CSE 123b
Communications Software

Spring 2002

Lecture 6: Routing: Overview and
Distance Vector Algorithms

Stefan Savage

Some slides courtesy David Wetherall

Last class

- Goals of congestion control
  - Use allocated bandwidth efficiently
  - Avoid sending so quickly that the network has to drop packets
  - Avoid sending so slowly that the network is underutilized
- Approach taken by TCP
  - Congestion window limits outstanding packets
  - Adjust congestion window in response to packet losses (AIMD)
  - Slow start
  - Fast retransmit/ fast recovery

This class

- New topic: **routing**

Overview

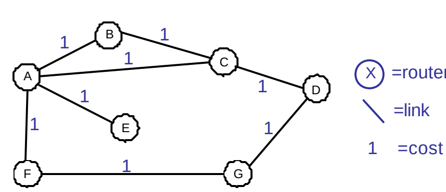
- **Intro & Design choices**
- **Intra-domain routing**
  - **Distance vector**
  - Link state
- Inter-domain routing
  - Policy

Intra-domain routing

- Routing **within** a network/organization
- A **single** administrative domain
- Overall goals
  - Provide intra-network connectivity
  - Adapt quickly to failures or topology changes
  - **Optimize** use of network resources
- Problem statement
  - Network is a directed graph G=(V,E)
  - Routers are vertices, links are edges labeled with some metric
    - » For simplicity ignore hosts, they are part of each V
  - For each V, find the shortest path to every other V

Network as a Graph

- Routing is essentially a problem in graph theory
- Find “best” path between every pair of vertices



Routing Questions

- How to choose best path?
  - Defining “best” can be slippery
- How to scale to millions of users?
  - Minimize control messages and routing table size
- How to adapt to failures or changes?
  - Node and link failures, plus message loss

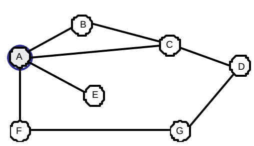
What does a router do?

- **Forwarding**
  - Move packet from input link to the appropriate output link
  - Purely local computation
  - Must go be very fast (executed for ever packet)
- **Routing**
  - Doing work so you’re sure that the “next hop” actually leads to the destination
  - Global decisions; distributed computation and communication
  - Can go slower (only important when topology changes)

What’s in a Routing Table?

- The routing table at A, for example, lists at a minimum the next hops for the different destinations

Dest	Next Hop
B	B
C	C
D	C
E	E
F	F
G	F



Kinds of routing/forwarding

- Source routing
  - Complete path in packet
- Virtual circuits
  - Set up path out-of-band and store path identifier in routers
  - Local path identifier in packet
- **Destination-based routing**
  - Router looks up address in forwarding table
  - Forwarding table contains (address, next-hop) tuples

Source routing

- Routing
  - Host computes path
  - Must know global topology and detect failures
  - Packet contains complete ordered path information
    - » I.e. node A then D then X then J…
  - Requires variable length path header
- Forwarding
  - Router looks up next hop in packet header, strips it off and forwards remaining packet
  - Very quick forwarding, no lookup required
  - Very flexible
- In practice
  - ad hoc networks (DSR), SANs (Myrinet), and for debugging on the Internet (LSR,SSR)

Virtual circuits

- Routing
  - Hosts sets up path out-of-band, requires connection setup
  - Write (input id, output id, next hop) into each router on path
  - Flexible (one path per flow)
- Forwarding
  - Send packet with path id
  - Router looks up input, swaps for output, forwards on next hop
  - Repeat until reach destination
  - Table lookup for forwarding (why faster than IP lookup?)
- In practice
  - ATM: fixed VC identifiers and separate signaling code
  - MPLS: ATM meets the IP world (why? *traffic engineering*)

Destination-based routing

- Routing
  - All addresses are globally known
    - » No connection setup
  - Host sends packet with destination address in header
    - » No path state; only routers need to worry about failure
  - Distributed routing protocol used to routing tables
- Forwarding
  - Router looks up destination in table
    - » Must keep state proportional to destinations rather than connections
  - (Address, next-hop) tuple
  - Lookup address, send packet to next-hop link
    - » All packets follow same path to destination
- In Practice: IP routing

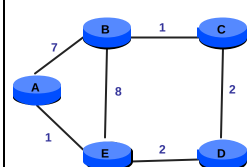
Three approaches to routing

- **Static**
  - Type in the right answers and hope they are always true
- **Distance vector**
  - Tell your neighbors when you know about everyone
- **Link state**
  - Tell everyone what you know about your neighbors

Distance Vector routing

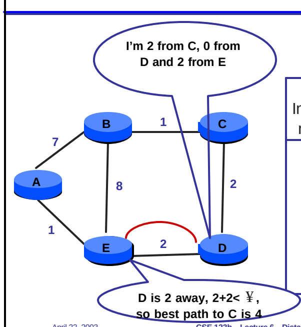
- Assume
  - Each router knows own address & cost to reach neighbors
- Goal
  - Calculate routing table containing next-hop information for every destination at each router
- **Distributed Bellman-Ford algorithm**
  - Each router maintains a vector of costs to all destinations
    - » Initialize neighbors 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 shorter, switch to it

Initial conditions



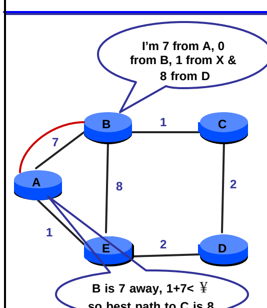
Info at node	Distance to Node A	B	C	D	E
A	0	7	¥	¥	1
B	7	0	1	¥	8
C	¥	1	0	2	¥
D	¥	¥	2	0	2
E	1	8	¥	2	0

E receives D's vector



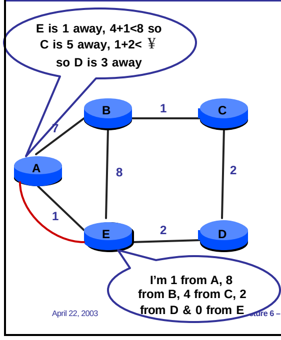
Info at node	Distance to Node A	B	C	D	E
A	0	7	¥	¥	1
B	7	0	1	¥	8
C	¥	1	0	2	¥
D	¥	¥	2	0	2
E	1	8	4	2	0

A receives B's vector



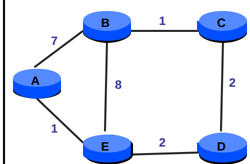
Info at node	Distance to Node A	B	C	D	E
A	0	7	8	¥	1
B	7	0	1	¥	8
C	¥	1	0	2	¥
D	¥	¥	2	0	2
E	1	8	4	2	0

A receives E's vector



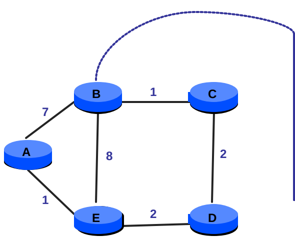
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	5	3	1
B	7	0	1	¥	8
C	¥	1	0	2	¥
D	¥	¥	2	0	2
E	1	8	4	2	0

Final state



Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

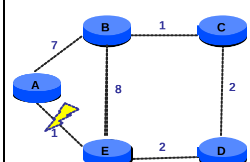
View from a node (B)



Dest	Next hop		
	A	**E**	**C**
A	7	9	**6**
C	12	12	**1**
D	10	10	**3**
E	8	8	**5**

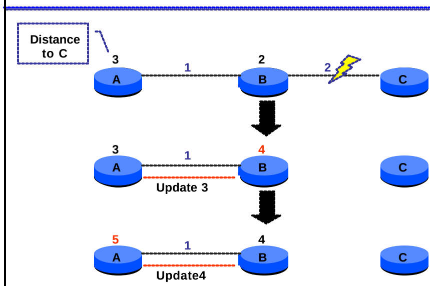
Link failure

- **A marks distance to E as ¥ , and tells B**
- **E marks distance to A as ¥ , and tells B and D**
- **B and D recompute routes and tell C, E and E**
- **etc… until converge**



Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	8	10	12
B	7	0	1	3	5
C	8	1	0	2	4
D	3	3	2	0	2
E	12	5	4	10	0

Problems: *Count to Infinity*



Etc…

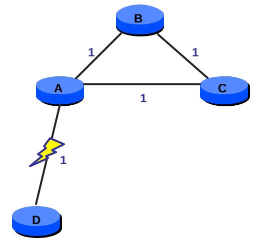
Why?

- Updates don't contain enough information
- Can't totally order bad news above good news
- B's accepts A's path to C that is *implicitly* through B!
- Aside: this also causes delays in convergence

Solutions

- **Hold downs**
  - As metric increases, delay propagating information
  - Limitation: Delays convergence
- **Split horizon**
  - Never advertise a destination through its next hop
    - » A doesn't advertise C to B
  - Poison reverse: Send negative information when advertising a destination through its next hop
    - » A advertises C to B with a metric of ¥
  - Limitation: Only works for "loop"s of size 2
- **Loop avoidance**
  - Full path information in route advertisement
  - Explicit queries for loops (e.g. DUAL)

How split horizon/pv fails



- **A tells B & C that D is unreachable**
- **B tells C that D is unreachable**
- **B tells A that D is reachable with cost=3 (since route is through C, split horizon doesn't apply)**
- **A tells C that D is reachable through A (cost=4)**
- **Etc…**

Other issues

- When to send route updates?
- **Periodically**
  - Limits granularity of failure recovery
  - Global synchronization can cause packet loss
- **Jittered**
  - Random offset from periodic deals with synchronization problem
- **Triggered**
  - Send updates immediately when metric changes
  - Converges more quickly, but causes flood of packets

Routing Information Protocol (RIP)

- DV protocol with hop count as metric
  - Infinity value is 16 hops; limits network size
  - Includes split horizon with poison reverse
- Routers send vectors every 30 seconds
  - With triggered updates for link failures
  - Time-out in 180 seconds to detect failures
- RIPv1 specified in RFC1058
  - www.ietf.org/rfc/rfc1058.txt
- RIPv2 (adds authentication etc.) in RFC1388
  - www.ietf.org/rfc/rfc1388.txt

Key Concepts

- Routing is a global process, forwarding is local one
- The Distance Vector algorithm and RIP
  - Simple and distributed exchange of shortest paths.
  - Weak at adapting to changes (loops, count to infinity)

For next time…

- No new reading… although review the section about link state routing