Link State Routing

Observation: loop can be prevented if each node knows the actual network topology

In link-state routing, each node:
• floods the network with the state (up, down) of its links
• uses Dijkstra’s Shortest Path First (SPF) algorithm to compute a shortest-path tree

What is advertised:
• DV: all nodes reachable from me, advertised to all neighbors
• LS: all my immediate neighbors, advertised to all nodes

Dijkstra’s Shortest Path First (SPF) Algorithm

A greedy algorithm for solving single-source shortest path problem
• assume non-negative edge weights
• even if we’re only interested in the path from s to a single destination, d, we need to find the shortest path from s to all vertices in G (otherwise, we might have missed a shorter path)
• if the shortest path from s to d passes through an intermediate node u, i.e., \( P = \{s, \ldots, u, \ldots, d\} \), then \( P' = \{s, \ldots, u\} \) must be the shortest path from s to u
Dijkstra’s Shortest Path First (SPF) Algorithm

SPF(startnode s)
{ // Initialize
  table = createtable(|V|); // stores spf, cost, predecessor
  table[*].spf = false; table[*].cost = INFINITY;
  pq = createpq(|E|); // empty pq
  table[s].cost = 0;
  pq.insert(0, s); // pq.insert(cost, v)
  while (!pq.isempty()) {
    v = pq.getMin();
    if (!table[v].spf) { // not on sp tree
      table[v].spf = true;
      for each u = v.neighbors() {
        newcost = weight(u, v) + table[v].cost;
        if (table[u].cost > newcost) {
          table[u].cost = newcost;
          table[u].pred = v;
          pq.insert(newcost, u);
        }
      }
    }
  }
  extract SPF from table;
}

Dijkstra’s SPF Example (init)

<table>
<thead>
<tr>
<th>u</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>F</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>F</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>F</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>F</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>F</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>F</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>

Diagram: Graph with nodes a, b, c, d, e, f and edges showing distances and paths.
### Dijkstra’s SPF Example ($v = s = b$)

<table>
<thead>
<tr>
<th>$u$</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>F</td>
<td>3</td>
<td>$b$</td>
</tr>
<tr>
<td>$b$</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$c$</td>
<td>F</td>
<td>5</td>
<td>$b$</td>
</tr>
<tr>
<td>$d$</td>
<td>F</td>
<td>$\infty$</td>
<td></td>
</tr>
<tr>
<td>$e$</td>
<td>F</td>
<td>$\infty$</td>
<td></td>
</tr>
<tr>
<td>$f$</td>
<td>F</td>
<td>$\infty$</td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image)

### Dijkstra’s SPF Example ($v = a$)

<table>
<thead>
<tr>
<th>$u$</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>T</td>
<td>3</td>
<td>$b$</td>
</tr>
<tr>
<td>$b$</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$c$</td>
<td>F</td>
<td>4</td>
<td>$a$</td>
</tr>
<tr>
<td>$d$</td>
<td>F</td>
<td>8</td>
<td>$a$</td>
</tr>
<tr>
<td>$e$</td>
<td>F</td>
<td>$\infty$</td>
<td></td>
</tr>
<tr>
<td>$f$</td>
<td>F</td>
<td>$\infty$</td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image)
Dijkstra’s SPF Example ($v = c$)

<table>
<thead>
<tr>
<th>$u$</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>T</td>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>$b$</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$c$</td>
<td>T</td>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>$d$</td>
<td>F</td>
<td>6</td>
<td>c</td>
</tr>
<tr>
<td>$e$</td>
<td>F</td>
<td>8</td>
<td>c</td>
</tr>
<tr>
<td>$f$</td>
<td>F</td>
<td>$\infty$</td>
<td></td>
</tr>
</tbody>
</table>

Dijkstra’s SPF Example ($v = d$)

<table>
<thead>
<tr>
<th>$u$</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>T</td>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>$b$</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$c$</td>
<td>T</td>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>$d$</td>
<td>T</td>
<td>6</td>
<td>c</td>
</tr>
<tr>
<td>$e$</td>
<td>F</td>
<td>8</td>
<td>c</td>
</tr>
<tr>
<td>$f$</td>
<td>F</td>
<td>11</td>
<td>d</td>
</tr>
</tbody>
</table>
Dijkstra’s SPF Example ($v = e$)

<table>
<thead>
<tr>
<th>$u$</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>T</td>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>$b$</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$c$</td>
<td>T</td>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>$d$</td>
<td>T</td>
<td>6</td>
<td>c</td>
</tr>
<tr>
<td>$e$</td>
<td>T</td>
<td>8</td>
<td>c</td>
</tr>
<tr>
<td>$f$</td>
<td>F</td>
<td>9</td>
<td>e</td>
</tr>
</tbody>
</table>

Dijkstra’s SPF Example ($v = f$)

<table>
<thead>
<tr>
<th>$u$</th>
<th>spf</th>
<th>cost</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>T</td>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>$b$</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$c$</td>
<td>T</td>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>$d$</td>
<td>T</td>
<td>6</td>
<td>c</td>
</tr>
<tr>
<td>$e$</td>
<td>T</td>
<td>8</td>
<td>c</td>
</tr>
<tr>
<td>$f$</td>
<td>T</td>
<td>9</td>
<td>e</td>
</tr>
</tbody>
</table>
Dijkstra’s SPF Algorithm

Algorithm complexity: $N$ nodes
- each iteration: extract minHeap $O(\log(N))$
- total $O(|N|\log(N))$

Each neighbor of each node could also potentially go thru the minHeap once: $O(|E|\log(|N|))$

Total: $O(|N|\log(|N|)+|E|\log(|N|)) = O(|E| \log |N|)$
- $(|E| \geq |N| - 1$ for a connected graph)

Oscillations possible:
e.g., link cost = amount of carried traffic, asymmetric link cost

OSPF (Open Shortest Path First)

“Open”: publicly available
Uses Link State algorithm
- LS packet dissemination
  - advertisements disseminated to entire network
    (via flooding protocol: forward to all interfaces except the incoming one)
  - advertisement carried in OSPF messages directly over IP (rather than TCP or UDP)
- route computation using Dijkstra’s algorithm
- topology map at each node
  - OSPF is not loop free due to delay in topology propagation
  - maintaining LS database consistency is hard due to router reboot:
    - how to determine which LS is newer?
OSPF (Open Shortest Path First)

Advance features (not in RIP):

- security: all OSPF messages authenticated (to prevent fake advertisement)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set to “low” for best effort; high for real time)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
  - Hierarchical OSPF in large domains
Hierarchical OSPF

Two-level hierarchy: local area, backbone.
• Link-state advertisements only in area
• each node has detailed area topology; only know direction (shortest path) to nets in other areas.

Area border routers: “summarize” distances to nets in own area, advertise to other Area Border routers.
Backbone routers: run OSPF routing limited to backbone.
Boundary routers: connect to other AS’s.

Internet inter-AS routing: BGP

BGP (Border Gateway Protocol): the de facto standard
BGP provides each AS a means to:
1. Obtain subnet reachability information from neighboring ASes.
2. Propagate the reachability information to all routers internal to the AS.
3. Determine “good” routes to subnets based on reachability information and policy.

Allows a subnet to advertise its existence to rest of the Internet: “I am here”
Router Architecture Overview

Two key router functions:
run routing algorithms/protocol (RIP, OSPF, BGP)
forwarding datagrams from incoming to outgoing link

Input Port Functions

Decentralized switching:
given datagram dest., lookup output port using forwarding table in input port memory
goal: complete input port processing at ‘line speed’
queuing: if datagrams arrive faster than forwarding rate into switch fabric
Three types of switching fabrics

**Switching Via Memory**

*First generation routers:* traditional computers with switching under direct control of CPU packet copied to system’s memory speed limited by memory bandwidth (2 bus crossings per datagram)
Switching Via a Bus

- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Switching Via An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network
Output Ports

Buffering required when datagrams arrive from fabric faster than the transmission rate
Scheduling discipline chooses among queued datagrams for transmission

Output port queueing

buffering when arrival rate via switch exceeds output line speed
queueing (delay) and loss due to output port buffer overflow!
How much buffering?

RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity $C$
- e.g., $C = 10$ Gps link: 2.5 Gbit buffer
Recent recommendation: with $N$ flows, buffering equal to $\frac{\text{RTT} \cdot C}{\sqrt{N}}$

Input Port Queuing

Fabric slower than input ports combined -> queueing may occur at input queues

Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
queueing delay and loss due to input buffer overflow!
DHCP: Dynamic Host Configuration Protocol

**Goal:** allow host to *dynamically* obtain its IP address from network server when it joins network
Can renew its lease on address in use
Allows reuse of addresses (only hold address while connected an “on”)  
Support for mobile users who want to join network (more shortly)

**DHCP overview:**
- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

DHCP client-server scenario
DHCP client-server scenario

DHCP can return more than just allocated IP address on subnet:
- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address)
DHCP: example

Connecting laptop needs its IP address, addr of first-hop router, addr of DNS server:
- Use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux’ed to IP demux’ed, UDP demux’ed to DHCP

DCP server formulates DHCP ACK containing client’s IP address, IP address of first-hop router for client, name & IP address of DNS server
- Encapsulation of DHCP server, frame forwarded to client, demux’ing up to DHCP at client
- Client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router
DHCP: wireshark output (home LAN)

Message type: Boot Request (1)
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0xb3a11b7
Seconds elapsed: 0
Boostrap flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP Request
Option: (61) Client Identifier
Length: 7; Value: 010016D323688A;
Hardware type: Ethernet
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=15,l=20) Domain Name = "nomad"
Option: (55) Parameter Request List
Length: 11; Value: 010F03062C2E2F1F21F92B
1 = Subnet Mask; 15 = Domain Name
3 = Router; 6 = Domain Name Server
44 = NetBIOS over TCP/IP Name Server
......

IP addresses: how to get one?

Q: How does network get subnet part of IP addr?
A: gets allocated portion of its provider ISP's address space

ISP's block   11001000  00010111  00010000  00000000  200.23.16.0/20
Organization 0 11001000  00010111  00010000  00000000  200.23.16.0/23
Organization 1 11001000  00010111  00010010  00000000  200.23.18.0/23
Organization 2 11001000  00010111  00010100  00000000  200.23.20.0/23
......
Organization 7 11001000  00010111  00011110  00000000  200.23.30.0/23
NAT: Network Address Translation

Motivation: local network uses just one IP address as far as outside world is concerned:
- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus).
NAT: Network Address Translation

Implementation: NAT router must:

- **outgoing datagrams: replace** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  
  ... remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

- **remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair

- **incoming datagrams: replace** (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table
NAT: Network Address Translation

16-bit port-number field:
• 60,000 simultaneous connections with a single LAN-side address!

NAT is controversial:
• routers should only process up to layer 3
• violates end-to-end argument
  • NAT possibility must be taken into account by app designers, eg, P2P applications
• address shortage should instead be solved by IPv6

NAT traversal problem

client wants to connect to server with address 10.0.0.1
• server address 10.0.0.1 local to LAN (client can’t use it as destination addr)
• only one externally visible NATted address: 138.76.29.7

solution 1: statically configure NAT to forward incoming connection requests at given port to server
• e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
NAT traversal problem

solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
- learn public IP address (138.76.29.7)
- add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration

NAT traversal problem

solution 3: relaying (used in Skype)
- NATed client establishes connection to relay
- External client connects to relay
- relay bridges packets between to connections