# An Overview of Routing Theory

IP Routing

Jim Binkley
Portland State University

# Routing Theory

- topologies and scalability
- basic tools & ideas & attributes
  - » static vs dynamic, flooding, tunnels, control theory
- some issues; e.g., congestion
- algorithms: vector-distance vs link-state

# fundamental ideas

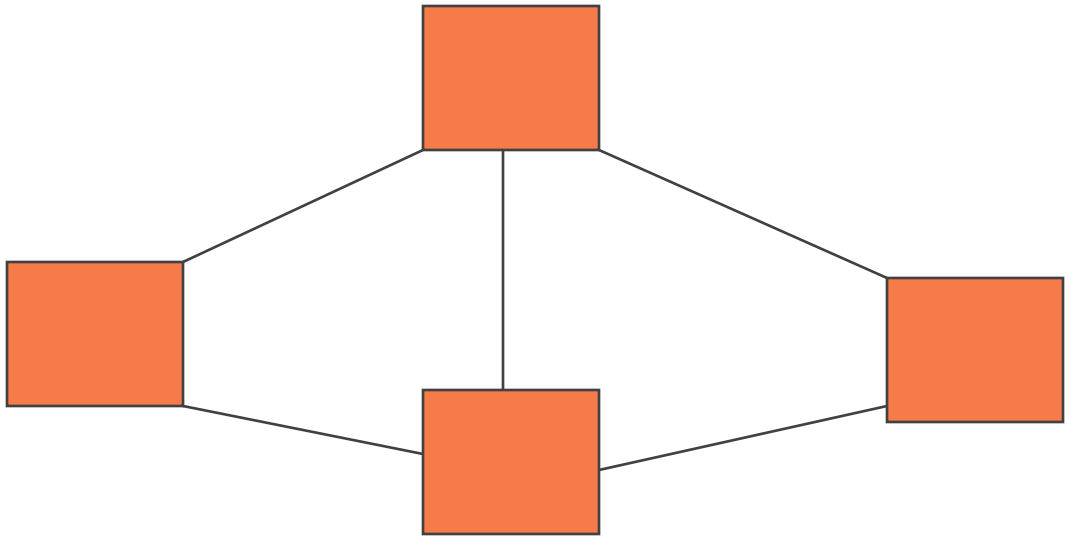
- **routing** - finding a path from one end to the other for a packet
- we need one or more **algorithms** that are most likely **distributed** amongst a set of hosts and router
- what are the properties of said algorithm?
- what issues affect it?

# 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)

# a routing domain

a **routing domain** == set of routers under same admin running same routing protocol



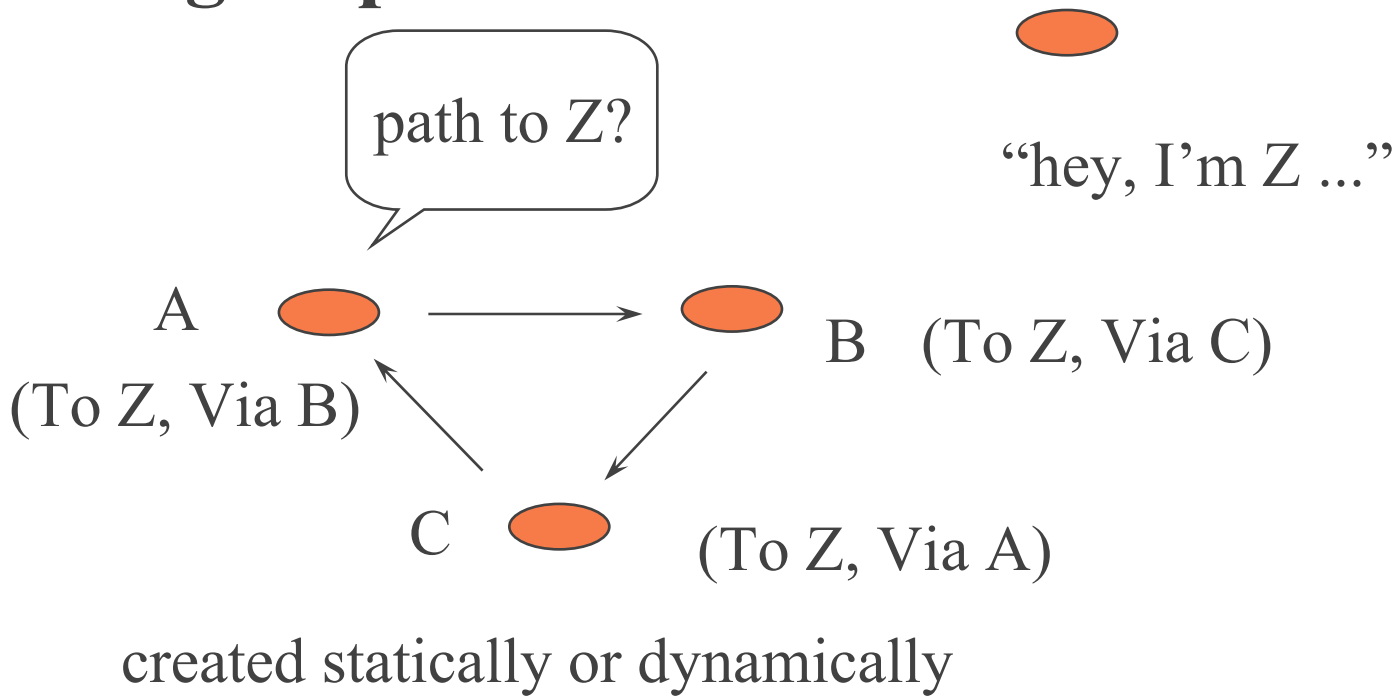
e.g., all these routers are controlled by Joe Bob Inc, run OSPF

# ideal routing algorithm

- correct - what if algorithms too complex?
- robust - can deal with router reboot?
  - two problems: loss of router and loss of link (i/f failure ...)
- stable - do routing changes stabilize in distributed system?
  - **convergence** - a state, all routers have "same" routing table
- efficient - all routing and no data not good
- topologically flexible, can allow **aggregation**
- maintainable - admin not too complicated
- **scalable** to many routers, many hosts? (distributed)
- deadlock? - routing **loop-free**?
- secure - can intruders inject routes?

# classic problem

- **routing loop**




created statically or dynamically

# topology

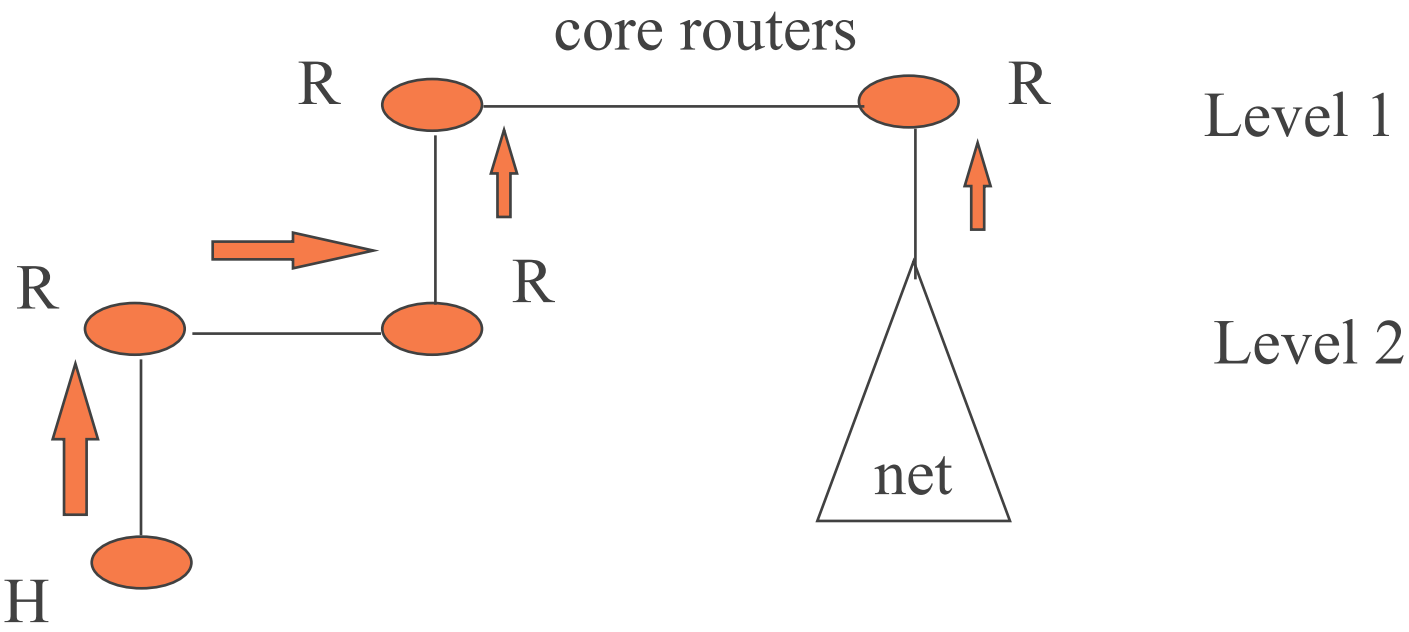
- Tanenbaum mentions logical absurdities
- **no router -** every host wired to every other host ( mesh )

  N * N wires, go ahead add a host...
- **1 router -** for all hosts ( star )
  - 1 heck of a routing table
  - this is why core can have problems
  - single point of failure?!
  - everything should be fully connected?!

# 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




this could be the model for an intranet OR
the Internet as a whole (multiple core rings)

# levels

- the top level has to know everything
  - in a routing domain, routers know all the routers at level 1
  - in BGP, top of Inet, routers may have full routing table
- routers may also choose to hide interior structure (subnet/CIDR) in order to **aggregate** routing info
  - reduce routing table size elsewhere

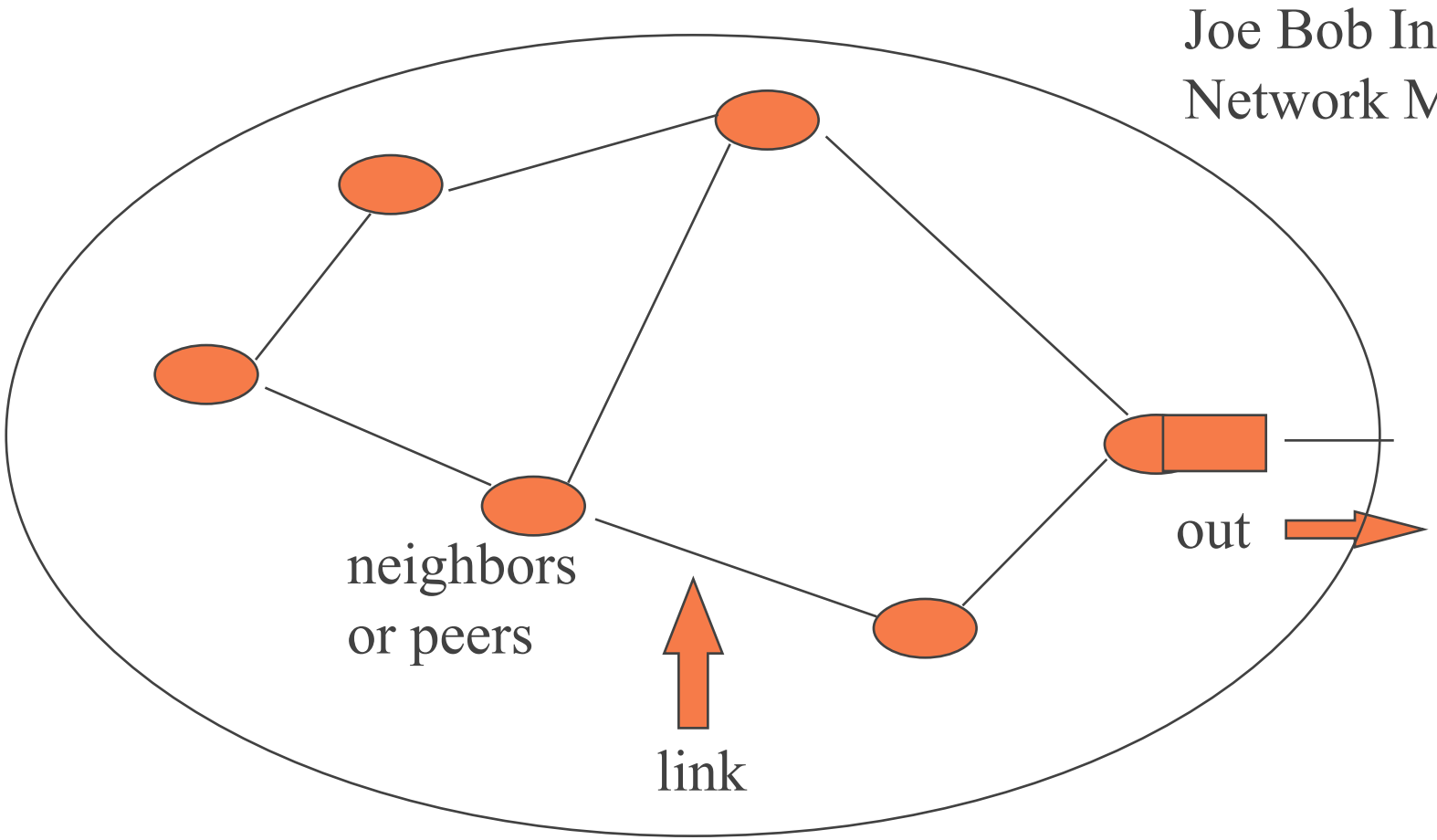
# divide routing world into 3 parts

| topology | IETF | ISO/OSI |
|---|---|---|
| same “link” or wire | none, intra-link? | none, intra-link? |
| enterprise or campus | Interior Gateway Protocol - IGP | intra-domain routing protocol |
| between enterprises | Exterior Gateway Protocol - EGP | inter-domain |

# protocols acc. to topology

| topology | IETF | ISO/OSI |
|---|---|---|
| intra-link | ARP | ES-IS |
| intra-domain | RIP, RIP(2), OSPF | IS-IS |
| inter-domain | EGP, BGP(4) | IDPR |

# the Interior - RIP or OSPF




Jim Binkley

the "world" (from router POV)

# scalability is a LARGE concern

- 10's of hosts and a few routers - static routing
- what about putting all of those toasters on the Internet? (what if the net/hosts bigger?)
- 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 (again true post CIDR)
  - ip address allocation from class C slice 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?

# Some current BGP issues

- # entries > 100000, growing faster than a linear rate?
  - CIDR help in mid-90s has worn out?
- # entries that are /24 or bigger?
  - I.e., not enough aggregation
  - entries may need dampening
- aggravated by mesh redundancy
  - if multi-homed more paths announced to world

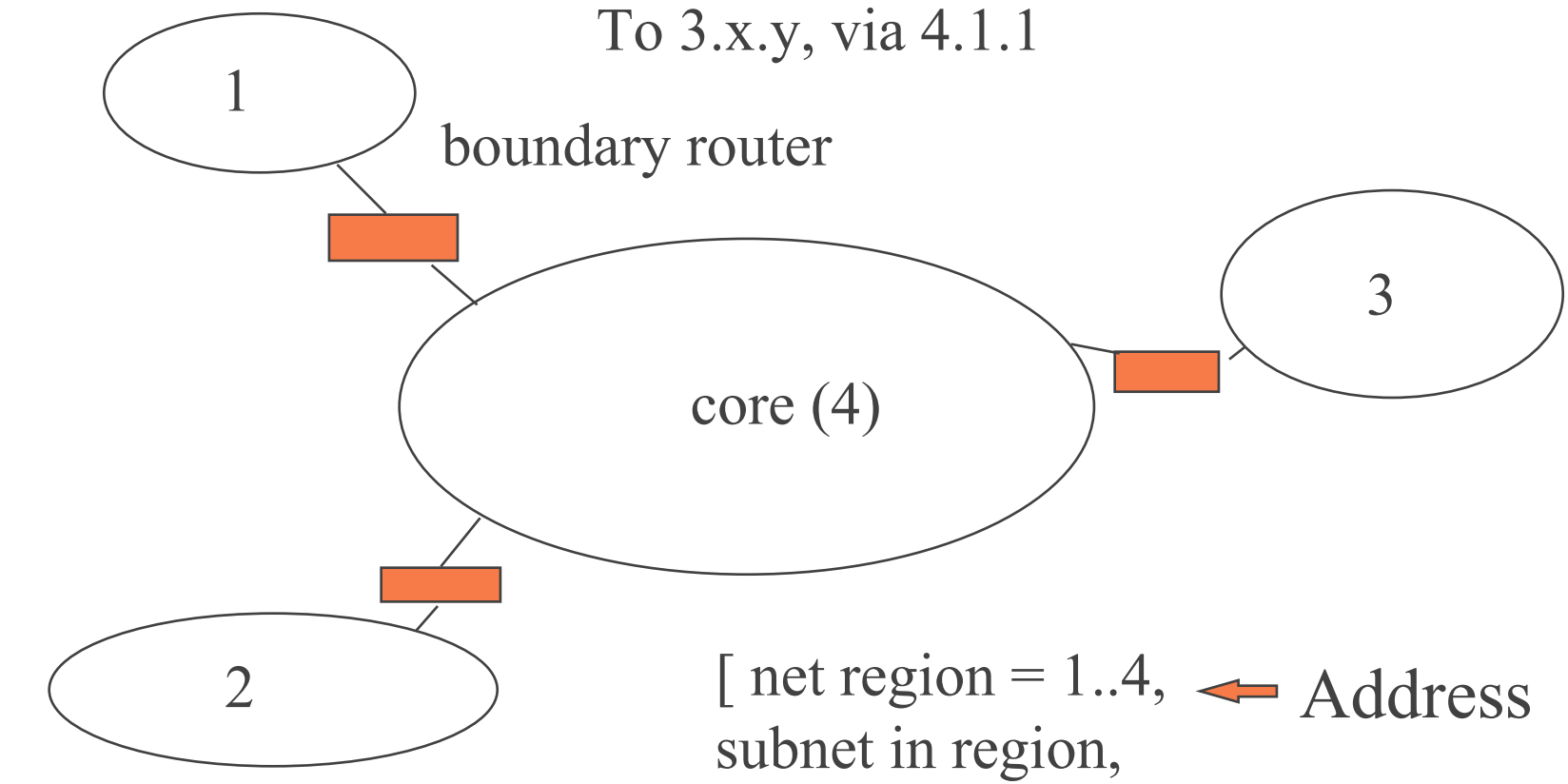
# N**2 is not our friend

- Has a tendency to show up in routing protocols/theory because naturally:
- routers (nodes) times links (connections)
- remember Andrew T. and what happens if all nodes directly connect!
- The problem here is memory storage in individual core routers
  - how much memory if a core router must remember: 1. All link metrics, 2. All host attributes, 3. Host X host attributes

# to increase scalability, 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, find points of aggregation
- IP is moving towards this (CIDR)
- phone companies have prefixes (not enough... nobody is perfect)
- NAT, MIP(tunnels), IPv4+AS, all ways of doing this (to say nothing of 128bit addresses)

# hierarchy picture



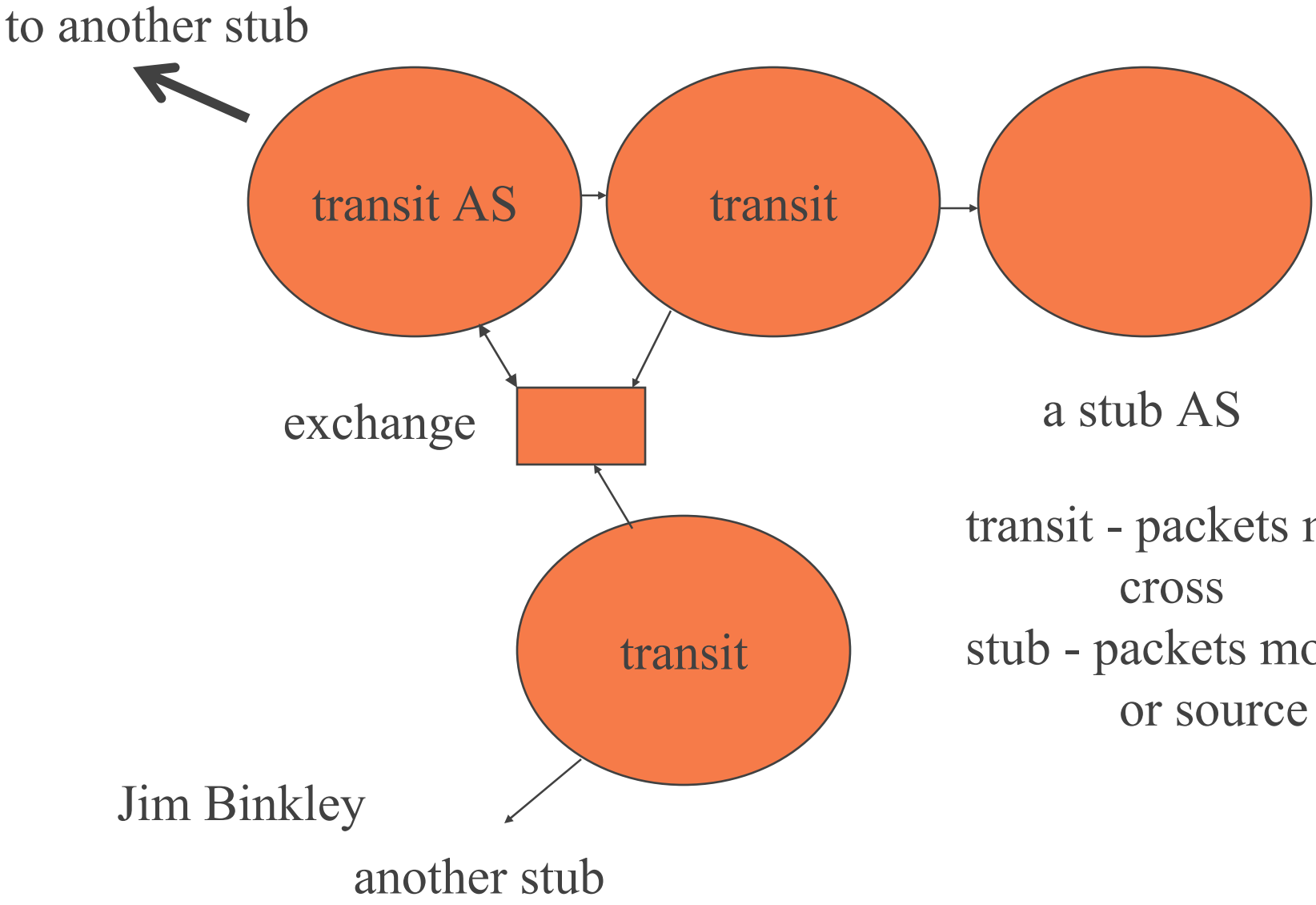
# general solution to problem exists

- **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.

# btw, this is related to a magical architectural design principle

- add a **layer of indirection** == POWER
- consider IPv6 address (ip part, IEEE mac part)
- Mobile-IP 2-part AWAY address (FA Care-Of-Address, Mobile-IP Home Address)
  - irony here is the power of this is not well understood
- BGP IPv4 “address” is (set of AS #s, IP dst)
- ahem: NAT since this may add to the IP address space pool (a few real ip addr, mapped to many private ip address), under severe constraints

# Inet has multiple centers




transit - packets mostly cross

stub - packets mostly sink or source (ES)

# 3 ways routes get into routing tables

- static - loaded by network admins
  - put into config to auto-load at boot
  - route add 10.0.0.0 -gw gateway-ip
- dynamic routing
  - done by routing daemons running a routing protocol
- kneejerk modification to host routes by ICMP redirects
  - typically way to populate routing table with default route

# static routing

- 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) - policy routing possible
- static cons:
  - can't react dynamically to crashed router (still need two routers to react though...), anti-redundancy
  - not scalable, not good if you have 100 routers to configure right now (or 1000 hosts)

# dynamic routing

- we need a distributed routing protocol
- goals:
  - intelligent (sic?) choice of route based on metric (single more likely than multiple)
  - automatic population of routing table to avoid manual setup
  - redundancy of routing path; i.e., convergence post path failure due to loss of router, interface, or sudden onset of backhoe

# theory: routing protocol types

- **centralized** versus **distributed** versus **end-node**
  - center is necessary, can't have defaults everywhere
    - » why not just compute and put in center?
  - if end nodes have route info, and big net,
    - » not scalable, then route tables huge - need to limit size
- some sort of distributed system is conventional answer

# types/tools - source routing

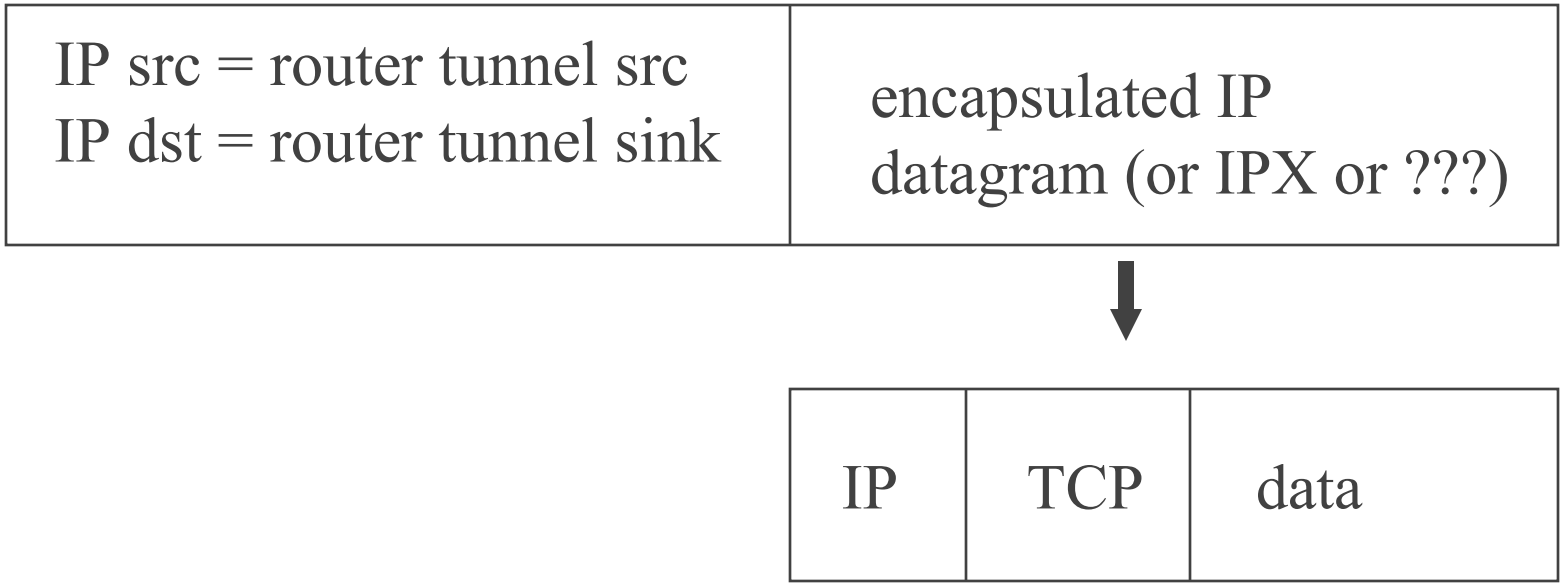
- source-routing vs hop-by-hop
  - end node has exact PATH and datagram follows that path: ( first to Joe, then to Bob, then to Grandma's )
  - IP option, but rarely used (strict/loose)
  - challenge to security: what if hostile entity convinces you to route all packets through it?
    - » maybe it can 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?

# routing tunnels are basic tool

- tunnels are example of source routing (doh!)
  - strict or loose?
- IP datagram with IP header (IPIP)
  - MBONE DVMRP, Mobile-IP (unicast)
- IPX, APPLETALK datagram inside IP header
  - Cisco GRE tunnels designed as general way to do tunnels (GENERIC ROUTING ENCAPSULATION)
  - IPv6, IPv4 IPv6, or IPv6 IPv4
- security tunnels, IP IPSEC IP, basis of VPNs

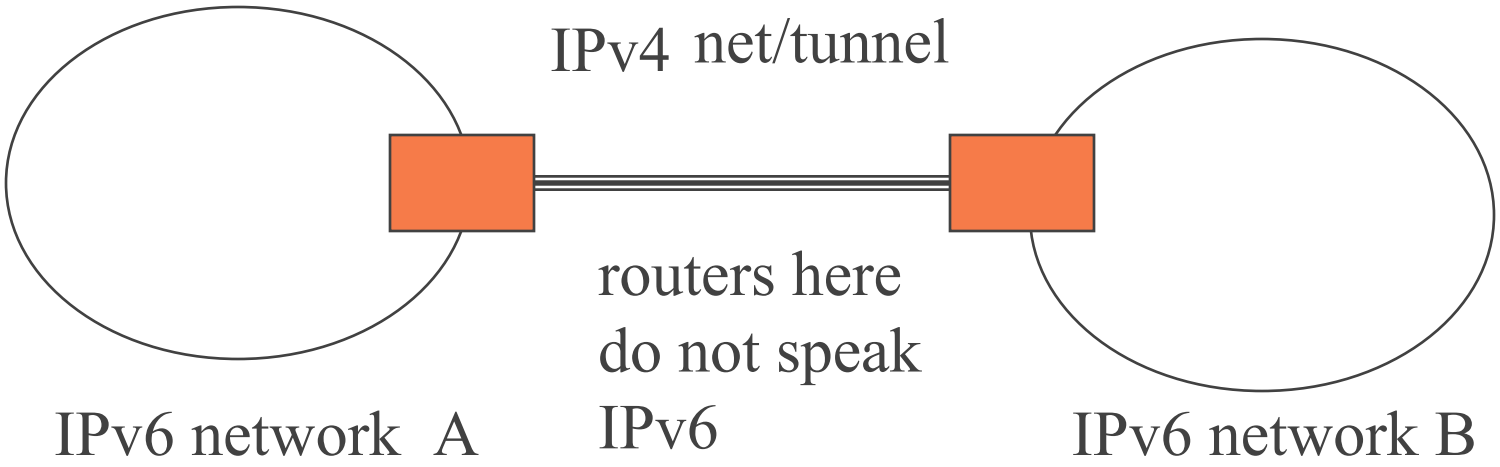
# encapsulation scheme

IPIP (btw: IP next proto == 4 for IPIP)



# IP tunneling mesh

IPv4 | IPv6 datagram virtual network

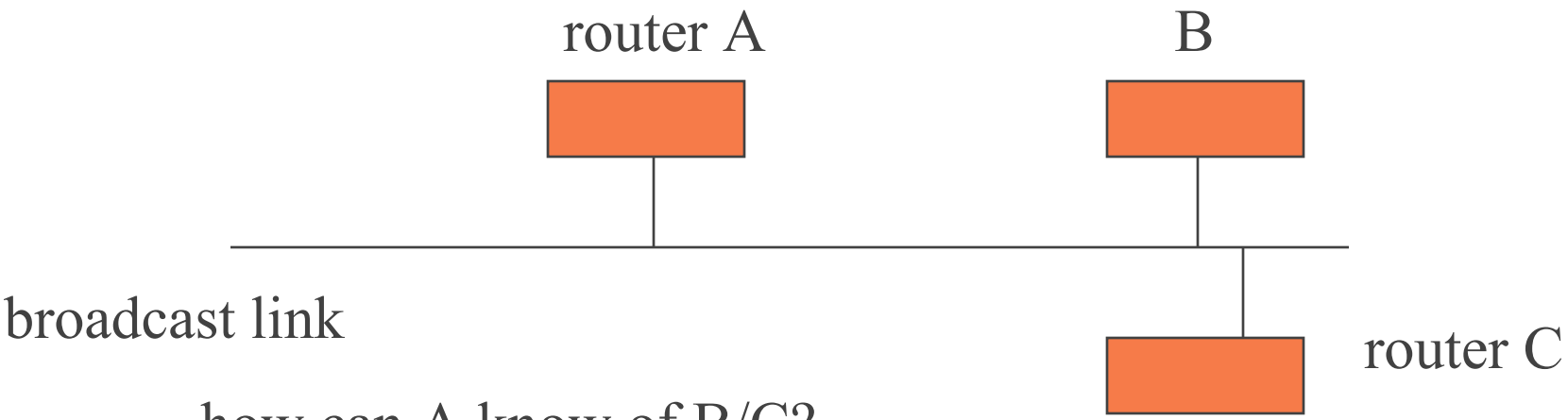



route to net B, forward to A border router, which
will encapsulate and send to B border router

# control-theory

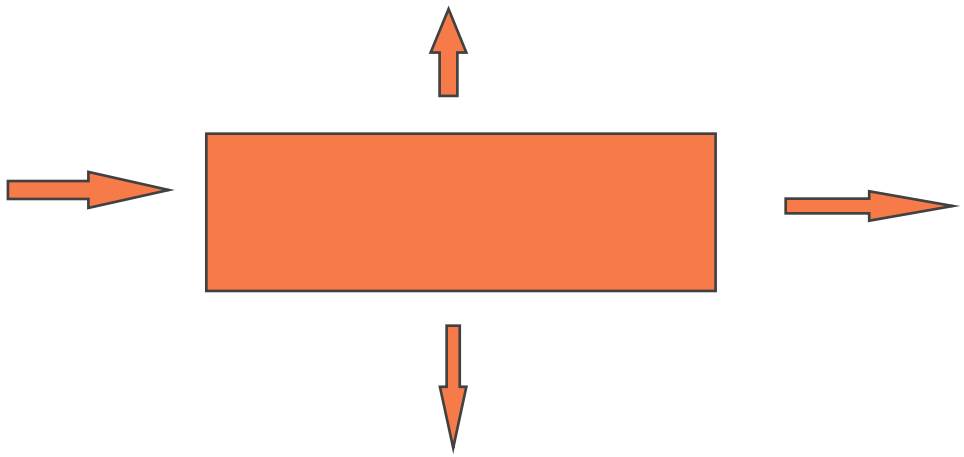
- routing is **control**, without it, data cannot flow
- chicken&egg problem: routing in terms of **addressability** must exist a priori before routing can occur
  - I can't send you packets if I can't find you ...
  - think about this in terms of every routing protocol
  - addressability is fundamentally link dependent
  - sometimes we rely on manual configuration (serial)
  - sometimes information automatically gathered (arp or multicast on ethernet)

# addressability problem



# types/tools - flooding

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



# be careful with flooding...

# flooding, cont.

- important routing algorithm “tool” - used in many routing algorithms in some sense
- strong pro and con/s
- **pro - perfect routing**, you follow the best path (redundancy here is a feature!)
- **con - “perfect congestion”** - you use up too much bandwidth
- **con** – you may loop packets

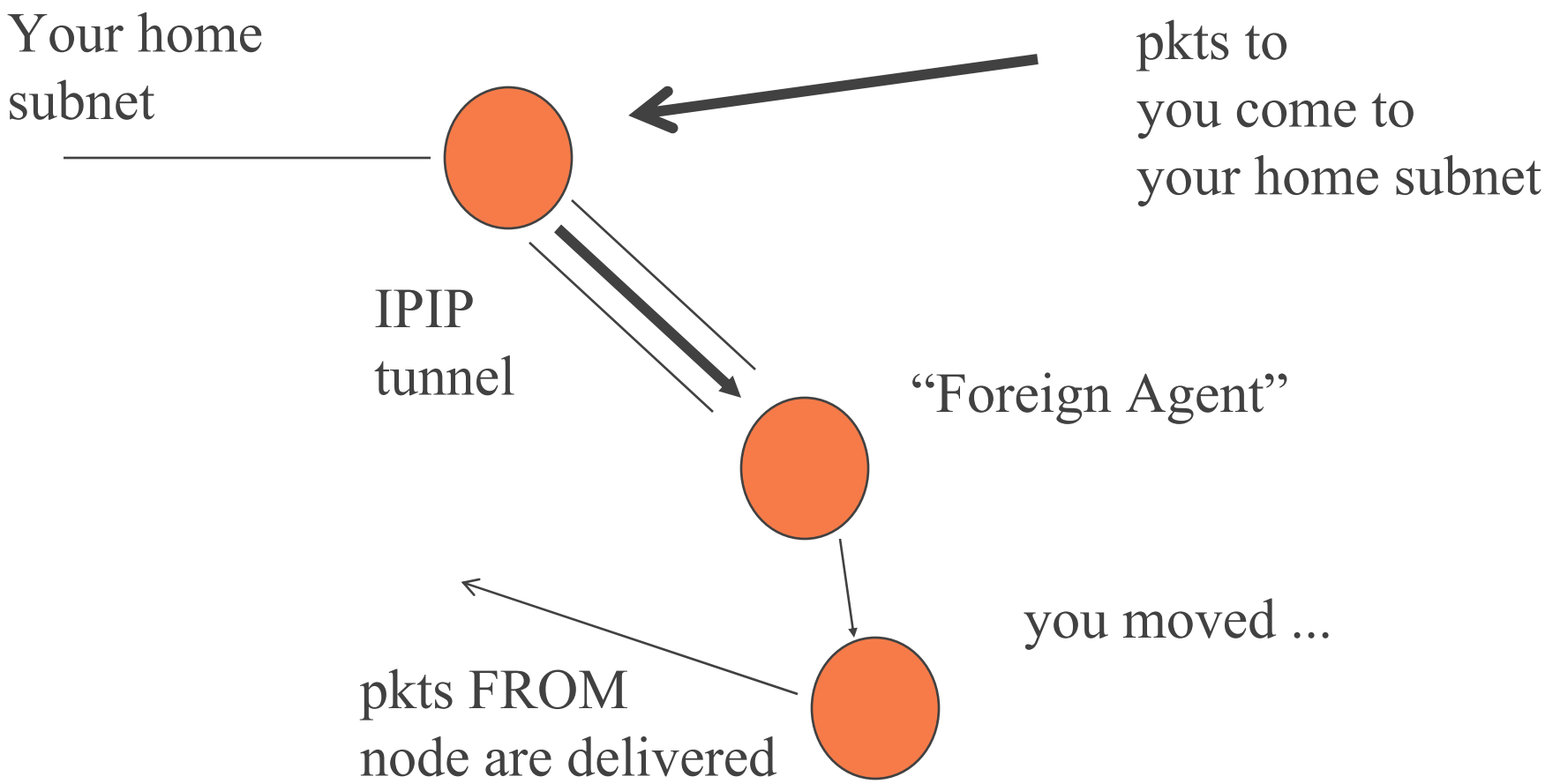
# flooding may be constrained of course

- you might simply use broadcast to send info
- you may pass judgement on the info before you send it out the “other” interfaces
  - not forward it if it is not new
  - or interesting
- many routing protocols rely on some sort of flooding (even BGP, although not link-layer)

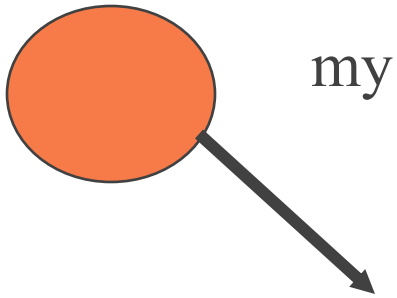
# routing is 2 1-way problems

- path between A and Z may not be the same
- asymmetric routing NOT unusual
- consider Mobile-IP
  - solves problem of packets TO remote host
    - » packets tunneled from HOME to AWAY
  - not packets FROM remote host
    - » ordinary “default” routing
- consider multi-homed stub with default route

# Mobile-IP as one-way IPIP tunnel

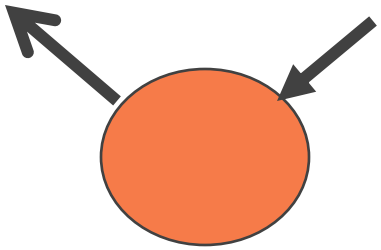


# asymmetric paths



my routing domain

the wilds of the Inet ....

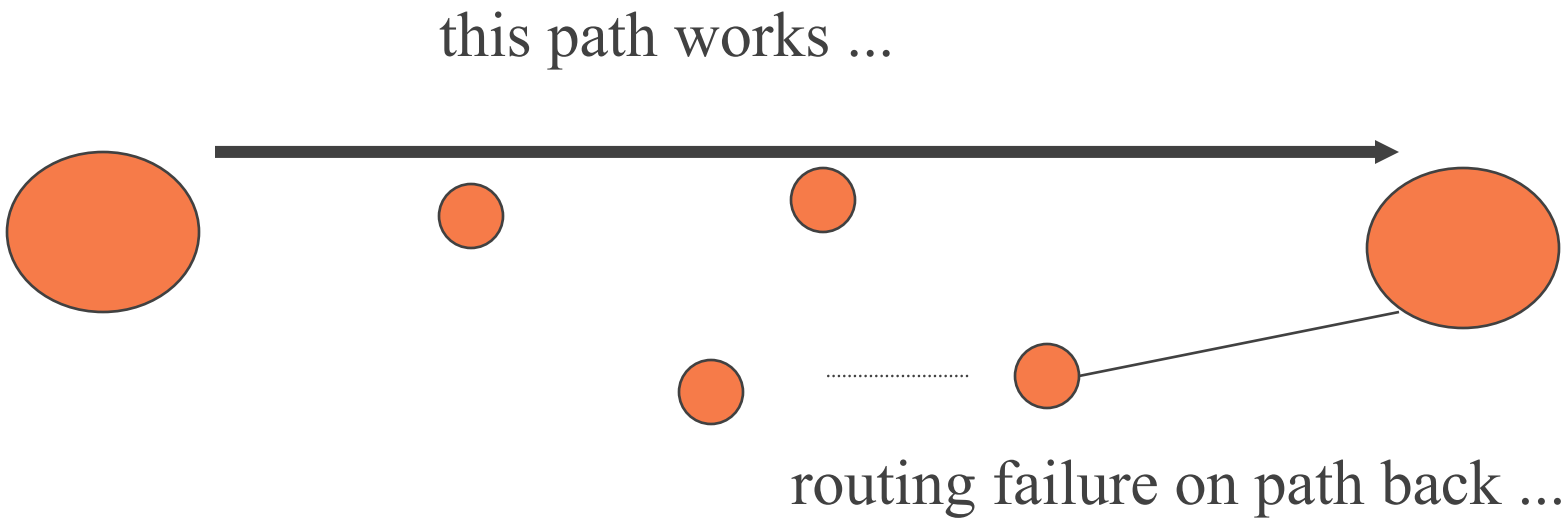


pkts in

default
route

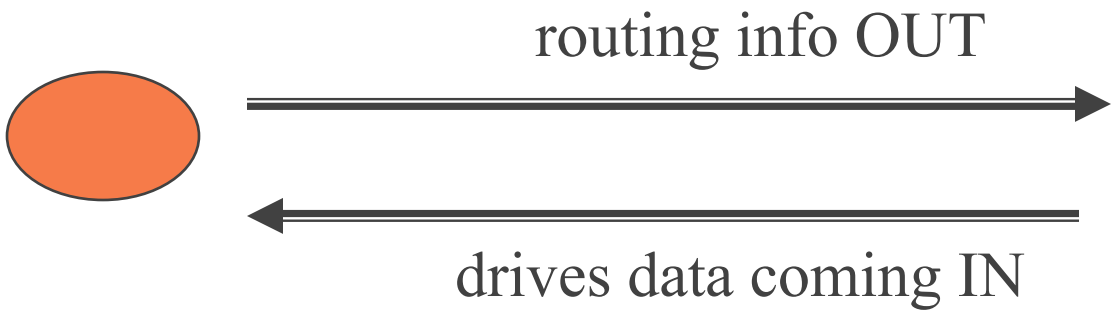
multi-homed stub
with default route

# you may have to debug twice



# control idea: routes vs data

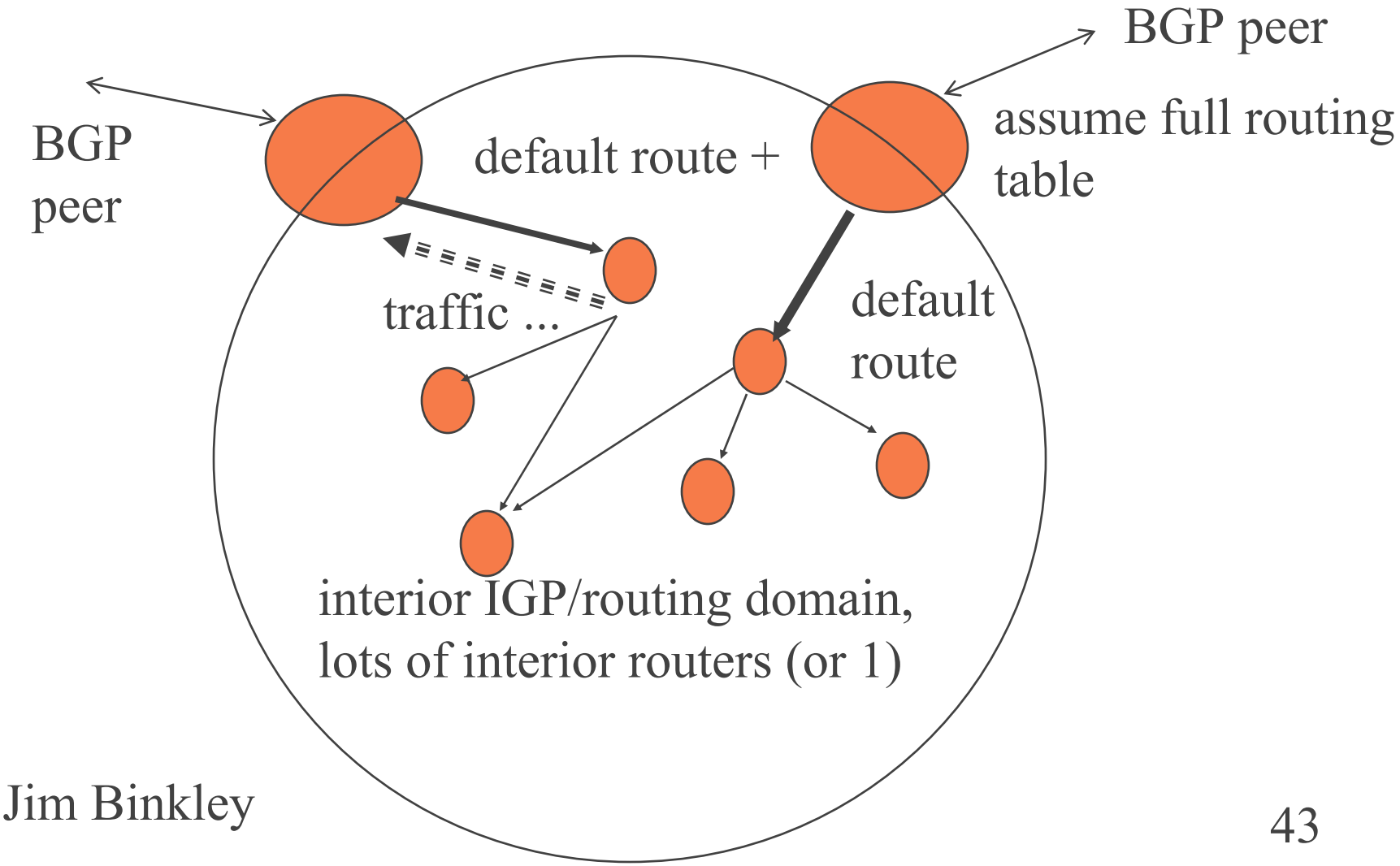
- on a given link, you might SEND routing info, which will cause
- data to COME BACK on that path




here I am, talk to me!!!

**data flows opposite of control**

# example: multi-homed stub, 2 default routes



# 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?
- obviously circuit-switches don’t have this problem **once circuit is set, but they waste bandwidth**

# congestion?

- ways to prevent congestion:
  - add carrying capacity - big pipe theory
  - 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, **this is END to END method**, routers not involved (David Clark/MIT, end to end paper)
  - some schemes have routers forward or pass back congestion bits

# congestion schemes - routers involved

- IP sends back ICMP source quench message to sender (deprecated)
  - 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)
    - » destination calls sender with POTS? "shutup"

# 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
- large issue in QOS terms

# issues - link costs

- we need a **metric**, which one?
  - **cost**? not appropriate within domain 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
    - » time is typically an app ...

# issues - link costs (metrics)

- if link costs change, that information must converge of course
- **we typically only use one metric within a domain**
- **we typically only use static metrics, not dynamic metrics (e.g., not congestion, but hop count)**
- we could have multiple metrics in use?!
  - say hop count, power use, radio strength for mobile wireless nodes?
- 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; .e.g., 1 control., 2. isochronous., 3 bursty
- **policy-based routing** (pb constraints) - decree a certain path, or outlaw a certain path
  - source and static routing can be useful here
  - BGP claims to do “policy” - recognize that many routing theory types think NOT ENUF!!!

# packet color overview

1. 3 mechanisms needed for packet classification
2. some way to setup a logical circuit
    1. end to end protocol or router-router protocol
        1. packets with classifier X should be treated like first-class passengers
    2. per packet classifier (flow ID, mac address)
    3. router/switch scheduling algorithm
        1. if packet classifier, then give it a buffer

# issues - some misc. ones

- **load-balancing**- if we have two equal routes to X, can we split the load between them?
  - **equal cost multi path**
- **migrating** routing algorithms - you have RIP and now you want to switch to OSPF (and no rebooting multiple times)
  - can you run both?, switching one by one is disruptive
- **route redistribution** - at routing domain boundary, how exactly do you take info from one routing protocol and inject into another? (Cisco route maps)
- **routing partition** repair - if two paths to one net, and one goes down, can routers fix it in face of **hierarchy**?

# a bit more on that

- **partition**
  - basically means 2 nets, 1 link, the link blew up (or came back), now what?
    - » is there a way to glue the two nets together thru some other path?
    - » is there a way to optimize traffic when the nets reconnect in order to minimize control traffic
- **aggregation** - in general, if routing is control, can we minimize it
  - by lumping addresses together ...
  - downside./s: too much routing, too many routes

# 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 (good point, Dr. Perlman)
- you flood your destination, hopcount info to your directly connected neighbor routers
- RIP is an example, but so is BGP, IGRP, EIGRP, and, remember the fuzzballs ... (time as metric)

# types: link-state

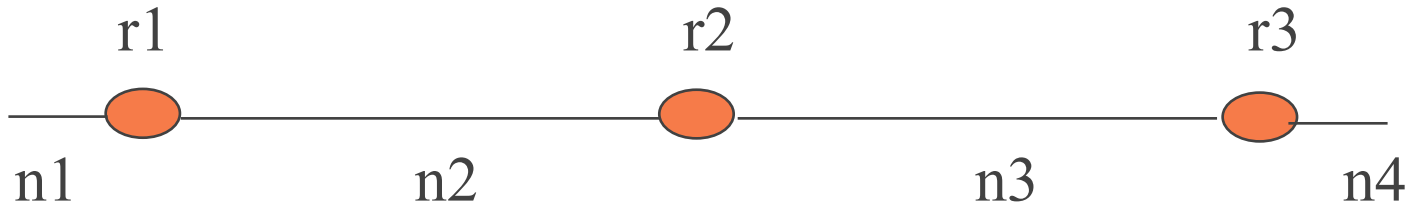
- link-state or shortest path first (SPF)
  **"tell the world about your neighbors"**
- find out who is up locally, and flood that information to the entire set of routers
- they can use the "link-state" to build a shortest path map to everybody
- LS is compute-intensive. VD is bandwidth intensive.

# vector-distance algorithm

- examples: RIP, BGP, IGRP, EIGRP
- 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?

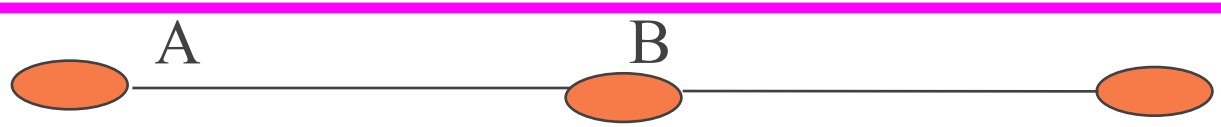



r1 table: (n1, 1)
(n2, 1)

# slow convergence/count to infinity

- vector-distance like this (RIP) 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)

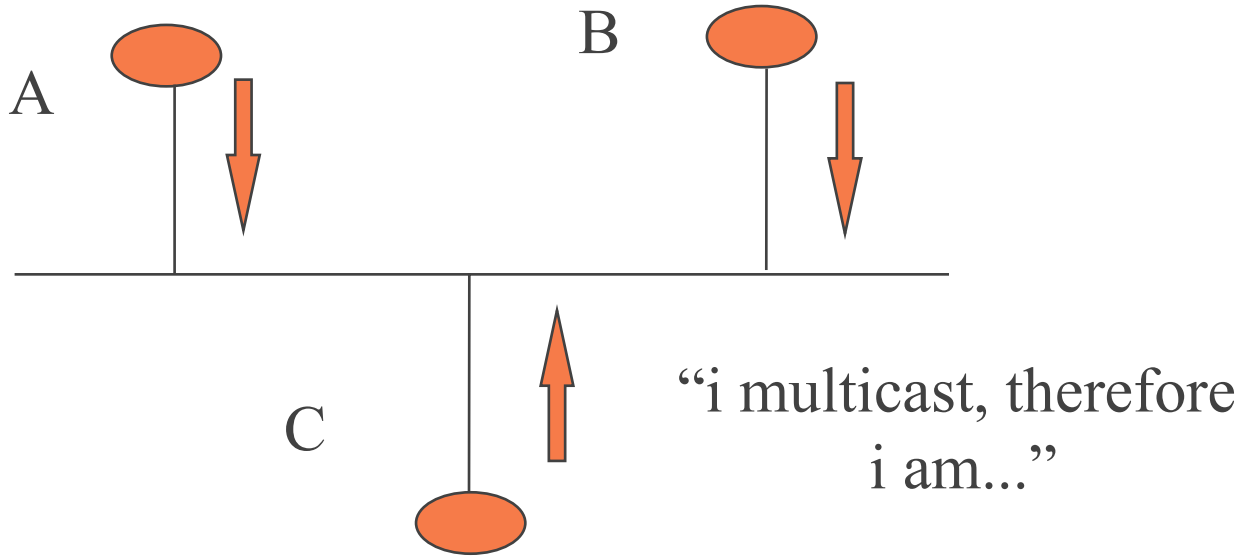
- A tells B that its distance to C is infinity
  - (because B is the direction A gets the info from)
- when link goes away, B will know that there is no path to C, and tell A
- doesn't work in all cases

# link-state algorithm

- "tell the world about your neighbors"
- examples: OSPF, IS-IS, NLSP, IDPR
- 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

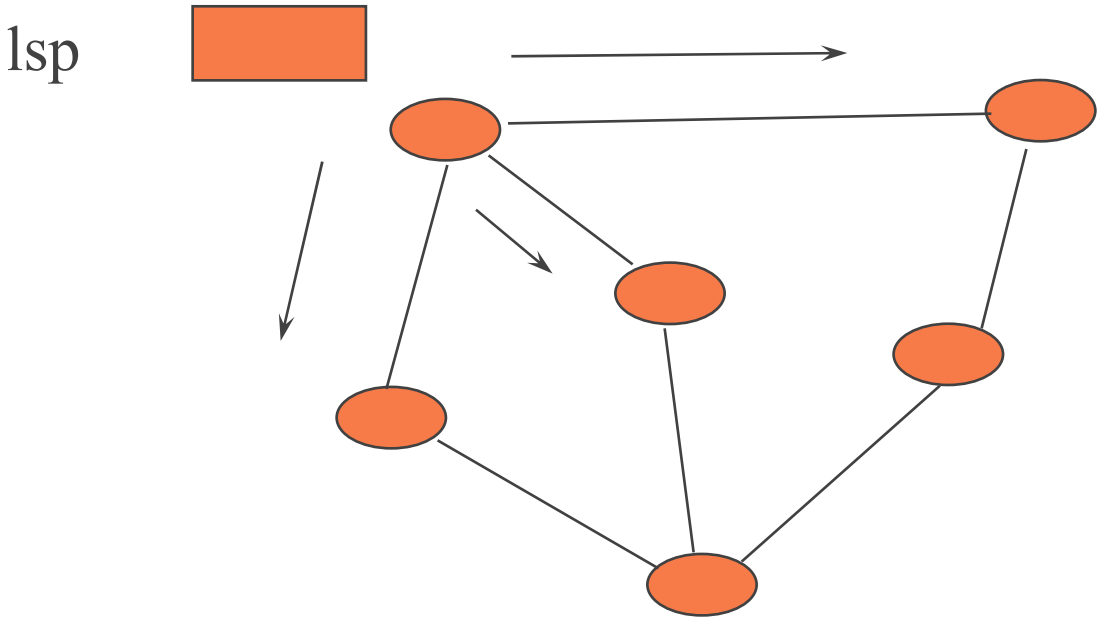
# link-state#determine link-state

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



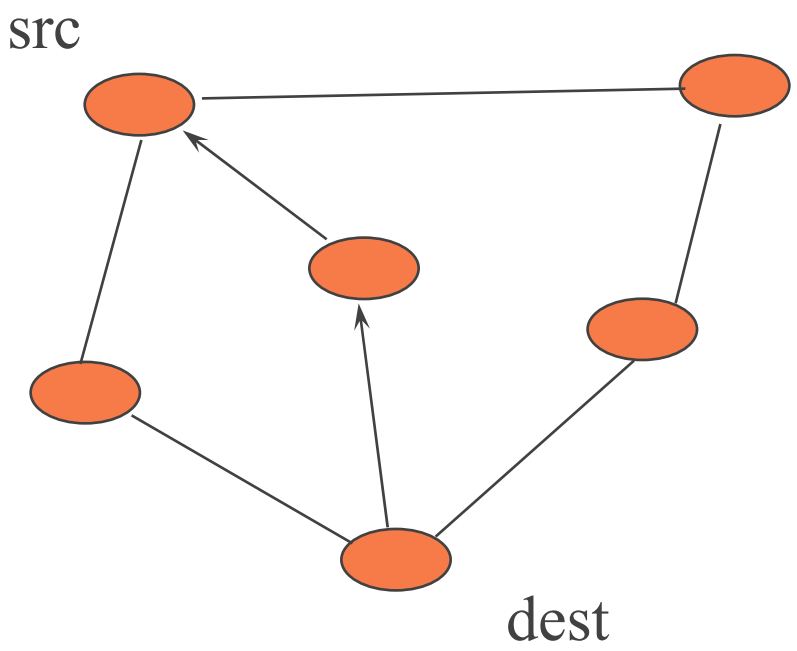
# link-state#send LSP

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



# 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

# basic tools for admins:

- **ping**, basic reachability, pkt loss, latency
- **traceroute**, latency and hop count
  - remember paths may be asymmetric
  - traceroute "-g" may/may not work
- **ttcp**, memory to memory xfer speed
  - Cisco is building it into IOS now
  - can measure tcp-based thruput, also UDP packet fling (one way)..
- sniffers, free tcpdump (www.tcpdump.org).

# higher-level wonders

- very well may involve SNMP
  - **MRTG** and its descendants
    - » Cricket, and other front-ends
  - graphs SNMP integers on strip-charts
    - » port byte counts in/out
- Commercial tools
  - commercial monitors, sniffers, intrusion detection
  - snmp RMON probes or rough/free equivalents (ntop)
  - network mgmt. GUI wonders (HP openview/Cisco/IBM, Network Associates, etc., etc.)