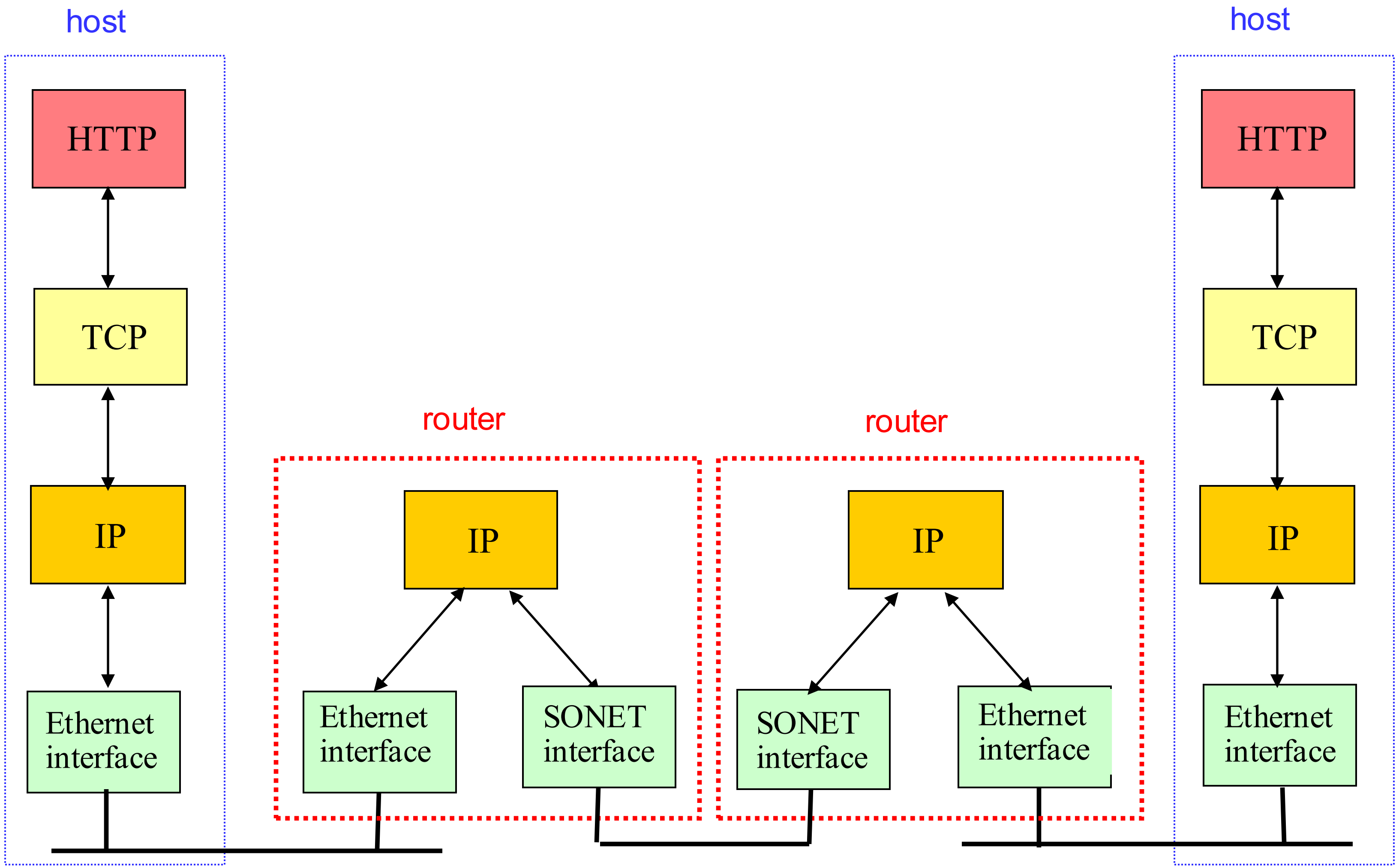
A. All of Them

B. Transport through Physical

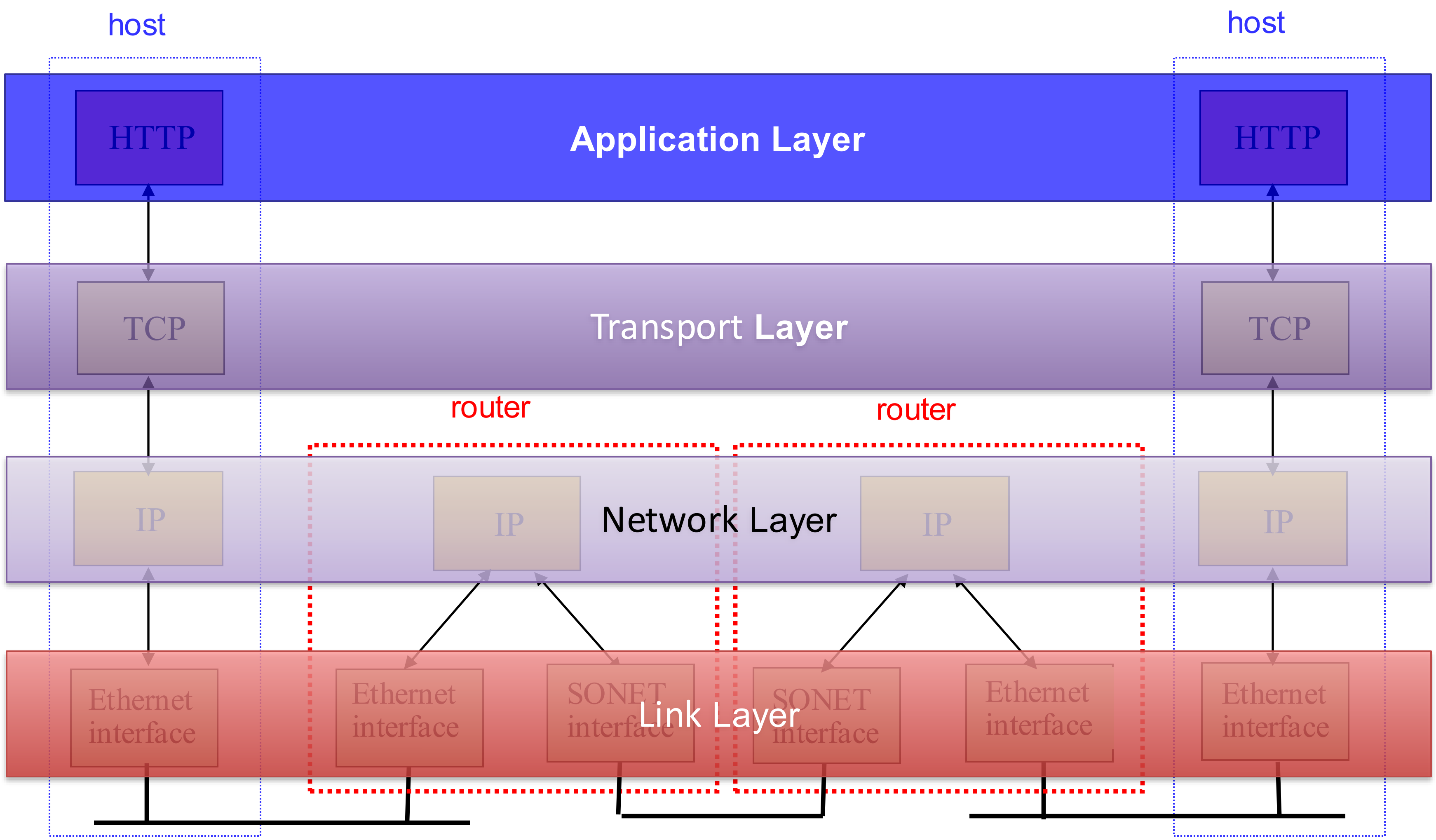
C. Network, Link and Physical

D. Link and Physical

TCP/IP Protocol Stack



TCP/IP Protocol Stack



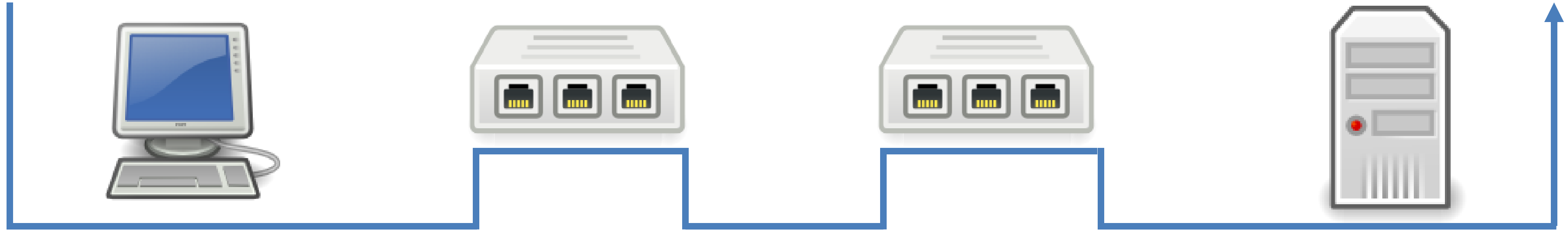
Networks have many concerns, such as reliability, error checking, naming and data ordering. Who/what should be responsible for addressing them? (Why? Which ones belong in which location?)

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

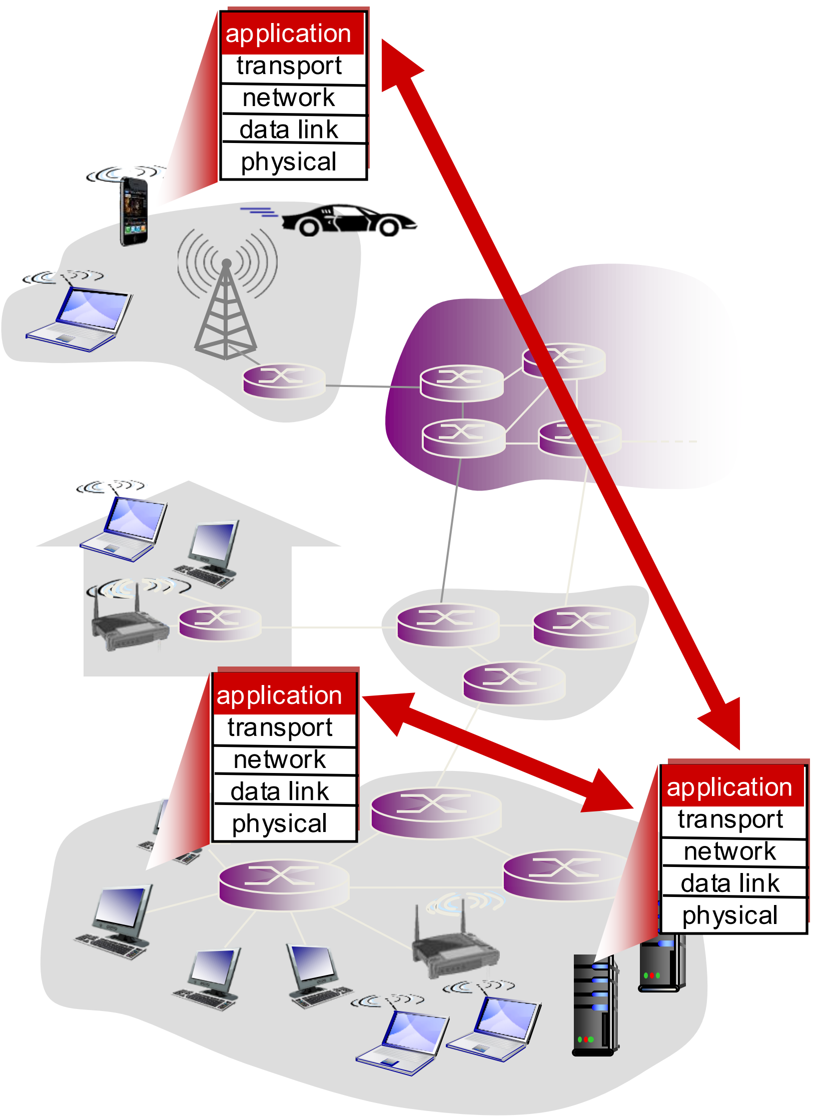


- Don’t provide a function at lower level of abstraction (layer) if you have to do it at higher layer anyway - *unless there is a very good performance reason to do so.*
- Examples: error control, quality of service
- Reference: Saltzer, Reed, Clark, “End-To-End Arguments in System Design,” ACM Transactions on Computer Systems, Vol. 2 (4), 1984.

Creating a network app

write programs that:

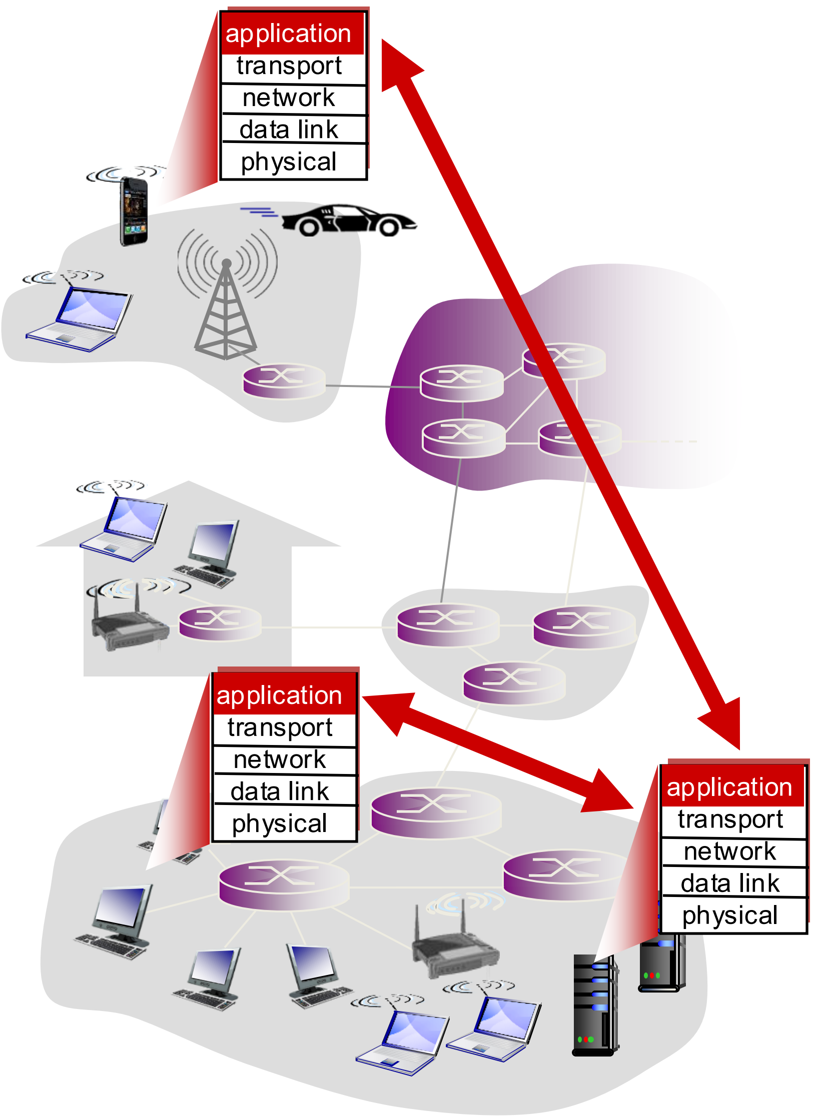
- run on (different) *end systems*
- communicate over network
- e.g., web server s/w communicates with browser software



Creating a network app

no need to write software for network-core devices!

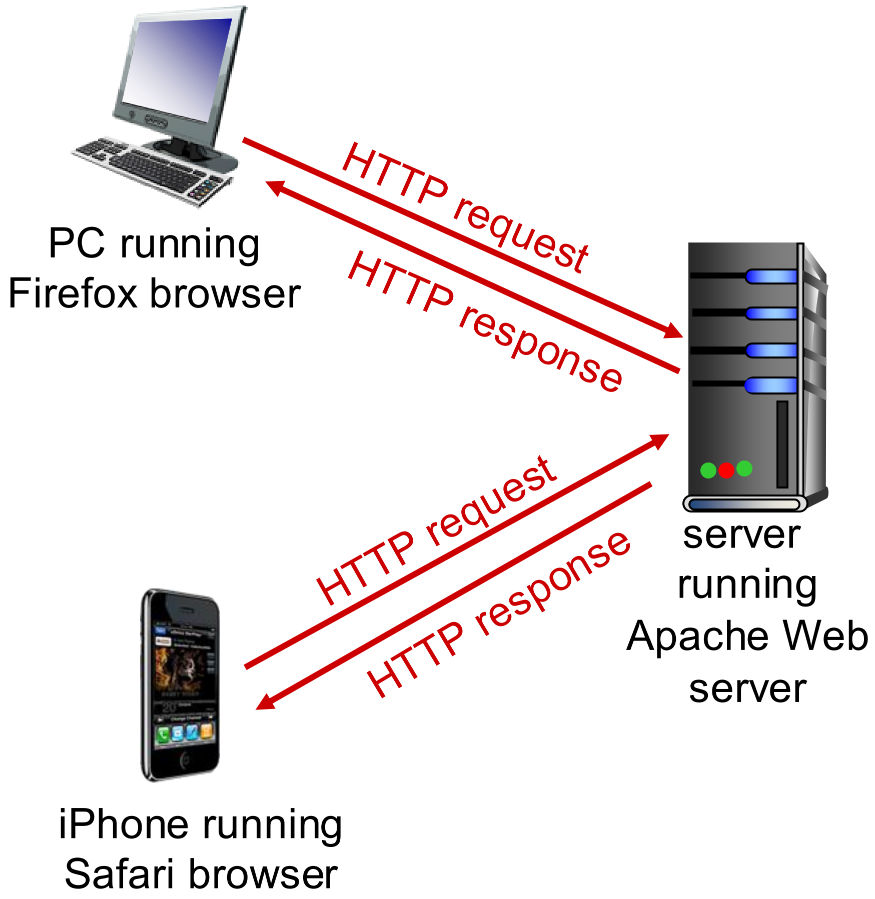
- network-core devices <u>do not run user applications</u>
- applications on end systems
  - rapid app development, propagation



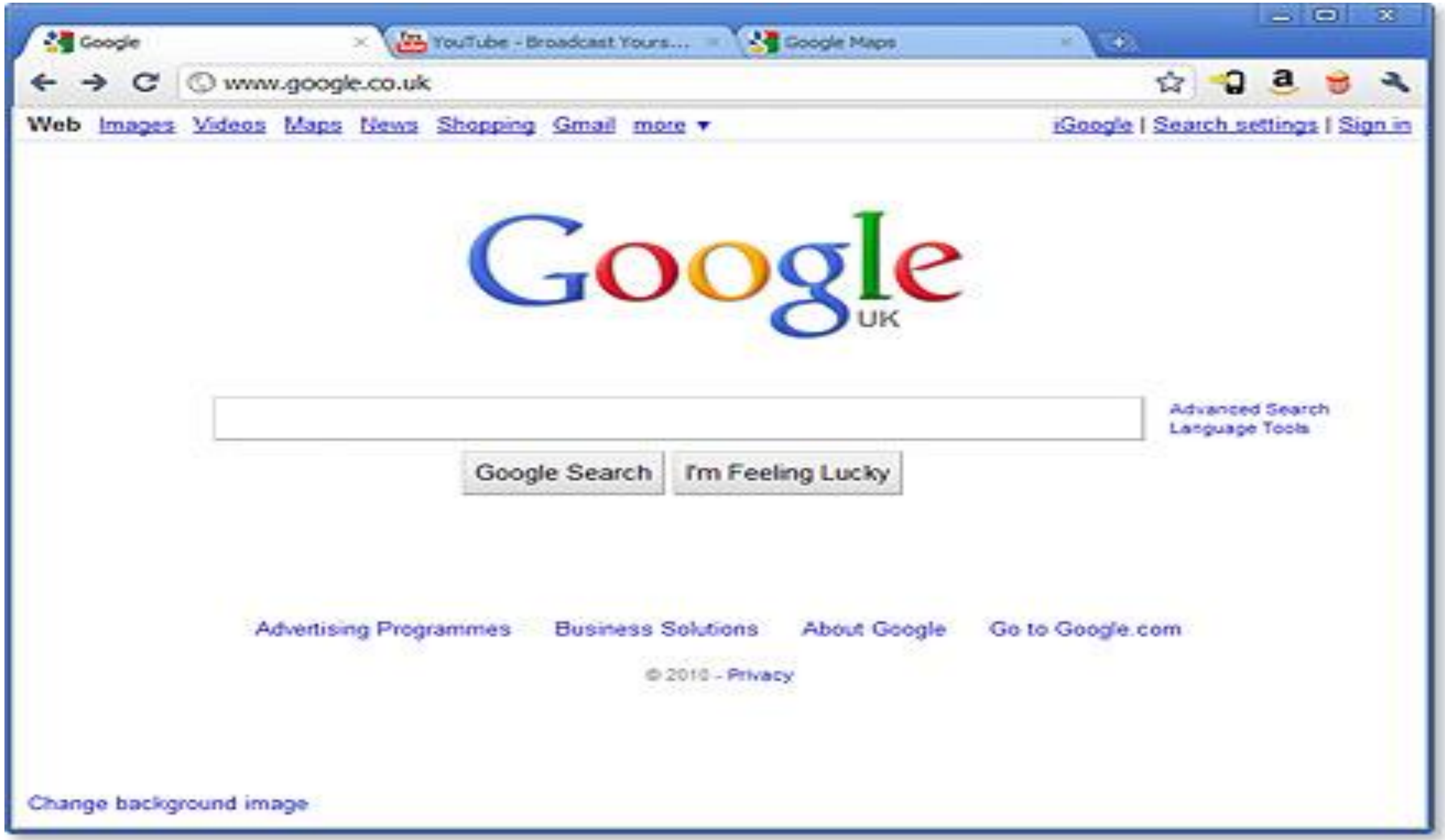
HTTP: HyperText Transfer Protocol

Client/Server model

- client: browser that uses HTTP to request, and receive Web objects.
- server: Web server that uses HTTP to respond with requested object.



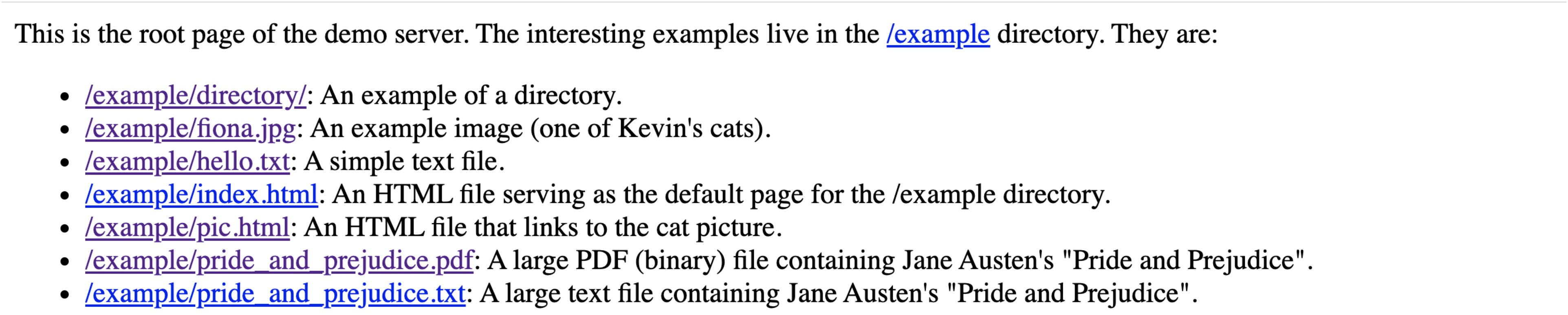
What IS A Web Browser?



HTTP and the Web

- web page consists of objects
- object can be: an HTML file (index.html)

demo.cs.swarthmore.edu/index.html

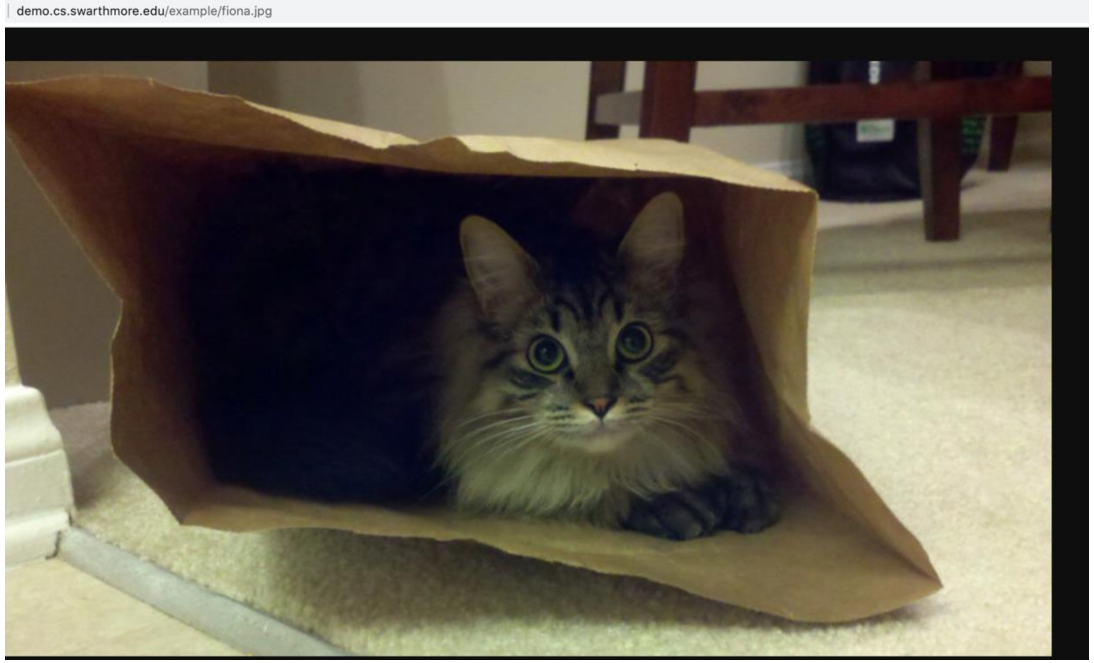


This is the root page of the demo server. The interesting examples live in the /example directory. They are:

- /example/directory/: An example of a directory.
- /example/fiona.jpg: An example image (one of Kevin's cats).
- /example/hello.txt: A simple text file.
- /example/index.html: An HTML file serving as the default page for the /example directory.
- /example/pic.html: An HTML file that links to the cat picture.
- /example/pride_and_prejudice.pdf: A large PDF (binary) file containing Jane Austen's "Pride and Prejudice".
- /example/pride_and_prejudice.txt: A large text file containing Jane Austen's "Pride and Prejudice".

Web objects

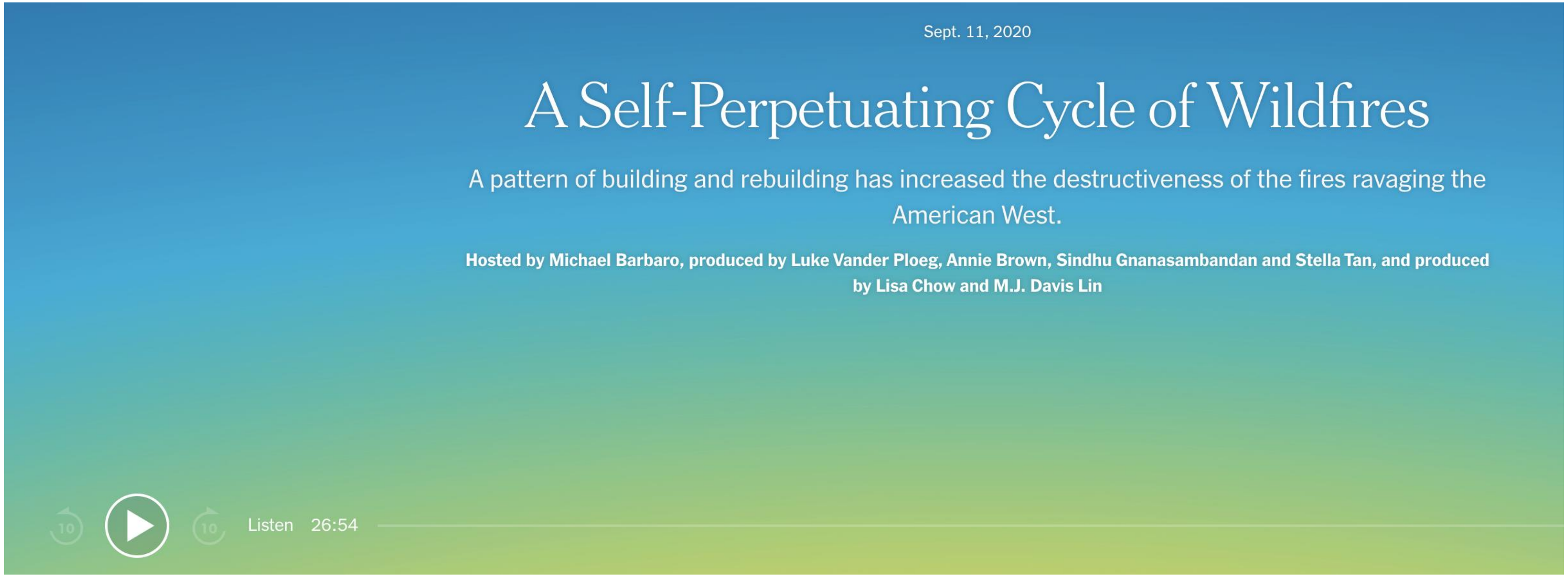
- web page consists of objects
- object can be: JPEG image



demo.cs.swarthmore.edu/example/fiona.jpg

Web objects

- web page consists of objects
- object can be: audio file



Sept. 11, 2020

A Self-Perpetuating Cycle of Wildfires

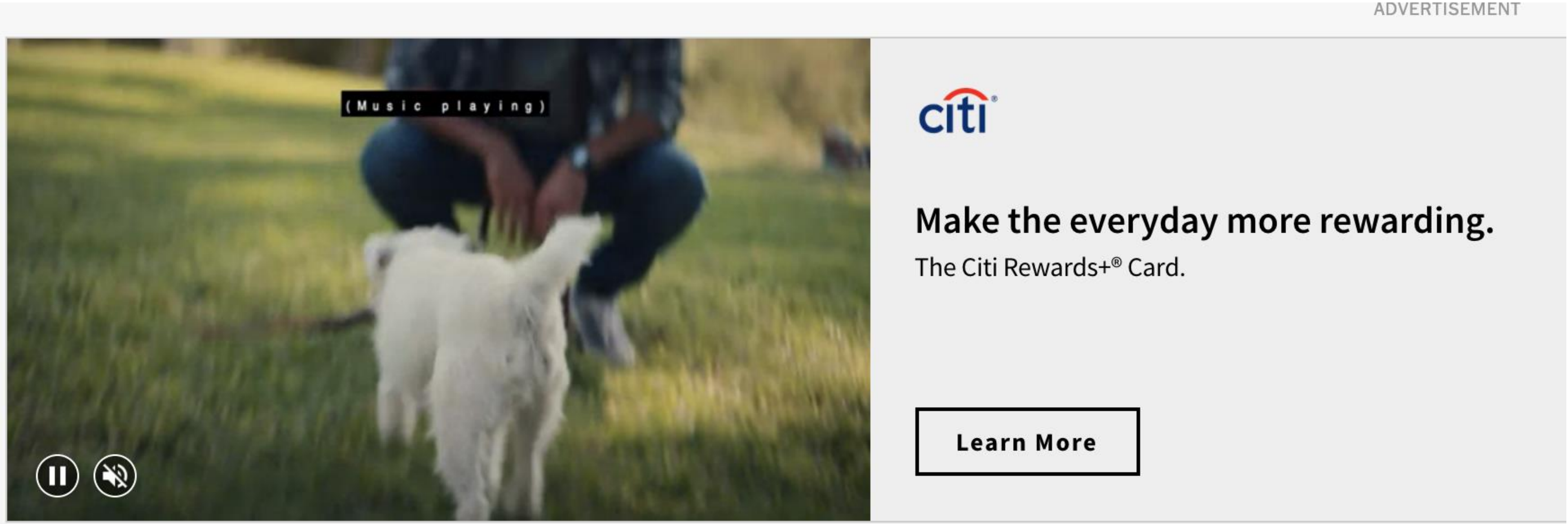
A pattern of building and rebuilding has increased the destructiveness of the fires ravaging the American West.

Hosted by Michael Barbaro, produced by Luke Vander Ploeg, Annie Brown, Sindhu Gnanasambandan and Stella Tan, and produced by Lisa Chow and M.J. Davis Lin

Courtesy: New York Times

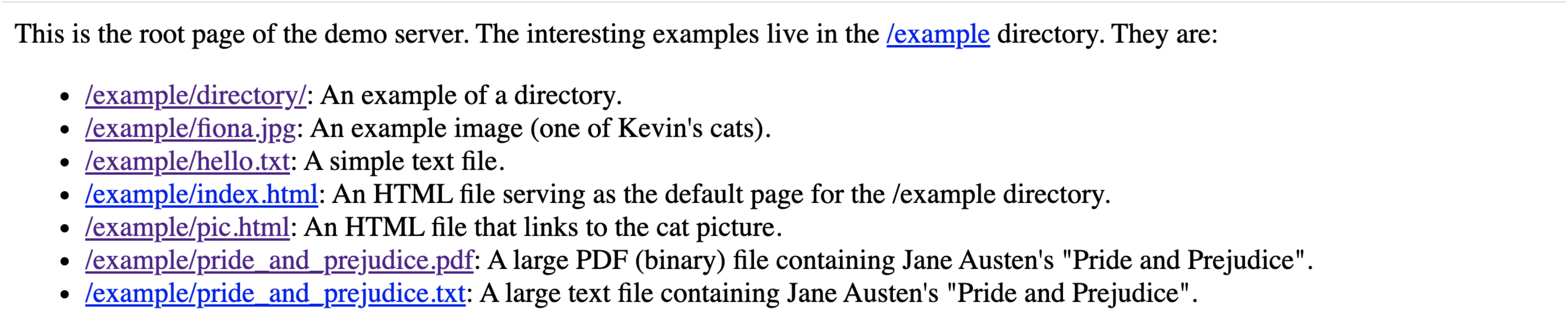
Web objects

- web page consists of objects
- object can be: video, java applets, etc.



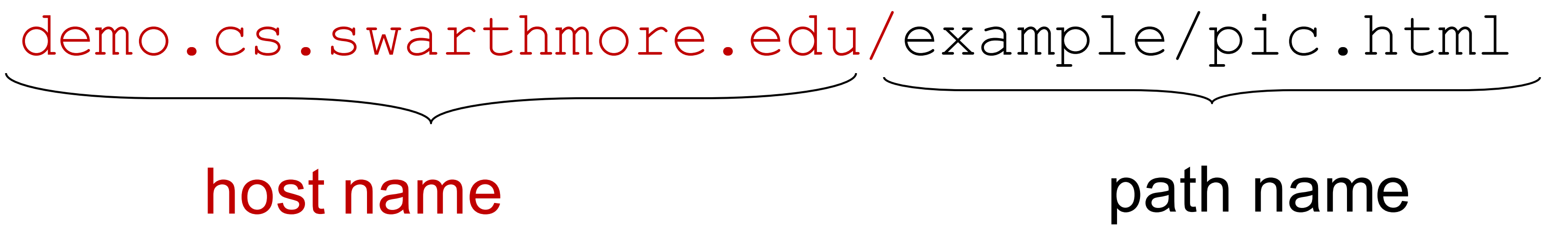
HTTP and the Web

- a web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL, e.g.,

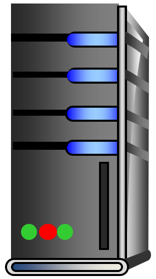


This is the root page of the demo server. The interesting examples live in the /example directory. They are:

- /example/directory/: An example of a directory.
- /example/fiona.jpg: An example image (one of Kevin's cats).
- /example/hello.txt: A simple text file.
- /example/index.html: An HTML file serving as the default page for the /example directory.
- /example/pic.html: An HTML file that links to the cat picture.
- /example/pride_and_prejudice.pdf: A large PDF (binary) file containing Jane Austen's "Pride and Prejudice".
- /example/pride_and_prejudice.txt: A large text file containing Jane Austen's "Pride and Prejudice".



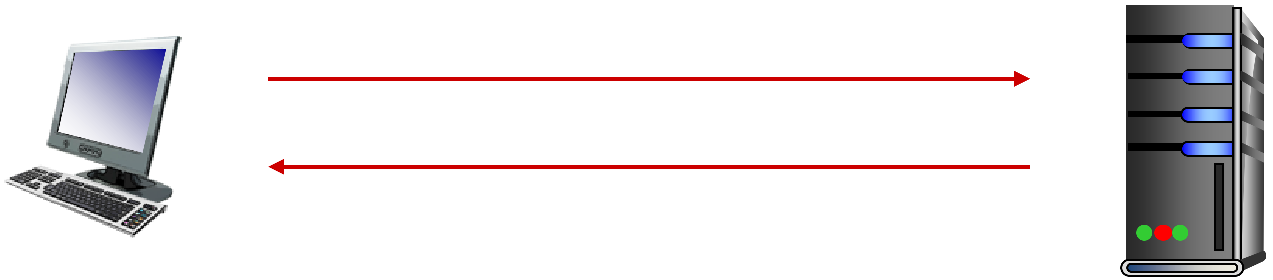
HTTP Overview



1. User types in a URL.

$$\underbrace{\text{http://some.host.name.tld/}}_{\text{host name}}\underbrace{\text{directory/name/file.ext}}_{\text{path name}}$$

HTTP Overview



2. Browser establishes connection with server using the Sockets API.

Calls socket()   // create a socket

Looks up “some.host.name.tld” (DNS: getaddrinfo)

Calls connect() // connect to remote server

Ready to call send() // Can now send HTTP requests

HTTP Overview

3. Browser requests data the user asked for

GET /directory/name/file.ext HTTP/1.0

Host: some.host.name.tld

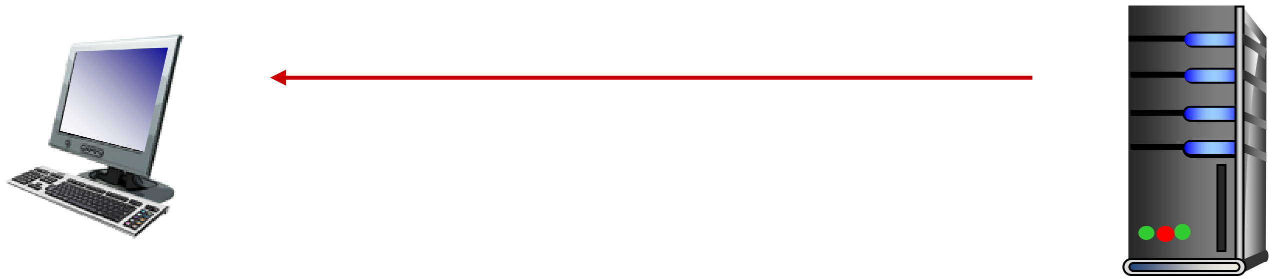
Required fields

[other optional fields, for example:]

User-agent: Mozilla/5.0 (Windows NT 6.1; WOW64)

Accept-language: en

HTTP Overview



4. Server responds with the requested data.

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 1299
Date: Sun, 01 Sep 2013 21:26:38 GMT
[Blank line]
(Data data data data…)
```

HTTP Overview

5. Browser renders the response, fetches any additional objects, and closes the connection.

HTTP Overview

1. User types in a URL.
2. Browser establishes connection with server.
3. Browser requests the corresponding data.
4. Server responds with the requested data.
5. Browser renders the response, fetches other objects, and closes the connection.

It’s a document retrieval system, where documents point to (link to) each other, forming a “web”.

HTTP Overview (Lab 1)

1. User types in a URL.
2. Browser establishes connection with server.
3. Browser requests the corresponding data.
4. Server responds with the requested data.
5. ~~Browser renders the response, fetches other objects,~~ Save the file and close the connection.

It's a document retrieval system, where documents point to (link to) each other, forming a "web".

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet demo.cs.swarthmore.edu 80
```

Opens TCP connection to port 80 (default HTTP server port) at example server.

Anything typed is sent to server on port 80 at demo.cs.swarthmore.edu

Trying out HTTP (client side) for yourself

2. Type in a GET HTTP request:

(Hit carriage
return twice) This
is a minimal, but complete,
GET request to the HTTP
server.

```
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu
(blank line)
```

3. Look at response message sent by HTTP server!

Example

```
$ telnet demo.cs.swarthmore.edu 80
Trying 130.58.68.26...
Connected to demo.cs.swarthmore.edu.
Escape character is '^]'.
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

HTTP/1.1 200 OK
Vary: Accept-Encoding
Content-Type: text/html
Accept-Ranges: bytes
ETag: "316912886"
Last-Modified: Wed, 04 Jan 2017 17:47:31 GMT
Content-Length: 1062
Date: Wed, 05 Sep 2018 17:27:34 GMT
Server: lighttpd/1.4.35
```

Response headers

Example

```
$ telnet demo.cs.swarthmore.edu 80
Trying 130.58.68.26...
Connected to demo.cs.swarthmore.edu.
Escape character is '^]'.
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu
```

Response headers

```
<html><head><title>Demo Server</title></head>
<body>
.....
</body>
</html>
```

Response body (This is what you should be saving in lab 1.)

Stuff for Monday Sep 8