CS 43: Computer Networks

02: Protocols & Layering

September 5, 2018

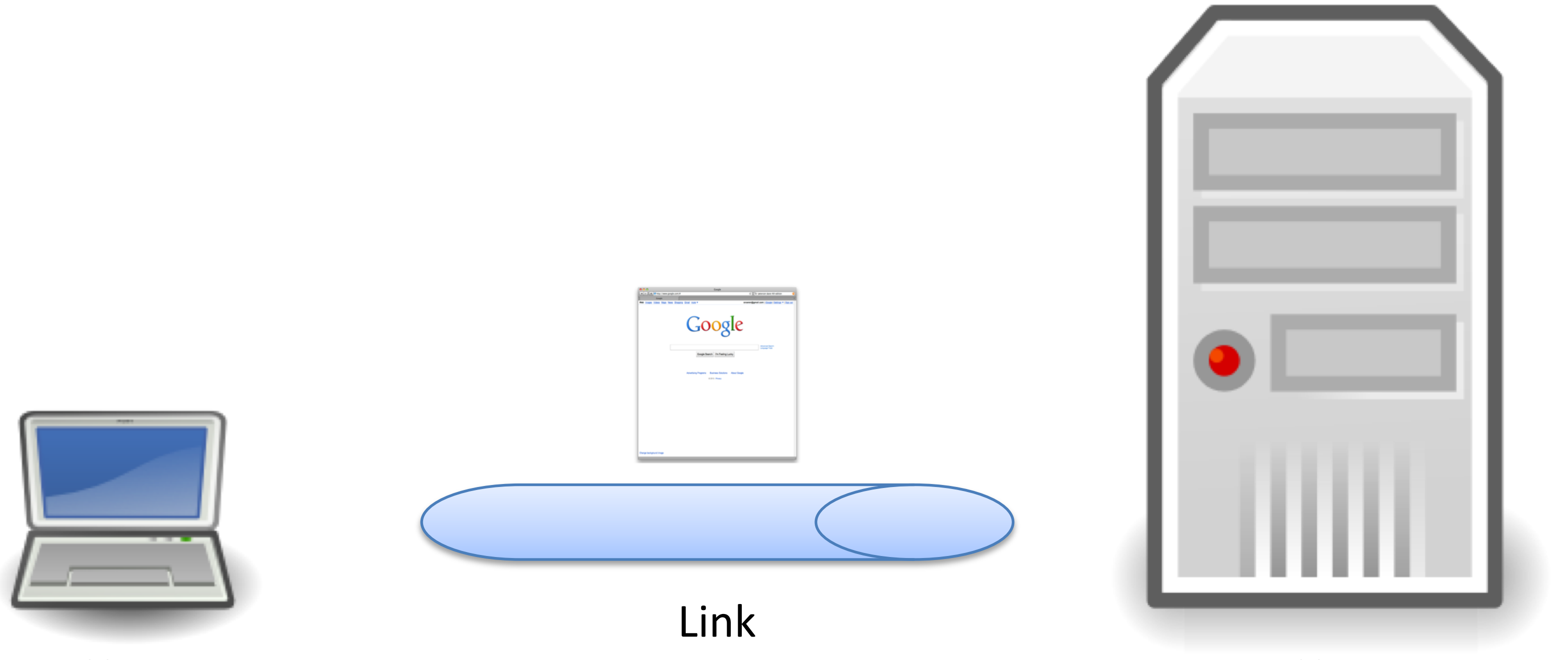
SWARTHMORE COLLEGE

Announcements

- Please fill in choice lab section and conflicts if you have any with either of the lab sections.
- Choose your lab partner on Piazza for lab-2!

Last Class:
Send information from one host to another

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



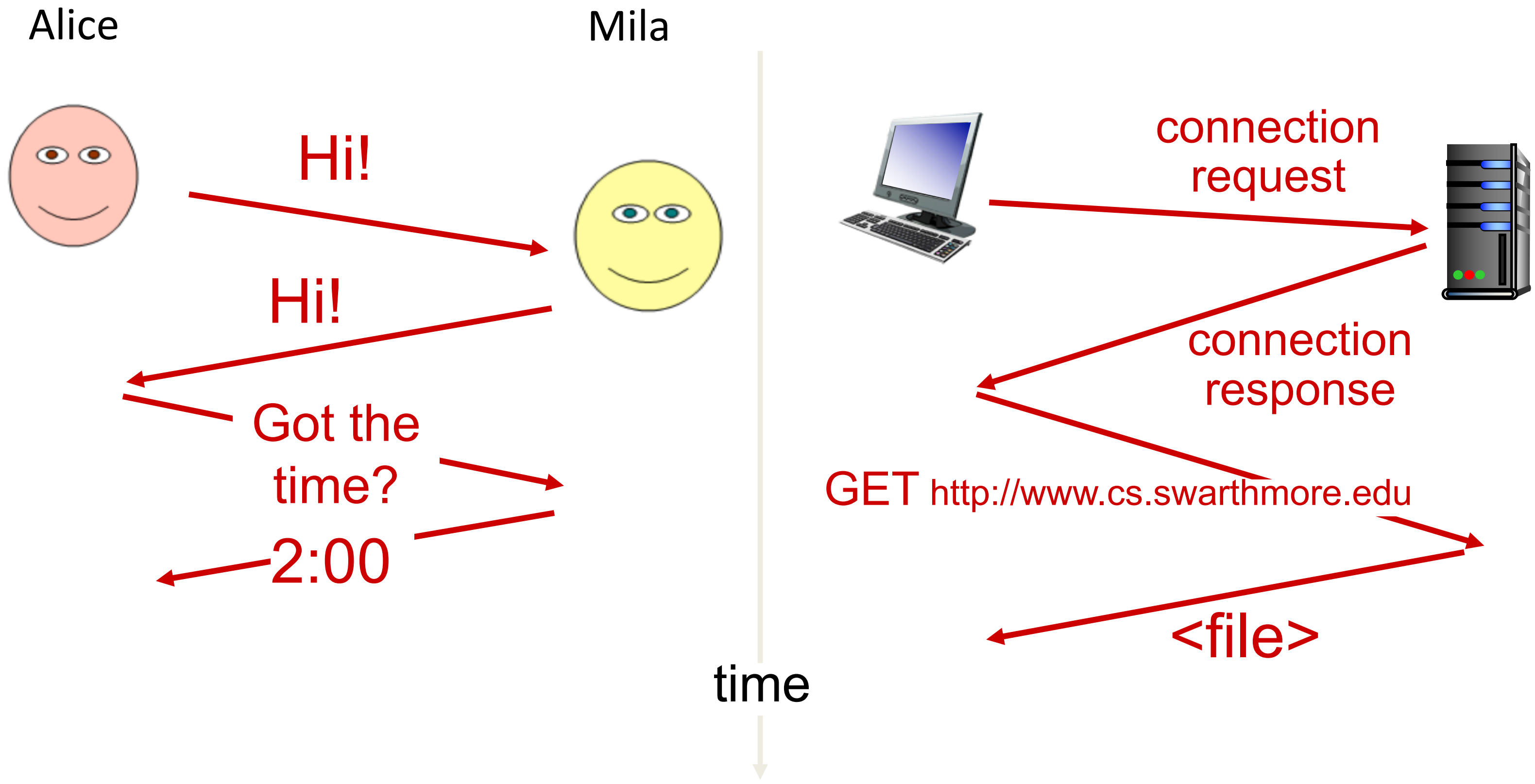
Host
(PC)

Link

Host
(Server)

What is a protocol?

Protocol: message format + transfer procedure



Alice

Mila

Hi!

Hi!

Got the time?

2:00

time

connection request

connection response

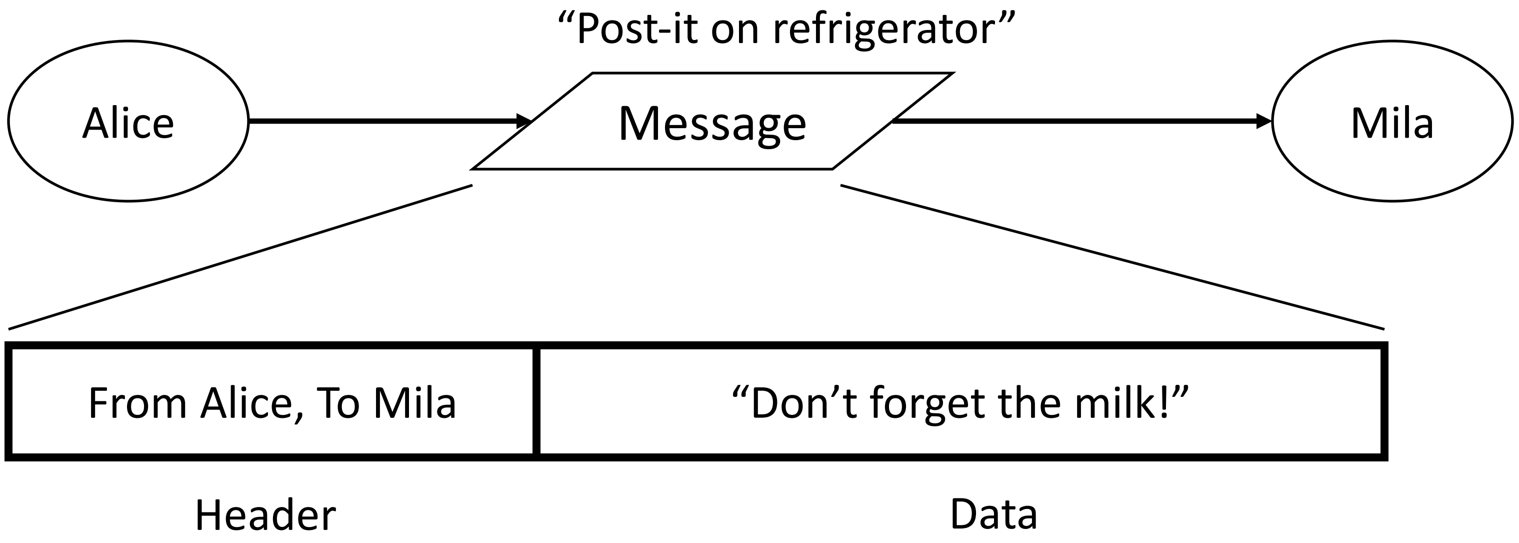
GET http://www.cs.swarthmore.edu

<file>

Human Protocol

Network Protocols
(defined in RFCs)

A “Simple” analogous task: Post-it Note



“Post-it on refrigerator”

Alice

Message

Mila

From Alice, To Mila

“Don’t forget the milk!”

Header

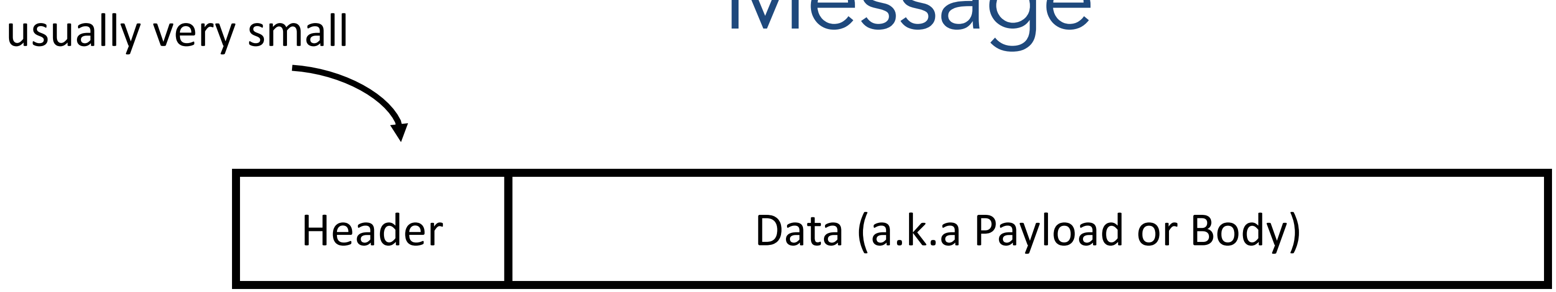
Data

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

Message



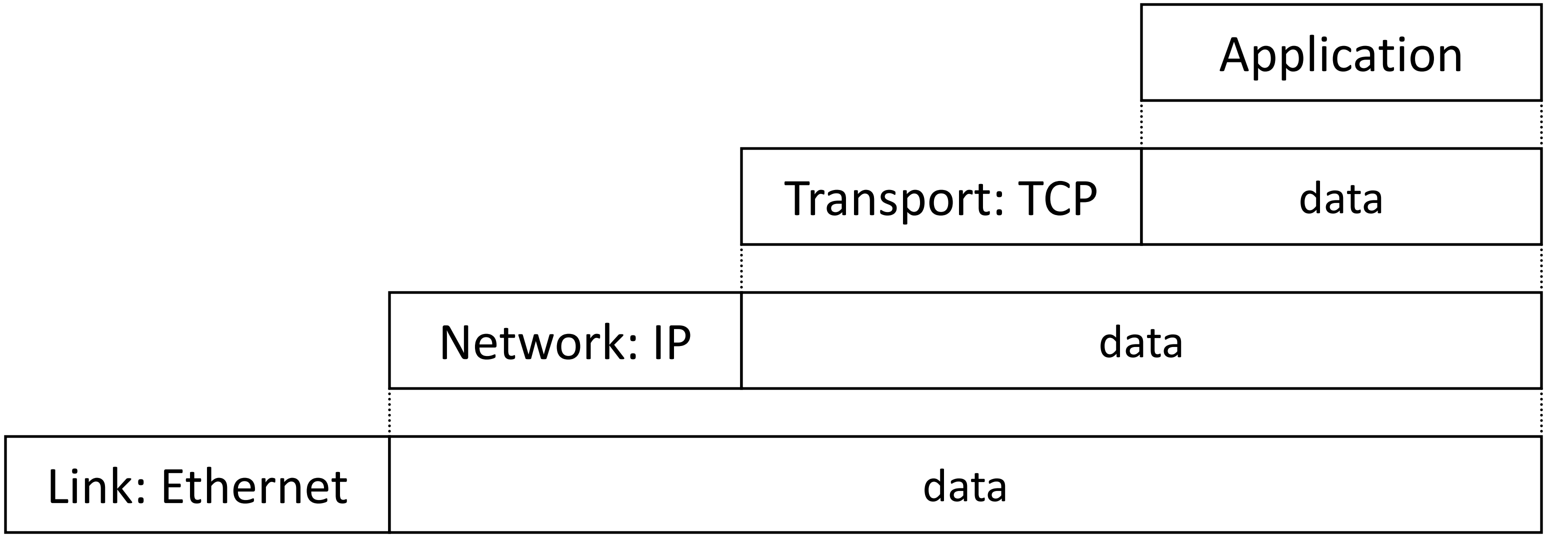
usually very small

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)

Message Encapsulation



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

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 effect delivery of letter from Alice to Mila

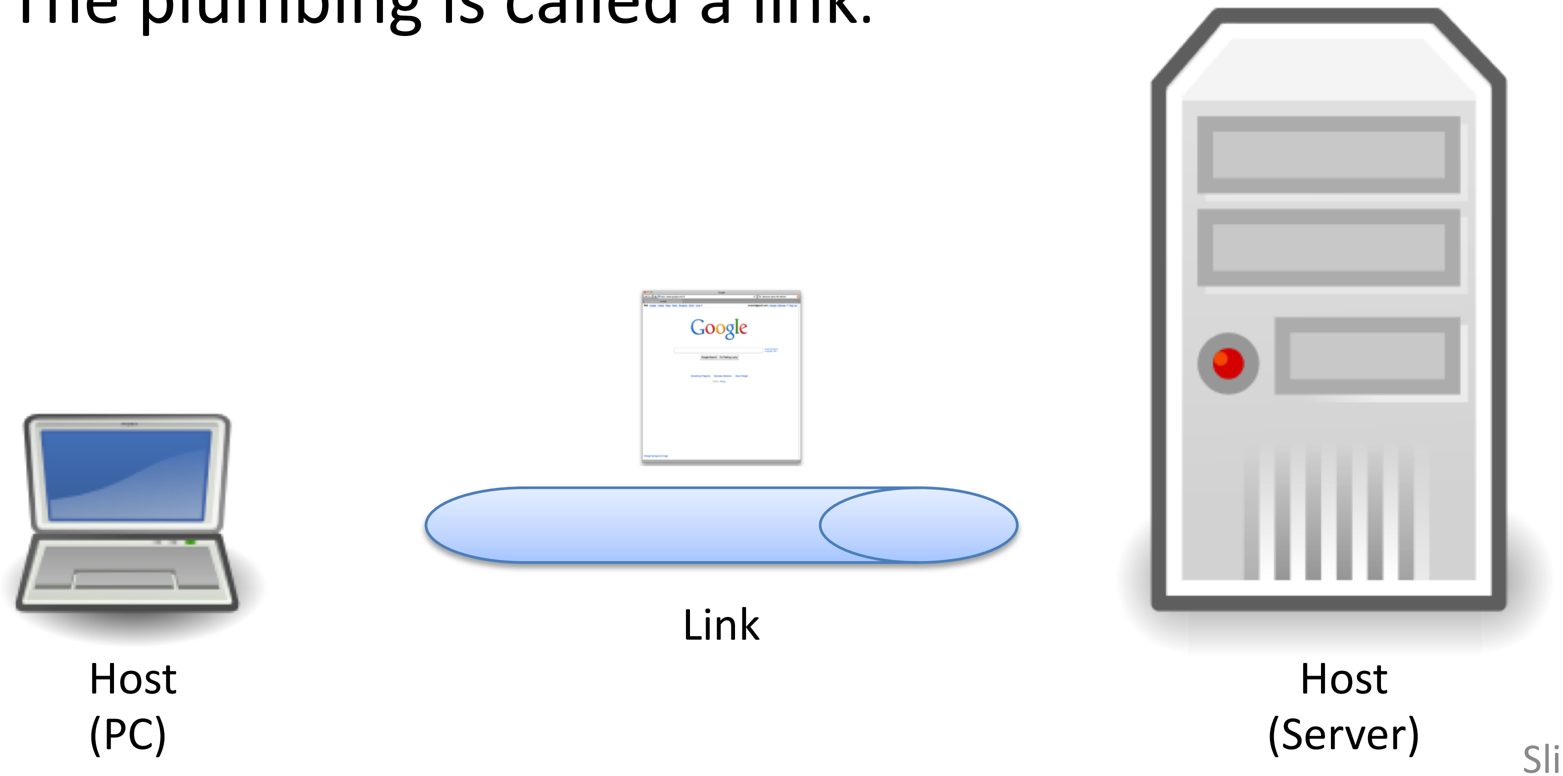
Abstraction!

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

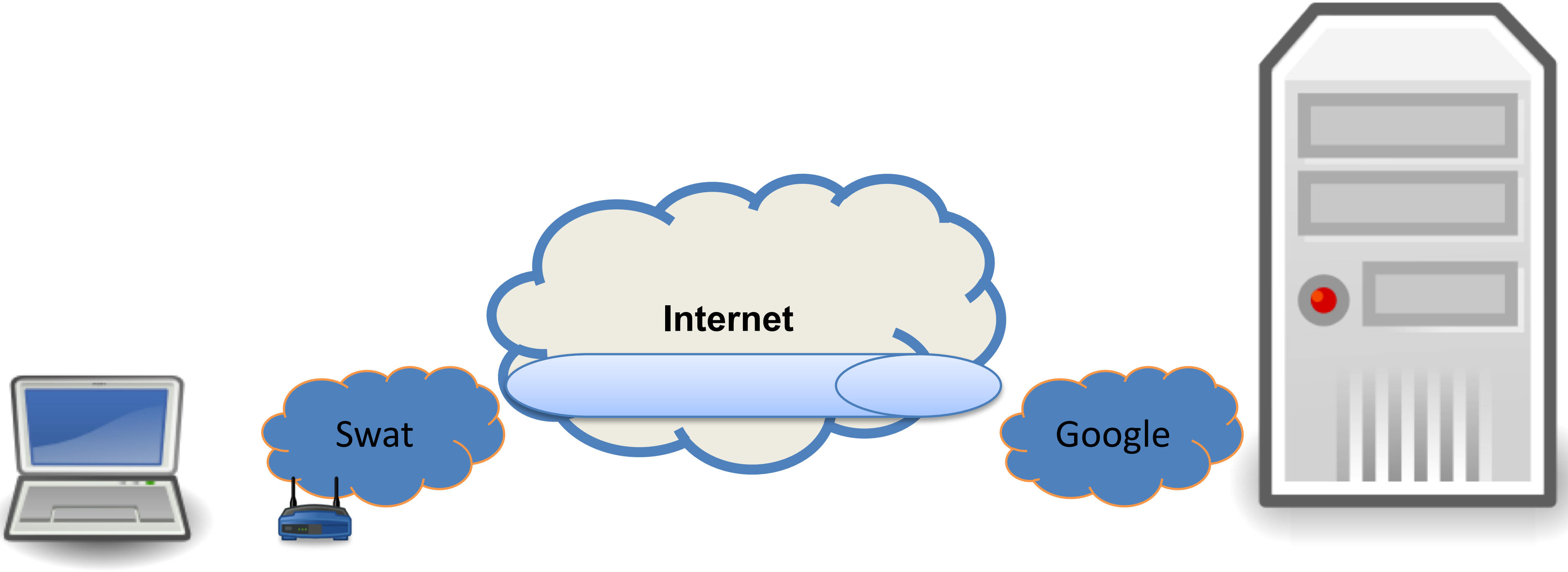
A "Simple" Task

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

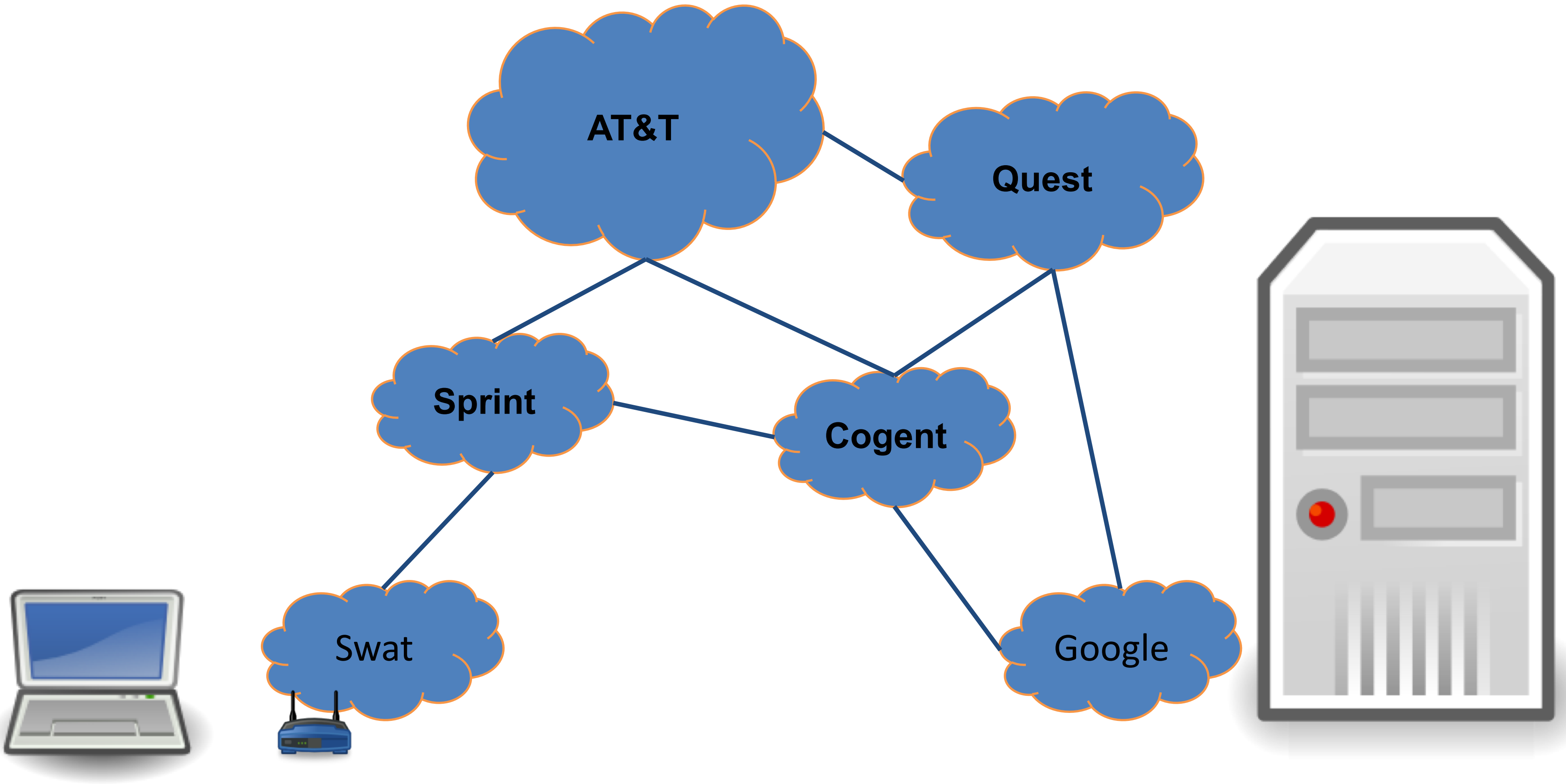
- hosts: endpoints of a network
- The plumbing is called a link.



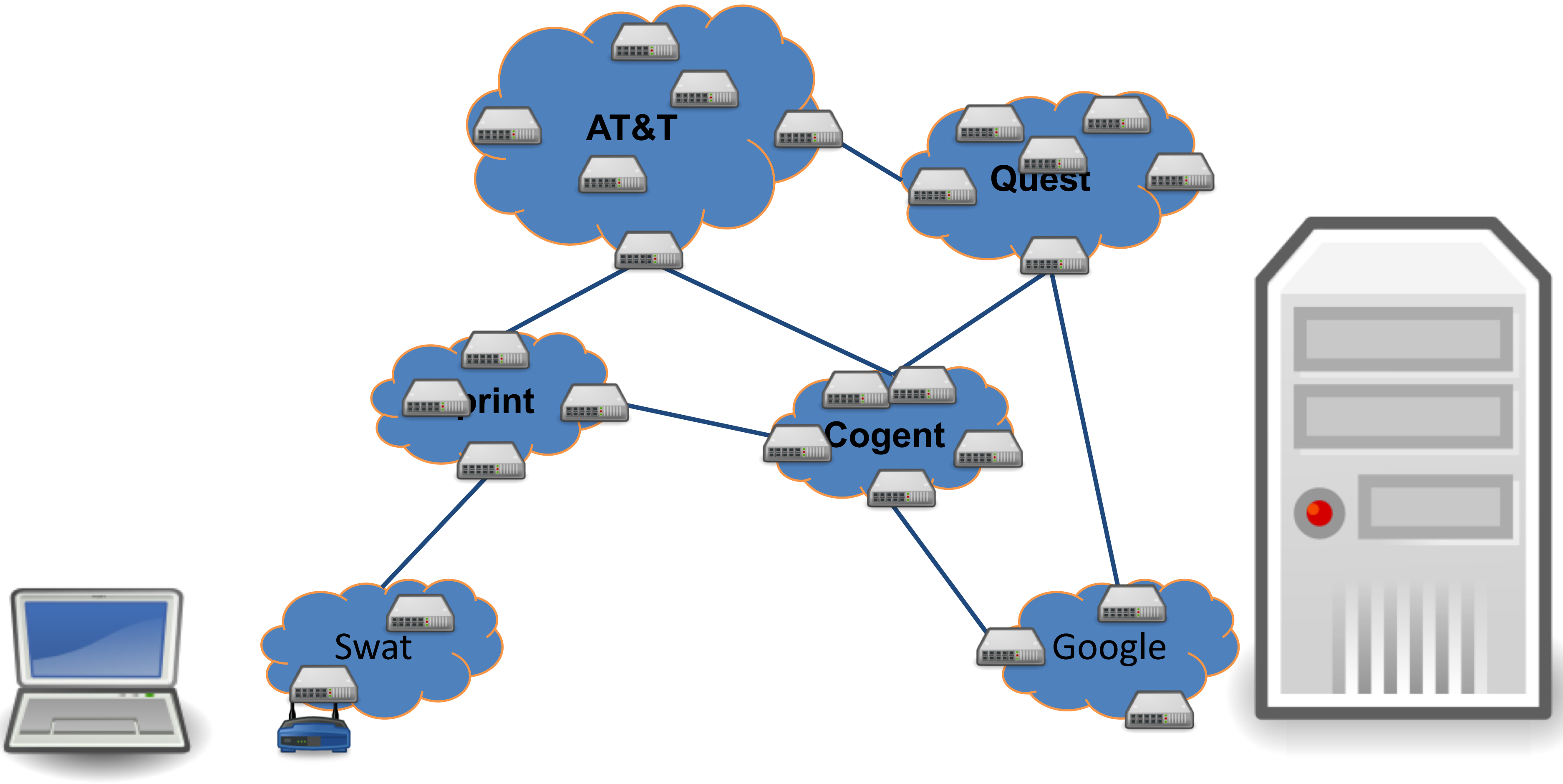
Not Really So Simple…



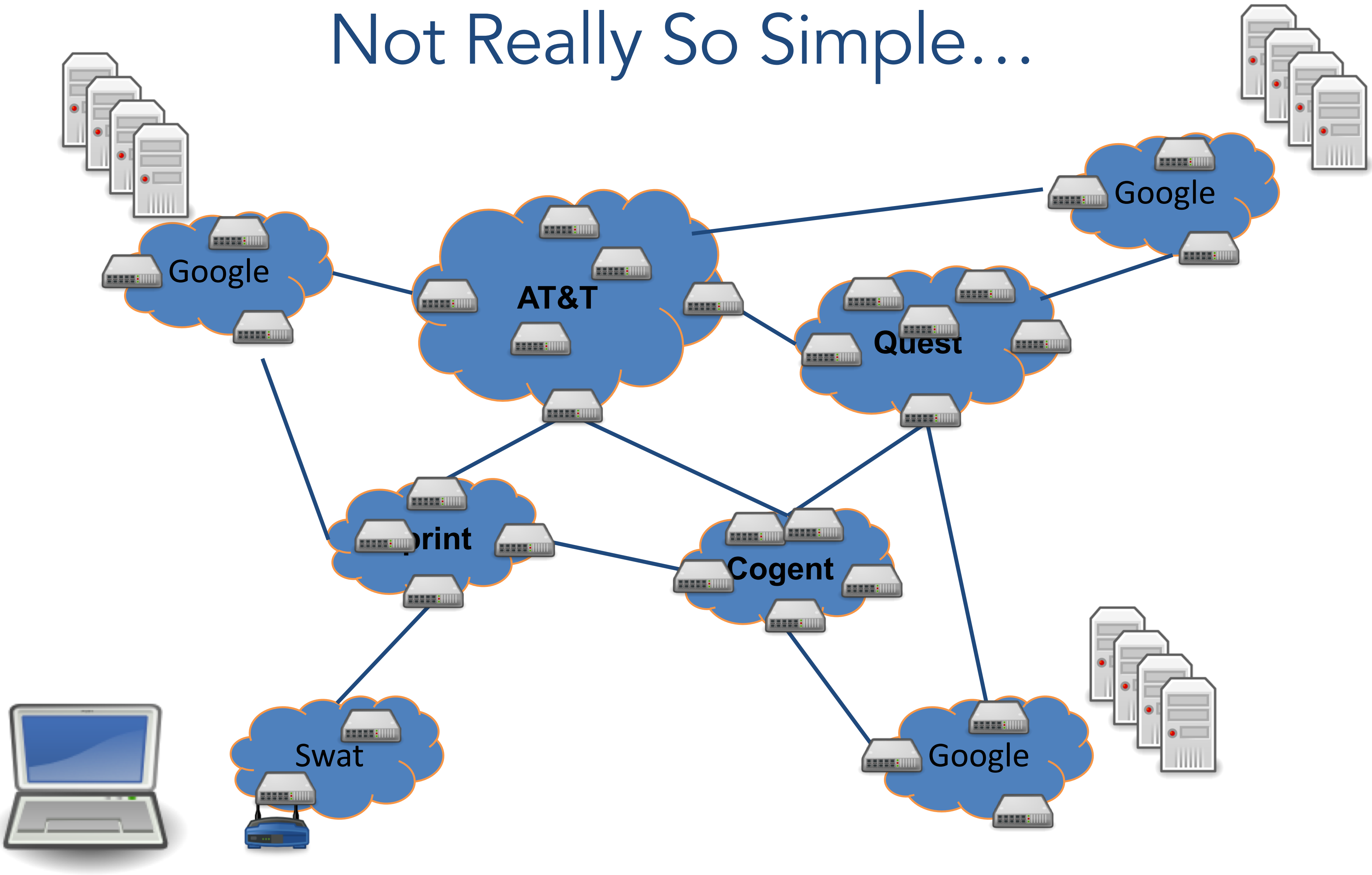
Not Really So Simple…



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.)

Today

- Whose primary responsibility is it to deliver packets? (a.k.a the End-to-end argument).
- OSI Model and Layering
- Application layer protocols: HTTP

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, and data ordering. Who/what should be responsible for addressing them? (Why?)

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.

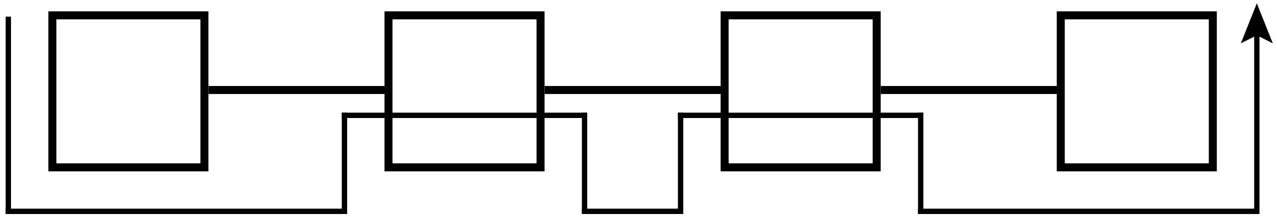
Networks have many concerns, such as reliability, error checking, and data ordering. Who/what should be responsible for addressing them? (Why?)

A. The network should take care of these for us.

B. The communicating hosts handle these in the Internet

C. Some other entity should solve these problems.

The "End-to-End" Argument



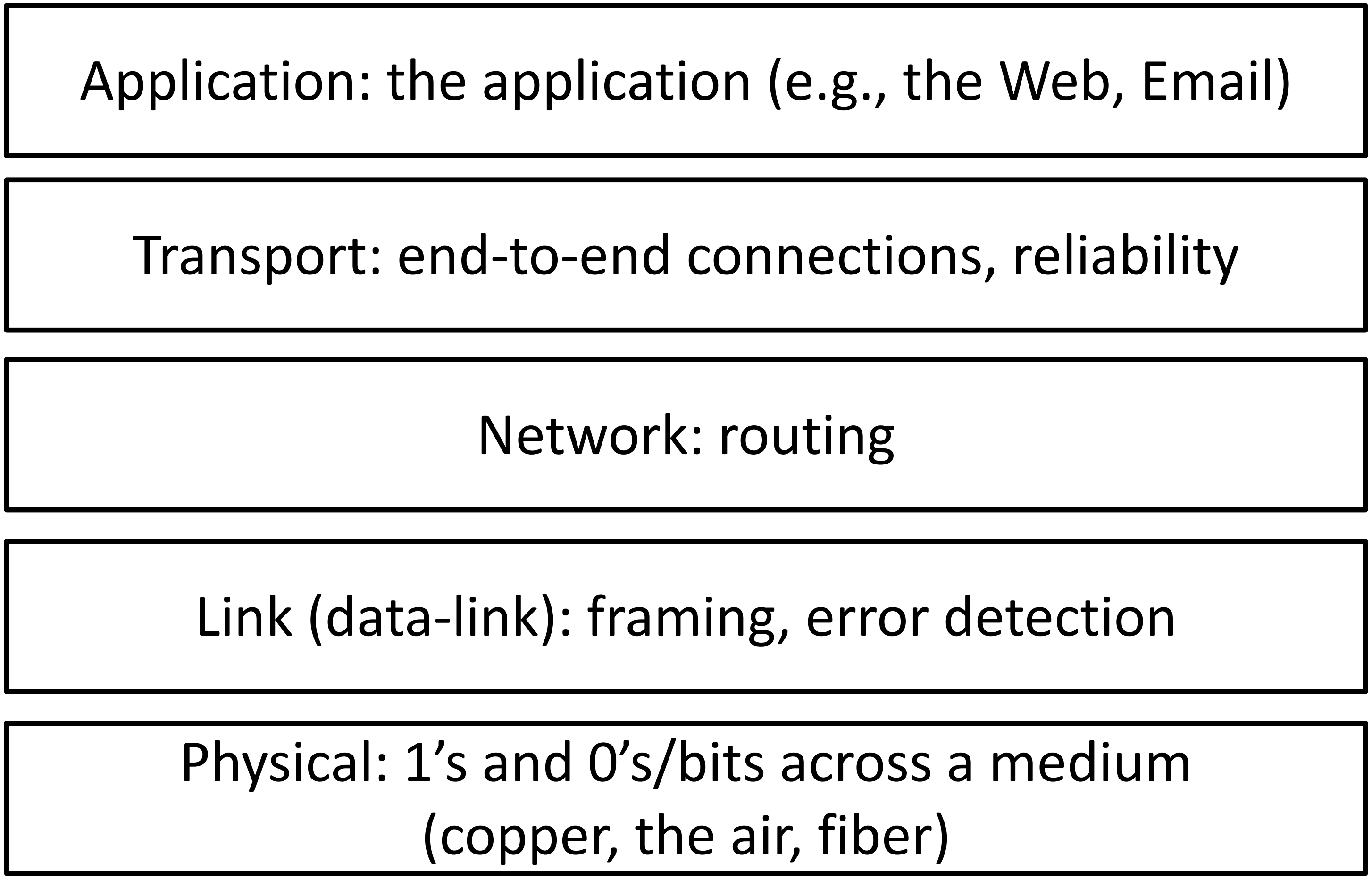
Don't provide a function at lower layer if you have to do it at higher layer anyway ...

... 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), pp. 277-288, 1984.

Five-Layer Internet Model



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

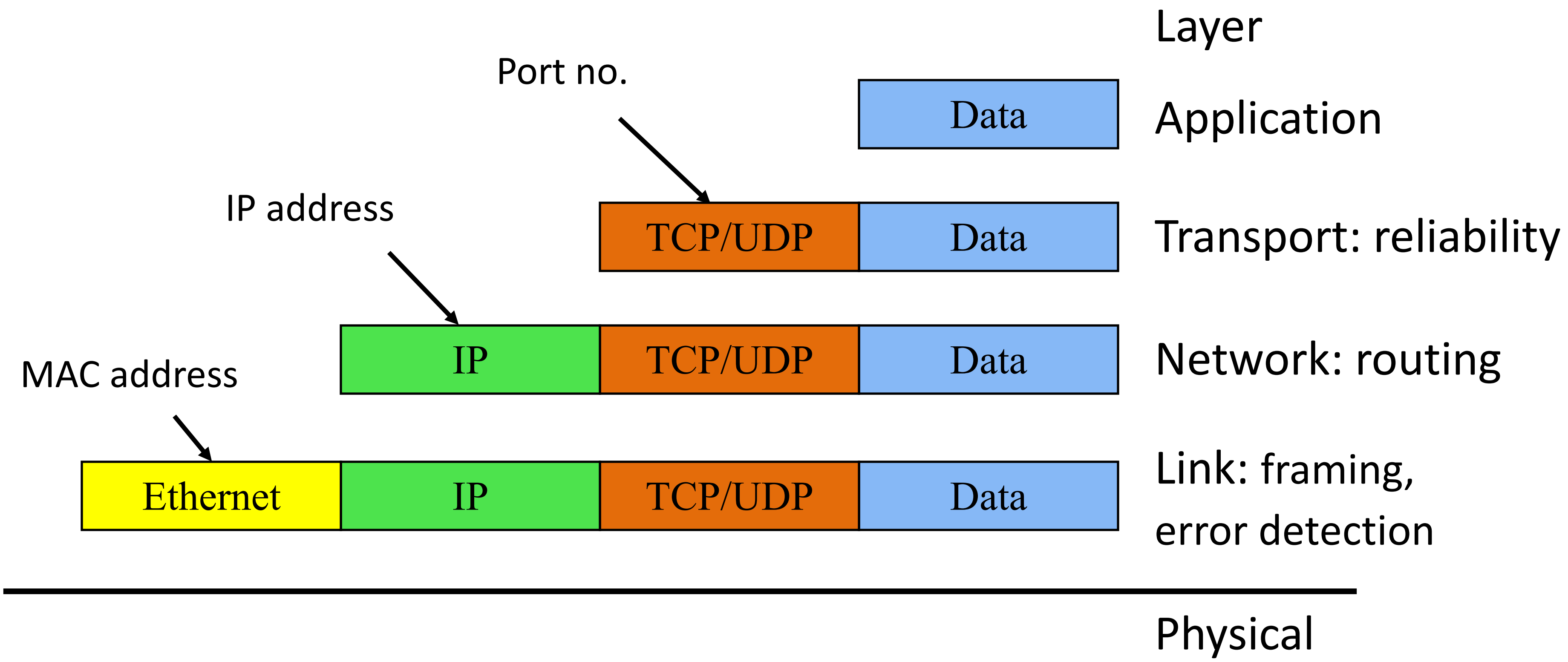
Transport: end-to-end connections, reliability

Network: routing

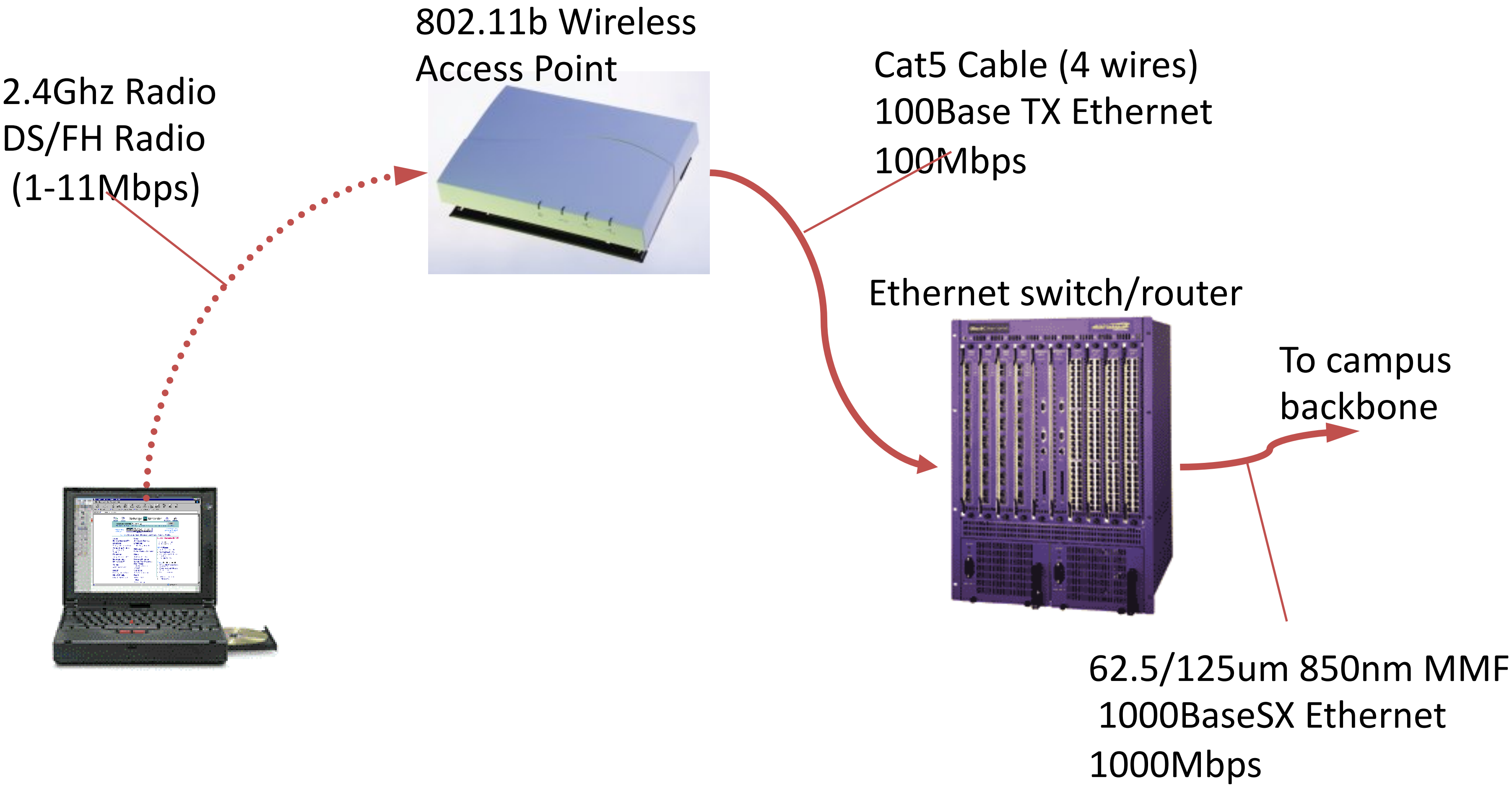
Link (data-link): framing, error detection

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

Layering and encapsulation



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

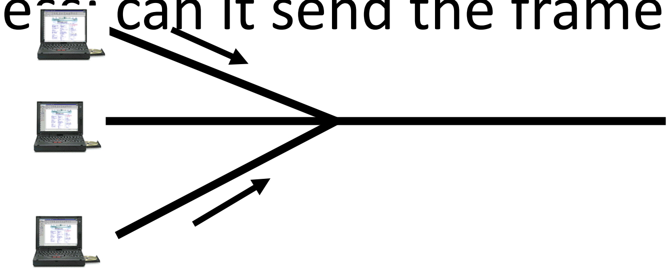
Ethernet switch/router

To campus
backbone

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

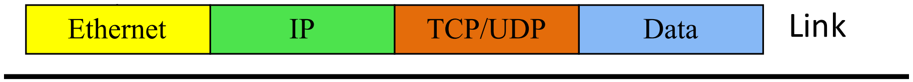
Link Layer (Ethernet, WiFi, Cable)

- 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”



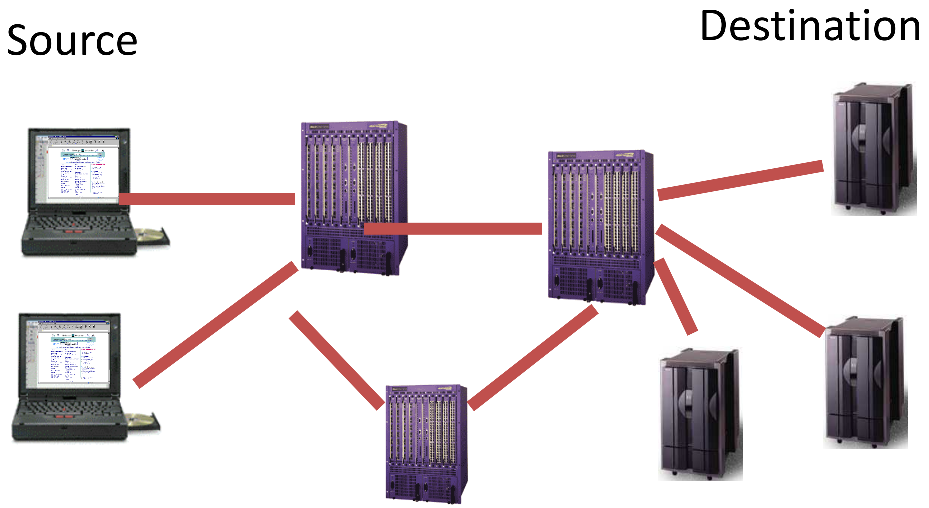
Ethernet	IP	TCP/UDP	Data

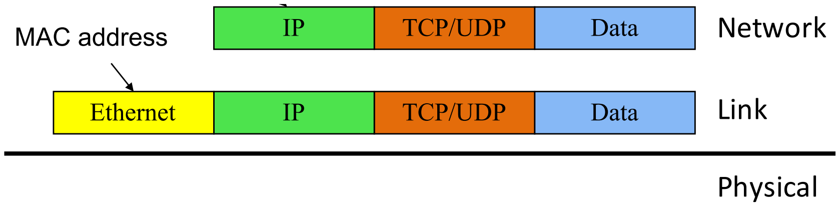
Link

Physical

Network Layer (IP)

- **Routers**: choose paths through network





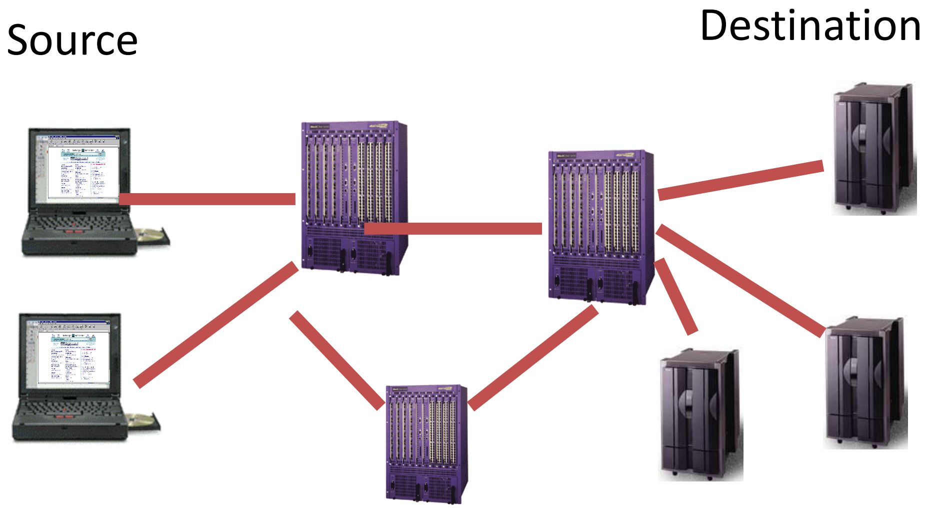
MAC address

IP | TCP/UDP | Data — Network

Ethernet | IP | TCP/UDP | Data — Link

Physical

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

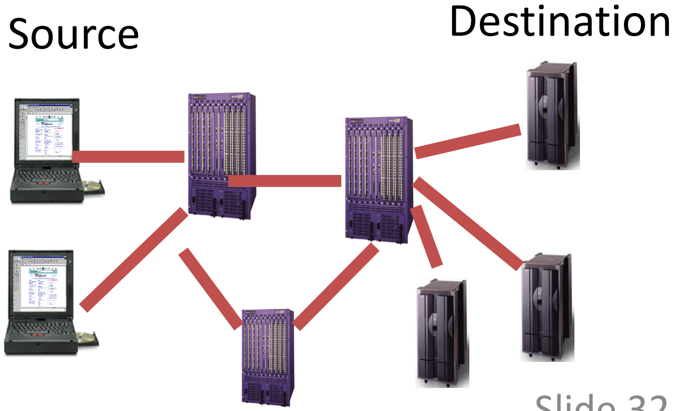


You're asked to design the Internet. Which do you choose for routing a conversation ("flow") over the network? (and why)

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.

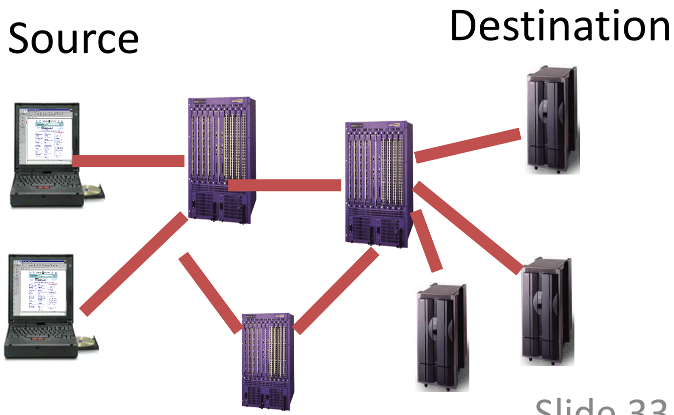


You're asked to design the Internet. Which do you choose for routing a conversation ("flow") over the network? (and why)

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

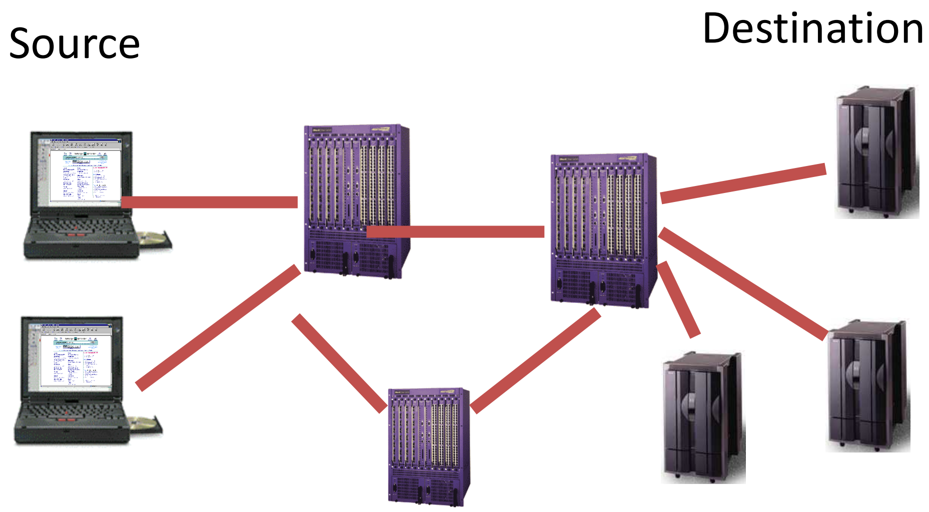
B. In the Internet the path choice is reevaluated for each of the flow's messages.

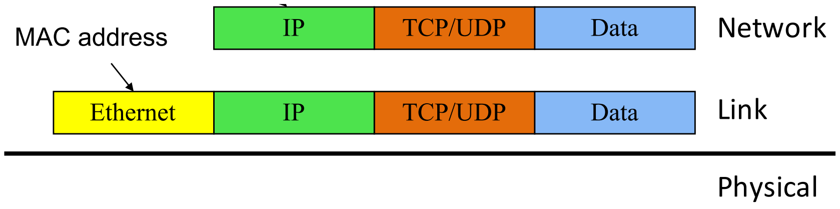
C. I would do something else.



Network Layer (IP)

- **Routers**: choose paths through network
  - *Circuit switching*: guaranteed channel for a session
  - *Packet switching*: statistical multiplexing of independent pieces of data




MAC address

IP | TCP/UDP | Data — Network

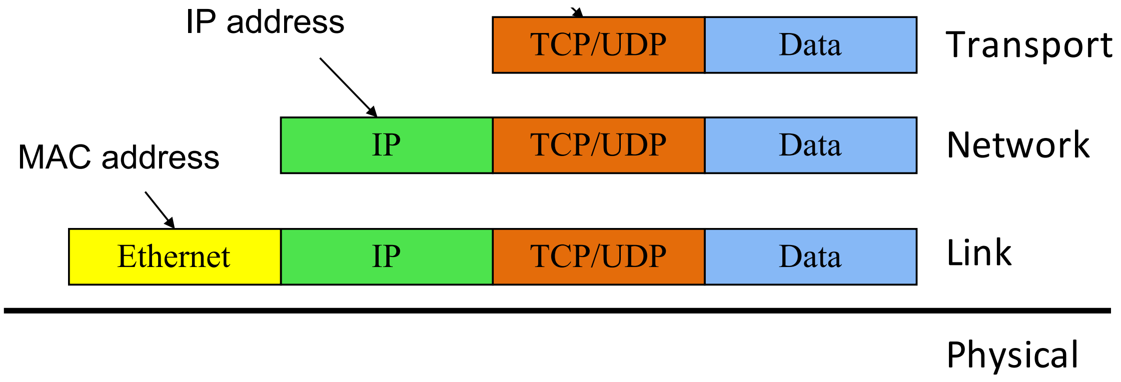
Ethernet | IP | TCP/UDP | Data — Link

Physical

Transport Layer (TCP, UDP)

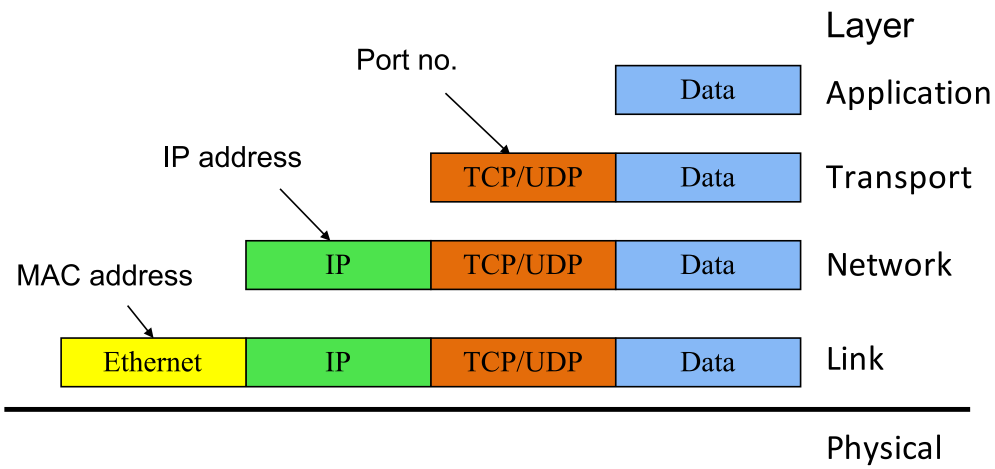
- Provides
  - Ordering
  - Error checking
  - Delivery guarantee
  - Congestion control
  - Flow control

- Or doesn't!

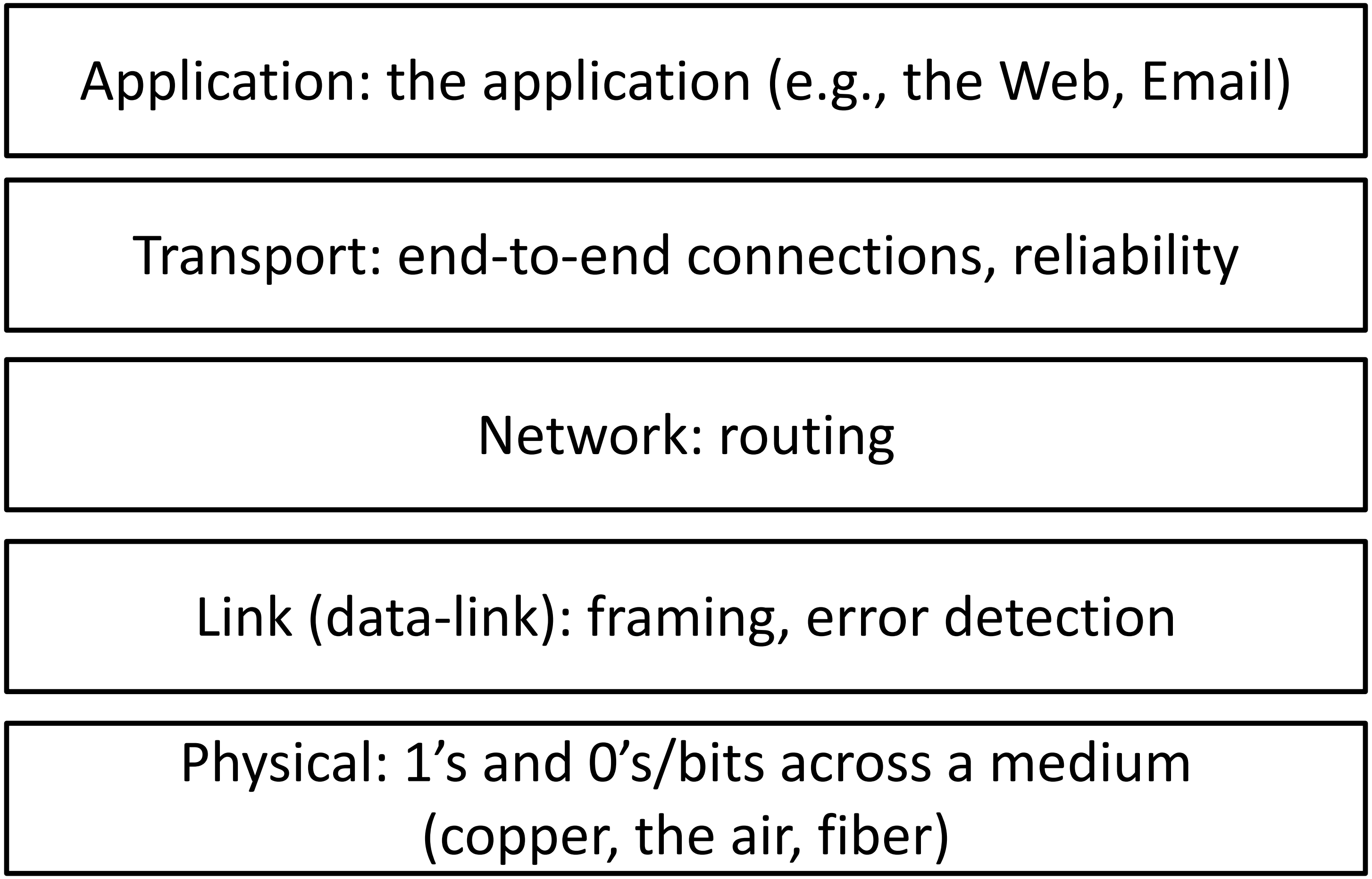


Application Layer (HTTP, FTP, SMTP, Skype)

- Does whatever an application does!



Five-Layer Internet Model



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

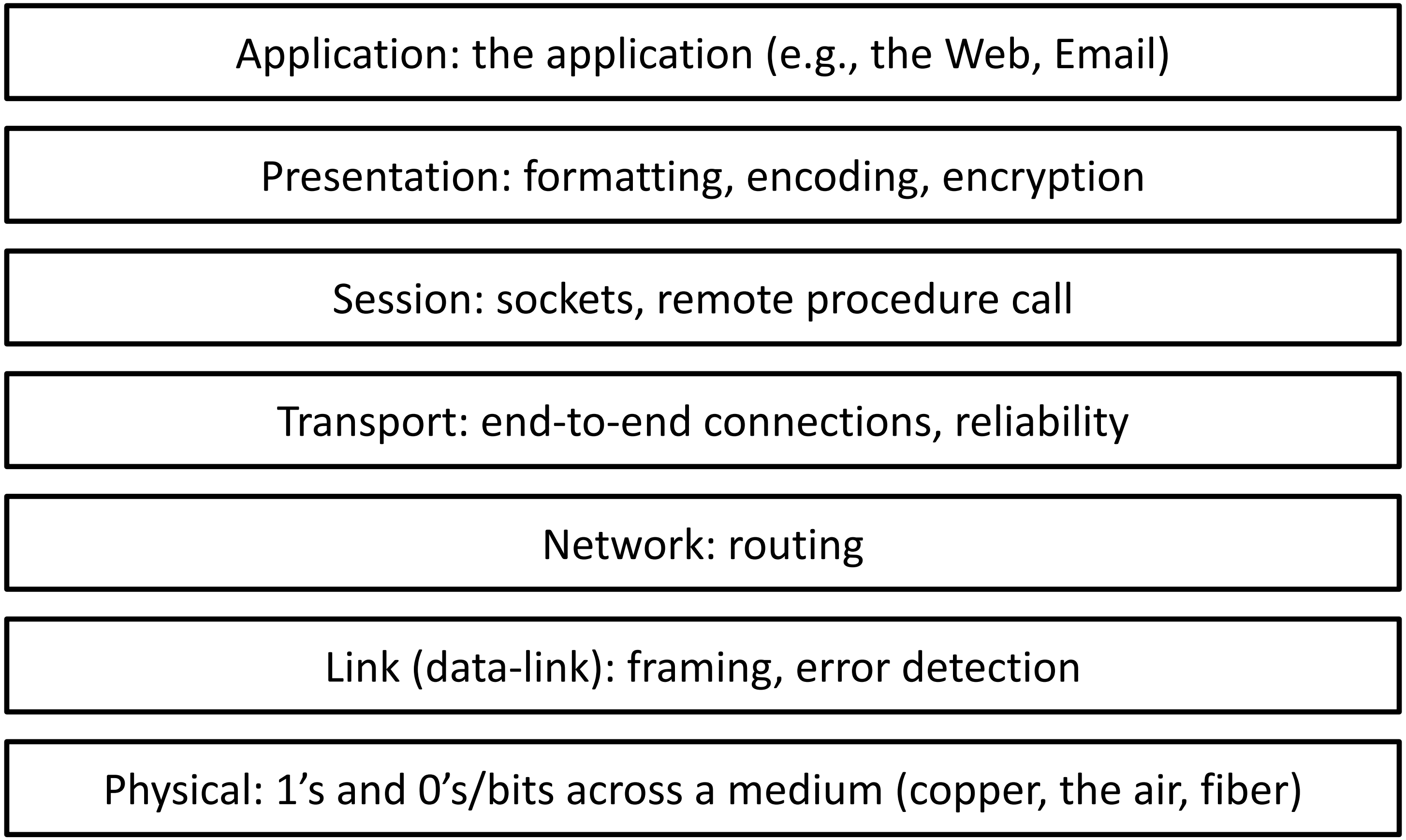
Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

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

OSI Seven-Layer Model



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

Presentation: formatting, encoding, encryption

Session: sockets, remote procedure call

Transport: end-to-end connections, reliability

Network: routing

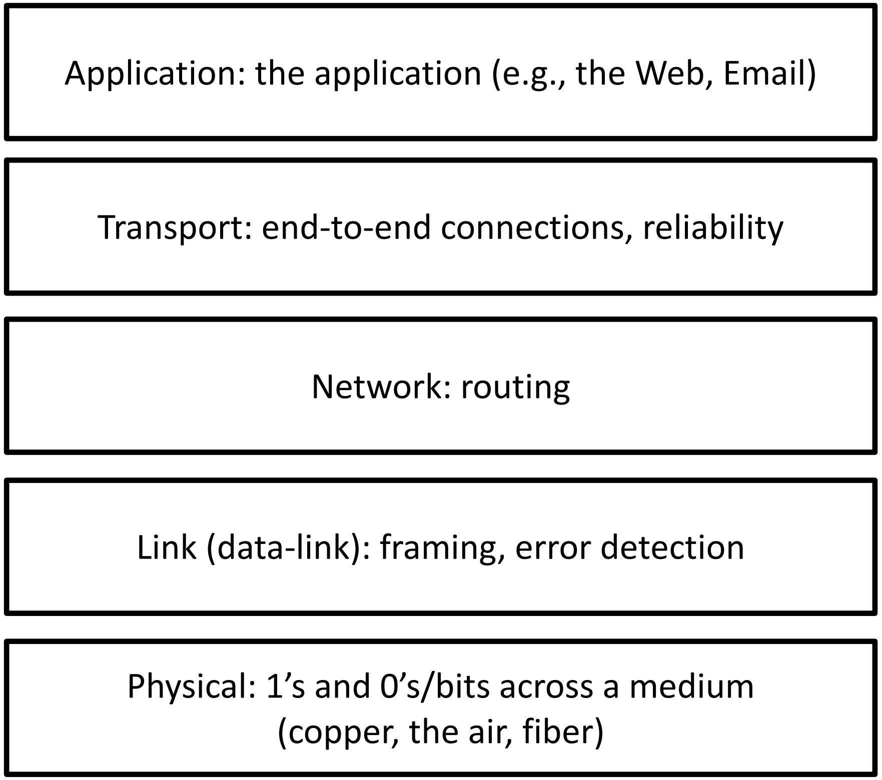
Link (data-link): framing, error detection

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

Layering and separation of functions is..

A. Great! It has a nice clean design and we can use easily swap any protocol we want at any layer.

B. Not really... there are some glaring disadvantages to it.



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

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

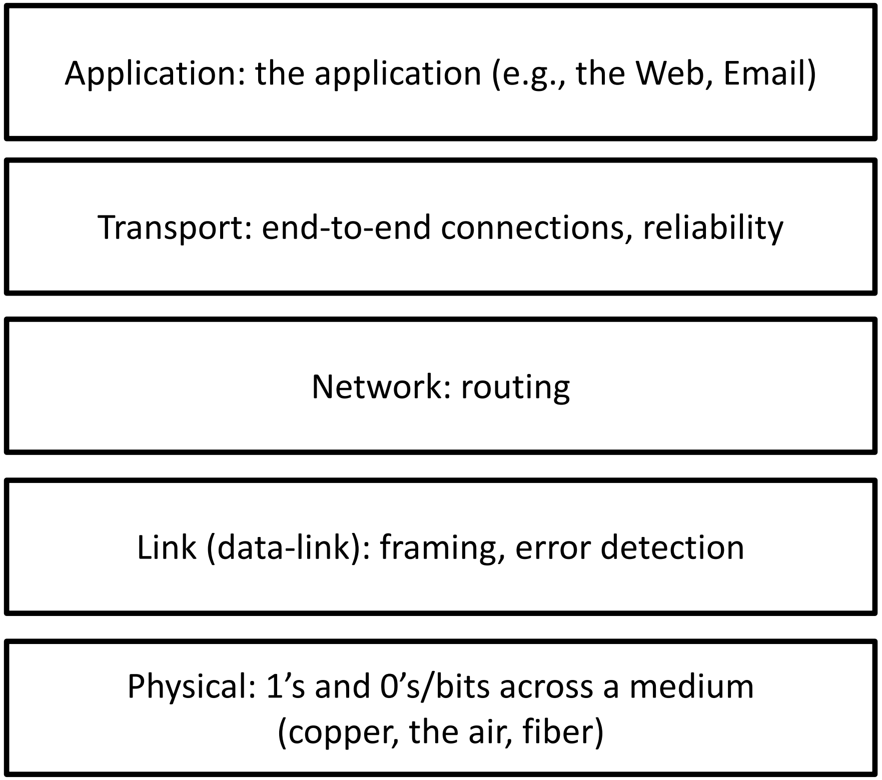
Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)

Layering and separation of functions is..

A. Great! It has a nice clean design and we can use easily swap any protocol we want at any layer.

B. Not really... there are some glaring disadvantages to it.

(A and B): The network layer, is one layer where every entity has to agree on a common addressing protocol.



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

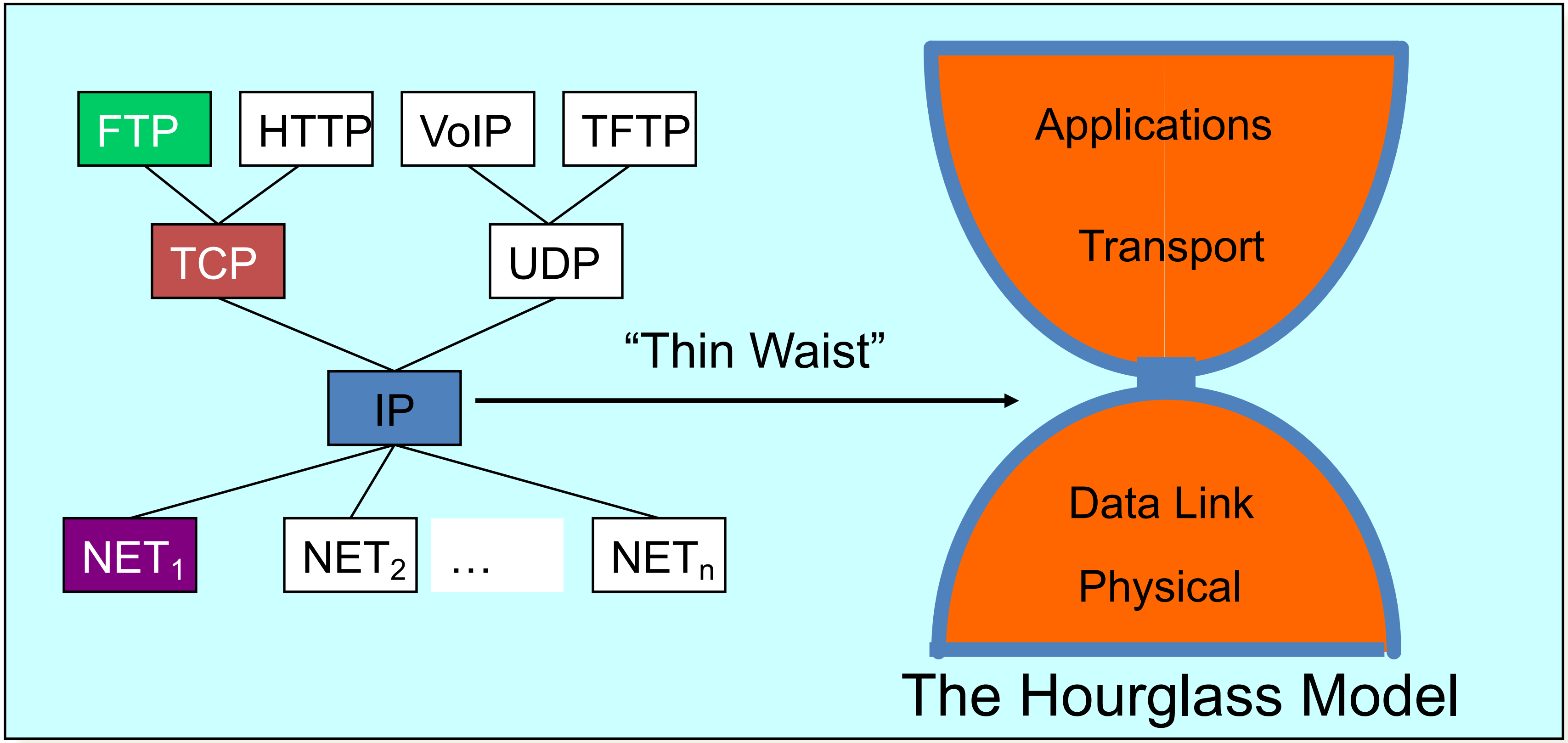
Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

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

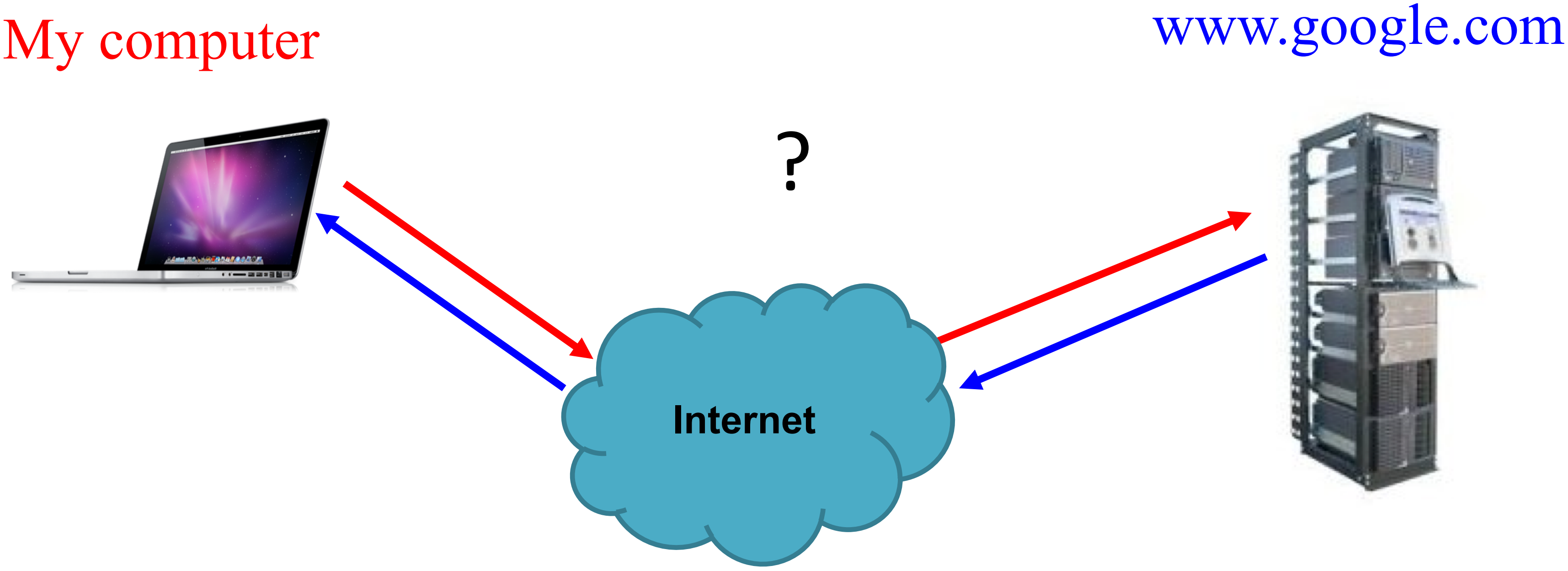
Internet Protocol Suite



The Hourglass Model

Putting this all together

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



My computer

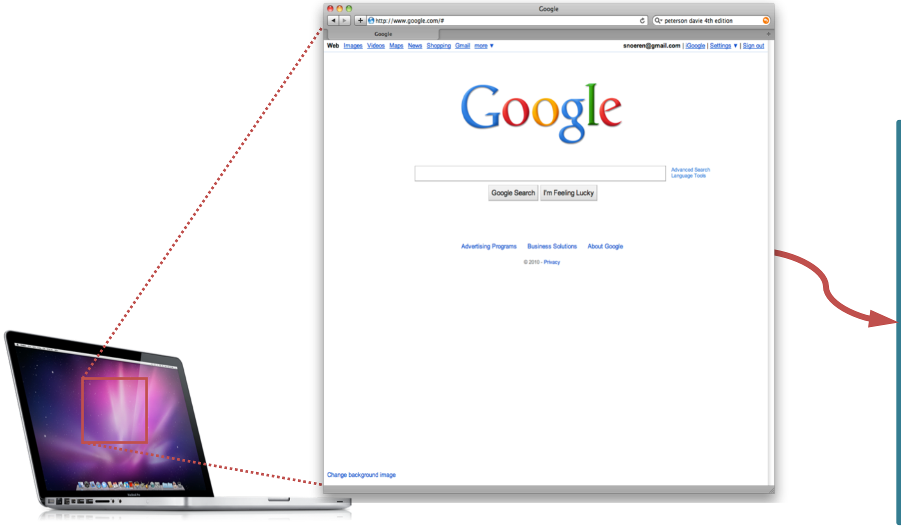
www.google.com

?

Internet

Web request (HTTP)

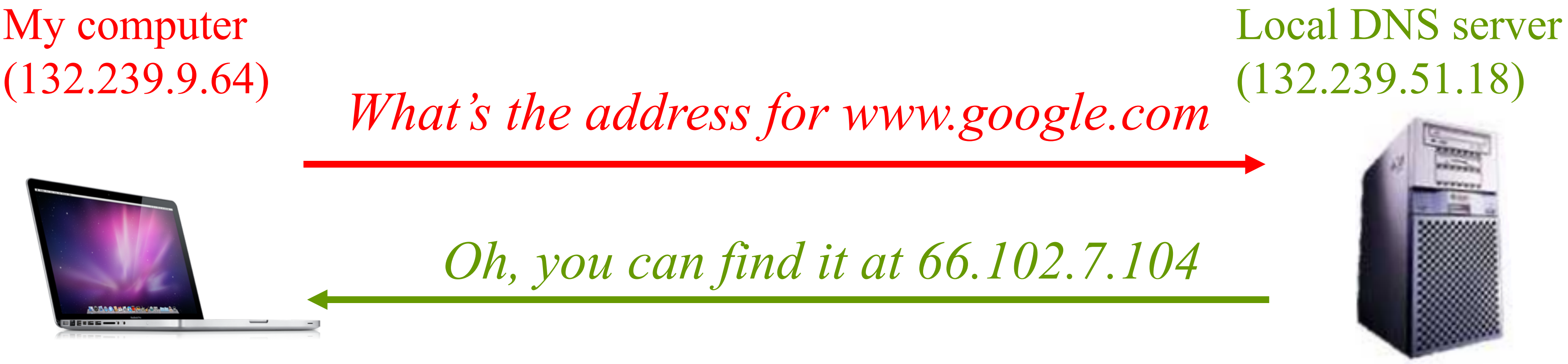
- Turn click into HTTP request



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

Name resolution (DNS)

- Where is www.google.com?



My computer
(132.239.9.64)

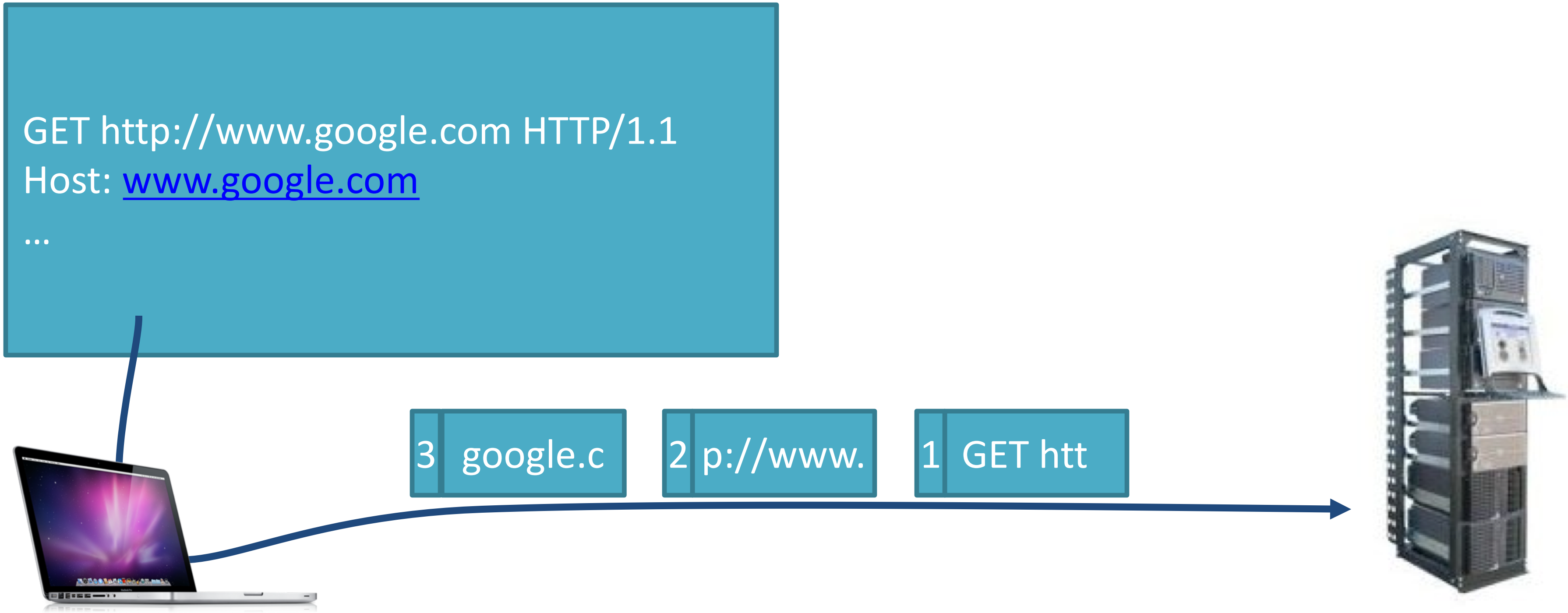
What's the address for www.google.com

Local DNS server
(132.239.51.18)

Oh, you can find it at 66.102.7.104

Data transport (TCP)

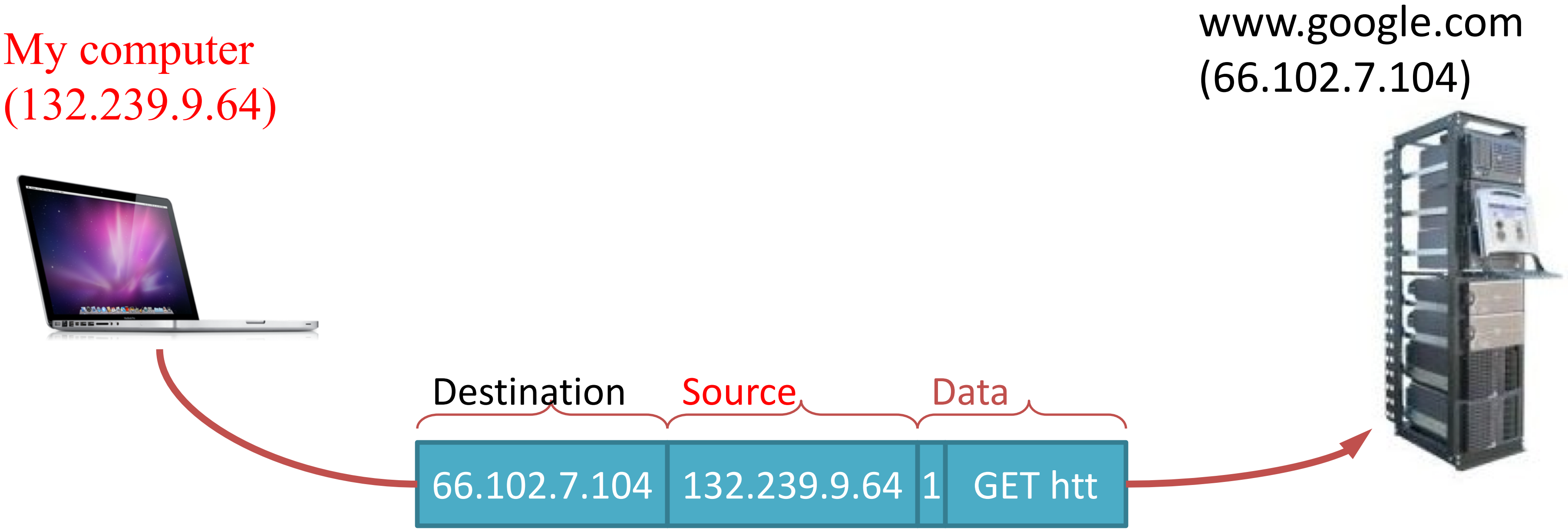
- Break message into packets (TCP segments)
- Should be delivered reliably & in-order



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

Global Network Addressing

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



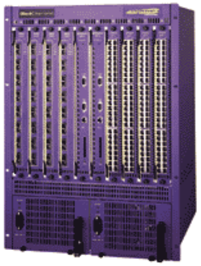
My computer
(132.239.9.64)

www.google.com
(66.102.7.104)

(IP) At Each Router

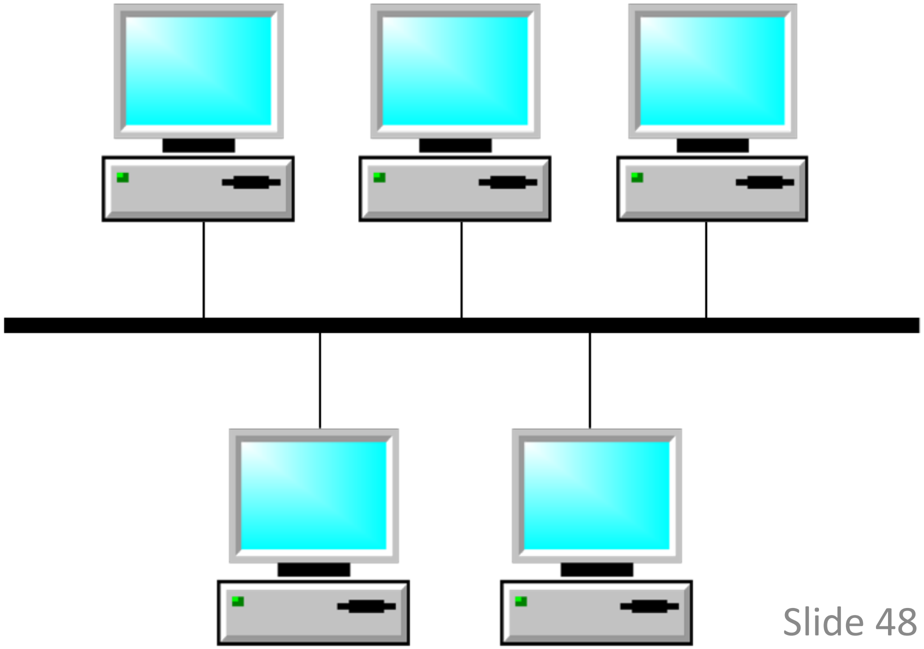
- Where do I send this to get it closer to Google?

- Which is the best route to take?

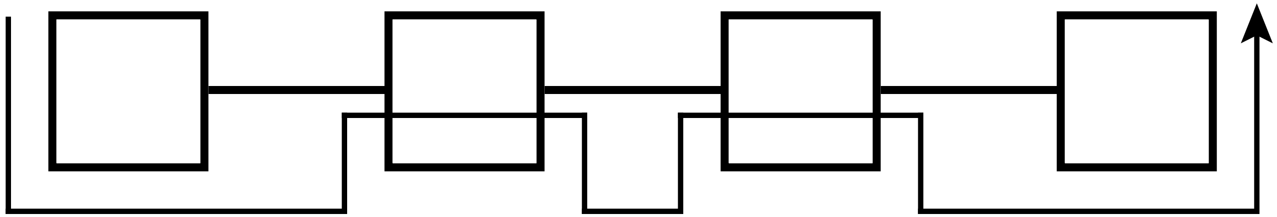


Link & Physical Layers

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



The “End-to-End” Argument



Don’t provide a function at lower 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), pp. 277-288, 1984.

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

A. All of Them

B. Transport through Physical

C. Network, Link and Physical

D. Link and Physical

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

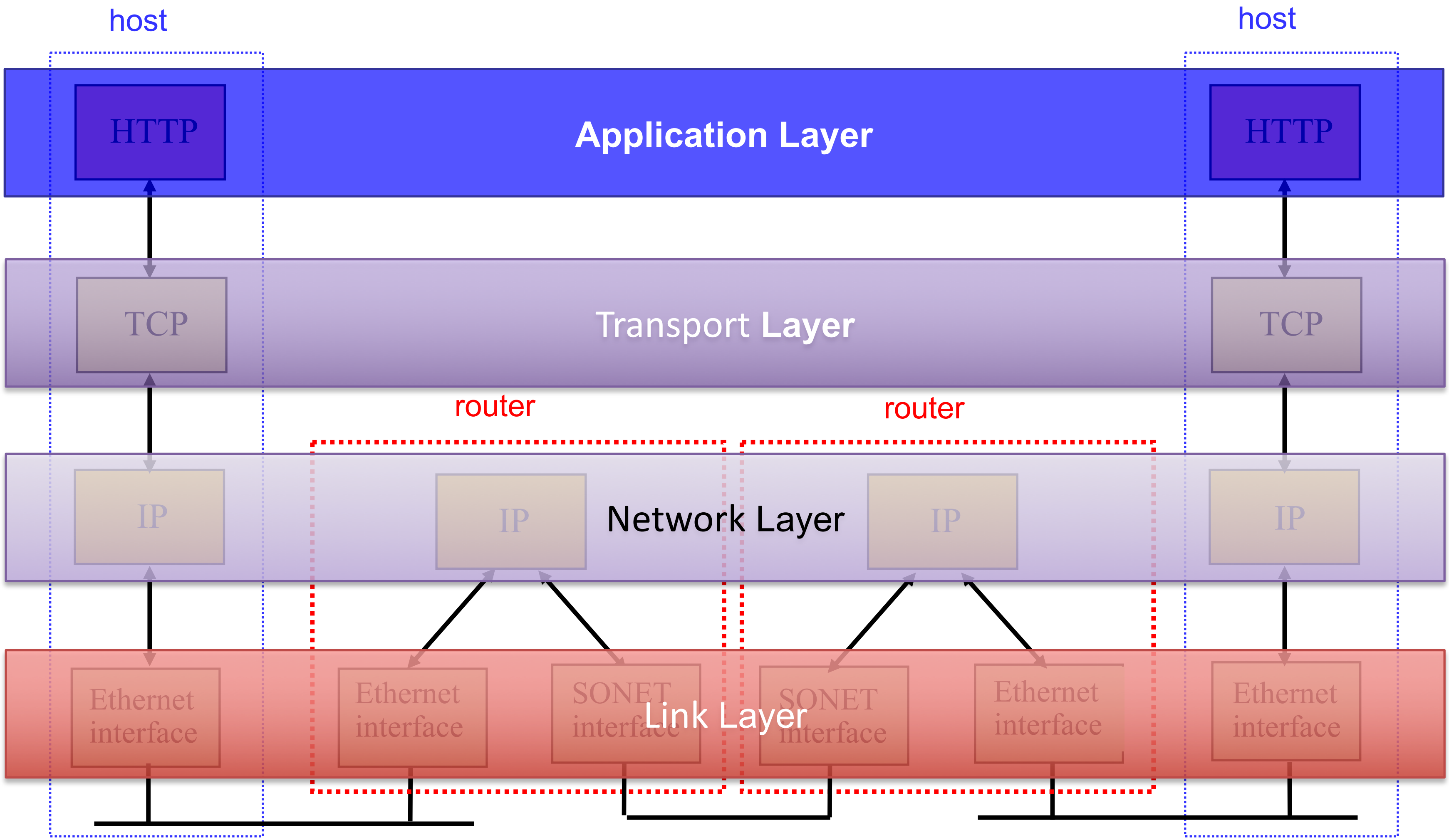
A. All of Them

B. Transport through Physical

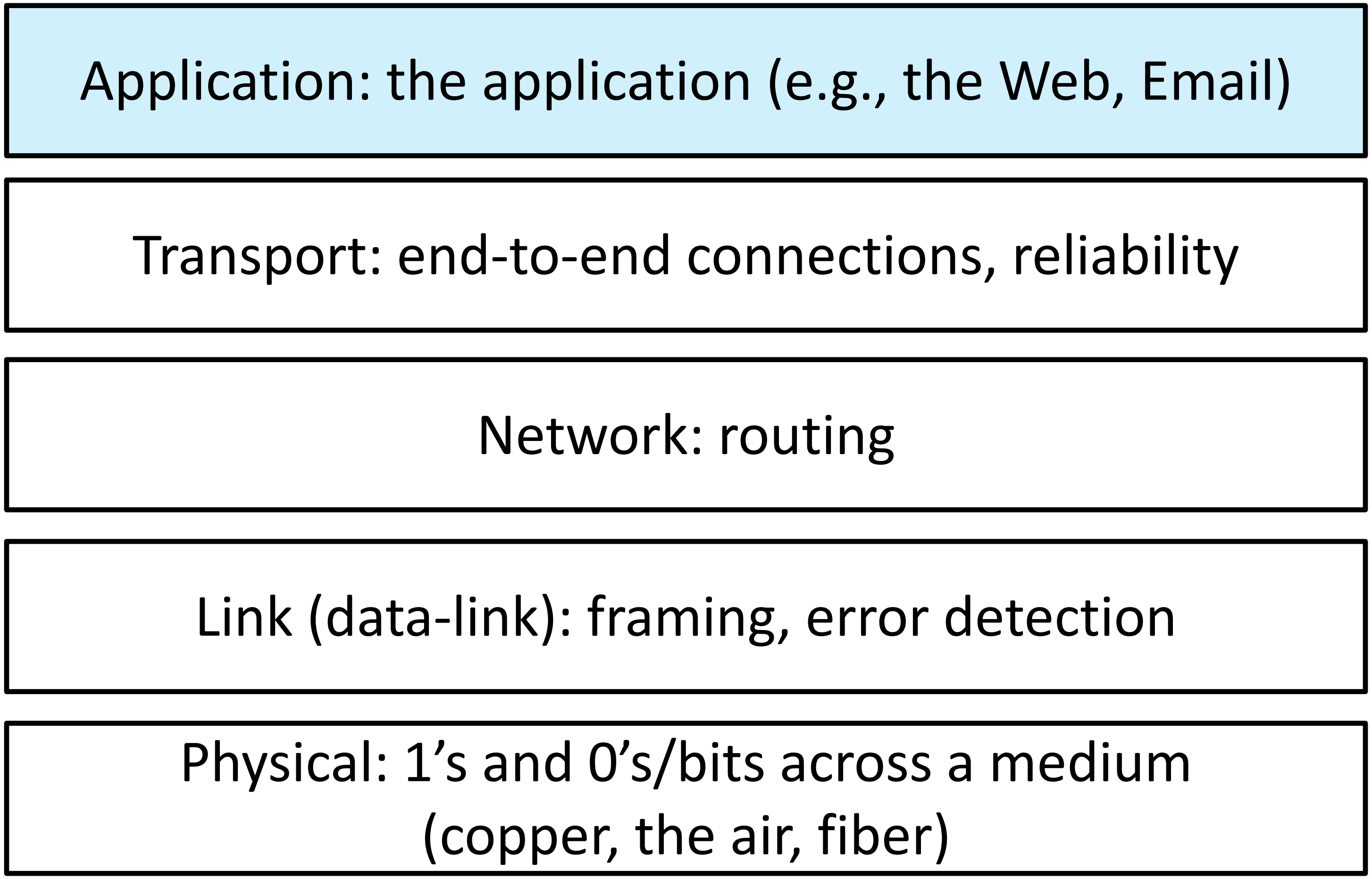
C. Network, Link and Physical

D. Link and Physical

TCP/IP Protocol Stack



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)

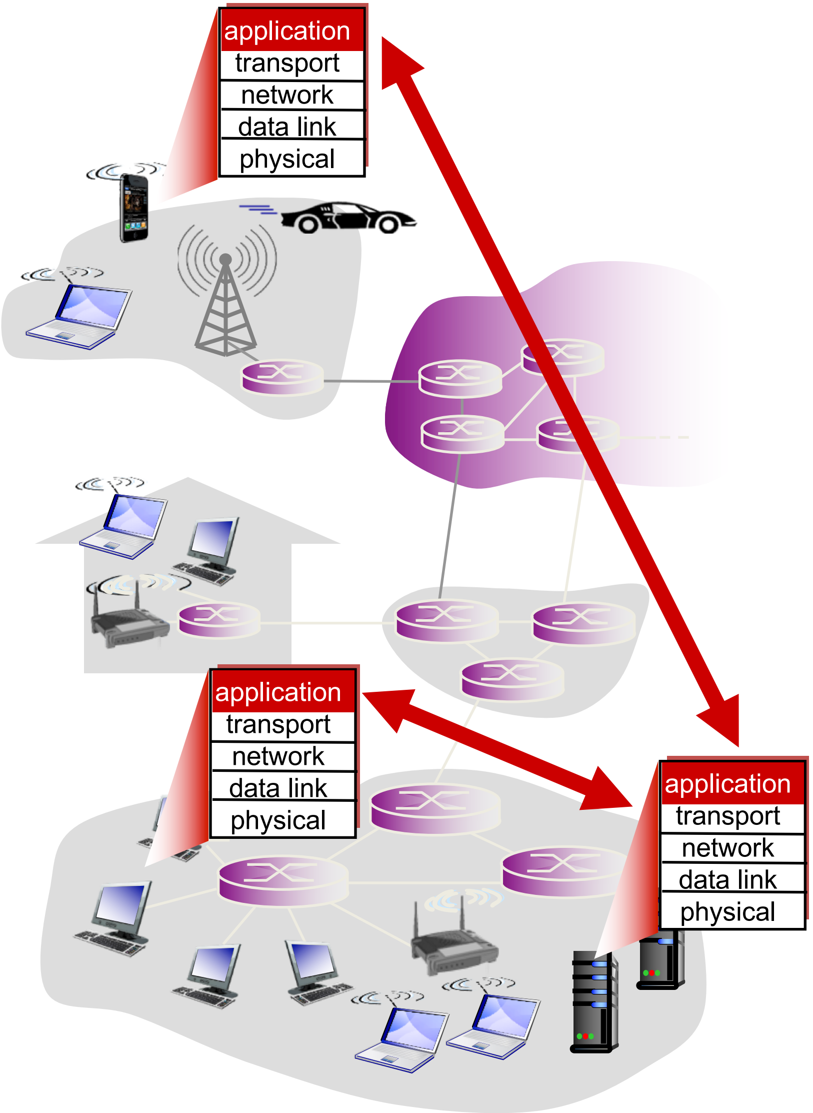
Creating a network app

write programs that:

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

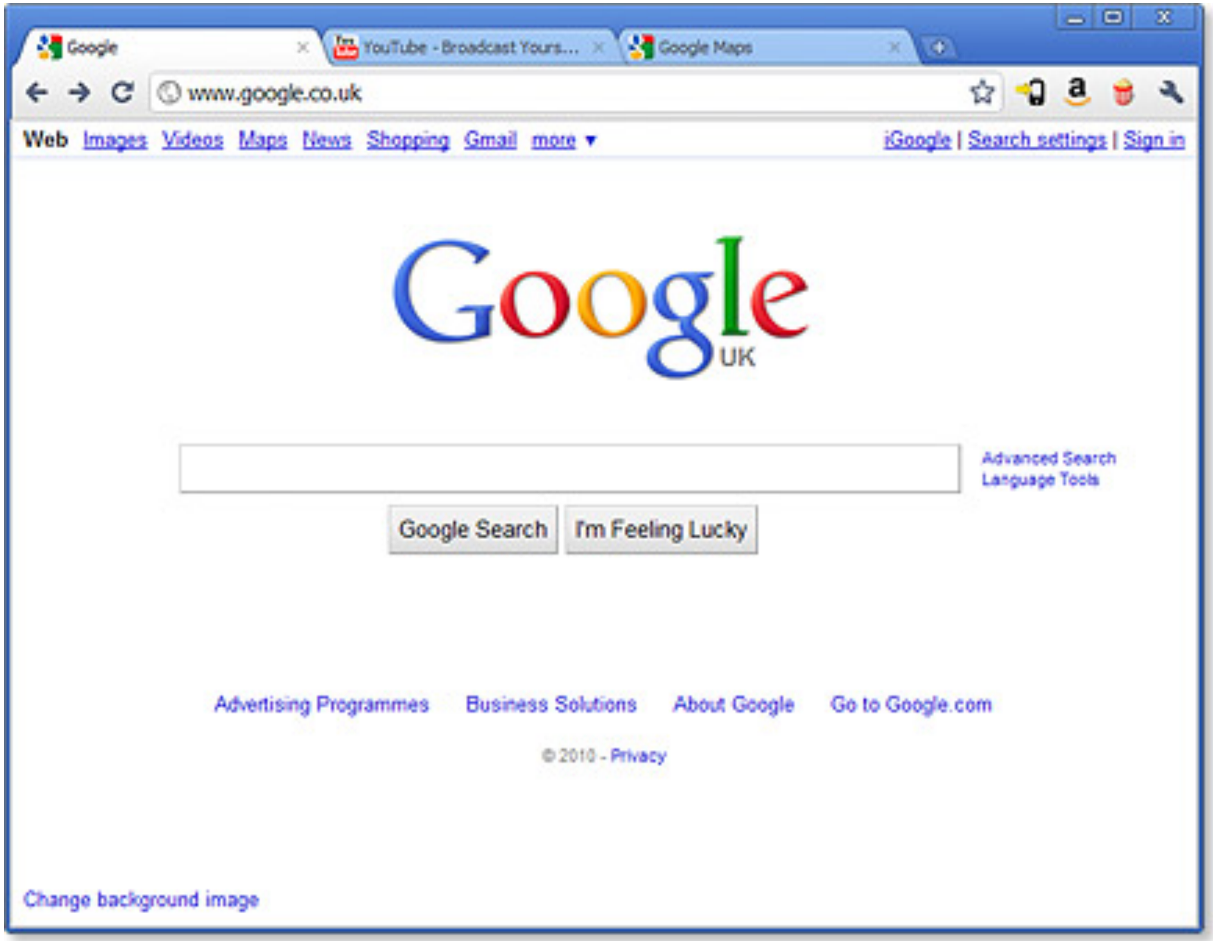
no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation



Adapted from Kurose and Ross

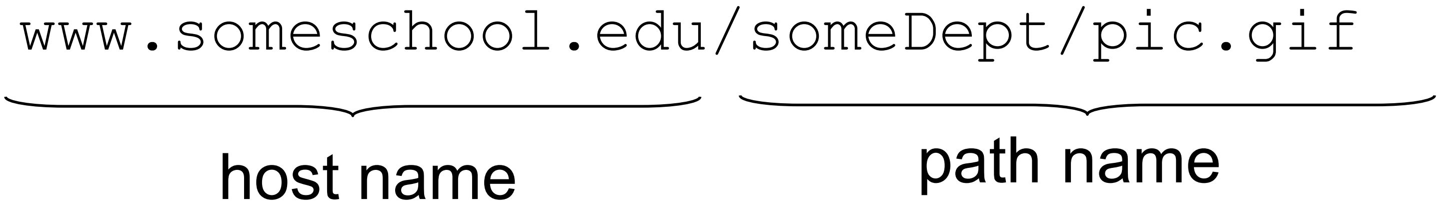
What IS A Web Browser?



HTTP and the Web

First, a review…

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL, e.g.,

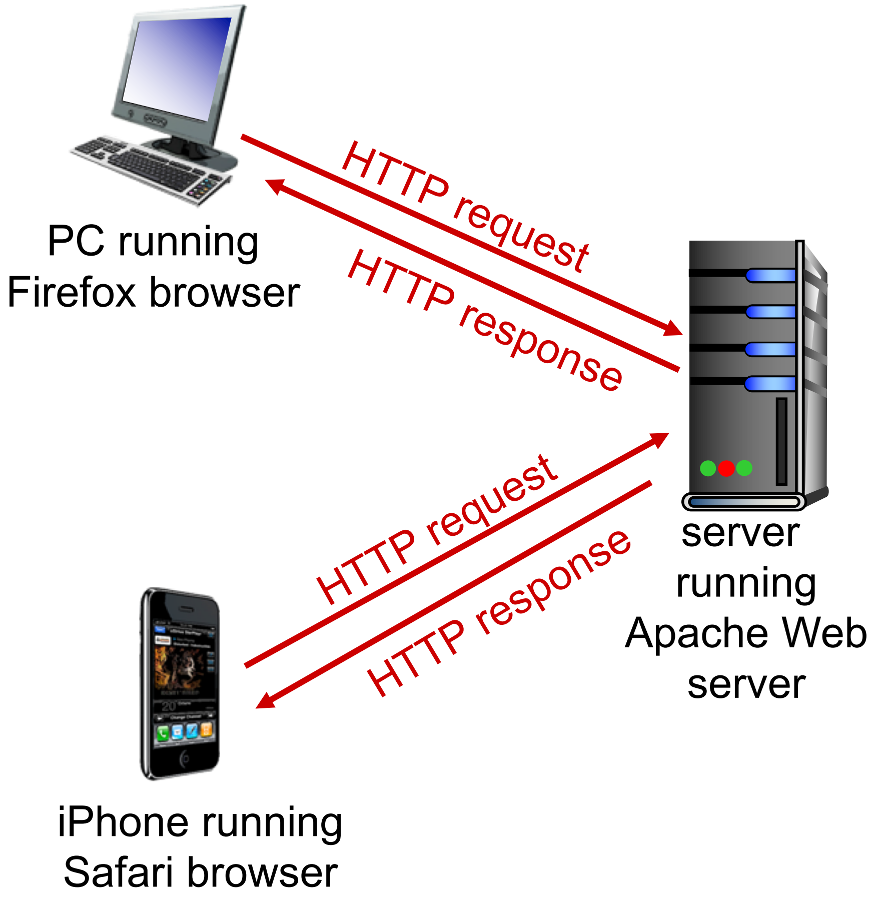


```
www.someschool.edu/someDept/pic.gif
```

Adapted from: Kurose & Ross

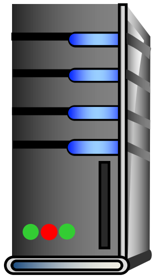
HTTP: Hypertext transfer protocol

- client/server model
  - client: browser that requests, receives, (using HTTP protocol) and “displays” Web objects
  - server: Web server sends (using HTTP protocol) objects in response to requests


PC running Firefox browser
HTTP request
HTTP response
server running Apache Web server
HTTP request
HTTP response
iPhone running Safari browser

Adapted from: Kurose and Ross

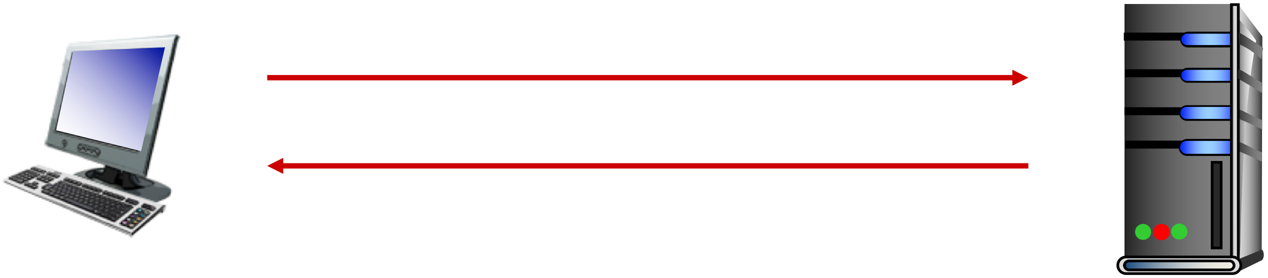
HTTP Overview



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

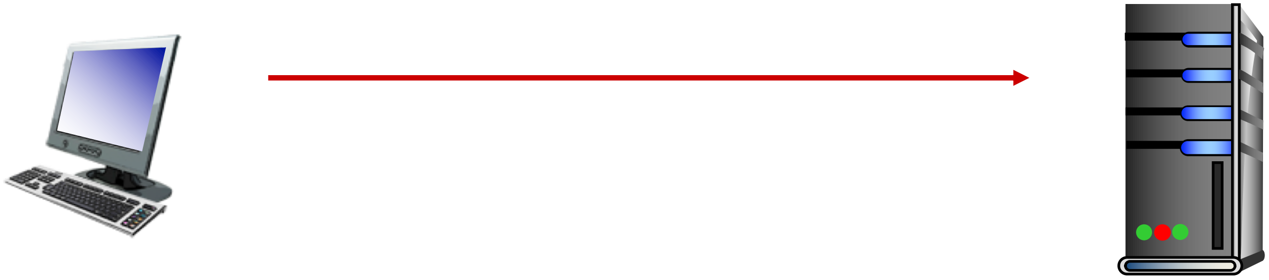


2. Browser establishes connection with server.

Looks up “some.host.name.tld”

Calls connect()

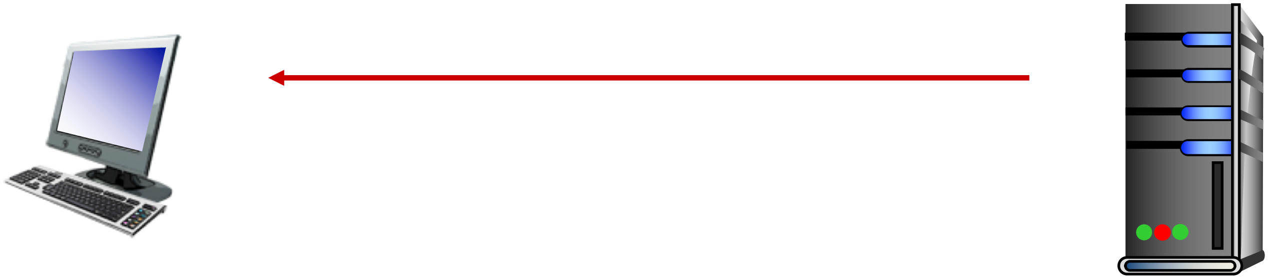
HTTP Overview



3. Browser requests the corresponding data.

```
GET /directory/name/file.ext HTTP/1.0
Host: some.host.name.tld
[other optional fields, for example:]
User-agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
Accept-language: en
```

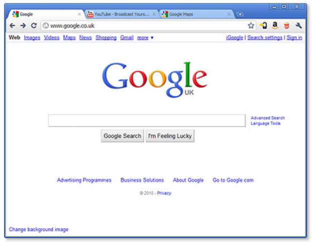
HTTP Overview



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,~~ and closes 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

2. Type in a GET HTTP request:

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

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

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
```

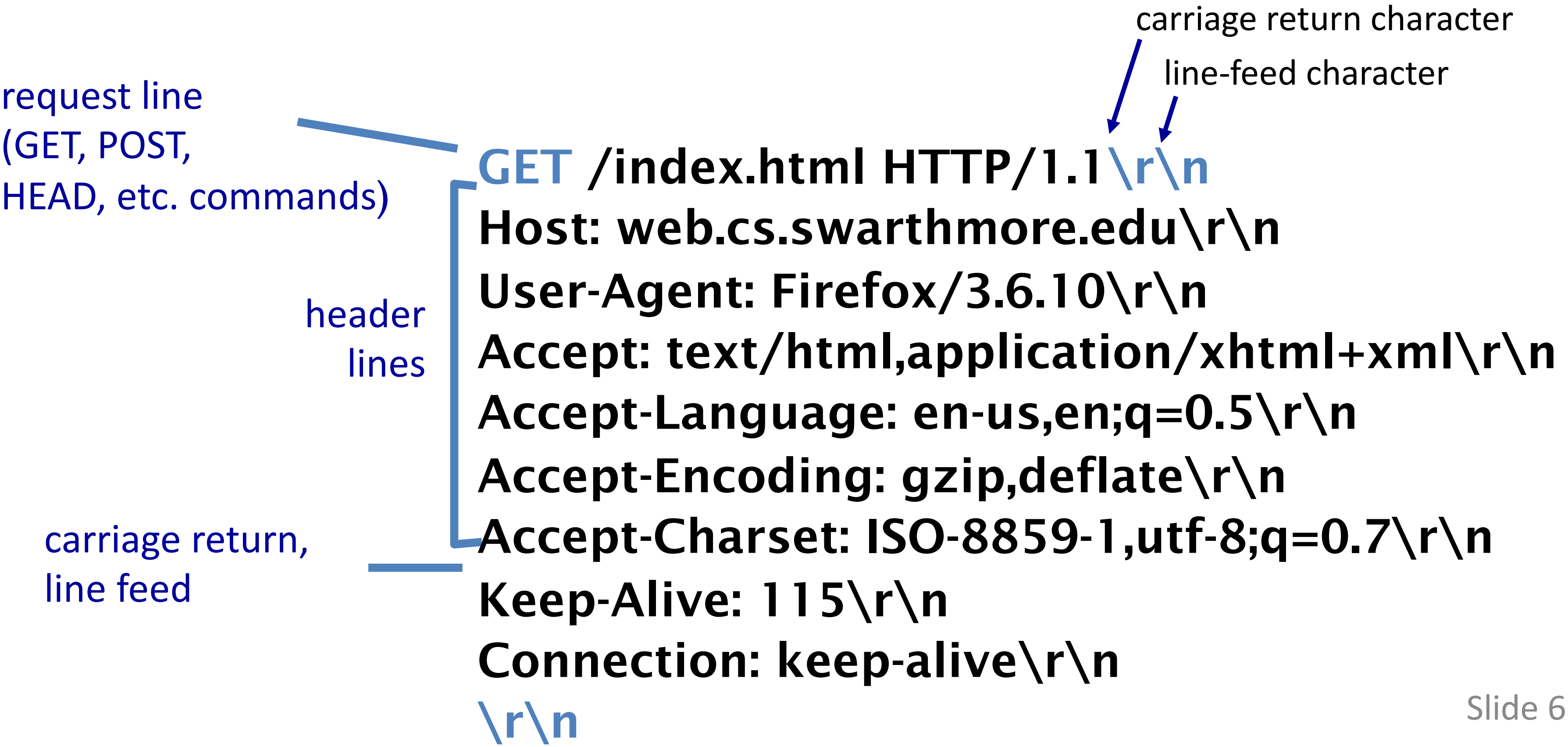
Response
headers

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

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

HTTP request message

- two types of HTTP messages: **request, response**
- **HTTP request message:** ASCII (human-readable format)



carriage return character

line-feed character

request line
(GET, POST,
HEAD, etc. commands)

header
lines

carriage return,
line feed

```
GET /index.html HTTP/1.1\r\n
Host: web.cs.swarthmore.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```