COS 461: Computer Networks
Course Review
(12 weeks in 80 minutes)

Spring 2009 (MW 1:30-2:50 in CS 105)
Mike Freedman

Teaching Assistants: Wyatt Lloyd and Jeff Terrace
http://www.cs.princeton.edu/courses/archive/spr09/cos461/
What You (hopefully) Learned in This Course

- **Skill: network programming**
  - Socket programming
  - Implementing protocols

- **Knowledge: how Internet works**
  - IP protocol suite
  - Internet architecture
  - Applications (Web, DNS, P2P, ...)

- **Insight: key concepts**
  - Protocols
  - Resource allocation
  - Naming
  - Layering
Topics

• **Link layer:**
  – Ethernet and CSMA/CD
  – Wireless protocols and CSMA/CA
  – Spanning tree, switching and bridging
  – Translating addrs: DHCP and ARP

• **Network layer:**
  – IPv4, addressing, and forwarding
  – IP routing
    • Link-state and distance vector
    • BGP: path vector, policies
  – IP multicast and anycast
  – Middleboxes: NATs, firewalls
  – Tunneling: MPLS, IPSec
  – Addt. Considerations: mobility, DTNs

• **Transport layer:**
  – Socket interface
  – UDP
  – TCP
    • Reliability
    • Congestion Control
  – Reliable multicast

• **Application layer:**
  – Translating names: DNS
  – HTTP and CDNs
  – Overlay networks
  – Peer-to-peer and DHTs
  – Email
Link Layer
Link-Layer Services

• Encoding
  – Representing the 0s and 1s

• Framing
  – Encapsulating packet into frame, adding header and trailer
    – Using MAC addresses, rather than IP addresses

• Error detection
  – Errors caused by signal attenuation, noise.
    – Receiver detecting presence of errors
Multiple Access Protocol

• Single shared broadcast channel
  – Avoid having multiple nodes speaking at once
  – Otherwise, collisions lead to garbled data

• Multiple access protocol
  – Distributed algorithm for sharing the channel
  – Algorithm determines which node can transmit

• Classes of techniques
  – Channel partitioning: divide channel into pieces
    – Time-division multiplexing, frequency division multiplexing
  – Taking turns: passing a token for right to transmit
  – Random access: allow collisions, and then recover
Key Ideas of Random Access

• **Carrier Sense (CS)**
  – *Listen before speaking, and don’t interrupt*
  – Checking if someone else is already sending data
  – ... and waiting till the other node is done

• **Collision Detection (CD)**
  – *If someone else starts talking at the same time, stop*
  – Realizing when two nodes are transmitting at once
  – ...by detecting that the data on the wire is garbled

• **Randomness**
  – *Don’t start talking again right away*
  – Waiting for a random time before trying again
CSMA/CD Collision Detection
Wireless: Avoidance, Not Detection

• Collision detection in wired Ethernet
  – Station listens while transmitting
  – Detects collision with other transmission
  – Aborts transmission and tries sending again

• Problem #1: cannot detect all collisions
  – Hidden terminal problem
  – Fading

• Problem #2: listening while sending
  – Strength of received signal is much smaller
  – Expensive to build hardware that detects collisions

• So, 802.11 does not do collision detection
Medium Access Control in 802.11

• Collision avoidance, not detection
  – First exchange control frames before transmitting data
    • Sender issues “Request to Send” (RTS), including length of data
    • Receiver responds with “Clear to Send” (CTS)
  – If sender sees CTS, transmits data (of specified length)
  – If other node sees CTS, will idle for specified period
  – If other node sees RTS but not CTS, free to send

• Link-layer acknowledgment and retransmission
  – CRC to detect errors
  – Receiving station sends an acknowledgment
  – Sending station retransmits if no ACK is received
  – Giving up after a few failed transmissions
Scaling the Link Layer

• Ethernet traditionally limited by fading signal strength in long wires
  – Introduction of hubs/repeaters to rebroadcast

• Still a maximum “length” for a Ethernet segment
  – Otherwise, two nodes might be too far for carrier sense to detect concurrent broadcasts

• Further, too many nodes in shorter Ethernet can yield low transmissions rates
  – Constantly conflict with one another
Bridges/Switches: Traffic Isolation

- Switch breaks subnet into LAN segments
- Switch filters packets
  - Frame only forwarded to the necessary segments
  - Segments can support separate transmissions
## Comparing Hubs, Switches, Routers

<table>
<thead>
<tr>
<th>Feature</th>
<th>Hub/Repeater</th>
<th>Bridge/Switch</th>
<th>Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic isolation</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Plug and Play</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Efficient routing</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Cut through</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Self Learning: Building the Table

• When a frame arrives
  – Inspect the source MAC address
  – Associate the address with the incoming interface
  – Store the mapping in the switch table
  – Use a time-to-live field to eventually forget the mapping

Switch learns how to reach A
Solution: Spanning Trees

• Ensure the topology has no loops
  – Avoid using some of the links when flooding
  – ... to avoid forming a loop

• Spanning tree
  – Sub-graph that covers all vertices but contains no cycles
  – Links not in the spanning tree do not forward frames
Evolution Toward Virtual LANs

• In the olden days...
  – Thick cables snaked through cable ducts in buildings
  – Every computer they passed was plugged in
  – All people in adjacent offices were put on the same LAN
  – Independent of whether they belonged together or not

• More recently...
  – Hubs and switches changed all that
  – Every office connected to central wiring closets
  – Often multiple LANs (k hubs) connected by switches
  – Flexibility in mapping offices to different LANs

Group users based on organizational structure, rather than the physical layout of the building.
Example: Two Virtual LANs

Red VLAN and Orange VLAN
Switches forward traffic as needed
Network Layer
# IP Packet Structure

<table>
<thead>
<tr>
<th>4-bit Version</th>
<th>4-bit Header Length</th>
<th>8-bit Type of Service (TOS)</th>
<th>16-bit Total Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16-bit Identification</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3-bit Flags</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-bit Fragment Offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>8-bit Protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Source Address: What if Source Lies?

• Source address should be the sending host
  – But, who’s checking, anyway?
  – You could send packets with any source you want

• Why would someone want to do this?
  – Launch a denial-of-service attack
    • Send excessive packets to the destination
    • ... to overload the node, or the links leading to node
  – Evade detection by “spoofing”
    • But, the victim could identify you by the source address
    • So, you can put someone else’s source address in packets
  – Also, an attack against the spoofed host
    • Spoofed host is wrongly blamed
    • Spoofed host may receive return traffic from receiver
Hierarchical Addressing: IP Prefixes

- IP addresses can be divided into two portions
  - Network (left) and host (right)
- **12.34.158.0/24** is a 24-bit prefix
  - Which covers $2^8$ addresses (e.g., up to 255 hosts)
Classful Addressing

• In the olden days, only fixed allocation sizes
  – Class A: 0*
    • Very large /8 blocks (e.g., MIT has 18.0.0.0/8)
  – Class B: 10*
    • Large /16 blocks (e.g., Princeton has 128.112.0.0/16)
  – Class C: 110*
    • Small /24 blocks (e.g., AT&T Labs has 192.20.225.0/24)
  – Class D: 1110*
    • Multicast groups
  – Class E: 11110*
    • Reserved for future use

• This is why folks use dotted-quad notation!
CIDR: Hierarchal Address Allocation

- Prefixes are key to Internet scalability
  - Address allocated in contiguous chunks (prefixes)
  - Routing protocols and packet forwarding based on prefixes
  - Today, routing tables contain ~200,000 prefixes (vs. 4B)
Two types of addresses

- Provider independent (from IANA)
- Provider allocated (from upstream ISP)

- Provider allocated addresses seem to offer more potential for aggregation (and reducing routing table size), but not always so...
Scalability: Address Aggregation

Provider is given 201.10.0.0/21

- 201.10.0.0/22
- 201.10.4.0/24
- 201.10.5.0/24
- 201.10.6.0/23

Routers in rest of Internet just need to know how to reach 201.10.0.0/21. Provider can direct IP packets to appropriate customer.
But, Aggregation Not Always Possible

*Multi-homed* customer *(201.10.6.0/23)* has two providers. Other parts of the Internet need to know how to reach these destinations through *both* providers.
CIDR Makes Packet Forwarding Harder

• Forwarding table may have many matches
  – E.g., entries for 201.10.0.0/21 and 201.10.6.0/23
  – The IP address 201.10.6.17 would match *both*!
  – Use *Longest Prefix Matching*

• Can lead to routing table expansion
  – To satisfy LPM, need to announce /23 from both 1 and 2
Two types of addresses

- Provider independent (from IANA)
- Provider allocated (from upstream ISP)

Provider allocated addresses seem to offer more potential for aggregation (and reducing routing table size), but not always so…

- Multi-homing a PA address
- Traffic engineering between multiple links to same single provider
Internet-wide Internet Routing

- **AS-level topology**
  - Destinations are IP prefixes (e.g., 12.0.0.0/8)
  - Nodes are Autonomous Systems (ASes)
  - Edges are links and business relationships

![Diagram showing internet topology with nodes and edges connecting client to web server.](image-url)
Intradomain routing
(Interior Gateway Protocol – IGP)

Link-state:
- Keep complete map of all links
- Fast convergence
- Node can advertise incorrect link cost
- Each node computes only its own table
- OSPF, IS-IS, ...

Distance Vector:
- Keep only next-hop and cost information for each destination
- Convergence time varies (can be loops, count-to-infinity)
- DV node can advertise incorrect path cost
- Each node’s table used by others (error propagates)
- RIP, ...
Path-Vector Routing

• Extension of distance-vector routing
  – Support flexible routing policies
  – Avoid count-to-infinity problem

• Key idea: advertise the entire path
  – Distance vector: send distance metric per dest d
  – Path vector: send the entire path for each dest d
BGP Route

- Destination prefix (e.g., 128.112.0.0/16)
- Route attributes, including
  - AS path (e.g., “7018 88”)
  - Next-hop IP address (e.g., 12.127.0.121)
BGP Policy: Applying Policy to Routes

• **Import policy**
  – Filter unwanted routes from neighbor
    • E.g. prefix that your customer doesn’t own
  – Manipulate attributes to influence path selection
    • E.g., assign local preference to favored routes

• **Export policy**
  – Filter routes you don’t want to tell your neighbor
    • E.g., don’t tell a peer a route learned from other peer
  – Manipulate attributes to control what they see
    • E.g., make a path look artificially longer than it is
Customer-Provider Relationship

• Customer needs to be reachable from everyone
  – Provider tells all neighbors how to reach the customer
• Customer does not want to provide transit service
  – Customer does not let its providers route through it

Traffic to the customer

Traffic from the customer
Peer-Peer Relationship

- Peers exchange traffic between customers
  - AS exports *only* customer routes to a peer
  - AS exports a peer’s routes *only* to its customers
  - Often the relationship is settlement-free (i.e., no $$\$$)

Traffic to/from the peer and its customers
Identify the peer/transit links!
Extending the network layer

• Anycast
• Multicast
• Middleboxes
Motivation for IP anycast

• Failure problem: client has resolved IP address
  – What if IP address can represent many servers?

• Load-balancing/failover via IP addr, rather than DNS

• IP anycast is simple reuse of existing protocols
  – Multiple instances of a service share same IP address
  – Each instance announces IP address / prefix in BGP / IGP
  – Routing infrastructure directs packets to nearest instance of the service
    • Can use same selection criteria as installing routes in the FIB
  – No special capabilities in servers, clients, or network
Downsides of IP anycast

- Many Tier-1 ISPs ingress filter prefixes > /24
  - Publish a /24 to get a “single” anycasted address: Poor utilization
- Scales poorly with the # anycast groups
  - Each group needs entry in global routing table
- Not trivial to deploy
  - Obtain an IP prefix and AS number; speak BGP
- Subject to the limitations of IP routing
  - No notion of load or other application-layer metrics
  - Convergence time can be slow (as BGP or IGP convergence)
- Failover doesn’t really work with TCP
  - TCP is stateful; other server instances will just respond with RSTs
  - Anycast may react to network changes, even though server online
- Root name servers (UDP) are anycasted, little else
IP Multicast

• Simple to use in applications
  – Multicast “group” defined by IP multicast address
    • IP multicast addresses look similar to IP unicast addr
    • 224.0.0.0 to 239.255.255.255 (RPC 3171)
  – Best effort delivery only
    • Sender issues single datagram to IP multicast address
    • Routers delivery packets to all subnetworks that have a receiver “belonging” to the group

• Receiver-driven membership
  – Receivers join groups by informing upstream routers
  – Internet Group Management Protocol (v3: RFC 3376)
Middleboxes

• Middleboxes are intermediaries
  – Interposed in-between the communicating hosts
  – Often without knowledge of one or both parties

• Examples
  – Network address translators
  – Firewalls
  – Traffic shapers
  – Intrusion detection systems
  – Transparent Web proxy caches
  – Application accelerators
Two Views of Middleboxes

• An abomination
  – Violation of layering
  – Cause confusion in reasoning about the network
  – Responsible for many subtle bugs

• A practical necessity
  – Solving real and pressing problems
  – Needs that are not likely to go away

• Would they arise in any edge-empowered network, even if redesigned from scratch?
Port-Translating NAT

• Map outgoing packets
  – Replace source address with NAT address
  – Replace source port number with a new port number
  – Remote hosts respond using (NAT address, new port #)

• Maintain a translation table
  – Store map of (src addr, port #) to (NAT addr, new port #)

• Map incoming packets
  – Consult the translation table
  – Map the destination address and port number
  – Local host receives the incoming packet
Transport Layer
Two Basic Transport Features

- **Demultiplexing**: port numbers
- **Error detection**: checksums
User Datagram Protocol (UDP)

• Datagram messaging service
  – Demultiplexing of messages: port numbers
  – Detecting corrupted messages: checksum

• Lightweight communication between processes
  – Send messages to and receive them from a socket
  – Avoid overhead and delays of ordered, reliable delivery
Transmission Control Protocol (TCP)

• **Stream-of-bytes service**
  – Sends and receives a stream of bytes, not messages

• **Reliable, in-order delivery**
  – Checksums to detect corrupted data
  – Sequence numbers to detect losses and reorder data
  – Acknowledgments & retransmissions for reliable delivery

• **Connection oriented**
  – Explicit set-up and tear-down of TCP session

• **Flow control**
  – Prevent overflow of the receiver’s buffer space

• **Congestion control**
  – Adapt to network congestion for the greater good
Establishing a TCP Connection

- Three-way handshake to establish connection
  - Host A sends a **SYN**chronize (open) to the host B
  - Host B returns a SYN **ACKnowledgment** (**SYN ACK**)
  - Host A sends an **ACK** to acknowledge the SYN ACK

Each host tells its ISN to the other host.
TCP “Stream of Bytes” Service
...Emulated Using TCP “Segments”

Segment sent when:
1. Segment full (Max Segment Size),
2. Not full, but times out, or
3. “Pushed” by application.
Reliability: TCP Acknowledgments

Host A

ISN (initial sequence number)

Sequence number = 1st byte

TCP Data

TCP HDR

Host B

ACK sequence number = next expected byte

TCP Data

TCP HDR
Detecting losses

- Packet lost
- ACK lost
- Early timeout
  - DUPLICATE PACKET
  - DUPLICATE PACKETS
Flow control: Sliding window

- Allow a larger amount of data “in flight”
  - Allow sender to get ahead of the receiver
  - ... though not too far ahead
Where Congestion Happens: Links

• Simple resource allocation: FIFO queue & drop-tail

• Access to the bandwidth: first-in first-out queue
  – Packets transmitted in the order they arrive

• Access to the buffer space: drop-tail queuing
  – If the queue is full, drop the incoming packet
TCP Congestion Window

• Each TCP sender maintains a congestion window
  – Maximum number of bytes to have in transit
  – I.e., number of bytes still awaiting acknowledgments

• Adapting the congestion window
  – Decrease upon losing a packet: backing off
  – Increase upon success: optimistically exploring
  – Always struggling to find the right transfer rate

• Both good and bad
  – Pro: avoids having explicit feedback from network
  – Con: under-shooting and over-shooting the rate
Leads to the TCP “Sawtooth”

Window

Loss

halved

But, could take a long time to get started!
Slow Start and the TCP Sawtooth

Window

Loss

halved

Duplicate ACK

Exponential “slow start”
Repeating Slow Start After Timeout

Window

Loss

Timeout

halved

Slow start in operation until it reaches half of previous cwnd.
Extensions

• Tail drop in routers lead to bursty loss and synchronization of senders
  – Led to Random Early Detection (RED)

• Packets dropped and retransmission when unnecessary
  – Led to Explicit Congestion Notification (ECN)
Application layer

DNS
HTTP and CDNs
P2P and DHTs
Three Hierarchical Assignment Processes

• **Host name:** www.cs.princeton.edu
  – **Domain:** registrar for each top-level domain (e.g., .edu)
  – **Host name:** local administrator assigns to each host

• **IP addresses:** 128.112.7.156
  – **Prefixes:** ICANN, regional Internet registries, and ISPs
  – **Hosts:** static configuration, or dynamic using DHCP

• **MAC addresses:** 00-15-C5-49-04-A9
  – **Blocks:** assigned to vendors by the IEEE
  – **Adapters:** assigned by the vendor from its block
Mapping Between Identifiers

• **Domain Name System (DNS)**
  – Given a host name, provide the IP address
  – Given an IP address, provide the host name

• **Dynamic Host Configuration Protocol (DHCP)**
  – Given a MAC address, assign a unique IP address
  – … and tell host other stuff about the Local Area Network
  – To automate the boot-strapping process

• **Address Resolution Protocol (ARP)**
  – Given an IP address, provide the MAC address
  – To enable communication within the Local Area Network

DHCP and ARP use L2 broadcast....DNS is app-layer protocol
DNS: Distributed Hierarchical DB

- **com**, **edu**, **org**
  - **bar**
    - **west**, **east**
      - **foo**, **my**

- **ac**, **uk**, **zw**
  - **ac**, **cam**, **usr**
    - **usr.cam.ac.uk**

- **arpa**
  - **in-addr**
    - **12**, **34**, **56**

- **my.east.bar.edu**, **12.34.56.0/24**
Recursive vs. Iterative Queries

• **Recursive query**
  – Ask server to get answer for you
  – E.g., request 1 and response 8

• **Iterative query**
  – Ask server who to ask next
  – E.g., all other request-response pairs
DNS security

• DNS cache poisoning
  – Ask for www.evil.com
  – Additional section for (www.cnn.com, 1.2.3.4, A)
  – Thanks! I won’t bother check what I asked for

• DNS hijacking
  – Let’s remember the domain. And the UDP ID.
  – 16 bits: 65K possible IDs
    • What rate to enumerate all in 1 sec? ~32 Mbps
  – Prevention: Also randomize the DNS source port

• Weaknesses led to DNSSec
  – Chain of signatures from root to authoritative DNS server
HTTP Request Example

GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.intel-iris.net
Connection: Keep-Alive
One page, lots of objects

- Dynamic HTML: 19.6 KB
- Static content: 6.2 MB
  - 1 flash movie
  - 18 images
  - 5 style sheets
  - 3 scripts
TCP Interaction: Short Transfers

• Multiple connection setups
  – Three-way handshake each time

• Round-trip time estimation
  – Maybe large at the start of a connection (e.g., 3 seconds)
  – Leads to latency in detecting lost packets

• Congestion window
  – Small value at beginning of connection (e.g., 1 MSS)
  – May not reach a high value before transfer is done

• Detecting packet loss
  – Timeout: slow 😞
  – Duplicate ACK
    • Requires many packets in flight
    • Which doesn’t happen for very short transfers 😞
Persistent HTTP

Non-persistent HTTP issues:
• Requires 2 RTTs per object
• OS must allocate resources for each TCP connection
• But browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:
• Server leaves connection open after sending response
• Subsequent HTTP messages between same client/server are sent over connection

Persistent without pipelining:
• Client issues new request only when previous response has been received
• One RTT for each object

Persistent with pipelining:
• Default in HTTP/1.1
• Client sends requests as soon as it encounters referenced object
• As little as one RTT for all the referenced objects
Web Proxy Caches

• User configures browser: Web accesses via cache

• Browser sends all HTTP requests to cache
  – Object in cache: cache returns object
  – Else: cache requests object from origin, then returns to client
Content Distribution Networks (CDNs)

- Content providers are CDN customers

**Content replication**

- CDN company installs thousands of servers throughout Internet
  - In large datacenters
  - Or, close to users
- CDN replicates customers’ content
- When provider updates content, CDN updates servers
How to perform server selection?

• Routing based (IP anycast)
  – Pros: Transparent to clients, works when browsers cache failed addresses, circumvents many routing issues
  – Cons: Little control, complex, scalability, TCP can’t recover, ...

• Application based (HTTP redirects)
  – Pros: Application-level, fine-grained control
  – Cons: Additional load and RTTs, hard to cache

• Naming based (DNS selection)
  – Pros: Well-suited for caching, reduce RTTs
  – Cons: Request by resolver not client, request for domain not URL, hidden load factor of resolver’s population
    • Much of this data can be estimated “over time”
Consistent Hashing

• Construction
  – Assign each of C hash buckets to random points on mod \(2^n\) circle; hash key size = \(n\)
  – Map object to random position on circle
  – Hash of object = closest clockwise bucket

• Desired features
  – Balanced: No bucket responsible for large number of objects
  – Smoothness: Addition of bucket does not cause movement among existing buckets
  – Spread and load: Small set of buckets that lie near object

• Used layer in P2P Distributed Hash Tables (DHTs)
Extended consistent hashing to large-scale systems

- **Chord**: each node has small view of network
  - $k$ immediate successors
  - $\log n$ long-distance “fingers”

- **Performing lookup($k$)**
  - Greedily route to closest nodeid
  - Each step get $\frac{1}{2}$ closer
  - Takes $\log n$ hops
Topics

• Link layer:
  – Ethernet and CSMA/CD
  – Wireless protocols and CSMA/CA
  – Spanning tree, switching and bridging
  – Translating addr: DHCP and ARP

• Network layer:
  – IPv4, addressing, and forwarding
  – IP routing
    • Link-state and distance vector
    • BGP: path vector, policies
  – IP multicast and anycast
  – Middleboxes: NATs, firewalls
  – Tunneling: MPLS, IPSec
  – Addt. Considerations: mobility, DTNs

• Transport layer:
  – Socket interface
  – UDP
  – TCP
    • Reliability
    • Congestion Control
  – Reliable multicast

• Application layer:
  – Translating names: DNS
  – HTTP and CDNs
  – Overlay networks
  – Peer-to-peer and DHTs
  – Email