Recap: Course Goals

- Learn the main architectural concepts and technological components of communication networks, with the Internet as the overarching example
  - Understand how the Internet works
  - And why the Internet is the way it is

- Apply what you learned in three mini-class projects
  - Comprehensive P2P application
  - Simulation as an analysis tool: TCP variations
Communication Networks Taxonomy

- Communication networks classified on how nodes exchange information:

  Communication Network
  - Switched Communication Network
  - Broadcast Communication Network
  - Circuit-Switched Communication Network
  - Packet-Switched Communication Network
  - Datagram Network
  - Virtual Circuit Network

Internet is Based on Packet Switching

- Node in a packet switching network
Design Principles

- Layering
  - How to break network functionality into modules

- End-to-End Argument
  - Where to implement functionality

Layering

- Layering is a particular form of modularization

- System is broken into a vertical hierarchy of logically distinct entities (layers)

- Service provided by one layer is based solely on the service provided by layer below

- Rigid structure: easy reuse, performance suffers
End-to-End Argument: The Moderate Interpretation

- Think twice before implementing functionality in the network

- If hosts can implement functionality correctly, implement it at a lower layer only as a performance enhancement

- But do so only if it imposes no burden on applications that don’t require it

ISO OSI Reference Model for Layers

- Application
- Presentation
- Session
- Transport
- Network
- Datalink
- Physical
Mapping Layers onto Routers and Hosts

- Seven layers
  - Lower three layers are implemented everywhere
  - Next four layers are implemented only at hosts

Internet’s Hourglass

- Application Layer: e-mail, WWW, phone...
  - TCP, UDP...
- Network Layer: IP
  - Ethernet, PPP...
- Datalink Layer: CSMA, async, sonet...
- Physical Layer: copper, fiber, radio...
Physical Layer

- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send a bit
- **Protocol**: coding scheme used to represent a bit, voltage levels, duration of a bit
- **Examples**: coaxial cable, optical fiber links; transmitters, receivers

Datalink Layer

- **Service**:
  - Framing (attach frame separators)
  - Send data frames between peers
  - Medium access: arbitrate the access to common physical media
  - Error detection and correction
- **Interface**: send a data unit (packet) to a machine connected to the same physical media
- **Protocol**: layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)…
Medium Access Protocols

- **Channel partitioning**
  - Divide channel into smaller “pieces” (e.g., time slots, frequency, code selection)
  - Allocate a piece to given node for exclusive use

- **Random access**
  - Allow collisions
  - “Recover” from collisions

- **“Taking-turns”**
  - Tightly coordinate shared access to avoid collisions

CSMA: Carrier Sense Multiple Access

- **CS (Carrier Sense)** implies that each node can distinguish between an idle and a busy link

- **Sender operations:**
  - If channel sensed idle: transmit entire packet
  - If channel sensed busy, defer transmission
    - Various retry algorithms
**Reliable Transmission**

- Problem: obtain correct information once errors are detected
- Solutions:
  - Use error correction codes
  - Use retransmission
- Algorithmic challenges:
  - Achieve high link utilization, and low overhead

---

**Network Layer**

- **Service:**
  - Deliver a packet to specified network destination
  - Perform segmentation/reassemble
  - Others
    - Packet scheduling
    - Buffer management

- **Interface:** send a packet to a specified destination
- **Protocol:** define global unique addresses; construct routing tables
Datagram (Packet) Switching

Host A

Host B

router 1

router 2

router 3

router 5

router 6

router 7

Host C

Host D

Host E

Network

Application

Transport

Network

Link

Physical

Packet Forwarding

- At each router the packet destination address
  1. Is matched according to longest prefix matching rule
  2. Packet is forwarded to the corresponding output port

Forwarding table

<table>
<thead>
<tr>
<th>Destination Address</th>
<th>Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.15.xxx.xxx</td>
<td>3</td>
</tr>
<tr>
<td>128.15.11.xxx</td>
<td>2</td>
</tr>
<tr>
<td>16.82.100.xxx</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Internet

Router

data 10 7 16.25.31.10 128.15.11.12
Routing

- Intra-domain: use link state or distance vector protocols
- Inter-domain: use path vector protocol

Network: Intra-domain Routing Protocols

- Based on unreliable datagram delivery
- Distance vector
  - Routing Information Protocol (RIP), based on Bellman-Ford
  - Each neighbor periodically exchange reachability information to its neighbors
  - Minimal communication overhead, but it takes a long time to converge, i.e., in proportion to the maximum path length
- Link state
  - Open Shortest Path First (OSPF), based on Dijkstra
  - Each network periodically floods immediate reachability information to other routers
  - Fast convergence, but high communication and computation overhead
Multicast Approaches

- Kind of trees
  - Source specific trees
  - Shared trees

- Tree computation methods
  - Link state
  - Distance vector

Multicast

- Two types of multicast trees:
  - Source-specific
  - Shared

- Example:
  - 10 routers
  - 8 hosts (A-H)
  - Multicast group consisting of four hosts: A, C, D, F
For each source, we built a tree to distribute packets to all receivers.
- Ideally, the source-specific tree is the union of shortest paths from source to each receiver.

Example:
- Sender: A
- Receivers: B, D, F

Source Specific Tree
Shared Tree

- Build one tree that is shared by all receivers
- Not optimal, but require routers to maintain less state

Distance Vector Multicast Routing Protocol (DVRMP)

- An elegant extension to DV routing
- Use shortest path DV routes to determine if link is on the source-rooted spanning tree
- Three steps in developing DVRMP
  - Reverse Path Flooding
  - Reverse Path Broadcasting
  - Truncated Reverse Path Broadcasting
Reverse Path Flooding

- What: node X forwards packet from node Y to
  - all its neighbors (except Y),
  - iff Y is the next hop of X to Y

- How: just use unicast routing tables

- Why: eliminate the loops of simple flooding protocol
Reverse Path Broadcasting

- What: node X forwards packet to node Y iff X is next hop of Y to S

- How: X infer this info from routing messages (see multicast lecture)

- Why: Avoid a router receiving duplicate packets

Truncated Reverse Broadcasting

- What: don’t forward packets to non-members

- How: use prune messages

- Why: eliminate unnecessary forwarding
Truncated Reverse Broadcasting

• What: don’t forward packets to non-members

• How: use prune messages

• Why: eliminate unnecessary forwarding

Pruning Details

• Prune (Source,Group) at leaf if no members
  - Send Non-Membership Report (NMR) up tree

• If all children of router R send NRM, prune (S,G)
  - Propagate prune for (S,G) to parent R

• On timeout:
  - Prune dropped
  - Flow is reinstated
  - Downstream routers re-prune

• Note: a soft-state approach
Transport Layer

- **Service:**
  - Demultiplexing
  - Others:
    - Error-free delivery
    - Flow-control
    - Congestion-control

- **Interface:** send message to specific destination

- **Protocol:** implements reliability and flow control

- **Examples:** TCP and UDP

End-to-End View

- Process A sends a packet to process B
End-to-End Layering View

Transmission Control Protocol (TCP)

- Reliable, in-order, and at most once delivery
- Messages can be of arbitrary length
- Provides multiplexing/demultiplexing to IP
- Provides congestion control and avoidance
- Application examples: file transfer, chat
View from a Single Flow

- Knee – point after which
  - Throughput increases very slow
  - Delay increases fast

- Cliff – point after which
  - Throughput starts to decrease very fast to zero (congestion collapse)
  - Delay approaches infinity

TCP Congestion Control

- Measure available bandwidth
  - Slow start: fast, hard on network
  - AIMD: slow, gentle on network

- Use packet loss to detect congestion

Initially:
- cwnd = 1;
- ssthresh = infinite;

New ack received:
- if (cwnd < ssthresh)
  /* Slow Start*/
  cwnd = cwnd + 1;
- else
  /* Additive increase */
  cwnd = cwnd + 1/cwnd;

Timeout:
- /* Multiplicative decrease */
  ssthresh = cwnd/2;
  cwnd = 1;
The big picture

- Problem: too inefficient
  - Takes too long to detect a loss: retransmission timeouts > 500ms
  - Always start with a cwnd=1 after loss

Fast Retransmit and Fast Recovery

- Fast retransmit: retransmit after 3 duplicated ACKs (DupACKs)
- Fast recovery: start from cwnd/2 instead of 1 after loss detected by 3 DupACKs
  - Still set cwnd to 1 if loss is detected by timeout
- Fast recovery & retransmit: implemented in TCP Reno
TCP Flavors

- TCP-Tahoe
  - cwnd = 1 whenever drop is detected

- TCP-Reno
  - cwnd = 1 on timeout
  - cwnd = cwnd/2 on dupack

- TCP-newReno
  - TCP-Reno + improved fast recovery

- TCP-Vegas, TCP-SACK

Random Early Detection (RED)

- Basic premise:
  - Router should signal congestion when the queue first starts building up (by dropping a packet)
  - But router should give flows time to reduce their sending rates before dropping more packets

- Therefore, packet drops should be:
  - Early: don’t wait for queue to overflow
  - Random: don’t drop all packets in burst, but space drops out
Packet Scheduling

- Decide when and what packet to send on output link
- Make sure that the flow gets the allocated bandwidth
  - Usually implemented at output interface of a router

Packet Scheduling: Example

- Make sure that at any time the flow receives at least the allocated rate $r_a$
  - Canonical example of such scheduler: Weighted Fair Queueing (WFQ)

- Fair Queueing: each flow receives $\min(r, f)$, where
  - $r$ – flow arrival rate
  - $f$ – link fair rate (see next slide)

- Weighted Fair Queueing (WFQ) – associate a weight with each flow
Fair Rate Computation: Example

- If link congested, compute \( f \) such that

\[
\sum_i \min(r_i, f) = C
\]

\[
f = 4; \\
\min(8, 4) = 4 \\
\min(5, 4) = 4 \\
\min(2, 4) = 2
\]

Fluid Flow System

- Use bit-by-bit round robin:
  - In each round for each flow send a number of bits equal to its weight

<table>
<thead>
<tr>
<th></th>
<th>Packet Size (bits)</th>
<th>Packet inter-arrival time (ms)</th>
<th>Rate (C) (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>1000</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Flow 2</td>
<td>500</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

\[
\text{Area (C x transmission time) = packet size}
\]
Implementation In Packet System

- Packet (Real) system: packet transmission cannot be preempted.
- Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system

<table>
<thead>
<tr>
<th>Service in fluid flow system</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (ms)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Select the first packet that finishes in the fluid flow system

<table>
<thead>
<tr>
<th>Packet system</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>4</th>
<th>5</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Token Bucket and Arrival Curve

- Parameters
  - $r$ – average rate, i.e., rate at which tokens fill the bucket
  - $b$ – bucket depth
  - $R$ – maximum link capacity or peak rate (optional parameter)
- A bit is transmitted only when there is an available token
- Arrival curve – maximum number of bits transmitted within an interval of time of size $t$

- $r$ bps
- $b$ bits
- $\leq R$ bps
- regulator
- $b R/(R-r)$
- slope $R$
- slope $r$
Traffic Enforcement: Example

- \( r = 100 \text{ Kbps} \); \( b = 3 \text{ Kb} \); \( R = 500 \text{ Kbps} \)

(a) \( T = 0 \): 1Kb packet arrives

(b) \( T = 2\text{ms} \): packet transmitted
   \[ b = 3\text{Kb} - 1\text{Kb} + 2\text{ms} \cdot 100\text{Kbps} = 2.2\text{Kb} \]

(c) \( T = 4\text{ms} \): 3Kb packet arrives

(d) \( T = 10\text{ms} \): packet needs to wait until enough tokens are in the bucket!

(e) \( T = 16\text{ms} \): packet transmitted

Per-hop Reservation

- End-host: specify
  - the arrival rate characterized by token-bucket with parameters \((b, r, R)\)
  - the maximum maximum admissible delay \(D\)
- Router: allocate bandwidth \(r_a\) and buffer space \(B_a\) such that
  - no packet is dropped
  - no packet experiences a delay larger than \(D\)
Recap: Quality of Service

- Three kinds of QoS approaches
  - Link sharing, DiffServ, IntServ

- Some basic concepts:
  - Differentiated dropping versus service priority
  - Per-flow QoS (IntServ) versus per-aggregate QoS (DiffServ)
  - Admission control: parameter versus measurement
  - Control plane versus data plane
  - Controlled load versus guaranteed service
  - Codepoints versus explicit signaling

- Various mechanisms:
  - Playback points
  - Token bucket
  - RSVP PATH/RESV messages


Services: Basic DNS Features

- Hierarchical namespace
  - As opposed to original flat namespace

- Distributed storage architecture
  - As opposed to centralized storage (plus replication)

- Client--server interaction on UDP Port 53
  - But can use TCP if desired
Services: DNS Name Servers

- Local name servers:
  - Each ISP (company) has local default name server
  - Host DNS query first goes to local name server

- Authoritative name servers:
  - For a host: stores that host’s (name, IP address)
  - Can perform name/address translation for that host’s name

- Can also do IP to name translation, but won’t discuss

Security: Motivation

- Internet currently used for important services
  - Financial transactions, medical records
- Used in near future for even more critical services
  - 911 (VoIP), surgical operations, energy system control, transportation system control
- Networks more open than ever before
  - Global, ubiquitous Internet, wireless
- Malicious Users
  - Selfish users: want more network resources than you
  - Malicious users: would hurt you even if it doesn’t get them more network resources
Recap: Security

- Buffer overflow attack
- Worms
- Denial of service (DoS) attack

- Security requirements
- Cryptographic algorithms
  - How does DES and RSA work (no proof for RSA)
- Authentication algorithms
- Public key management, digital certificates (high level)

Recap: Services Provided by the Internet

- Shared access to computing resources
  - telnet (1970's)
- Shared access to data/files
  - FTP, NFS, AFS (1980's)
- Communication medium over which people interact
  - email (1980's), on-line chat rooms, instant messaging (1990's)
  - audio, video (1990's)
    - replacing telephone network?
- Medium for information dissemination
  - USENET (1980's)
  - WWW (1990's)
    - replacing newspaper, magazine?
  - Audio, video (late 90's)
    - replacing radio, CD, TV?
  - File sharing (late 90's)
- 21st Century: mobility + ubiquity + heterogeneity
Final Exam

- Open Peterson and Davie, Open Notes!
  - Crib sheets ok if you like
- Comprehensive, but greater focus on material since midterm
  - Link layer/media access, network services and applications (DNS, Web, CDNs, P2P systems), security, mobility + THE BIG PICTURE
- Questions similar in format to the first midterm
  - Problem set-up descriptions + multipart fill-ins
  - Will try to be more precise about what we are looking for
- All answers on the exam sheets we hand out
- Bring PENCIL, ERASER, no calculators needed

Potluck: Webteall-based p2p Sharing Network

- Murali runs his implementation of Webtella on four nodes
  - You can browse Potluck by pointing your browser to any one of the URLs below:
    http://cory.eecs.berkeley.edu:9000/
    http://cube.cs.berkeley.edu:9000/
    http://nova.cs.berkeley.edu:9000/
    http://rhombus.cs.berkeley.edu:9000/
- See newsgroup on how to participate!