The Internet: Protocols and Security

Announcements

- Monday:
  - Lab Exam 2
  - PS 10 due Mon 9:00AM
- Friday: Exam 3

Lab Exam Monday

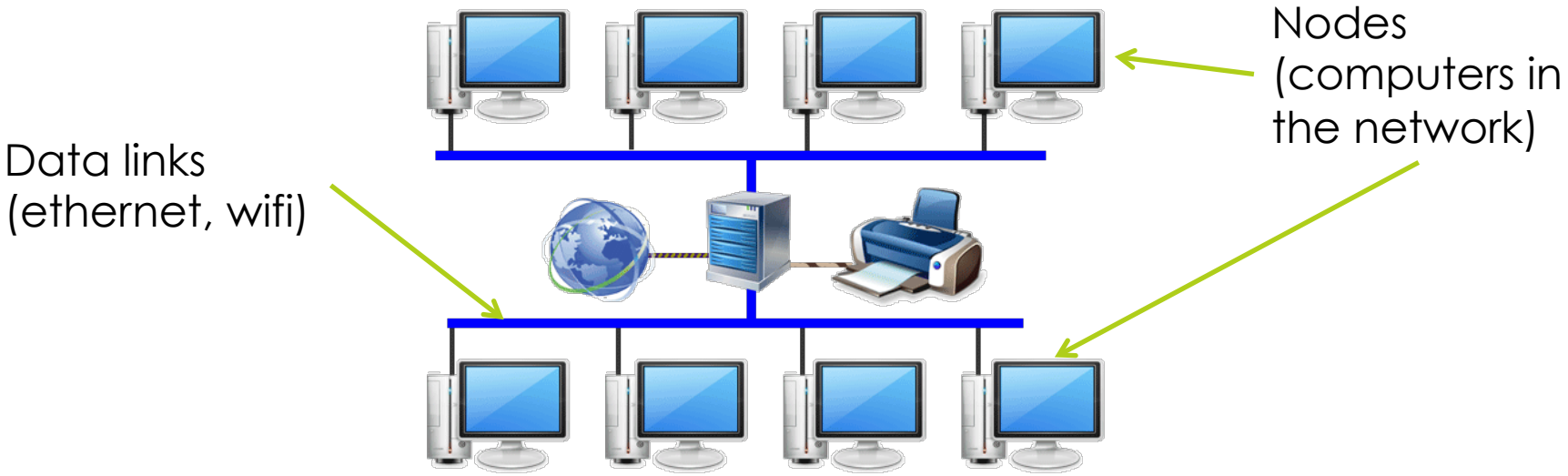
- Bring your laptops
- 4 questions + Reference Sheet
- tkinter
  - Graphics
  - Including geometry
- 2 dimensional data collections
- Recursive functions
- Random functions

Review from Wednesday:

- Computer Networks
  - Protocols
  - Addressing
  - Packet switching
- Some history
- End-to-end principle
- Net neutrality

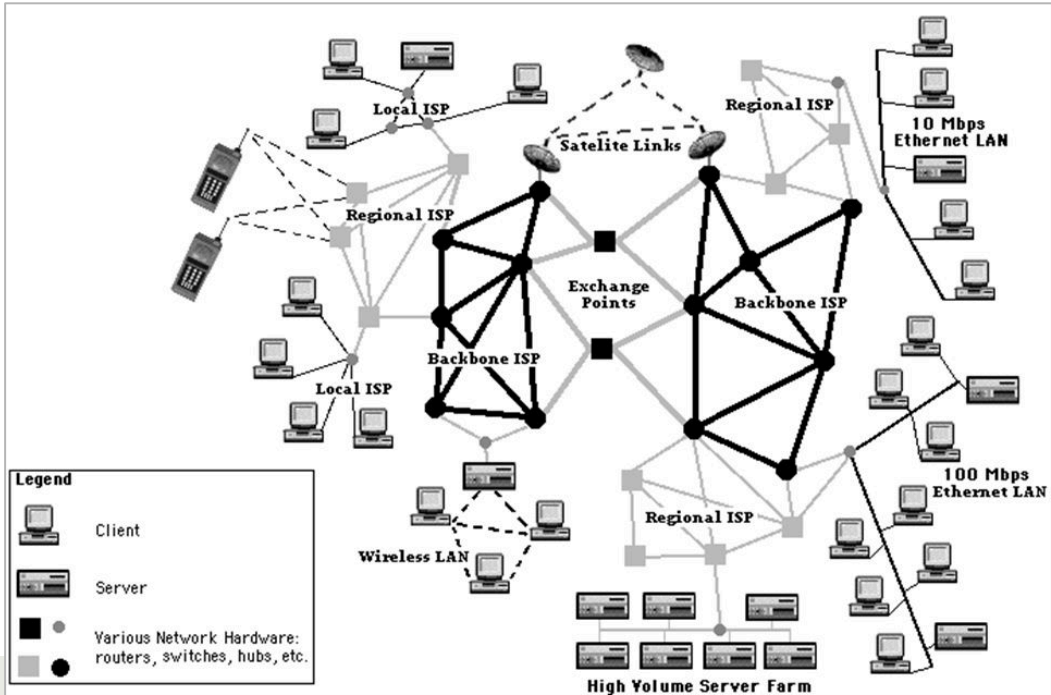
Computer Networks

- A computer network is
  a set of independent computer systems
  connected by telecommunication links
  for the purpose of
  sharing information and resources

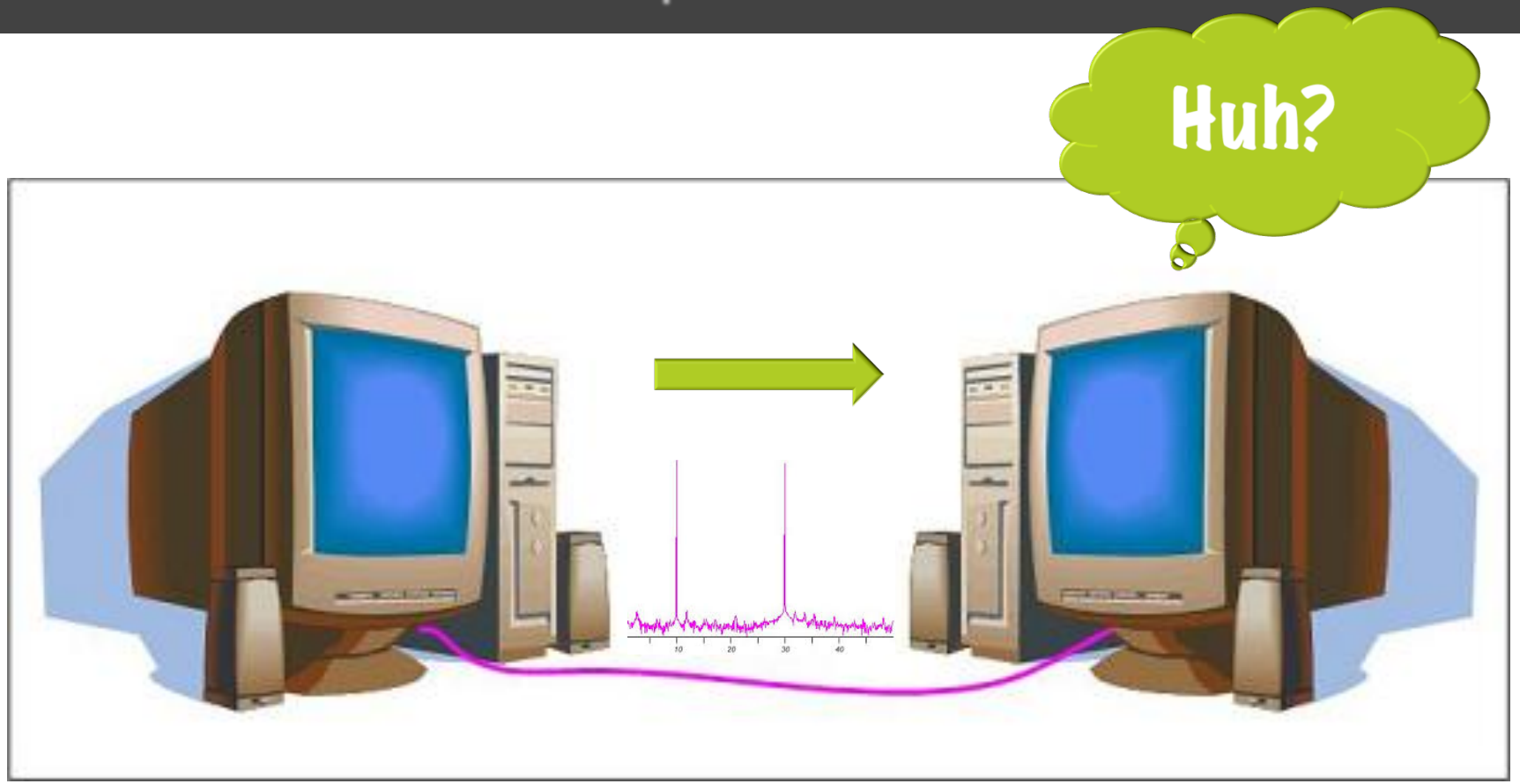


The internet

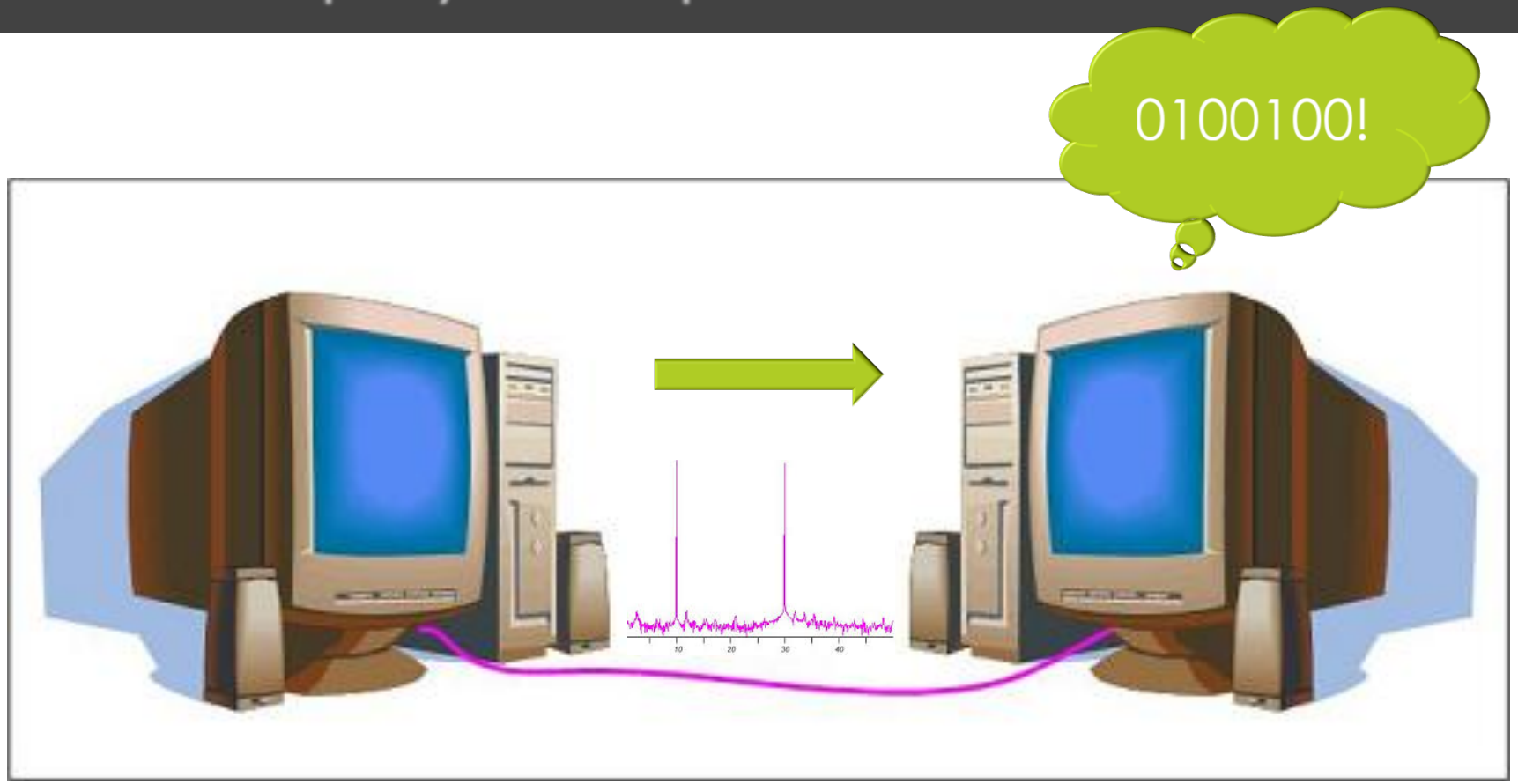
- a global system of interconnected computer networks
- *the biggest computer network of all: the* network of networks



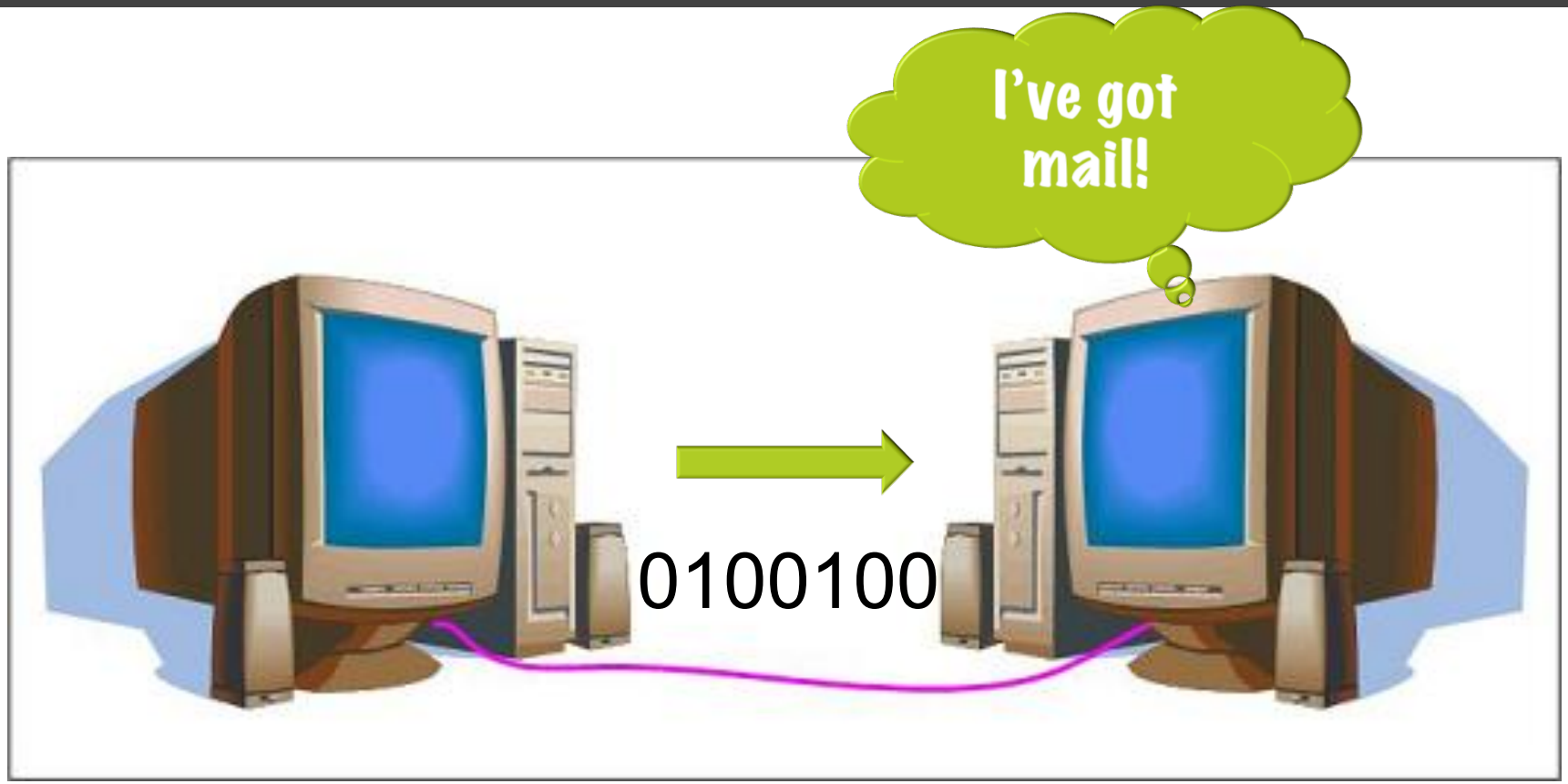
The need for protocols



With physical protocols



With higher-level protocols



IP Addresses

IP stands for "Internet Protocol"

- Each computer on the Internet is assigned an IP Address consisting of four numbers between 0 and 255 inclusive

____ . ____ . ____ . ____

Example: 128. 2. 13. 163

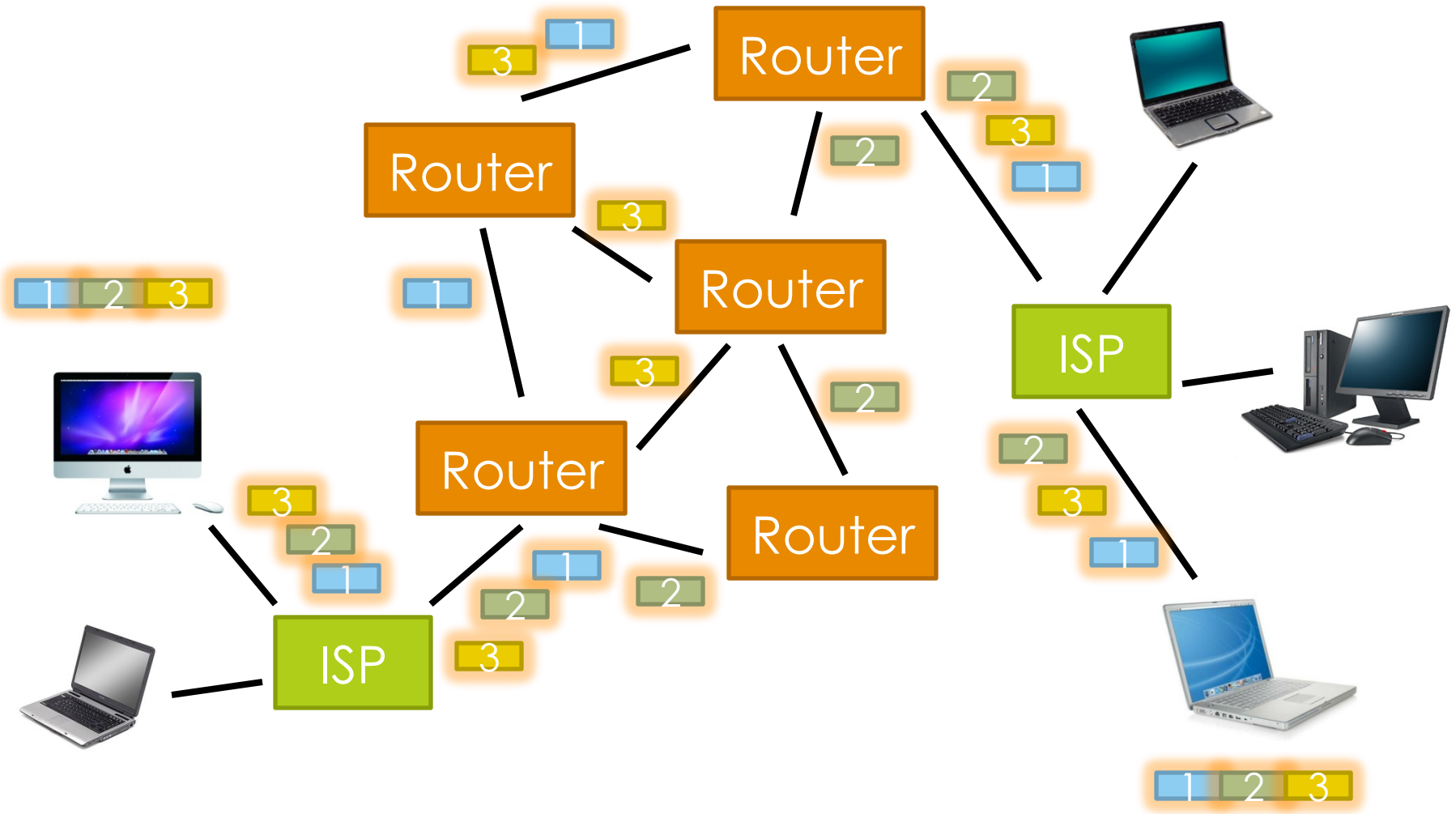
Data sent on the Internet must always be sent *to some IP address*

- How many bits per address?
- How many computers can be on the Internet at the same time?

Packet Switching

- Two network nodes (e.g. computers) communicate by **breaking the message up into small packets**
  - each packet sent separately
  - with a serial number and a destination address.
- *Routers* forward packets toward destination
  - table stored in router tells it which neighbor to send packet to, based on IP address of destination
- Packets may be received at the destination in any order
  - may get lost (and retransmitted)
  - serial numbers used to put packets back into order at the destination

Packet Switching



Routing and Internet structure

- **Core** → provides transport services to edges
  - Routers forward packets
  - Internet Service Providers (ISPs) provide data transmission media (fiber optic etc.)
  - domain name servers (DNS) provide directory of *host* names (more on this next time)
- **Edges** → provide the services we humans use
  - individual users, "hosts"
  - private networks (corporate, educational, government…)
  - business, government, nonprofit services

End-to-end principle
Core architectural guideline

- Idea: *routers should stick to getting data quickly from its source to its destination!*
  - they can be fast and stupid
- Everything else is responsibility of edges, *e.g.*
  - error detection and recovery
  - confidentiality via encryption
  - …

Benefits of End-to-end

- Speed and flexibility
- Support for innovation: routers need know nothing about apps using their services
- Equality of uses: routers can't discriminate based on type of communication (*net neutrality*)

Net neutrality principle

- All communications are treated equally
  - regardless of source, destination, or type

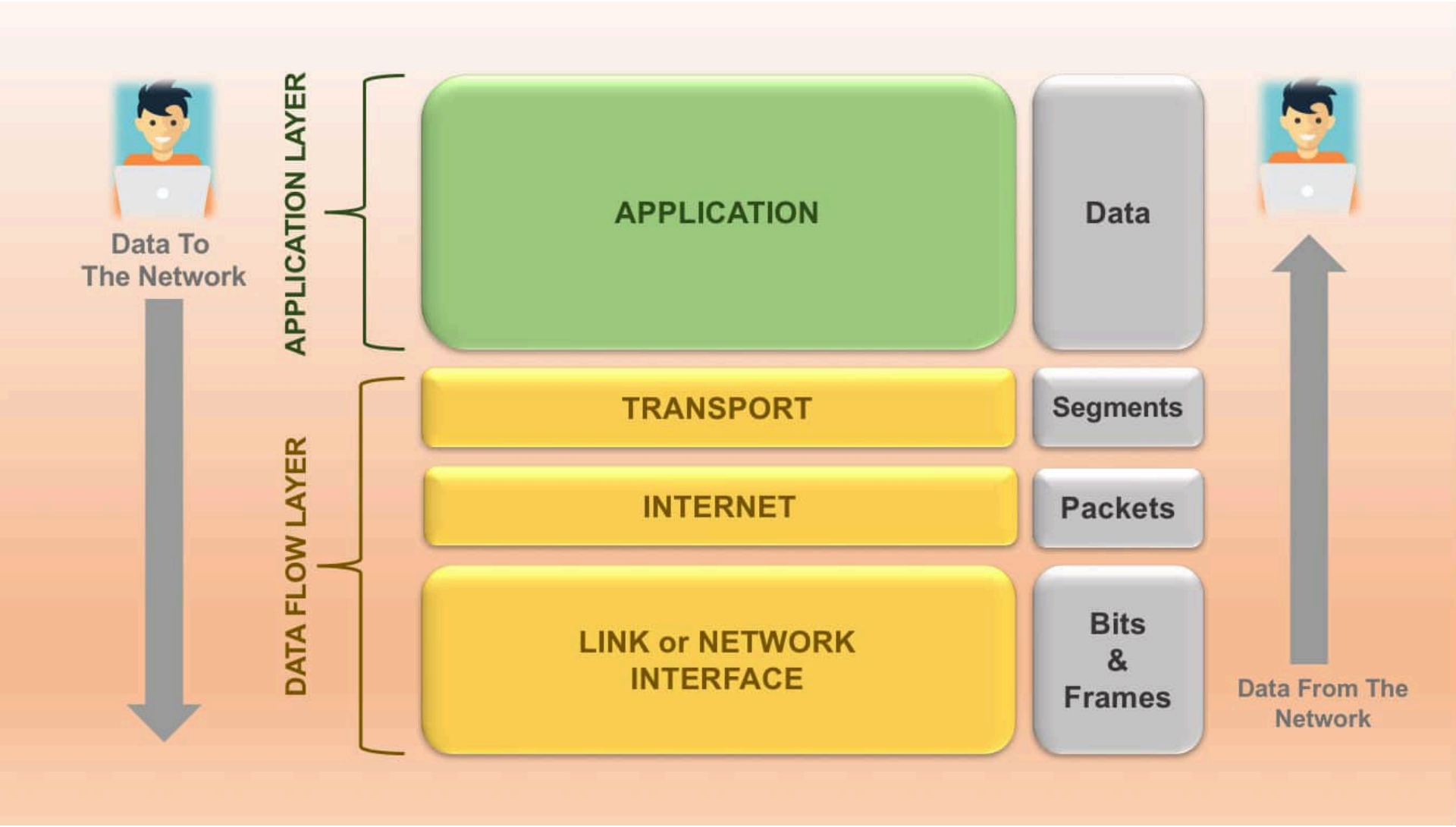
Higher Protocols

"Higher" and "lower" level protocols

- Network protocols are organized in *layers*
  - "Higher" layers use services provided by "lower" layers
  - Each layer is responsible for a type of service

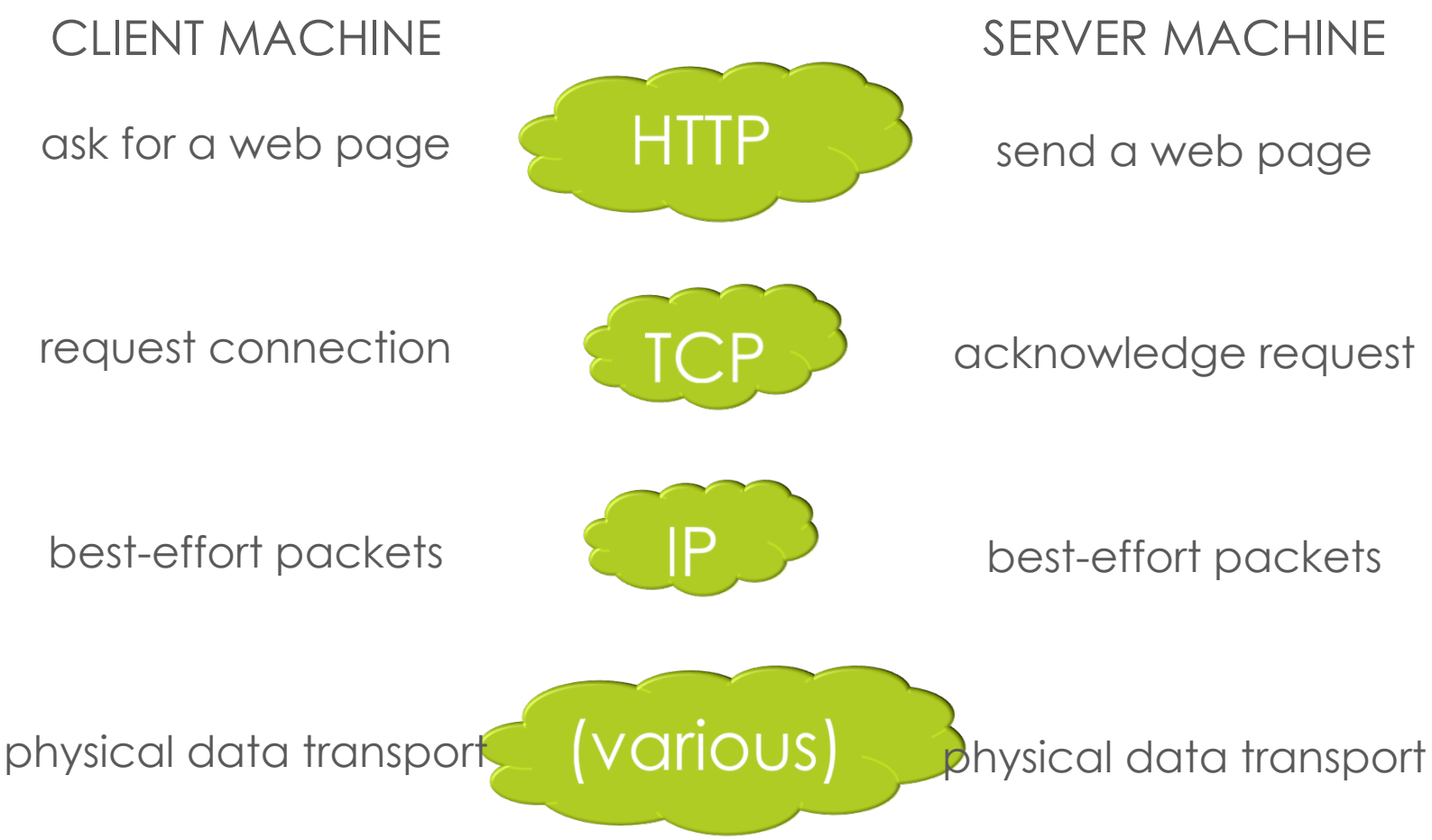
Layers of the Internet ("higher" to "lower")

- Application Layer provides services to human beings
  - *e.g.* browser, email client, Skype
- Transport Layer provides services to applications
  - converts between application messages and IP packets
  - figures out which application to deliver a message to
  - possibly detects and corrects delivery errors
- Internet Layer provides services to transport layer
  - determines next "hop" for a packet and sends it there
- Link Layer provides services to internet layer
  - physically converts between signals and bits



From: https://medium.com/@james_aka_yale/the-4-layer-internet-model-network-engineers-need-to-know-e78432614a4f

Example: Layering the Web



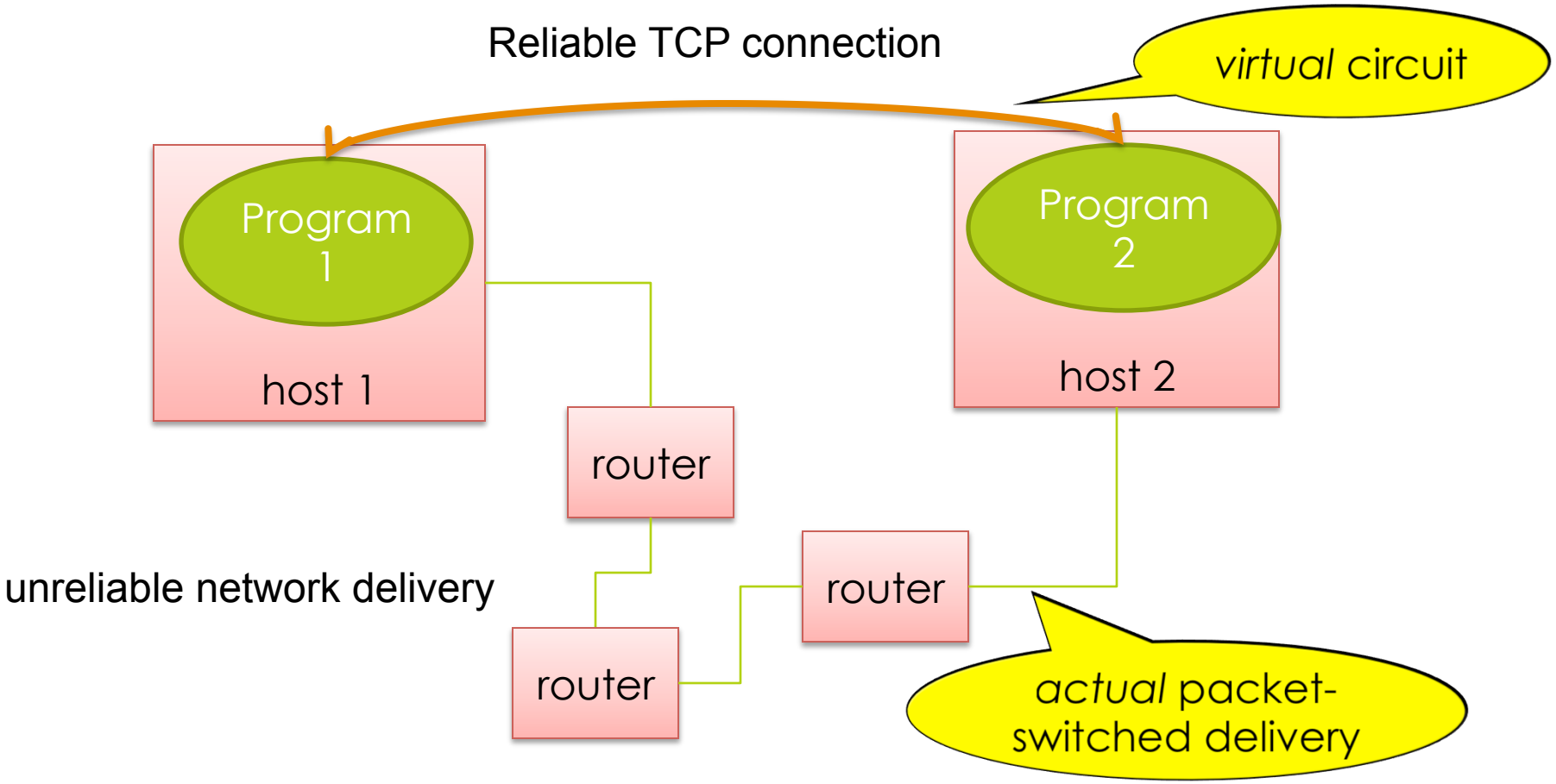
Transport Layer

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.

Domain names

from **98.139.183.24** to **yahoo.com**

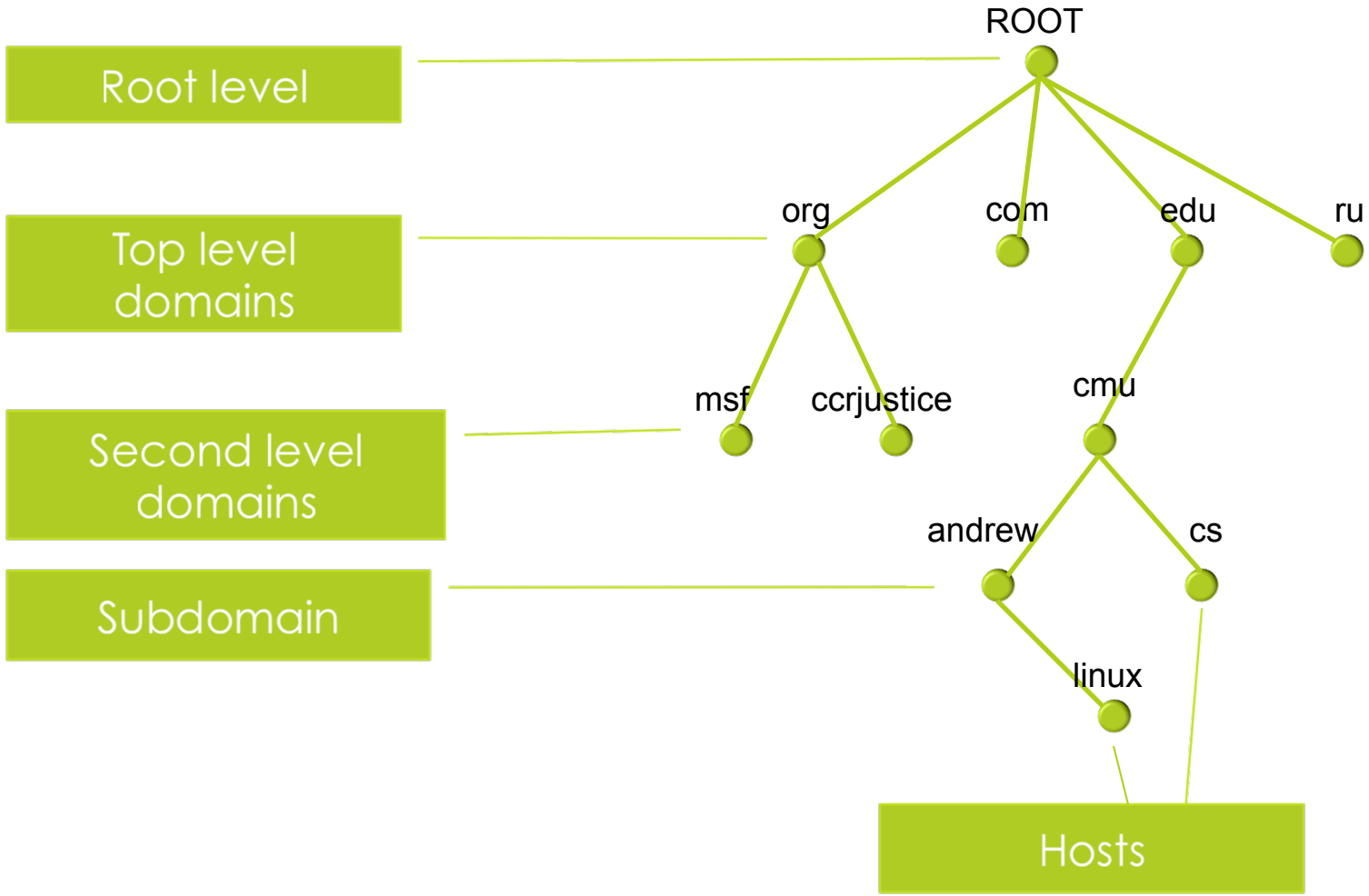
From names to IP addresses

- URL: http://www.andrew.cmu.edu/user/nbier/15110/index.html
- Email address: nbier@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

- 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

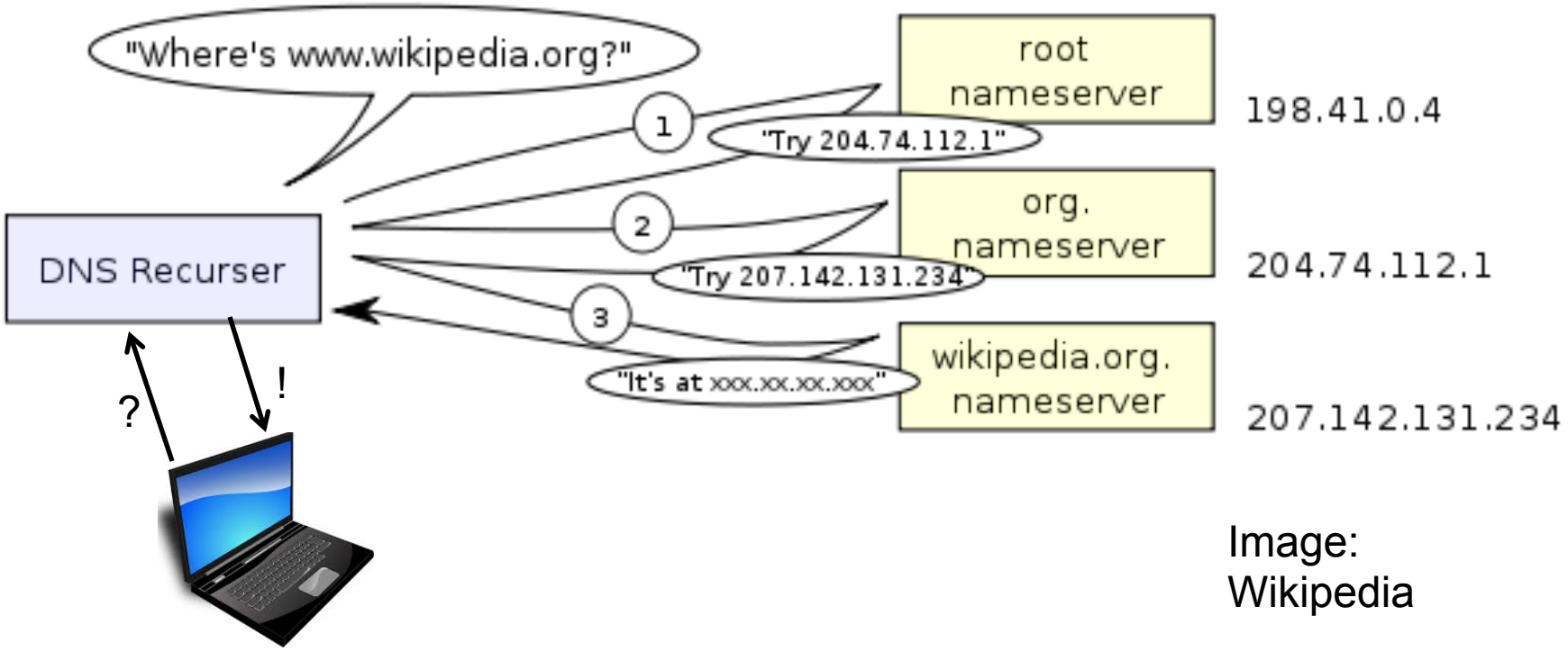
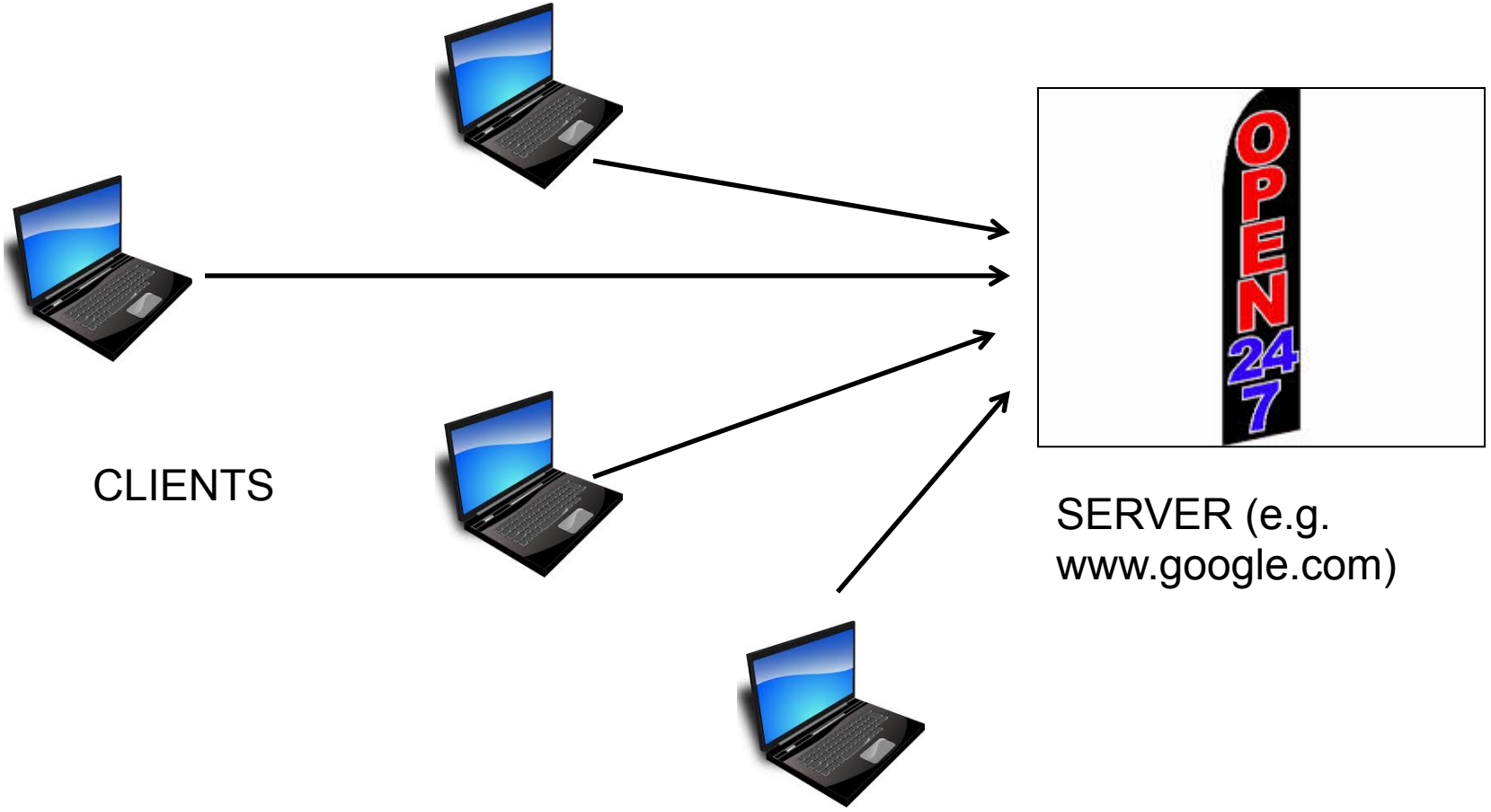


Image: Wikipedia

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, Summer '17,
Example web page</title>
</head>
<body>
<h1>Hello World!</h1>
</body></html>
```

- 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

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

Separate responsibilities

- Message:"GET /~15110/index.html HTTP/1.1"

Request/get web page

- TCP segment:
  control information including sequence number, so-called port number for web server;
  + message

Connect client and server reliably

- IP packet:
  control info including source address, destination address, fragment sequencing information + TCP segment

Best-effort packet switching

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