Routing

Outline
Algorithms
Scalability

Overview

• Forwarding vs Routing
  – forwarding: to select an output port based on
destination address and routing table
  – routing: process by which routing table is built
• Network as a Graph

• Problem: Find lowest cost path between two nodes
• Factors
  – static: topology
  – dynamic: load

Distance Vector

• Each node maintains a set of triples
  – (Destination, Cost, NextHop)
• Directly connected neighbors exchange updates
  – periodically (on the order of several seconds)
  – whenever table changes (called triggered update)
• Each update is a list of pairs:
  – (Destination, Cost)
• Update local table if receive a “better” route
  – smaller cost
  – came from next-hop
• Refresh existing routes; delete if they time out
Routing Loops

• Example 1
  – F detects that link to G has failed
  – F sets distance to G to infinity and sends update to A
  – A sets distance to G to infinity since it uses F to reach G
  – A receives periodic update from C with 2-hop path to G
  – A sets distance to G to 3 and sends update to F
  – F decides it can reach G in 4 hops via A
• Example 2
  – link from A to E fails
  – A advertises distance of infinity to E
  – B and C advertise a distance of 2 to E
  – B decides it can reach E in 3 hops; advertises this to A
  – A decides it can read E in 4 hops; advertises this to C
  – C decides that it can reach E in 5 hops...

Loop-Breaking Heuristics

• Set infinity to 16
• Split horizon
• Split horizon with poison reverse
Link State

- **Strategy**
  - send to all nodes (not just neighbors)
  - information about directly connected links (not entire routing table)

- **Link State Packet (LSP)**
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet

Link State (cont)

- **Reliable flooding**
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically
    - increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP
    - discard when TTL=0

Route Calculation

- **Dijkstra’s shortest path algorithm**
- Let
  - $N$ denotes set of nodes in the graph
  - $l(i, j)$ denotes non-negative cost (weight) for edge $(i, j)$
  - $s$ denotes this node
  - $M$ denotes the set of nodes incorporated so far
  - $C(n)$ denotes cost of the path from $s$ to node $n$

\[
M = \{ s \} \\
\text{for each } n \text{ in } N - \{ s \} \\
C(n) = l(s, n) \\
\text{while } (M \neq N) \\
\quad M = M \cup \{ w \} \text{ such that } C(w) \text{ is the minimum for all } w \text{ in } (N - M) \\
\quad \text{for each } n \text{ in } (N - M) \\
\quad C(n) = \min\{C(n), C(w) + l(w, n)\}
\]
Metrics

- Original ARPANET metric
  - measures number of packets queued on each link
  - took neither latency or bandwidth into consideration

- New ARPANET metric
  - stamp each incoming packet with its arrival time (AT)
  - record departure time (DT)
  - when link-level ACK arrives, compute
    \[ \text{Delay} = (DT - AT) + \text{Transmit + Latency} \]
  - if timeout, reset DT to departure time for retransmission
  - link cost = average delay over some time period

- Fine Tuning
  - compressed dynamic range
  - replaced Delay with link utilization

How to Make Routing Scale

- Flat versus Hierarchical Addresses
- Inefficient use of Hierarchical Address Space
  - class C with 2 hosts (2/255 = 0.78% efficient)
  - class B with 256 hosts (256/65535 = 0.39% efficient)

- Still Too Many Networks
  - routing tables do not scale
  - route propagation protocols do not scale

Internet Structure

Recent Past
Internet Structure

Today

Subnetting

- Add another level to address/routing hierarchy: subnet
- Subnet masks define variable partition of host part
- Subnets visible only within site

<table>
<thead>
<tr>
<th>Class B address</th>
<th>Network number</th>
<th>Host number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111111111111111</td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>Subnet mask (255.255.255.0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network number</th>
<th>Subnet ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subnet Example

<table>
<thead>
<tr>
<th>Subnet mask: 255.255.255.128</th>
<th>Subnet number: 128.96.34.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.96.34.128</td>
<td>128.96.34.129</td>
</tr>
<tr>
<td>Subnet mask: 255.255.255.128</td>
<td>Subnet number: 128.96.34.0</td>
</tr>
<tr>
<td>128.96.34.128</td>
<td>128.96.34.129</td>
</tr>
<tr>
<td>Subnet mask: 255.255.255.128</td>
<td>Subnet number: 128.96.34.0</td>
</tr>
<tr>
<td>128.96.34.128</td>
<td>128.96.34.129</td>
</tr>
<tr>
<td>Next Hop: Interface 0</td>
<td>Interface 1</td>
</tr>
<tr>
<td>Forwarding table at router R1</td>
<td></td>
</tr>
</tbody>
</table>
Forwarding Algorithm

D = destination IP address
for each entry (SubnetNum, SubnetMask, NextHop)
   D1 = SubnetMask & D
   if D1 = SubnetNum
      if NextHop is an interface
         deliver datagram directly to D
      else
         deliver datagram to NextHop
   else
      • Use a default router if nothing matches
      • Not necessary for all 1s in subnet mask to be contiguous
      • Can put multiple subnets on one physical network
      • Subnets not visible from the rest of the Internet

Super-netting

• Assign block of contiguous network numbers to nearby networks
• Called CIDR: Classless Inter-Domain Routing
• Represent blocks with a single pair
  \( (\text{first-network-address}, \text{count}) \)
• Restrict block sizes to powers of 2
• Use a bit mask (CIDR mask) to identify block size
• All routers must understand CIDR addressing

Route Propagation

• Know a smarter router
  – hosts know local router
  – local routers know site routers
  – site routers know core router
  – core routers know everything
• Autonomous System (AS)
  – corresponds to an administrative domain
  – examples: University, company, backbone network
  – assign each AS a 16-bit number
• Two-level route propagation hierarchy
  – interior gateway protocol (each AS selects its own)
  – exterior gateway protocol (Internet-wide standard)
Popular Interior Gateway Protocols

• RIP: Route Information Protocol
  – developed for XNS
  – distributed with Unix
  – distance-vector algorithm
  – based on hop-count
• OSPF: Open Shortest Path First
  – recent Internet standard
  – uses link-state algorithm
  – supports load balancing
  – supports authentication

EGP: Exterior Gateway Protocol

• Overview
  – designed for tree-structured Internet
  – concerned with reachability, not optimal routes
• Protocol messages
  – neighbor acquisition: one router requests that another
    be its peer; peers exchange reachability information
  – neighbor reachability: one router periodically tests if
    the other is still reachable; exchange HELLO/ACK
    messages; uses a k-out-of-n rule
  – routing updates: peers periodically exchange their
    routing tables (distance-vector)

BGP-4: Border Gateway Protocol

• AS Types
  – stub AS: has a single connection to one other AS
    • carries local traffic only
  – multihomed AS: has connections to more than one AS
    • refuses to carry transit traffic
  – transit AS: has connections to more than one AS
    • carries both transit and local traffic
• Each AS has:
  – one or more border routers
  – one BGP speaker that advertises:
    • local networks
    • other reachable networks (transit AS only)
    • gives path information
**BGP Example**

- Speaker for AS2 advertises reachability to P and Q
  - network 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached directly from AS2

- Speaker for backbone advertises
  - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2).

- Speaker can cancel previously advertised paths

---

**IP Version 6**

- Features
  - 128-bit addresses (classless)
  - multicast
  - real-time service
  - authentication and security
  - autoconfiguration
  - end-to-end fragmentation
  - protocol extensions

- Header
  - 40-byte “base” header
  - extension headers (fixed order, mostly fixed length)
    - fragmentation
    - source routing
    - authentication and security
    - other options