Announcements

• Reminders:
  – Project I is due on Monday, Sept. 25th.
  – Homework 1 is due on Friday, Sept. 29th.
Internet Protocol Interoperability

Overlays (running at hosts)

Virtual Network Infrastructure (runs globally)

Networks (run locally)
IPv4 Packet Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Header Length</td>
<td>4</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8</td>
</tr>
<tr>
<td>Length</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>16</td>
</tr>
<tr>
<td>Flags</td>
<td>4</td>
</tr>
<tr>
<td>Offset</td>
<td>16</td>
</tr>
<tr>
<td>Time to Live</td>
<td>8</td>
</tr>
<tr>
<td>Protocol</td>
<td>8</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
</tr>
<tr>
<td>Source Address</td>
<td>32</td>
</tr>
<tr>
<td>Destination</td>
<td>32</td>
</tr>
<tr>
<td>Options</td>
<td>Variable length</td>
</tr>
<tr>
<td>Pad</td>
<td>64</td>
</tr>
<tr>
<td>DATA</td>
<td>256</td>
</tr>
</tbody>
</table>

CSE331 Fall 2004
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 $2^{14} \approx 16,500$ 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

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

- **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>11111111111111111111111111</td>
</tr>
</tbody>
</table>

**Subnetted Address:**

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

Subnet mask: 255.255.255.128
Subnet #: 128.96.34.0

- H1
  - 128.96.34.15

- R1
  - 128.96.34.1
  - 128.96.34.130

- Subnet mask: 255.255.255.128
- Subnet #: 128.96.34.128

- 128.96.34.129

- H2
  - 128.96.34.139

- Subnet mask: 255.255.255.0
- Subnet #: 128.96.33.0

- H3
  - 128.96.33.14

- 128.96.33.1

- Subnet mask: 255.255.255.128
- Subnet #: 128.96.33.0
Subnets, continued

• Mask is bitwise-ANDed with address
• This is done at routers
• Router tables in this model:
  – \(<\text{Subnet #, Subnet Mask, NextHop}>\)
• 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)
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
Spanning Trees (Abstractly)

- Given a connected graph G
- A *spanning tree* is an acyclic, connected subgraph of G that contains all the nodes.
Spanning Tree Algorithm (Abstractly)

- Pick a root node
- Compute shortest paths to root
- Need to break ties
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

- 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>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>

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
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