Broadcast Routing

Broadcasting: sending a packet to all N receivers

- routing updates in LS routing
- service/request advertisement in application layer (e.g., Novell)

Broadcast algorithm 1: N point-to-point sends

- send packet to every destination, point-to-point
- wasteful of bandwidth
- requires knowledge of all destinations

Broadcast algorithm 2: flooding

- when node receives a broadcast packet, send it out on every link
- node may receive many copies of broadcast packet, hence must be able to detect duplicates

Broadcast Routing: Reverse Path Forwarding

Goal: avoid flooding duplicates

Assumptions:

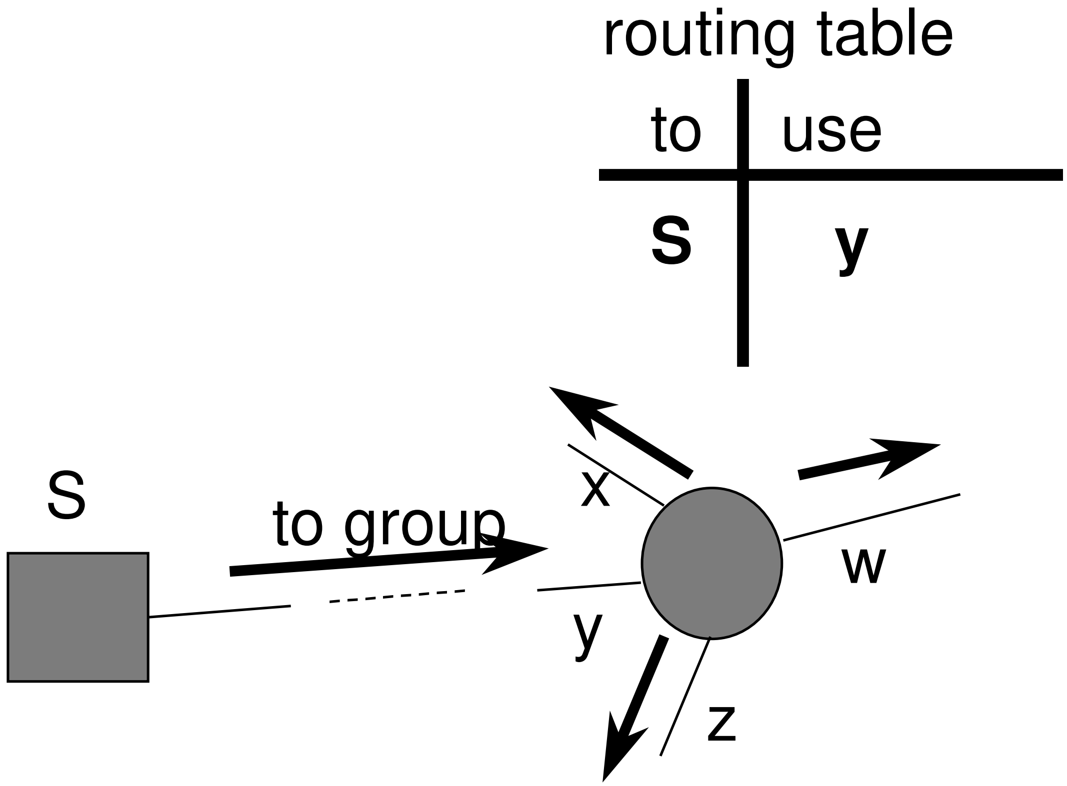
- A wants to broadcast
- all nodes know predecessor node on shortest path back to A

Reverse path forwarding: if node receives a broadcast packet

- if packet arrived on predecessor on shortest path to A, then flood to all neighbors
- otherwise ignore broadcast packet - either already arrived, or will arrive from predecessor

Reverse Path Forwarding

- flood if packet arrives from source on link that router would use to send packets to source
- otherwise discard
- rule avoids flooding loops
- uses shortest path tree from destinations to source (reverse tree)



routing table
to
use
S
y
S
to group
x
w
y
z

Distributing routing information

Q: is broadcast algorithm like reverse path forwarding good for distributing Link State updates (in LS routing)?

A:

First try (at LS broadcast distribution):

- each router keeps a copy of most recent LS packet (LSP) received from every other node
- upon receiving LSP(R) from router R:
  - if LSP(R) not identical to stored copy

    then store LSP(R), update LS info for R, and flood LSP(R)

    else ignore duplicate

How can this protocol fail?

2nd Try (at LS Broadcast Distribution)

Each router puts a sequence number on its LSP's

- upon receiving LSP(R) from R

    if (seq # > seq # of stored copy) of LSP(R)

        then store LSP(R), update LS info for R, and flood LSP(R)

        else ignore duplicate

How can this protocol fail?

3rd Try (at LS Broadcast Distribution)

- use "large" sequence numbers
- add time-based "age" field
  - each router decreases age field value as LSP(R) sits in memory
  - locally timeout (forget) LSR(R) routing info if age is zero
  - don't flood packet with age zero
- remove queued (for outgoing transmission) but unsent LSP(R) before flooding newer LSP(R)

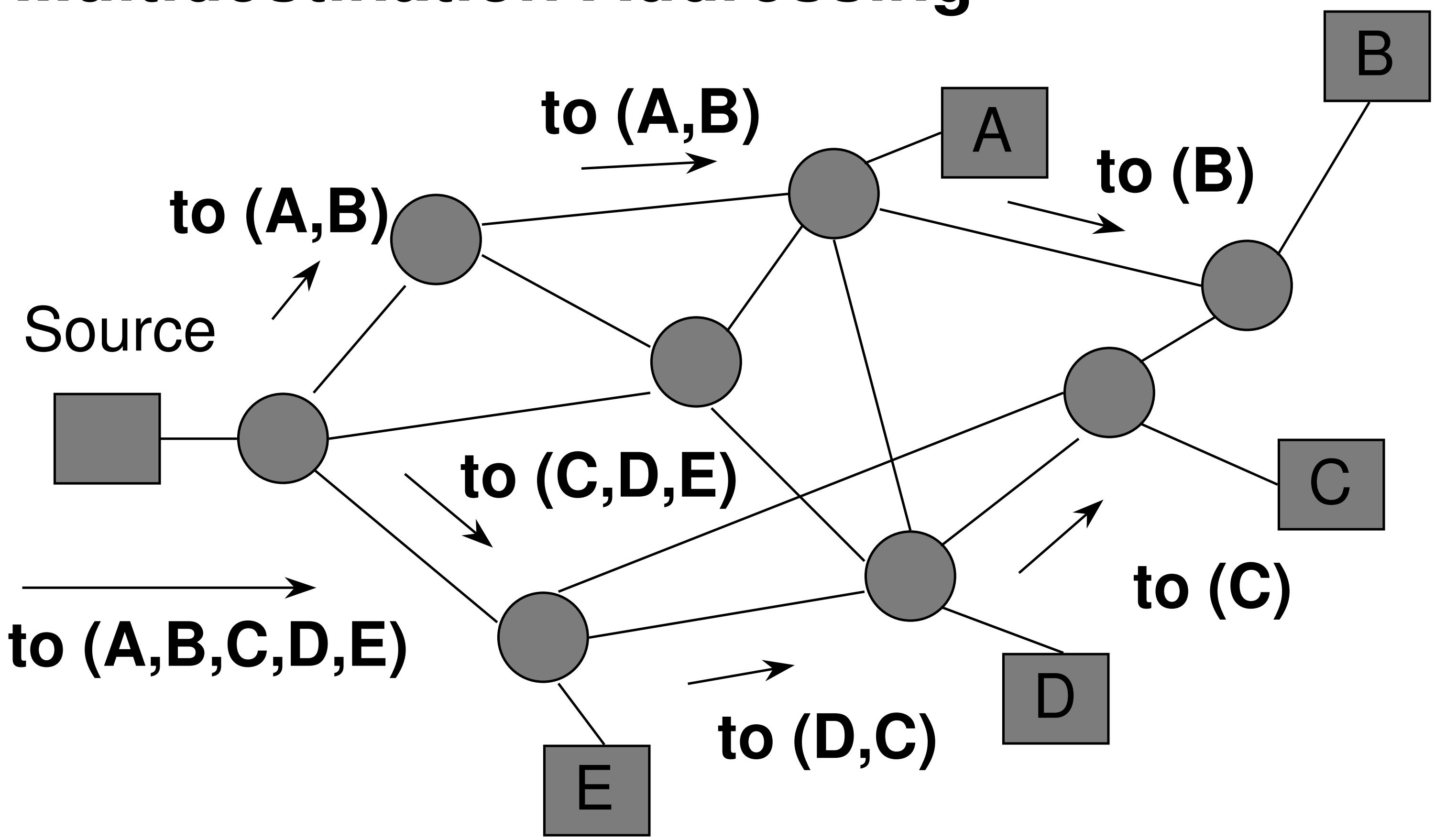
Multicast Routing

GOAL: deliver packet from one sender to many (but not all) other hosts

- deliver to M hosts in N-host network (M<N)
- *option 1:* sender establishes M point-to-point connections
- *option 2:* sender sends one packet, which is duplicated and forwarded, as needed by routers:
  - router A duplicates packet
  - router B selectively forwards

Basic Multicast Routing Protocols

- **Multidestination Addressing**

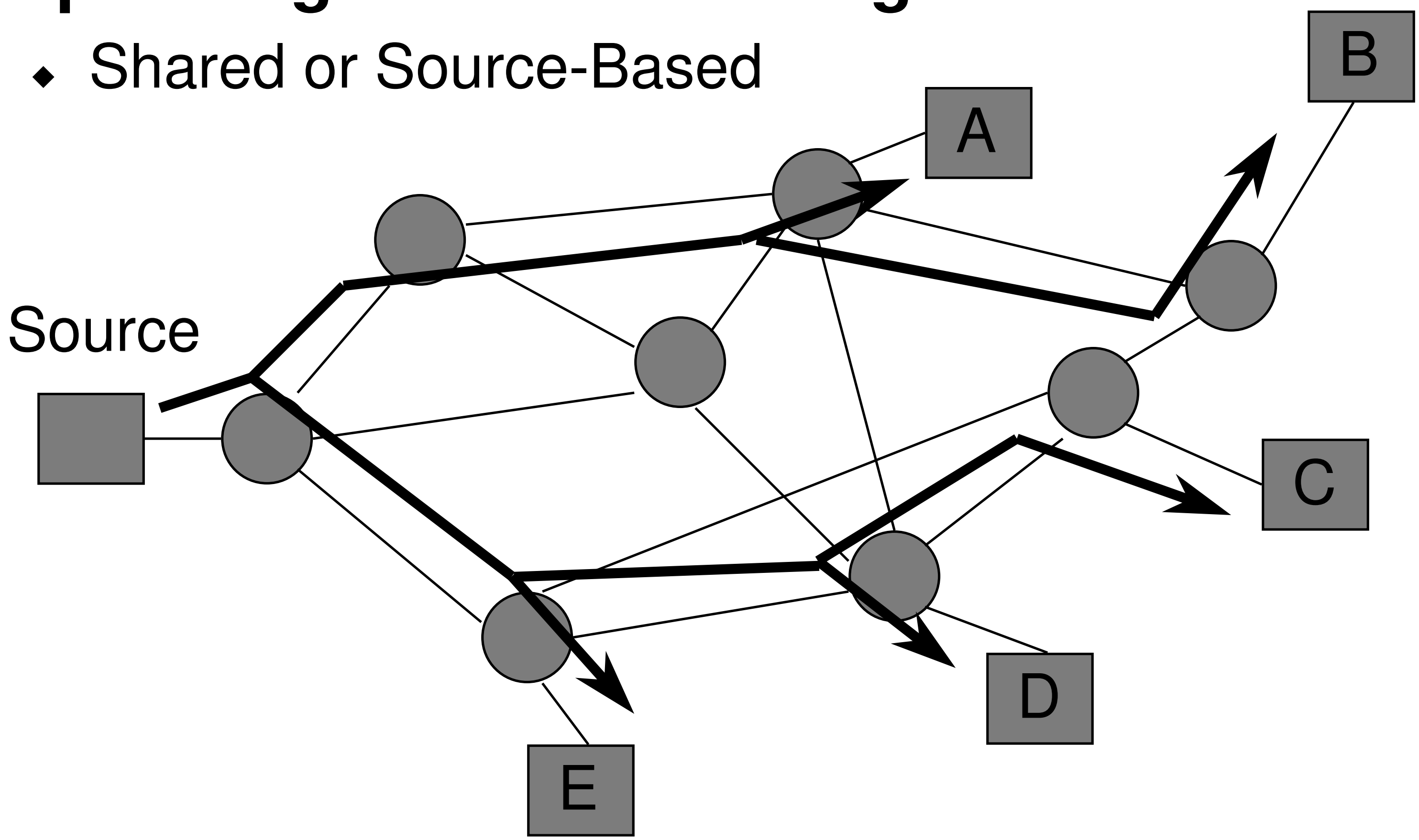


Multicast Abstraction

- multicast address associated with multicast group
- hosts join/leave multicast group
- sender sends packet to multicast address (destination)
- routers deliver to hosts that joined group address
- sender does not have to belong to multicast group

Basic Multicast Routing Protocols

- **Spanning Tree Forwarding**
  - Shared or Source-Based



Shared Tree VS Source-Based Tree

- RPF routes over **source-based tree**
  - good delay properties
  - per source overhead
- spanning tree forwarding uses **shared tree**
  - per group overhead
  - higher delays
  - more Traffic Concentration

DVMRP

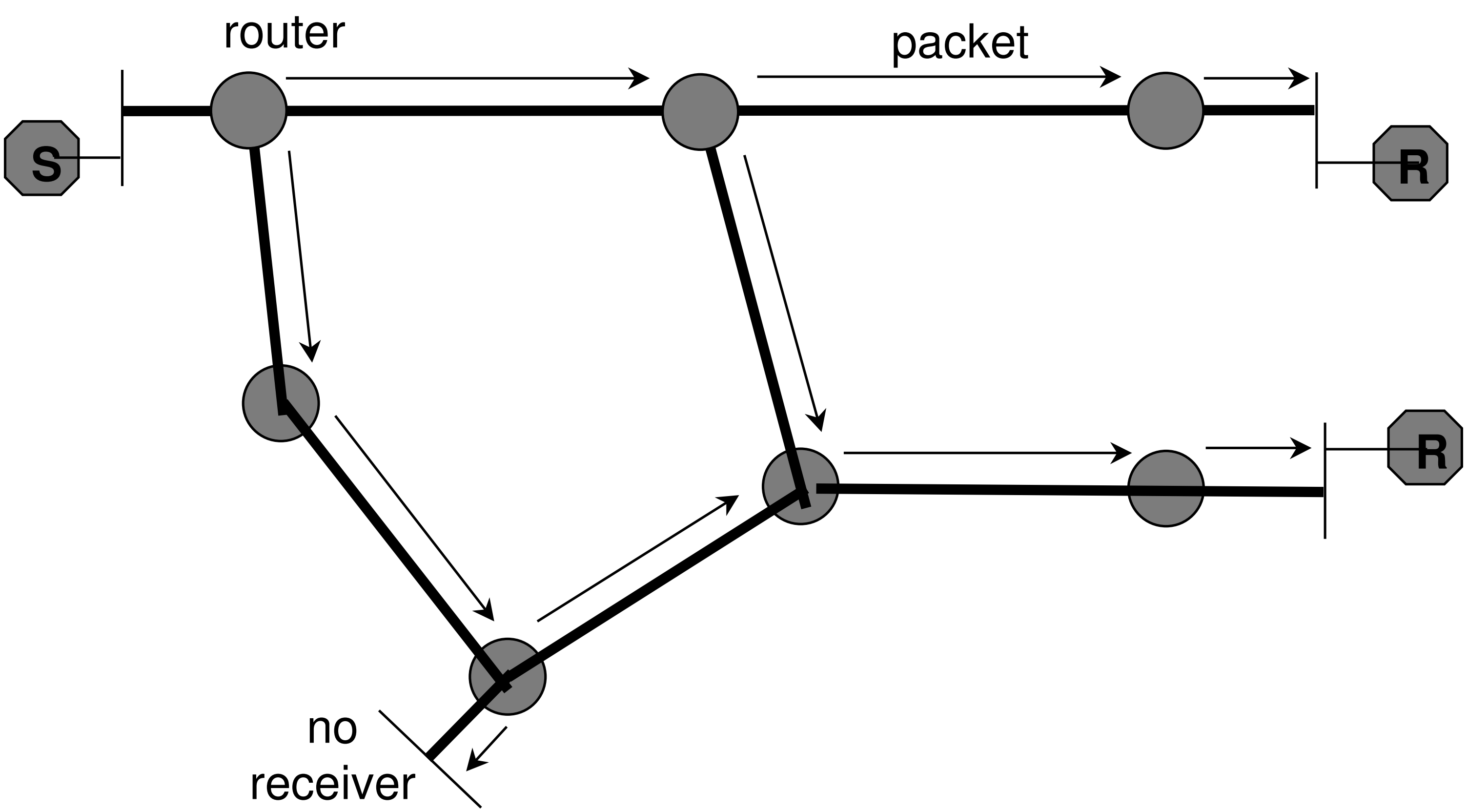
- Distance Vector Multicast Routing Protocol
  - an enhancement of Reverse Path Forwarding that :
    - uses Distance Vector Routing Packets for building tree
    - prunes broadcast tree links that are not used (non-membership reports)

Multicast Forwarding in DVMRP

1. check incoming interface: discard if not on shortest path to source
2. forward to all outgoing interfaces
3. don’t forward if interface has been *pruned*
4. prunes time out every minute

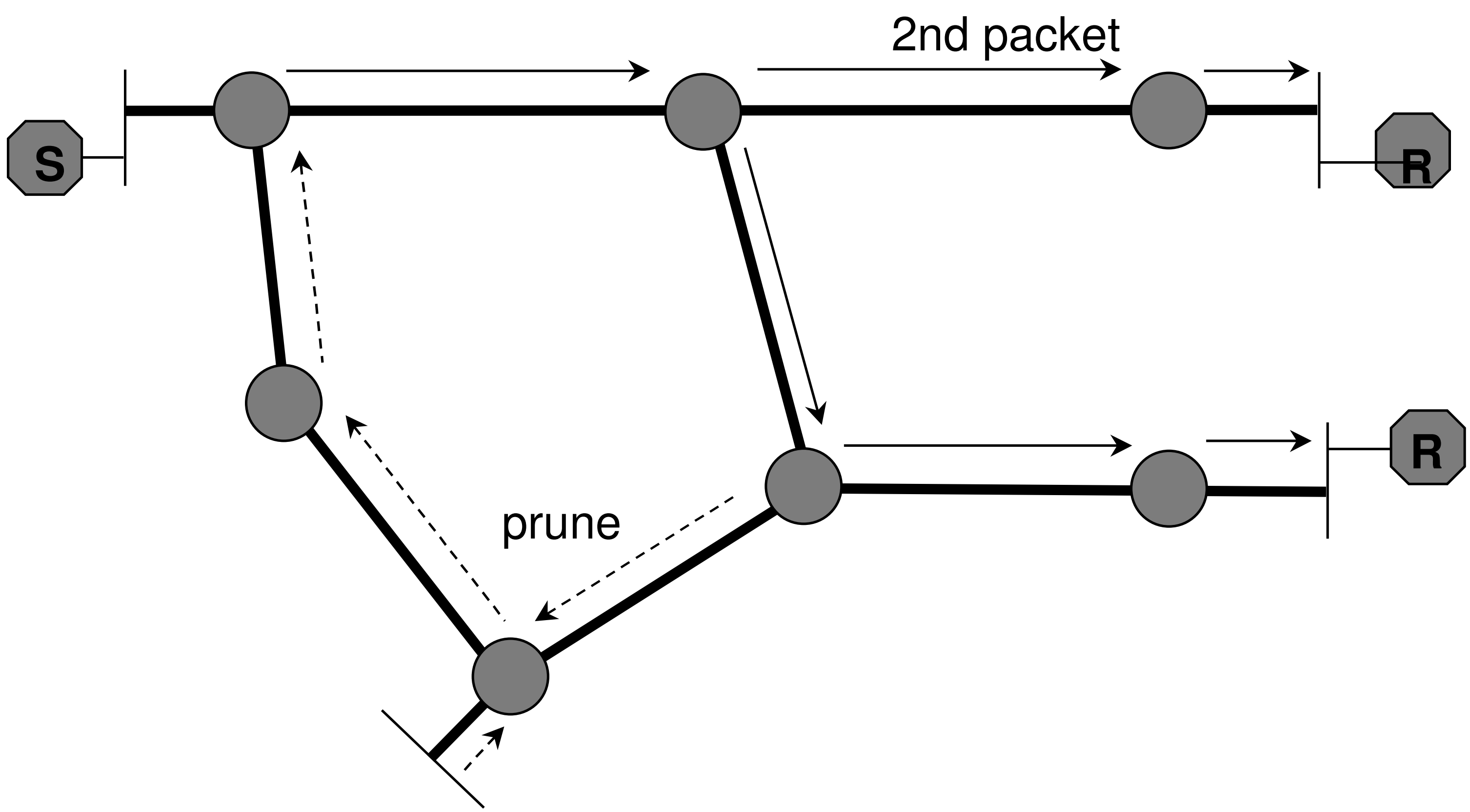
DVMRP Forwarding (cont.)

Basic idea is to flood and prune



DVMRP Forwarding (cont.)

Prune branches where no members and branches not on shortest paths



Overheard on Mbone Mailing List!

- “Help, we are unable to send prunes”
- Response:

“Well, have you tried to send plums? Raisins or grapes? ……

Perhaps your multicast implementation does not support fruit at all?”

Link State Multicast Routing

- Link-State Multicast Routing
- routers maintain topology DBs
- group-membership/link-state broadcast by routers to advertise links with members
- routers compute and cache pruned SPTs

Hierarchical Routing

Problem: as size of network grows, routing table, complexity grows

- millions of nodes (hosts, routers) in Internet

Solution: hierarchically aggregate nodes into "regions" (domain)

- node have full knowledge of routes, topological structure within region
- one (or more) nodes in region responsible for routing to the outside

Teminology:

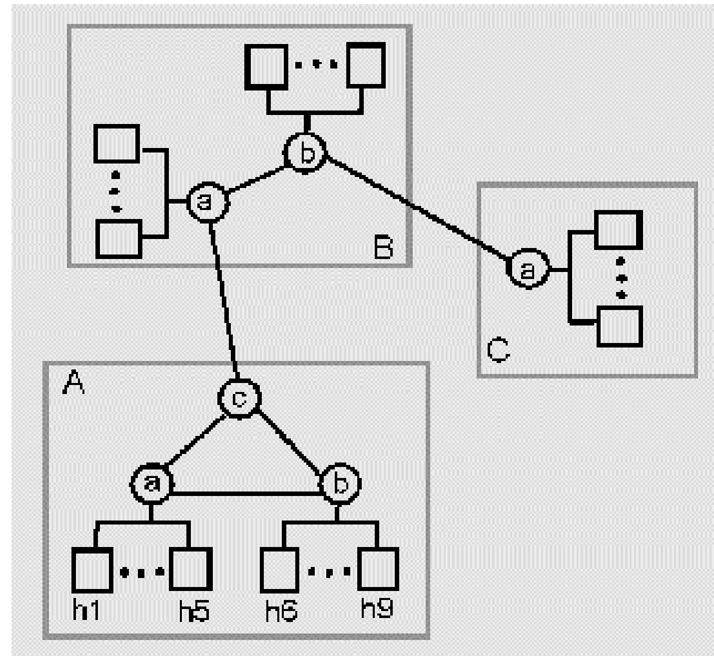
- intradomain routing: within domain
- interdomain routing: between domains
- autonomous system (AS): domain, region, administrative domain
- gateway: routes to/from domain, a.k.a. border router

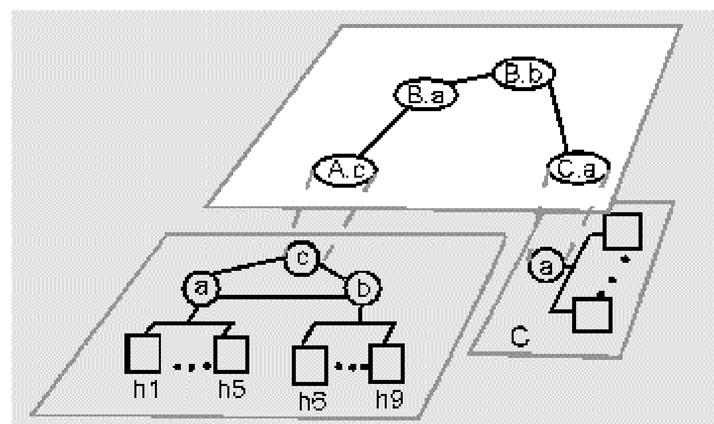
Hierarchical Routing (cont)

Three domains: A, B, C

A.a, A.b A.c run intradomain routing protocol

A.c, B.a, B.b, C.a run interdomain routing protocol among themselves





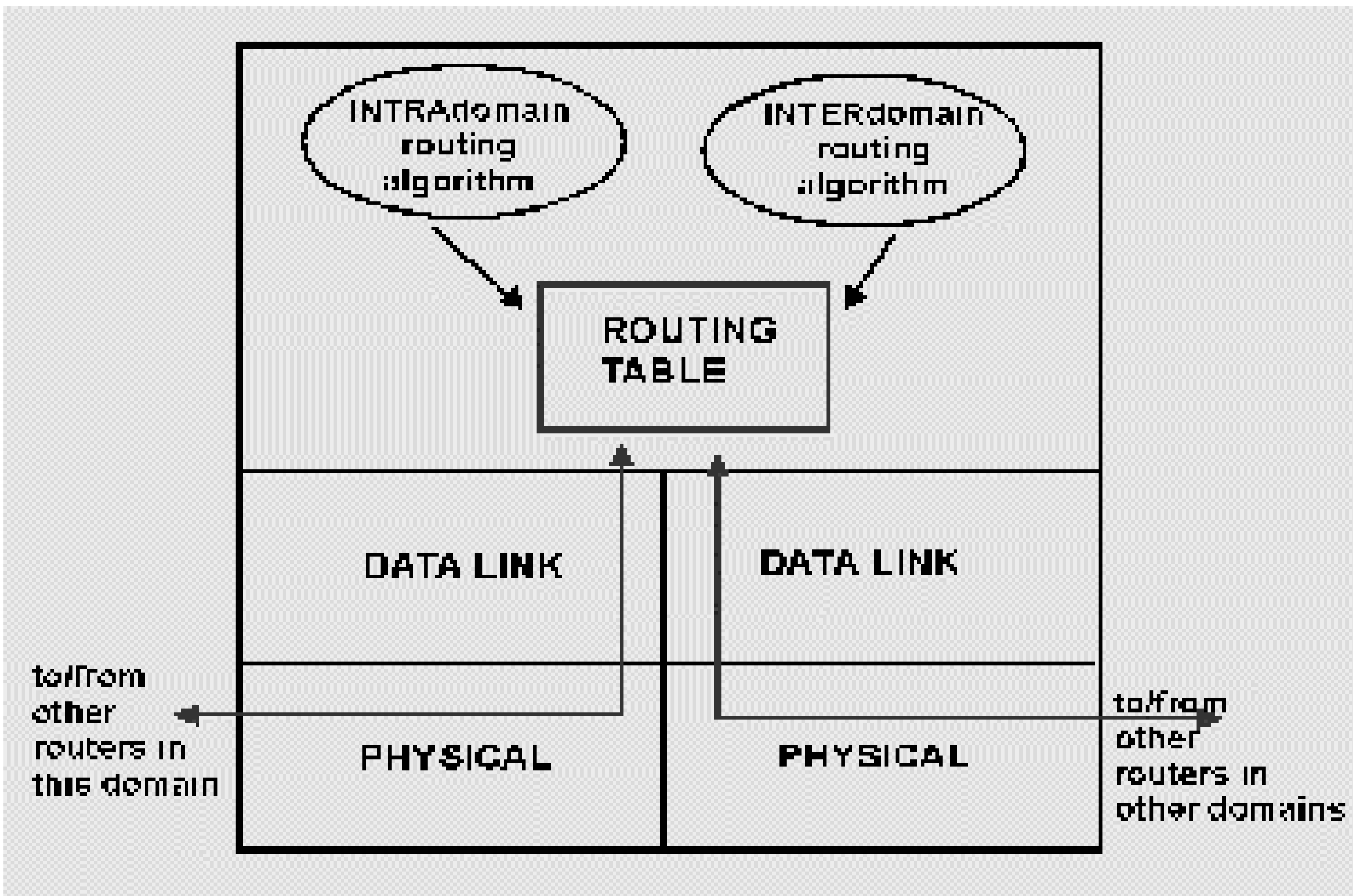
Hierarchical Routing (cont)

Different routing protocols can be used for interdomain and intradomain routing

A.a routing table:

destination	next hop
h6	A.b
.	A.b
h9	A.b
all other (default route)	A.c

A look inside A.c



Hosts and routers

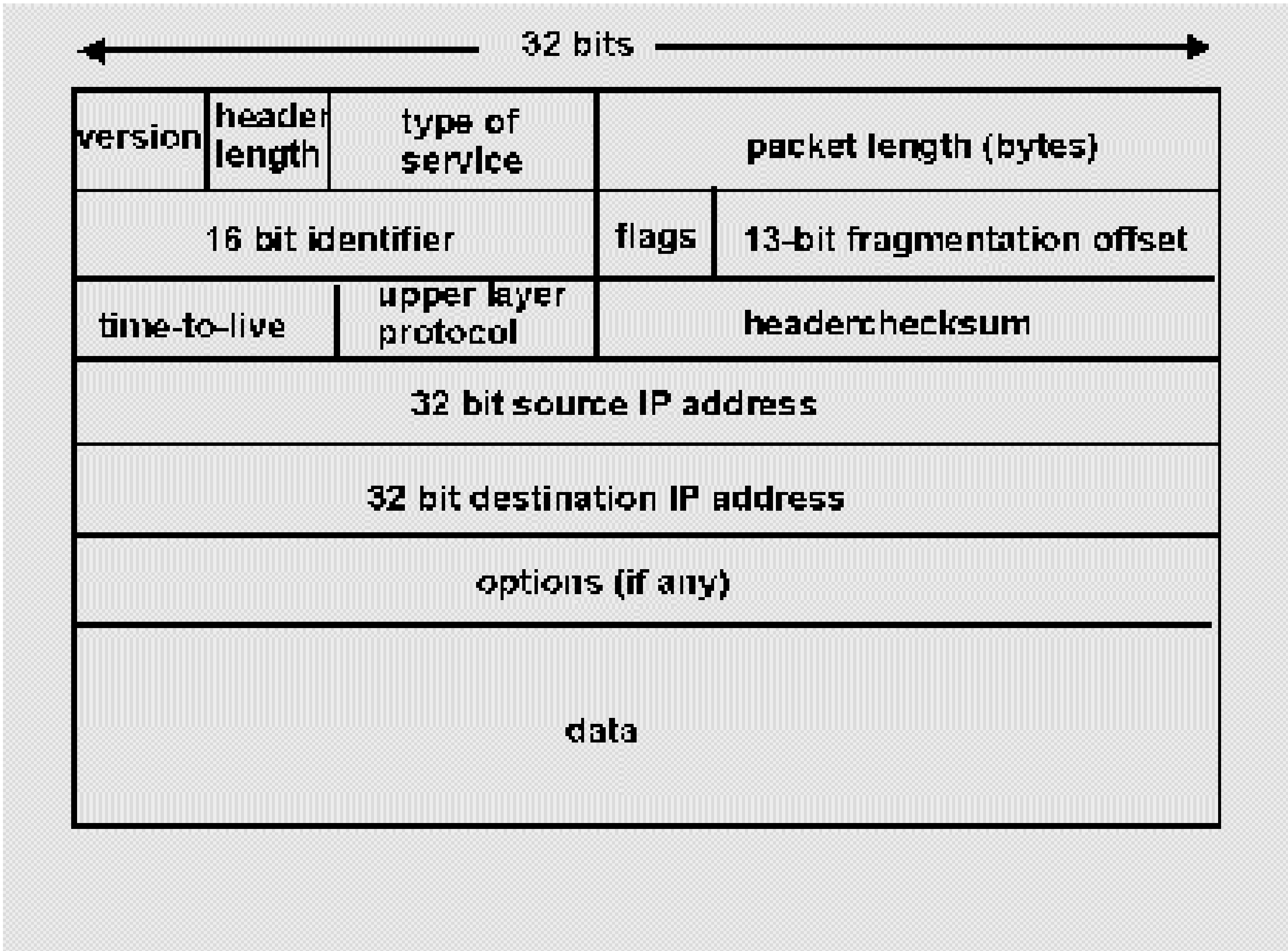
Hosts (end systems) typically perform no routing

- start packets on their way
- send packets to nearest router

Q: how do hosts learn identity of nearby router:

- ***A1:*** IP address of router hard-coded into file (see /etc/networks on many UNIX systems)
- ***A2:*** router discovery: RFC 1256
  - router periodically broadcasts its existence to attached hosts
  - host (on startup) broadcasts query (who is my router) on attached links/LANs

Network Layer Case Study: the Internet



Network Layer Case Study: the Internet

Fields in IP packet:

- ***version number:*** (of IP protocol), current version is 4, new version is 6
- ***header length:*** because of options, length of header is variable
- ***TOS:*** not used, idea was to allow different levels of reliability, real-time, etc
- ***packet length:*** header plus data
- ***identifier:*** used with IP fragmentation to identify fragments belonging to same original IP packet
- ***flags:*** 2 bits: do not fragment, more fragments

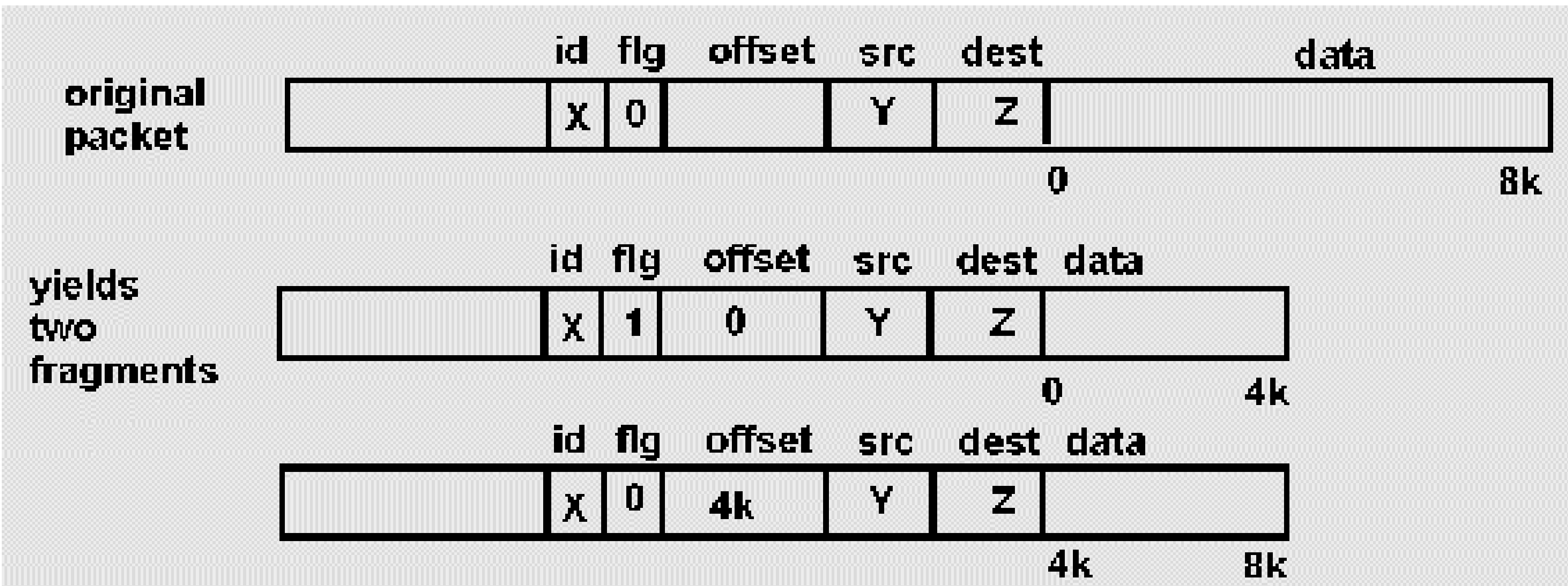
Network Layer Case Study: the Internet

- ***fragmentation offset:*** if this a fragment, where it belongs in original packet
- ***time-to-live:*** decremented by each router, so a packet will not loop forever in the net
- ***protocol:*** which upper layer protocol to demultiplex to. See RFC 1700
- ***header checksum:*** recomputed at each hop, as TTL changes
- ***source, dest IP address:*** of original sender, and eventual recipient

IP Fragmentation and Reassembly

- transport layer packet may be too big to send in single IP packet
- underlying data link protocol will constraint maximum IP length
- fragmentation: IP packet divided into fragments by IP
  - each fragment becomes its own IP packet
  - each address has same identifier, source, destination address
  - fragment offset gives offset of data from start of original packet
  - more fragment bit: 0 means last bit in this fragment
  - fragments not reassembled until final destination

Fragmentation Example



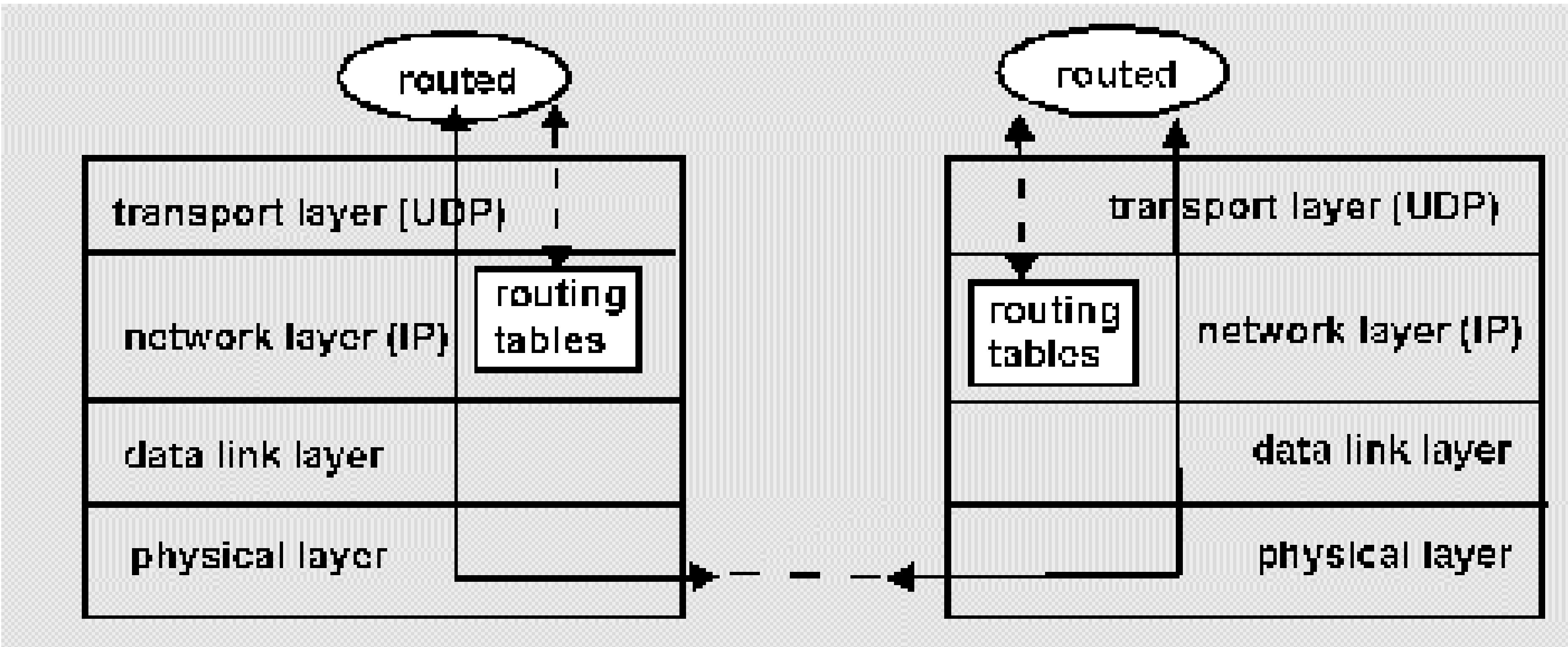
Internet Intradomain Routing: RIP

RIP: Routing Information Protocol, uses distance vector algorithm, with link costs of 1

- shortest path
- routing table sent to neighbors every 30 seconds, or when route costs change

Implemented as a daemon (user-level process)

- communicates with other attached router using UDP packets
  - **note:** UDP packets can be lost!
  - if route via neighbor not updated in 3 minutes, timeout route (set cost to infinity)
- called **routed** on UNIX systems



RIP-2

- support for authentication (a cleartext password)

A RIP routing table

Example table taken from freya.cs.umass.edu:

~ netstat -rn (note: on freya.cs.umass.edu)

Destination	Gateway	Flags	Refcnt	Use	Interface
127.0.0.1	127.0.0.1	UH	25	2260	lo0
Default	128.119.40.254	UG	5	15223	ln0
128.119	128.119.40.195	U	28	188671	ln0

Internet Intradomain Routing: OSPF

OSPF: open shortest path first

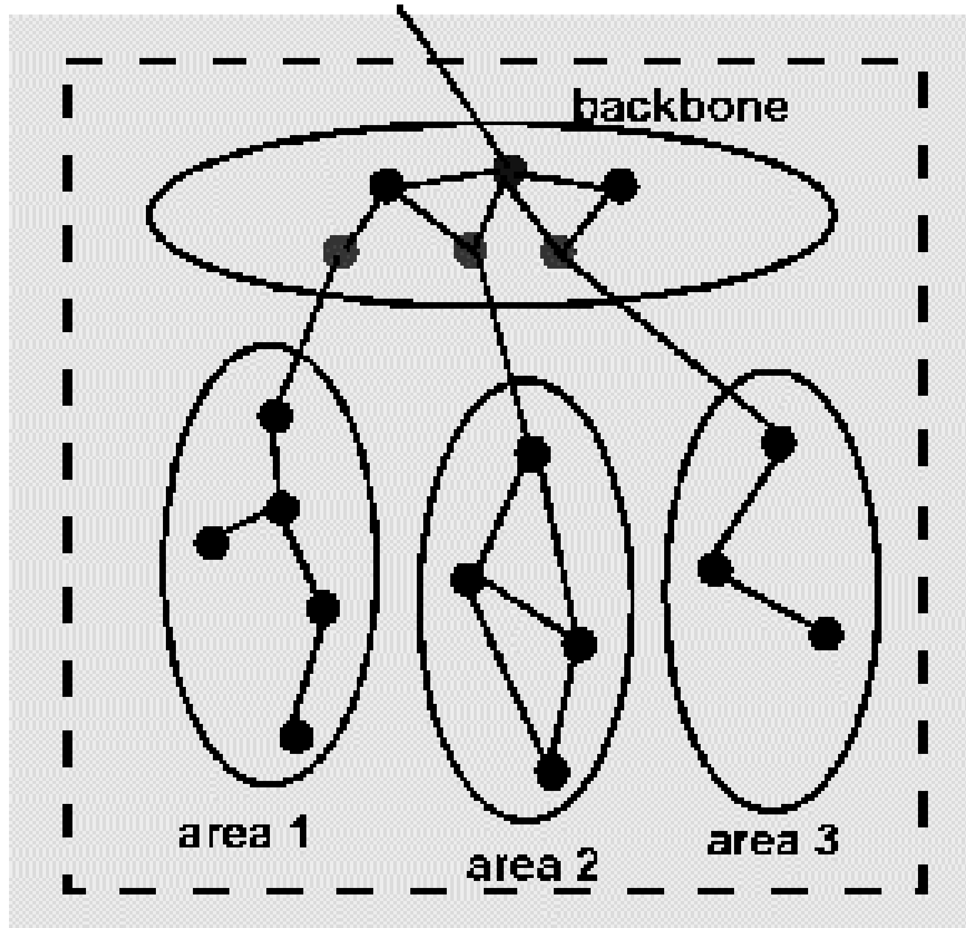
- open: a published standard (RFC 1247)
- interior gateway protocol: for intradomain outing within an autonomous system (AS)
- uses link state algorithm to determine routes
  - each outgoing link (interface) assigned dimensionless cost
  - different cost can be used for different TOS
  - ***load balancing:*** with several equal-cost-paths to destination, will distribute load across both paths

OSPF: Support for hierarchy

- autonomous system divided into "areas"
- one area designated "backbone"
  - area border routers in backbone route between areas
  - other routers in backbone also
- AS boundary router talks to outside world

Internet Intradomain Routing: OSPF (cont)

- **area router: red**
- **boundary router: blue**



Intra-area routing:

- never cross backbone

To get from one area to another:

- source area -> backbone -> destination area

Interdomain Internet Routing: BGP

BGP: Border Gateway Protocol

- routing between nodes in different autonomous systems (i.e., routing between networks)
- RFC 1267, 1268
- uses a distance vector approach

Policy-Based Routing

- rather than costs to destinations, BGP routers exchange full path information (networks crossed) to destination
- router can decide on policy basis which route to take
  - e.g. "traffic from my AS should not cross AS's a,b,c,d"

BGP implementation

- **BGP** implemented as a daemon (user-level process)
- communicates with other BGP routers using TCP