Administrivia

• Last time
  – IP Basics, Fragmentation, ARP, routing tables
  – Subnetting and supernetting, CIDR
  – Configuration/DHCP

• Today
  – More odds and ends; ICMP, NAT
  – Routing
IP Basics

- Single Layer 3 protocol to run on top of everything
- Makes all networking hardware look the same to all networking software
- Provides no additional services (end-to-end principle)
- One place to solve all Internetworking problems
  - Fragmentation/reassembly
  - Heterogeneity
  - Aggregatable address space
  - Scalable routing
Addresses and Forwarding

- **INET address** `<network>:` `<host>`
  - Mask: `<11 ... 1>:` `<0 ... 0>`, with as many 1’s as there are bits in the network field
  - Alternative representation: `<network>/length`, where length is the number of bits in the network field

- **Forwarding:**
  - Look at the “network” part only
    - Get it by AND’ing destination address with Mask
  - Get Most specific matching routing table entry (Longest Prefix Match)
Subnetting, Supernetting and CIDR

• Original address schemes, network size is fixed depending on address class (A, B or C)

• Subnetting moves bits from the host field to the network field (making more networks, each with fewer hosts)

• Supernetting moves bits from the host field to the network field (combining networks into a bigger one)

• Generalizing subnetting and supernetting gives us CIDR
Another Subnetting Example

Network number | Host number
--- | ---

Class B address

111111111111111111111111 | 00000000
Subnet mask (255.255.255.0)

Network number | Subnet ID | Host ID
--- | --- | ---

Subnetted address

Subnet mask: 255.255.255.128
Subnet number: 128.96.34.0

128.96.34.15
H1

128.96.34.1
R1

128.96.34.130

Subnet mask: 255.255.255.128
Subnet number: 128.96.34.128

128.96.34.129

128.96.33.14
H3

128.96.33.1
R2

128.96.34.139

Subnet mask: 255.255.255.0
Subnet number: 128.96.33.0
Configuration: Who assigns IP addresses to hosts?

- 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
  - RARP, BOOTP, DHCP
Network Address Translation (NAT)

- One machine has a legitimate IP address and is connected to the Internet
- Internally, other machines are assigned private addresses
- NAT machine establishes “proxy” connections on behalf of the other machines
- Only NAT machine is visible to the outside
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...)

SUNY-BINGHAMTON – CS528 FALL '07 LEC. #11
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 packets
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
Routing

- Network can be seen as a graph (static or dynamic?)
  - The costs on the edges correspond to some metric (latency, 1/bandwidth, congestion, etc.)

- Routing: find the cheapest path between any two nodes – how?
  - Distributed or centralized? Static or Dynamic?

- Two general Approaches – Distance Vector and Link State
Distance Vector/Bellman Ford

- Each node maintains a table of the best route it knows to every destination
  - Each entry contains (Destination, distance, next hop)

- Each node exchanges its table with neighbors
  - Periodically
  - Whenever something changes (triggered)

- Update route if it is better than what you have currently
  - If it is shorter
  - If it came from the next hop for the current route

- Delete routes if they timeout (why?)
Distance Vector Example

E  ??
F  ??
G  ??
H  ??

A

E 10 wa
F 30 wa
G 50 wa
H 10 wa

D

E 25 wb
F 20 wb
G 10 wb
H 15 wb

B

C

E 8  wc
F 50 wc
G 15 wc
H 40 wc
Distance Vector Example (2)

```
E 10 C
F 30 B
G 17 C
H 15 A

E 10 wa
F 30 wa
G 50 wa
H 10 wa

E 25 wb
F 20 wb
G 10 wb
H 15 wb

E 8 wc
F 50 wc
G 15 wc
H 40 wc
```
Distance Vector Example (3)

```
E 10 C
F 30 B
G 17 C
H 15 A
```

```
E 10 wa
F 30 wa
G 22 D
H 10 wa
```

```
E 20 D
F 20 wb
G 10 wb
H 15 wb
```

```
E 8  wc
F 32 D
G 15 wc
H 17 D
```
• Track the operation of distance vector in this network; all edges cost 1
Distance Vector Discussion

- Does the algorithm always converge?
- What happens if the topology changes (link breaks or is added)?
  - How are these situations detected?
- Does the algorithm adapt and find a table?
  - Sometimes!!
- Problem: updates take a long time to propagate
- Problem: stale updates that contain wrong information may continue to propagate
- RIP (Routing Information Protocol) is based on distance vector
  - In use since the 70s
  - Still in use despite problems
  - unix routed
  - RFC 1058 and 1723
• What happens if the link between B and D is broken?
Count to Infinity (1)

SUNY-BINGHAMTON – CS528 FALL ’07 LEC. #11
Count to Infinity (2)
Count to Infinity

- When is the problem resolved? What happens when the costs reach 10?
- What happens if C crashes?
- Is there anyway to avoid this problem? Not really; some heuristics
- Idea: Split horizon – don’t tell the route to the node you learned it from
  - Does it work?
- Idea: pick infinity to be a small number (e.g., 16 hops)
Split Horizon not Perfect

- Idea: Path vectors – include the full path (or a portion of it), detect loops
RIP Problems

- Slow to converge – one hop every update
- Infinity too small
  - Any problems making it larger?
- Routing metric is number of hops
- Utilizes only a single path even when multiple are available
Link State Routing

• Idea: Send updates to all nodes in the network with information about only your immediate neighbors

  – The update consists of a Link State Packet (LSP) which holds
    ∗ Source node id
    ∗ Cost of the link to each directly connected node
    ∗ A sequence number (SEQNO)
    ∗ A time to live (TTL)
  – How to send the packet to all nodes in the network?

• Each node has the full topology of the network, and can carry out shortest path computation locally to find optimal paths
Reliable Flooding

- Each node forwards the packet first time it sees to all outgoing links
- Store most recent LSP from each node
- Generate a new LSP periodically; increment its SEQNO
- Decrement TTL of each stored LSP; delete if it reaches 0
Route Calculation

\[ M = \{s\}; \]
\[ \text{for each } n \text{ in } (N - \{s\}) \]
\[ C(n) = l(s, n); \]
\[ \text{while } (N \neq M) \]
\[ M = M + \{w\} \text{ such that } C(w) \text{ is minimum of } c(x) \text{ for all } x \text{ in } (N-M); \]
\[ \text{for each } n \text{ in } (N-M) \]
\[ C(n) = \text{Min}(C(n), C(w) + l(w, n)); \]

- Classic Dijkstra’s shortest path algorithm
- \(N\) denotes the set of nodes in the graph
- \(l(i,j)\) denotes non-negative cost for edge \((i,j)\)
- \(s\) is the node carrying out the computation
- \(M\) is the set of “visited” nodes
- \(C(n)\) is the cost of the path from \(s\) to \(n\)
Discussion

• How does link state perform compared to distance vector?

• Neither approach scales gracefully
  – Distance vector – communication with neighbors; update packet includes information on all nodes
  – Link state – packet includes information only on neighbors; communicate with all nodes

• Must reduce size (or propagation extent) and size of routing tables
Example: Open Shortest Path First (OSPF)

- Link state protocol
- OSPF layered on top of IP
- Implemented in UNIX as gated (which also supports RIP)
- In use since 1990 (predecessors available earlier)
- RFC 2328
- Has several interesting features beyond “vanilla” link state
OSPF

- Neighbors are discovered using hello messages
- Measure/request cost to/from each neighbor
- Transmit routing information to all other routers (reliable flooding)
- Shortest Path based on Dijkstra’s algorithm, with a modification to obtain all shortest paths to a destination (not just one)
  - Packets are routed on all shortest paths to avoid congestion
- Incorporates hierarchy in the form of areas
• Note R11, it is connected to two areas, but not to backbone
Optimizations

- Router and Network LSA flooded within area only
- Area-border (A,B) routers produce summary LSAs for their own areas and flood the backbone as well as other areas
  - Interior routers do not receive summary LSA floods or AS-external LSA floods
  - Instead, they use their area-border router as default router
- Area border router advertises a single route for the entire area
  - cost is the maximum for the area
- In general, how to determine link cost?