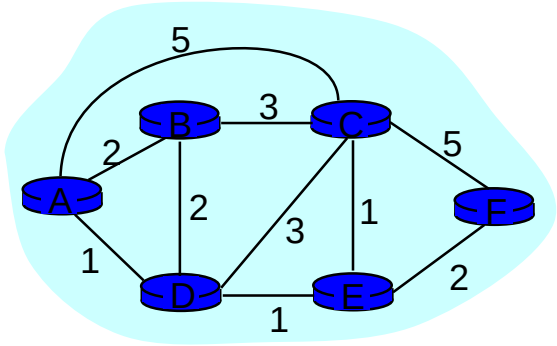
IF2230 Jaringan Komputer

Routing

Robithoh Annur
Andreas Bara Timur
Monterico Andrian

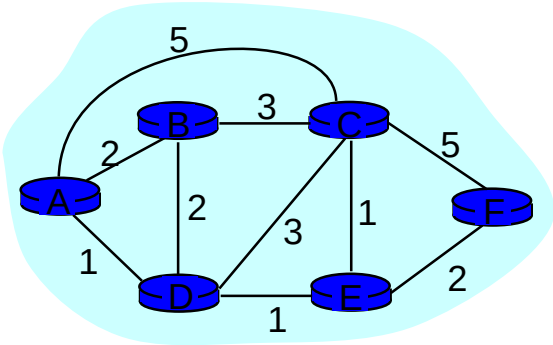
Routing

- Goal: determine a “good” path through the network from source to destination
  - Good means usually the shortest path
- Network modeled as a graph
  - Routers → nodes
  - Link →edges
    - Edge cost: delay, congestion level,…



Basic Routing Problem

- Assume
  - A network with N nodes, where each edge is associated a cost
  - A node knows only its neighbors and the cost to reach them
- How does each node learn how to reach every other node along the shortest path?



Routing: Issues

- How are routing tables determined?
- Who determines table entries?
- What info is used in determining table entries?
- When do routing table entries change?
- Where is routing info stored?
- How to control routing table size?

Answer these questions, we are done!

Routing Paradigms

- Hop-by-hop Routing
  - Each packet contains destination address
  - Each router chooses next-hop to destination
    - routing decision made at each (intermediate) hop!
    - packets to same destination may take different paths!
  - Example: IP's default datagram routing
- Source Routing
  - Sender selects the path to destination precisely
  - Routers forward packet to next-hop as specified
    - Problem: if specified path no longer valid due to link failure!
  - Example:
    - IP's loose/strict source route option
    - virtual circuit setup phase in ATM (or MPLS)

Routing Algorithms/Protocols

Issues Need to Be Addressed:

- Route selection may depend on different criteria
  - Performance: choose route with the smallest delay
  - Policy: choose a route that doesn't cross .gov network
- Adapt to changes in network topology or condition
  - Self-healing: little or no human intervention
- Scalability
  - Must be able to support a large number of hosts, routers

Centralized vs. Distributed Routing Algorithms

Centralized:

- A centralized route server collects routing information and network topology, makes route selection decisions, then distributes them to routers

Distributed:

- Routers cooperate using a distributed protocol
  - to create mutually consistent routing tables
- Two standard distributed routing algorithms
  - Distance Vector (DV) routing
  - Link State (LS) routing

Link State *vs* Distance Vector

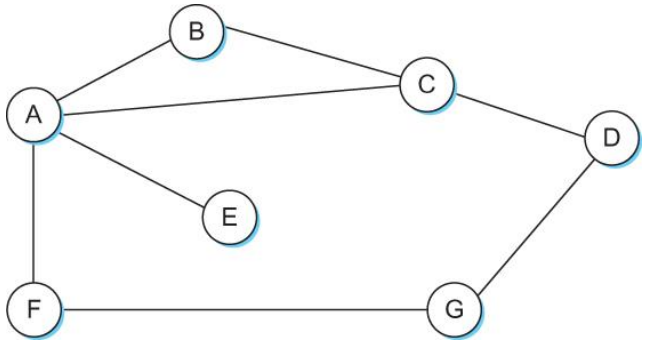
- Both assume that
  - The address of each neighbor is known
  - The cost of reaching each neighbor is known
- Both find global information
  - By exchanging routing info among neighbors
- Differ in the information exchanged and route computation
  - LS: tells every other node its distances to neighbors
  - DV: tells neighbors its distance to every other node

Distance Vector

- The distance vector routing algorithm is sometimes called as Bellman-Ford algorithm
- Every T seconds each router sends its table to its neighbor each each router then updates its table based on the new information
- Problems include fast response to good new and slow response to bad news. Also too many messages to update

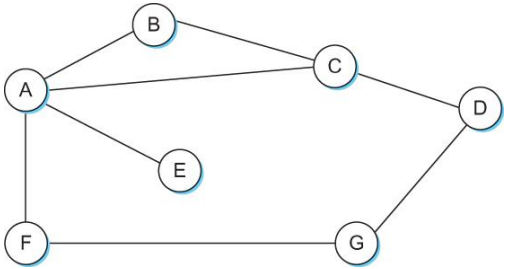
Distance Vector

- Each node constructs a one dimensional array (a vector) containing the “distances” (costs) to all other nodes and distributes that vector to its immediate neighbors
- Starting assumption is that each node knows the cost of the link to each of its directly connected neighbors



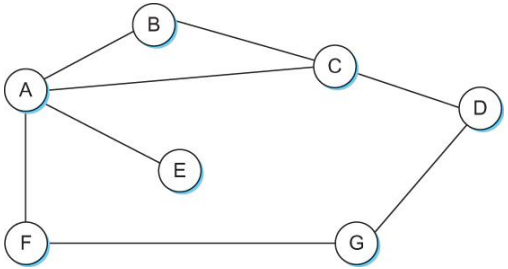
Distance Vector

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0



Initial distances stored at each node (global view)

Distance Vector



Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

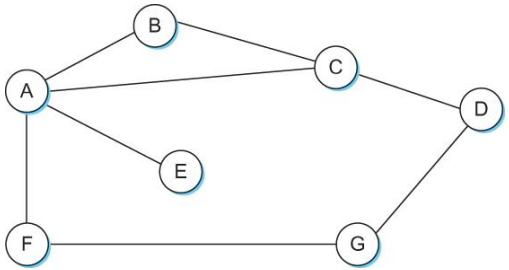
Initial routing table at node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Final routing table at node A

Distance Vector

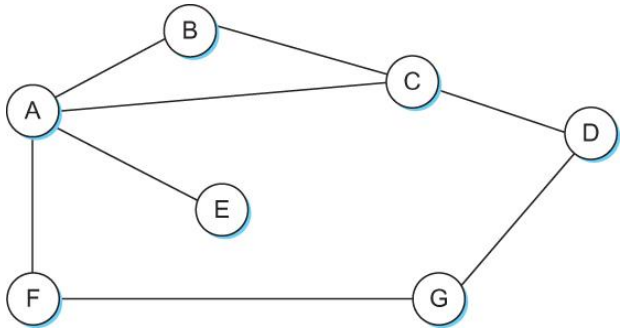
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Final distances stored at each node (global view)

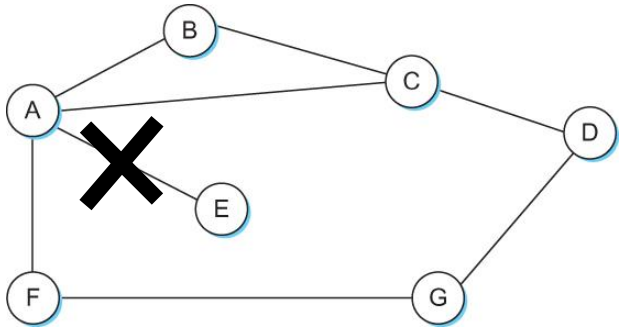
Distance Vector

- When a node detects a link failure
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update to A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and sends update to F
  - F decides it can reach G in 4 hops via A



Distance Vector

- Slightly different circumstances can prevent the network from stabilizing
  - Suppose the link from A to E goes down
  - In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E
  - Depending on the exact timing of events, the following might happen
    - Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A
    - Node A concludes that it can reach E in 4 hops and advertises this to C
    - Node C concludes that it can reach E in 5 hops; and so on.
    - This cycle stops only when the distances reach some number that is large enough to be considered infinite
      - **Count-to-infinity problem**



Count-to-infinity Problem

- Use some relatively small number as an approximation of infinity
- For example, the maximum number of hops to get across a certain network is never going to be more than 16

- One technique to improve the time to stabilize routing is called *split horizon*
  - When a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor
  - For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update

- In a stronger version of split horizon, called *split horizon with poison reverse*
  - B actually sends that back route to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E
  - For example, B sends the route (E, ∞) to A

Link State Algorithm

- Basic idea: Distribute link state packet to all routers
  - Topology of the network
    - Cost of each link in the network
- Each router independently computes optimal paths
  - From itself to every destination
  - Routes are guaranteed to be loop free if
    - Each router sees the same cost for each link
    - Uses the same algorithm to compute the best path

Link State Routing

Strategy: Send to all nodes (not just neighbors) information about directly connected links (not entire routing table).

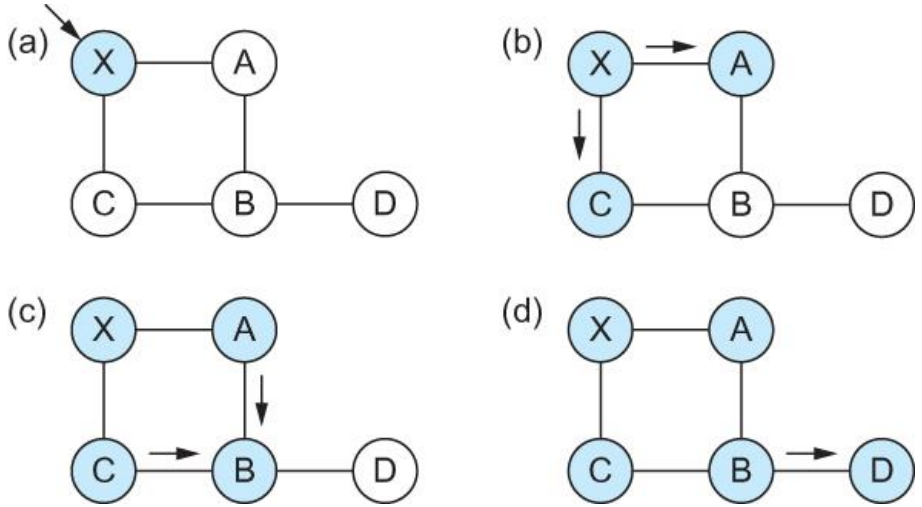
- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet
- Reliable Flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically; increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP; discard when TTL=0

Topology Dissemination

- Each router creates a set of link state packets (LSPs)
  - Describing its links to neighbors
  - LSP contains
    - Router id, neighbor's id, and cost to its neighbor
- Copies of LSPs are distributed to all routers
  - Using controlled flooding
- Each router maintains a topology database
  - Database containing all LSPs

Link State

Reliable Flooding



Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

Shortest Path Routing

- Dijkstra's Algorithm - Assume non-negative link weights
  - N: set of nodes in the graph
  - l((i, j): the non-negative cost associated with the edge between nodes i, j $\in$N and l(i, j) = ∞ if no edge connects i and j
  - Let s $\in$N be the starting node which executes the algorithm to find shortest paths to all other nodes in N
  - Two variables used by the algorithm
    - M: set of nodes incorporated so far by the algorithm
    - C(n) : the cost of the path from s to each node n
    - The algorithm

```
M = {s}
For each n in N – {s}
     C(n) = l(s, n)
while ( N ≠ M)
   M = M ∪ {w} such that C(w) is the minimum
                              for all w in (N-M)
   For each n in (N-M)
           C(n) = MIN (C(n), C(w) + l(w, n))
```

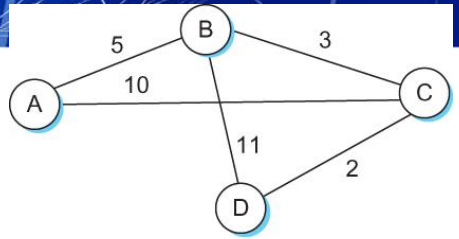
Shortest Path Routing

- In practice, each switch computes its routing table directly from the LSP's it has collected using a realization of Dijkstra's algorithm called the *forward search algorithm*
- Specifically each switch maintains two lists, known as **Tentative** and **Confirmed**
- Each of these lists contains a set of entries of the form (Destination, Cost, NextHop)

Shortest Path Routing

- The algorithm
  - Initialize the **Confirmed** list with an entry for myself; this entry has a cost of 0
  - For the node just added to the **Confirmed** list in the previous step, call it node **Next**, select its LSP
  - For each neighbor (Neighbor) of **Next**, calculate the cost (Cost) to reach this Neighbor as the sum of the cost from myself to Next and from Next to Neighbor
    - If Neighbor is currently on neither the **Confirmed** nor the **Tentative** list, then add (Neighbor, Cost, Nexthop) to the **Tentative** list, where Nexthop is the direction I go to reach Next
    - If Neighbor is currently on the **Tentative** list, and the Cost is less than the currently listed cost for the Neighbor, then replace the current entry with (Neighbor, Cost, Nexthop) where Nexthop is the direction I go to reach Next
  - If the **Tentative** list is empty, stop. Otherwise, pick the entry from the **Tentative** list with the lowest cost, move it to the **Confirmed** list, and return to Step 2.

Shortest Path Routing



Step	Confirmed	Tentative	Comments
1	(D,0,–)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,–)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,–) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,–) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,–) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,–) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,–) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

Link State *vs* Distance Vector

- Tells everyone about neighbors
- Controlled flooding to exchange link state
- Dijkstra's algorithm
- Each router computes its own table
- May have oscillations
- Open Shortest Path First (OSPF)

- Tells neighbors about everyone
- Exchanges distance vectors with neighbors
- Bellman-Ford algorithm
- Each router's table is used by others
- May have routing loops
- Routing Information Protocol (RIP)

Link State vs. Distance Vector (cont'd)

Message complexity

- LS: $O(n^2*e)$ messages
  - n: number of nodes
  - e: number of edges
- DV: O(d*n*k) messages
  - d: node's degree
  - k: number of rounds

Time complexity

- LS: O(n*log n)
- DV: O(n)

Convergence time

- LS: O(1)
- DV: O(k)

Robustness: what happens if router malfunctions?

- LS:
  - node can advertise incorrect *link* cost
  - each node computes only its *own* table
- DV:
  - node can advertise incorrect *path* cost
  - each node's table used by others; error propagate through network

Routing in the Real World

Our routing study thus far - idealization

- all routers identical
- network "flat"

How to do routing in the Internet

- scalability and policy issues

scale: with 200 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

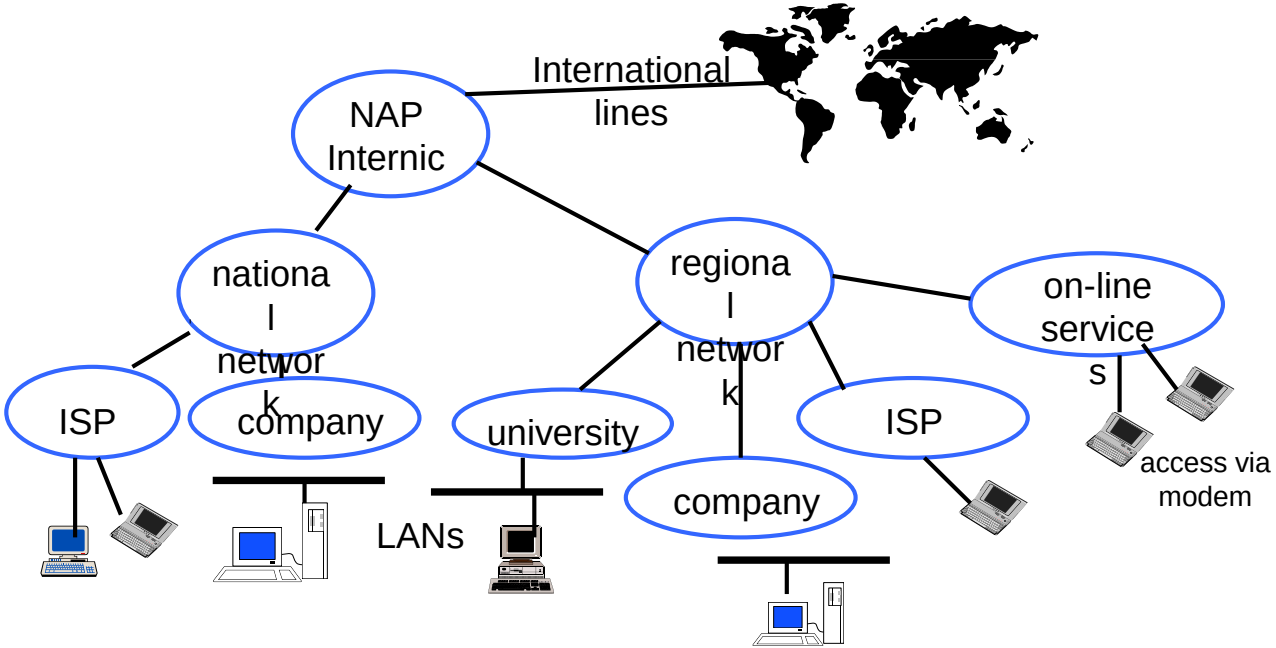
Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other hierarchically:
  - Stub AS: small corporation: one connection to other AS's
  - Multihomed AS: large corporation (no transit): multiple connections to other AS's
  - Transit AS: provider, hooking many AS's together

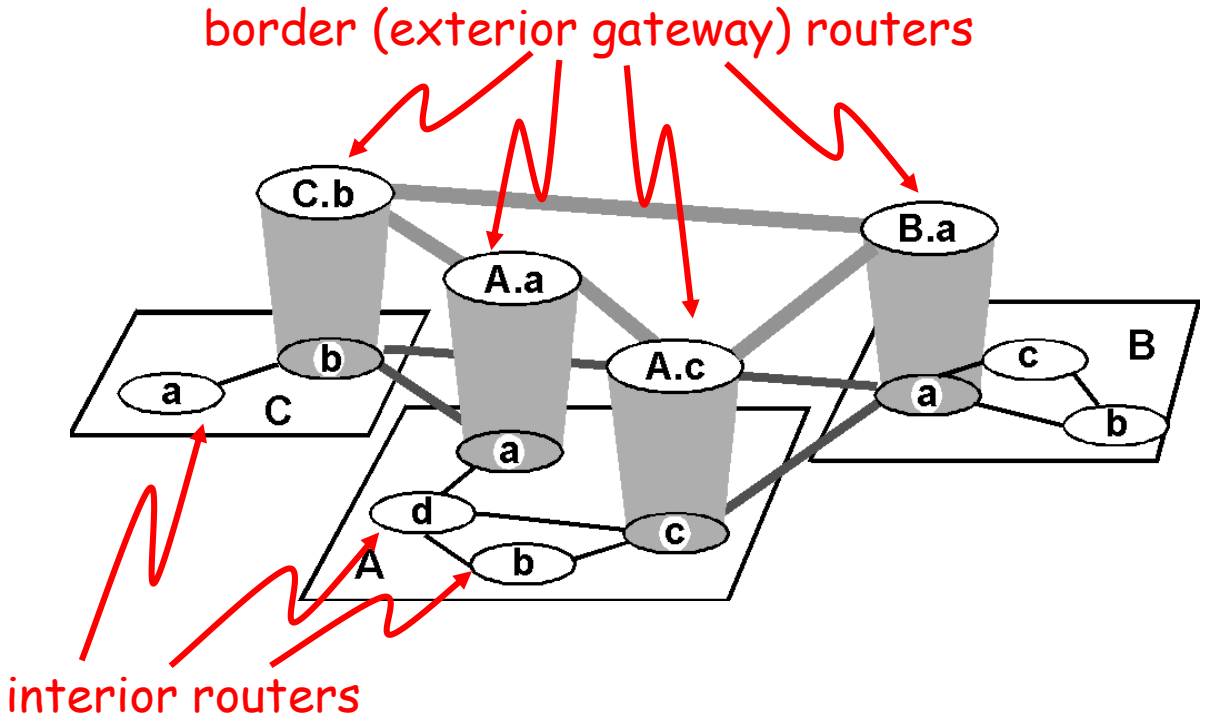
- Two-level routing:
  - Intra-AS: administrator responsible for choice of routing algorithm within network
  - Inter-AS: unique standard for inter-AS routing: BGP

Internet Architecture

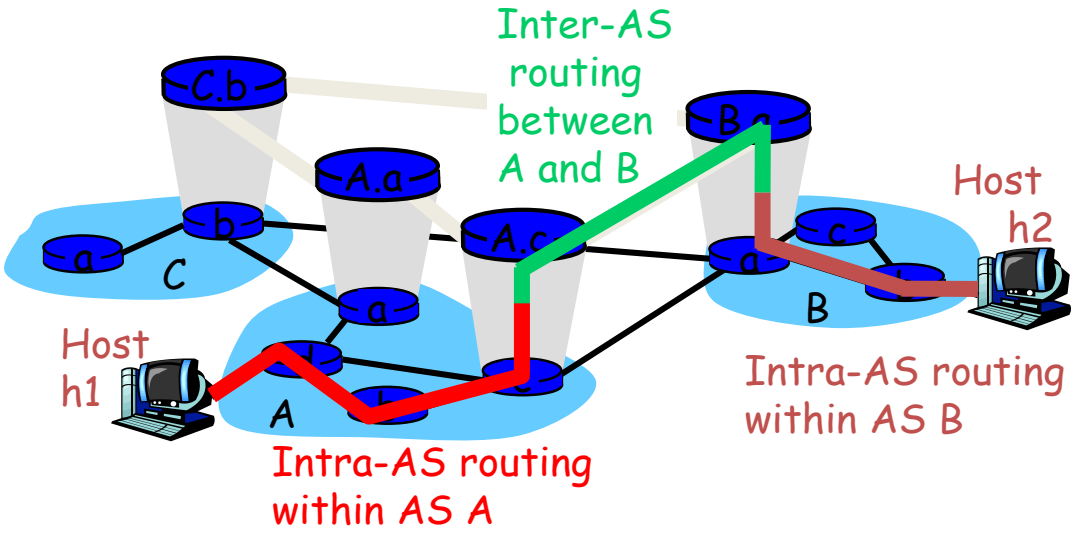
Internet: “networks of networks”!



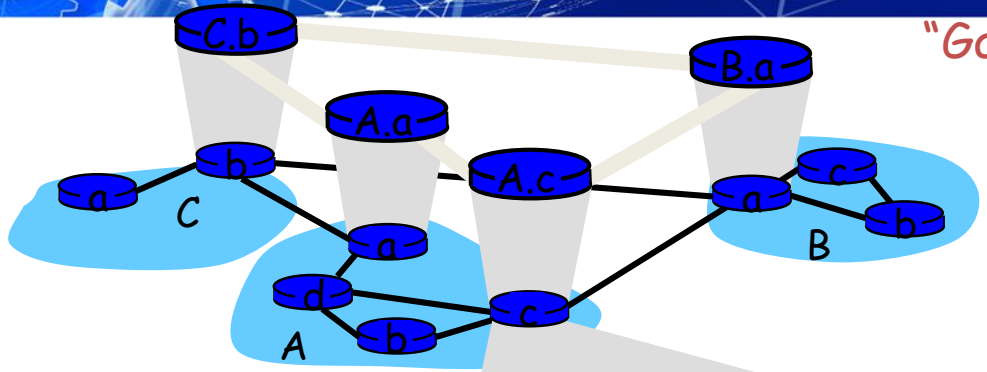
Internet autonomous system AS Hierarchy



Intra-AS vs. Inter-AS Routing

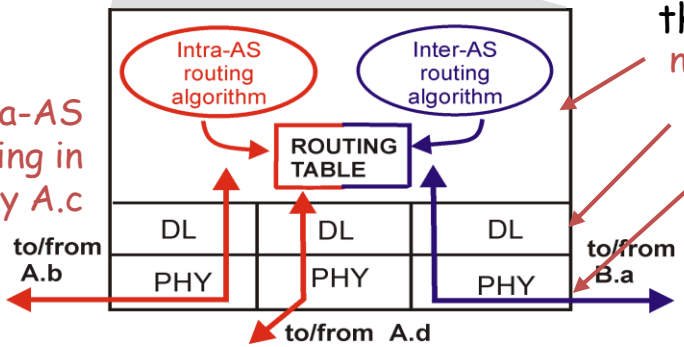


Intra-AS and Inter-AS Routing



"Gateways":

- perform inter-AS routing amongst themselves
- perform intra-AS routing with other routers in their AS



inter-AS, intra-AS routing in gateway A.c

network layer

link layer

physical layer

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IS-IS: Intermediate System to Intermediate System (OSI Standard)
  - EIGRP: Extended Interior Gateway Routing Protocol (Cisco proprietary)

Why Different Intra- and Inter-AS Routing?

Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:

- hierarchical routing saves table size, update traffic

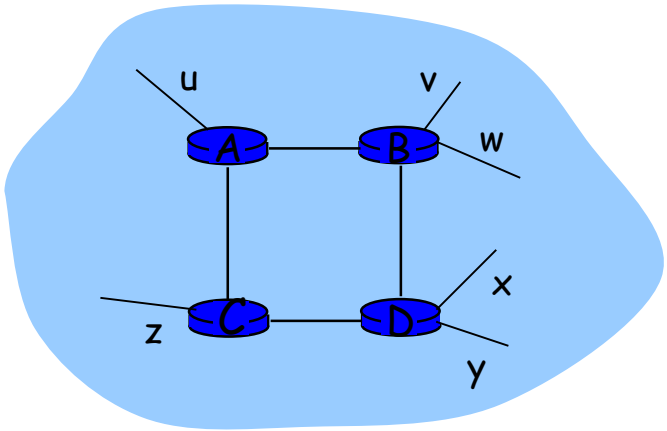
Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)

Number of hops from source router A to various subnets:

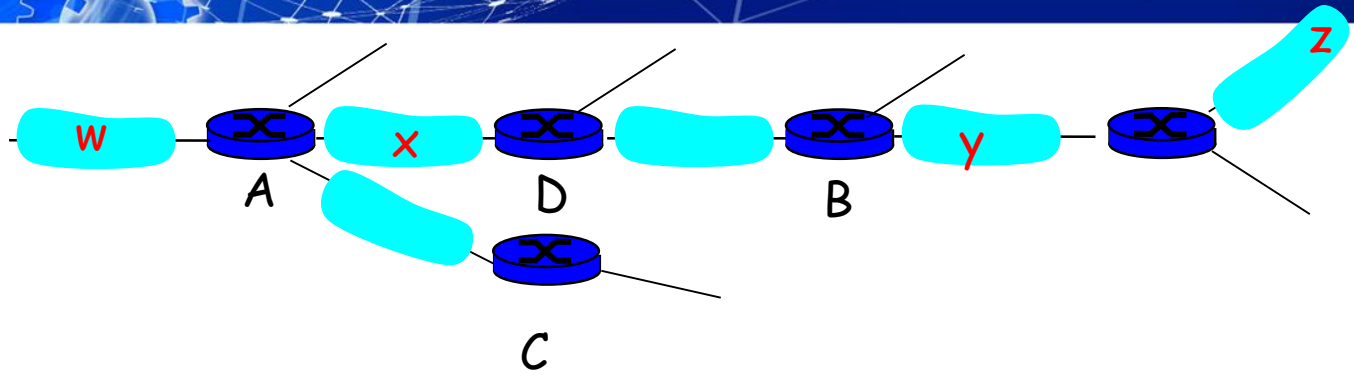


destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- Each advertisement: list of up to 25 destination nets within AS

RIP: Example



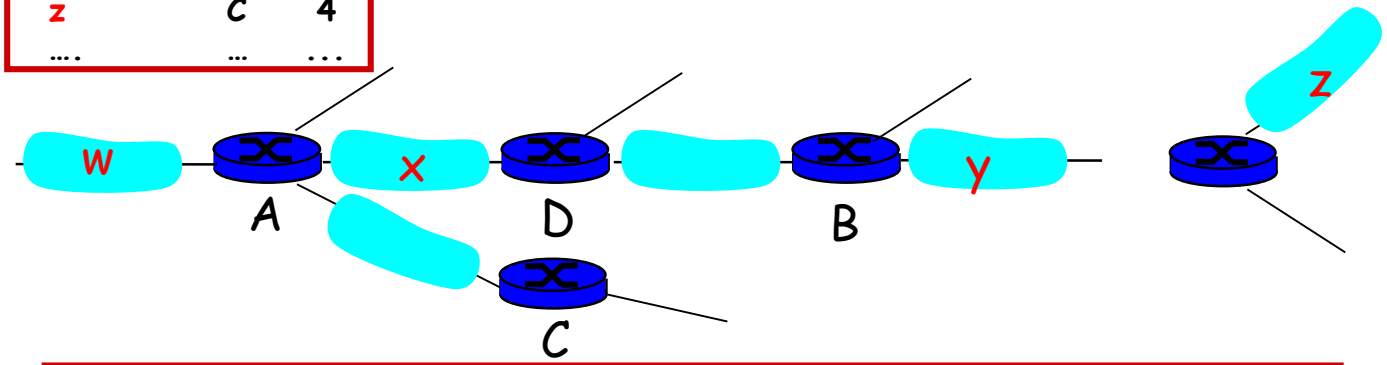
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing table in D

RIP: Example

Dest	Next	hops
w	-	-
x	-	-
z	C	4
….	…	…

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	….

Routing table in D

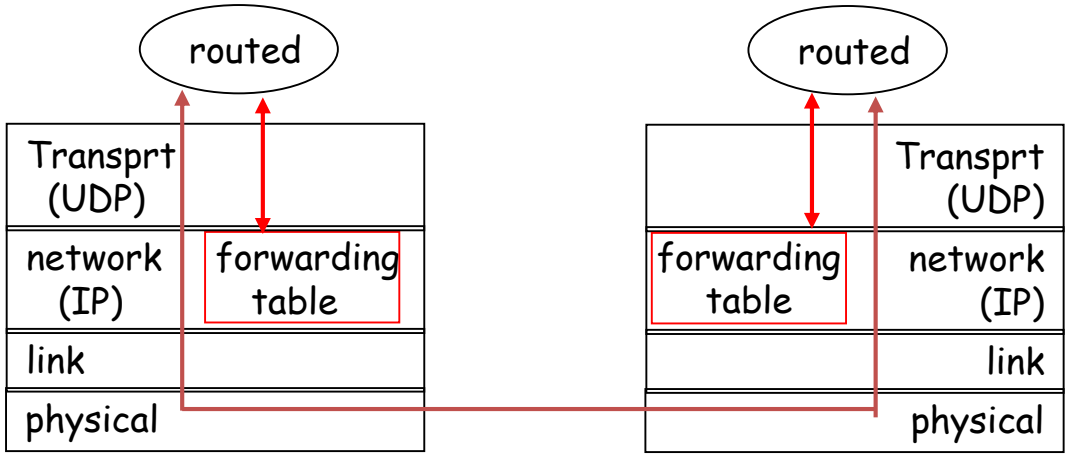
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



OSPF (Open Shortest Path First)

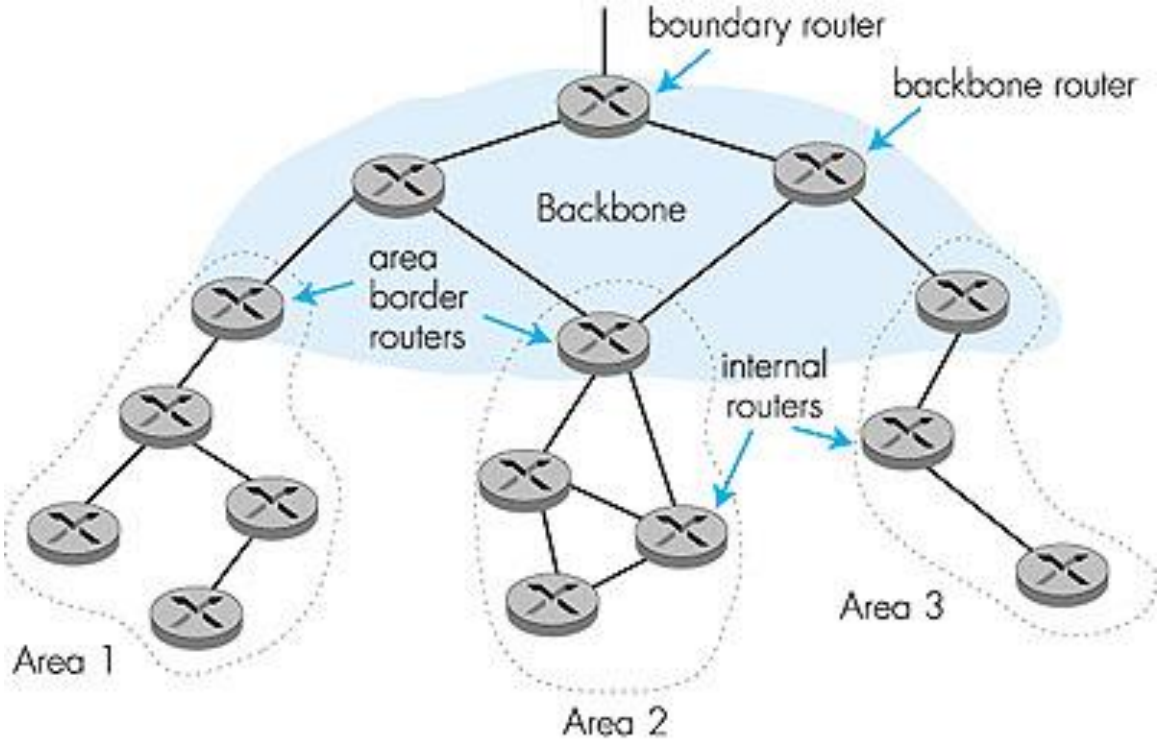
- “open”: publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm

- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)
  - Carried in OSPF messages directly over IP (rather than TCP or UDP

OSPF “advanced” features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort; high for real time)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
  - Communications between areas via backbone
- **Area border routers:** “summarize” distances to nets in own area, advertise to other Area Border routers.
- **Backbone routers:** run OSPF routing limited to backbone.
- **Boundary routers:** connect to other AS’s.

Inter-AS Routing in the Internet: BGP

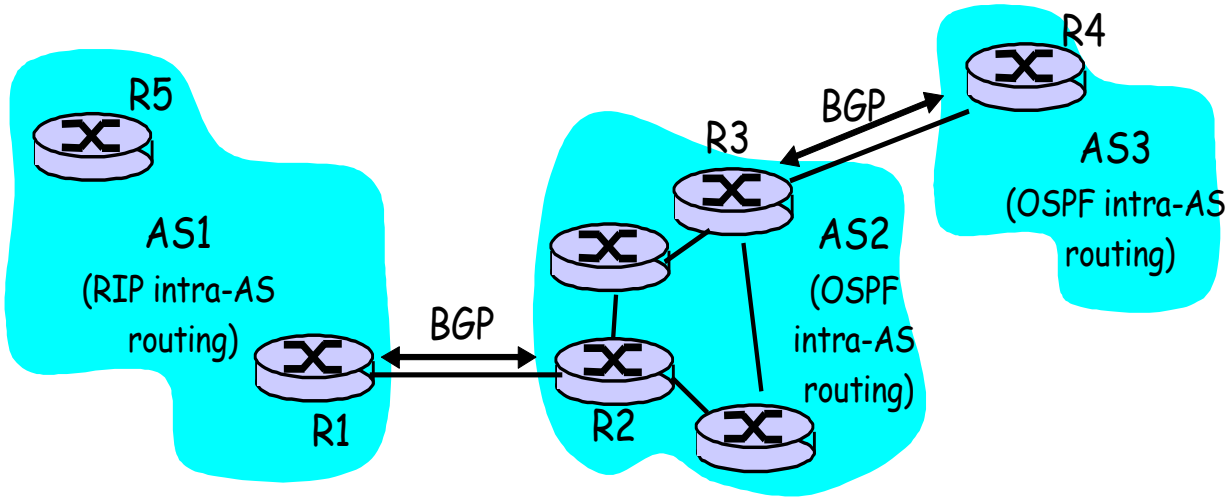


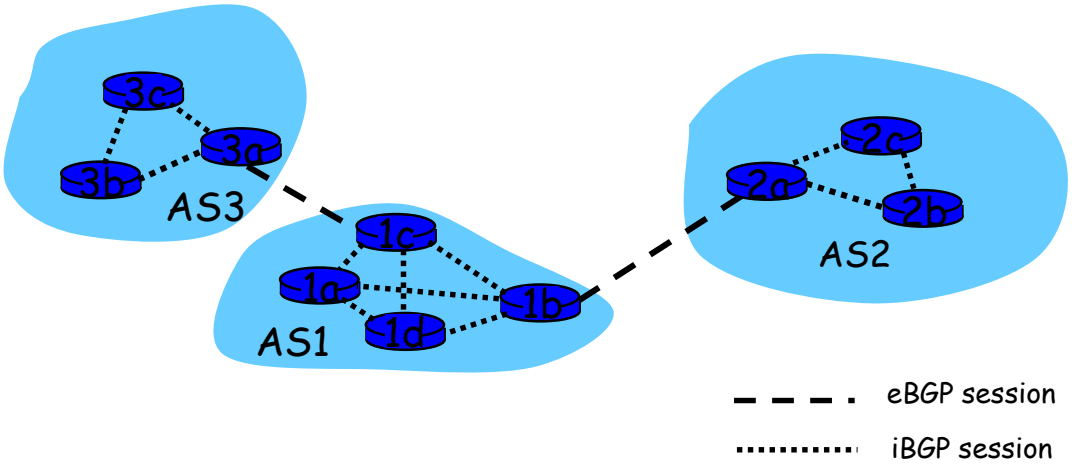
Figure 4.5.2-new2: BGP use for inter-domain routing

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate the reachability information to all routers internal to the AS.
  3. Determine "good" routes to subnets based on reachability information and policy.
- Allows a subnet to advertise its existence to rest of the Internet: *"I am here"*

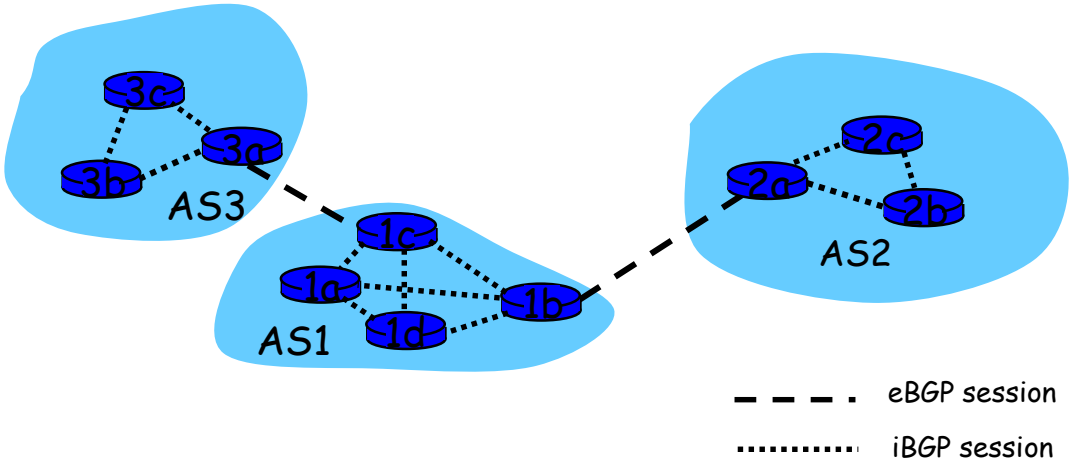
BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP conctns: BGP sessions
- Note that BGP sessions do not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



Distributing reachability info

- With eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- 1c can then use iBGP to distribute this new prefix reach info to all routers in AS1
- 1b can then re-advertise the new reach info to AS2 over the 1b-to-2a eBGP session
- When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.



Path attributes & BGP routes

- When advertising a prefix, advert includes BGP attributes.
  - prefix + attributes = “route”
- Two important attributes:
  - AS-PATH: contains the ASs through which the advert for the prefix passed: AS 67 AS 17
  - NEXT-HOP: Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)
- When gateway router receives route advert, uses import policy to accept/decline.

BGP route selection

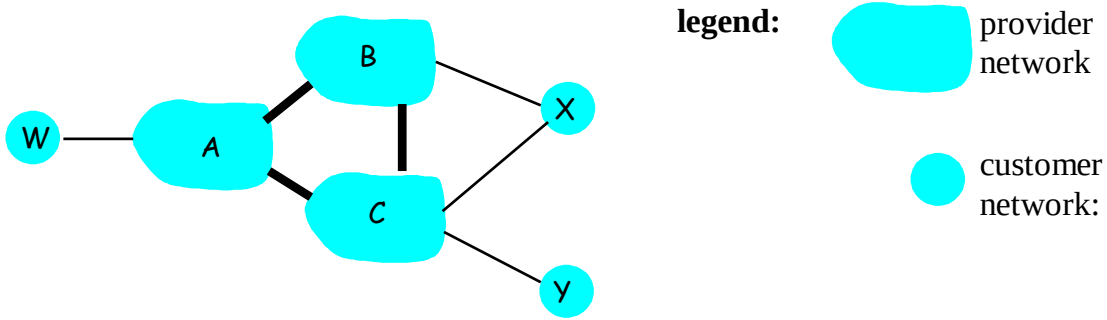
- Router may learn about more than 1 route to some prefix. Router must select route.
- Elimination rules:

  1. Local preference value attribute: policy decision
  2. Shortest AS-PATH
  3. Closest NEXT-HOP router: hot potato routing
  4. Additional criteria

BGP messages

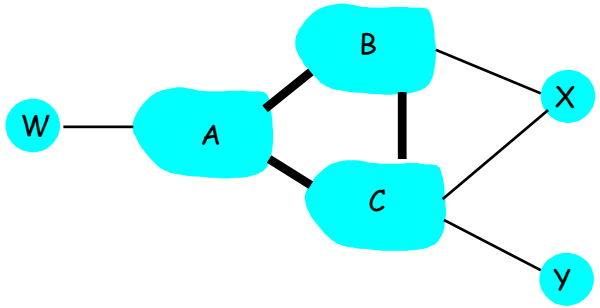
- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

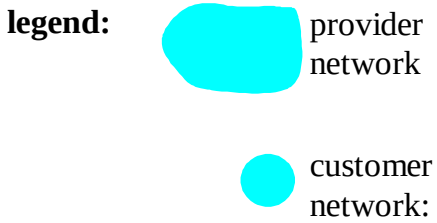
BGP routing policy



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2)





- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Why different Intra- and Inter-AS routing ?

Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:

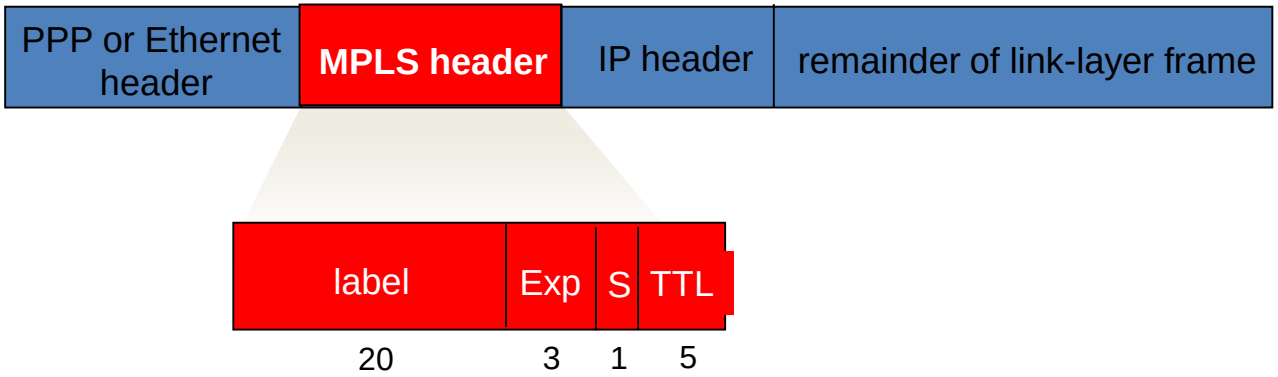
- hierarchical routing saves table size, reduced update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

Multi-Protocol Label Switching (MPLS)

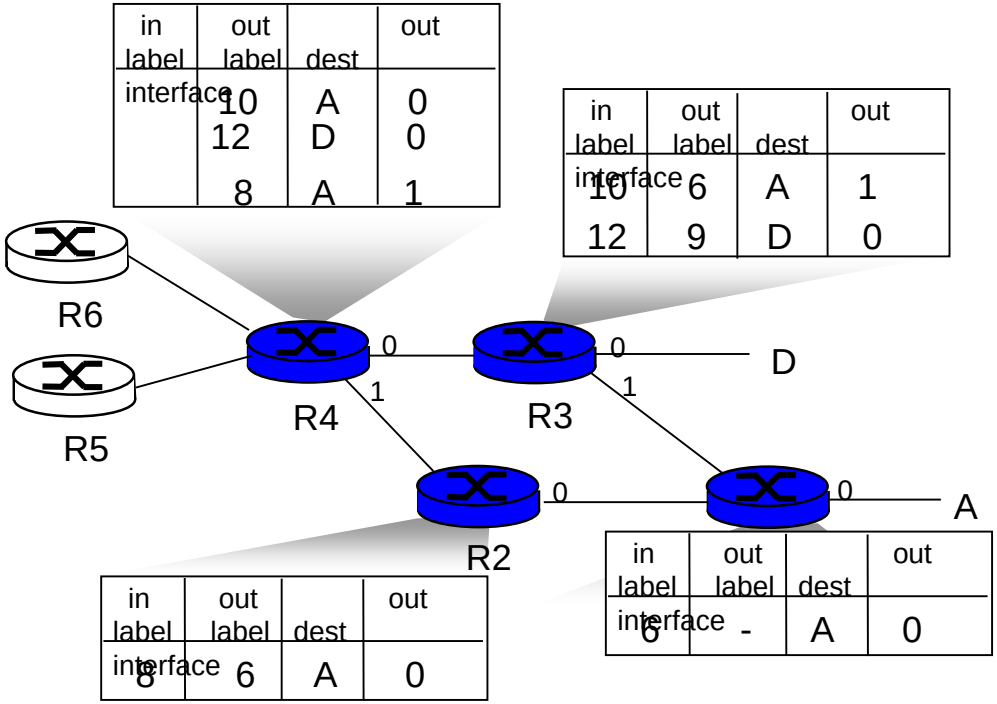
- initial goal: speed up IP forwarding by using fixed length label (instead of IP address) to do forwarding
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!



MPLS Capable Routers

- a.k.a. label-switched router
- forwards packets to outgoing interface based only on label value (don't inspect IP address)
  - MPLS forwarding table distinct from IP forwarding tables
- signaling protocol needed to set up forwarding
  - RSVP-TE, LDP
  - forwarding possible along paths that IP alone would not allow (e.g., least cost path routing) !!
  - use MPLS for traffic engineering
- must co-exist with IP-only routers

MPLS Forwarding Tables



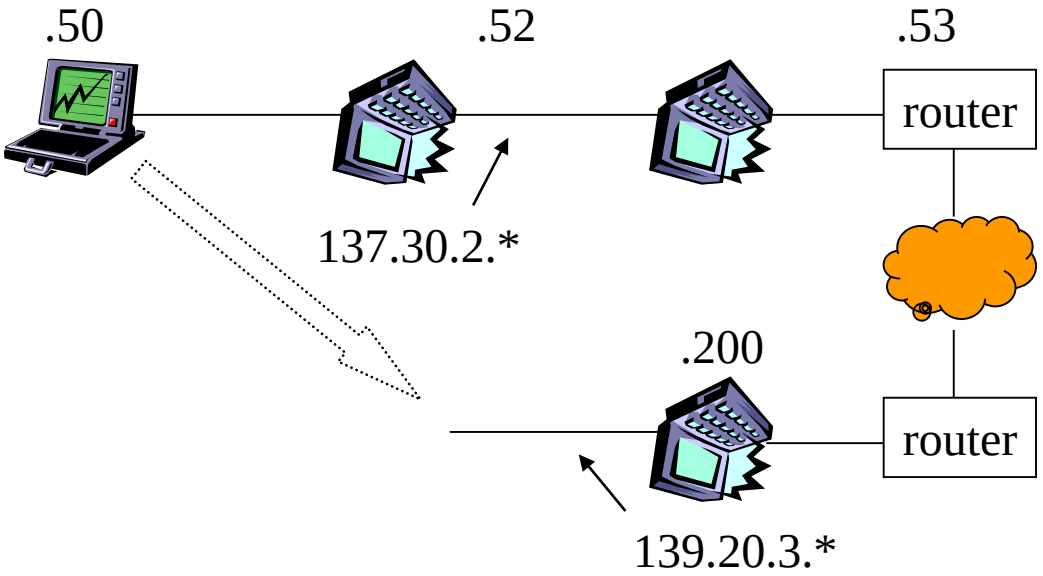
Why Mobile IP?

- Need a protocol which allows network connectivity across host movement
- Protocol to enable mobility must not require massive changes to router software, etc.
- Must be compatible with large installed base of IPv4 networks/hosts
- Confine changes to mobile hosts and a few support hosts which enable mobility

Internet Protocol (IP)

- Network layer, "best-effort" packet delivery
- Supports UDP and TCP (transport layer protocols)
- IP host addresses consist of two parts
  - **network id** + **host id**
- By design, IP host address is tied to home network address
  - Hosts are assumed to be wired, immobile
  - Intermediate routers look only at network address
  - Mobility without a change in IP address results in un-route-able packets

IP Routing Breaks Under Mobility



Why this hierarchical approach? Answer: **Scalability!**
Millions of network addresses, billions of hosts!

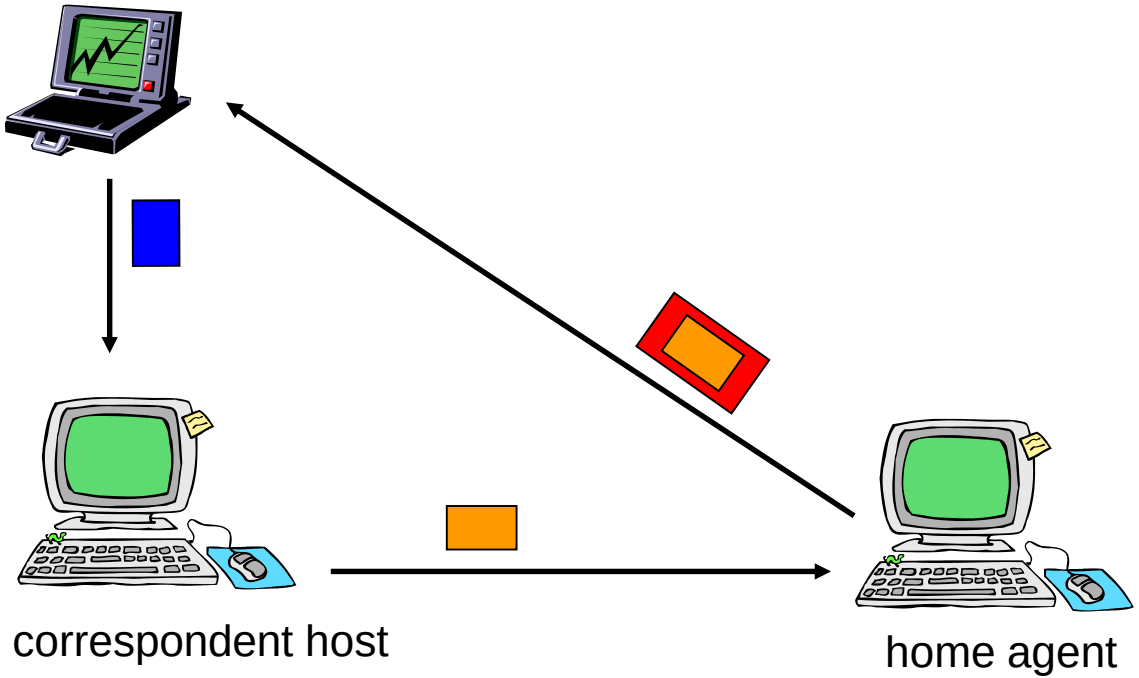
Mobile IP: Basics

- Proposed by IETF (Internet Engineering Task Force)
  - Standards development body for the Internet
- Mobile IP allows a mobile host to move about without changing its ***permanent*** IP address
- Each mobile host has a ***home agent*** on its ***home network***
- Mobile host establishes a ***care-of*** address when it's away from home

Mobile IP: Basics, Cont.

- ***Correspondent host*** is a host that wants to send packets to the mobile host
- Correspondent host sends packets to the mobile host's IP permanent address
- These packets are routed to the mobile host's home network
- Home agent forwards IP packets for mobile host to current care-of address
- Mobile host sends packets directly to correspondent, using permanent home IP as source IP

Mobile IP: Basics, Cont.

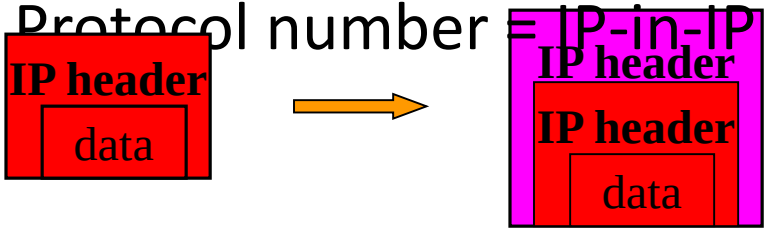


Mobile IP: Care-of Addresses

- Whenever a mobile host connects to a remote network, two choices:
  - care-of can be the address of a ***foreign agent*** on the remote network
    - foreign agent delivers packets forwarded from home agent to mobile host
  - care-of can be a temporary, foreign IP address obtained through, e.g., DHCP
    - home agent ***tunnels*** packets directly to the temporary IP address
- Regardless, care-of address must be ***registered*** with home agent

IP-in-IP Tunneling

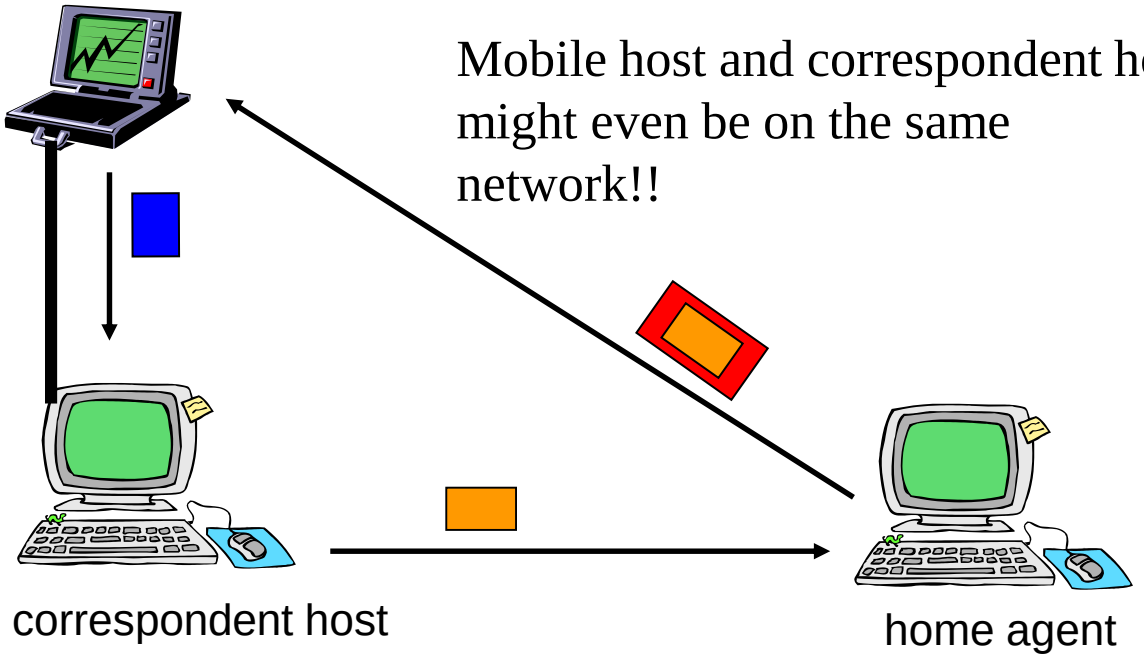
- Packet to be forwarded is encapsulated in a new IP packet
- In the new header:
  - Destination = care-of-address
  - Source = address of home agent
  - Protocol number = IP-in-IP



At the Other End...

- Depending on type of care-of address:
  - Foreign agent or
  - Mobile host
- … strips outer IP header of tunneled packet, which is then fed to the mobile host
- Aside: Any thoughts on advantages of foreign agent vs. co-located (foreign IP) address?

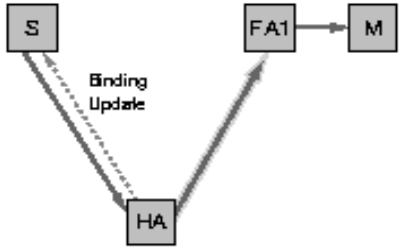
Routing Inefficiency



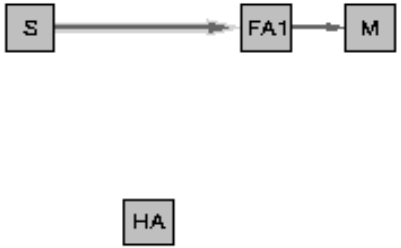
Route Optimizations

- Possible Solution:
  - Home agent sends current care-of address to correspondent host
  - Correspondent host caches care-of address
  - Future packets tunneled directly to care-of address
- But!
  - An instance of the cache consistency problem arises...
  - Cached care-of address becomes stale when the mobile host moves
  - Potential security issues with providing care-of address to correspondent

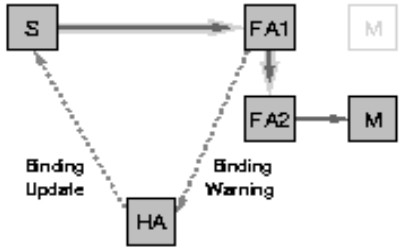
Possible Route Optimization



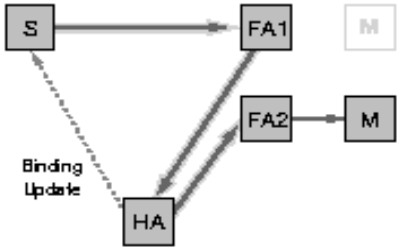
(a) Sending the first packet to a mobile host



(b) Sending subsequent packets to a mobile host



(c) Sending the first packet after a mobile host moves



(d) Tunneling the packet in case the the cache entry has been dropped