Logistics

• Proj 3 Due Tonight, 11:59 PM
• HW3 Due Tomorrow, 11:59 PM
• Course Evals Fixed
• Review Session for Wednesday's Lecture
• Final Exam: Thursday, 5-8PM
Recall: Two-Phase Commit

• We can’t solve the General’s Paradox
  • No simultaneous action
  • But we can solve a related problem

• Distributed Transaction: Two (or more) machines agree to do something or not do it atomically

• Extra tool: Persistent Log
  • If machine fails, it will remember what happened
  • Assume log itself can’t be corrupted
Recall: Two-Phase Commit

- One machine (*coordinator*) initiates the protocol
- It asks every machine to *vote* on transaction

- Two possible votes:
  - Commit
  - Abort

- Commit transaction only if *unanimous* approval
Recall: Coordinator’s State Machine

- **INIT**
  - **WAIT**
    - **ABORT**
      - **COMMIT**

  - **Send**: VOTE-REQ

  - **Recv**: START
    - **Send**: VOTE-REQ

  - **Recv**: VOTE-ABORT
    - **Send**: GLOBAL-ABORT

  - **Recv**: all VOTE-COMMIT
    - **Send**: GLOBAL-COMMIT
Recall: Worker’s State Machine

INIT

READY

ABORT

COMMIT

Recv: VOTE-REQ
Send: VOTE-ABORT

Recv: VOTE-REQ
Send: VOTE-COMMIT

Recv: GLOBAL-ABORT

Recv: GLOBAL-COMMIT
Recall: Blocking

• What if both coordinator and a worker fail?
• The remaining workers can still consult each other
• But they can’t reach a conclusion on what to do!

Why?
• If all remaining workers in INIT, we still don’t know state of failed worker $w$
• $w$ may have been first to be notified of a commit, and then coordinator and $w$ crashed
Distributed Consensus

• Two- and Three-Phase commit make a decentralized decision
• Example: Changing the value of a key among all replicas for the key

• But they are hardly the only solutions to this problem
Better Agreement in Face of Failure

• Idea: If a **majority** of nodes agree, **commit**
• If a minority don’t participate, ignore them

• Assumes a *fail-stop* model: if a machine breaks, it won’t send/receive messages

• Algorithms that do this: **Paxos, Raft**
Beyond Fail-Stop

• What if nodes don’t just stop talking when they fail?
• What if they send incorrect information?
• Or what if nodes are actively malicious?

• This is called the Byzantine Failure Model
• Do Paxos/Raft still work even if just a minority of nodes exhibit Byzantine failures? No
CAP Theorem

• Originally proposed by Eric Brewer (Berkeley)

1. **Consistency** – changes appear to everyone in same sequential order

2. **Availability** – can get a result at any time

3. **Partition Tolerance** – system continues to work even when one part of network can't communicate with the other

• Impossible to achieve all 3 at the same time (pick two)
CAP Theorem

- What do we do if a network partition occurs?
- Prefer Availability: Allow the state at some nodes to disagree with the state at other nodes (AP)
- Prefer Consistency: Reject requests until the partition is resolved (CP)
Example: Two-Phase Commit

• When a partition occurs, 2PC is:

• **Consistent:** Reads never return wrong values

• **Not Available:** Writes block until partition is resolved and unanimous approval is possible

• So 2PC is a **CP** system
What about AP Systems?

• Partition occurs, but both groups of nodes continue to accept requests

• Consequence: State might diverge between the two groups (e.g., different updates are executed)

• When communication is restored, there needs to be an explicit recovery process
  • Resolve conflicting updates so everyone agrees on system state once again

• Relevant Systems: Dynamo, Coda File System
Recall: Networking Challenge

• Many different applications
  • Email, Web, Online Games, etc.

• Many different network styles and technologies
  • Wireless, Wired, Optical, etc.

• How do we organize all of this complexity?
Layering

• Complex services from simpler ones
  1. Physical and Link Layers (Wi-Fi, Ethernet, …)
     • Unreliable, local exchange of limited-size frames
  2. Network (IP) – routing between local networks
     • Unreliable, global exchange of limited-size packets
  3. Transport (e.g., TCP) – Glue
     • Reliable (with retries), ordering, stream of bytes
  4. Application – Everything on top of sockets
The Internet Hourglass

The Hourglass Model

Applications
Transport
Data Link
Physical

The Hourglass Model

 protocols:

SMTP
HTTP
DNS
NTP
TCP
UDP
IP

 physical:

Ethernet
SONET
802.11

 Copper
Fiber
Radio
Implications of Hourglass

- There is only **one** Network-Layer Protocol: **IP**
- Allows networks to interoperate
- Above IP: Applications function on all networks
- Below IP: Change network’s construction without disturbing applications
- One drawback: Changing IP itself (e.g. transitioning to IPv6) very involved
Sockets

• Programmer's interface to transport layer

• Looks just like a file with a file descriptor
  • read adds to queue, write removes from it

• Same abstraction for any kind of network

• This class has covered TCP sockets
Recall: HTTP

- Application protocol for The Web
  - Retrieve a specific object, upload data, etc.

- Runs on top of TCP (sockets)

- Like any protocol, stipulates:
  - **Syntax**: Content sent over socket connection
  - **Semantics**: Meaning of a message
    - Valid replies and actions taken upon message receipt

- Arguably a form of RPC
HTTP Messages

• Text-based: We just send character strings over our TCP socket connection
• To make a request, browser might write something like the following on a socket:

GET /hello.html HTTP/1.0
Host: 128.32.4.8:8000
Accept: text/html
User-Agent: Chrome/45.0.2454.93
Accept-Language: en-US, en; q=0.8
Reliable Byte Streams: TCP

- Sliding window of unacknowledged packets
- Deliver to user process in correct order
- Resend if no acknowledgement received

Sequence Numbers

- Sent
  - Sent
  - not acknowledged
  - Not yet sent

- Received
  - Received
  - Buffered
  - Not yet received

Sender

Receiver
Recall: Using Acknowledgements

- Checksum: Detect garbled packets
- Receiver sends a packet to acknowledge when a packet received and ungarbled
  - No acknowledgement? **Resend** after timeout
- What if acknowledgement dropped?
  - Packet is resent (wasteful), second chance to acknowledge
Window-Based Acknowledgements

• Send up to $N$ packets without ack
• Both source and destination need to store $N$ packets
• Each packet has sequence number

\[ \text{Queue} \]

\[ \begin{align*}
N &= 5 \\
pkt\#0 &\quad \text{pkt#4} \\
ack\#0 &\quad \text{ack#4}
\end{align*} \]
Internet Protocol Features

• Routing – an IP packet goes anywhere
  • Just need the destination IP address

• Fragmentation – split big messages into smaller pieces
  • Think about downloading a file
  • Maximum size 64K
  • Reassemble at destination
  • Hides differences in physical layers

• Multiple protocols running on top
  • ICMP, TCP, UDP, …
IP Addresses

• In IPv4: a 32-bit integer used as the destination of an IP packet
  • Often written as four dot-separated integers (0-255)
  • Example: DNS for "eecs.berkeley.edu" resolves to 23.185.0.1

• Extended to 128 bits in IPv6
Internet: Hierarchical Network

• Not every host connected to every other one
• Use a network of Routers to connect subnets together
IP Address Subnets

• With IP, all addresses in same subnet share same prefix of bits
  • e.g. all hosts have addresses of the form 128.32.x.y

• **Mask:** Number of matching prefixing bits
  • Expressed as single number, e.g. 24
  • Or all matching bits set to one address notation, e.g. 255.255.255.0

• Subnet is identified by 32-bit value, with all bits that differ among machines set to 0, followed by slash and mask, e.g. 128.32.0.0/16
Special Reserved Subnets in IPv4

• **127.0.0.0/8**: loopback (same machine)
  • **127.0.0.1**: Localhost

• **10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16**: private
  • Not available globally
Hosts in IP

- **Internet Host**: A computer connected to the internet, assigned one or more IP addresses for routing

- Each host does not necessarily have a globally unique IP address
  - Groups of machines may share an IP address
  - In this case, machines have private addresses behind a **Network Address Translation Gateway**
  - Very typical for residential networks
Simple Network Terminology

• **Local Area Network (LAN):** Designed to cover a small geographical area
  • Topology: multi-access bus, ring, or star network
  • Very fast communication speeds (e.g. 1 Gbps)
  • Fast and cheap to broadcast to all nodes

• **Wide Area Network (WAN):** Links geographically separated sites
  • Point-to-point connections over long-haul lines
  • Slower communication speeds
  • Broadcast usually requires multiple messages
Wide Area Networks

If you want to know more: https://www.submarinecablemap.com/
IP Routing

• **Routing**: The process of forwarding packets hop by hop through routers to reach their destination

• Need more than just a destination address!
  • Need a path

• Think about an Amazon package: destination address in Berkeley isn't enough to get it here
  • Needs to go to nearest airport, fly to SF
  • Go from SF to East Bay distribution center
  • And so on…
IP Routing

- Each router maintains a *routing table*
- Look up destination address in table
- Decide which outgoing link to use to send packet, gets it closer to destination
- Don't need 1 entry per IP address, instead routing is by subnet
- Routing table entry contains:
  - Destination address range + output link closer to dest.
- Routing table also has a default entry as a fallback
Setting up Routing Tables

• Internet has no centralized state!
  • No single machine knows entire topology
  • Topology is constantly changing (faults, reconfiguration)

• Instead, routers use a dynamic algorithm to populate their routing tables
  • Neighboring routers (with a direct link between them) exchange information
Setting up Routing Tables

• Possible algorithm to set up routing table:
  1. Each entry has a cost, includes number of hops to destination, congestion, etc.
  2. Neighbors periodically exchange routing tables
  3. If my neighbor knows a cheaper route to a subnet, forward traffic for that subnet to the neighbor
  4. Factor in +1 hop for sending to neighbor itself

• In reality: Internet has networks of different scales
  • Different algorithms run at different scales
IP Packet Format

IP Ver4

Time to Live (hops)

Type of transport protocol

IP Header Length

16-bit identification

16-bit source IP address

16-bit destination IP address

IP header 20 bytes

Flags & Fragmentation to split large messages

Size of packet (header+data)

16-bit header checksum

32-bit destination IP address

Type of transport protocol

IP Ver4

Time to Live (hops)

Type of transport protocol

IP Header Length

16-bit identification

16-bit source IP address

16-bit destination IP address

IP header 20 bytes

Flags & Fragmentation to split large messages

Size of packet (header+data)

16-bit header checksum

32-bit destination IP address

Options (if any)

Data
What does a router care about?

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| IP Version 4                  | IP Version 4 is a 4-bit field that identifies the version of the IP protocol.
| Time to Live (hops)           | Time to Live (TTL) is a 8-bit field that indicates the number of hops a packet can take.
| Type of transport protocol    | The protocol field is an 8-bit field that specifies the type of transport protocol used.
| 32-bit Source IP address      | The source IP address is a 32-bit field that contains the source IP address.
| 32-bit Destination IP address| The destination IP address is a 32-bit field that contains the destination IP address.
| 16-bit Identification         | The identification field is a 16-bit field that identifies the fragment.
| Flag                          | Flags are a 3-bit field that provides control information for the IP header.
| Total Length (16-bits)        | The total length field is a 16-bit field that indicates the total length of the packet.
| Size of packet (header+data)  | The size of packet is the size of the packet, including the header and data.
| IP Header Length              | The IP header length field is a 16-bit field that indicates the length of the header.
| Protocol                      | The protocol field is an 8-bit field that specifies the type of transport protocol used.
| Header Checksum               | The header checksum is a 16-bit field that provides a checksum for the IP header.
| Data                          | The data field contains the actual data being transmitted.
Layering

• Complex services from simpler ones
  1. Physical and Link Layers (Wi-Fi, Ethernet, …)
     • Unreliable, local exchange of limited-size frames
  2. Network (IP) – routing between local networks
     • Unreliable, global exchange of limited-size packets
  3. Transport (e.g., TCP) – Glue
     • Reliable (with retries), ordering, stream of bytes
  4. Application – Everything on top of sockets
Broadcast Networks

- Shared communication medium: A Bus
Broadcast Networks

• Shared Communication Medium

• Examples:
  • Original Ethernet
  • All types of wireless (WiFi, Cellular Networks, …)
  • Coaxial Cable (e.g., Cable Internet)
Broadcast Networks

- How do we get unicast (message to one host) from broadcast (message to all hosts)?
- Put Header on Front of Packet: Dest | Packet
  - Often handled directly in hardware
How do we specify a destination?

• Link Layer has its own addressing scheme: **Media Access Control (MAC) Addresses**

• MAC Address: 48 bits uniquely bound to your hardware device (hard coded by manufacturer)

• Remember: MAC Addresses only used for exchange of a frame across a single link/broadcast domain

• IP Address can change (e.g. you move from campus to home network)
  • MAC address is constant
Data Link Layer

- MAC Dest. Address
- MAC Src. Address
- ...

Network Layer

Datalink Layer

Frame Hdr.

Frame Payload

Physical Layer

101010100110101110

Network Layer

Datalink Layer

Frame Hdr.

Frame Payload

Physical Layer

101010100110101110
Media Access Control

- **Arbitration**: Who can use the broadcast medium when?

- Early example: Aloha Network (1970's), packet radio within Hawaii
  - Use checksum in frame header to detect error
  - If two senders try to send at same time, both packets get garbled, wait and re-send later

- Problems:
  - If network is too busy, no one gets through
  - Need senders to retry again at different times
Example: Original Ethernet

• All computers communicate on shared wire (bus)

• **CSMA/CD: Carrier Sense Multiple Access with Collision Detection**

• Carrier Sense: Don't send unless medium is idle
  
  • *Listen before Speaking*

• Collision Detection: Determine if sent packet was trampled by someone else on bus
  
  • If so, abort, wait and retry
Adaptive Randomized Backoff

• Wait a random amount of time before retransmitting again

• Why? Two machines involved in collision don't want to retransmit at same time
  • Just causes another collision!

• Increase wait time after each retry to adjust to how busy the shared medium is
Original Ethernet

- All frames delivered to all hosts
- If host is not intended recipient, drops frame
Link Layer Switches

- Inspects destination MAC address of incoming packet
- Forwards on relevant outgoing link
Switches vs Routers

• Routers operate at network layer, understand IP addresses
  • Build routing tables by exchanging information with neighbors, can also manually configure

• Switches operate at link layer, understand MAC addresses
  • "Self Learning" – build switching tables automatically by inspecting source MAC address of frames received on different links
  • Unknown MAC address: just broadcast
Point-to-Point Networks

• Switches make an Ethernet LAN operate more like a point-to-point network
• No shared medium: Physical wire connected to only two specific nodes
Putting it all together

Application Layer

Transport Layer

Network Layer

Datalink Layer

Physical Layer

Data

Data

Transport Hdr.

Net. Hdr.

Trans. Hdr.

Net. Hdr.

Trans. Hdr.

Net. Hdr.

Trans. Hdr.

Net. Hdr.

Trans. Hdr.

Net. Hdr.

Trans. Hdr.

Data

Data

Data

Data

Data

Data

Data

Data

101010100110101110

101010100110101110

101010100110101110

101010100110101110

101010100110101110

101010100110101110

101010100110101110

101010100110101110

101010100110101110

101010100110101110
Putting it all Together

- One technicality: a Switch works at the data link layer (can parse frame headers) but is *transparent* to communication endpoints
- Host does not specify Switch MAC address as destination of frame