Routing in data networks

Martin Heusse

Routing in data networks

- Three sub-tasks
  - ✓ Switching
  - ✓ Path establishment
  - ✓ Topology discovery
- The IP case
  - ✓ No path establishment/call setup
- The ATM case
  - ✓ Cell switching
  - ✓ Virtual circuit setup
  - ✓ Routing: “hand made” or PNNI

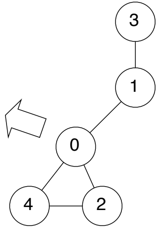
Routing in LANs

- Tree topology
- Diffusion based routing, started from destination
  - ✓ Only one path to the destination
  - ✓ The path to a node starts on the interface from which its packets emerge!

Layer 3 routing

- Routing table

destination	*interface towards*
1	1
2	2
4	4
3	1



Example (BSD $\rightarrow$ IP + ARP) :

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	129.88.38.254	UGSc	21	1119	en0	
127.0.0.1	127.0.0.1	UH	9	9326	lo0	
129.88.38/24	link#2	UC	0	0	en0	
129.88.38.1	0:3:ba:0:d5:f	UHLW	4	7589	en0	1183
129.88.38.153	127.0.0.1	UHS	0	2	lo0	

Layer 3 routing (cont.)

- Hop by hop routing
- Topology discovery
  - ✓ Hand made
  - ✓ Automatic (dynamic): routing daemons

IP: one has to avoid to form routing loops or “black holes”

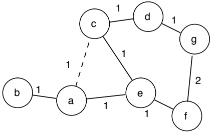
- The packets TTL is a only a worst case, partial solution
- (Almost) single field that routers modify along the packet’s trip through the net (if no fragmentation)

Dynamic routing

- Shortest path routing
  - ✓ Based on a static link metric(s)
  - → Loop free paths
  - → Independent of the load
- Benefits
  - ✓ Resilience to link breaks
  - → The network heals
- Attention :
  - ✓ Transient loops
  - ✓ Constant background traffic
- Shortest path computation: distributed Bellman-Ford; Dijkstra if the entire topology is known

Distance Vector

- Examples : RIPv1; RIPv2 (CIDR)
- Data structure : destination / next hop / distance
- Update example: send (destination, distance) pairs



Routing table in A before a-c appears

dest.	next hop	distance
b	b	1
c	e	2
d	e	3
e	e	1
f	e	2
g	e	4

Routing table in C before a-c appears

dest.	next hop	distance
b	e	3
a	e	2
d	d	1
e	e	1
f	e	2
g	d	2

Routing table in A after a-c appeared

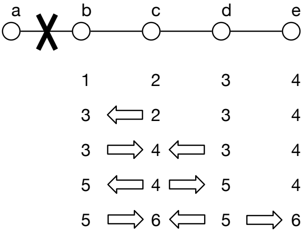
dest.	next hop	distance
b	b	1
c	c	1
d	c	2
e	e	1
f	e	2
g	c	3

DV (2)

- Used on the original ARPANET
- **Periodical** emission of DV (RIP: every 30s)
  or when a topology change is detected (link comes up, update received...)
- **Time stamping** of all entries: outdated information eventually vanishes
  Entries only last 3 minutes (5 missed updates)

DV limitations

Count to infinity

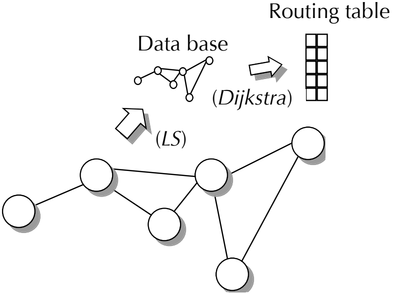


This process stops only when it reaches the upper limit (16)
"Split horizon" partially solves the problem

Graceful failure: send ∞ to neighbors

Link States routing protocols

- Examples : OSPF; IS-IS ; PNNI
- The protocol maintains a database that represents the network (or at least a part of it)
- Principle:



OSPF

- OSPF is the recommended routing protocol for within an autonomous system
- LSA (Link State Advertisement): topology information unit
- Hello process: neighbor discovery and check
- Reliable flooding of LSAs
- Local shortest path computation
- LSAs are time stamped and bear a serial number.
  The routers are responsible for refreshing the LSAs they inject

Flooding

- The easiest way of spreading information through a network!
- No topological information is required!
- Principle of flooding

Routing-less routing

- *ARP proxy*
  - ✓ Why? :
    - ▸ Layer 2 network $\neq$ Layer 3 network
    - ▸ Behind PPP link (see example)
    - ▸ Behind IP tunnel
  - ✓ Principe :
    - ▸ The access router answers ARP requests directed to the remote host
- ICMP redirect
  - ✓ ICMP is the IP signaling protocol
    TTL issues ($\rightarrow$ `traceroute`); packet size problem (ICMP frag. needed)
  - ✓ ICMP REDIRECT… (example)

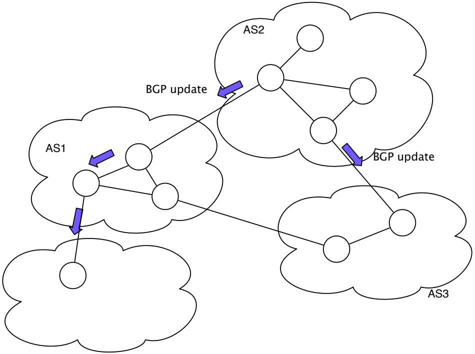
External routing

- Each operator is an “Autonomous System”

  AS exchange route updates using BGP
- Each route update carry
  1. A prefix
  2. The AS path (and a few other things…)

⇒
- ✓ Loop free path computation (path vector algorithm)
- ✓ Keeps track of several paths at the same time
- ✓ Insensitive to effective path length (The external path length is the number of ASs)
- ✓ Route filtering (avoid using given networks)
- ✓ Not totally immune to some kind of count to infinity

External routing (cont.)



Bellman-Ford Algorithm

We want to discover the distance to s from all vertices in (V, E).

- Data structure : distance to s from any vertex ($d(i)$); predecessor node ($p(i)$).

- Initialization : $d(s) = 0$; $d(j) = d_{sj}$ for all $j \in N(s)$.

- Iteration : repeat N times (N is the graph diameter)
  for all edges (jk) de E :

  if $d(k) > d(j) + d_{jk})$
  $d(k) \leftarrow d(j) + d_{jk}$; $p(k) \leftarrow j$

Dijkstra Algorithm

We want to discover the distance to s from all vertices in (V, E).

- data structure : Set of marked vertices : M;
  For each vertex i: distance to s $d(i)$; predecessor node $p(i)$.
- Initialization :
  $M = \{s\}$; $d(s) = 0$; $d(j) = d_{sj}$ for all $j \in N(s)$, ∞ otherwise;
  $p(j) = s \ \forall j \in N(s)$.
- Step 1: find next node to consider
  find $i \notin M$ such that $d(i) = \min_{j \notin M}(d(j))$; $M \leftarrow M \cup \{i\}$.
  if $M = V$ then stop.
- Step 2 : Update distances
  $\forall j \in N(i)$ s.t. $j \notin M$:
  If $d(j) > \min_{k \in N(j)}(d(k) + d_{kj})$ then:

  $p(j) \leftarrow \inf_{k \in N(j)}(d(k) + d_{kj})$; $d(j) \leftarrow \min_{k \in N(j)}(d(k) + d_{kj})$;
  end if
- Go back to step 1.