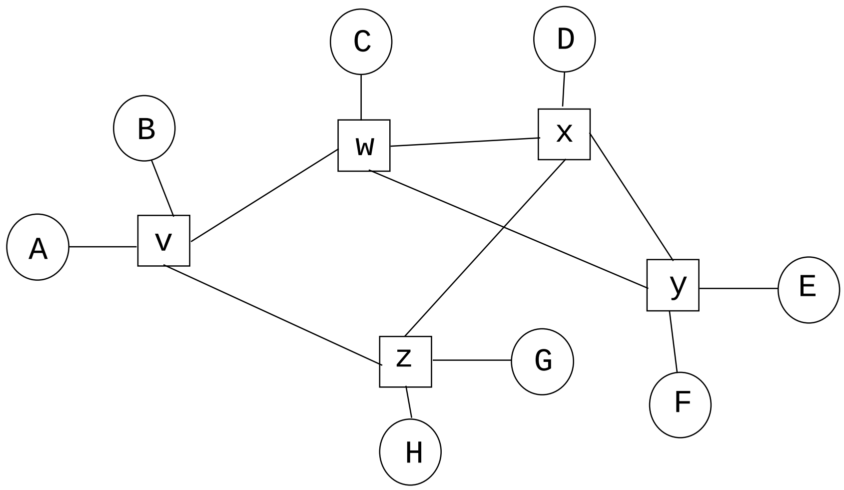
# CSE/EE 461 Lecture 6
# Routing

Tom Anderson
tom@cs.washington.edu

## Roadmap

- *LAN vs. Internet routing*
  - *MAC vs. IP addresses*
- *Forwarding mechanisms*
  - *Source routing*
  - *Global addresses*
  - Virtual circuits
- Routing algorithms
  - spanning tree
  - distance vector
  - link state

# Forwarding Mechanics Example




# Virtual Circuits (ATM, MPLS)

- Each connection has destination address; each packet has virtual circuit ID (VCI)
- Each switch has forwarding table of connection -> next hop
  - at connection setup, allocate virtual circuit ID (VCI) at each switch in path
  - (input #, input VCI) -> (output #, output VCI)
    - At v: (west=A, 12) -> (east=w, 2)
    - At w: (west=v, 2) -> (south=y, 7)
    - At y: (north=w, 7) -> (south=F, 4)

# Virtual Circuits

- Advantages
  - more efficient lookup (smaller tables)
  - more flexible (different path for each circuit)
  - can reserve bandwidth at connection setup
- Disadvantages
  - still need to route connection setup request
  - more complex failure recovery

# Comparison

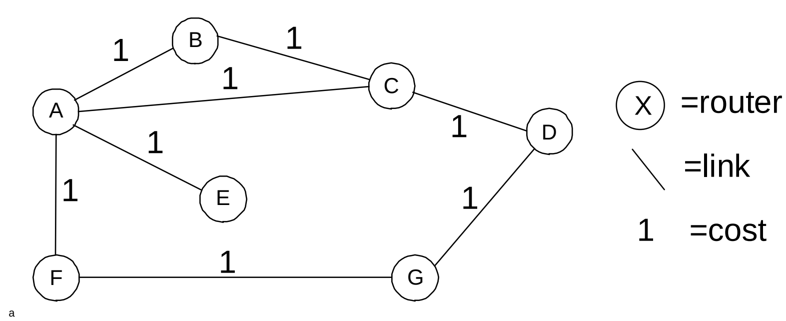
| | Source routing | Global addresses | Virtual circuits |
|---|---|---|---|
| Header size | worst | OK ~ large addrs | best |
| Router table size | none | # of hosts (prefixes) | # of circuits |
| Forward overhead | best | Prefix matching | Pretty good |
| Setup overhead | none | none | Same as global addr forwarding |
| Error recovery | Tell all hosts | Tell all routers | Tear down circuit and reroute |

# Routing Questions

- How to choose best path?
  - Defining "best" is slippery
- How to scale to billions of hosts?
  - Minimize control messages and routing table size
- How to adapt to failures or changes?
  - Node and link failures, plus message loss
  - We will use distributed algorithms
- Use global or local knowledge?
  - Inconsistencies can cause loops and oscillations

# Network as a Graph

- Routing is essentially a problem in graph theory
  - switches = nodes; links = edges; delay/hops = cost
- Need dynamic computation to adapt to changes

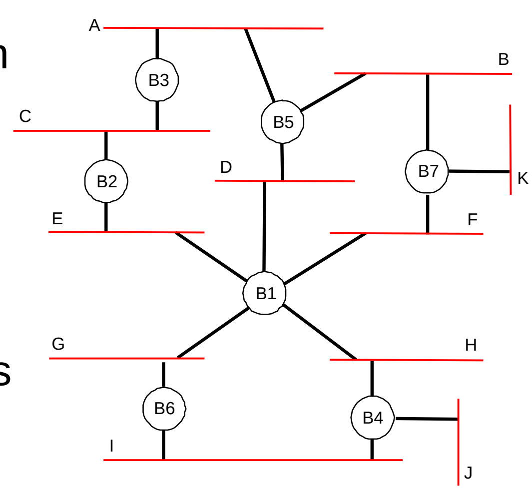


# Routing Alternatives

- Spanning Tree (Ethernet)
  - Convert graph into a tree; route only along tree
- Distance vector (RIP, BGP)
  - exchange routing tables with neighbors
  - no one knows complete topology
- Link state (OSPF)
  - send everyone your neighbors
  - everyone computes shortest path

# Spanning Tree Example

- Convert graph into a tree; route only along the tree
- Simple and avoids loops



# Spanning Tree Algorithm

- Distributed algorithm to compute spanning tree
  - Robust against failures, needs no organization
- Outline:
  - Elect a root node of the tree (lowest address)
  - Grow tree as shortest distances from the root (using lowest address to break distance ties)

# Algorithm

- Bridges periodically exchange config messages
  - Contain: best root seen, distance to root, bridge address
- Initially, each bridge thinks it is the root
  - Each bridge tells its neighbors its address
- On receiving a config message, update position in tree
  - Pick smaller root address, then
  - Shorter distance to root, then
  - Bridge with smaller address
- Periodically update neighbors
  - Add one to distance to root, send downstream
- Turn off forwarding on ports except those that send/receive “best”

## Algorithm Example

- Message format: (root, dist to root, bridge)
- Sample messages sequences to and from B3:
  - B3 sends (B3, 0, B3) to B2 and B5
  - B3 receives (B2, 0, B2) and (B5, 0, B5) and accepts B2 as root
  - B3 sends (B2, 1, B3) to B5
  - B3 receives (B1, 1, B2) and (B1, 1, B5) and accepts B1 as root
  - B3 wants to send (B1, 2, B3) but doesn't as its nowhere "best"
  - B3 receives (B1, 1, B2) and (B1, 1, B5) again … stable
  - Data forwarding is turned off to A

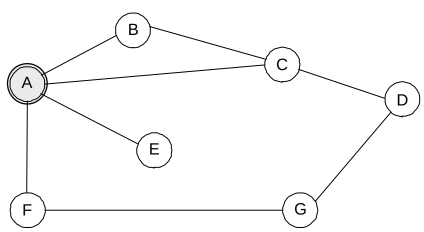
## Some other details

- Configuration information is aged
  - If the root fails a new one will be elected
- Reconfiguration is damped
  - Adopt new spanning trees slowly to avoid temporary loops

# Distance Vector Routing

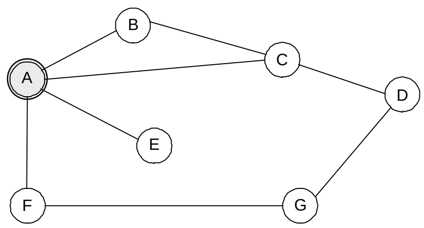
- Each router periodically exchanges messages with neighbors
  - best known distance to each destination (“distance vector”)
- Initially, can get to self with 0 cost
- On receipt of update from neighbor, for each destination
  - switch forwarding tables to neighbor if it has cheaper route
  - update best known distance
  - tell neighbors of any changes
- Absent topology changes, will converge to shortest path

# DV Example: Initial Table at A




| Dest | Cost | Next |
|---|---|---|
| A | 0 | here |
| B | ∞ | - |
| C | ∞ | - |
| D | ∞ | - |
| E | ∞ | - |
| F | ∞ | - |
| G | ∞ | - |

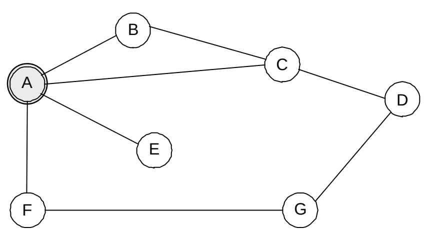
# DV Example: Table at A, step 1




| Dest | Cost | Next |
|---|---|---|
| A | 0 | here |
| B | 1 | B |
| C | 1 | C |
| D | ∞ | - |
| E | 1 | E |
| F | 1 | F |
| G | ∞ | - |

# DV Example: Final Table at A

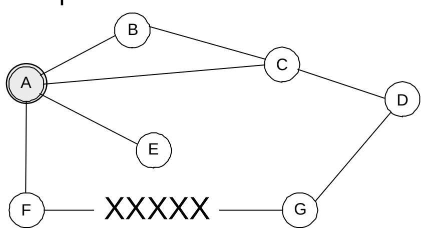
Reached in two iterations
=> simple example




| Dest | Cost | Next |
|---|---|---|
| A | 0 | here |
| B | 1 | B |
| C | 1 | C |
| D | 2 | C |
| E | 1 | E |
| F | 1 | F |
| G | 2 | F |

# What if there are changes?

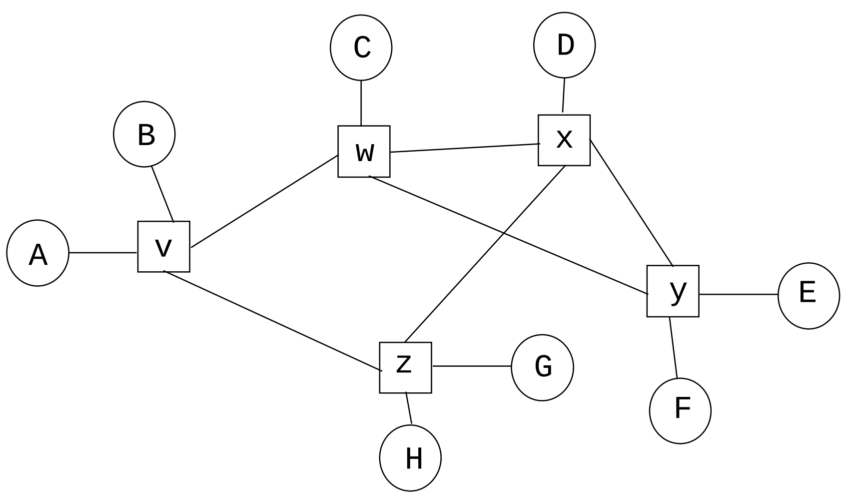
- Suppose link between F and G fails
  - F notices failure, sets its cost to G to infinity and tells A
  - A sets its cost to G to infinity too, since it can't use F
  - A learns route from C with cost 2 and adopts it




a

| Dest | Cost | Next |
|---|---|---|
| A | 0 | here |
| B | 1 | B |
| C | 1 | C |
| D | 2 | C |
| E | 1 | E |
| F | 1 | F |
| G | 3 | F |

# A More Complex Example



## A More Complex Example

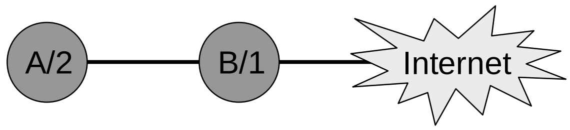
- Step 0: v knows about itself, A, B
- Step 1: v learns about C, G, H
- Step 2: v learns about D, E, F
  - D from both w and z
- Step 3: v learns about alternate routes to C, E, F, G, H

## Why Hop Count as Cost Metric?

- Latency as metric used in original ARPAnet
  - dynamically unstable
  - penalized satellite links
- Hop count yields unique loop-free path
  - reflects router processing overhead consumed by packet
- Can we design a dynamically stable adaptive routing algorithm?

# Count To Infinity Problem

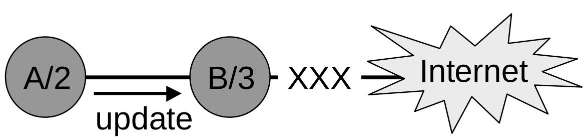
- Simple example
  - Costs in nodes are to reach Internet




- Now link between B and Internet fails …

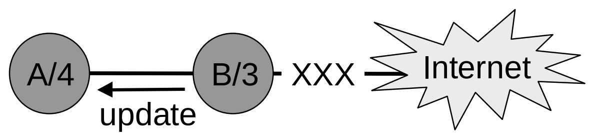
# Count To Infinity Problem

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



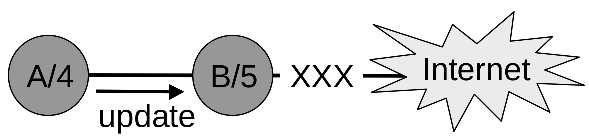
## Count To Infinity Problem

- A hears from B and increases its cost




## Count To Infinity Problem

- B hears from A and (surprise) increases its cost
- Cycle continues and we “count to infinity”




- Packets caught in the crossfire loop between A and B

## Solutions

- Split horizon
  - Router never advertises the cost of a destination back to its next hop – that's where it learned it from!
  - Solves trivial count-to-infinity problem
- Poison reverse (RIP)
  - go farther: advertise infinity back to source
  - vulnerable to more complex topology changes
- Path vector (BGP)
  - announce entire path to each destination
  - easy to check for loops

## 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