Lecture 26: Final Review

CSE 123: Computer Networks
Alex C. Snoeren

Project 2 due tonight!
Exam Overview

- Focus on topics since the midterm
  - But everything is fair game…

- Roughly the same style and length of the midterm
  - It won’t take the whole time
  - We’ll spend the first few minutes of the exam period announcing the Espresso Prize winners

- As before, you can bring a crib sheet
  - One double-sided 8.5x11” paper with anything on it
Router Functional Architecture

Control Plane
- Complex
- Per-control action
- May be slow

Data plane
- Simple
- Per-packet
- Must be fast
Interconnect architecture

- Input & output connected via switch fabric

- Kinds of switch fabric
  - Shared Memory
  - Bus
  - Crossbar

- How to deal with transient contention?
  - Input queuing
  - Output queuing
Routing Challenges

- How to choose best path?
  - Defining “best” can be slippery

- How to scale to millions of users?
  - Minimize control messages and routing table size
  - Divide task into intra- and inter-AS

- How to adapt to failures or changes?
  - Node and link failures, plus message loss
Forwarding Options

- **Source routing**
  - Complete path listed in packet

- **Virtual circuits**
  - Set up path out-of-band and store path identifier in routers
  - Local path identifier in packet

- **Destination-based forwarding**
  - Router looks up address in forwarding table
  - Forwarding table contains (address, next-hop) tuples
Link-state Routing

- Routing is a distributed algorithm
  - React to changes in the topology
  - Compute the paths through the network

- Shortest-path link state routing
  - Flood link weights throughout the network
  - Compute shortest paths as a sum of link weights
  - Forward packets on next hop in the shortest path

- Motivation
  - Global information allows optimal route computation
  - Straightforward to implement and verify
Dijkstra’s Shortest Path

- Graph algorithm for single-source shortest path tree

\[
\begin{align*}
S & \leftarrow \{\} \\
Q & \leftarrow \text{<remaining nodes keyed by distance>} \\
\text{While } Q \neq \{\} & \\
& \quad \text{u } \leftarrow \text{extract-min}(Q) \\
& \quad S \leftarrow S \text{ plus } \{u\} \\
& \quad \text{for each node } v \text{ adjacent to } u \\
& \quad \quad \text{“relax” the cost of } v \\
\end{align*}
\]
Distance Vector Routing

Iterative, asynchronous: each local iteration caused by:
- Local link cost change
- Distance vector update message from neighbor

Distributed:
- Each node notifies neighbors only when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:
- wait for (change in local link cost or message from neighbor)
- recompute estimates
- if distance to any destination has changed, notify neighbors
Bellman-Ford Algorithm

- Define distances at each node $X$
  - $d_x(y) = \text{cost of least-cost path from } X \text{ to } Y$
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors $V$

\[
d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}
\]
Counting to Infinity Problem

Distance to C

Etc…
Mitigation Strategies

- **Hold downs**
  - As metric increases, delay propagating information
  - Limitation: Delays convergence

- **Loop avoidance**
  - Full path information in route advertisement
  - Explicit queries for loops (e.g. DUAL)

- **Split horizon**
  - Never advertise a destination through its next hop
    - A doesn’t advertise C to B
  - **Poison reverse:** Send negative information when advertising a destination through its next hop
    - A advertises C to B with a metric of ∞
    - Limitation: Only works for “loop”s of size 2
Autonomous Systems

- Internet is divided into **Autonomous Systems**
  - Distinct regions of administrative control
  - Routers/links managed by a single “institution”
  - Service provider, company, university, …

- Hierarchy of Autonomous Systems
  - Large, tier-1 provider with a nationwide backbone
  - Medium-sized regional provider with smaller backbone
  - Small network run by a single company or university

- Interaction between Autonomous Systems
  - Internal topology is not shared between ASes
  - … but, neighboring ASes interact to coordinate routing
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 destination
  - Path vector: send the *entire path* for each destination
A Simple 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)
Business Relationships

- Neighboring ASes have business contracts
  - How much traffic to carry
  - Which destinations to reach
  - How much money to pay

- Common business relationships
  - Customer-provider
    » E.g., Princeton is a customer of USLEC
    » E.g., MIT is a customer of Level3
  - Peer-peer
    » E.g., UUNET is a peer of Sprint
    » E.g., Harvard is a peer of Harvard Business School
Each node can apply local policies
- Path selection: Which path to use?
- Path export: Which paths to advertise?

Examples
- Node 2 may prefer the path “2, 3, 1” over “2, 1”
- Node 1 may not let node 3 hear the path “1, 2”
AS Relationships

- Flatter and much more densely interconnected Internet
- Disintermediation between content and "eyeball" networks
- New commercial models between content, consumer and transit

**Settlement Free**
- Pay for BW
- Pay for access BW
Multicast: Different Options

Source-based tree
- Efficient trees; low delay, even load
- Per-source state in routers (S,G)

Shared-tree
- Higher delay, skewed load
- Per-group state only (G)
Naming Processes/Services

- Process here is an abstract term for your Web browser (HTTP), Email servers (SMTP), hostname translation (DNS)

- How do we identify for remote communication?
  - Process id or memory address are OS-specific and transient

- So TCP and UDP use Ports
  - 16-bit integers representing mailboxes that processes “rent”
  - Identify process uniquely as (IP address, protocol, port)
User Datagram Protocol (UDP)

- Provides *unreliable message delivery* between processes
  - Source port filled in by OS as message is sent
  - Destination port identifies UDP delivery queue at endpoint
- Connectionless (no state about who talks to whom)

<table>
<thead>
<tr>
<th></th>
<th>SrcPort</th>
<th>DstPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Checksum</td>
<td>Length</td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

0 16 31
TCP Header Format

- Ports plus IP addresses identify a connection (4-tuple)

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>0-10</td>
</tr>
<tr>
<td>DstPort</td>
<td>10-20</td>
</tr>
<tr>
<td>SequenceNum</td>
<td>20-24</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>24-28</td>
</tr>
<tr>
<td>HdrLen</td>
<td>28-30</td>
</tr>
<tr>
<td>Flags</td>
<td>30-31</td>
</tr>
<tr>
<td>AdvertisedWindow</td>
<td>31</td>
</tr>
<tr>
<td>Checksum</td>
<td>0</td>
</tr>
<tr>
<td>UrgPtr</td>
<td>4</td>
</tr>
<tr>
<td>Options (variable)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
TCP Delivery

Application process

Write bytes

TCP
Send buffer

Transmit segments
Segment Segment ... Segment

TCP
Receive buffer

Read bytes

Application process
Three-Way Handshake

- Opens both directions for transfer

Active participant (client)

SYN, SequenceNum = x

SYN + ACK, SequenceNum = y,
Acknowledgment = x + 1

ACK, Acknowledgment = y + 1

Passive participant (server)

+data

CSE 123 – Lecture 26: Final Review
Congestion Control

- How fast should a sending host transmit data?
  - Not to fast, not to slow, just right…

- Should not be faster than the sender’s share
  - Bandwidth allocation

- Should not be faster than the network can process
  - Congestion control

- Congestion control & bandwidth allocation are separate ideas, but frequently combined
Rough definition: “When an increase in network load produces a decrease in useful work”

Why does it happen?
- Sender sends faster than bottleneck link speed
- Packets queue until dropped
- In response to packets being dropped, sender retransmits
- All hosts repeat in steady state…
Proactive vs. Reactive

- **Congestion avoidance**: try to stay to the left of the knee
- **Congestion control**: try to stay to the left of the cliff
AIMD

- Increase slowly while we believe there is bandwidth
  - Additive increase per RTT
  - $cwnd += 1$ full packet / RTT

- Decrease quickly when there is loss (went too far!)
  - Multipliclicative decrease
  - $cwnd /= 2$
TCP’s Various Phases

Slow Start + Congestion Avoidance + Fast Retransmit + Fast Recovery

Round-trip times vs cwnd

Fast recovery
NA(p)T Example

1: host 10.0.0.4 sends packet to 132.239.8.45:80

2: NAT router changes packet source addr from 10.0.0.1:3345 to 138.76.29.7:5001, updates table

NAT translation table

<table>
<thead>
<tr>
<th>WAN side addr</th>
<th>LAN side addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.29.7:5001</td>
<td>10.0.0.4:3345</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

3: Reply arrives dest. address: 138.76.29.7:5001

4: NAT router changes packet dest addr from 138.76.29.7:5001 to 10.0.0.4:3345
Managing Overload

- Buffer Management
  - FIFO
  - RED

- Traffic Policing and Scheduling
  - Token Buckets
  - WFQ
RED Operation

Max thresh

Min thresh

Average Queue Length

P(drop)

1.0

max_p

min_th

max_th

Avg queue length

CSE 123 – Lecture 26: Final Review
Token Bucket Basics

- **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

\[
\text{Maximum # of bits sent} = \frac{b \cdot R}{R - r}
\]

- Graph showing the relationship between time, bits, and the rates $r$ and $R$. The graph illustrates how the maximum number of bits sent is determined by the parameters.
Maintain a queue for each flow
  - What is a flow?

Implements **max-min fairness**: each flow receives \( \min(r_i, f) \), where
  - \( r_i \) – flow arrival rate
  - \( f \) – link fair rate (see next slide)

**Weighted Fair Queuing** (WFQ) – associate a weight with each flow
Packet-Based WFQ

- Select the first packet that finishes in the fluid flow system.
Network-wide QoS

- **Integrated services**
  - Motivated by need for end-to-end guarantees
  - On-line negotiation of per-flow requirements
  - End-to-end per-router negotiation of resources
  - Complex

- **Differentiated services**
  - Motivated by economics (multi-tier pricing)
  - No per-flow state
  - Not end-to-end and not guaranteed services
  - Simple
802.11 Summary

- Common technology for local-area wireless
- Uses CSMA/CA
- Needs to handle hidden terminal problem
- Challenges due to asymmetric ranges
CSMA/CA

- Cannot detect collision w/half-duplex radios

- Wireless MAC protocols often use collision avoidance techniques, in conjunction with a (physical or virtual) carrier sense mechanism

- Collision avoidance
  - Nodes negotiate to reserve the channel.
  - Once channel becomes idle, the node waits for a randomly chosen duration before attempting to transmit.
When A wants to send a packet to B, A first sends a Request-to-Send (RTS) to B.

On receiving RTS, B responds by sending Clear-to-Send (CTS), provided that A is able to receive the packet.

When C overhears a CTS, it keeps quiet for the duration of the transfer.

- Transfer duration is included in both RTS and CTS.
Parting thoughts…

● Good luck finishing up Project 2
   ◆ Don’t use the late days…

● Please complete your CAPE survey online

● Final exam: MONDAY 8-11 AM

● Good luck and have a great summer!