Recap

• Spanning tree algorithm
• IPv4

• Today:
  – Subnetting
  – ARP / ICMP / DNS
  – Routing
Scaling Problems

• Not enough network numbers.
  – Class C network with 2 nodes wastes 253 IP addresses
  – Class B network with ~300 nodes wastes 64,000 IP addresses
  – Only $2^{14} \sim 16,500$ class B networks

• Routing information too cumbersome.
  – More networks means larger routing tables
Subnetting

• Idea: One IP network number allocated to several physical networks.
  – The multiple physical networks are called *subnets*
  – Should be close together (why?)
  – Useful when a large company (or university!) has many physical networks.
IP addresses

- Hierarchical, not flat as in Ethernet

A

```
0  | 7       | 24
---+---------+---
0   | Network | Host
```

B

```
1  | 14      | 16
---+---------+---
1  0 | Network | Host
```

C

```
1  1  0 | 21      | 8
---+---------+---
1  1  0 | Network | Host
```

- Written as four decimal numbers separated by dots: 158.130.14.2
Subnet Numbers

• **Solution: Subnetting**
  – All nodes are configured with *subnet mask*
  – Allows definition of a *subnet number*
    • All hosts on a physical subnetwork share the same *subnet number*

<table>
<thead>
<tr>
<th>Subnet Mask (255.255.255.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111111111111111111111111</td>
</tr>
</tbody>
</table>

Subnetted Address:

<table>
<thead>
<tr>
<th>Network number</th>
<th>Subnet ID</th>
<th>Host ID</th>
</tr>
</thead>
</table>
Example of Subnetting

Subnet mask: 255.255.255.128
Subnet #: 128.96.34.0

128.96.34.1

Subnet mask: 255.255.255.128
Subnet #: 128.96.34.128

128.96.33.1

Subnet mask: 255.255.255.0
Subnet #: 128.96.33.0
Subnets, continued

• Mask is bitwise-ANDed with address
• This is done at routers
• Router tables in this model:
  – <Subnet #, Subnet Mask, NextHop>
• Subnetting allows a set of physical networks to look like a single logical network from elsewhere
Forwarding Algorithm

D = destination IP address
for each forwarding table entry
(SubnetNumber, SubnetMask, NextHop)
    D1 = SubnetMask & D
    if D1 = SubnetNumber
        if NextHop is an interface
            deliver datagram directly to destination
        else
            deliver datagram to NextHop (router)
ARP - Address Resolution Protocol

• Problem:
  – Need mapping between IP and link layer addresses.

• Solution: ARP
  – Every host maintains IP–Link layer mapping table (cache)
  – Timeout associated with cached info (15 min.)

• Sender
  – Broadcasts “Who is IP addr X?”
  – Broadcast message includes sender’s IP & Link Layer address

• Receivers
  – Any host with sender in cache “refreshes” time-out
  – Host with IP address X replies “IP X is Link Layer Y”
  – Target host adds sender (if not already in cache)
ICMP: Internet Control Message Protocol

- Collection of error & control messages
- Sent back to the source when Router or Host cannot process packet correctly
- Error Examples:
  - Destination host unreachable
  - Reassembly process failed
  - TTL reached 0
  - IP Header Checksum failed
- Control Example:
  - Redirect – tells source about a better route
Domain Name System

- System for mapping mnemonic names for computers into IP addresses.
  
  zeta.cis.upenn.edu  →  158.130.12.244

- Domain Hierarchy
- Name Servers
- Name Resolution
Domain Name Hierarchy

edu  com  gov  mil  org  net

cornell ... upenn  cisco ... yahoo  nasa ... nsf  arpa ... navy ...

cis  seas  wharton ...
Hierarchy of Name Servers

- Root Name Server
  - Cornell Name Server
  - Upenn Name Server
    - CIS Name Server
    - SEAS Name Server
    - Wharton Name Server
Records on Name Servers

• < Name, Value, Type, Class >
• Types
  – A Host to address mappings
  – NS Name server address mappings
  – CNAME Aliases
  – MX Mail server mappings
• Class IN for IP addresses
Name resolution

- Client
- Local Name server: 198.168.0.100
- Root Name server
- Upenn Name server
- CIS Name server

Connections:
- Client to Local Name server: zeta.cis.upenn.edu
- Local Name server to Root Name server: 128.196.128.233
- Root Name server to zeta.cis.upenn.edu
- zeta.cis.upenn.edu to 198.168.0.1
- 198.168.0.1 to Local Name server
- Local Name server to Upenn Name server
- Upenn Name server to zeta.cis.upenn.edu
- zeta.cis.upenn.edu to 198.168.0.100
- 198.168.0.100 to CIS Name server
IP Routing

- Begin by partitioning problem:
  - Intradomain Routing
    - Inside *administrative domains* (AD’s)
  - Interdomain Routing
    - Between administrative domains (e.g., companies)
    - Exterior Gateway Protocol (EGP)
    - Border Gateway Protocol (BGP) [Replaced EGP]
Intradomain Routing

• RIP - Routing Information Protocol
  – Uses distance vector algorithm
  – Limited to small nets; <15 hops

• OSPF - Open Shortest Path First
  – Augmented version of link-state
  – Augmentation includes authentication, load-balancing, and defined areas
Distance Vector Algorithm (RIP)

• Similar to the Spanning Tree Algorithm
  – Except that information about distance to ALL nodes is forwarded (not just info. about root.)
  – Sometimes called Bellman-Ford algorithm

• Each node constructs a Distance Vector
  – Contains distances (costs) to reach all other nodes
  – Initially:
    • Distance to neighbors = 1
    • Distance to others = ∞
  – Routing table reflects node’s beliefs
Example Network Graph

A’s initial information

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>∞</td>
<td>-</td>
</tr>
</tbody>
</table>
Iteration Steps

• Each host sends its DV to its neighbors
• Neighbors can update their distance vectors and routing information accordingly.
  – As in spanning tree, the nodes ignore worse information
  – Update any better routes
• If host changed its tables, send new DV to neighbors

• After a few iterations, routing information converges
Example Iteration Steps

F sends A its DV.
- A discovers that G can be reached in two hops through F

C sends A its DV.
- A discovers that D can be reached in two hops through C

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>F</td>
</tr>
</tbody>
</table>
Details

• Note: No single host has all routing information.

• When to send update vectors?
  – When your routing table changes (triggered)
  – Periodically (“I’m alive!”)

• Detecting link/node failure
  – (1) Periodically exchange “I’m alive!” messages.
  – (2) Timeout mechanism