CSE 222
Graduate Networking

Winter 2001

Lecture 7: Routing

Geoffrey M. Voelker

Roadmap

- We've discussed
  - How to connect networks together
  - How to reliably transmit packets across networks
  - How to cooperatively share network resources
- Now we'll look at how we get packets from one end host to another across connected networks
  - Theory this time
  - Practice next
- Not discussing…(perhaps another time)
  - Hubs, bridges, and switches
  - Arbitration of shared media (broadcast radio, Ethernet bus, token ring)

Routing Overview

- Today
  - Routing algorithm fundamentals
    - » Distance vector
    - » Link state
  - Case study on IP router internals [Partridge98]
- Next time
  - Internet routing: BGP
  - Realization of Internet routing today:
    - » IP addresses, prefixes, CIDR
    - » BGP, ASes, and ISPs
  - Effects of Internet routing as perceived by the end points

Routing Challenges

- How to choose best path?
  - Shortest path, but shortest according to what metric?
- How to scale to millions of users?
  - Minimize control messages and routing table size
- How to adapt to failures and changes?
  - Node and link failures, message loss
  - Use distributed algorithms
    - » These algorithms run on same network that is being computed
    - » Also have to tolerate node and link failures, message loss

Distributed Routing

- Two basic algorithms for distributed routing

1. Distance vector (RIP, BGP)
   - Exchange routing tables with neighbors
   - No one knows complete topology
   - Currently used between admin domains
2. Link state (OSPF)
   - Send everyone your neighbors
   - Everyone computes shortest path, global knowledge
   - Currently used within admin domains

Note: Slides thanks to David Weatherall (UW)

Routing As A Graph Problem

- Represent the network as a graph
  - Routers → nodes
  - Links → edges
  - Delay, hops → cost
  - "Efficiently" → compute shortest path
- Ignore end hosts, assume connected to a router
- Graph theory heaven

Distance Vector Routing

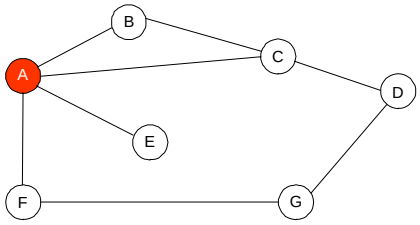
- Assume:
  - Each router knows only address & cost of neighbors
- Goal:
  - Calculate routing table containing next-hop information for each destination at each router
- Distance vector approach:
  - Tell neighbors about learned distances to all destinations
  - Distributed and iterative

Distance Vector Algorithm

- Each router maintains a vector of costs to all destinations and a routing table
  - Initialize neighbors with known cost, others with infinity
- Periodically send copy of distance vector to neighbors
  - On reception of a vector
    - » If neighbor's path to a destination is better, switch to it
    - » Update cost in vector and next hop in routing table
- Assuming no changes, will converge to shortest paths
  - Of course, there are changes…

Distance Vector Example

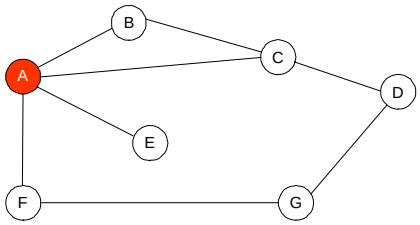
- Graph with 7 nodes
- Initial cost vector at node A
  - Routing from A's perspective



Dest	Cost	Next
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	E
G	∞	-

Example Result

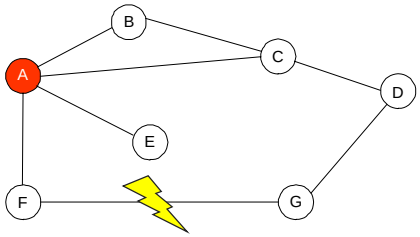
- Reached in a single iteration
  - Simple example



Dest	Cost	Next
B	1	B
C	1	C
D	**2**	C
E	1	E
F	1	E
G	**2**	F

Link Failure

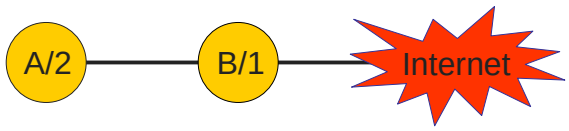
- Suppose link between F and G fails
  - F notices failure, sets its cost to G to infinity and tells A
  - A sets its cost to G to infinity, too, since it learned it from F
  - A learns route from C with cost 2 and adopts it



Dest	Cost	Next
B	1	B
C	1	C
D	**2**	C
E	1	E
F	1	E
G	**3**	C

Count To Infinity Problem

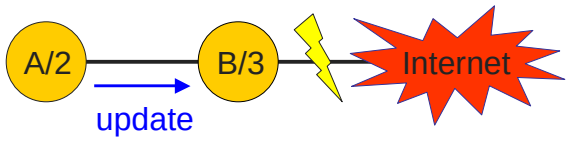
- Simple example
  - Nodes represent costs to reach Internet



- Now link between B and Internet fails…

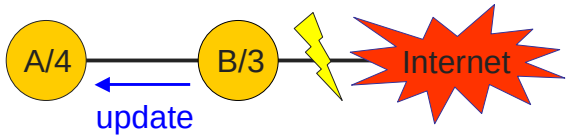
Count To Infinity Problem (2)

- B hears of a route to the Internet via A with cost 2
- So B switches to the “better” (but wrong!) route



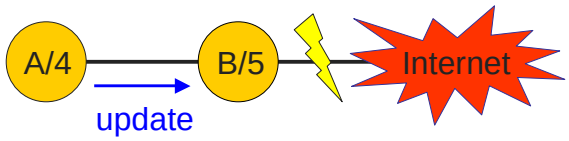
Count To Infinity Problem (3)

- A hears from B and increases its cost



Count To Infinity Problem (4)

- Cycle continues as we count to infinity



Split Horizon

- How do we avoid counting to infinity?
- Split horizon
  - Router never advertises the cost of a destination back to its next hop
    - » That's where we learned it from!
- Poison reverse
  - One step further → advertise back infinity
- Unfortunately, DV protocols still subject to the same problem with more complicated topologies
  - State not stable as routes being updated
  - Many enhancements proposed

Routing Information Protocol

- RIP is a DV protocol with hop count as metric
  - Infinity value is 16 hops
    - » Effectively limits network size
  - Includes split horizon with poison reverse
- Routers send vectors every 30 seconds (keep-alives)
  - Link failures trigger immediate updates
  - Timeout in 180 seconds to detect failures
- Specifications
  - RIPv1 specified in RFC1058
    - » http://www.ietf.org/rfc/rfc1058.txt
  - RIPv2 (adds authentication etc.) in RFC1388
    - » http://www.ietf.org/rfc/rfc1388.txt

Why Hop Count?

- Latency used in original ARPAnet
  - Dynamically unstable (function of load)
  - Penalized satellite links (might favor bandwidth over delay)
- Hop count yields unique loop-free path
  - Reflects router processing overhead consumed by packet

Link State Routing

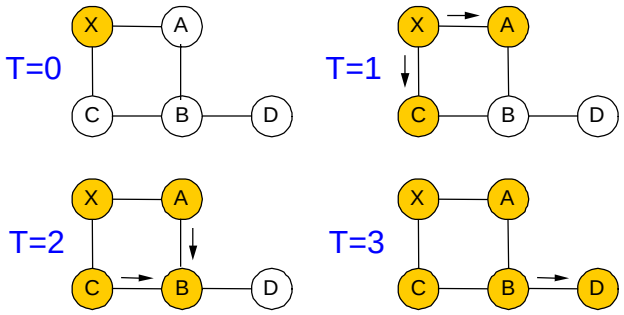
- Same assumptions and goals, but different approach:
  - Tell all routers the topology, have each compute best paths
  - Two phases
    - » Topology dissemination (flooding)
    - » Shortest-path calculation (Dijkstra's algorithm)
- Motivation
  - In DV, routers hide their computation, making it difficult to decide what to do when there are changes
    - » Each router only has knowledge of its neighbors, not global
  - With LS, faster convergence and better stability (hopefully)
  - More complex, though

Flooding

- Each router maintains a link state database and periodically sends link state packets (LSPs) to neighbor
  - LSP contains [router, neighbors, costs]
- Each router forwards LSPs not already in its database on all ports except where received
  - Each LSP will travel over the same link at most once in each direction
- Flooding is fast, and can be made reliable with ACKs

Flooding Example

- LSP generated by X at T=0
- Nodes become orange as they receive it



Link State Problems

- When link/router fails need to remove old data…how?
  - LSPs carry sequence numbers to distinguish new from old
  - Send a new LSP with cost infinity to signal a link down
- What happens when a rouer fails and restarts?
  - What sequence # should it use? Don't want data ignored
  - One option: Age LSPs and send with cost 0 to purge
  - Router can listen at startup to learn right sequence #
- What happens if the network is partitioned and heals?
  - Different LS databases must be synchronized
  - Use version #s

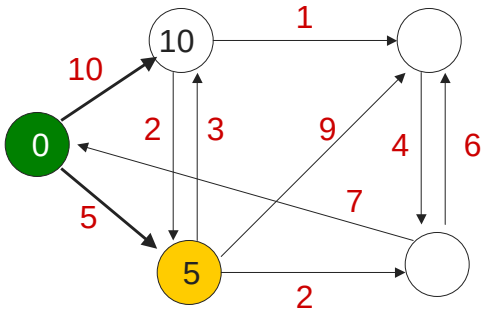
Dijkstra's Algorithm

- Graph algorithm for single-source shortest path

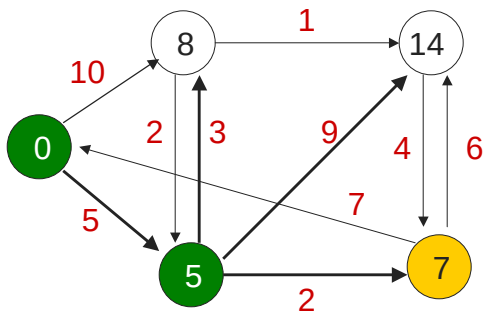
```
S ← {}
Q ← <all nodes keyed by distance>
While Q != {}
        u ← extract-min(Q)
        S ← S plus {u}
        for each node v adjacent to u
                "relax" the cost of v
```

← u is done

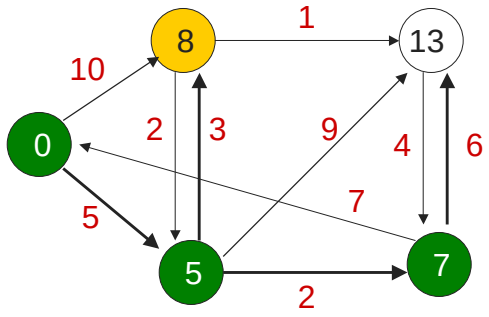
Example – Step 2



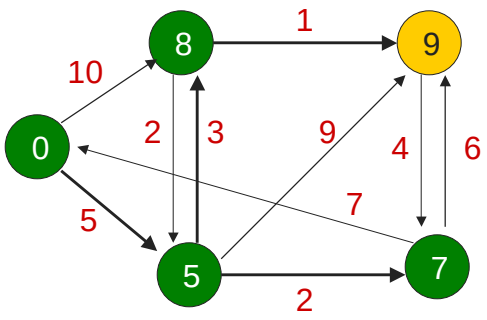
Example – Step 3



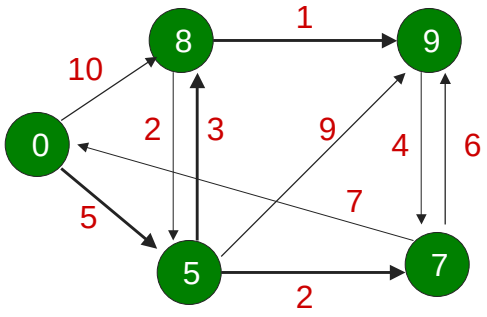
Example – Step 4



Example – Step 5



Example – Done



Link State Summary

- Note that each router
  - Executes this algorithm
  - Knows global state of network

Open Shortest Path First

- OSPF is most widely-used link state protocol today
- Basic link state algorithm plus many features:
  - Authentication of routing messages
  - Extra hierarchy: Partition into routing areas
  - Load balancing: Multiple equal cost routes

Distributed Routing Summary

- Determining global state is challenging
  - Hard to collect
  - Can be out-of-date once collected
  - Need to summarize in a locally-relevant way
- Inconsistencies in local/global state can cause
  - Loops
  - Oscillations, esp. when adapting to load

A 50 Gb/s IP Router

- Not going to go into this in too much detail
  - No papers on routing basics, thought you would enjoy this
- High-level points
  - Instructions/packet needs to be very low
    - » Can't make fancy routing decisions per packet
    - » Even IP options are off the fast path
    - » FYI: Varghese is the expert on high-speed, efficient IP lookups
  - General-purpose processor → flexibility
    - » Can update forwarder and network processor as innovations are made in IP routing (e.g., RED, QoS)
  - Surprised that the network processor runs BSD Unix?
    - » Cisco's version is IOS

Discussion

- What is the impact of not doing IP header checksum?
  - What if all routers in a path did not check checksum?
- How do you implement RED?

  One minor question that the paper didn't answer for me was this: They make mention of running RED on this router, but I didn't understand how RED would be able to mark (or drop) a packet, or how it could compute average queue size for that matter! It seems that the separation of the forwarding decisions from the routing decisions breaks RED because what they call the network processor (where RED would run) doesn't see every packet come through, so how can it decide which packet to drop?
- Is it fast enough? How can you decide?

  Unfortunately, the paper's router improvements are not great enough to keep routers alive for that much longer.

For Next Time...

- Read Section 4.3
- Read Paxson96, Norton00