Week 6 - Wednesday

COMP 3400

Last time

- What did we talk about last time?
- Exam 1!
- Before that:
  - Review
- Before that:
  - Shared memory
  - Semaphores

Questions?

Assignment 4

Networking

Looking back at IPC

- Thus far, we have talked about communicating between processes on the *same* machine
- IPC on a single machine can be used in different ways:
  - Complex applications that are spread into separate processes so that one process crashing doesn't bring everything down
    - Example: Web browser tabs running on separate processes
  - Tying together simpler applications by having them talk to each other
    - Example: Linking together command-line processes with pipes
- Things can go wrong, but communication between processes is fundamentally reliable

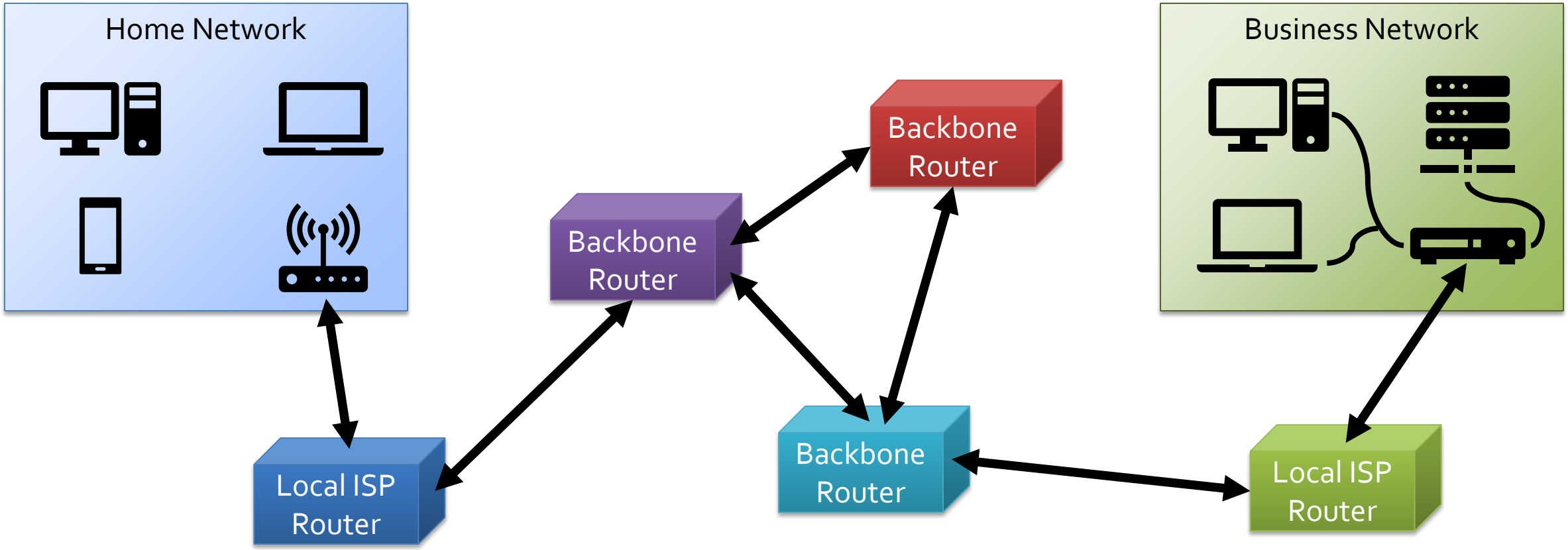
Networking

- IPC can be extended to processes working on *different* machines
  - The goal is still to do more complex computation than would be possible or convenient with a single process
- However, communicating between processes on different machines is much less **reliable**
- This unreliability is particularly strong on the Internet
  - Servers can crash
  - Network outages can cause machines to be unreachable
  - General chaos means that everything can be working reasonably well and yet messages are sometimes lost

Internet

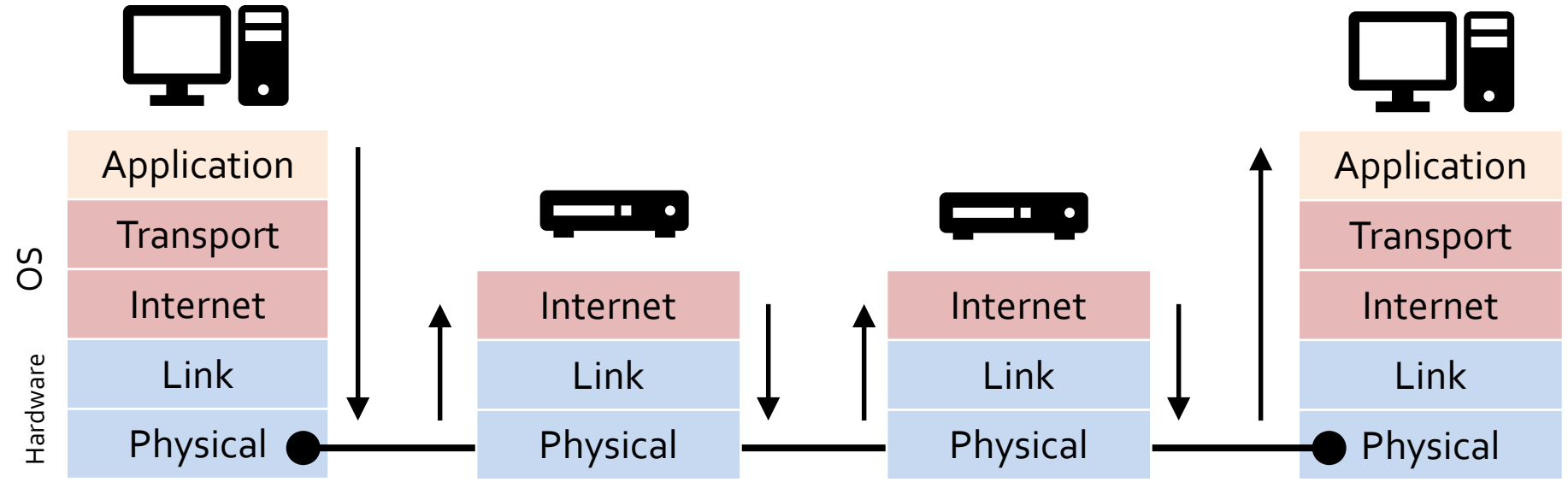
- The Internet is a network of networks
- There are all kinds of protocols for how messages can be sent across it
- Messages:
  - Start with hosts (computers) at the network edge
  - Get forwarded through routers
  - Pass through the backbone of the Internet maintained by large telecommunications companies
    - Backbone providers sell access to Tier 1 ISPs who sell access to lower level ISPs and eventually consumers
  - And eventually reach some other host

Visualization



Layer models

- Networking always involves layers
- Each layer talks to the one above and below it and can often be swapped out for different protocols that provide similar services
- For this class, we'll be talking about a five layer Internet model
  - Simpler than the 7 layer OSI model
  - Remember that the purpose of models is to understand complex systems
- Different people use different names for the same layers



Layers

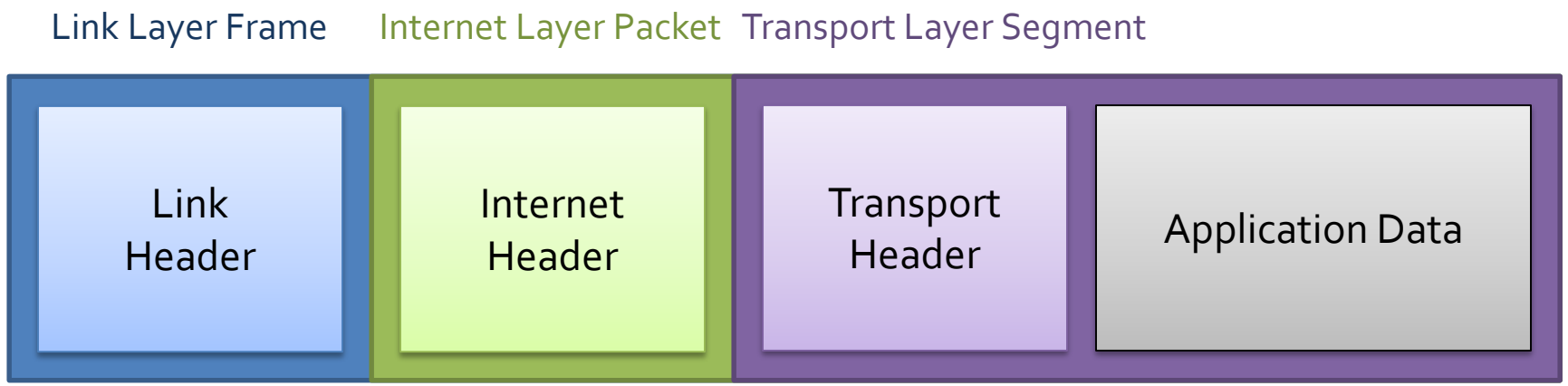
- Application layer
  - Logical endpoints of communication
  - Actual processes that are talking to each other
  - Example protocols: HTTP, FTP, SSH
- Transport layer
  - Implemented as sockets, the software endpoints of communication
  - Provides message passing system calls
  - Breaks down messages into fixed-size segments
  - Demultiplexes: Takes all messages arriving at the machine and sends them to appropriate processes by port number
  - Example protocols: TCP and UDP

Layers continued

- Internet layer
  - Provides point-to-point communication between hosts and routers
  - Uses addresses to determine the logical location of hosts
  - Determines the route that packets will travel along
- Link layer
  - Sends packets between devices on the same network
  - Closely tied to hardware
  - Example protocols: Ethernet, WiFi, Bluetooth
- Physical layer
  - Actual hardware
  - Interprets electrons or radio waves as bits

Packets

- Every system of computer networking we'll talk about uses packets
  - A **packet** is chunk of data with header information like ports and addresses
- Each layer encapsulates the data from the layer above it for transmission
  - The application data is usually bigger than the headers, but we don't draw it to scale
- When being technical, each layer calls its packets different things:
  - Transport layer: **segments**
  - Internet layer: **packets**
  - Link layer: **frames**
- **Datagram** is also used as a synonym for packet
- Be aware that all these terms get thrown around



Network protocols

- The protocol stack has a layered architecture
- Network applications usually use a client-server or a peer-to-peer architecture
- Whether client-server or peer-to-peer, one host creates a server socket to listen and another host creates a socket to connect to the server socket
- The most significant difference is that, in peer-to-peer, hosts are serving as both client and server
- In client-server, the client is often considered untrusted and must go through some authentication to get services from the server

Naming and addressing

- Local IPC used a name that often mapped to a path in the file system
- Networked IPC usually needs more information:
  - Host to connect to
  - Sometimes port
  - File or resource being requested
- A standard form for this information is a uniform resource identifier (URI):

```
URI = scheme:[//authority]path[?query][#fragment]
```

  - Note that brackets mark optional entities

Breaking down the URI

```
URI = scheme:[//authority]path[?query][#fragment]
```

- **scheme**
  - Application layer protocol being used
- **path**
  - Data fields joined together with delimiters
  - File paths are common, using / as a delimiter
- **query**
  - Gives additional user input
  - Example: data that can be used in a database query when dynamically generating a webpage
- **fragment**
  - Customizes the view
  - Example: used to link to a particular part of an HTML page
- **authority**

```
authority = [userinfo@]host[:port]
```

- **host** is a domain name or an IP address
- **port** number shows which process to contact
- **userinfo** gives user information like account name

Examples

scheme — path — fragment
http://localhost:8080/helloworld?user=alice#top
authority — query

scheme
mailto:dbowie@gmail.com
path

scheme
news:comp.systems.networks.uri.announcements
path

RFCs

- The Internet Engineering Task Force (IETF) is the body in charge of the standards for Internet communication protocols
- The standards are defined with community involvement in documents called **requests for comment** (RFCs)
  - Downloadable here: https://tools.ietf.org/rfc/
- RFCs are the official standards for how things are supposed to work
  - Some things don't have RFCs
  - Big companies often extend the protocols or violate them

RFC	Purpose
768	UDP: unreliable transport layer protocol
791	IPv4: network layer protocol (version 4)
793	TCP: reliable transport layer protocol
959	FTP: file transfer protocol
1034, 1035	DNS: domain name translation database
2026	Defines the RFC process
2131	DHCP: dynamic IP addressing
2616	HTTP/1.1: serving web pages
3986	Structure and interpretation of a URI
8200	IPv6: network layer protocol (version 6)

Backus-Naur Form

- RFCs often include descriptions of messages or protocols using Backus-Naur Form (BNF)
- BNF is a way to express context-free grammars
- A grammar is a list of rewriting rules showing the ways a structure can be broken down into simpler pieces
- Special syntax:
  - | represents a choice between two alternatives
  - * represents o or more repetitions (sometimes put in front or in back, depending on the BNF style)
  - Brackets ([ ]) represent optional items
- (Partial) example for English:
  - <sentence> ::= <subject> <predicate>
  - <subject> ::= <noun> | <pronoun>
  - <noun> ::= "wombat" | "bear" | "car" | …
  - <pronoun> ::= "he" | "she" | "it" | "they" | "we"
  - <predicate> ::= <verb> | <verb> <noun>
  - <verb> ::= "sings" | "walks" | "murders" | …

BNF for HTTP

- Here's part of the BNF for HTTP messages:

```
HTTP-message     = Request | Response
Request          = Request-line
                   *(( general-header
                    | request-header
                    | entity-header ) CRLF)
                   CRLF
                   [ message-body ]
Request-Line     = Method SP Request-URI SP HTTP-Version CRLF
```

Example HTTP message

- This HTTP message makes a GET request for **`index.html`**

```
GET index.html HTTP/1.0
Accept-Charset: iso-8859-5, Unicode-1-1
Date: Tue, 19 Nov 2018 08:12:31 GMT
Accept-Encoding: *

```

- It also follows the BNF on the previous slide:
  - A request line
  - Three headers (followed by a **`CRLF`** each)
  - No message body (because that's optional)
  - A final extra **`CRLF`** at the end

Upcoming

Next time…

- Sockets

Reminders

- Work on Assignment 4
  - Due next Monday
- Read section 4.4