6.263 Data Communication Networks

Lecture 3: Internet Routing

(some slides are taken from I. Stoica and N. Mckewon & T. Griffin)

Dina Katabi

dk@mit.edu
www.nms.csail.mit.edu/~dina

Books

Text Book

- **Data Communication Networks, by Dimitri Betsekas and Robert Gallager**

Recommended

- **Computer Networks - A Systems Approach, by Larry L. Peterson and Bruce S. Davie, 3rd edition.**

Lecture 1: Taxonomy of Networks

(Mainly notes. Reading: P&D 3.1 pp. 164-180)

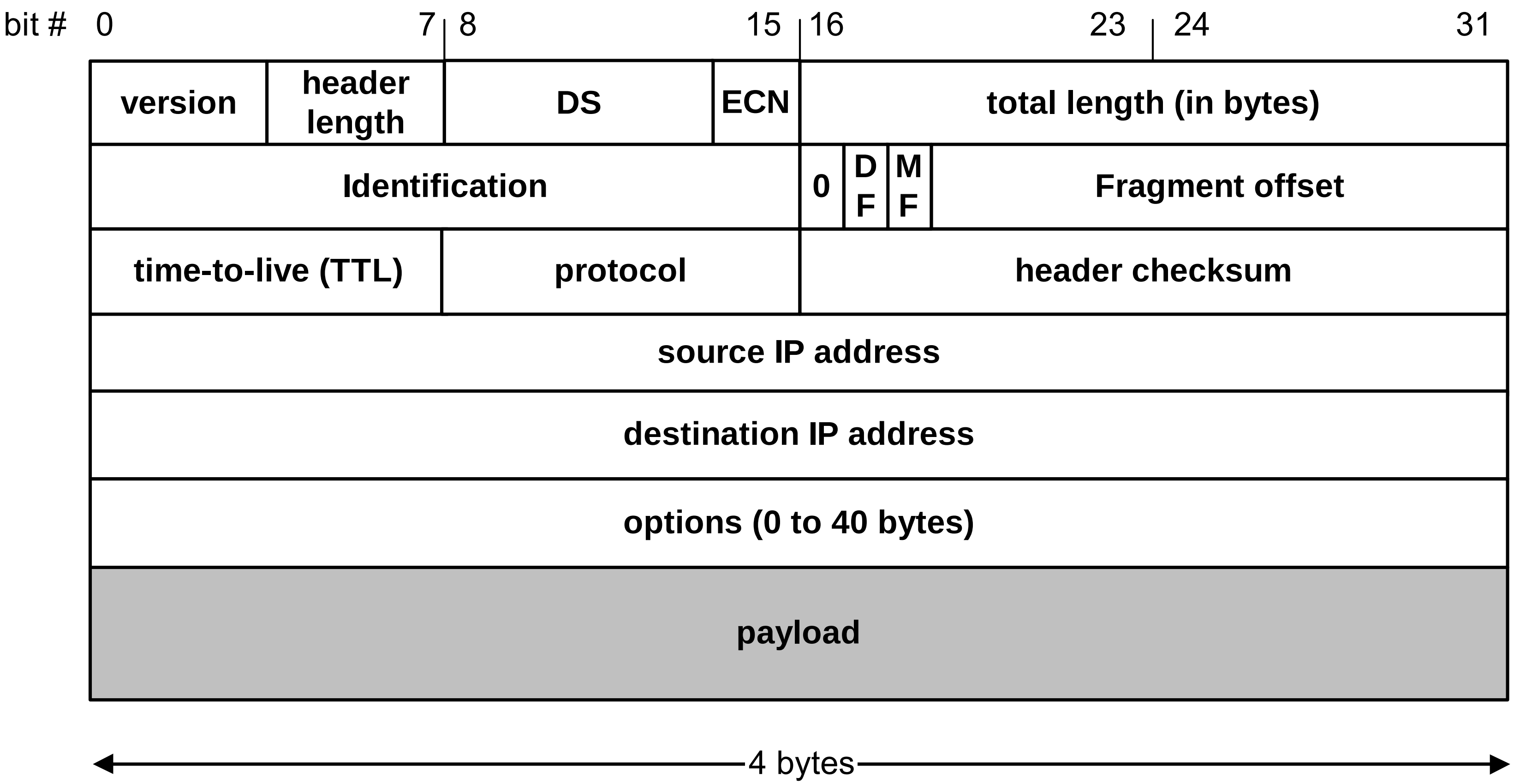
Lecture 2: Routing Algorithms

Bertsekas and Gallager

Lecture 3: Internet Routing (The real picture)

(Notes. Also you can find the material in P&D 4.2 except 4.2.5, 4.3.1, 4.3.2, 4.3.3. For algs., see B&G-5.2.3)

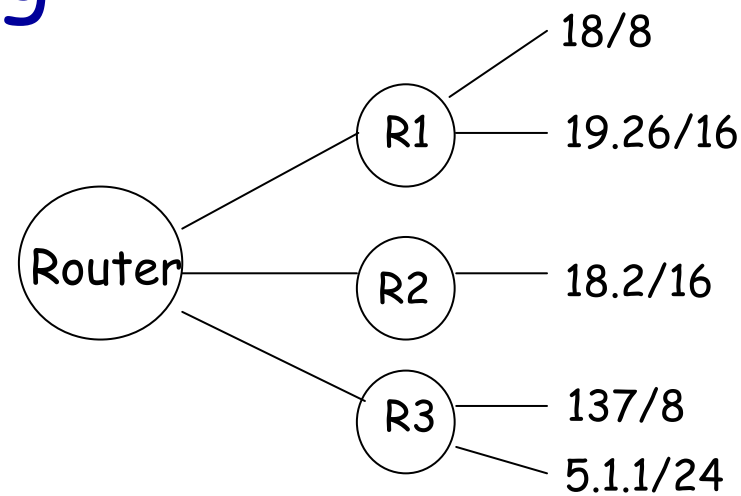
IP Addresses & IP Header



- IP address: e.g., 18.26.0.1
- IP prefix:
  - e.g., 18/8 contains 2^{24} addresses
  - e.g, 18.26/16 or 18.26.0/24

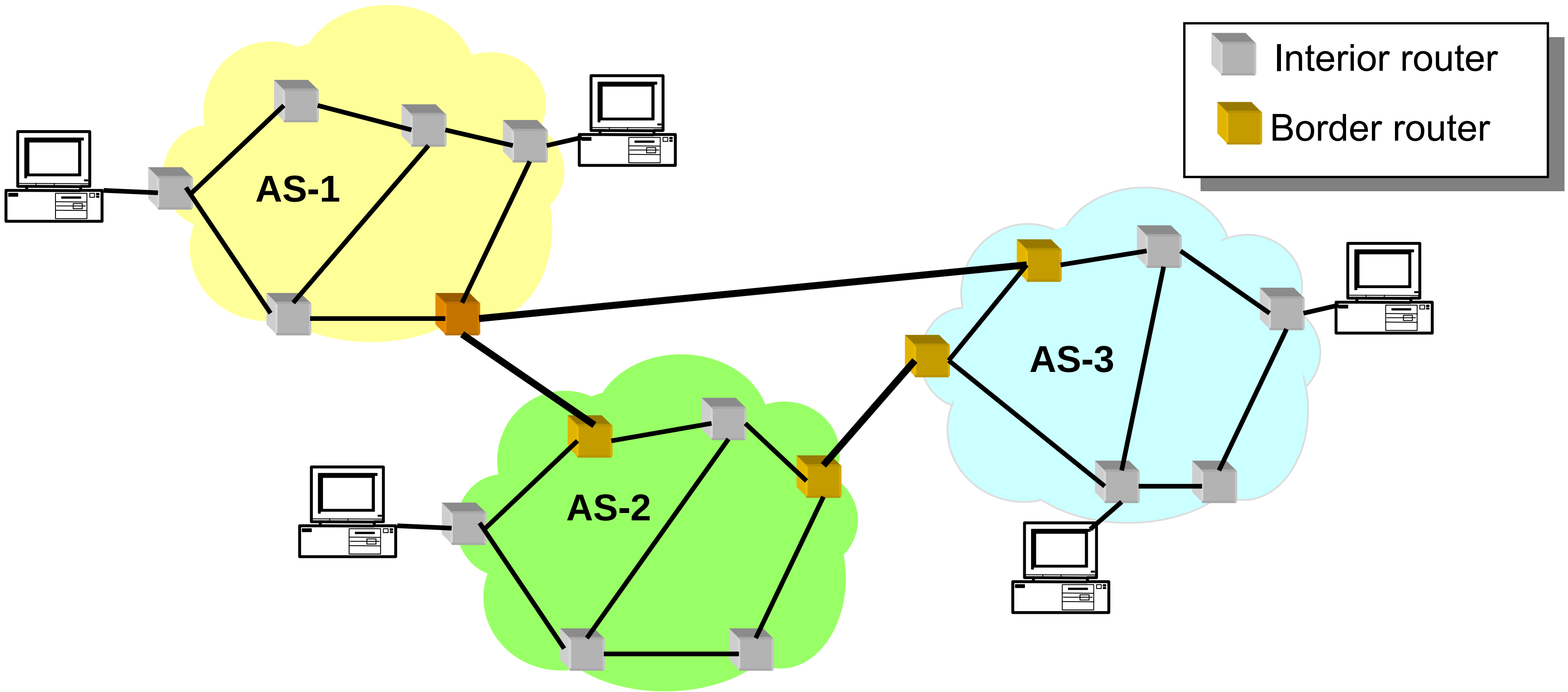
Forwarding

- What's the difference between routing and forwarding?
  - Routing is finding the path
  - Forwarding is the action of sending the packet to the next-hop toward its destination
- Each router has a forwarding table
  - Forwarding tables are created by a routing protocol
- Forwarding:
  - When a packet arrives,
    - router looks up the dst. IP in the forwarding table
    - finds nexthop
    - queues packet to be sent on the interface connected to nexthop
- Forwarding needs to be fast



Dest. Addr	Next-hop
18/8	R1
18.2/16	R2
19.26/16	R1
5.1.1/24	R3
137/8	R2

Picture of the Internet



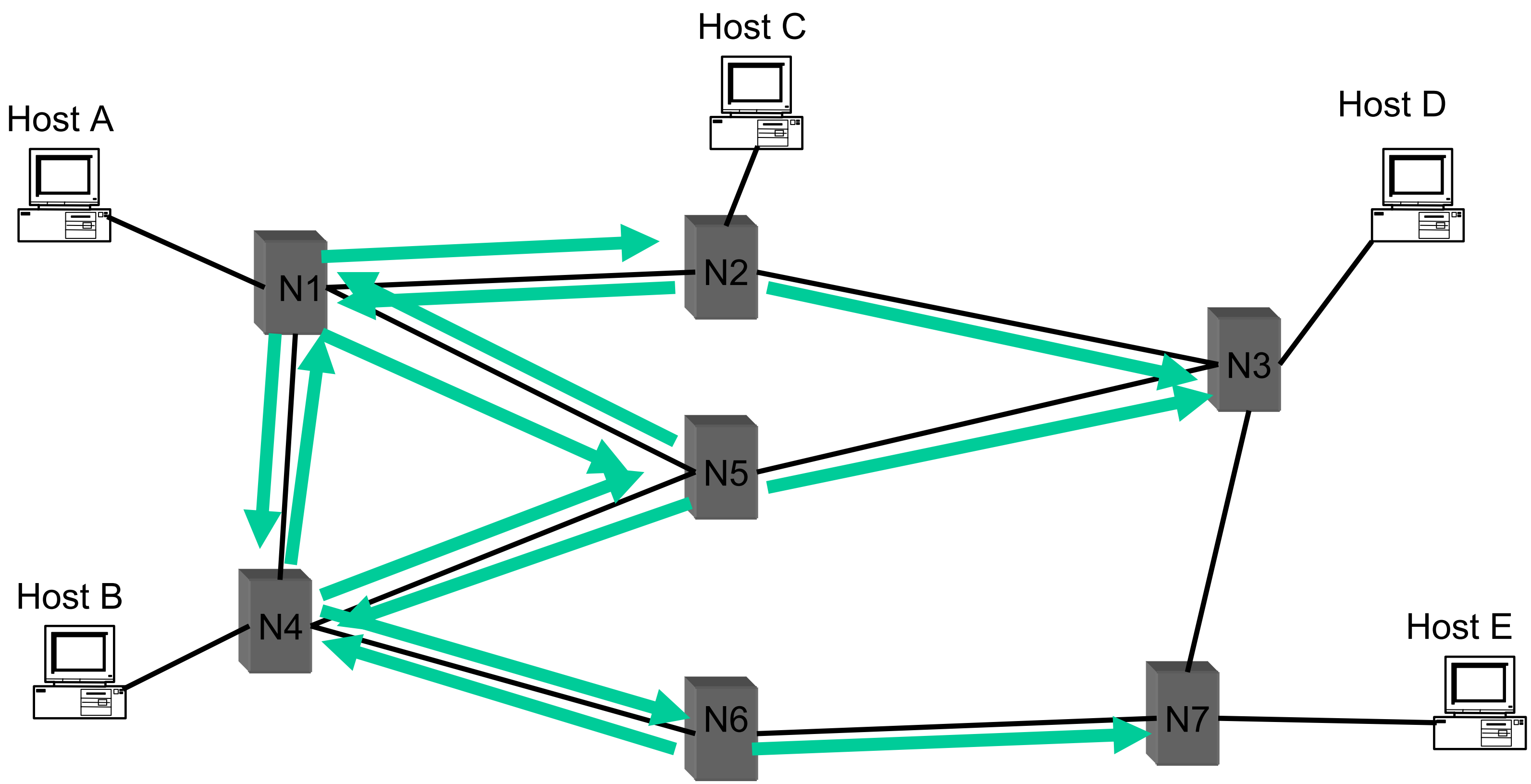
- Internet: A collection of Autonomous Systems (ASes)
  - E.g., MIT network, AT&T, Quest, Sprint, …
- Routing is hierarchical
  - Intra-Domain Routing inside an AS
  - Inter-Domain Routing between ASes

Two Main Approaches to Intra-domain Routing

- Distance Vector Protocols
  - E.g., RIP
  - Based on Distributed Bellman-Ford Algorithm
  - Periodically, each node tells its neighbors about its shortest route to all other nodes in the network
- Link State Protocols
  - E.g., OSPF
  - Based on Dijkstra Algorithm
  - Periodically, each node tells all nodes about its neighbors (it floods that information)

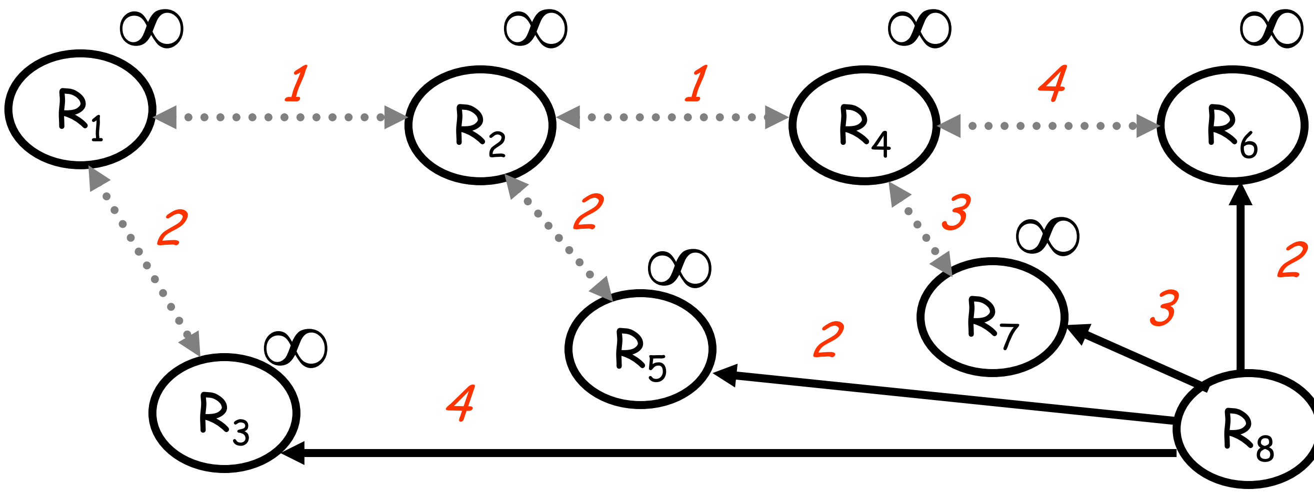
Distance Vector: Control Traffic

- When the routing table of a node changes, the node sends its table to its neighbors
- A node updates its table with information received from its neighbors



Distributed Bellman-Ford Algorithm

Example: Compute routes to R8

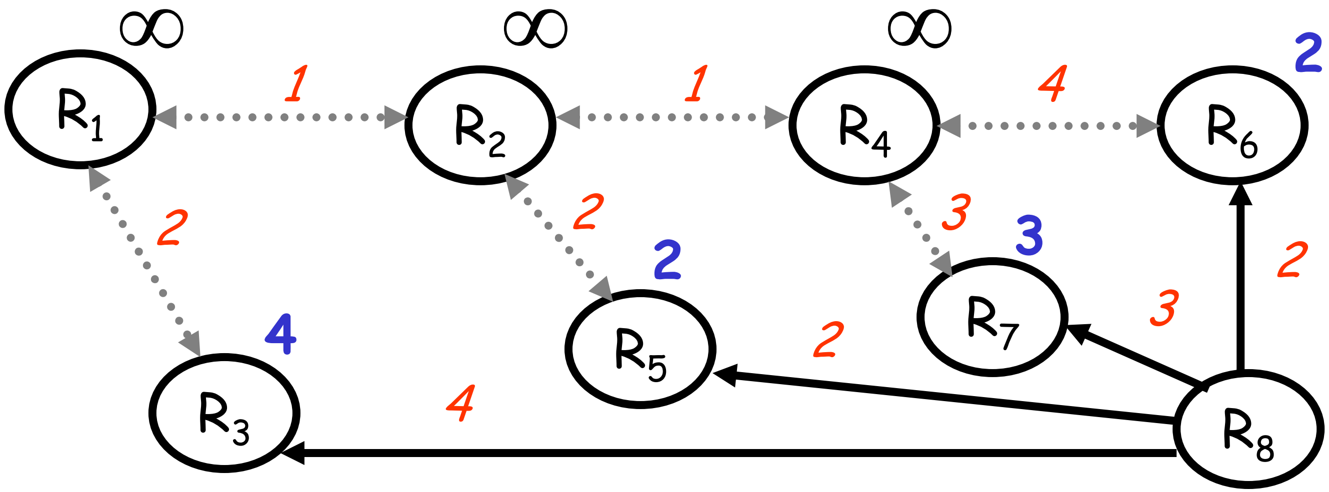


Initial State: All routers except R8 set their route length to ∞. R8 sets its route length to 0.

Distributed Bellman-Ford Algorithm

Example: Compute routes to R8

R_1	Inf
R_2	Inf
R_3	4, R_8
R_4	Inf
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



Iterative step:

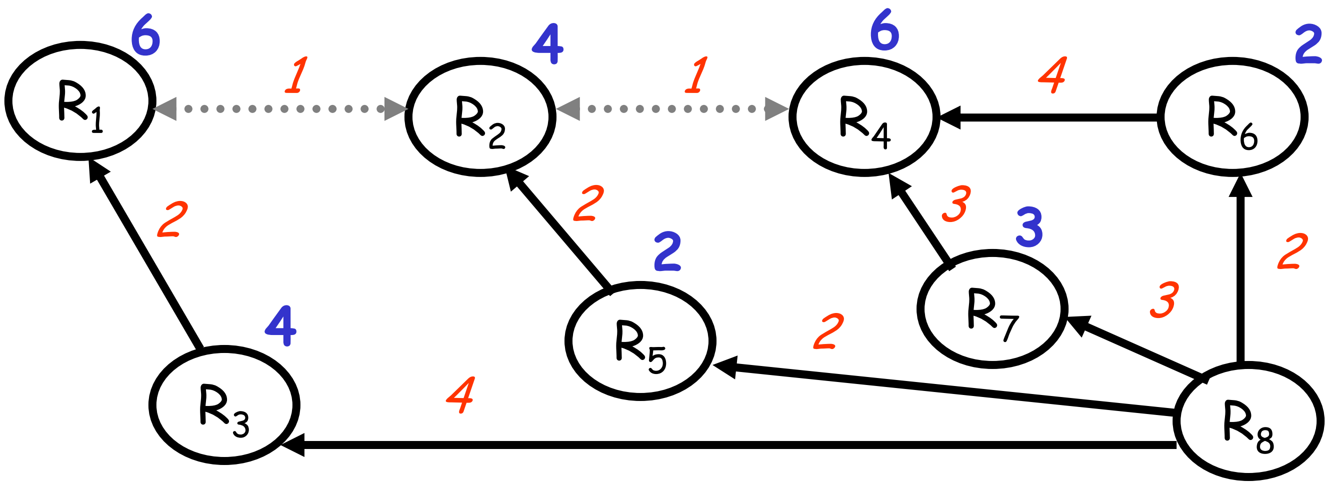
- Every T seconds, each router tells its neighbors its lowest-cost to R8
- When a router receives a message from neighbor j, it updates its path cost as:
  - If (*j is my current nexthop*) then cost= *j's cost to R8 + cost of the link to j*
  - else cost =*min(current cost, j's cost to R8 + cost of the link to j)*

Note, routing tables at each router have both the next-hop and the cost

Distributed Bellman-Ford Algorithm

Example: Compute routes to R8

R_1	6, R3
R_2	4, R5
R_3	4, R_8
R_4	6, R7
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8

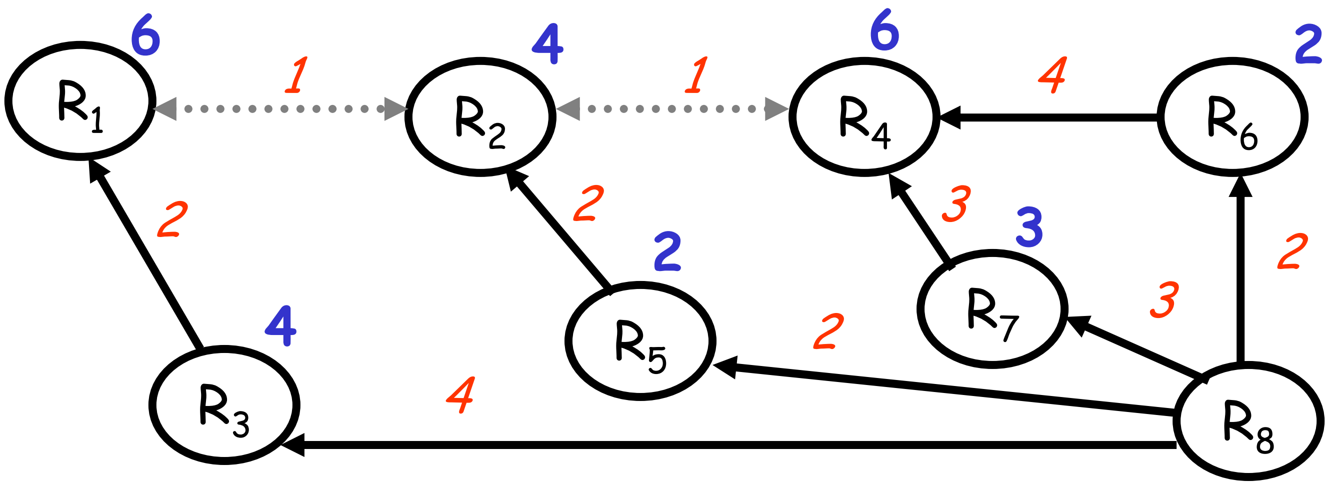


Repeat until no costs change

Distributed Bellman-Ford Algorithm

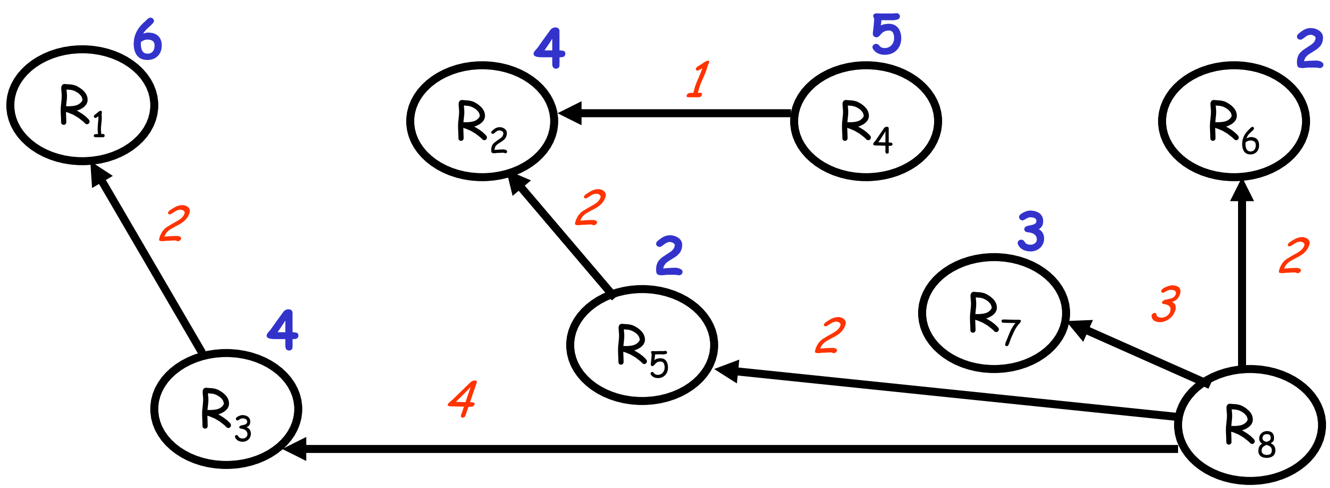
Example: Compute routes to R8

R_1	6, R3
R_2	4, R5
R_3	4, R_8
R_4	6, R7
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



Solution

R_1	6, R3
R_2	4, R5
R_3	4, R_8
R_4	5, R2
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



Formally

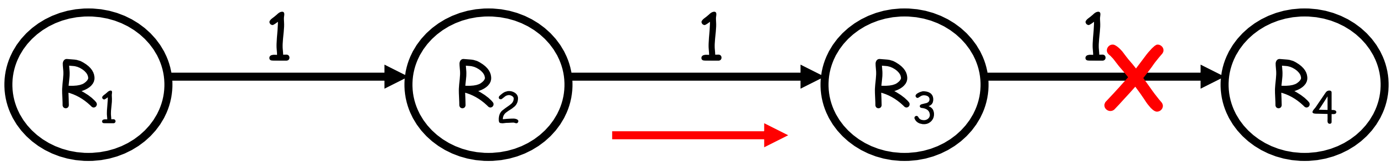
At router *A*, assume c(*A*,V) is the cost of the edge *A*-V and D(*A*,V) is the cost of the path from *A* to V

```
Initialization:
  for all neighbors V  do
    if V adjacent to A
      D(A, V) = c(A,V);
  else
      D(A, V) = ∞;

loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
      for all destinations Y through V do
         D(A,Y) =  D(A,Y) + d
   else if (update D(V, Y) received from V)
      for all destinations Y do
         if (destination Y through V)
           D(A,Y) = c(A,V) + D(V, Y);
         else
            D(A, Y) = min(D(A, Y), c(A, V) + D(V, Y));
   if (there is a new minimum for destination Y)
     send D(A, Y) to all neighbors
  forever
```

A Problem with Bellman-Ford

"Bad news travels slowly"



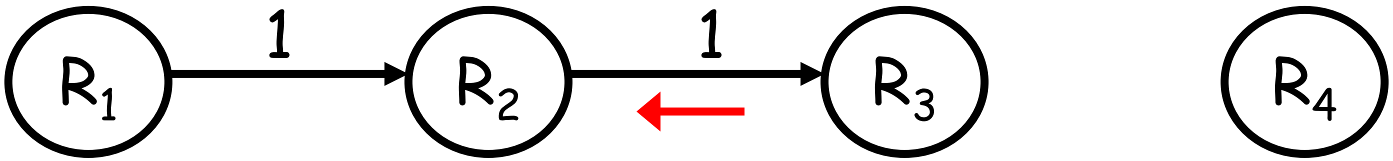
Consider the calculation of distances to R_4:

Time	R_1	R_2	R_3
0	3,R_2	2,R_3	1, R_4
1	3,R_2	2,R_3	**3,R_2**
2			
3			

$R_3 \rightarrow R_4$ fails

A Problem with Bellman-Ford

"Bad news travels slowly"



Consider the calculation of distances to R_4:

Time	R_1	R_2	R_3
0	3,R_2	2,R_3	1, R_4
1	3,R_2	2,R_3	3,R_2
2	3,R_2	**4,R_3**	3,R_2
3	5,R_2	4,R_3	5,R_2
…	"Counting to infinity"		…

$R_3 \rightarrow R_4$ fails

How are These Loops Caused?

- Observation 1:
  - R3's metric increases
- Observation 2:
  - R2 picks R3 as next hop to R4; But, the implicit path from R2 to R4 includes R3!

Counting to Infinity Problem
Solutions

1. Set infinity = “some small integer” (e.g. 200). Stop when count = 200.
2. Split Horizon: Because R_2 received lowest cost path from R_3, it does not advertise cost to R_3
3. Split-horizon with poison reverse: R_2 advertises infinity to R_3

- *Can you think of another approach?*

Two Main Approaches to Intra-domain Routing

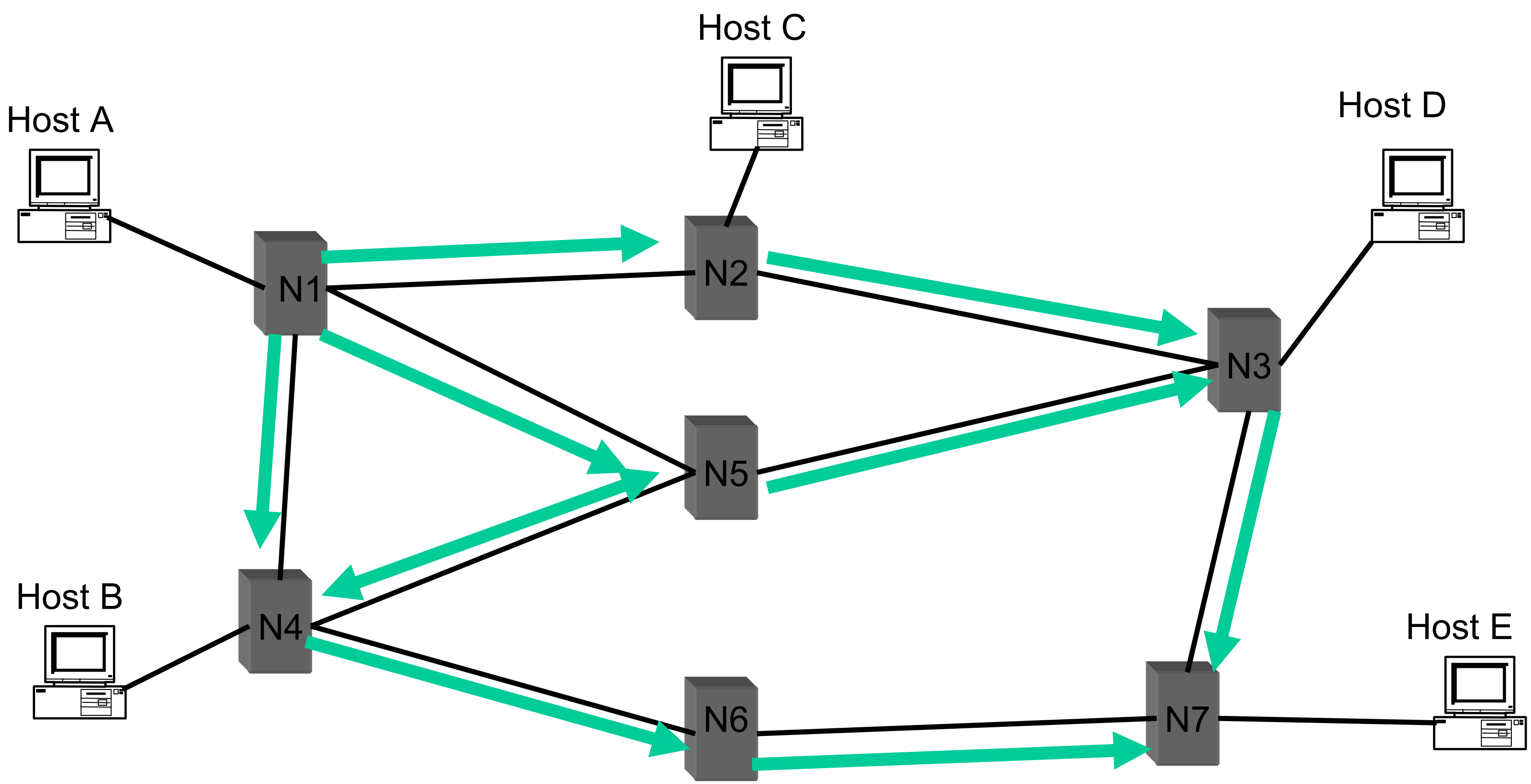
- Distance Vector Protocols
  - E.g., RIP
  - Based on Distributed Bellman-Ford Algorithm
  - Periodically, each node tells its neighbors about its shortest route to all other nodes in the network
- Link State Protocols
  - E.g., OSPF
  - Based on Dijkstra Algorithm
  - Periodically, each node tells all nodes about its neighbors (it floods that information)

Link State

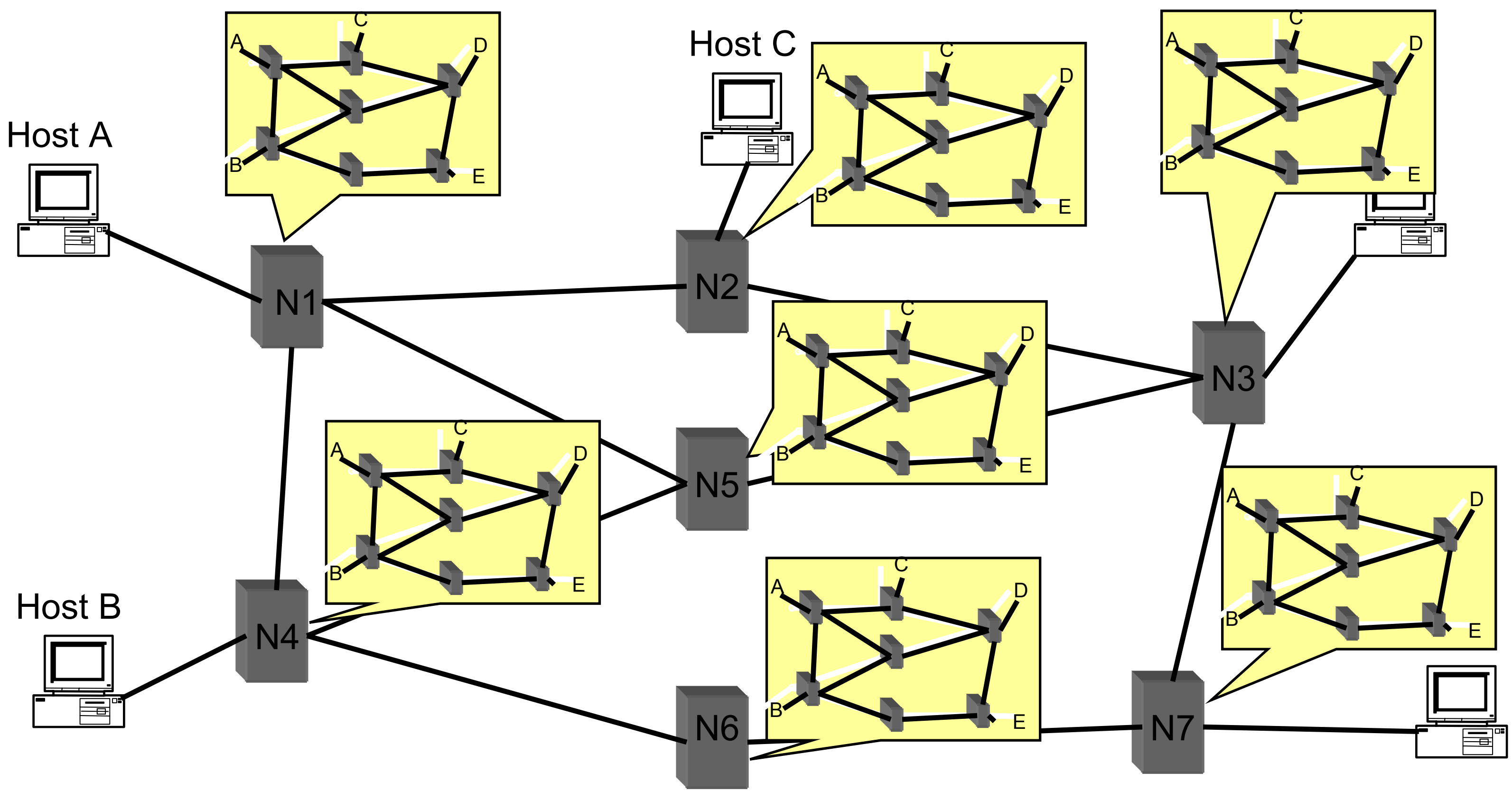
- Start condition
  - Each node is assumed to know its neighbors
- Phase 1: Each node learns the graph
  - Each node broadcasts its neighbor list to all other nodes in a message called LSP
    - LSP contains: Node ID, List of neighbors and the link cost to reach them, Sequence number (needed to avoid stale information from propagating), Time to live (TTL)
  - Reliable flooding mechanism
    - When node J receives LSP from node K,
      - if LSP is the most recent LSP from K that J has seen so far, J saves it in database and forwards a copy on all links except the link on which the LSP was received, otherwise, it discards the LSP
- Phase 2: Each node computes shortest paths
  - Each node locally computes shortest paths to all other nodes from global state
  - It uses Dijkstra's shortest path tree (SPT) algorithm

Link State: Control Traffic

- Each node floods info about its neighbors to every node in the network
- Each node ends up knowing the entire network topology → use Dijkstra to compute the shortest path to every other node



Link State: Node State



Dijkstra's Shortest Path First Algorithm

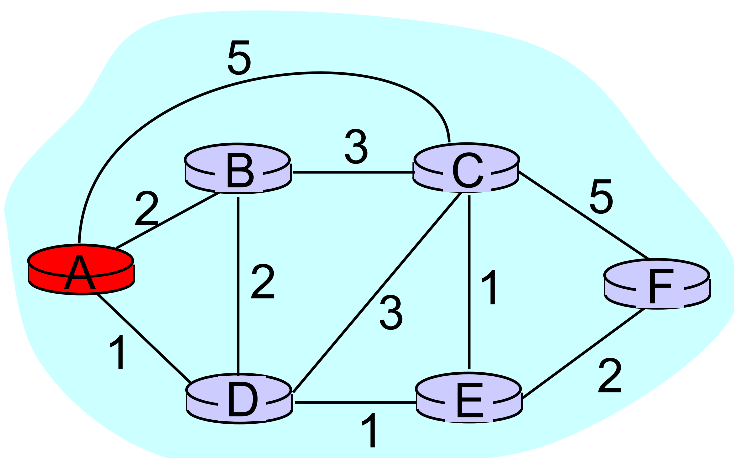
- Assumptions:
  - Costs are positive
  - Each router has the graph. *Is it scalable?*
- For each source, the alg. finds the spanning tree routed at the source

Example: Dijkstra's Algorithm

Node A uses Dijkstra to find routes from A to all nodes

D(x) is distance to X and p(x) is parent of node X in the spanning tree rooted at A

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						



Initialization:

S = {A};

for all nodes *v*

if *v* adjacent to *A*

then D(v) = c(A,v);

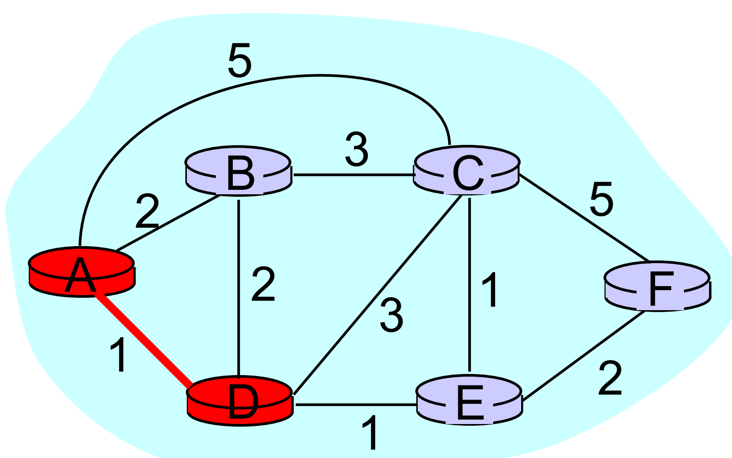
else D(v) = ∞;

…

Example: Dijkstra's Algorithm

Node A uses Dijkstra to find routes from A to all nodes

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						



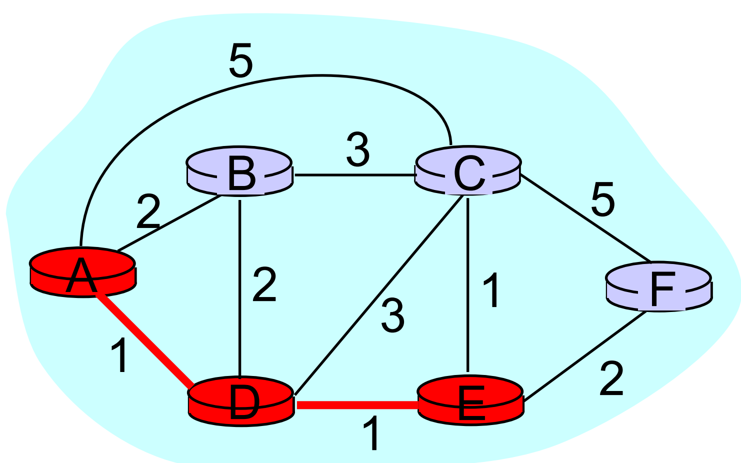

```
…
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
  update D(v) for all v adjacent
   to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
   until all nodes in S;
```

<u>Dijkstra's Key Idea:</u> At each step, find the next closest node

Example: Dijkstra's Algorithm

Node A uses Dijkstra to find routes from A to all nodes

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						

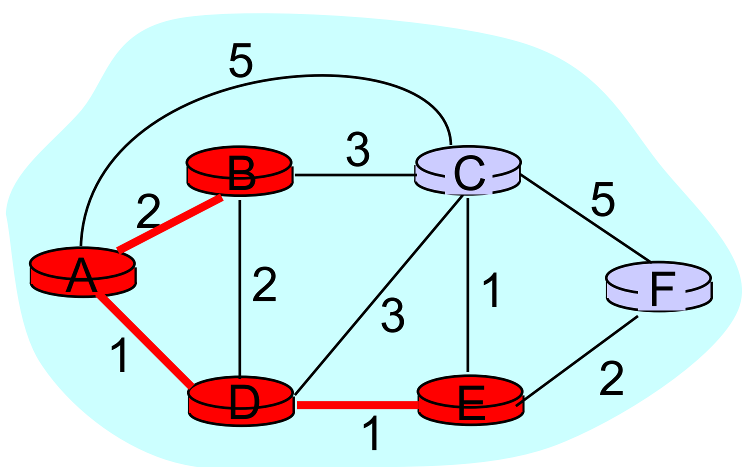



```
…
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
  update D(v) for all v adjacent
   to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

Node A uses Dijkstra to find routes from A to all nodes

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						

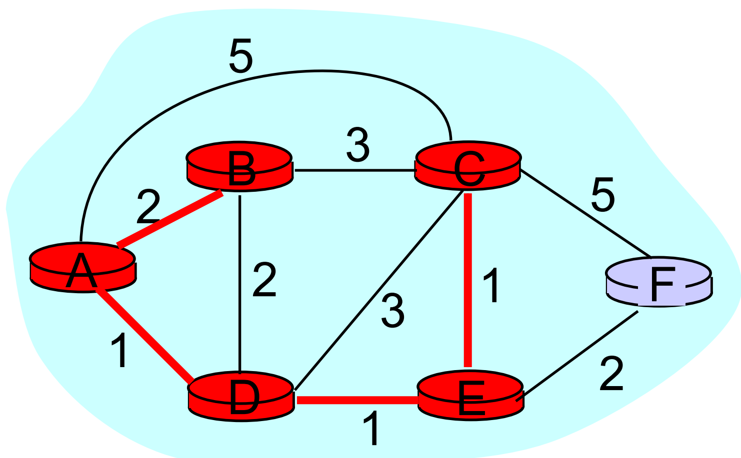



```
...
Loop
    find w not in S s.t. D(w) is a minimum;
    add w to S;
   update D(v) for all v adjacent
    to w and not in S:
       D(v) = min( D(v), D(w) + c(w,v) );
    until all nodes in S;
```

Example: Dijkstra's Algorithm

Node A uses Dijkstra to find routes from A to all nodes

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						

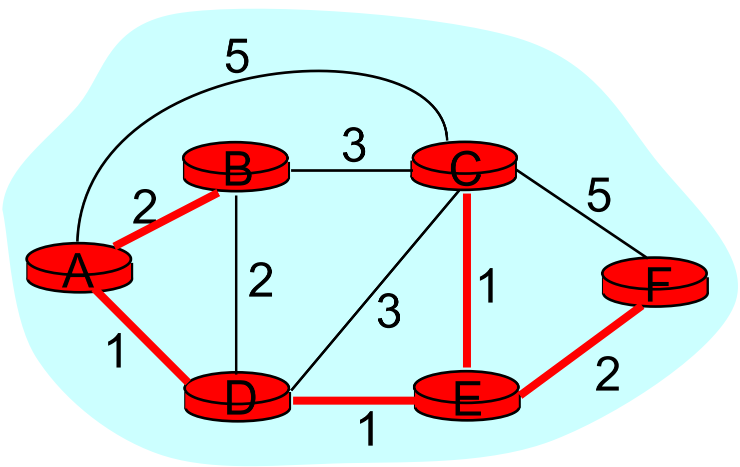



```
...
Loop
    find w not in S s.t. D(w) is a minimum;
    add w to S;
   update D(v) for all v adjacent
    to w and not in S:
       D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

Node A uses Dijkstra to find routes from A to all nodes

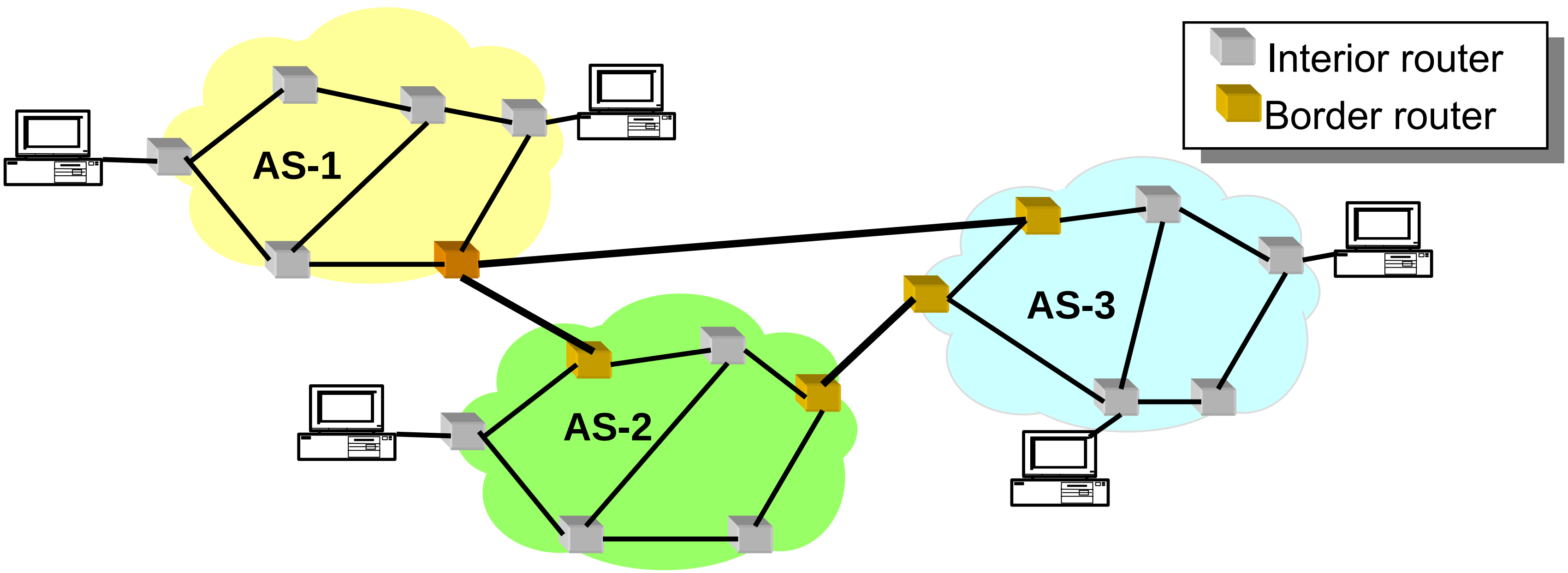
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					



…
Loop
find w not in S s.t. D(w) is a minimum;
add w to S;
update D(v) for all v adjacent to w and not in S:
D(v) = min(D(v), D(w) + c(w,v));
until all nodes in S;

Inter-Domain Routing & BGP

Picture of the Internet



- Intra-domain routing inside an AS
- Iner-domain routing between ASes

Internet Hierarchy

- What is an Autonomous System (AS)?
  - A set of routers under a single technical administration, using an *intra-domain routing protocol* (IGP) and common metrics to route packets within the AS and using an *inter-domain routing protocol* (EGP) to route packets to other ASes
  - Sometimes ASes use multiple intra-domain routing protocols and metrics, but appear as a single AS to other ASes
- Each AS is assigned a unique ID

AS Numbers (ASNs)

ASNs are 16 bit values.
64512 through 65535 are "private"

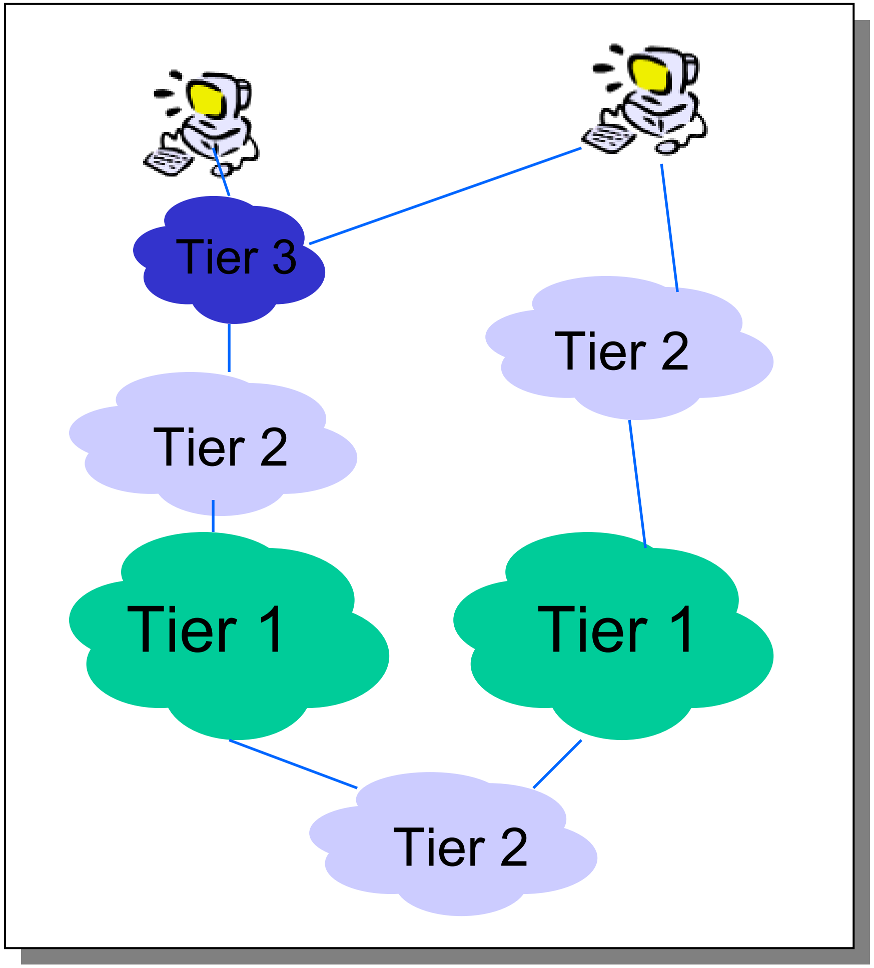
Currently over 16,000 in use.

- Genuity: 1
- MIT: 3
- Harvard: 11
- UC San Diego: 7377
- AT&T: 7018, 6341, 5074, ...
- UUNET: 701, 702, 284, 12199, ...
- Sprint: 1239, 1240, 6211, 6242, ...
- ...

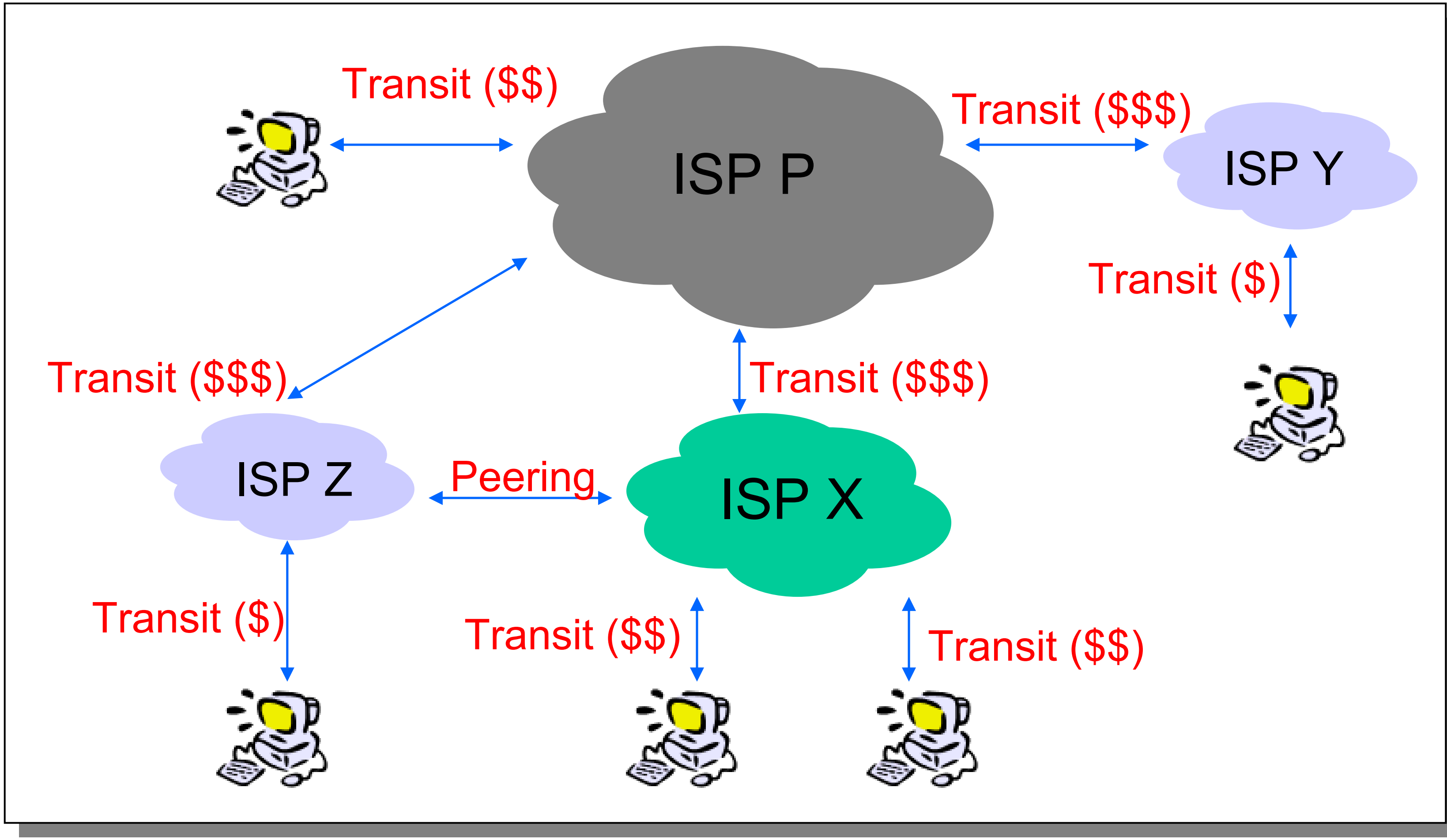
ASNs represent units of routing policy

A Logical View of the Internet

- Internet connectivity is provided by commercial entities called ISPs, who compete for profit yet have to cooperate to provide connectivity
  - Each ISP has its own AS (sometimes multiple ASes)
- Not all ISPs are created equal
  - Tier 1 ISP
    - "Default-free" global reachability info
  - Tier 2 ISP
    - Regional or country-wide
  - Tier 3 ISP
    - Local



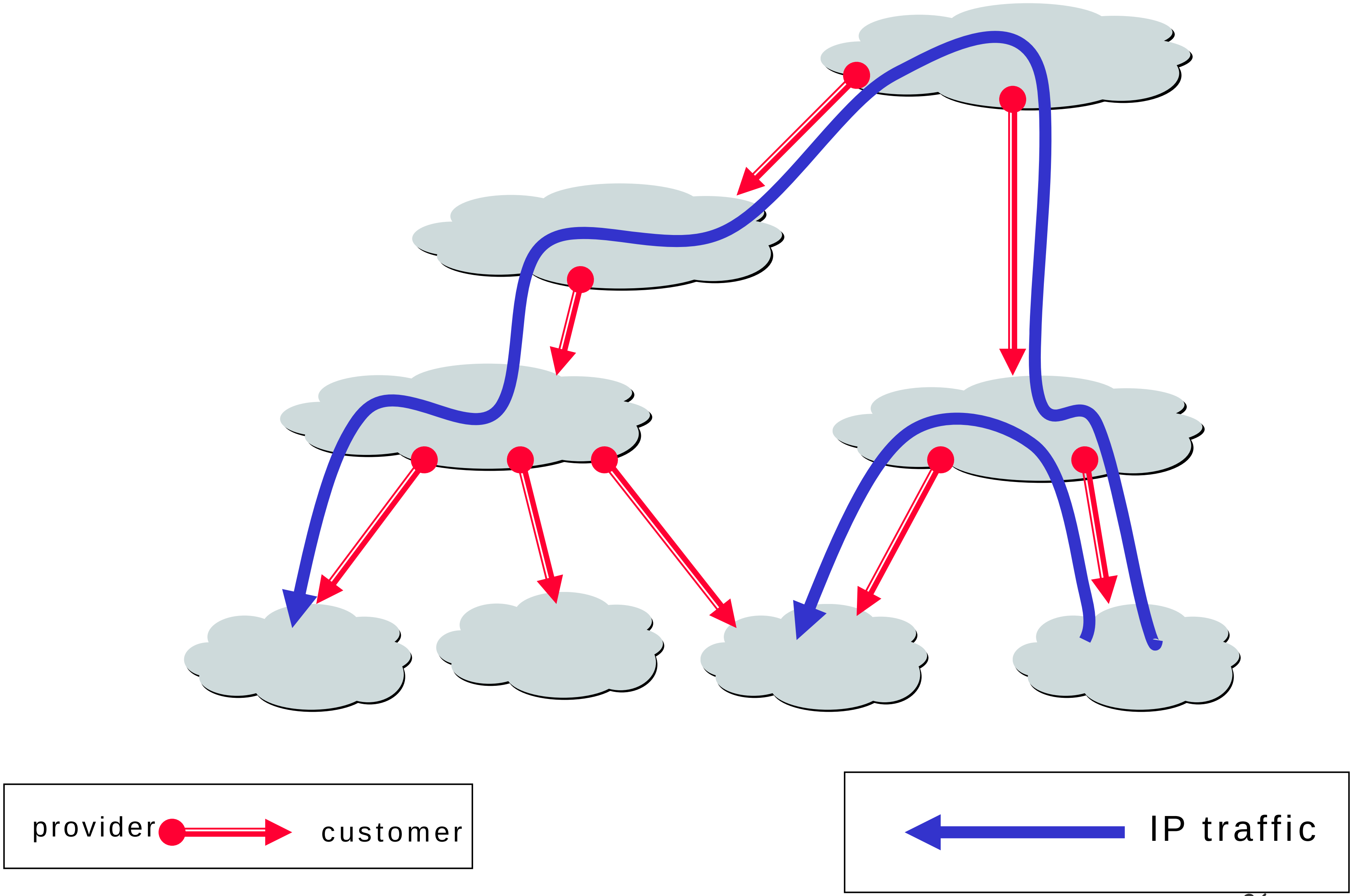
Inter-AS Relationship: Transit vs. Peering



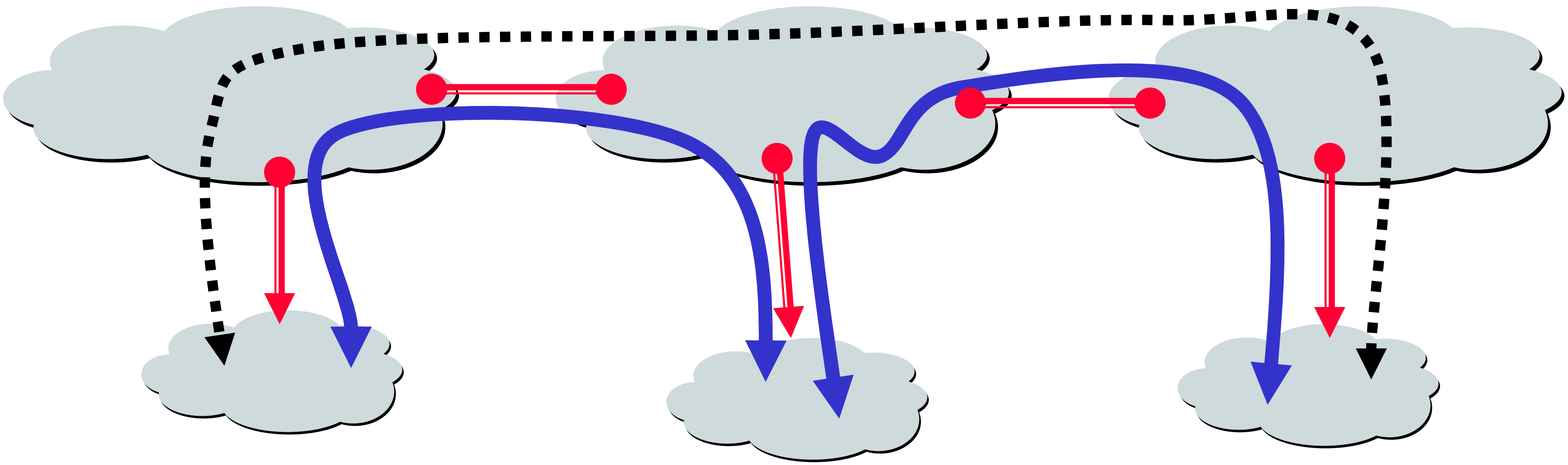
Policy Impact on Routing

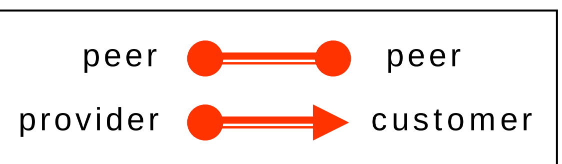
- AS relationships
  - Customer-provider
  - Peers
- Want "Valley-free" routes
  - Number links as (+1, 0, -1) for provider, peer and customer links
  - In any path, you should only see sequence of +1, followed by at most one 0, followed by sequence of -1

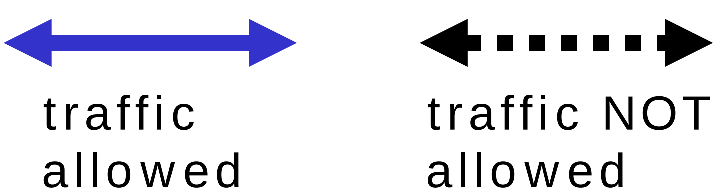
Customer-Provider Hierarchy



The Peering Relationship





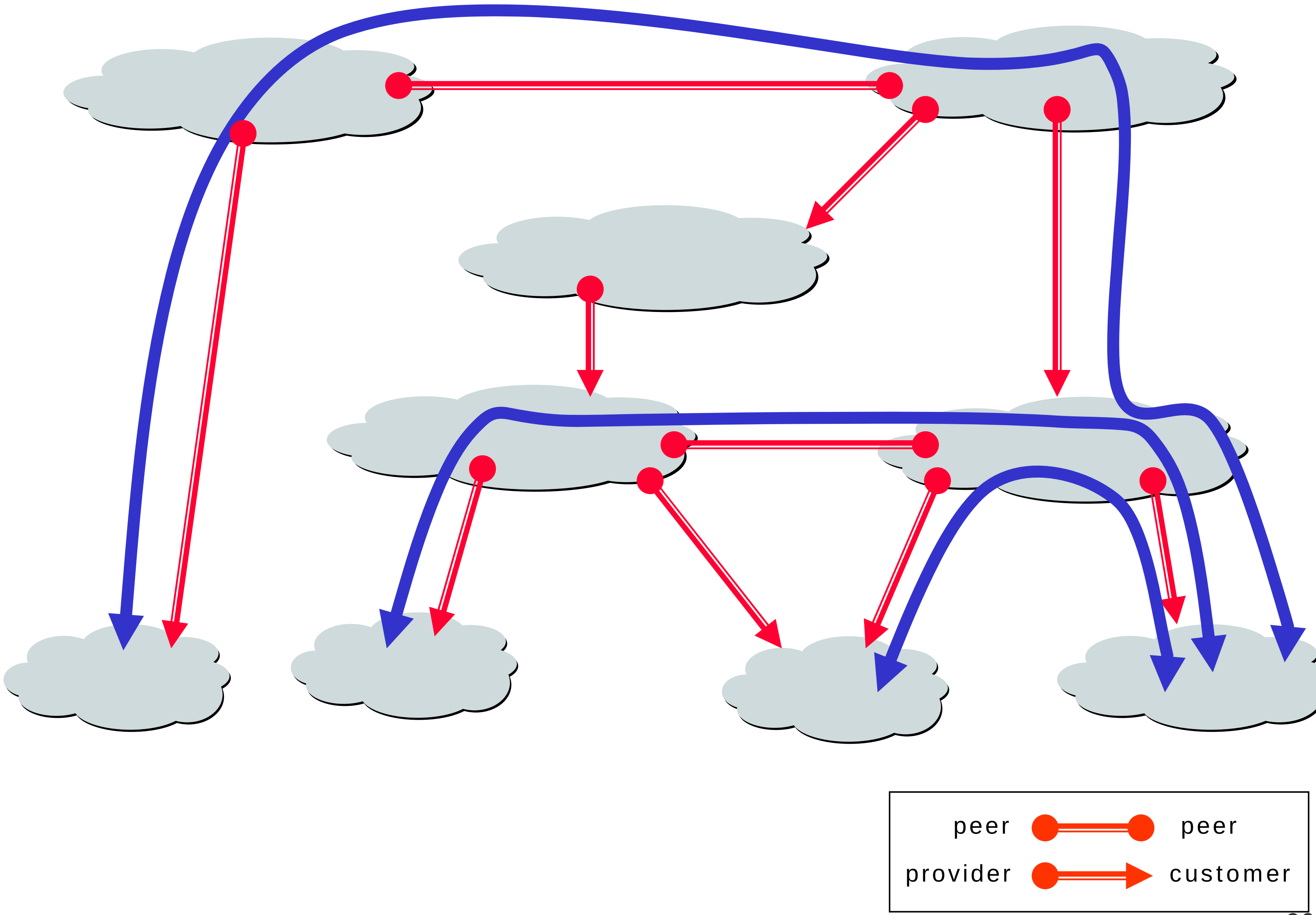


Peers provide transit between their respective customers

<u>Peers do not provide transit between peers</u>

Peers (often) do not exchange $$$

Peering Provides Shortcuts



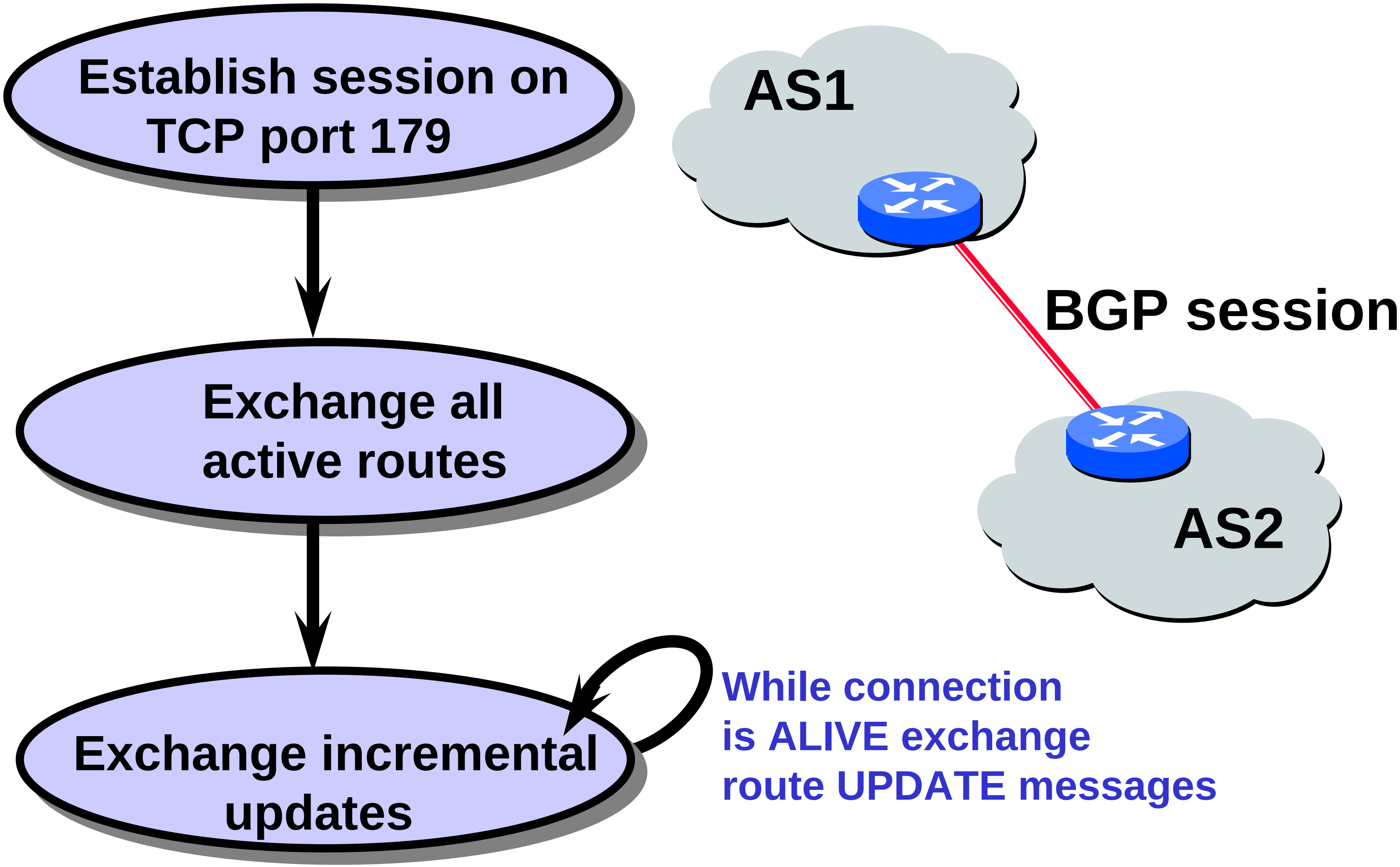
Policy-Based Routing

- Policies are used to force customer-provider-peer relationships, backup links, load balancing, …
- Can't use shortest path routing
  - No universal metric – policy-based decisions
  - Main characteristic of shortest path does not hold (i→x→j is shortest route, then x→ j is shortest route)
- Problems with distance-vector:
  - Bellman-Ford algorithm may not converge, and may loop
- Problems with link state:
  - Metric used by different routers are not the same → loops
  - LS database too large – entire Internet
  - May expose policies to other AS's

BGP: Distance Vector with Path

- Each routing update carries the entire path
  - e.g.,: destination 18.26/16 is reachable using {AS1, AS3, AS11}
- When AS receives a routing update
  - Reject routes with loops
    - To detect loops check whether my AS is already in path
- AS remembers loop-free routes
- For each destination, the AS chooses the best route according its policies.
- AS advertises a neighbor routes to a subset of all the destinations, depending on its policy
  - E.g., I might hide from you that I know how to get to destination X, because I don't want to deliver your messages to X
- AS advertises to neighbors only those routes that it uses
  - Ensures that if i→x→j is the used route, then x→j is the used route
  - *What happens if an AS advertises routes that it doesn't use?*
- Advantage:
  - Metrics are local - AS chooses path, protocol ensures no loops

BGP Operations (Simplified)



Four Types of BGP Messages

- **Open** : Establish a peering session.
- **Keep Alive** : Handshake at regular intervals.
- **Notification** : Shuts down a peering session.
- **Update** : Announcing new routes or withdrawing previously announced routes.
- Announcement = prefix + attributes
  - Attributes are used to choose among all routes for the same prefix
  - Example attribute is AS_PATH, which is used to discover loops

Implementing Customer/Provider and Peer/Peer relationships using BGP

- BGP provides capability for enforcing various policies
- Policies are **<u>not</u>** part of BGP: they are provided to BGP as configuration information
- BGP enforces policies by
  1. choosing paths from multiple alternatives (importing routers)
  2. controlling advertisement to other AS's (exporting routes)

Importing Routes

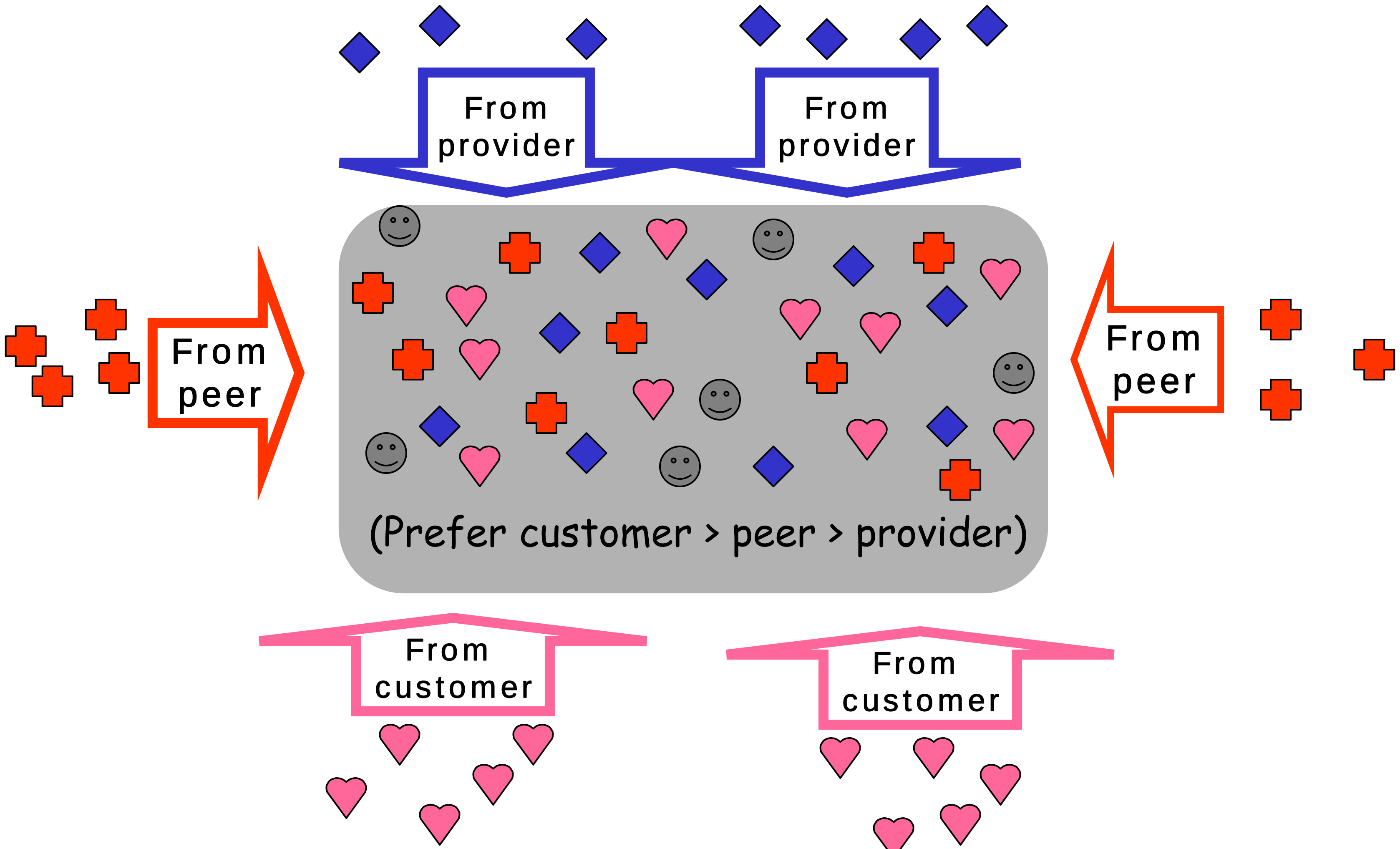
- Based on route attributes
  - First, Prefer customer > peer > provider
  - Then, Shortest AS PATH length
  - Then, look at other route attributes

Exporting Routes

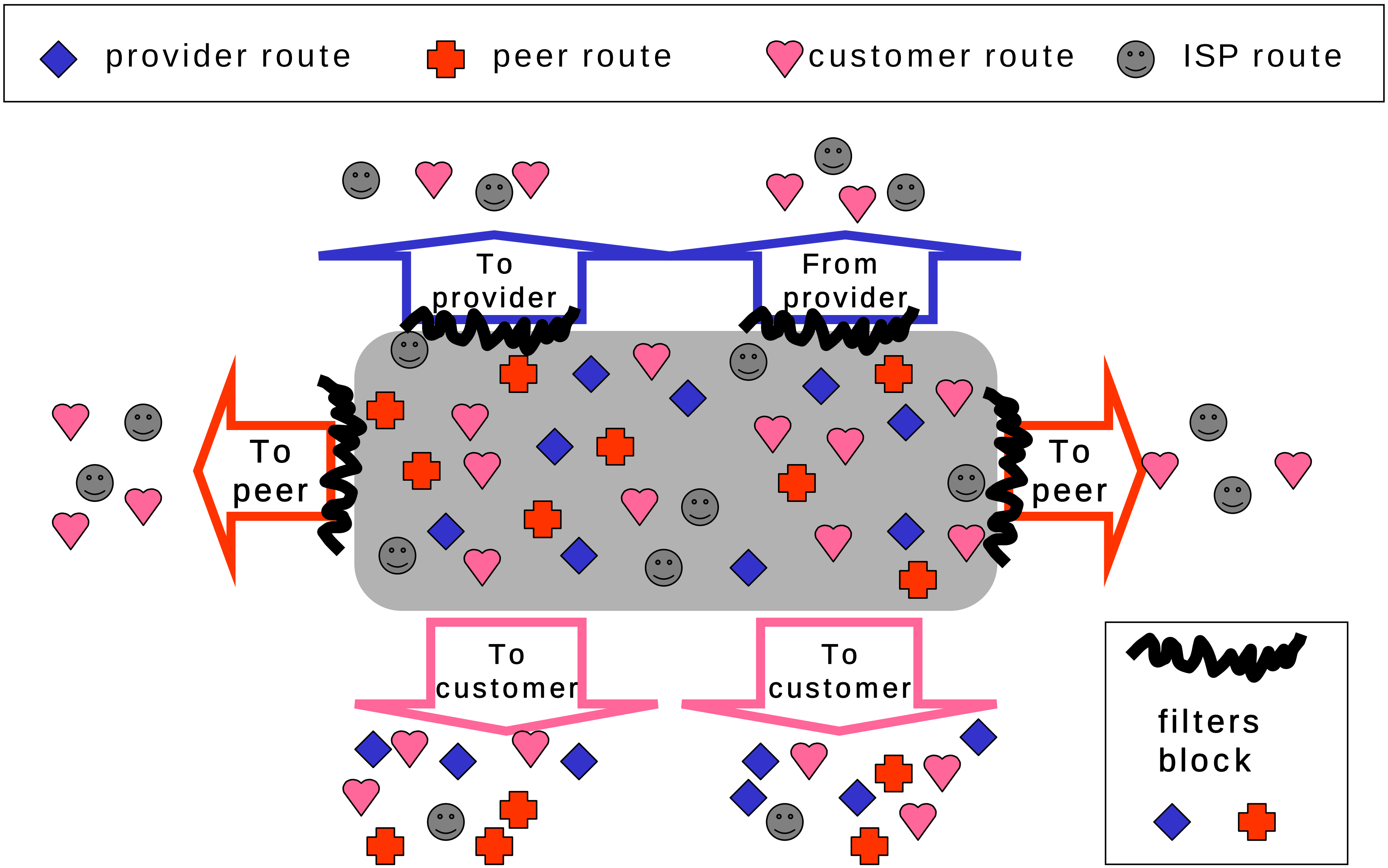
- When an AS exports a route, others can use the AS to forward packets along that route
- Rules:
  - Export customers routes to everyone
    - why?
  - Export routes to your own addresses to everyone
    - Why?
  - Don't export routes advertised to you by your provider (may advertise them to customers)
    - Why?
  - Don't export routes advertised to you by your peer (may advertise them to customers)
    - Why?

Import Routes

provider route　　peer route　　customer route　　ISP route



Export Routes



What problem is BGP solving?

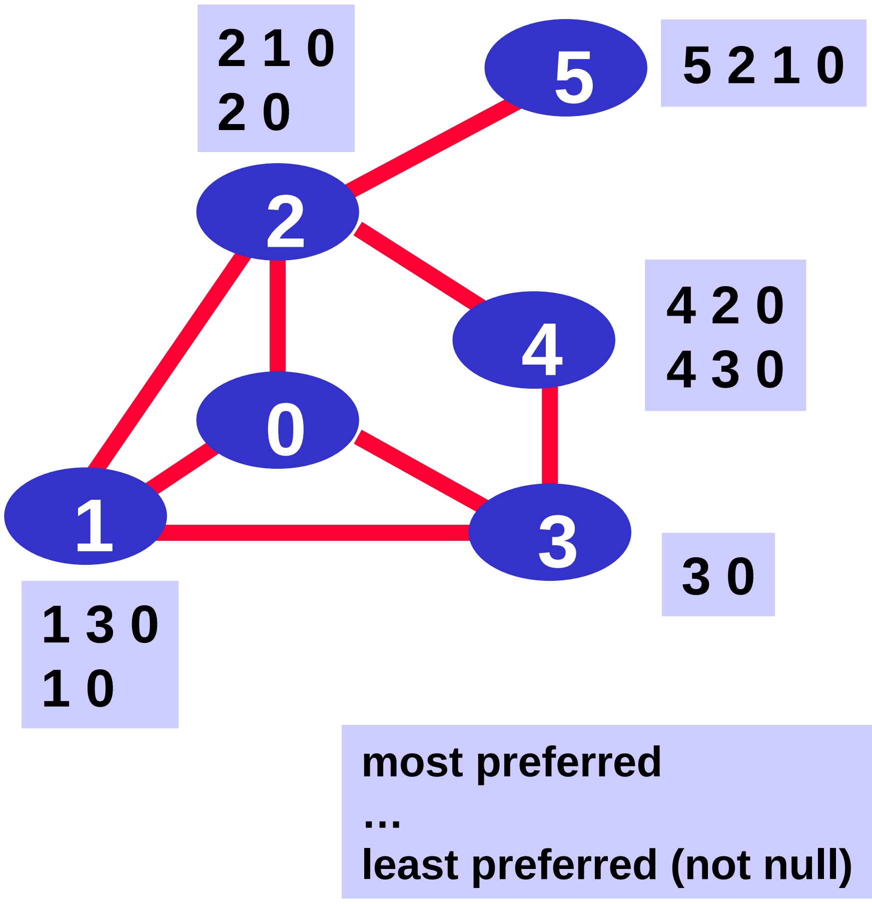
Underlying problem	Distributed means of computing a solution.
Shortest Paths	RIP, OSPF, IS-IS
X?	BGP

X could

- aid in the design of policy analysis algorithms and heuristics
- aid in the analysis and design of BGP and extensions
- help explain some BGP routing anomalies
- provide a fun way of thinking about the protocol

An instance of the *Stable Paths Problem* (SPP)

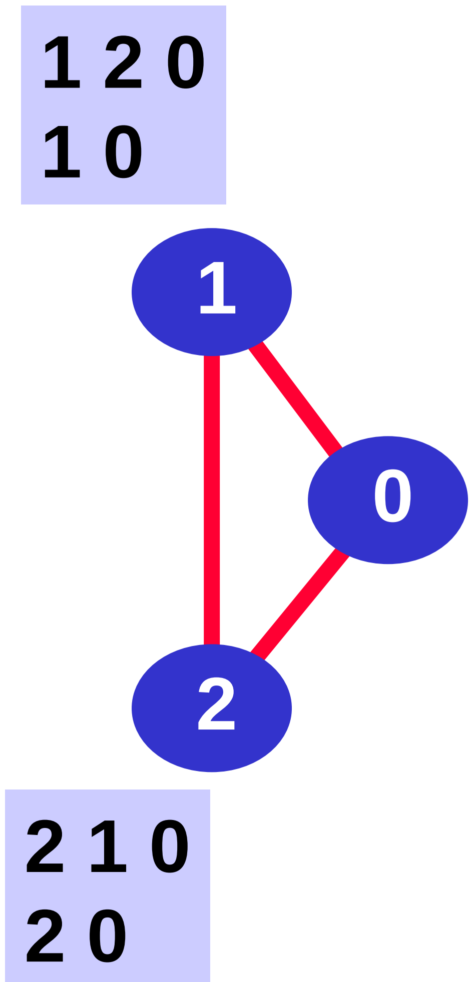
- *A graph of nodes and edges,*
- Node 0, called *the origin,*
- For each non-zero node, a set or permitted paths to the origin. This set always contains the "null path".
- *A ranking of permitted paths at each node.*



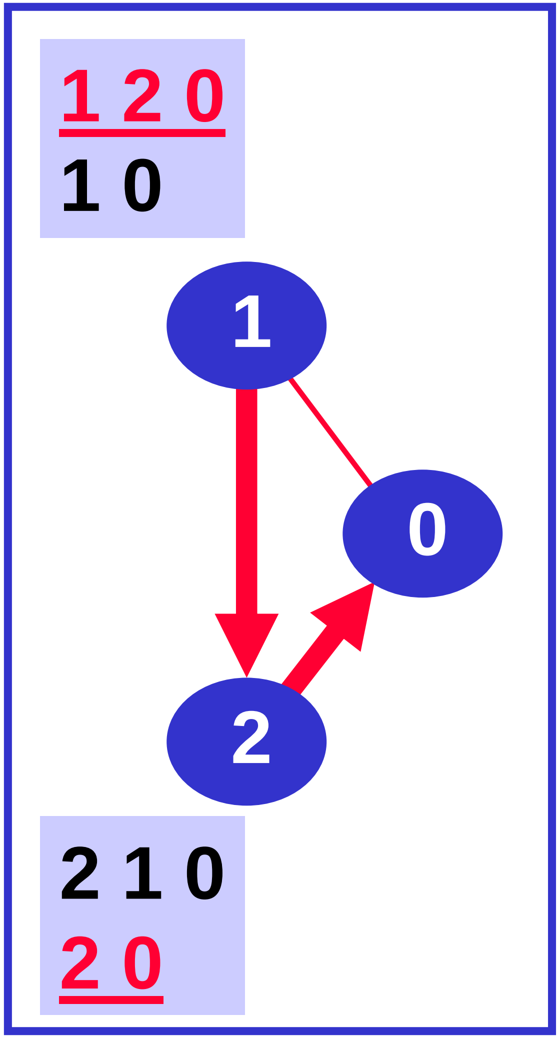
When modeling BGP : nodes represent BGP speaking routers, and 0 represents a node originating some address block

Yes, the translation gets messy!

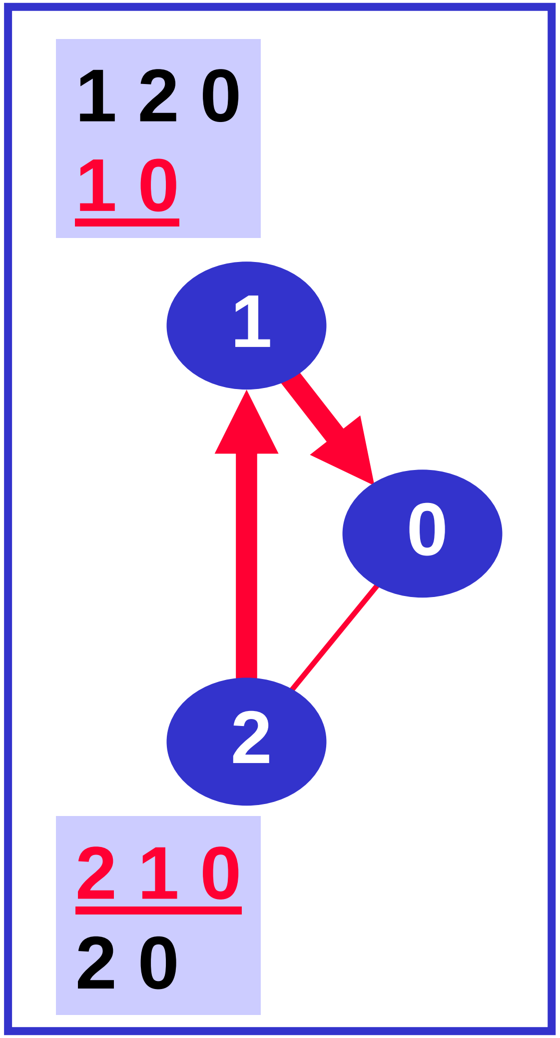
An SPP may have multiple solutions



DISAGREE

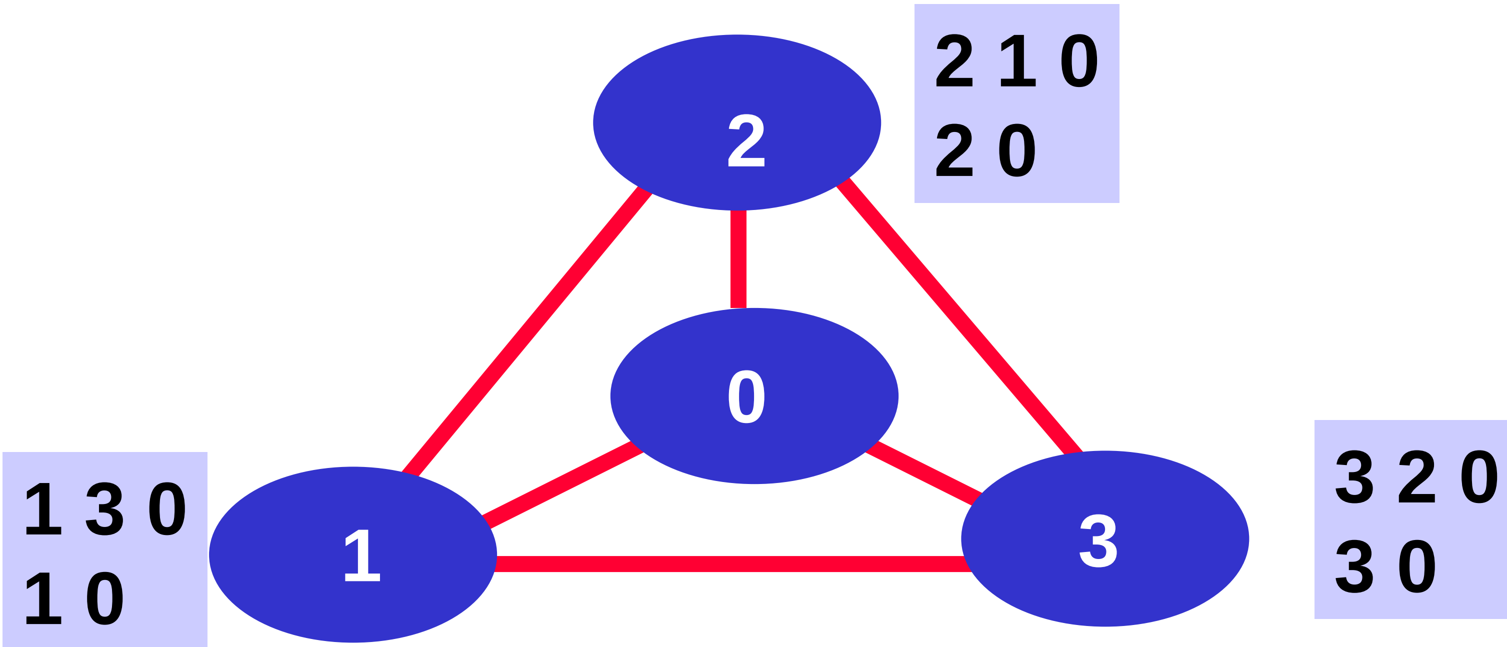


First solution



Second solution

BAD GADGET : No Stable Solution

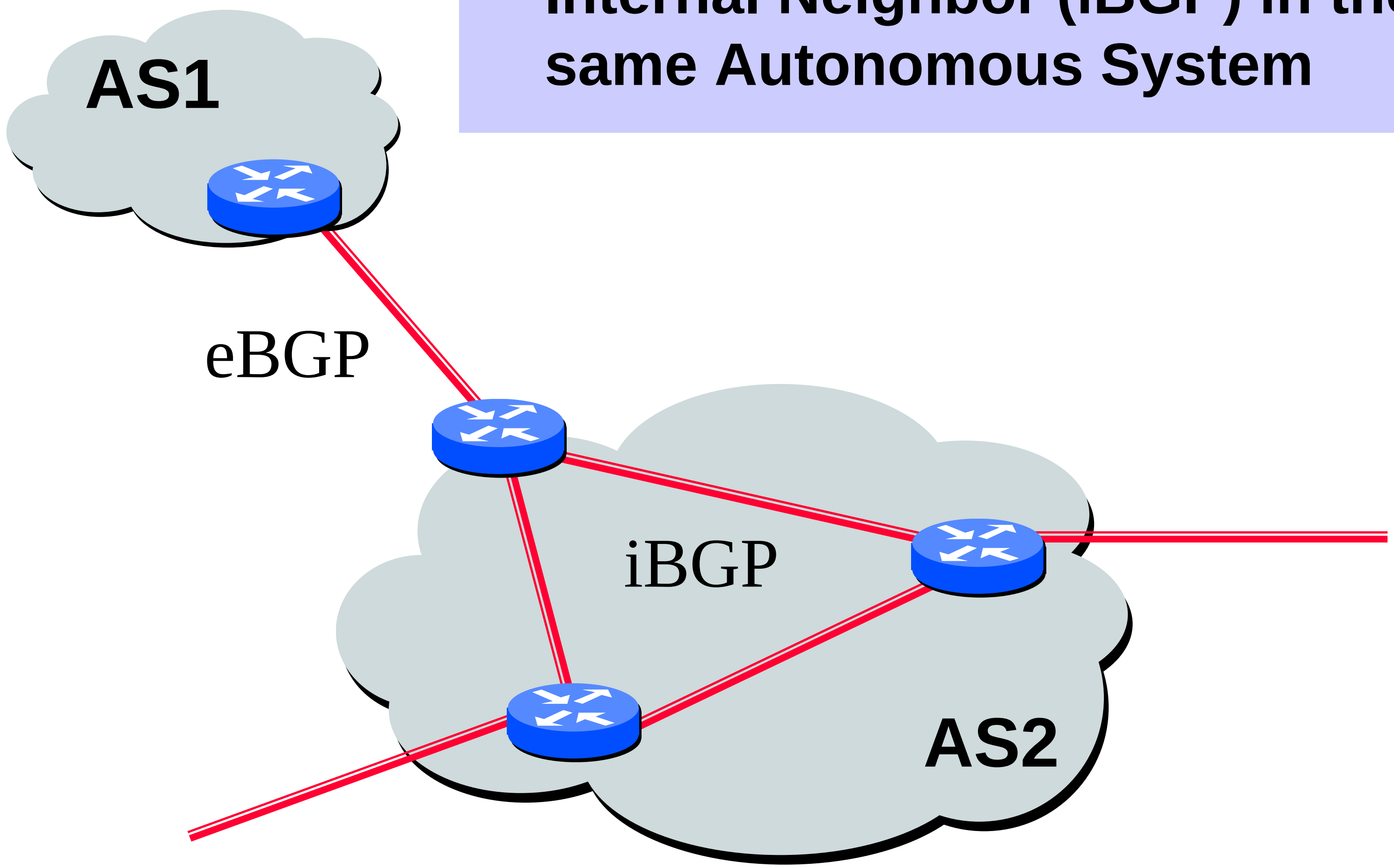


Known Results

- Internet routing has serious convergence problems
  - Result 1 [Griffin et al.]: BGP does not satisfy the stable paths problem.
  - Result 2 [Rexford et al.]: If every AS follows a set of guidelines (valley-free routes) then Internet routing should not have convergence problems.

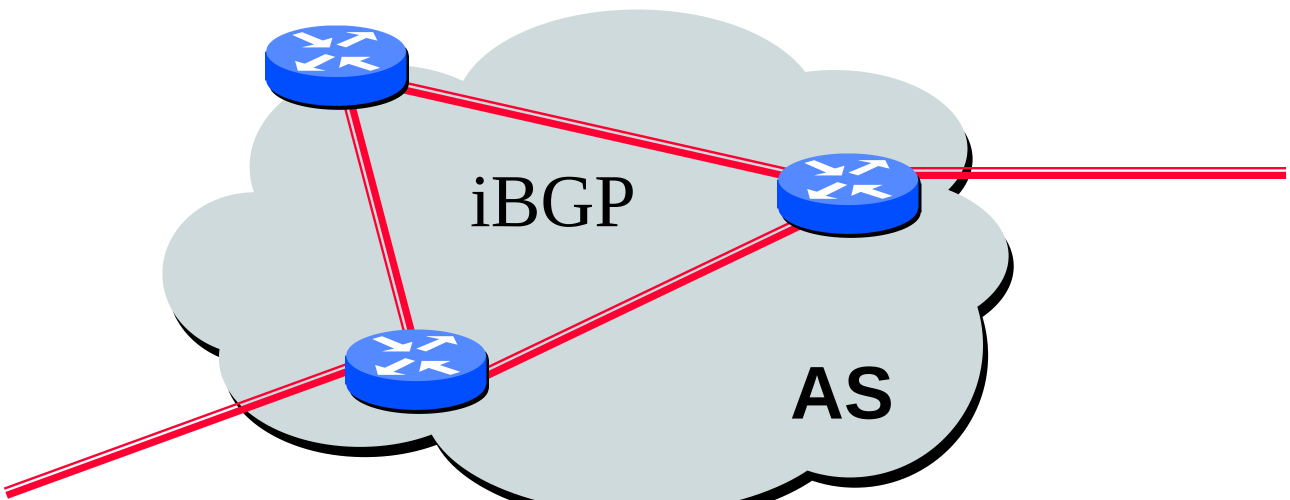
Two Types of BGP Neighbor Relationships

- **External Neighbor (eBGP) in a different Autonomous Systems**
- **Internal Neighbor (iBGP) in the same Autonomous System**



iBGP

- AS has more than one router participating in eBGP
- iBGP is run between BGP routers in the same AS to allow all of them to obtain a complete and consistent view of external routes



Internal BGP (iBGP)

- Same messages as eBGP
- Different rules about re-advertising prefixes:
  - Prefix learned from eBGP can be advertised to iBGP neighbor and vice-versa, but
  - Prefix learned from one iBGP neighbor cannot be advertised to another iBGP neighbor
    - Reason: no AS PATH within the same AS and thus danger of looping.

We learned

- Inter-domain routing uses policy
- As a result, routing is not a simple optimization of a single number which can be done using shortest path algorithms
- BGP is designed to route based on policies