Routing Concepts

TCP/IP class

Jim Binkley

Routing - Concepts

- intro concepts
  - topologies
  - types/properties
  - issues
- vector-distance
- link-state
- see Radia Perlman's *Interconnections*, for more information

issues/properties

- **routing** - finding a path from one end to the other for a packet
- we need one or more **algorithms** that are most likely **distributed** amongst a set of hosts and router
- what are the properties of said algorithm?
- what issues affect it?

elements of a routing scheme

- **routing protocols** that allow info to be gathered and distributed - routing agents communicate with these protocols
- **routing algorithms** - may be distributed, use protocols and data to determine and disseminate paths
- **routing databases** (tables in routers) (to boardwalk, via new jersey, $100)

topology

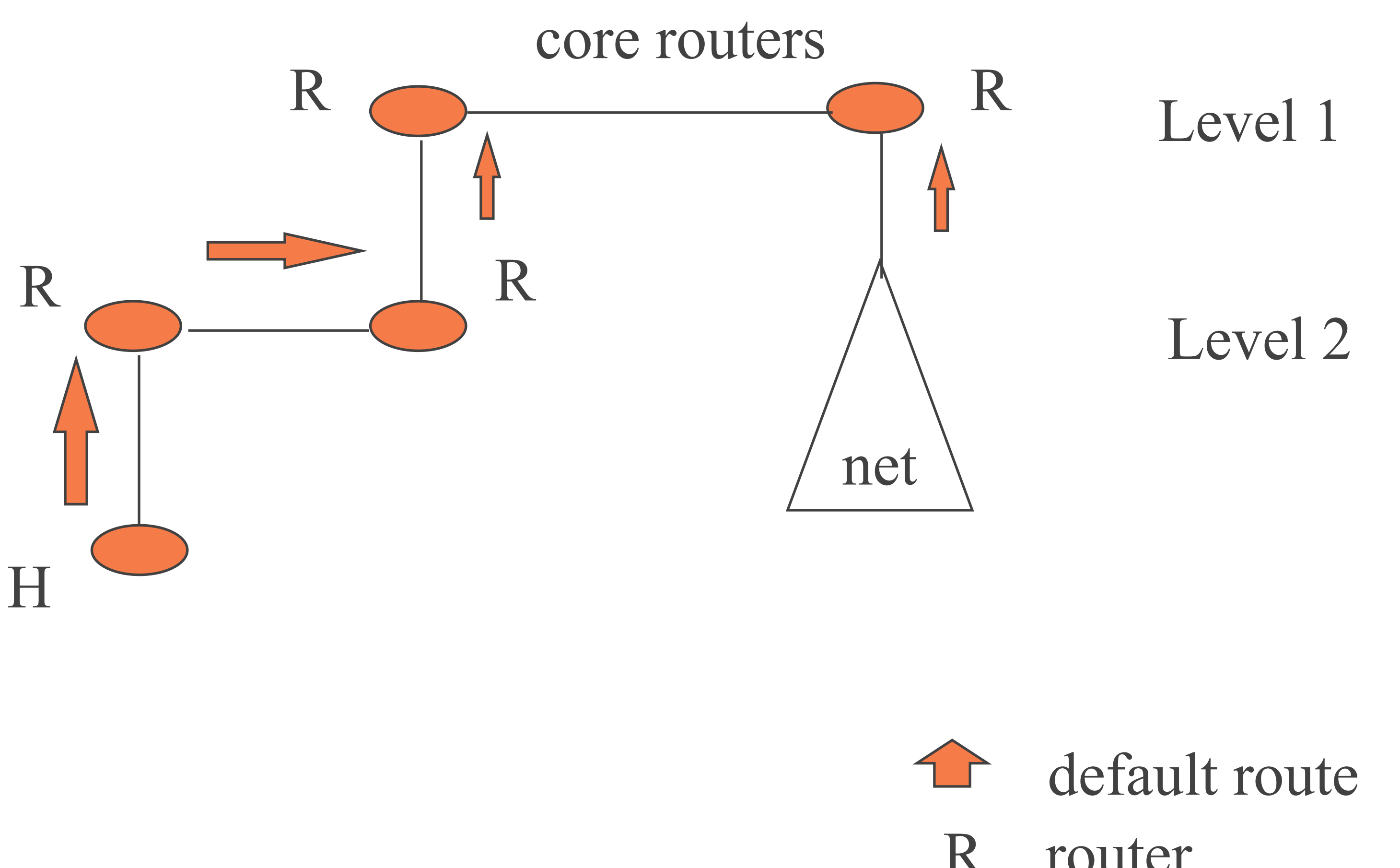
- Tanenbaum mentions logical absurdities
- **no router** - every host wired to every other host (mesh)

  N * N wires, go ahead add a host...
- **1 router -** for all hosts (star)
  - 1 heck of a routing table

one solution

- typically a flattened tree to give hierarchy
- at the top a small circle of core routers that know all the routes
- idea: **default route**
  - if you are not in the center AND you don’t know what to do with it, send it “UP” to smarter entity

routing hierarchy



connectionless/connected

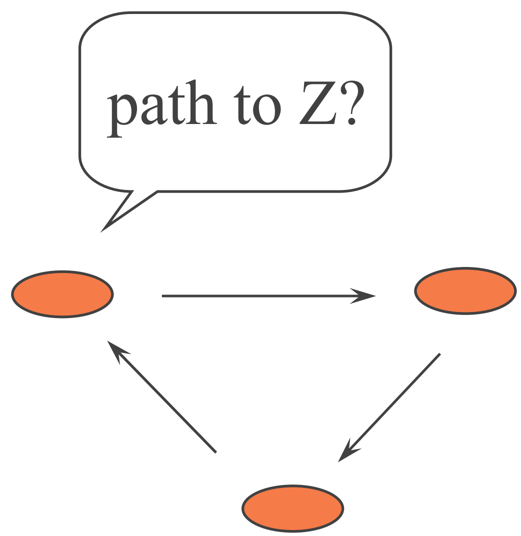
- network-layer in TCP/IP connectionless
- OSI stack has connected + connectionless
- connectionless - each packet routed via route table, paths may change dynamically
- connected - route setup at “connect” time in circuit switch, torn down at disconnect
- routing problems still similar

ideal routing algorithm

- fair or unfair (what if we want isochronous data ?)
- correct - what if algorithms too complex?
- robust - can deal with router reboot?
- stable - do routing changes stabilize in distributed system?
- efficient - all routing and no data not good
- topologically flexible
- maintainable - admin not too complicated
- scalable to many routers, many hosts? (distributed)
- deadlock? - loops?
- secure (+ clean, cheerful, takes out the trash, etc.)

classic problem

- **routing loop**



types of routing algorithms

- static versus dynamic
- static pros:
  - simple, may be easiest thing to do in simple topology, especially for leaf host with 1 router only
  - you may be smarter than the routers (want a path they won't give you)
- static cons:
  - can't react dynamically to crashed router (still need two routers to react though...)
  - not scalable

routing algorithm types

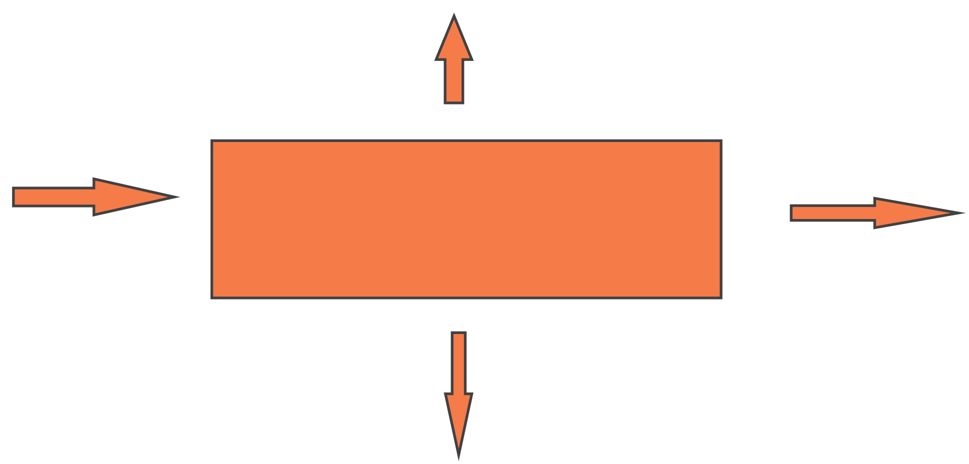
- centralized versus distributed versus end-node
  - center is necessary, can’t have defaults everywhere
  - if end nodes have route info, and big net, then route tables huge - need to limit size

types/tools - source routing

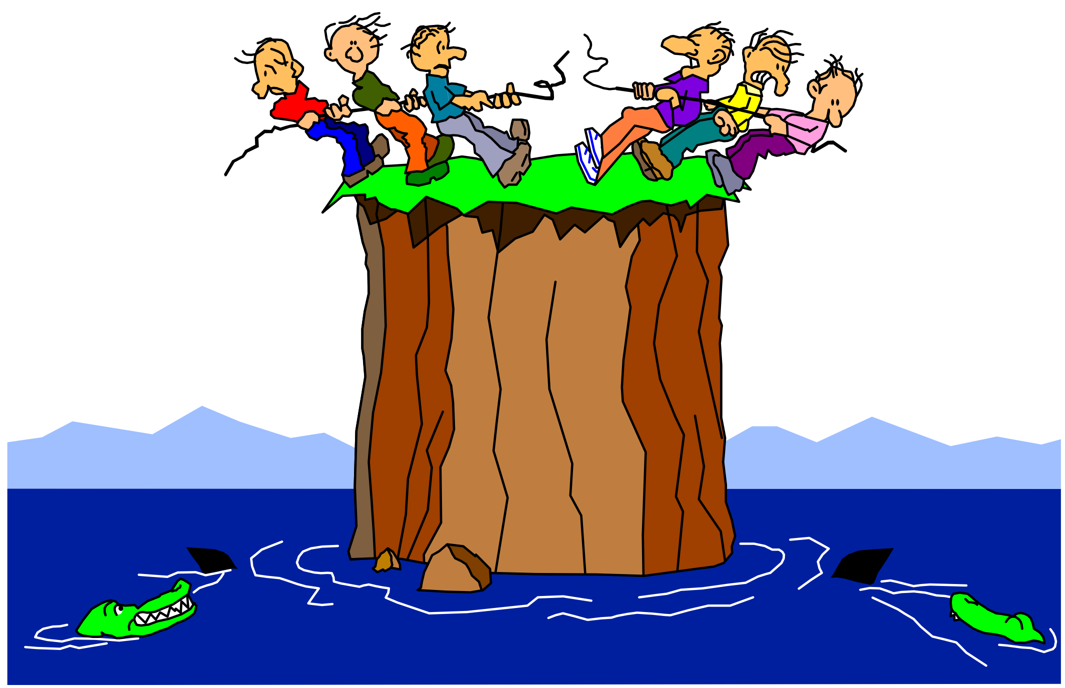
- source-routing vs hop-by-hop
  - end node has exact PATH and datagram follows that path: (first to Joe, then to Bob, then to Grandma's)
  - IP option, but rarely used
  - challenge to security: what if hostile entity convinces you to route all packets through it?
    - » maybe it can masquerade as you after that?
  - still a possible tool - used in BGP, can communicate POLICIES, from Novell to Intel, please Skip Bellevue
  - as scalable as hop by hop?

types/tools - flooding

- flooding - assume N interfaces
  - packet comes in N(1)
  - packet goes out N(2)..N(N)



be careful with flooding...



types - flooding

- important routing algorithm “tool” - used in many routing algorithms in some sense
- strong pro and con
- **pro - perfect routing**, you follow the best path
- **con - “perfect congestion”** - you use up too much bandwidth

types: vector-distance

- vector-distance algorithms:
  “tell the neighbors about the world”
- **vector** is destination (net/host)
- **distance** is metric (hopcount)
- if we called it destination-metric, other people would understand
- you flood your destination, hopcount info to your directly connected neighbor routers
- RIP is an example

types: link-state

- link-state or shortest path first (SPF) "tell the world about your neighbors"
- find out who is up locally, and flood that information to the entire set of routers
- they can use the "link-state" to build a shortest path map to everybody
- LS is compute-intensive. VD is bandwidth intensive.

issues - scalability?

- 10’s of hosts and a few routers - static routing
- what about putting all of those toasters on the Internet?
- one home/office, to business/enterprise, to state, nation, planet, solar system, galaxy...
- components affected include the network address and the router hierarchy

scalability...

- ip's current problems
  - net/hosts via class or even subnet don't match number of hosts really utilized
  - too many routes in core tables
  - ip address allocation from class C slize of pie means in effect majority of numbers are wasted
- scalability affects addresses and how they are sliced up; also router hierarchy
- how much low-level info (e.g., link-level details) can we afford to disseminate upwards?

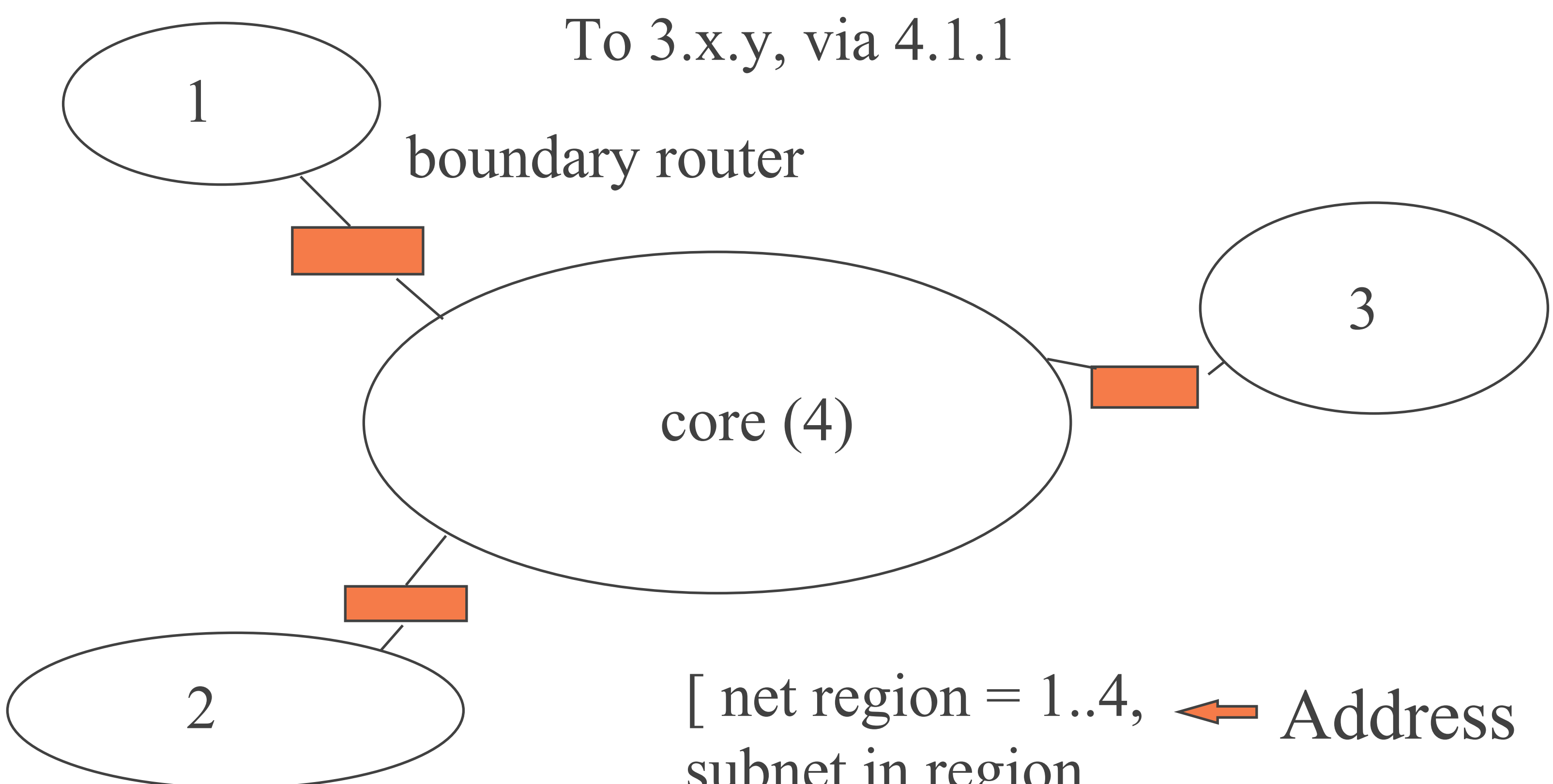
general solution to problem exists

- **when in doubt, add a new prefix** to address and a new SMALLER center to the world
- prefix must summarize internal structure
- divide world into
  center: (layer 1) boundary routers
  domains: (layer 2) inside routers
- boundary routers have summary routes in them, not all.

add a prefix...

- domain routers as 1st assumption - assume they know all the routes.
- if that isn't scalable, then add new hierarchy and introduce new layer of structure
- IP is moving towards this (CIDR)
- phone companies have prefixes (not enough... nobody is perfect)

hierarchy picture



[net region = 1..4,
subnet in region,
host] ← Address

issues - congestion

- does not refer to bronchial condition
- connectionless routers have only so many buffers, too many packets, they drop them
- things get worse at the “freeway exchanges”
- does routing protocol add congestion burden?
- how do we prevent/detect congestion?
- obviousally circuit-switches don’t have this problem **once circuit is set, but they waste bandwidth**

congestion?

- prevent congestion:
  - add carrying capacity
  - shutup, especially if high-volume src
- how do we notify network about it?
  - TCP detects congestion when sender notes that ACKS are missing, slow, or duplicated, sender slows rate of sending
  - some schemes have routers forward or pass back congestion bits

congestion

- IP sends back ICMP source quench message to sender
  - pro: you sent it the right way
  - con: you poured gas on the fire
- ISO CNLP sets flag in network header
  - pro: doesn’t add data to net
  - con: congestion notification is sent to the DESTINATION (oh, goody)

congestion cont.

- TCP solution is not bad BUT
  - what about protocols that use Internet that don’t implement or can’t sensibly implement it?
    - » NFS or Novell could but don’t, drive TCP out
    - » audio/video transmission is steady-state data flow
- congestion detection is an open question

issues - link costs

- we need a **metric**, which one?
  - cost? not appropriate within enterprise but between; e.g., which long-distance company?
  - hop count - how many routers do we traverse
  - available bandwidth - go least congested route
  - speed of underlying network, use ATM as opposed to 1200 baud modem?
  - time: shortest path in terms of time

issues - link costs

- if link costs change, that information must converge of course
- answer now is that link cost is usually hop count only (1 metric)
- question: would more complex algorithms (if possible) that dynamically account for link costs do qualitative better job than current simple algorithms or just use bandwidth?

type of service

- my packets before your packets!
- might want to prioritize certain traffic classes
- might want to optimize on multiple metrics
- policy-based routing (pb constrains) - outlaw certain links or routers
  - source and static routing can be useful here

issues - some misc. ones

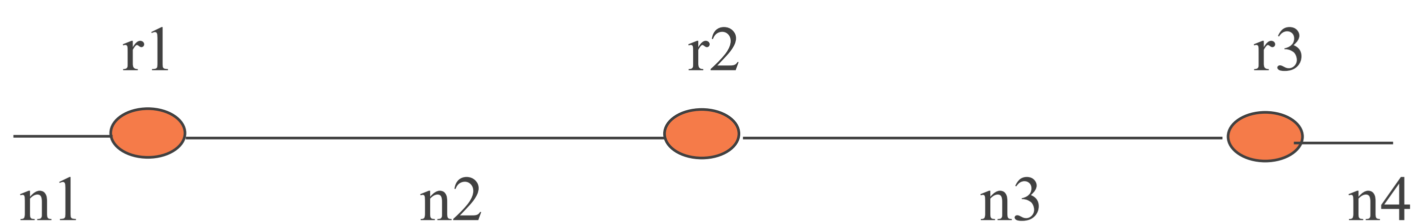
- load-splitting - if we have two same-level routers, can we split the load between them
- address matching - router needs to do this fast or may run out of buffer space
- migrating routing algorithms - you have RIP and now you want to switch to OSPF
  - can you run both?, switching one by one is disruptive
- partition repair - if two paths to one net, and one goes down, can routers fix it?

vector-distance algorithm

- examples: RIP, BGP
- algorithmic elements:
  - send: every N seconds out all connected interfaces broadcast 2-tuples :
    (to network X, hop count Y) ...
  - recv: if new tuple, add to routing table
    if better tuple, change existing
    if "dead" tuple, remove
  - timeout: if no refresh, timeout entry in N * Y seconds
    - » broadcast may be lost, therefore timeout is slower

vector-distance

- assume 3 routers, and that directly connected nets are in routing tables to start with. How does following converge?

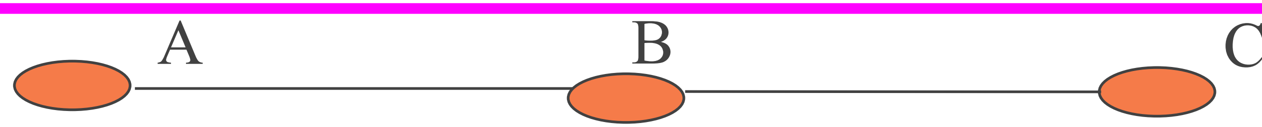


r1 table: (n1, 1)
(n2, 1)

slow convergence/count to infinity

- vector-distance like this has defects
- changes can be sent when they occur, but must recompute a bit so convergence takes time (made worse by possible loops)
- count to infinity problem can occur too - routing loop until hopcount reaches impossible value

count to infinity



C crashes, B knows C crashed but hasn't told A,
but unfortunately A talks to B first

B is told by A: I can get to C in two hops (and note
it doesn't mention to B that the path is thru B)

B says AHA!, that means I can get to C in three hops
and reports that to A

A says AHA!, it's now four hops to B and tells B

etc...

RIP max hop count (infinity) is 16

split-horizon fixup (vector-distance)

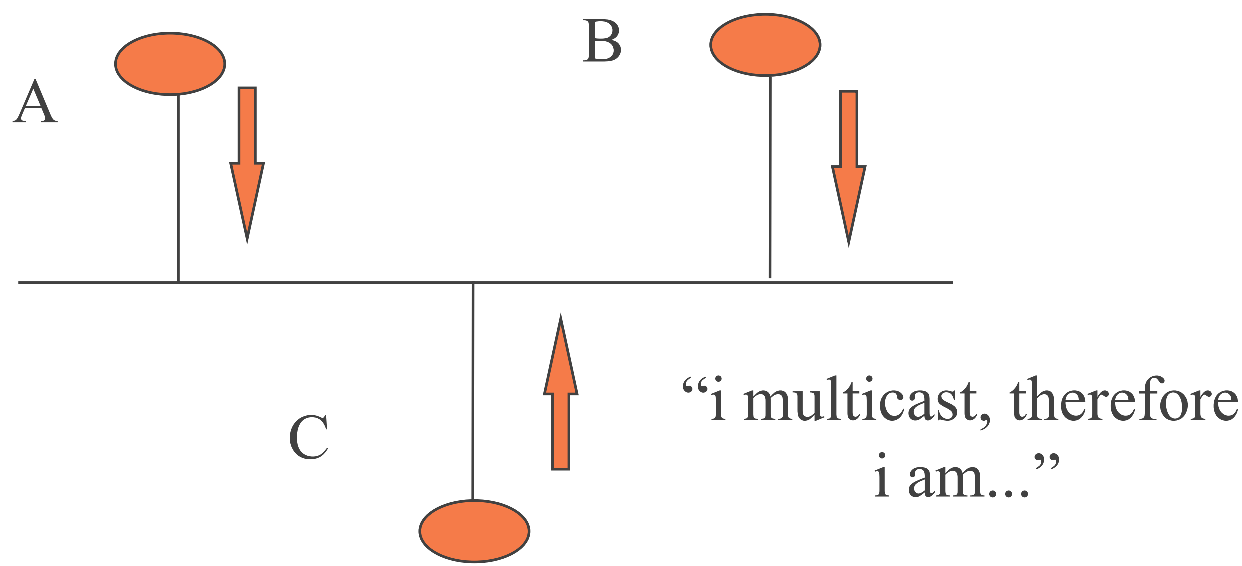
- A tells B that its distance to C is infinity
  - (because B is the direction A gets the info from)
- when link goes away, B will know that there is no path to C, and tell A
- doesn’t work in all cases

link-state algorithm

- “tell the world about your neighbors”
- link-state requires each participating router to keep map of complete topology
- in 3 parts
  - 1. determine neighbor connectivity
  - 2. send (“flood”) link-state packet that states which link neighbors are up
  - 3. use Dijkstra shortest-path first to compute best path to that network

link-state#determine link-state

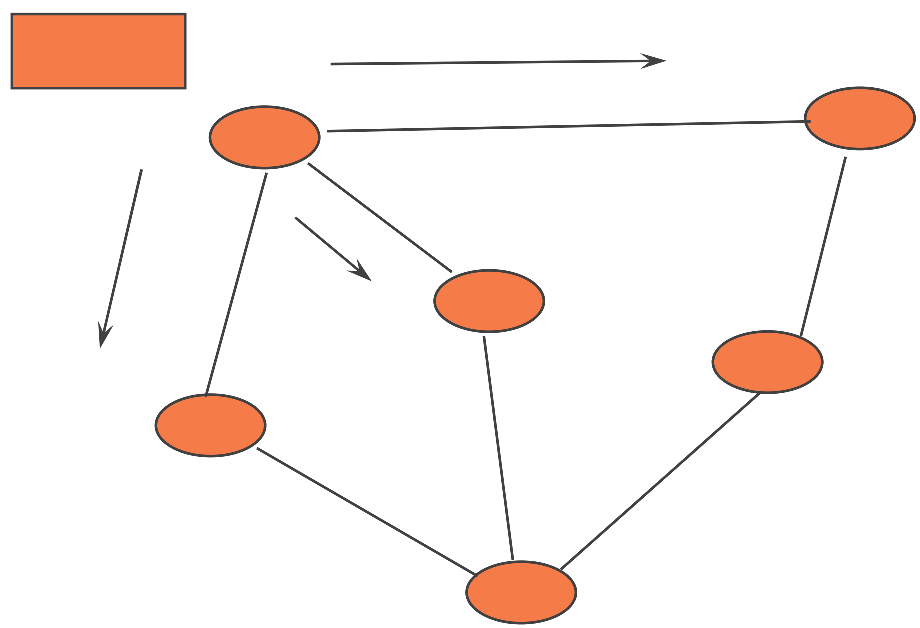
- "ping" neighbors to determine if they are up or they may broadcast (multicast) their existance



link-state#send LSP

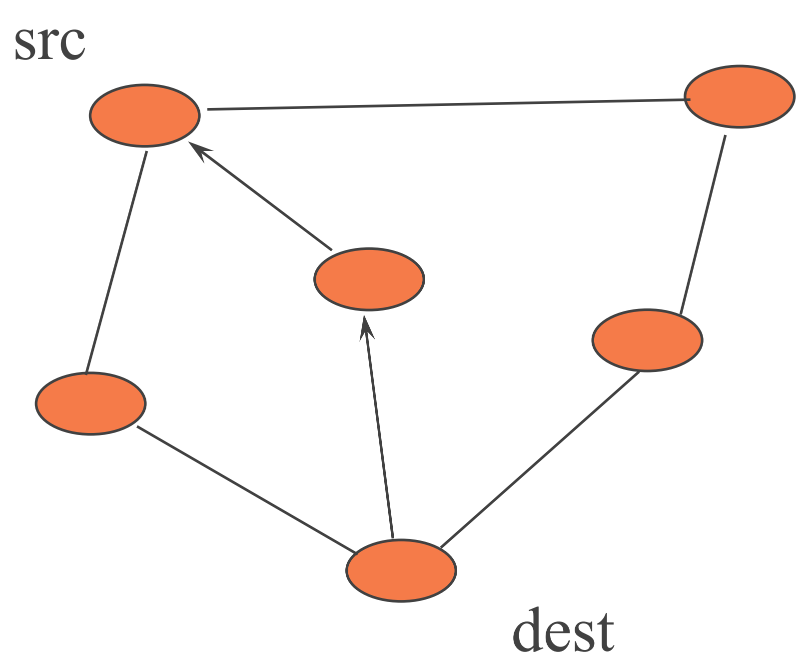
- each participating router “floods” (very carefully) routing domain with LSP

lsp



link-state#compute shortest path

- each participating router takes LSPs, stores them, and computes shortest path to sender



link-state: pros/cons

- pros
  - converges faster, no count to infinity problem + router can forward LSP immediately, must recompute DV
  - more functionality; e.g., **each router has map of net**, can make network debugging easier
- cons
  - more compute than vd (does this matter?)
- tossups
  - bandwidth? vd broadcasts summary version of route table, ls routers send LSP around net