Operating Systems

- File Systems
  - Reading: http://www.cs.berkeley.edu/~brewer/cs262/FFS.pdf

- Scheduling
  - Processes: where do they come from?

- Bootstrapping
  - How does the system start?

Networking

- How do computers communicate?
  - Two computers connected by a direct wire?
  - Relatively straight forward: move bits across wire
  - Internet?
  - Many computers
  - All around the world
  - With other communications going on...
  - And un-reliable links
  - And tons of different systems, media, protocols...
  - Pretty complicated, so how could we possibly manage it?

- Abstraction
  (oh right, the answer to like...everything)

7-layer OSI model

1 Physical Layer

- Defines physical how physical media work
  - Pin layout
  - Voltages
  - Timing Requirements

- Examples:
  - Cat 5 Cable ("Ethernet Cable")
  - Wireless radio signal specifications

- Not much interesting to say here
2 Data-link Layer

- How to move bits across the wires in a meaningful way
- Communication between two computers on same physical network
- May include some error checking
- Example: Ethernet
  - Data transmitted in frames
  - Frame has:
    - Pre-amble: used to detect collisions
    - Header: source and destination MAC address
    - Payload: actual data
    - CRC check: detect corrupt data
- Carrier Sense, Multiple Access, Collision Detect (CSMACD)
  - Carrier Sense: listen for if anyone else transmitting
  - Multiple Access: can wire up many computers to it
  - Collision Detect: two transmissions at once? Detect and retry
- Ethernet uses CSMACD for multiple systems on a network
- Detection of collisions?
  - Pre-amble is fixed pattern
  - Network card senses medium while transmitting
  - Mismatch with expected? Collision
  - Collision happens?
    - Exponential backoff
    - Pick random number of time units
    - Retry
    - Fail again? Pick random number from 2x as big a range
- Analogy: crowded dinner party
  - Try to talk. Someone else talking? Wait. Try again. Fail again? Wait longer

Abstraction: Joys and Limitations

- Joys of abstraction:
  - Can build an Ethernet card without any info about higher layers
  - Will work with all of them
- Limitations:
  - 7-layer model’s abstraction not perfect
  - Ethernet protocol imposes max limit on cable length
    - E.g., layer 2 constrains layer 1
    - This arises from the need to detect collisions before finishing sending

3 Network Layer

- Layer 2 let’s computers on same network talk
- Layer 3 let’s computers talk across networks
  - Addressing
  - How do we specify what computer to talk to?
  - Routing
  - How do we get from here to there?
- Example: IP protocol
  - IPv4 and IPv6: pretty similar in most core regards
  - Best effort delivery
  - Addressing (IP addresses)
  - Routing
  - Analogy: Mailing a letter

Our messages so far

- Header says what network layer protocol the payload is
Computer 1 wants to send data to Computer 2
- For now, assume it knows IP address (we'll see DNS later)
- Has direct connection to its ISP... but then what?
  - The internet is a big place after all...

IP Routing
- IP addresses are hierarchical
  - May not know how to find 74.125.130.105...
  - But know which way to go to get to 74._______
  - Move one step closer
  - Within 74 network, know how to find 74.125
  - Then 74.125.130
  - Then find 74.125.130.105
- Analogy:
  - How do I get to 2200 Mission College Blvd, Santa Clara, CA?

Routing Basics
- Routing is done with tables
  - CIDR notation: 40.1.0.0/16
  - Match first 16 bits of 40.1.0.0, ignore remaining 16 bits
  - Each number in IP addr is 8 bits, written in decimal
  - Find match, entry tells what link to send out on
  - Example
    - 40.0.0.0/8 => Link 0
    - 50.1.0.0/16 => Link 1
    - 50.2.0.0/16 => Link 2
    - 50.3.20.0/24 => Link 3
    - 50.3.42.0/24 => Link 1

Routing: More Complex
- Approach one: Static Routing
  - Enter all routes
  - Let system run
  - Hope nothing goes down
  - Works fine for small networks
- Reality:
  - Network links/systems go down
  - Often multiple paths to same place
  - Changing traffic patterns = changing fastest route
**Distance Vector Protocols**

- Routers
  - Know distances to immediate neighbors
  - Compute distance vector
    - How far to any destination from all known info
  - Transmit distance vectors to neighbors
  - Now know more info, so repeat process

**Dynamic Routing**

- A, B, C, D are endpoints
- 1, 2, 3, 4 are routers
- Edge labels are delay (lower is better)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5 =&gt; A</td>
<td>3 =&gt; C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8 =&gt; 1</td>
<td>4 =&gt; 3</td>
<td>6 =&gt; 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 =&gt; B</td>
<td>1 =&gt; 8</td>
<td>1 =&gt; 6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6 =&gt; 4</td>
<td>3 =&gt; 4</td>
<td>4 =&gt; 5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8 =&gt; 2</td>
<td>2 =&gt; 3</td>
<td>7 =&gt; 3</td>
<td></td>
</tr>
</tbody>
</table>

- Note that routes may improve after a path is discovered
  - May find one long path, then a shorter path later
  - This is fine: tables get updated, and continue to propagate
  - But what if...
Cable to D gets snipped?

• Cable to D gets snipped?

But 3 advertises a route of length 3, so I can get there in 4?

Also, 3 will now revise its estimate for D to 5 (4 + 1) [and 4 updates too]

5 then updates from revised data

And so on…
Count-to-infinity

- Algorithm will slowly "count to infinity"
  - Actually: count to max value it holds
  - Then throw away the route, concluding there is no path there
- Packets sent in meanwhile?
  - IP: Time To Live (TTL)
  - Starts at fixed value (e.g., 255)
  - Decremented every time the packet is forwarded
  - Packet dropped when TTL == 0
- Traceroute: uses TTL fields to probe to different distances
  - Uses ICMP protocol to get response on TTL
- Note: many fancier routing schemes, we aren’t covering

7-layer OSI model

- 7 Application Layer
- 6 Presentation Layer
- 5 Session Layer
- 4 Transport Layer
- 3 Network Layer
- 2 Data-link Layer
- 1 Physical Layer

- IP: hierarchical addresses + best effort delivery

Our messages so far

- IP Header has
  - Src IP
  - Dest IP
  - Payload type (what protocol)
  - Other info
- Side note: IP does fragmentation to fit within frame size
  - We aren’t covering that

4: Transport Layer

- Reliability (if used)
  - Acknowledgements of data receipt
  - Retries of failed data
- Flow Control
  - Restrict rate of data sending
- Multiplexing/De-multiplexing data
  - E.g., Ports: identify which program some data is for
  - Keep data streams separate

5: Session Layer

- Concept of "a connection"
  - Establish/terminate
  - (OSI includes a variety of obscure features not often used)
- TCP: combines these two layers together
  - Sets up/terminates sessions
  - Has sequence numbering for packets
  - Acknowledges (ACKs) packets that are received
  - Establishes flow control (responds to congestion by throttling sending)

TCP

- We’ll draw diagram with computers on each side
- Time goes down
- Three messages above (TCP’s "3 way handshake")
To open a new connection:
- 1 computer sends SYN ("Hi, lets talk")
  - All messages have sequence numbers including SYN
  - First sequence number of a new connection is random
  - TCP sequence numbers by byte

Other computer
- ACKs (Acknowledges) the message (says what sequence # it ACKs)
- Also sends SYN "Hey sure, lets talk"

First computer then ACKs this SYN
- And probably sends data along with the ACK
- TCP control info (SYN, ACK, FIN): bits in TCP header
  - Packets can have multiple control bits on + carry data

Data going right in blue
- ACKS coming left in green
  - note: ACK # ed by expected next data
- Sliding window (flow control)
  - Limit amount of un-ACKed data at a time

Data may get re-ordered in network
- One packet takes one route, another takes another
- TCP: no problem
  - Sender re-orders data properly
  - Sends ACK for as much data as it has

Data may also get lost in the network
- E.g., router is backlogged, can’t handle it has to drop from queue
- TCP will re-send un-ACKed data after a timeout
TCP: Duplicate Data

- Receiver may get duplicate data
  - 7000-7499 gets lost
  - But 7500-7599 arrives
  - Then sender re-sends both: no ACK for either (why not?)
  - No problem: receiver drops duplicate (can tell: sequence #)

TCP: Closing connection

- Connection closed with FIN message
  - Receiver ACKs
  - Other side may close (with FIN) [typical]
    - Or remain open: can still send data
    - Side that closed cannot send, but should receive/ACK
      SYN/ACK may be one message

Flow control: Sliding Window

- Problem:
  - Congestion -> dropped packets
  - Dropped packets -> Retries
  - Retries = duplicates of data -> More congestion
    - Vicious cycle...
- TCP implements flow control with a sliding window
  - Limitation of amount of un-ACKed data out at a time
    - Retry required? Shrink window
      - Assumes congestion, tries to avoid it
    - No retries in a while? Grow window back
      - Maybe it cleared up?

TCP: Closing connection

- What if ACK for FIN gets lost?
  - FIN gets retried... but other side expects connection is closed?
  - TCP has a state to handle this
    - Connection expected to be closed, but resources/state still held
    - Times out if no activity (assumes ACK got through if no retry)

7-layer OSI model

- TCP: It's the coolest thing since memory got sliced into pages!

Our messages so far

- TCP header has
  - Source/Dest port
    - Sequence numbers
  - Control bits (SYN/ACK/FIN)
    - Check sum over data
    - Other stuff

Preamble Header Header Header Payload CRC
6: Presentation Layer

- Responsible for data formats
  - Examples
    - Character encoding schemes
    - Serialization of objects
  - We're not really going to talk about it much

7: Application Layer

- Protocol specific to how applications want to communicate
  - Examples:
    - http
    - ftp
    - ssh
    - aim
    - SMTP
    - POP
    - ...
  - Again, not going into this much...

7-layer OSI model

- Flexibility Example: Wired vs Wireless
  - Change out two layers, rest stay the same

Network programming

- Coding networking code in...
  - Java: Look in java.net, start with Socket
  - C:
    - socket()
    - connect()
    - accept()
    - bind()
    - listen()

When I say FIN, you say....?

FIN
When I say FIN, you say....?

ACK

Summary:

- Networking Overview
  - 7-layer model
  - Emphasis on IP (Layer 3) and TCP (Layers 4 and 5)
  - Not comprehensive, but...
  - You are now at least conversant enough to discuss the OSI stack at parties