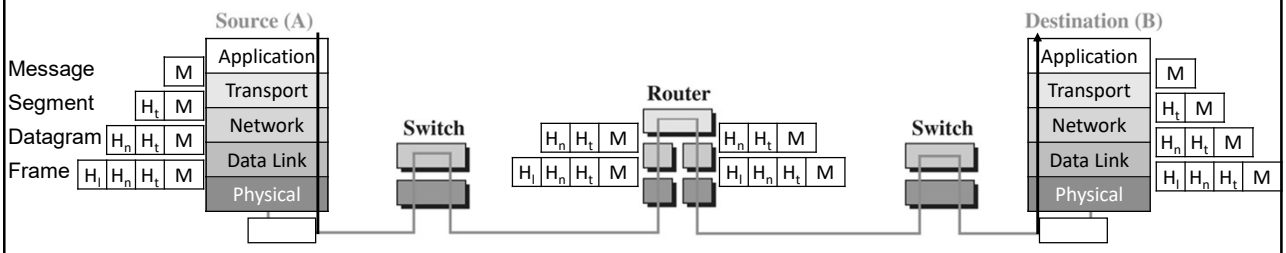
Welcome!

ELEC 8560 – Computer Networks

Network Layer: Control Plane

1

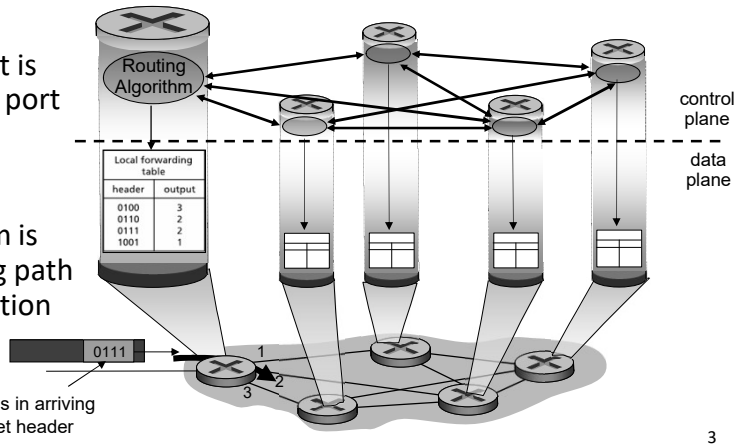
Recall: End-to-End Communication via Internet



2

Network Layer: Data Plane and Control Plane

- Network layer has two key functions: routing and forwarding
- Data plane
  - Local, per-router function
  - Determines how datagram arriving on router input port is forwarded to router output port
- Control plane
  - Network-wide logic
  - Determines how a datagram is routed among routers along path from source host to destination host (routing algorithms)



Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting

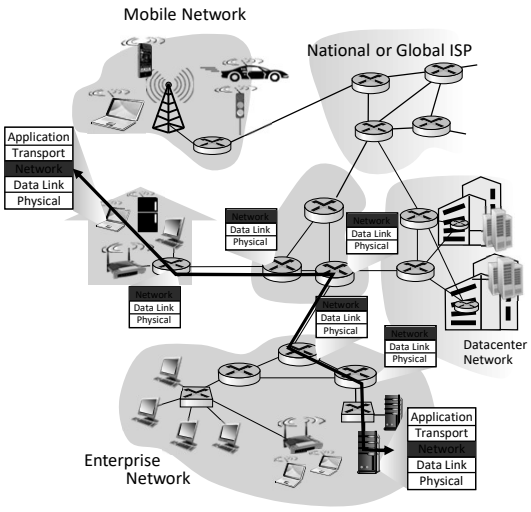
- Recommended reading: Forouzan – Chapter 8

Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting

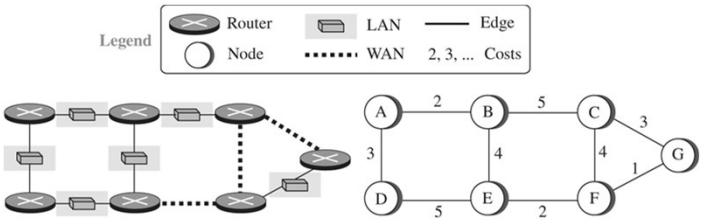
Unicast Routing

- In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables
- Only the routers that glue together the networks in the internet need forwarding tables
  - Source host delivers its packet to the default router in its local network
  - Destination host receives the packet from its default router in its local network



An Internet as a Graph

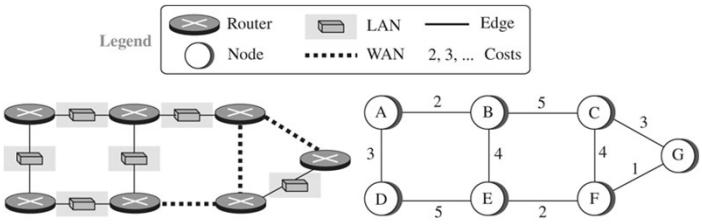
- To find the best route, an internet can be modeled as a graph
  - Best means least cost, fastest, least congested, etc.
- A graph is a set of nodes and edges (lines) that connect the nodes
- To model an internet as a graph, make each router as a node and each network between a pair of routers as an edge
- An internet is modeled as a weighted graph, in which each edge is associated with a cost defined by network operator



a. An internet b. The weighted graph

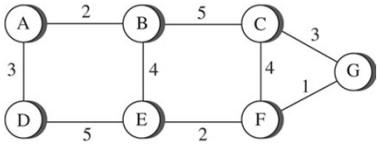
Least-Cost Routing

- One of the ways to interpret the best route from the source router to the destination router is to find the least cost between the two
  - That is, the source router chooses a route to the destination router in such a way that the total cost for the route is the least cost among all possible routes
  - For example, path A-B-E is the best route between A and E
- Hence, each router needs to find least-cost route to all other routers

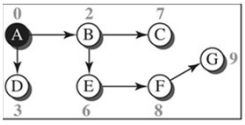


a. An internet b. The weighted graph

Least-Cost Trees

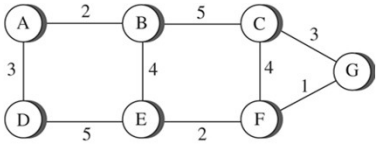


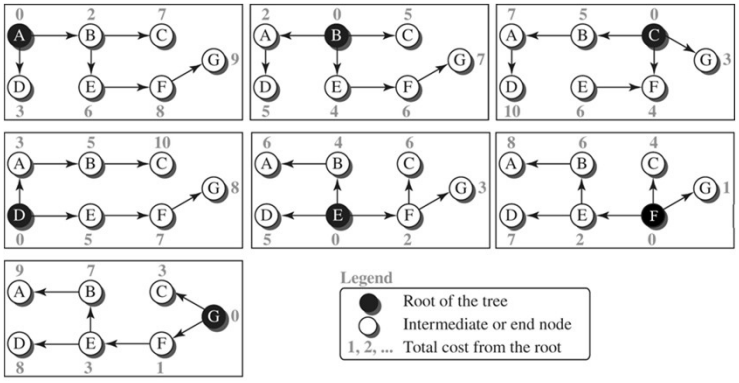
- If there are *N* routers in an internet, there are *N - 1* least-cost paths from each router to any other router
- Thus, we need *N × (N - 1)* least-cost paths for the whole internet
  - For example, if we have 8 routers in an internet, we need 56 least-cost paths
- Least-cost trees are a better way to see all paths:
  - A tree with the source router as the root that spans the whole graph (visits all other nodes) and in which the path between the root and any other node is the shortest
  - That is *N* trees, only one shortest-path tree for each node



Example: Least-Cost Trees

Show the least-cost tree for all nodes in the internet below.





Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting

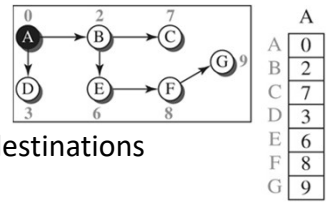
Routing Algorithms

- Goal: determine a path (route) with the least cost from sending host to receiving host, through a network of routers
  - Route is a sequence of routers packets traverse from given initial source host to final destination host
- Routing algorithms differ in the way they interpret the cost and the way they create the least-cost tree for each node
  - Example:
    - Distance-vector routing
    - Link-state routing
    - Path-vector routing

Outline

- Internet as a graph
- Routing algorithms
  - Distance-vector routing
- Routing protocols
- Multicasting

Distance-Vector Routing



	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

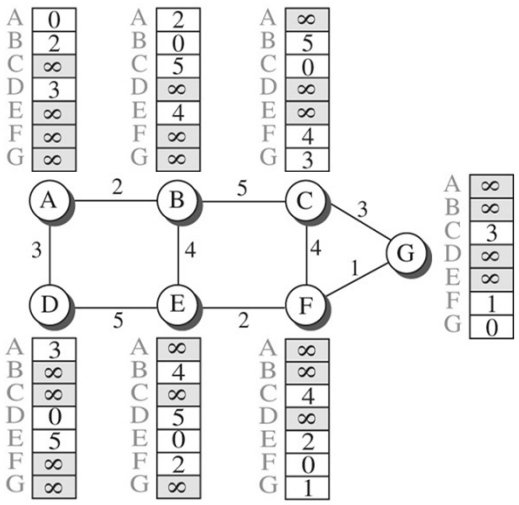
- A distance vector (DV) is a 1D array to represent a least-cost tree of a node
  - i.e., minimum distance between itself and all possible destinations
- In DV routing,
  - Each node first creates is its own least-cost tree with the initial information it has about its immediate neighbors
  - Incomplete trees are exchanged between immediate neighbors to make trees more and more complete to represent the whole internet

A router periodically (and upon changes) tells all of its neighbors what it knows about the whole internet (i.e., its routing table)

Example: Distance Vectors

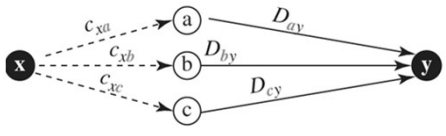
Show the initial distance vectors for the internet below.

Solution:



Bellman-Ford Equation

- The heart of DV routing
- Finds the cost of least-cost (shortest-distance) path between a source node *x* and a destination node *y*, through some intermediary nodes *(a, b, c, …)*
- When the costs between the source and the intermediary nodes *(c_{xa}, c_{xb}, …)* and the least costs between the intermediary nodes and the destination *(D_{ay}, D_{by}, …)* are given



$$D_{xy} = min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

Example: Bellman-Ford Equation

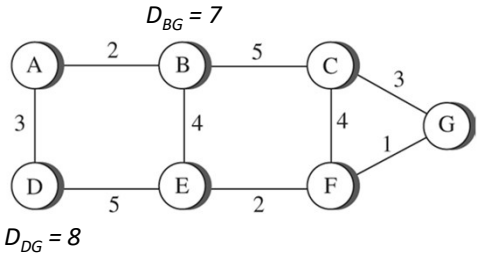
Use Bellman-Ford equation to find the shortest distance path from A to G.

Solution:

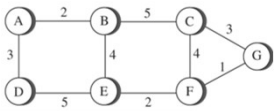
- Cost of least-cost path is

$$D_{AG} = min\{(c_{AB} + D_{BG}), (c_{AC} + D_{CG})\}$$
$$= min\{(2+7), (3+8)\}$$
$$= 9$$

- Then, node B is the next hop on the estimated least-cost path to destination G



Example: Updating Distance Vectors



Show the updated distance vectors for the same internet when B receives a copy of A's vector, then receives a copy of E's vector.

Solution:

New B			Old B	A
A	2	-	2	0
B	0	-	0	2
C	5	-	5	∞
D	5	A	∞	3
E	4	-	4	∞
F	∞	-	∞	∞
G	∞	-	∞	∞

B[ ] = min (B[ ] , 2 + A[ ])

a. First event: B receives a copy of A's vector.

New B			Old B	E
A	2	-	2	∞
B	0	-	0	4
C	5	-	5	∞
D	5	A	5	5
E	4	-	4	0
F	6	E	∞	2
G	∞	-	∞	∞

B[ ] = min (B[ ] , 4 + E[ ])

b. Second event: B receives a copy of E's vector.

Note:
X[ ]: the whole vector

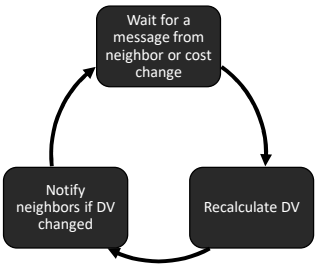
Distance-Vector Routing Algorithm

- Key idea:
  - Report: from time-to-time, each node sends its own distance vector estimate to neighbors
  - Update: when a node receives a new distance vector estimate from any neighbor
    - updates its own vector using Bellman-Ford equation
    - triggers a report to neighbors

Distance-Vector Routing Algorithm (cont.)

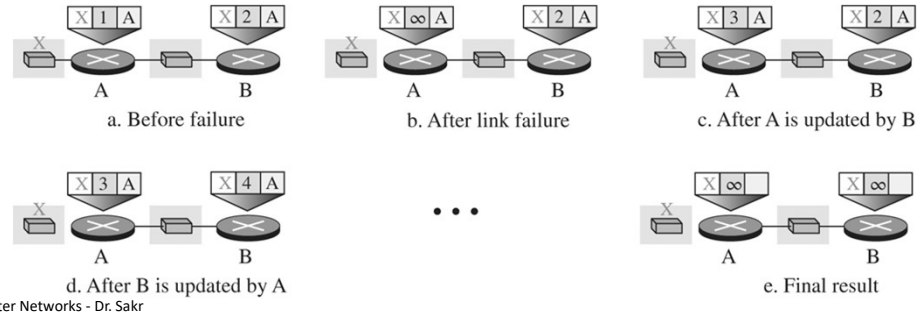
- The algorithm is run by its node independently and asynchronously

```
1   Distance_Vector_Routing ( )
2     // Initialize (create initial vectors for the node)
3     D[myself] = 0
4     for (y = 1 to N)
5       if (y is a neighbor)
6         D[y] = c[myself][y]
7       Else
8         D[y] = ∞
9     send vector {D[1], D[2], …, D[N]} to all neighbors
10    // Update (improve the vector with the vector received from a neighbor)
11    repeat (forever)
12      wait (for a vector D_w from a neighbor w or any change in the link)
13      for (y = 1 to N)
14        D[y] = min [D[y], (c[myself ][w] + D_w[y])] // Bellman-Ford equation
15      if (any change in the vector)
16        send vector {D[1], D[2], …, D[N]} to all neighbors
```



Count to Infinity: Two-node Instability

- Scenario:
  - A-X link fails, A updates its table
  - B sends its forwarding table to A before receiving A's table
  - A assumes B found a way to reach X, updates table, and send update
  - B thinks something change , updates table, and send update
  - …



Outline

- Internet as a graph
- Routing algorithms
  - Link-state routing
- Routing protocols
- Multicasting

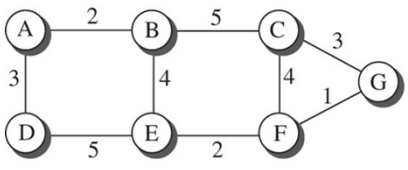
Link-State Routing

- Link-state (LS) defines the characteristic of a link (an edge) that represents a network in the internet
- In link-state routing, the cost associated with an edge defines the state (i.e., connectivity) of the link
  - Links with lower costs are preferred to links with higher costs
  - If the cost of a link is infinity, it means that link does not exist or broken
- To create a least-cost tree with this method, each node needs to have a complete map of the network to know the state of each link

A router periodically (and upon changes) tells the whole internet what it knows only about its neighbors

Link-State Database (LSDB)

- In LS routing, each router has information about the complete network topology
- The collection of states for all links is called LSDB
  - A 2D array where the value of each cell defines the cost of corresponding link
- There is only one LSDB for the whole internet
  - Each node needs to have a duplicate of it to create the least-cost tree



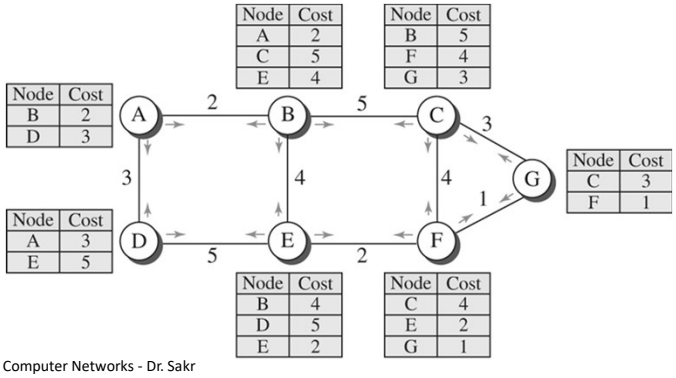
a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Link State Advertisement

- Each node periodically (and in case of connectivity changes) sends a short LS packet containing its identity and cost of links to neighbors
  - This called flooding: nodes broadcast LS packets about immediate neighbors to all interfaces to build LSDB
  - If new info received, update and send copy to all interfaces except the source



	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Link-State Routing Algorithm

- To form a least-cost tree for itself, each node independently use the shared LSDB and run the famous Dijkstra's Algorithm (pronounced "dyke-stra")
- Iterative algorithm uses the following steps:
  - The node chooses itself as the root of the tree, creates a tree with a single node, and sets the total cost of each node based on information in the LSDB
  - The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree
  - The node repeats step 2 until all nodes are added to the tree

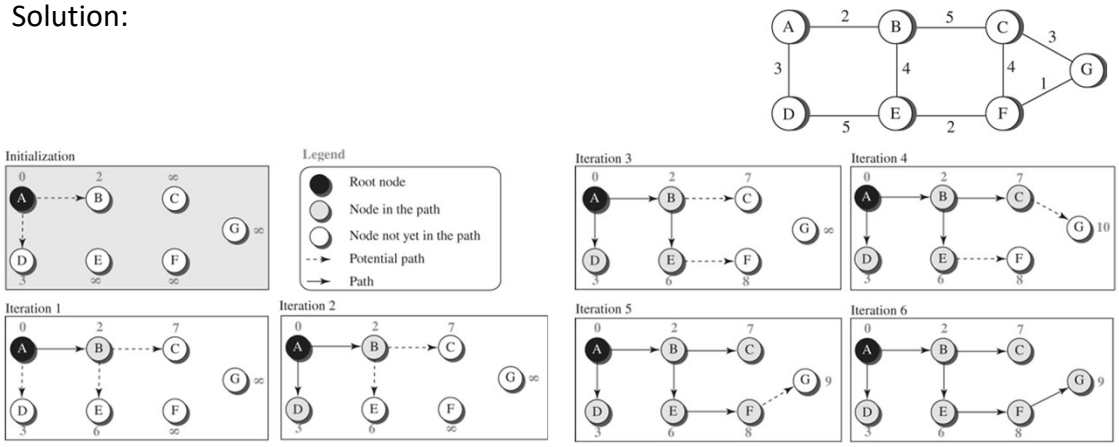
Link-State Routing Algorithm (cont.)

```
1   Link_State_Routing ( )
2     // Initialization
3     Tree = {root}       // Tree is made only of the root
4     for (y = 1 to N)    // N is the number of nodes
5       if (y is the root)
6         D [y] = 0       // D [y] is shortest distance from root to node y
7       else if (y is a neighbor)
8         D [y] = c[root][y]      // c [x] [y] is cost between nodes x and y in LSDB
9       else
10        D [y] = ∞
11    // Calculation
12    repeat (until (all nodes included in the Tree))
13      find a node w, with D[w] minimum among all nodes not in the Tree
14      Tree = Tree ∪ {w}     // Add w to tree
15      // Update distances for all neighbor of w
16      for (every node x, which is neighbor of w and not in the Tree)
17        D[x] = min{D[x], (D[w] + c[w][x])}
```


Example: Dijkstra's Algorithm

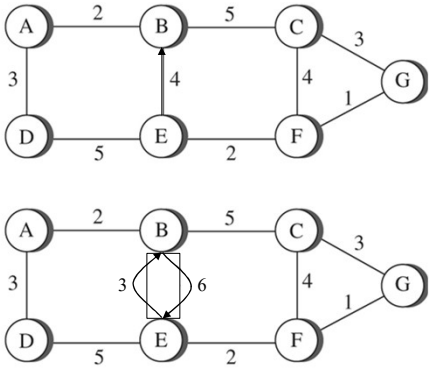
Show the least-cost tree for node A using Dijkstra' Algorithm.

Solution:



Notes on the Dijkstra's Algorithm

- The direction of the edge (arrow) is important if given



Outline

- Internet as a graph
- Routing algorithms
  - Path-vector routing
- Routing protocols
- Multicasting

Path-Vector Routing

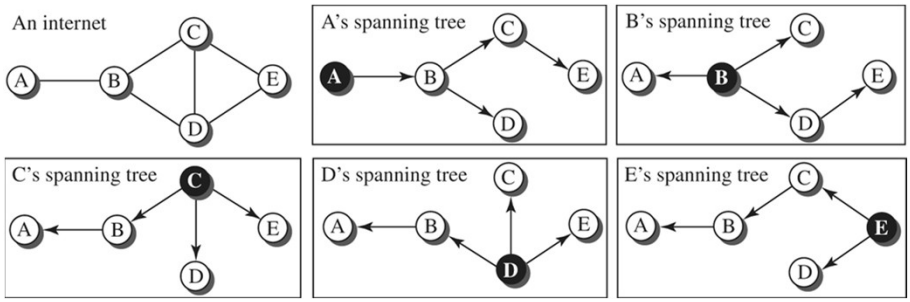
- In some cases, selecting the path with the least cost is not the priority
  - For example, a sender wants to prevent its packets from going through some routers in the internet
- Both link-state and distance-vector routing are based on the least-cost goal and do not allow a sender to apply specific policies to the route a packet may take
- In path-vector (PV) routing, routing table includes destination network, next router, and the path to reach destination

Path-Vector Routing (cont.)

- Path from a source to all destinations is determined by the best spanning tree
  - Best spanning tree, however, is not the least-cost tree
  - It is the tree determined by the source when it imposes its own policy
- If there is more than one route to a destination, the source can choose the route that meets its policy best
- A source may apply several policies at the same time
  - One of the common policies uses the minimum number of nodes to be visited
  - Another common policy is to avoid some nodes as the middle node in a route

Example: Spanning Trees

- Each source creates its own policy, which is to use minimum number of nodes to reach destination in this case
  - A and E selected spanning trees such that data does not go through D
  - B selected spanning trees such that data does not go through C



Creation of Spanning Tree

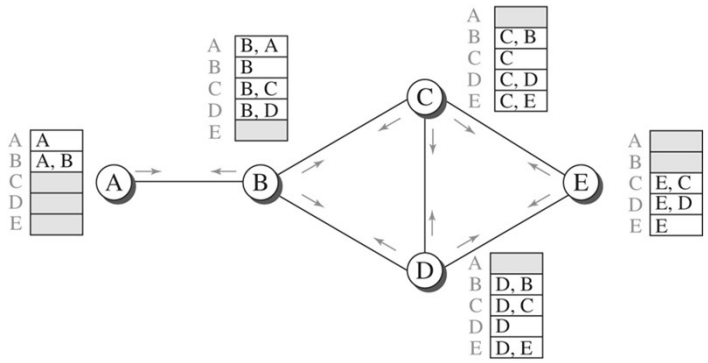
- Path-vector routing is an asynchronous and distributed routing algorithm (similar to distance-vector routing)
- Spanning trees are made, gradually and asynchronously, by each node
- When a node is booted, it creates a path vector based on the information it obtains about its immediate neighbor
  - A node sends greeting messages to its immediate neighbors to collect these pieces of information

Example: Path Vectors

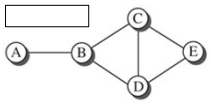
Show the initial path vectors for the internet below assume minimum number of hops.

Solution:

- Arrows show how path vectors are sent to immediate neighbors by each node



Example: Updating Path Vectors



Show the updated path vectors for the same internet when C receives a copy of B's path vector, then receives a copy of D's path vector.

Solution:

- Operator (+) means add *x* at the beginning of path

	New C	Old C	B
A	C, B, A		B, A
B	C, B	C, B	B
C	C	C	B, C
D	C, D	C, D	B, D
E	C, E	C, E	

C[ ] = best (C[ ], C + B[ ])

Event 1: C receives a copy of B's vector

	New C	Old C	D
A	C, B, A	C, B, A	
B	C, B	C, B	D, B
C	C	C	D, C
D	C, D	C, D	D
E	C, E	C, E	D, E

C[ ] = best (C[ ], C + D[ ])

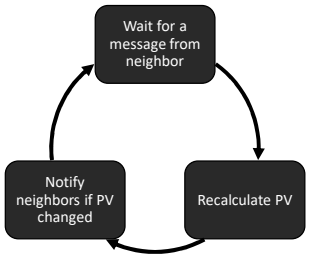
Event 2: C receives a copy of D's vector

Note:
X [ ]: vector X
Y: node Y

Path-Vector Routing Algorithm

- The algorithm is run by its node independently and asynchronously

```
1   Path-vector algorithm for a node
2     // Initialization
3     for (y = 1 to N)
4       if ( y is myself)
5         Path[ y] = myself
6       else if (y is a neighbor)
7         Path[ y] = myself + neighbor node
8       else
9         Path[ y] = empty
10    Send vector {Path[1], Path[2], …, Path[ y]} to all neighbors
11    // Update
12    repeat (forever)
13      wait (for a vector Path_w from a neighbor w)
14      for ( y = 1 to N)
15      if (Path_w includes myself)
16        discard the path                    // Avoid any loop
17      else
18        Path[ y] = best {Path[ y], (myself + Path_w[ y])}
19    If (there is a change in the vector)
20      Send vector {Path[1], Path[2], …, Path[ y]} to all neighbors
```

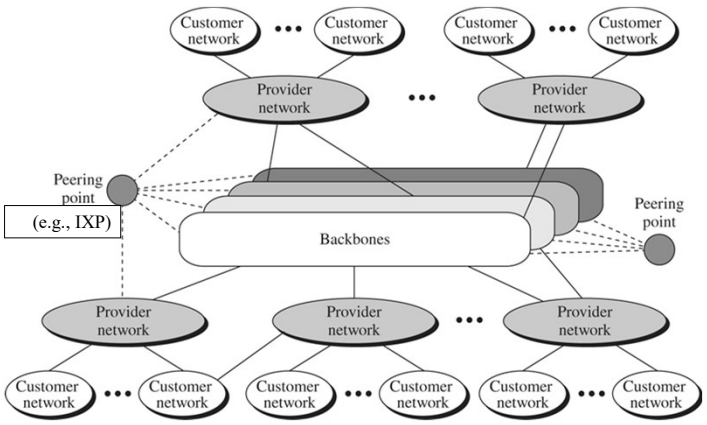


Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting

Internet Structure

- Internet today is a multi-backbone structure run by different private corporations
- It is made of a huge number of networks and routers that connect them

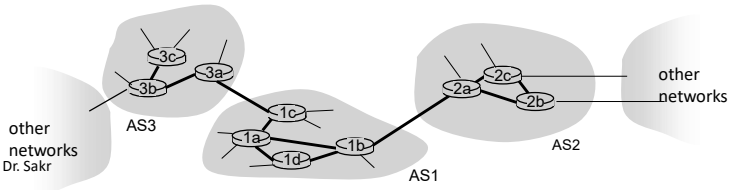


Routing in the Internet

- So far, routing algorithms are idealized (e.g., all routers are identical)
- In practice, routing cannot be done using one single protocol for two reasons:
  - Scalability: billions of destinations
    - Size of forwarding tables becomes huge
    - Searching for destinations becomes time-consuming
    - Updating tables creates a huge amount of traffic
  - Administrative: a network of networks
    - Each ISP is run by an administrative authority that may want to control routing in its own network

Hierarchical Routing: How to Make Routing Scalable

- Aggregate routers into regions known as autonomous systems (AS)
  - Each AS is given an autonomous number (ASN) by the ICANN
  - Each ISP is an AS that manages networks and routers under its control
- Intra-ISP: routing within same the AS
  - All routers in an AS run same intra-domain protocol
  - Routers in different AS's can run different intra-domain routing protocols
- Inter-ISP: routing among AS's
  - Gateway router: at the edge of each AS, has link(s) to router(s) in other AS's
  - Gateways perform inter-domain routing (as well as intra-domain routing)



Unicast Routing Protocols

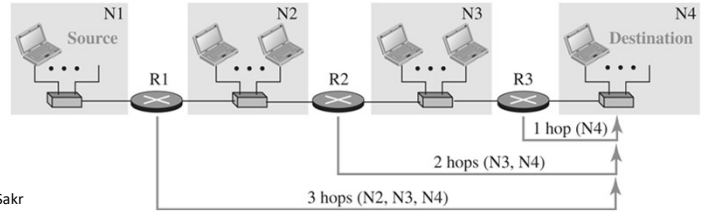
- We will discuss three common protocols used in the Internet:
  - Routing Information Protocol (RIP): [RFC 2453]
    - Intra-ISP
    - Based on the distance-vector algorithm
    - No longer widely used
  - Open Shortest Path First (OSPF): [RFC 2328]
    - Intra-ISP
    - Based on the link-state algorithm
  - Border Gateway Protocol (BGP): [RFC 4271]
    - Inter-ISP
    - Based on the path-vector algorithm

Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
  - RIP
- Multicasting

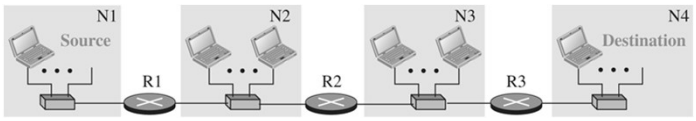
Routing Information Protocol (RIP)

- RIP is one of the oldest intra-domain routing protocols
  - Easy to configure
  - Not the preferred routing protocol: time to converge and scalability are poor
  - DV-based protocol
- RIP uses hop count as the routing metric (i.e., cost)
  - number of networks that need to be passed to reach the destination network (not host)
  - cost is defined between a router and the network in which destination host is located
  - network of the source host is not counted (packets delivered to default router)
  - Maximum hop count is 15 (i.e., 16 means no connection)



Forwarding Table

- Contains the following:
  - Address of destination network
  - Address of next router
  - Cost (needed to update forwarding tables)



Forwarding table for R1

Destination network	Next router	Cost in hops
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

Forwarding table for R2

Destination network	Next router	Cost in hops
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

Forwarding table for R3

Destination network	Next router	Cost in hops
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

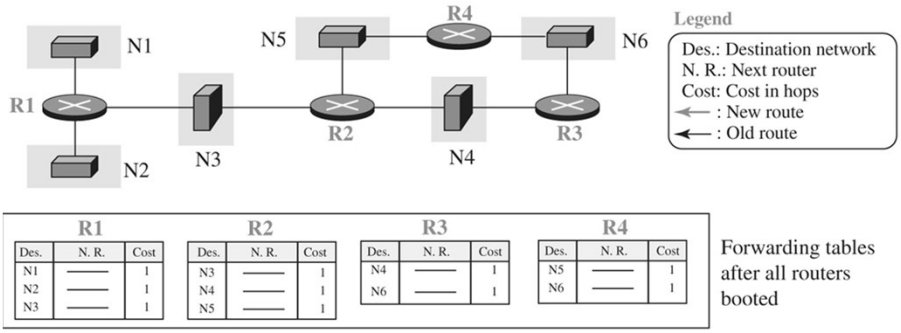
RIP Algorithm

- RIP protocol works as follows:
  - Each router initializes its forwarding table with a list of locally connected networks
  - Periodically, each router advertises its table over all RIP-enabled interfaces
  - When a router receives an update, it merges it with its table:
    - Adds one hop to each cost and change next router address to that of the sending router
    - If route does not exist in the old table, add
    - If route exists in the old table but the cost is higher, update
    - If route exists in the old table, the cost is lower, but the same next router, update (destinations may become unreachable over time)
    - Sort forwarding table using longest prefix first and begin using it to forward packets
  - If a router does not receive advertisements for a remote route for some time, it eventually times out that route and stops forwarding packets over it
    - Every network connected to every router eventually becomes known to all routers

Example: Initial RIP Forwarding Tables

Show the initial RIP forwarding tables after all routers are booted in the AS below.

Solution:



R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1

R2

Des.	N. R.	Cost
N3	—	1
N4	—	1
N5	—	1

R3

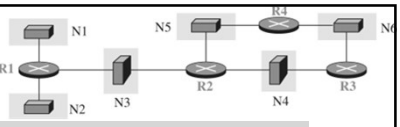
Des.	N. R.	Cost
N4	—	1
N6	—	1

R4

Des.	N. R.	Cost
N5	—	1
N6	—	1

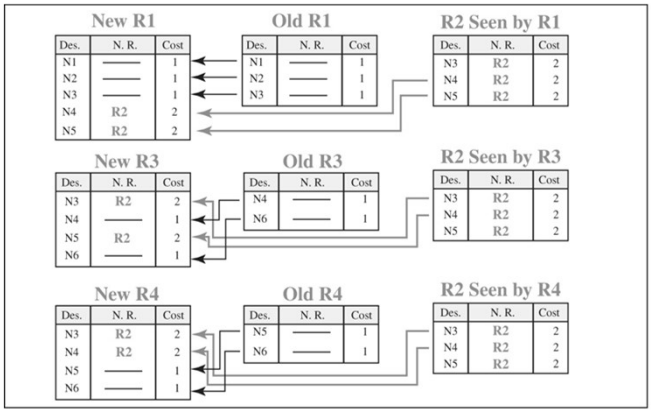
Forwarding tables after all routers booted

Example: Updating RIP Forwarding Tables



Show the changes in RIP forwarding tables of R1, R3, and R4 when they receive a copy of R2's table for the same AS.

Solution:



New R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1
N4	R2	2
N5	R2	2

Old R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1

R2 Seen by R1

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

New R3

Des.	N. R.	Cost
N3	R2	2
N4	—	1
N5	R2	2
N6	—	1

Old R3

Des.	N. R.	Cost
N4	—	1
N6	—	1

R2 Seen by R3

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

New R4

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	—	1
N6	—	1

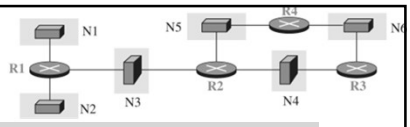
Old R4

Des.	N. R.	Cost
N5	—	1
N6	—	1

R2 Seen by R4

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

Example: Final RIP Forwarding Tables



Show the stabilized forwarding tables for all routers when there is no more change for the same AS.

Solution:

Final R1

Des.	N. R.	Cost
N1	——	1
N2	——	1
N3	——	1
N4	R2	2
N5	R2	2
N6	R2	3

Final R2

Des.	N. R.	Cost
N1	R1	2
N2	R1	2
N3	——	1
N4	——	1
N5	——	1
N6	R3	2

Final R3

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	——	1
N5	R2	2
N6	——	1

Final R4

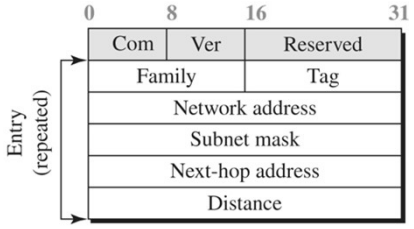
Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	R2	2
N5	——	1
N6	——	1

RIP Timers

- RIP uses three timers:
  - Periodic timer:
    - Regular advertisement
    - Randomly chosen between 25 to 35 sec
  - Expiration timer:
    - Validity of a route
    - Set to 180 sec when a router is updated
    - Route is considered expired and cost is set to 16 when timer reaches 0
      - Route is not purged yet and keep being advertised
  - Garbage collection timer:
    - Starts when a route expires
    - Set to 120 sec for that route
    - Route is purged when timer reached 0

RIP Messages

- RIP has two types of messages
  - Request: sent by a router that has just booted or has some expired entries
  - Response: update messages
    - Unsolicited:
      - Periodic updates every 25-35 sec
      - When there is a change in forwarding table
    - Solicited:
      - Answer a request message



Fields

Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

Performance of RIP

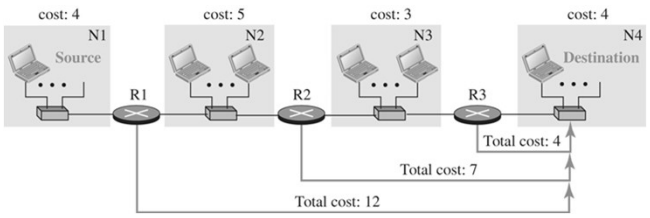
- Update Messages
  - Update messages in RIP have a very simple format
  - Only sent to neighbors; they are local
  - Do not normally create traffic or use a lot of bandwidth
- Convergence of Forwarding Tables
  - Converges slowly if the domain is large; however, by allowing only 15 hops, convergence is not a problem
  - Suffers from count-to-infinity and loops issues
- Robustness
  - Calculating tables is based on information received from immediate neighbors
  - If there is a failure or corruption in one router, the problem will propagate to all routers

Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
  - OSPF
- Multicasting

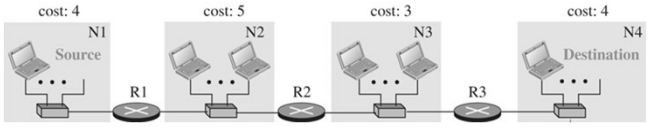
Open Shortest Path First (OSPF)

- An open protocol, which means the specification is a public document
- Intra-domain routing protocol like RIP, but based on LS routing protocol
- Cost of reaching a destination from the host is calculated from the source router to the destination network, similar to RIP
- Each link (network) can be assigned a weight based on throughput, round-trip time, reliability, hop count, etc.



Forwarding Table

- Each OSPF router creates a forwarding table using Dijkstra's algorithm to find the shortest-path tree between itself and the destination
- Contains the following:
  - Address of destination network
  - Address of next router
  - Cost (needed to update forwarding tables)



Forwarding table for R1

Destination network	Next router	Cost
N1	—	4
N2	—	5
N3	R2	8
N4	R2	12

Forwarding table for R2

Destination network	Next router	Cost
N1	R1	9
N2	—	5
N3	—	3
N4	R3	7

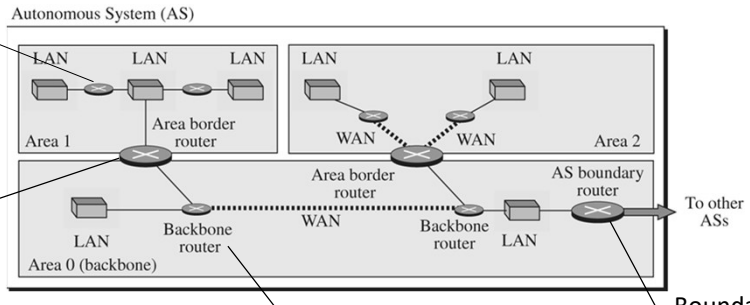
Forwarding table for R3

Destination network	Next router	Cost
N1	R2	12
N2	R2	8
N3	—	3
N4	—	4

Hierarchical OSPF: Areas in Autonomous Systems

- Two-level hierarchy: local area and backbone
  - Link-state advertisements flooded only in an area or a backbone
  - Each node has detailed area topology and only knows direction to reach other destinations

Local Routers:
- Flood LS in area only
- Compute routing within area
- Forward packets to outside via area border router



Area Border Routers:
- Summarize distances to destinations in own area
- Advertise in backbone

Backbone Router:
- Runs OSPF limited to backbone
- Flood LS in backbone only

Boundary Router:
Connects to other AS's

OSP Implementation

- OSPF protocol works as follows:
  - Each router initializes its forwarding table with a list of locally connected networks
  - Periodically, each router advertises the state of each link to all routers in the area for the formation of the LSDB
  - When a router receives an update, it uses Dijkstra's algorithm to find the shortest-path tree and create forwarding table
  - If a router does not receive advertisements for a remote route for some time, it eventually times out that route and stops forwarding packets over it
    - Every network connected to every router in the area eventually becomes known to all routers

Performance of OSPF

- Update Messages
  - Link-state messages in OSPF have a somewhat complex format
    - Five different types of messages
  - Flooded to the whole area
  - If the area is large, may create heavy traffic and use a lot of bandwidth
- Convergence of Forwarding Tables
  - When the flooding of LS packets is completed, each router can create its own shortest-path tree and forwarding table; convergence is fairly quick
  - Each router needs to run the Dijkstra's algorithm, which may take some time
- Robustness
  - OSPF protocol is more robust than RIP because, after receiving the completed LSDB, each router is independent and does not depend on other

Outline

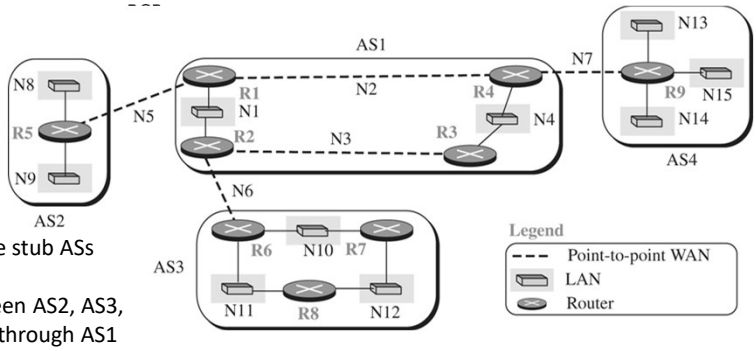
- Internet as a graph
- Routing algorithms
- Routing protocols
  - BGP
- Multicasting

Border Gateway Protocol (BGP)

- Intra-domain protocols, e.g., RIP and OSPF, enable routers to know how to reach a network within its own AS, routers do not know how to reach a network in another AS
- BGP is the de facto inter-domain routing protocol used in the Internet today
  - "Glue that hold the Internet together"
- Based on the path-vector algorithm to provide information about the reachability of networks in the Internet
  - allows subnet to advertise its existence, and the destinations it can reach, to the rest of Internet: *"I am here, here is who I can reach, and how"*
- Current version is BGP version 4 (BGP4) is a complex protocol
  - Point-to-point communication

eBGP and iBGP Connections

- BGP provides each AS a means to:
  - External BGP (eBGP):
    - Installed on border routers, obtain subnet reachability information from neighboring AS's
  - Internal BGP (iBGP):
    - Installed on all routers, propagate reachability information to all AS-internal routers



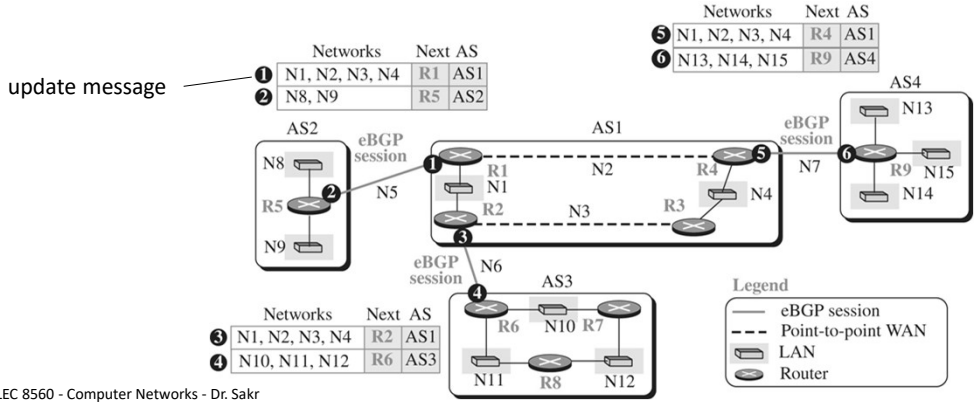
- AS2, AS3, and AS4 are stub ASs
- AS1 is a transient AS
- Data exchange between AS2, AS3, and AS4 should pass through AS1

Operation of eBGP

- BGP session: two BGP routers (called peers or speakers) continuously exchange BGP messages over TCP connection using port number 179 (later)
- eBGP variation allows two physically connected border routers in two different AS's to form pairs and exchange messages
  - Example: R1-R5 over WAN N5, R2-R6 over WAN N6, and R4-R9 over WAN N7
  - A logical connection (called session) is created over the physical connection

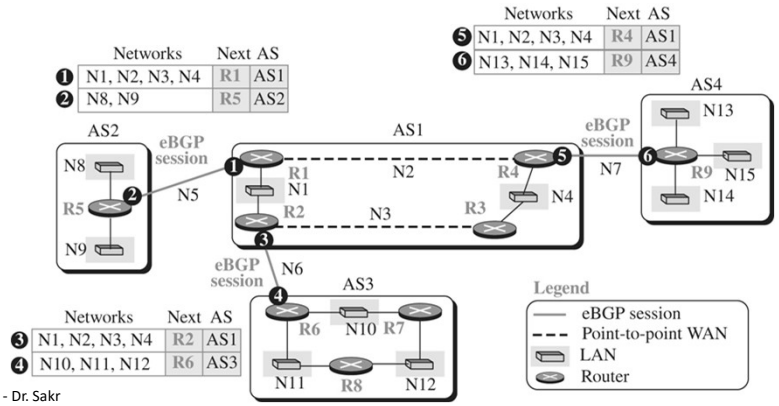
Example: Operation of eBGP

- Message ❶ is sent by R1 to tell R5 that N1-N4 can be reached by R1
  - R1 gets this information from the intra-domain forwarding table (RIP or OSPF)
  - R5 adds this information to its forwarding table
  - R5 will now forward all packets destined to N1-N4 to R1



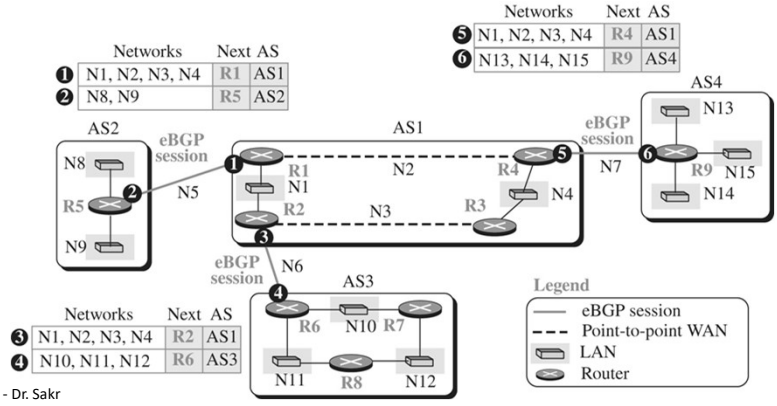
Example: Operation of eBGP (cont.)

- Problem 1:
  - Some border routers do not know how to route a packet destined for non-neighbor AS's
    - Example: R5 does not know how to route packets to networks in AS3 or AS4



Example: Operation of eBGP (cont.)

- Problem 2:
  - None of the non-border routers know how to route a packet destined for any network in other AS's
    - Example: R7 does not know how to route packets to networks in AS2 or AS4

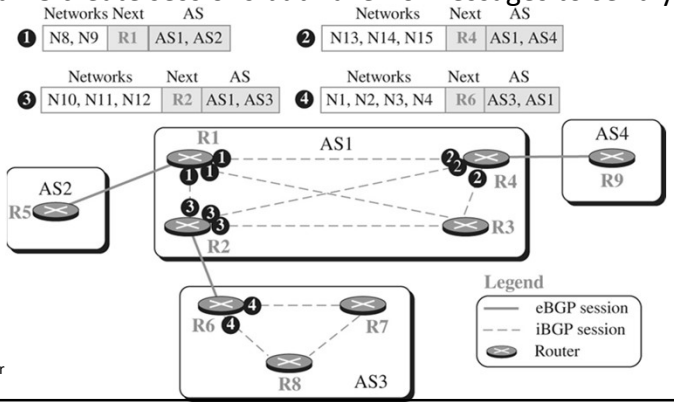


Operation of iBGP

- Solve both problems by allowing routers (border or non-border) in the same AS to form pairs exchange reachability information
  - 1 router in AS: there cannot be an iBGP session in AS (e.g., AS2 and AS4)
  - *n* routers in AS: there should be *n(n-1)/2* iBGP sessions in AS (e.g., 6 in AS2)
  - No flooding, each router advertise its reachability to the peer in the session

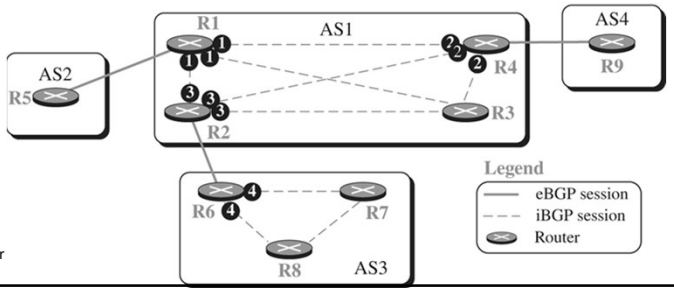
Example: Operation of eBGP and iBGP

- iBGP message ❶ is propagated by R1 to announce that N8-N9 are reachable through path AS1-AS2, but next router is R1
  - This message is sent to R2, R3, and R4 through three separate logical sessions
  - R2, R4, and R6 do the same (messages ❷, ❸, and ❹)
  - R3, R7, and R8 create sessions but have no messages to send yet



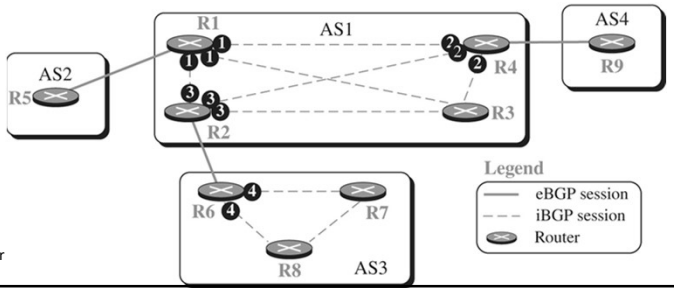
Example: Operation of eBGP and iBGP (cont.)

- It does not stop here*!*
  - R1 received iBGP update from R2 on the reachability info about AS3
  - It already knows reachability info about AS1
  - It combines and advertise a new eBGP update to R5
  - R5 now knows how to reach networks in AS1 and AS3

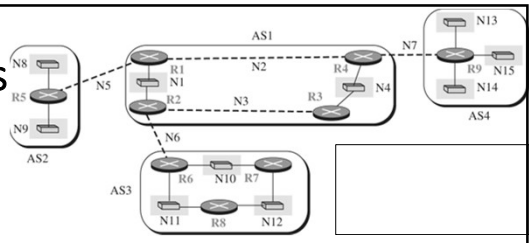


Example: Operation of eBGP and iBGP (cont.)

- It does not stop here*!*
  - Same after updates go from R4 to R1, R1 to R2, R1 to R3, and so on
  - Keeps going until there are no changes in previous updates
  - Each router combines all info from both eBGP and iBGP and creates a "path table" after applying the criteria to find the best path



Example: Finalized BGP Path Tables



Networks	Next	Path
N8, N9	R5	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Path table for R1

Networks	Next	Path
N8, N9	R1	AS1, AS2
N10, N11, N12	R6	AS1, AS3
N13, N14, N15	R1	AS1, AS4

Path table for R2

Networks	Next	Path
N8, N9	R2	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Path table for R3

Networks	Next	Path
N8, N9	R1	AS1, AS2
N10, N11, N12	R1	AS1, AS3
N13, N14, N15	R9	AS1, AS4

Path table for R4

Networks	Next	Path
N1, N2, N3, N4	R1	AS2, AS1
N10, N11, N12	R1	AS2, AS1, AS3
N13, N14, N15	R1	AS2, AS1, AS4

Path table for R5

Networks	Next	Path
N1, N2, N3, N4	R2	AS3, AS1
N8, N9	R2	AS3, AS1, AS2
N13, N14, N15	R2	AS3, AS1, AS4

Path table for R6

Networks	Next	Path
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Path table for R7

Networks	Next	Path
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Path table for R8

Networks	Next	Path
N1, N2, N3, N4	R4	AS4, AS1
N8, N9	R4	AS4, AS1, AS2
N10, N11, N12	R4	AS4, AS1, AS3

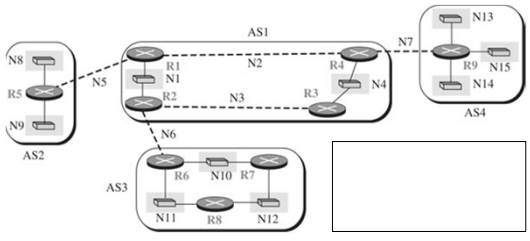
Path table for R9

Forwarding Tables after Injection from BGP

- It does not stop here*!*
  - BGP helps routers inside an AS to augment their forwarding tables
  - Inter-domain path tables created by BGP are not used for routing packets
  - Path tables are injected into forwarding tables (i.e., intra-domain RIP or OSPF) for routing packets

Example: Forwarding Tables after Injection from BGP

- A stub AS: add a default entry at the end of each forwarding table
  - Only one area border router



Des.	Next	Cost
N8	—	1
N9	—	1
0	R1	1

Table for R5

Des.	Next	Cost
N10	—	1
N11	—	1
N12	R7	2
0	R2	1

Table for R6

Des.	Next	Cost
N10	—	1
N11	R6	2
N12	—	1
0	R6	2

Table for R7

Des.	Next	Cost
N10	R6	2
N11	—	1
N12	—	1
0	R6	2

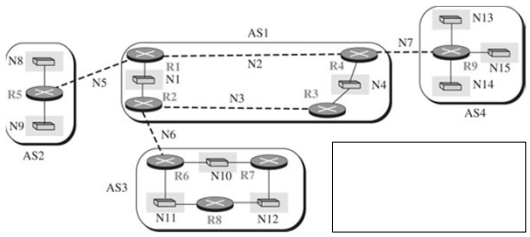
Table for R8

Des.	Next	Cost
N13	—	1
N14	—	1
N15	—	1
0	R4	1

Table for R9

Example: Forwarding Tables after Injection from BGP

- A transient AS: has to inject the whole contents of the path table
  - Multiple area border routers, no default entry
  - Address aggregation is used to shorten the table



Des.	Next	Cost
N1	—	1
N4	R4	2
N8	R5	1
N9	R5	1
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R1

Des.	Next	Cost
N1	—	1
N4	R3	2
N8	R1	2
N9	R1	2
N10	R6	1
N11	R6	1
N12	R6	1
N13	R3	3
N14	R3	3
N15	R3	3

Table for R2

Des.	Next	Cost
N1	R2	2
N4	—	1
N8	R2	3
N9	R2	3
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R3

Des.	Next	Cost
N1	R1	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R3	3
N12	R3	3
N13	R9	1
N14	R9	1
N15	R9	1

Table for R4

Path Attributes

- In both intra-domain routing protocols (RIP or OSPF), a destination is normally associated with two pieces of information:
  - Next hop: address of the next router to deliver the packet
  - Cost: defines the cost to the final destination
- Inter-domain routing is more involved and naturally needs more information about how to reach the final destination
- In BGP these pieces are called path attributes
- Example:
  - ORIGIN: source of routing information
  - AS-PATH: list of AS's through which the destination can be reached
  - NEXT-HOP: next router to which the packet should be forwarded

BGP Messages

- BGP uses four types of messages for communication between the BGP speakers across the AS's and inside an AS:
  - Open: opens a logical TCP connection to with a BGP peer; also authenticates sending peer
  - Update: advertises new path, withdraws old path, or both
  - Keepalive: keeps connection alive in absence of Update messages; also acknowledges Open request
  - Notification: reports errors in previous msg; also used to close connection
- All BGP packets share the same common header

Performance

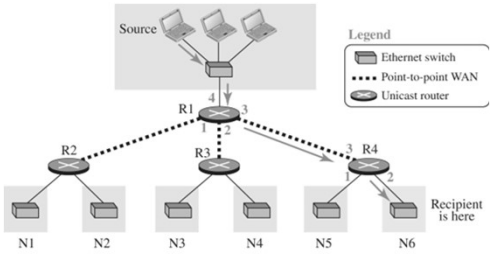
- BGP performance can be compared with RIP
- Similar to RIP:
  - BGP speakers exchange a lot of messages to create forwarding tables
  - Propagation of failure and corruptness also exists in BGP
- Unlike RIP:
  - BGP is free from loops and count-to-infinity

Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting

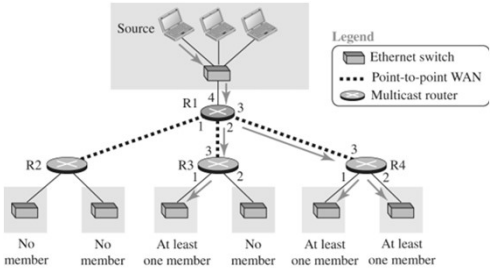
Unicasting vs. Multicasting vs. Broadcasting

- In unicasting, there is one source and one destination network
  - One-to-one relationship
  - Routers forward the packet to one and only one of its interfaces



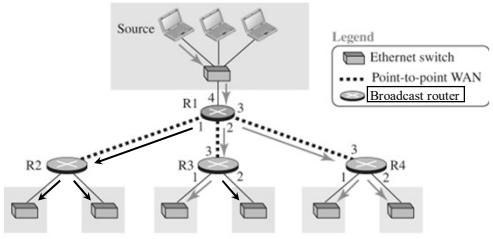
Unicasting vs. Multicasting vs. Broadcasting (cont.)

- In multicasting, there is one source and a group of destinations
  - One-to-many relationship (e.g., video conferencing, distance learning, etc.)
  - Source address is a unicast address, destination address is a group address
    - Each destination network has at least one member interested in receiving the datagram
    - Group address defines the members of the group
  - Routers may have to send copies of the packet through more than one of its interfaces

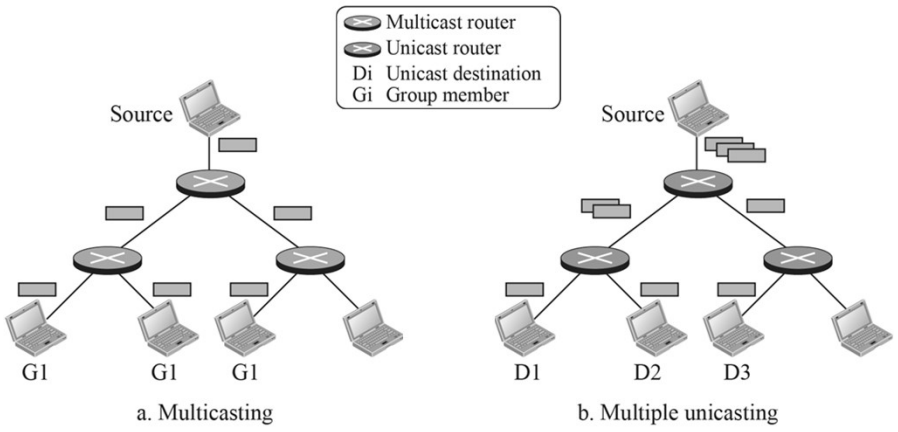


Unicasting vs. Multicasting vs. Broadcasting (cont.)

- In broadcasting, there is one source and all hosts in an internet are the destination
  - One-to-all relationship
  - Routers send copies of the packet through all of its interfaces
  - Not provided in this sense at the Internet level for obvious reasons
  - Controlled broadcasting, however, may be done in a domain (area or autonomous system)

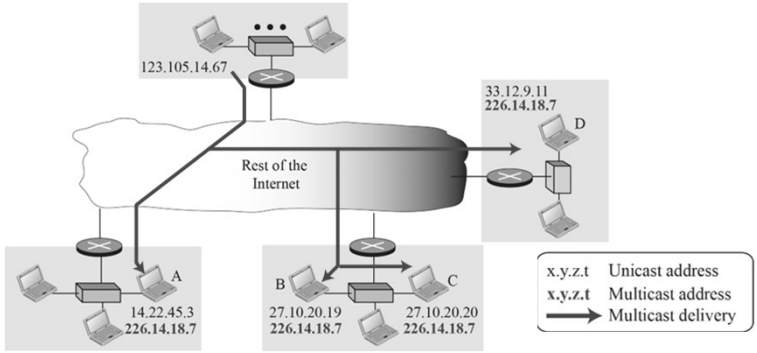


Why not Multiple Unicasting?



Multicasting Addresses

- Not practical to include the addresses of all recipients (sometimes thousands or millions) in the packet
  - Only one multicast address is used to define the group
- IPv4: CIDR 224.0.0.0/4 (or prefix 1110)
- IPv6: CIDR FF00::/8 (or block prefix 1111 1111)



Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting
  - Routing protocols

Routing protocols

- Distance Vector Multicast Routing Protocol (DVMRP)
  - Extension of RIP
- Multicast Open Shortest Path First (MOSPF)
  - Extension of OSPF
- Protocol Independent Multicast (PIM)

Outline

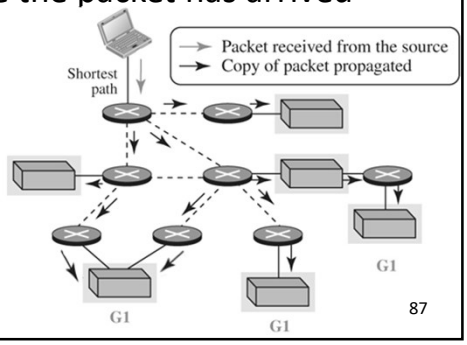
- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting
  - DVMRP

Distance Vector Multicast Routing Protocol (DVMRP)

- Extension of the Routing Information Protocol (RIP)
- Uses a source-based tree approach to multicasting
  - Routers make decision to forward traffic based on source address and not on destination address
- Each router that receives a multicast packet to be forwarded creates a source-based multicast tree from destination to source in three steps:
  - Reverse Path Forwarding (RPF):
    - create optimal tree between source and itself
  - Reverse Path Broadcasting (RPB):
    - create broadcast tree to all networks in the internet
  - Reverse Path Multicasting (RPM):
    - create multicast tree by cutting some branches to remove irrelevant networks

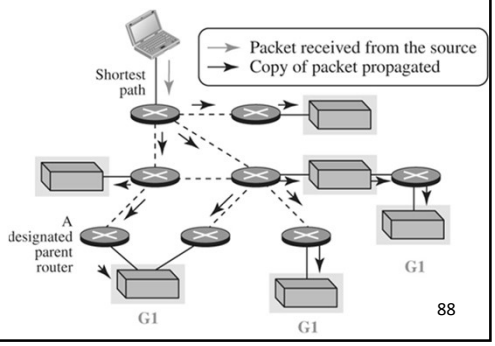
Reverse Path Forwarding (RPF)

- A router may receive multiple copies of the same multicast packet from different interfaces
- Routers forward only one copy that came through the interface with shortest path from the source to the router; other packets discarded
- To know which interface had the shortest-path, the router pretends it has a packet to send to the source from where the packet has arrived
  - Assumption: shortest path from A to B is also the shortest path from B to A (reverse path)
  - This prevents looping
- Note:
  - RPF does prevent each network from receiving more than one copy of the packet



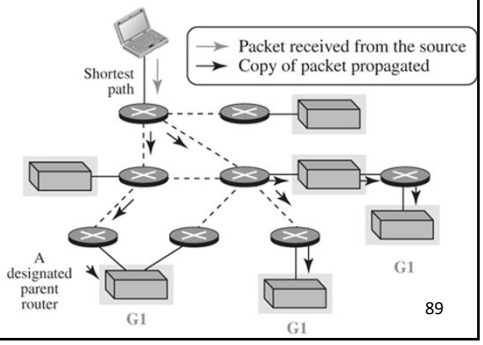
Reverse Path Broadcasting (RPB)

- One parent router is designated for each network
  - e.g., the router that has the shortest path to the source
- The network could accept multicast packets from parent router only
- When a router that is not the parent of the attached network receives a multicast packet, it simply drops the packet
- Note:
  - RPB does not multicast the packet, it broadcasts it to the whole internet, which is not efficient



Reverse Path Multicasting (RPM)

- To increase efficiency, the multicast packet must reach only those networks that have active members for that particular group
- To change the broadcast shortest-path tree to a multicast shortest-path tree, each router needs to prune (make inactive) the interfaces that do not reach a network with active members
  - This is corresponding to a particular source-group combination



Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting
  - MOSPF

Multicast Open Shortest Path First (MOSPF)

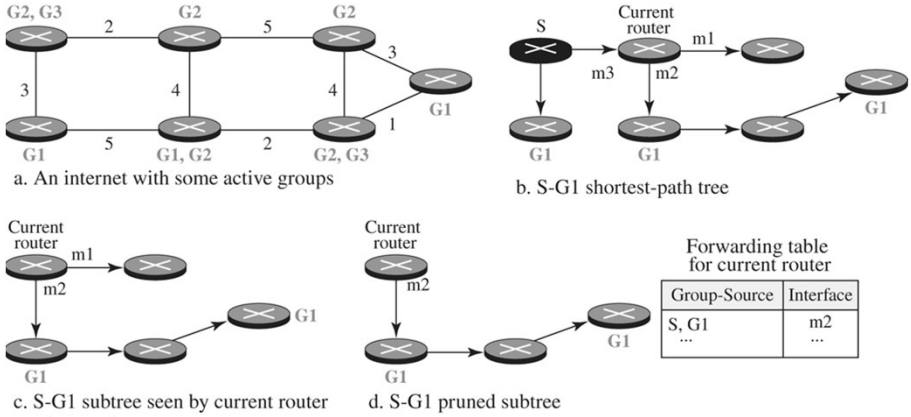
- Extension of the Open Shortest Path First (OSPF) protocol
- It also uses the source-based tree approach to multicasting
- To extend unicasting to multicasting, each router needs a database to show which interface has an active member in a particular group
  - besides the LSDB used for unicast link-state routing

MOSPF (cont.)

- Each router that receives a multicast packet from source S to be forwarded to destination group G as follows:
  1. Use Dijkstra's algorithm to create shortest-path tree with S as the root (not the router itself) and all destination in the internet with active members in G as the leaves using LSDB
  2. Router finds itself in the created tree to create a subtree
  3. Router prunes the broadcast subtree to create a multicast tree
  4. Forwards the packet out of only interfaces in the multicast tree

Example: Tree Formation in MOSPF

- This example shows how the source-based tree is made with the source as the root and changed to a multicast subtree with the current router as the root



a. An internet with some active groups

b. S-G1 shortest-path tree

c. S-G1 subtree seen by current router

d. S-G1 pruned subtree

Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting
  - PIM

Protocol Independent Multicast (PIM)

- A family of protocols that use a unicast routing protocol (which can be DV-based or LS-based) for its operation
- In other words, PIM needs to use the forwarding table of a unicast routing protocol to find the next router in a path to the destination, but it does not matter how the forwarding table is created
  - It does not build its own routing table
- PIM can work in two different modes:
  - Dense mode (DM): number of active members of a group is high
  - Sparse mode (SM): a few routers in the internet have active members in the group

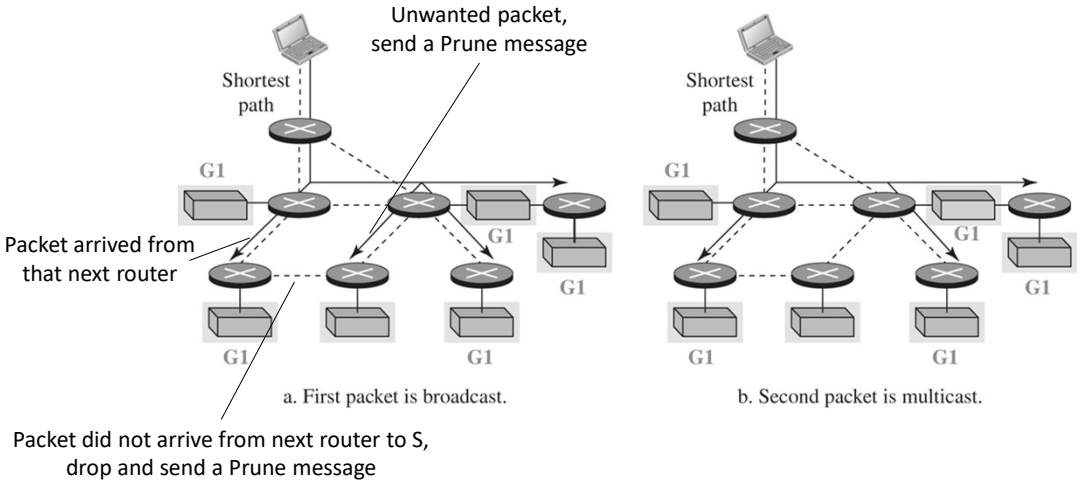
Protocol Independent Multicast-Dense Mode (PIM-DM)

- PIM works in dense mode when the number of routers with attached members is large relative to the number of routers in the internet
  - Multicast packet stream has receivers at most locations
- PIM-DM uses a source-based tree approach, similar to DVMRP, but simpler
  - Uses only two strategies described in DVMRP: RPF and RPM
- Unlike DVMRP, forwarding of a packet is not suspended awaiting pruning of the first subtree

Idea Behind PIM-DM

- Scenario: A router receives a multicast packet from source S destined to group G
- Router consults table of the underlying unicast protocol to find the next router if it wants to send a message to S (i.e., reverse direction)
  - If multicast packet did not arrive from that next router, drop and send a Prune message in that direction to prevent receiving future packets related to (S,G)
  - If multicast packet arrived from that next router, forward to all interfaces except the one(s) from which the packet arrived or a prune message related to (S,G) received
  - If multicast packet is unwanted, send a Prune message to the router up the stream
- Eventually, the broadcasting is changed to multicasting

Idea Behind PIM-DM (cont.)



Protocol Independent Multicast-Sparse Mode (PIM-SM)

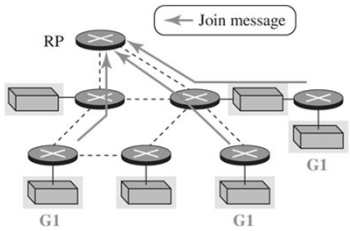
- PIM works in sparse mode when the number of routers with attached members is small relative to the number of routers in the internet
  - Multicast packet stream has relatively few receivers
  - The use of a protocol that broadcasts the packets until the tree is pruned is not justified
- PIM-SM uses a group-shared tree approach to multicasting
  - Explicitly constructs a tree from the sender to receivers in the multicast group
- PIM-SM selects one router to act as the rendezvous point (RP) for each active group
  - Other routers create a database to store group identifier and IP address of RP

Idea Behind PIM-SM

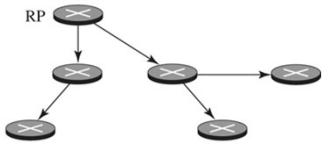
- Multicast communication is achieved in two steps:
  - Any router that has a multicast packet to send to a group of destinations first encapsulates the multicast packet in a unicast packet (tunneling) and sends it to the RP
  - The RP then decapsulates the unicast packet and sends the multicast packet to its destination
- To create a multicast tress rooted at the RP, PIM-SM uses join and prune messages to add/remove routers to the tree

Idea Behind PIM-SM (cont.)

- When a designated router finds out a member in a group (via IGMP):
  - Send a join message in a unicast packet destined to RP
  - Packet travels through the unicast short-path tree to reach RP
  - Any router along the path forwards the packet and adds the number of incoming and outgoing interfaces to its multicast forwarding table
  - That is, the joint message sent creates a path from RP to one of the networks with group members



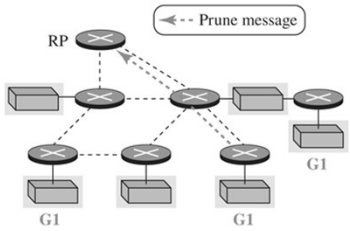
a. Three networks join group G1



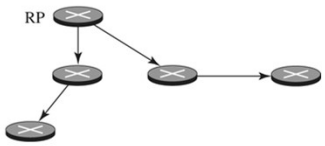
b. Multicast tree after joins

Idea Behind PIM-SM (cont.)

- When a designated router finds out no active member in a group (via IGMP):
  - Send a prune message in a unicast packet destined to RP
  - Packet travels through the unicast short-path tree to reach RP
  - Any router along the path forwards the packet and remove the entry from its multicast forwarding table



c. One network leaves group G1



d. Multicast tree after pruning

Outline

- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting
  - IGMP

Internet Group Management Protocol (IGMP)

- Used for collecting information about group membership
- One of the auxiliary protocols defined at the network layer
- IGMP uses two types of messages:
  - Query messages: sent periodically by a router to all hosts attached to it
    - Ask hosts to report about their membership in groups
  - Report messages: sent by a host as a response to a query message
- IGMP messages are encapsulated in an IP datagram with the value of the protocol field set to 2

Outline

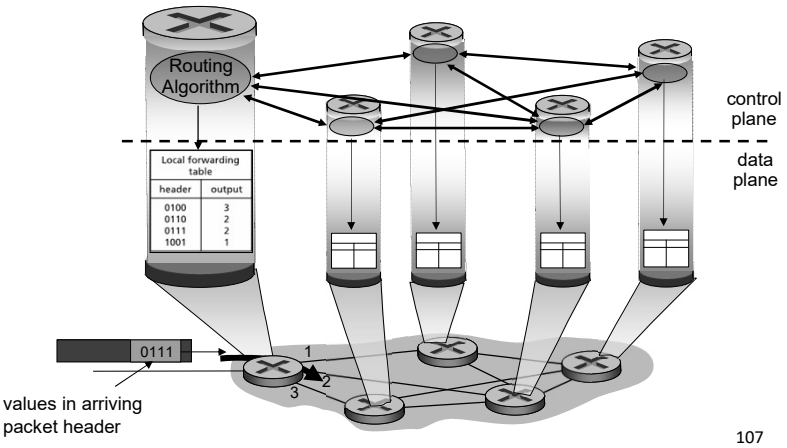
- Internet as a graph
- Routing algorithms
- Routing protocols
- Multicasting
- Remarks

Router Control Plane

- Note that routers need to establish logical connections which need higher-layer understanding
- A router needs some higher-layer functionality in its control plane to be able to use transport-layer and application-layer protocols
  - For example, RIP uses UDP and BGP uses TCP for route exchange
- In practice, routers contain switching hardware, runs proprietary implementation of Internet standard protocols (e.g., IP, RIP, IS-IS, OSPF, BGP, etc.) in proprietary router OS (e.g., Cisco IOS)

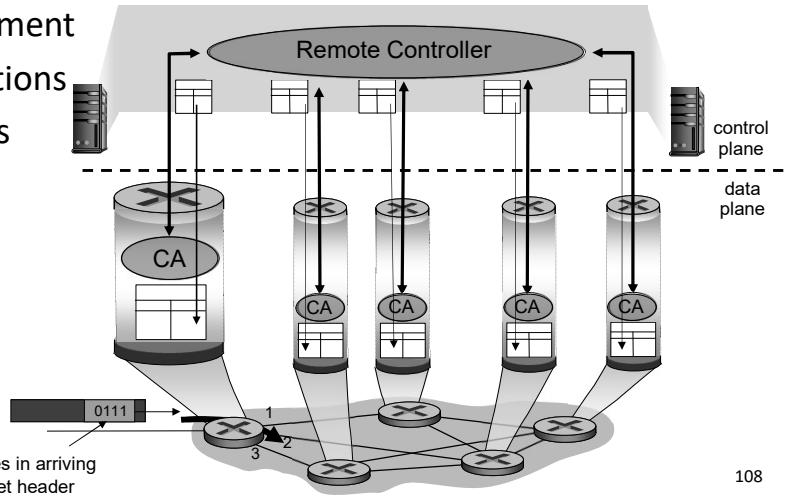
Router Control Plane (cont.)

- Individual routing algorithm components in each and every router interact in the control plane to compute forwarding tables
  - Per-router control plane



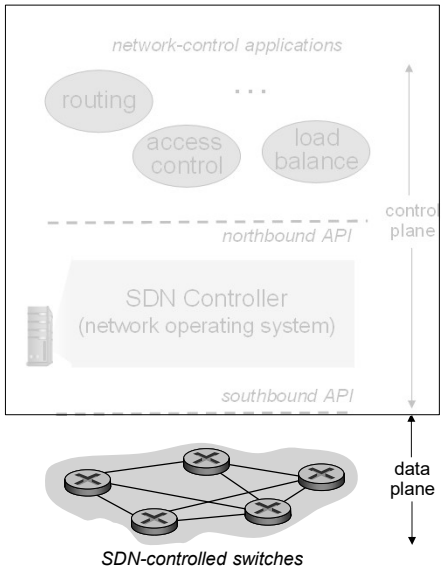
Software-defined Networking (SDN) Control Plane

- Remote controller interacts with local controller agents (CA) to compute and install forwarding tables in routers
- Easier network management
- No router misconfigurations
- Flexibility of traffic flows



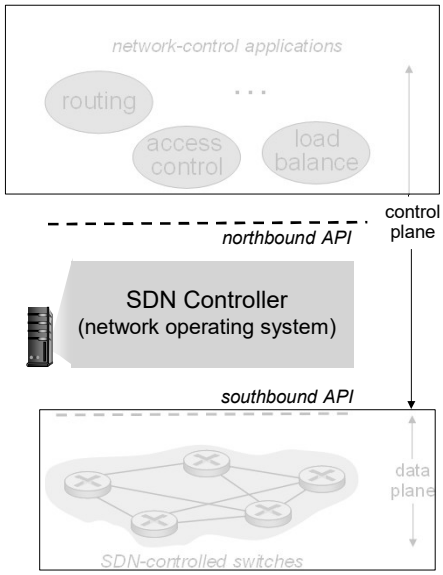
SDN (cont.)

- Data-plane switches:
  - fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
  - flow (forwarding) table computed, installed under controller supervision
  - API for table-based switch control
  - protocol for communicating with controller



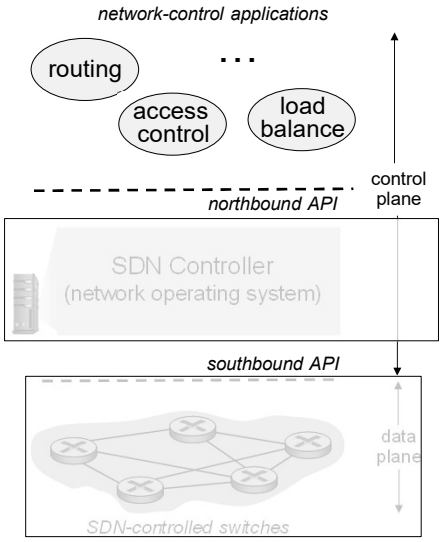
SDN (cont.)

- SDN controller (Network OS):
  - maintain network state information
  - interacts with network control applications "above" via northbound API
  - interacts with network switches "below" via southbound API
  - implemented as distributed system for performance, scalability, fault-tolerance, robustness, etc.



SDN (cont.)

- Network-control apps:
  - "brains" of control: implement control functions using lower-level services, API provided by SDN controller
  - unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller



Summary

- We covered:
  - Routing algorithms: DV-, LS-, and PV-based routing
  - Unicast routing protocols: RIP, OSPF, and BGP
  - Multicast routing protocols: DVMRP, MOSPF, and PIM