Administrivia/Last Time

- Project due Tuesday
- New Homework due 3/10: 2.39, 2.45, 3.15, 3.17, 3.22, 3.31, 4.7, 4.11,
- Recall that midterm is 3/17
- Last time
  - Level 2 switching: learning bridges and ATM
  - Spanning tree algorithm to get rid of loops
  - Network Caches
  - Issues and limitations of level 2 switching
- Today: we will continue with IP

Switching

To realize the advantages of switching from a performance perspective, we have to have a way of figuring out where the different hosts are – routing
- Static or dynamic “routing”?
- Switching at level 2 can bridge some, but not all the heterogeneity between technologies
- Traditionally, only simple frame conversion is done
- No address translation
- No fragmentation
- Simple routing, if any

Spanning Tree Algorithm (review)

- IEEE 802.1 is the standard for “LAN/MAN Bridging and Management”
- Idea: bridges disable ports to eliminate cycles
- General Algorithm:
  - Bridges elect a “tree root” bridge
  - Each bridge calculates shortest path to root
  - Bridges on each LAN elect a “designated bridge” such that
    - It is the closest bridge to the root
    - Break ties using bridge id

Spanning Tree Algorithm (review)

- How does that actually help?
  - Lets look at simple loop examples
    - Works, but not efficient (let alone optimal)
      - No hierarchy, relies on broadcast sometimes, centralized traffic bottleneck, vulnerable...
    - Ok solution for small networks (heavily used in LANs)

SUNY-Binghamton – CS428 Spring ’05 Lec. #9
Internetworking

- Switches have the potential of scaling to large networks
- Bridges (link-level switching) provided valuable lessons
  - Not scalable (consider spanning tree, common broadcasts, or addressing scheme)
  - Not heterogeneous (same MAC address family; compatible payloads, etc.)
- Scalability: must scale indefinitely
  - Addressing and Routing
  - Multicast and broadcast?
- Heterogeneity:
  - Users on different networks must be able to talk
  - Might need to cross several other networks on the way

Internet Protocol (IP; due to Karn and Cerf)

- Runs on all hosts (remember the “hour-glass” shape of the Internet Protocol Suite)
- Provides isolation from the networking technology
- Must provide one service model that is common to all possible underlying technologies
  - Connectionless best-effort delivery

Internet Protocol (IP)

IP Packet Format

- Version: 4 for IPv4
- Hlen: number of 32-bit words in the header
- TOS: Type of Service; can be used for QoS
- Length: Number of bytes in the datagram
- Id/Flags/Offset: used for fragmentation and reassembly
- TTL: Time to Live (maximum hop count allowed; why?)
- Protocol: key to identify higher level protocol (e.g., TCP)
- Checksum: applied to header (why not CRC?)
- Source and destination addresses (what addresses?)
- Options
- Payload
- Where does the “link-layer” header fit?
Fragmentation and Reassembly

- Each network has some Maximum Transmission Unit (e.g., ethernet 1500 bytes; PPP 532 bytes)
  - Restrict IP payload to the smallest MTU; or
  - Use fragmentation and reassembly if necessary

- Strategy:
  - Fragment when necessary (MTU < Datagram)
  - Fragments are self-contained IP packets
  - Try to avoid fragmentation at the source
  - Refragmentation is possible
  - Delay reassembly until destination
  - What if a fragment is lost?

<table>
<thead>
<tr>
<th>Each network has some Maximum Transmission Unit (e.g., ethernet 1500 bytes; PPP 532 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Restrict IP payload to the smallest MTU; or</td>
</tr>
<tr>
<td>- Use fragmentation and reassembly if necessary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategy:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Fragment when necessary (MTU &lt; Datagram)</td>
</tr>
<tr>
<td>- Fragments are self-contained IP packets</td>
</tr>
<tr>
<td>- Try to avoid fragmentation at the source</td>
</tr>
<tr>
<td>- Refragmentation is possible</td>
</tr>
<tr>
<td>- Delay reassembly until destination</td>
</tr>
<tr>
<td>- What if a fragment is lost?</td>
</tr>
</tbody>
</table>

Fragmentation and Reassembly – How?

<table>
<thead>
<tr>
<th>Ident is the packet sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-bit (flags field) is 1 in all but the last fragment</td>
</tr>
<tr>
<td>How would refragmentation be implemented?</td>
</tr>
<tr>
<td>“path MTU discovery” to minimize fragmentation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ident is the packet sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-bit (flags field) is 1 in all but the last fragment</td>
</tr>
<tr>
<td>How would refragmentation be implemented?</td>
</tr>
<tr>
<td>“path MTU discovery” to minimize fragmentation</td>
</tr>
</tbody>
</table>

Internet Addresses

<table>
<thead>
<tr>
<th>Class D [1110— - - - multicast group id - - -]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address broken into a network number and a host number</td>
</tr>
<tr>
<td>Why different address classes?</td>
</tr>
<tr>
<td>- Class A: 126 networks, each (2^{24}) hosts</td>
</tr>
<tr>
<td>- Class B: (2^{14}) networks, each (2^{16}) hosts</td>
</tr>
<tr>
<td>- Class C: (2^{21}) networks, each (2^{8}) hosts</td>
</tr>
<tr>
<td>What is an address associated with?</td>
</tr>
<tr>
<td>Is this scalable? How many maximum hosts?</td>
</tr>
</tbody>
</table>
Address Translation

- Need to map the IP address to a physical MAC address for:
  - Destination host (for a direct connection)
  - Next hop router
- Possible approaches:
  - Encode the Physical address as part of the IP address?
  - Statically construct translation tables (e.g., administrator does it)
  - Dynamically construct translations (ARP)

Forwarding Packets

- A local routing table is consulted to figure out the next hop
- Types of routes in a routing table
  - Direct vs. indirect routes
  - Host-specific vs. network specific vs. default routes

Address Resolution Protocol (ARP)

- Maintain a table of IP to physical address mappings; another network cache
- If translation not in table, broadcast asking for it
- Target hears broadcast, and replies with physical address
- Table entries expire; TTL usually a few minutes
- What happens on switched LANs? Proxy ARP

Direct Routes

- Machines on the same network, they don't need an intermediate router
- How do we know the physical address of the NIC on the other side (e.g., ethernet network)?
  - Need address translation
**Back to the Routing Table**

- Routes in the table map an IP address to the address of the next hop.
- Three different types of routes can exist in the routing table:
  - Host-specific: an entry giving the route to a specific host.
  - Network specific: an entry giving the route for all hosts in a network.
  - Default: an entry for routes to hosts we know nothing about.

**ARP Example**

```
[root@garnet]# /sbin/arp
Address   HWtype   HWaddress  Iface
128.226.123.1  ether  00:50:0B:F7:9C:00  eth0

[root@garnet]# /sbin/arp -Ds cs.binghamton.edu eth0
[root@garnet]# /sbin/arp
Address   HWtype   HWaddress  Iface
cs.binghamton.edu  ether  08:00:20:CD:7D:6A  eth0
128.226.123.1  ether  00:50:0B:F7:9C:00  eth0
```
**Example Forwarding Table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.226.123.0</td>
<td>*</td>
<td>255.255.255.0</td>
<td>eth0</td>
</tr>
<tr>
<td>127.0.0.0</td>
<td>*</td>
<td>255.0.0.0</td>
<td>lo</td>
</tr>
<tr>
<td>default</td>
<td>128.226.123.1</td>
<td>0.0.0.0</td>
<td>eth0</td>
</tr>
</tbody>
</table>

- Go through entries in order
- Logically AND destination IP with Mask. If output matches destination, use the entry.

**Configuration: Who assigns IP addresses to hosts?**

- Two approaches:
  - Static: parameters are hard wired
    - Conceptually easy, but difficult to administer
    - Necessary for servers to have a fixed point of attachment to the Internet
  - Dynamic:
    - Parameters obtained from a server
    - Easy to administer
    - Dynamic use of available addresses
    - How to do it?

**Route Examples**

- Masks are actually used instead of *

---

**Example Forwarding Table**

```
[root@garnet] netstat -r
Kernel IP routing table
Destination  Gateway   Genmask    Iface
128.226.123.0 * 255.255.255.0 eth0
127.0.0.0     * 255.0.0.0    lo
default       128.226.123.1 0.0.0.0 eth0
```

- Go through entries in order
- Logically AND destination IP with Mask. If output matches destination, use the entry.

**Configuration: Who assigns IP addresses to hosts?**

- Two approaches:
  - Static: parameters are hard wired
    - Conceptually easy, but difficult to administer
    - Necessary for servers to have a fixed point of attachment to the Internet
  - Dynamic:
    - Parameters obtained from a server
    - Easy to administer
    - Dynamic use of available addresses
    - How to do it?
Host Configuration Protocols

- Reverse ARP (RARP – RFC 903):
  - When machine is booted, it announces its hardware address and asks if anyone knows its IP address
  - RARP server replies and assigns the machine an IP address

- Boot Protocol (BOOTP 951):
  - Newer protocol, layered on top of UDP
  - New capability is to supply also the name of a machine that will serve the boot image
  - This allows diskless workstations to boot via the network

- Dynamic Host Configuration Protocol (DHCP – RFC 2131 and 2132):
  - Successor to BOOTP
  - Provides all the information necessary to configure the host
  - IPv4 address
  - Subnet Mask (to be explained soon)
  - DNS server (to be explained later)
  - Routing information
  - Servers manage a finite number of IP addresses
  - Addresses are leased to clients for finite leases
  - Renew lease if you need IP address longer

DHCP

When a host “wakes up”, it sends a DHCP-DISCOVER message (ip address 255.255.255.255)
- Message contains hosts MAC address
- Either the DHCP server or a DHCP relay agent receives the message
- If DHCP relay agent, it forwards the message to the DHCP server
- Server replies with an IP number and configuration information

Layered on top of UDP
What happens if a host turns itself off without releasing the address?

Internet Control Message Protocol (ICMP)

- A protocol for signalling/feedback among routers and hosts
- Not intended for use by applications; used to send information about problems
  - No multiplexing is provided (i.e., to a specific process). It is intended purely as a host-to-host mechanism
- Built on top of IP, but considered “at the same layer”
- To prevent message explosion, ICMP messages are not generated in response to errors experienced by other ICMP messages
  - Several types of information can be exchanged (bad packets, congestion, failed routes, etc.)
ICMP

- ICMP message types include:
  - Echo request/reply (with or without a timestamp)
  - Address mask request/reply
  - Parameter problem
  - Source quench (to control congestion)
  - Redirect (router knows a better route)
  - Destination unreachable (port, protocol, host)
  - TTL exceeded
  - Checksum failed ...

- What is the idea? I thought IP was unreliable??
- Several applications use ICMP (ping, traceroute...)

Traceroute

- Implemented by sending UDP datagrams to the destination
- First packet has TTL=1
  - Next hop replies that destination is unreachable (ICMP message); we learn what the next hop is
  - Increase TTL by 1, and repeat

Traceroute Example

[root@garnet] /usr/sbin/traceroute syracuse.edu
traceroute to syracuse.edu (128.230.18.35), 30 hops max, 38 byte pac
1 128.226.123.1 (128.226.123.1) 1.394 ms 0.772 ms 0.748 ms
2 128.226.100.1 (128.226.100.1) 6.363 ms 6.919 ms 5.027 ms
3 * 128.226.100.30 (128.226.100.30) 3.908 ms 4.659 ms
4 149.125.1.1 (149.125.1.1) 1.488 ms 2.788 ms 3.857 ms
5 199.109.4.38 (199.109.4.38) 14.012 ms 12.792 ms 11.748 ms
6 syru-vbns1.nysernet.net (199.109.4.13) 26.954 ms 21.260 ms 20
7 128.230.249.2 (128.230.249.2) 22.655 ms 20.399 ms 20.445 ms
8 128.230.93.1 (128.230.93.1) 20.780 ms 20.514 ms 22.336 ms
9 cwis01.syr.edu (128.230.18.35) 20.704 ms * 24.706 ms