



UNIT 11B

The Internet: Higher-level protocols

Demo

...

simple network access in Python

Sending email

```
# mail (run where there is a local mail server)

import smtplib
from email.mime.text import MIMEText

def mail_demo() :
    msg = MIMEText('Give me an A!')
    msg['Subject'] = 'My grade'
    msg['From'] = 'student@example.org'
    msg['To'] = 'teacher@andrew.cmu.edu'
    server = smtplib.SMTP('localhost')
    server.send_message(msg)
    server.quit()
```

Fetching a web page

```
# web (run this wherever)

from urllib.request import urlopen

def web_demo() :
    page = urlopen('https://www.cs.cmu.edu/~15110-n15/other/webDemo.html')
    print("Opened URL ", page.geturl())
    print("Contents:")
    for line in page :
        print(line.decode('ISO-8859-1'))
```

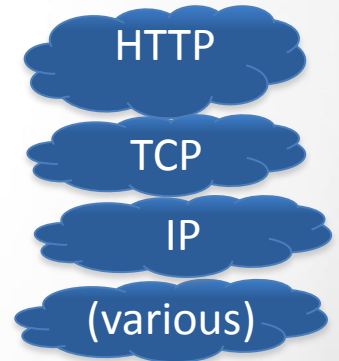
Higher-level protocols

...

Networking for human beings

“Higher” and “lower” level protocols

- Network protocols are organized in *layers*
 - IP packet delivery is the lowest *layer* of the Internet protocol *stack*
 - “Higher” layers use services provided by “lower” layers
 - Each layer is responsible for a type of service



Layers of the Internet ("higher" to "lower")

Application Layer

serves to human beings

- Handles requests from the user for data on the Internet
 - e.g. browser, email client, Skype

Transport Layer

serves to applications

- Handles splitting messages into packets for delivery.
 - converts between application messages and IP packets
 - figures out which application to deliver a message to
 - possibly detects and corrects delivery errors

Internet Layer

serves to transport layer

- Handles the task of sending packets across one or more networks.

Link Layer

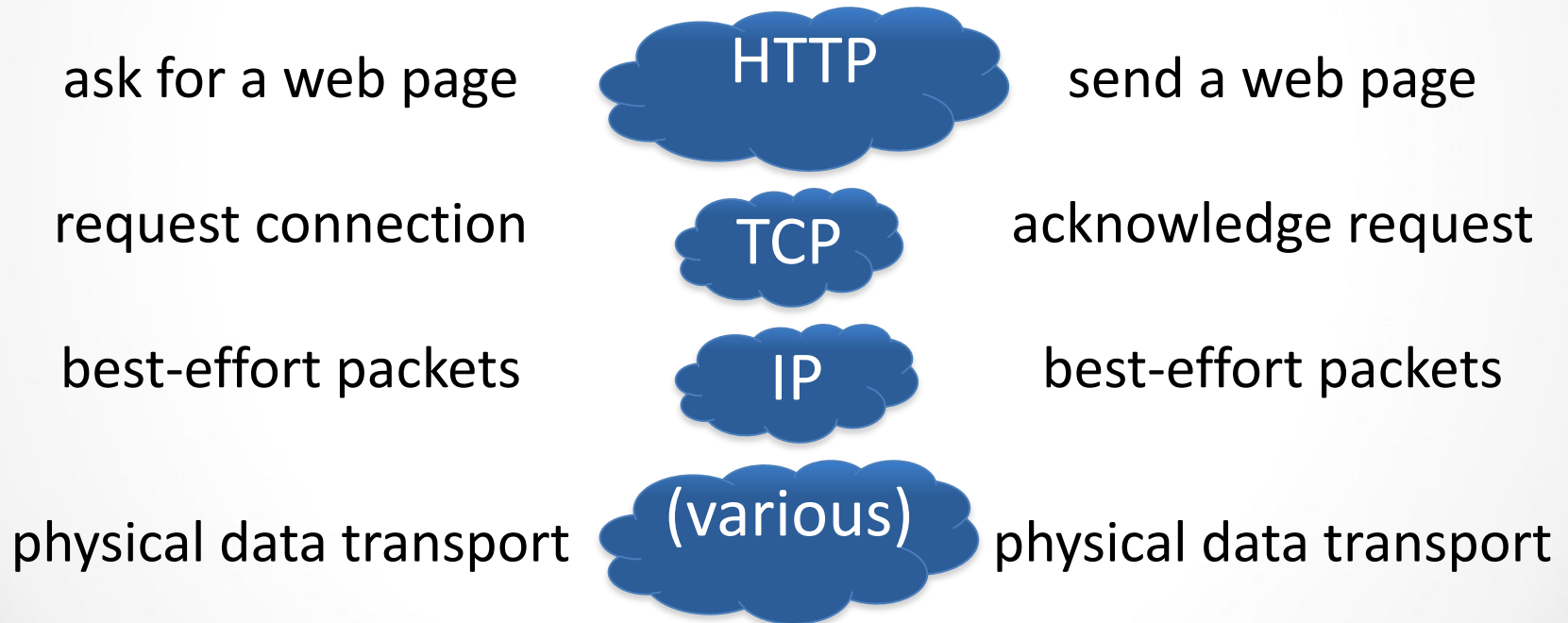
serves to internet layer

- Handles the physical transfer and reception of bits.

Example: Layering the Web

CLIENT MACHINE

SERVER MACHINE



Transport Layer

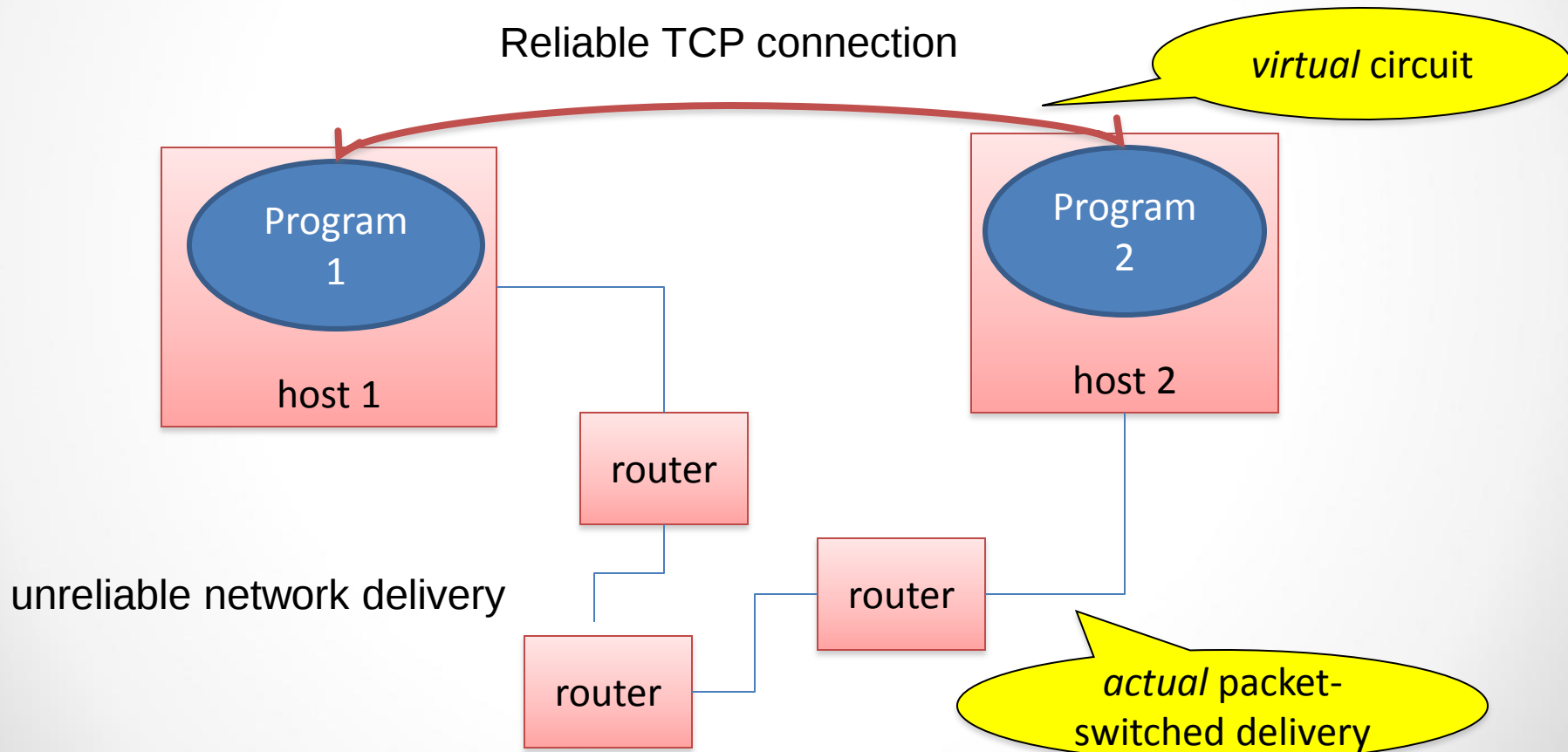
• • •

from IP packets to application messages

Transport Layer

- Splits **application messages** into **IP packets** and **maps applications to *port number***
 - IP address identifies machine, but port number identifies an application operating on that machine (web, email, etc.)
- Transport Control Protocol (**TCP**)
 - Creates a ***reliable* bi-directional** stream
(source address/port and destination address/port)
- User Datagram Protocol (**UDP**)
 - Creates a **single one-way** message to a remote application
(destination address/port)
 - used for voice, video, DNS lookup, ...

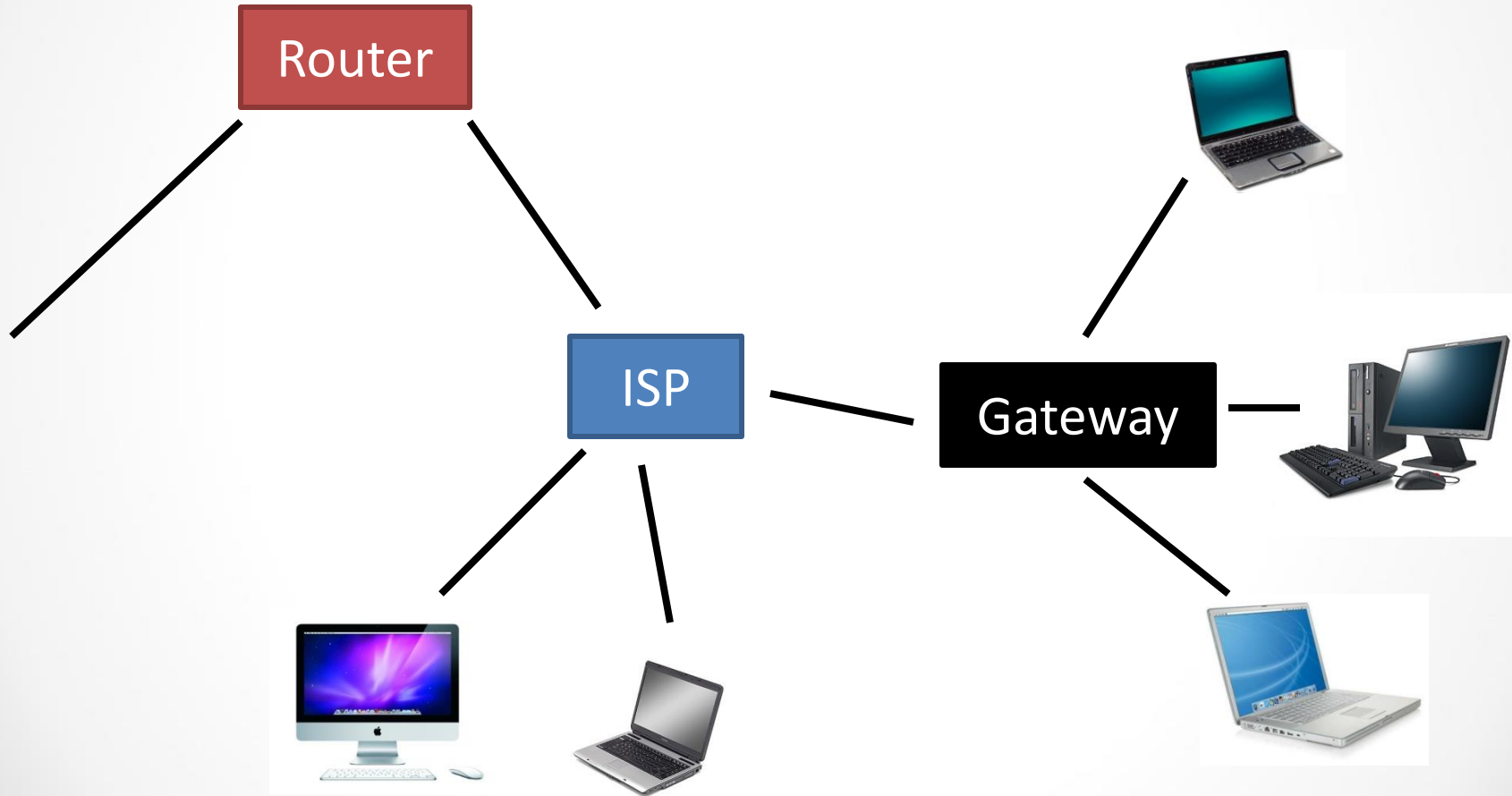
Transport Layer



Reliable Communication with TCP

- Suppose A and B are the TCP programs of two computers.
 - An application asks A to send a message to an application at B.
 - A breaks the message into several packets.
 - Each packet includes parity information, so B can check it for accuracy.
 - Packets are sent via IP.
 - B receives the packets.
 - If B is missing a packet or receives a corrupt packet, it can request retransmission.
 - If the packet is OK, B sends an acknowledgement.
 - If A doesn't get an acknowledgement, it will retransmit.
 - B assembles the incoming packets in order and provides the message to the appropriate application.

Network Address Translation (NAT)



Network Address Translation (NAT)

- Used to accommodate more users on the Internet, security, and administration.
- The gateway assigns an additional code called a port for each user. Packets are tagged with the port.
 - Against end-to-end delivery
- The gateway knows where to route the messages on the private network, but all messages from that private network share the same single IP address.

Remember the Class Activity

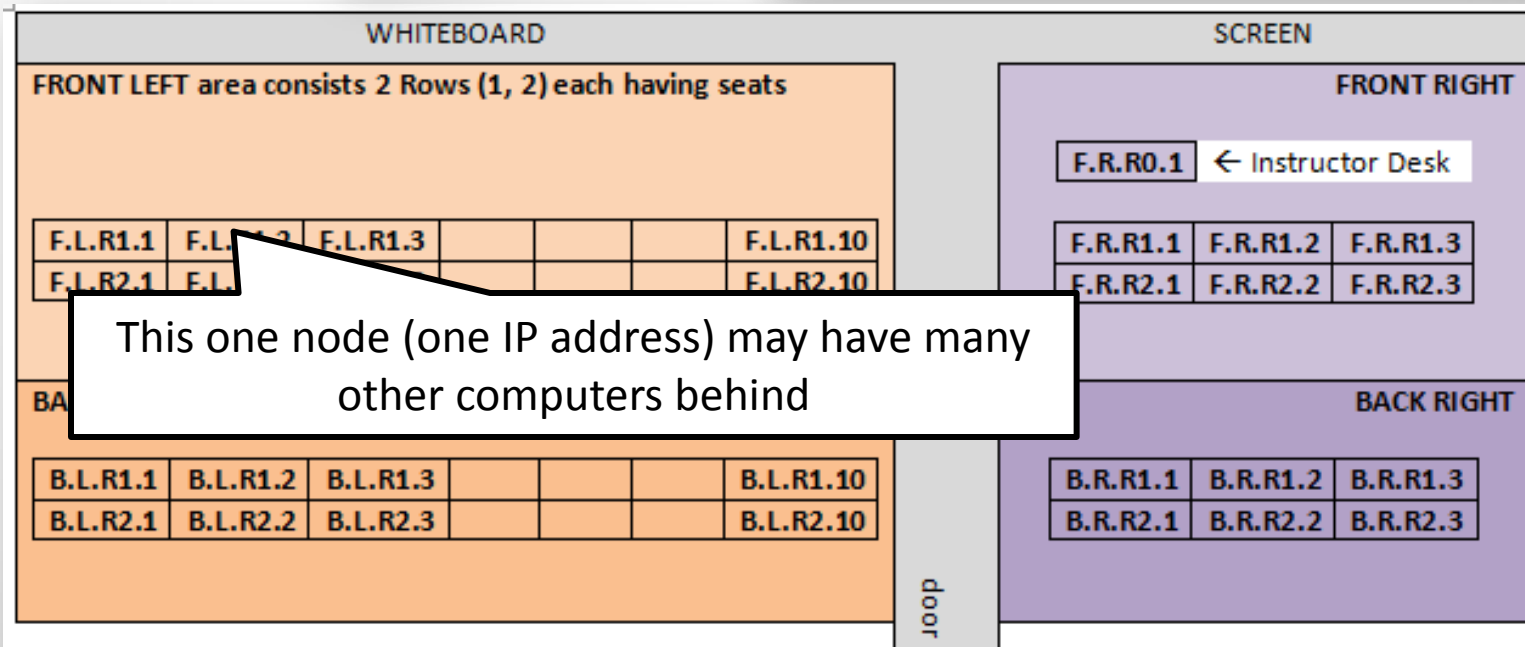
Sender:	Instructor.com	G
Receiver:	GreatCompany.com	
Serial Number:	1	

Sender:	Instructor.com	O
Receiver:	GreatCompany.com	
Serial Number:	2	

Sender:	Instructor.com	C
Receiver:	TheCompany.com	
Serial Number:	1	

Sender:	Instructor.com	O
Receiver:	TheCompany.com	
Serial Number:	2	

GreatCompany.com	____.____.____.____
TheCompany.com	____.____.____.____
CompanyTheGreat.com	____.____.____.____
Instructor.com	<u>F . R . R0 . 1</u>



Domain names

• • •

from 98.139.183.24 to yahoo.com

From names to IP addresses

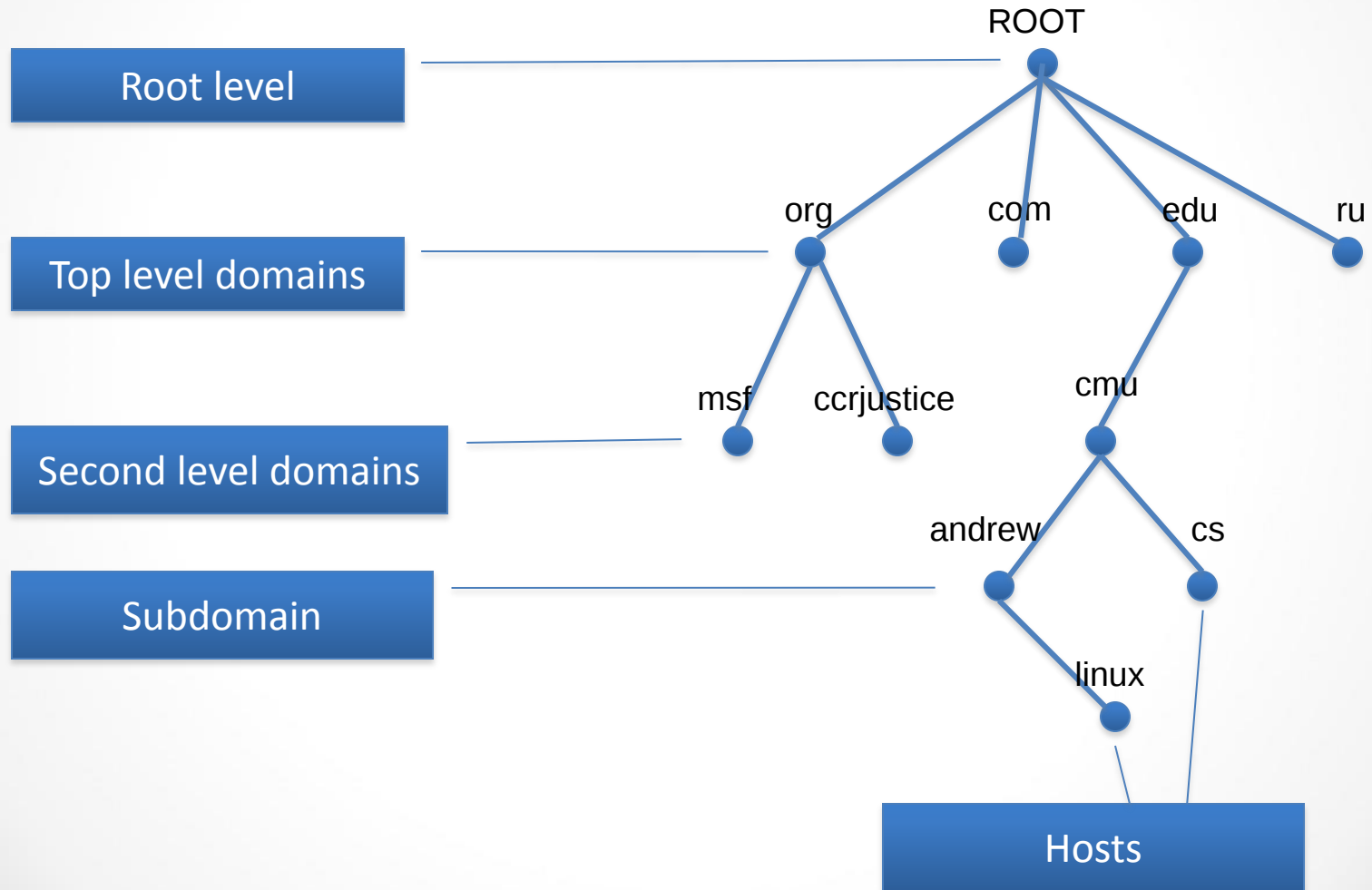
- URL: <http://www.cs.cmu.edu/~15110-n15/index.html>
- Email address: teacher@andrew.cmu.edu
- We don't want IP addresses in our URLs or email addresses—why not?
- Domain Name Service (DNS) *translates* names to addresses

DNS design

GreatCompany.com	____.____.____.____
TheCompany.com	____.____.____.____
CompanyTheGreat.com	____.____.____.____
Instructor.com	<u> F R R0 1 </u>

- Problem: so many names! How to make lookup fast?
- Solution: hierarchy of name servers
 - Each machine knows a name server, which knows how to find a **root name server**
 - root name servers know DNS servers for each **top-level domain** (e.g., "edu", "com", "net", "uk", "ru")
 - top-level domain servers know DNS servers for each **second-level domain** (e.g., "cmu.edu", "co.uk")
 - second-level domain servers know **each host** directly in their domain (e.g., "www.cmu.edu") and DNS servers for each **third-level domain** (e.g., "andrew.cmu.edu")

DNS Hierarchy (fragment)



DNS Lookup

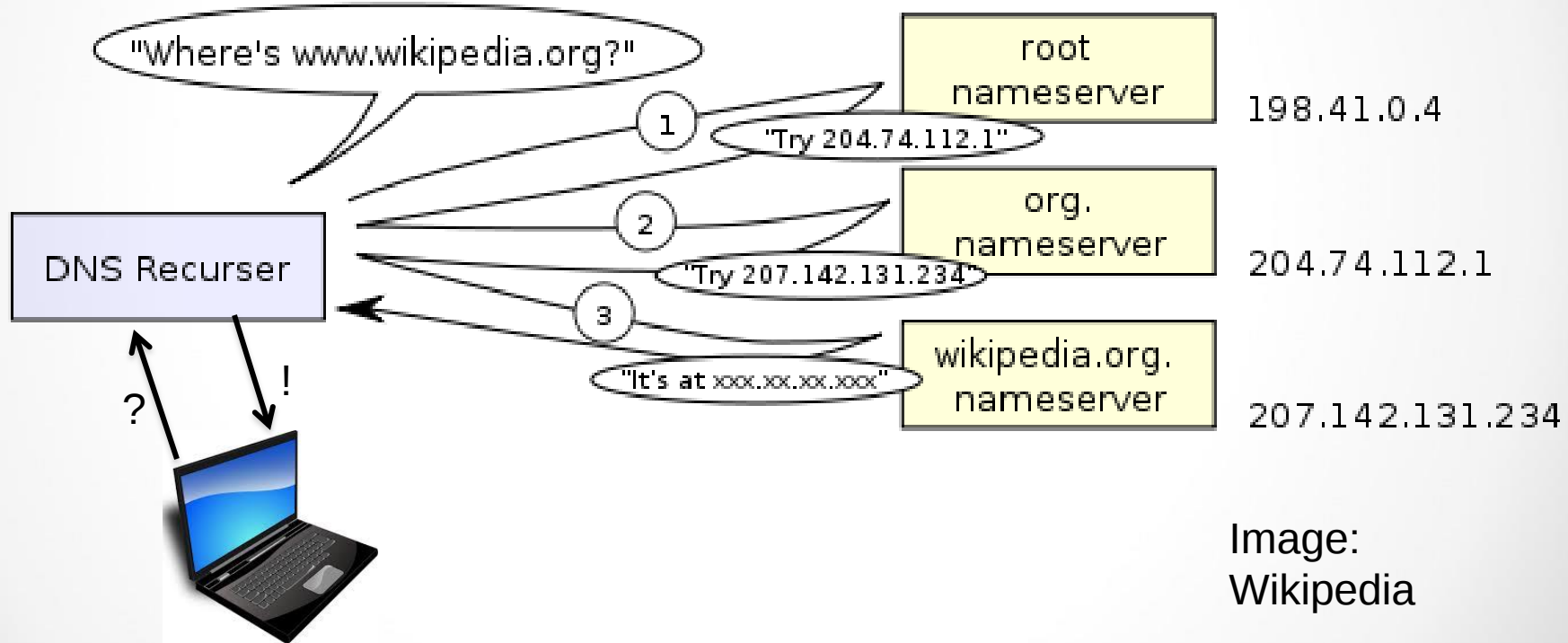


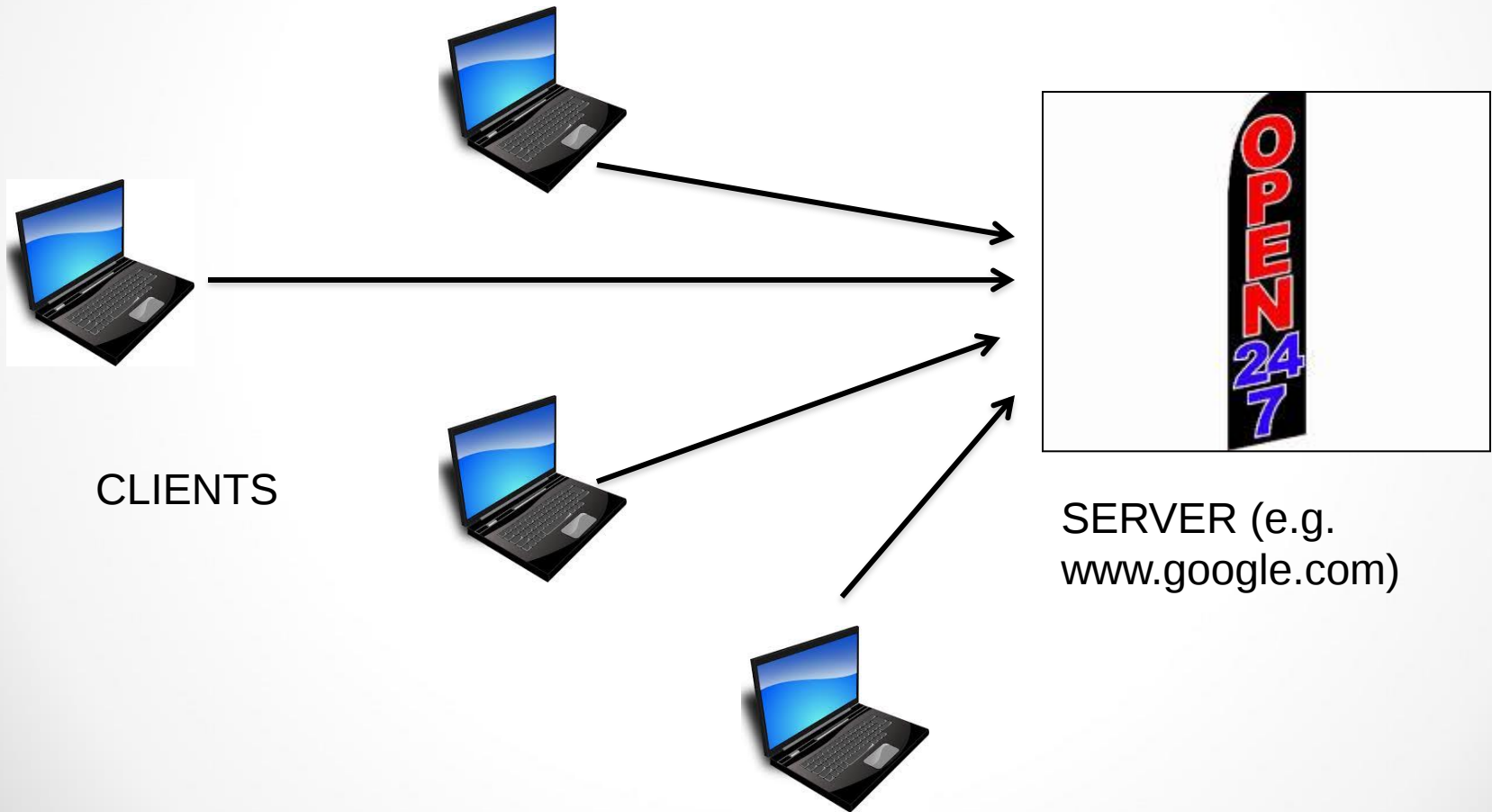
Image:
Wikipedia

Client-server architectures

• • •

web, mail, streaming video, and more

Client-server Architectures



Client-server Architectures

- Architecture: an organizing principle for a computing system
- Most common architecture for Internet applications: *client-server*
- Server is always on, waiting for requests
 - *server software* (e.g. Apache) tells TCP (transport layer software) on its own machine “please listen for messages with port number 80”
 - *client software* (e.g. Chrome) tells TCP “please send this message to machine xxx.xxx.xxx.xxx with port number 80”
 - TCP gives message to IP, which sends it through internet to server machine; IP at server machine delivers to TCP at server machine
 - TCP at the server machine delivers the message to Apache

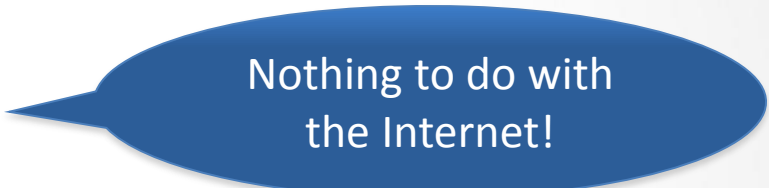
The Web

- World Wide Web = html + http
- html = HyperText Markup Language, an encoding
 - tells what a page should look like and
 - what other pages it links to
- http = HyperText Transfer Protocol
 - agreement on how client and server interact

HTML: an encoding

- Example: using your favorite plain-text editor create the following text file:

```
<html><head>
<title>15110, Spring '14,
Example web page</title>
</head>
<body>
<h1>Hello World!</h1>
</body></html>
```



Nothing to do with
the Internet!

- In a browser type its name in the address bar, e.g.

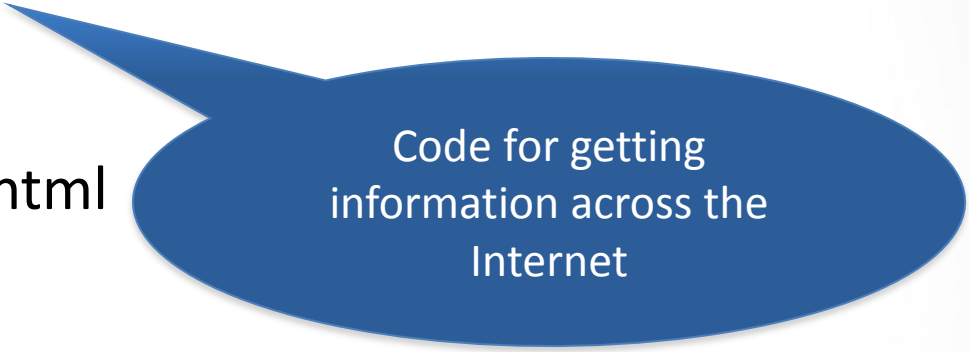
`file:///Users/pennyanderson/CMU/110/week11/example1.html`

HTML: networked hypertext

- Now add

```
<a href=http://en.wikipedia.org/wiki/Hello\_world\_program>  
Hello World!</a>
```

- save as example2.html
and load



Code for getting
information across the
Internet

HTTP: hypertext transfer protocol

- Protocol for communication between web **client** *application* (e.g. Chrome, Safare, IE, Firefox) and web **server** *application* (e.g. Apache)
- Agreement on how to ask for a web page, how to send data entered into a form, how to report errors (codes like *404 not found*), etc.

Uniform Resource Locators

- A Web page is identified by a Uniform Resource Locator (URL)
protocol://host address/page
- A URL

<http://www.cs.cmu.edu/~15110/index.html>

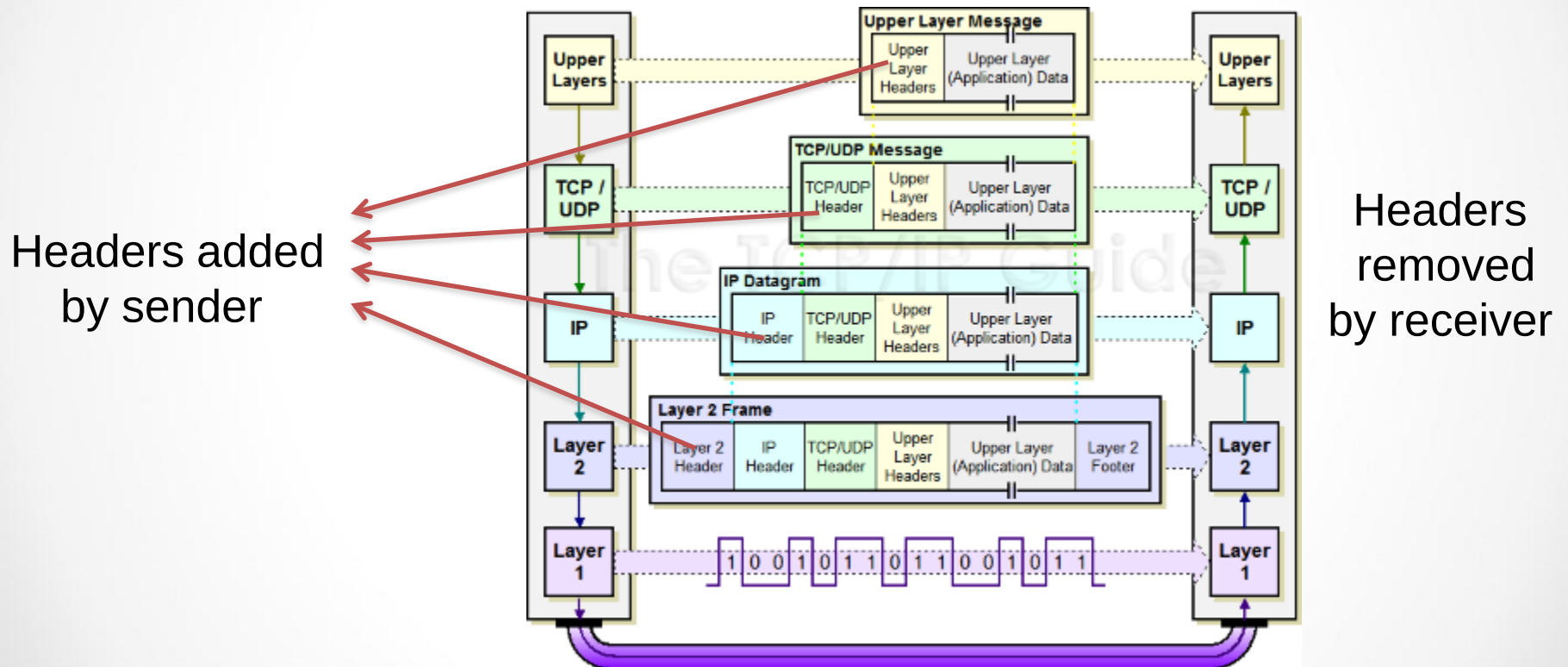


Protocol to use

Overview of web page delivery

1. Web browser (client) translates name of the server to an IP address (e.g. 128.2.217.13) (using DNS)
2. Establishes a TCP connection to 128.2.217.13 port 80
3. Constructs a message
GET /~15110/index.html HTTP/1.1
4. Sends the message using TCP/IP
5. Web server locates the page and sends it using services of TCP/IP
6. The connection is terminated

Layers and Encapsulation



Every protocol defines the layout of a *header*—a kind of envelope or *capsule* for a message

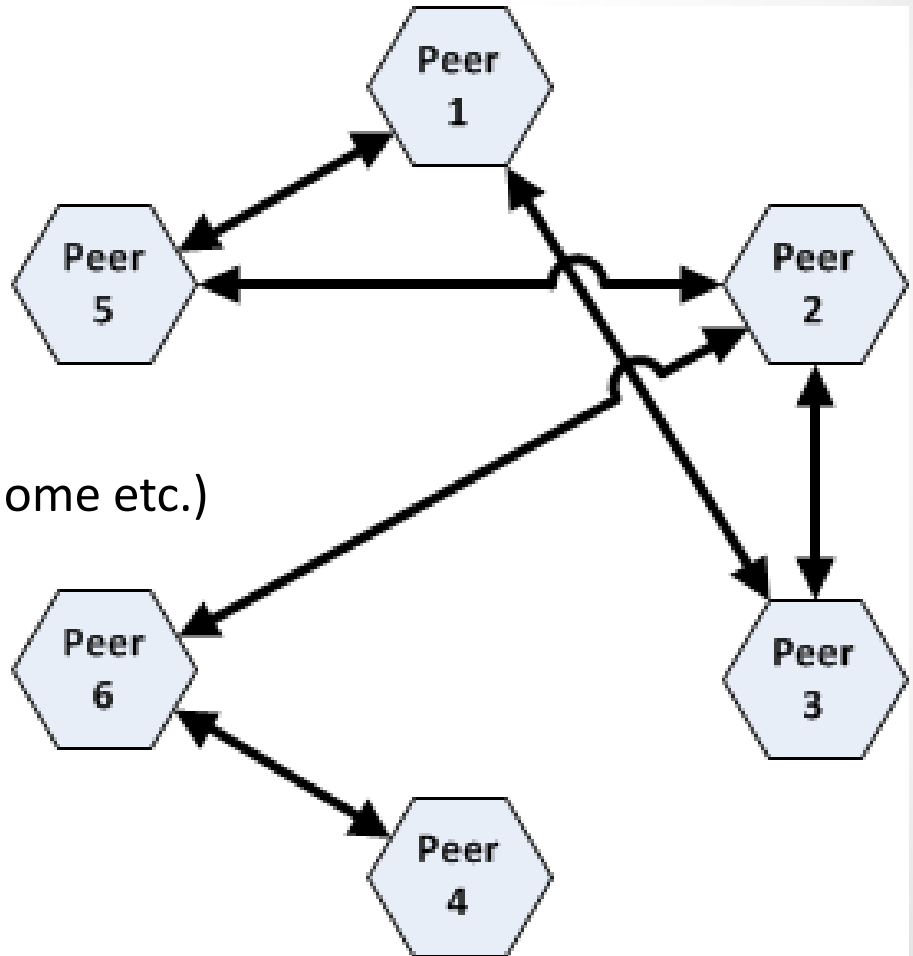
Peer-to-peer architectures

• • •

another widely used organizing principle for Internet applications

Peer-to-peer architecture

- Alternative to client-server
- Applications:
 - file sharing (BitTorrent etc.)
 - streaming media (Skype etc.)
 - Bitcoin
 - volunteer computing (SETI@home etc.)

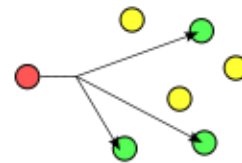


Streaming media

- One of the most demanding applications of the Internet
 - high bandwidth (lots of data really fast)
 - constant delivery rate to the user's screen (headphones, whatever)
 - reliable enough to create a seamless user experience
 - in *real time* for applications like videoconferencing
- *Remember, IP doesn't even guarantee that all packets sent will get there, or when!*

Technology to support streaming media

- Compression
 - reduces amount of data to be sent
- IP multicast
- Adaptive bitrate streaming
 - sends lower-quality images/sound in lower-speed conditions
- Content Delivery Network
 - spreads the work of sending the data over many machines
 - peer-to-peer



Summary

- Applications communicate on the Internet via ***application protocols*** like
 - HTTP for the web
 - SMTP for email
 - RTSP for streaming media
- **Application protocols** rely on
 - Domain Name Servers for name translation, and
 - *transport protocols* like
 - TCP for reliable two-way connections
 - UDP for one-way “datagrams”
- **Transport protocols** rely on IP for packet delivery

Next Time

Network security and cryptography

