Interior Gateway Protocols: RIP & OSPF

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu

Based in part upon slides of Prof. Raj Jain (OSU), S. Keshav (Cornell), J. Kurose (U Mass)

- Routing Tables & static routing
- Dynamic routing (inter- and intra-domain)
- Distance vector vs Link state routing
- RIP, RIPv2
- OSPF
- Refs: Chap 9, 10.
- Books: "Routing in Internet" by Huitema, "Interconnections" by Perlman

Routing vs. *Forwarding*

- Forwarding: select an output port based on destination address and routing table
- Routing: process by which routing table is *built*
  - Function of *finding paths* in a network.
  - Can display routing table using "netstat -rn"

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

Routing Table Structure

- Fields: *destination, gateway, flags, ...*
- ***Destination***: can be a host address or a network address. If the 'H' flag is set, it is the host address.
- ***Gateway:*** router/next hop IP address. The 'G' flag says whether the destination is directly or indirectly connected.
- U flag: Is route up ?
- G flag: router (indirect vs direct)
- H flag: host (dest field: host or n/w address?)

Route Table Setup

- Route Table setup by:
  a) 'route' command [Static]
  b) ICMP redirect message.[Static]
  c) routing daemon. Eg: 'routed' [Dynamic]

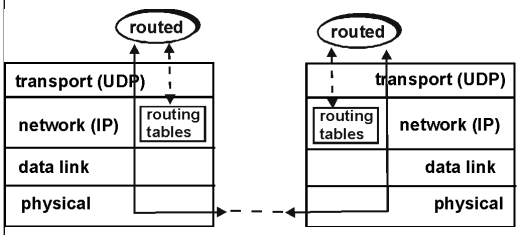
Static Routing

- Upon booting, default routes initialized from files. Eg: /etc/rc.net in AIX, /etc/netstart in BSD, /etc/rc.local in SUN/Solaris
- Use 'route' command to add new routes eg: route add default sun 1

Static Routing (Continued)

- ICMP redirect: sent to host by router when a "better" router exists on the same subnet.
- Alt: router discovery ICMP messages
  - Router solicitation request from host
  - Router advertisement messages from routers
- *Both ICMP methods are a form of "configuration" of static routes.*

Dynamic Routing Model



- A node makes a *local* choice depending on *global* topology: this is the fundamental problem

Key problem

- How to make correct local decisions?
  - each router must know *something about global state*
- Global state
  - inherently large
  - dynamic
  - hard to collect
- *A routing protocol must intelligently summarize relevant information*
- *Hard issues: consistency, completeness, scalability*

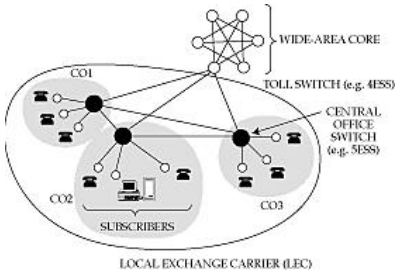
Requirements

- Minimize routing table space
  - fast to look up
  - less to exchange
- Minimize number & frequency of control pkts
- Robustness: avoid
  - black holes
  - loops
  - oscillations
- Use optimal path

Choices …

- *Centralized vs. distributed* routing
  - centralized is simpler, but prone to failure and congestion
- *Source-based vs. hop-by-hop*
  - how much is in packet header?
  - Intermediate: *loose source route*
- *Single vs. multiple path*
  - primary and alternative paths
- *State-dependent vs. state-independent*
  - do routes depend on current network state (e.g. delay)

Detour: Telephony routing



- 3-level hierarchy, with a fully-connected core
- AT&T: 135 core switches with nearly 5 million circuits
- LECs may connect to multiple cores

Telephony Routing algorithm

- If endpoints are within same CO, directly connect
- If call is between COs in same LEC, use one-hop path between COs
- Otherwise send call to one of the cores
- Only major decision is at toll switch
  - one-hop or two-hop path to the destination toll switch. (why don't we need longer paths?)
- Essence of problem:

  which *two-hop* path to use if *one-hop* path is full

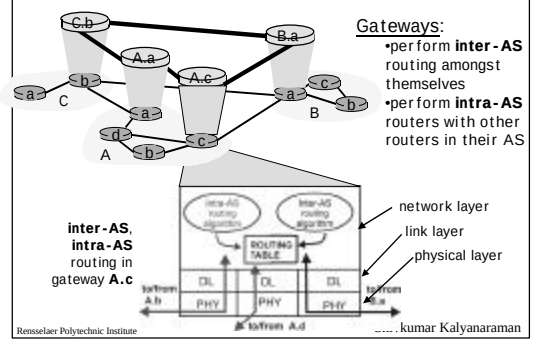
Features of telephone routing

- Stable load
  - can predict pairwise load throughout the day
  - can choose optimal routes in advance
- Extremely reliable switches
  - downtime is less than a few minutes per year
  - can assume that a chosen route is available
  - can't do this in the Internet
- Single organization controls entire core
  - can collect global statistics and implement global changes
- Very highly connected network
- Connections require resources (but all need the same)

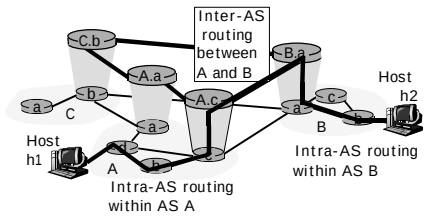
Internet Dynamic Routing

- Internet organized as *"autonomous systems" (AS).*
- Interior Gateway Protocols (IGPs) within AS.
  - Eg: RIP, OSPF, HELLO
- Exterior Gateway Protocols (EGPs) for AS to AS routing.
  - Eg: EGP, BGP-4

Intra-AS and Inter-AS routing



Intra-AS and Inter-AS routing: Example



Dynamic Routing Methods

- Source-based: chart route at src, given a map.
- Link state routing: Get *map* of network (in terms of *link states*) and calculate best route locally
- Distance vector: At every node, set up *distance signposts* to destination nodes (a vector)
  - Setup this by peeking at neighbors' signposts.

Distance Vector routing

- "Vector" of distances (signposts) to each possible destination at each router.
- How to find distances ?
  - Distance to local network is 0.
  - Look in neighbors' distance vectors, and add link cost to reach the neighbor
  - Find which direction yields minimum distance to to particular destination. Turn signpost that way.
  - Keep checking if neighbors change their signposts and modify local vector if necessary.
  - Called the "Bellman-Ford algorithm"
- Idea: At node i, for destination j:
  - Find neighbor/next-hop x that minimizes the sum

    D(i,j) = C(i,x) + D(x,j)

Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

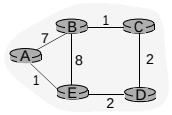
- each node communicates *only* with directly-attached neighbors

Distance Table data structure

- each node has its own
  - row for each possible destination
  - column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \underline{\textit{from}} \text{ X } \textit{to} \text{ Y, } \underline{\textit{via}} \text{ Z as next hop}$$
$$= c(X,Z) + \min_w\{D^Z(Y,w)\}$$

Distance Table: example



$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\}$
$= 2+2 = 4$

$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\}$
$= 2+3 = 5$ loop!

$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\}$
$= 8+6 = 14$ loop!

$D^E()$ — cost to destination via

destination	A	B	D
A	(1)	14	5
B	7	8	(5)
C	6	9	(4)
D	4	11	(2)

Distance table => routing table

$D^E()$ — cost to destination via

destination	A	B	D
A	(1)	14	5
B	7	8	(5)
C	6	9	(4)
D	4	11	(2)

Distance table → Routing table

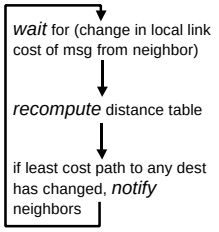
destination	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,4

Distance Vector Routing: *overview*

Iterative, asynchronous: each local iteration caused by:

- local link cost change
- message from neighbor: its least cost path change from neighbor
- neighbors then notify their neighbors if necessary

Each node:



Distance Vector *Algorithm*:

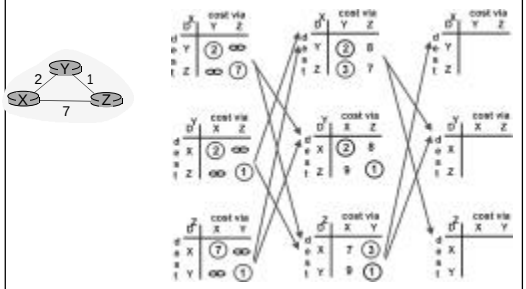
At all nodes, X:

```
1 Initialization:
2 for all adjacent nodes v:
3     D^X(*,v) = infty      /* the * operator means "for all rows" */
4     D^X(v,v) = c(X,v)
5 for all destinations, y
6     send min_w D^X(y,w) to each neighbor  /* w over all X's neighbors */
```

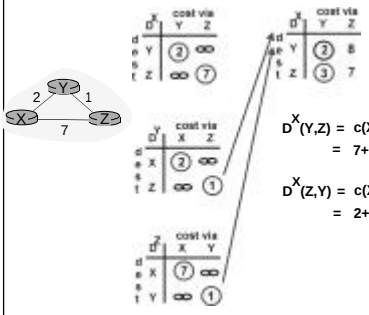
Distance Vector Algorithm (cont.):

```
8  loop
9   wait (until I see a link cost change to neighbor V
10        or until I receive update from neighbor V)
11
12  if (c(X,V) changes by d)
13    /* change cost to all dest's via neighbor v by d */
15    for all destinations y:  D^X(y,V) = D^X(y,V) + d
16
17  else if (update received from V wrt destination Y)
18    /* shortest path from V to some Y has changed  */
19    /* V has sent a new value for its  min   DV(Y,w) */
20    /* call this received new value is "newval"     */
21    for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23  if we have a new min_w D^X(Y,w) for any destination Y
24     send new value of min_w D^X(Y,w) to all neighbors
25
26  forever
```


Distance Vector Algorithm: *example*



Distance Vector Algorithm: *example*



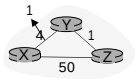
$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$
$= 7+1 = 8$

$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\}$
$= 2+1 = 3$

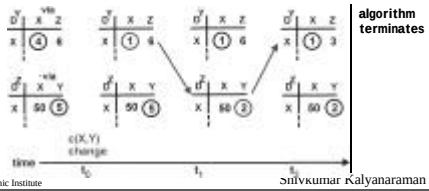
Distance Vector: *link cost changes*

Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)



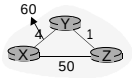
"good news travels fast"

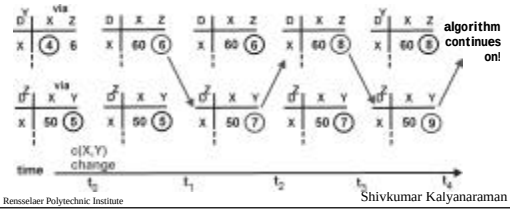


Distance Vector: *link cost changes*

Link cost changes:

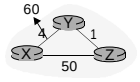
- good news travels fast
- bad news travels slow - "count to infinity" problem!

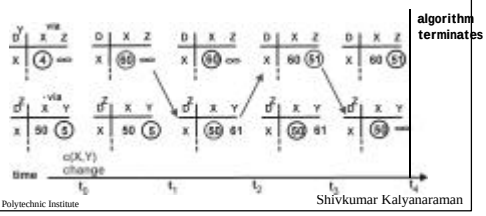




Distance Vector: *poisoned reverse*

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)





RIP: Routing Information Protocol

- Uses hop count as metric (max: 16 is infinity)
- Tables (vectors) "*advertised*" to neighbors every 30 s.
  - Each advertisement: upto 25 entries
- No advertisement for 180 sec: neighbor/link declared dead
  - routes via neighbor invalidated
  - new advertisements sent to neighbors (*Triggered updates)*
  - neighbors in turn send out new advertisements (if tables changed)
  - link failure info quickly propagates to entire net
  - *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIPv1 Problems (Continued)

- Split horizon/poison reverse does not guarantee to solve count-to-infinity problem
  - 16 = infinity => RIP for small networks only!
  - Slow convergence
- Broadcasts consume non-router resources
- RIPv1 does not support subnet masks (VLSMs)
  - No authentication

RIPv2

- Why ? Installed base of RIP routers
- Provides:
  - VLSM support
  - Authentication
  - Multicasting
  - "Wire-sharing" by multiple routing domains,
  - Tags to support EGP/BGP routes.
- Uses reserved fields in RIPv1 header.
- First route entry replaced by authentication info.

Link State Protocols

- Key: Create a network "*map*" at each node.
- 1. Node collects the state of its connected links and forms a "Link State Packet" (LSP)
- 2. Flood LSP => reaches every other node in the network and everyone now has a network map.
- 3. Given map, run Dijkstra's shortest path algorithm (SPF) => get paths to all destinations
- 4. Routing table = next-hops of these paths.

Dijkstra's algorithm

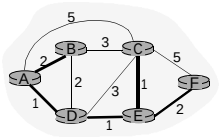
- Net topology, link costs known to all nodes
  - accomplished via *"link state broadcast"*
  - all nodes have same info
- computes least-cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- iterative: after *k iterations*, know *least cost path to k dest.*'s
- Notation:
- c(i,j): *link cost* from node i to j. cost infinite if not direct neighbors
- D(v): current value of *path cost* from source to dest. V
- p(v): *predecessor node* along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

Dijkstra's Algorithm

```
1  Initialization:
2    N = {A}
3    for all nodes v
4      if v adjacent to A
5        then D(v) = c(A,v)
6        else D(v) = infty
7
8  Loop
9    find w not in N such that D(w) is a minimum
10   add w to N
11   update D(v) for all v adjacent to w and not in N:
12      D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N
```

Dijkstra's algorithm: *example*

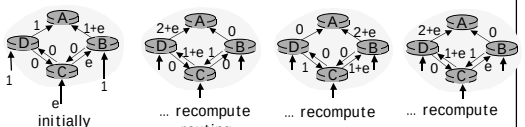
Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Dijkstra's algorithm, discussion

Algorithm complexity: n nodes
- each iteration: need to check all nodes, w, not in N
- n*(n+1)/2 comparisons: O(n**2)
- more efficient implementations possible: O(nlogn)

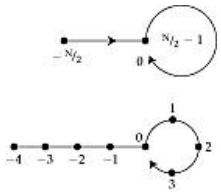
Oscillations possible:
- e.g., link cost = amount of carried traffic



Link State: Topology Dissemination

- A.k.a LSP distribution
- 1. *Flood* LSPs on links except incoming link
  - Require at most 2E transfers for n/w with E edges
- 2. *Sequence numbers* to detect duplicates
  - Why? Routers/links may go down/up
  - Problem: wrap-around => have large seq # space, *lollipop sequence*

Lollipop Sequence



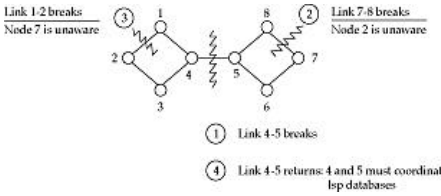
- Need a *unique* start sequence number
- Comparison: a is older than b if:
  - a < 0 and a < b
  - a > 0, a < b, and b-a < N/4
  - a > 0, b > 0, a > b, and a-b > N/4

Topology Dissemination (Continued)

- 3. *Age field* (similar to TTL)
  - Periodically decremented after acceptance
  - Zero => discard LSP & request everyone to do so
  - Router awakens => knows that all its old LSPs would have been purged and can choose a new initial sequence number

Recovering from a partition

- On partition, LSP databases can get out of synch

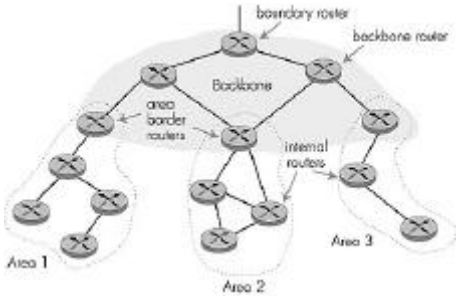


- Databases described by database descriptor records
- Routers on each side of a newly restored link talk to each other to update databases (determine missing and out-of-date LSPs) => selective synchronization

OSPF "advanced" features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (eg, satellite link cost set "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.

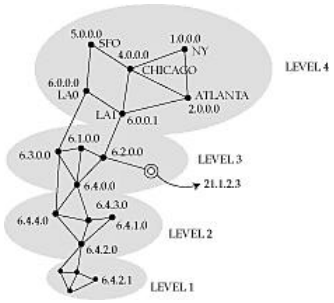
Hierarchical OSPF



Hierarchical OSPF

- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes 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 ASs (generate "*external*" records)

Example



External and *summary* records

- If a domain has multiple gateways
  - *External* records tell hosts in a domain which one to pick to reach a host in an external domain
    - e.g allows 6.4.0.0 to discover shortest path to 5.* is through 6.0.0.0
  - *Summary* records tell backbone (and areas) which gateway to use to reach an internal node
    - e.g. allows 5.0.0.0 to discover shortest path to 6.4.0.0 is through 6.0.0.0

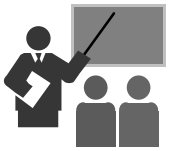
E-IGRP (Interior Gateway Routing Protocol)

- CISCO proprietary; successor of RIP (late 80s)
- Several metrics (delay, bandwidth, reliability, load etc)
- Uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Alg. (DUAL) based on *diffused computation*
  - Freeze entry to particular destination
  - Diffuse a request for updates
  - Other nodes may freeze/propagate the diffusing computation (tree formation)
  - Unfreeze when updates received.
  - Tradeoff: temporary un-reachability for some destinations

Link State vs. Distance Vector

- Link State (LS) advantages:
  - More stable (aka fewer routing loops)
  - Faster convergence than distance vector
  - Easier to discover network topology, troubleshoot network.
  - Can do better source-routing with link-state
  - Type & Quality-of-service routing (multiple route tables) possible
- Caveat: With path-vector-type (paths instead of distances) DV routing, these differences blur…

Summary



- Basic routing concepts, Route Tables
- Distance vector, Bellman-Ford, RIP, RIPv2
- Link state, Dijkstra, OSPF
- DUAL/EIGRP etc