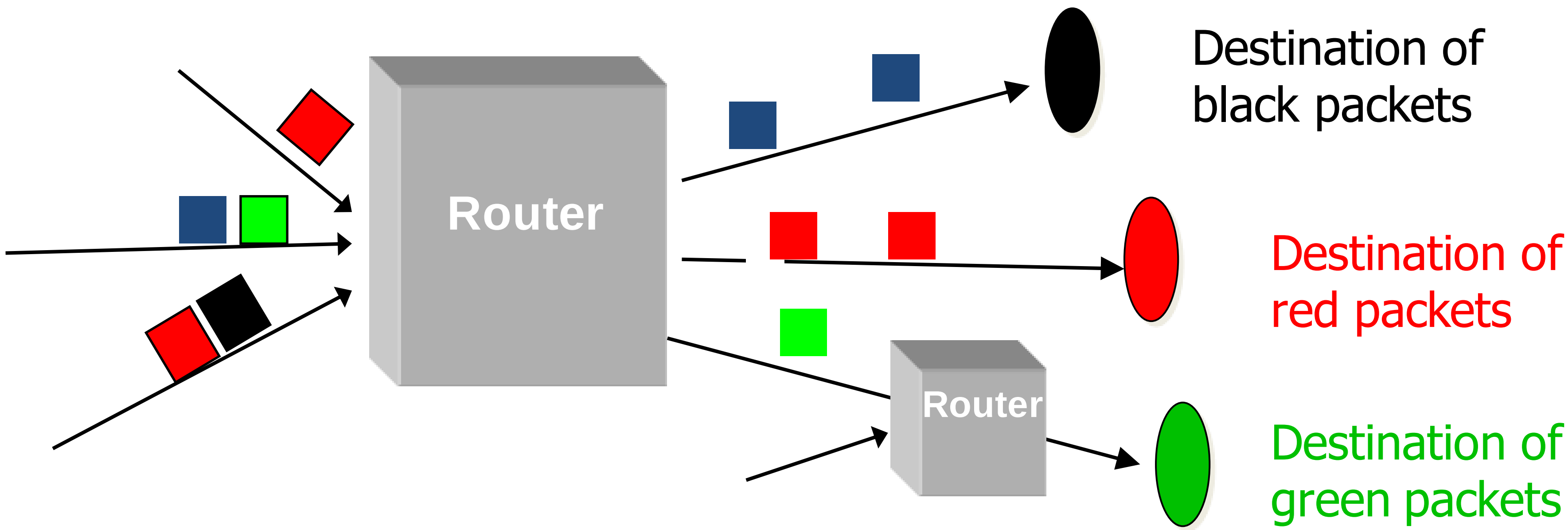
Computer Networks
Layering and Routing

Dina Katabi

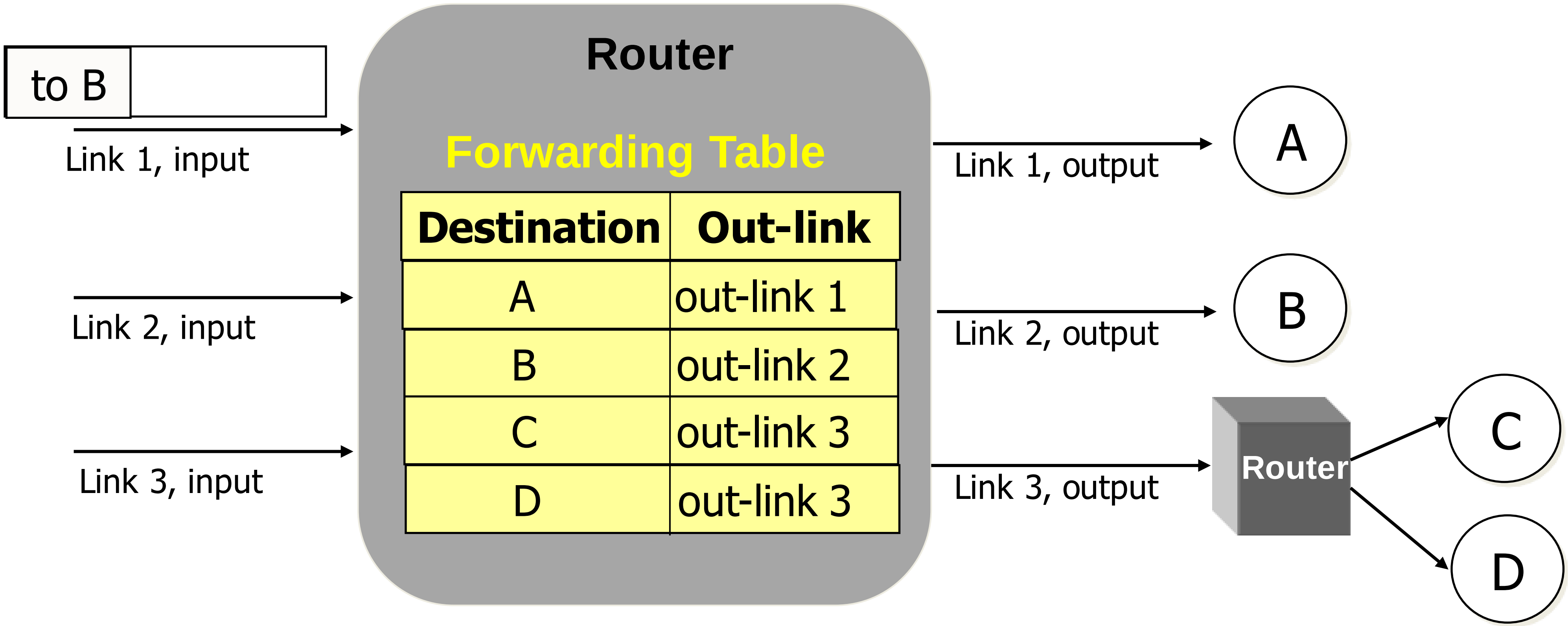
dk@mit.edu

The Job of a Router



- A router has input links and output links
- A router sends an input packet on the output link leading toward the packet's destination node
- A router does not care of who generated the packet

How does the router know which output link leads to a packet destination?



- Packet header has the destination
- Router looks up the destination in its table to find output link
- Table is built using a routing protocol

Distance Vector Routing

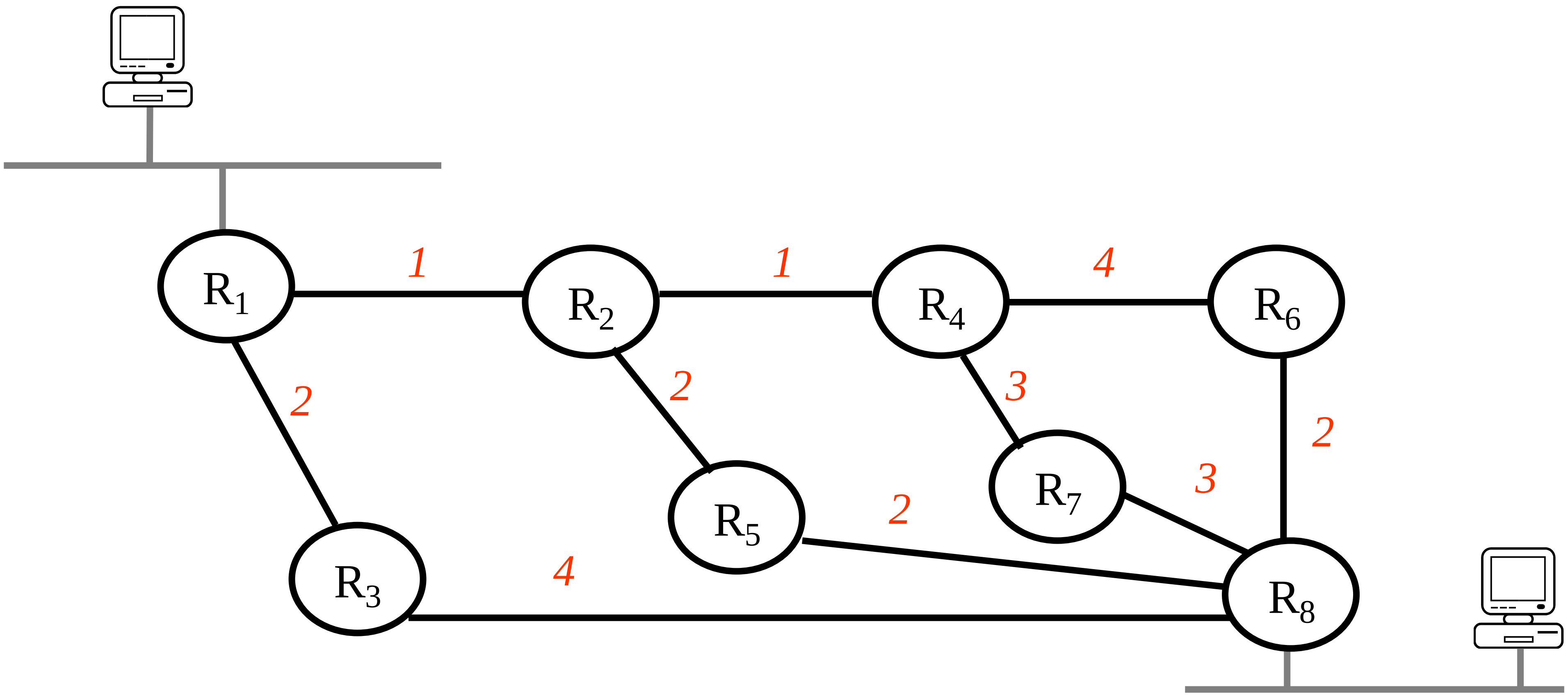
- Initialize
  - Distance to self is zero and next hop is self
  - Distance to anyone else is infinity

- Announce: Every T seconds
  - Tell neighbors distances to all destinations

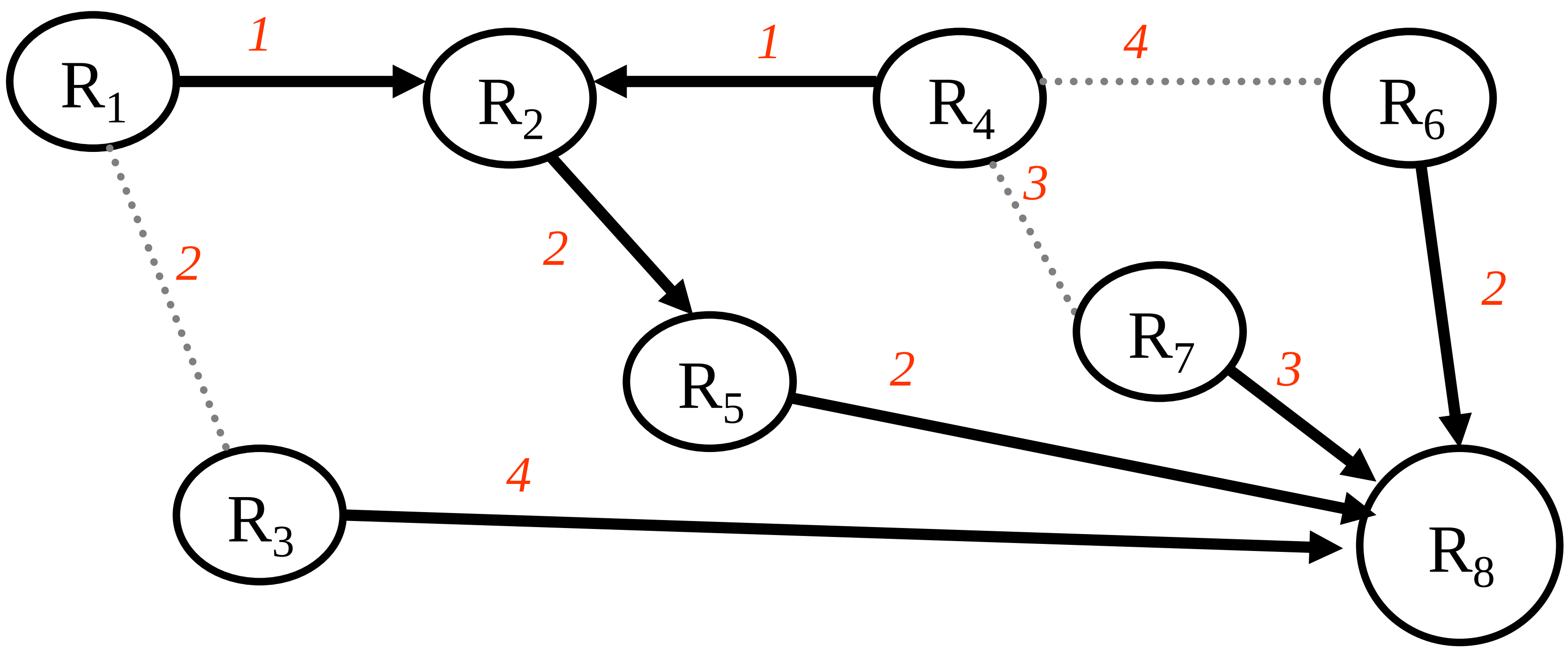
- Update route to dst. upon message from j
  - Distance via j = j's distance + weight of link to j
  - If distance via j is shorter than current distance, update routing table to go via j

Example

<u>Objective:</u> Determine the route from (R_1, …, R_7) to R_8 that minimizes the distance



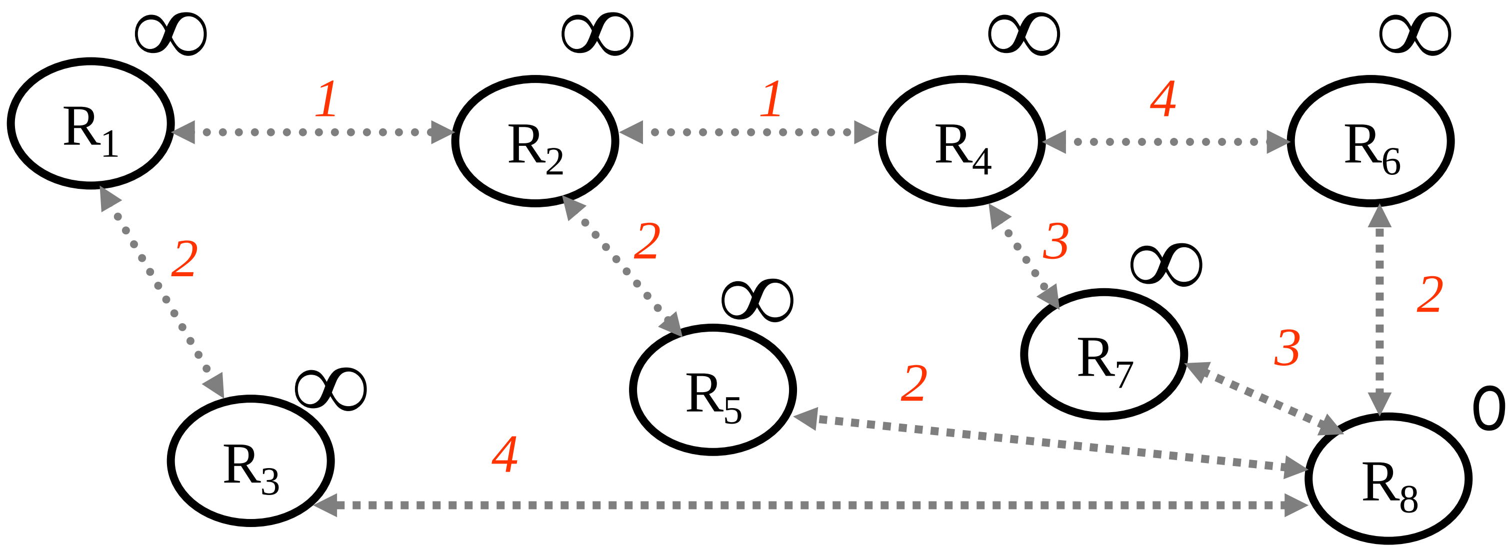
Solution is simple by inspection... (in this case)



The shortest paths from all sources to a destination (e.g., R_8) is the spanning tree routed at that destination.

Distance Vector Routing

Example

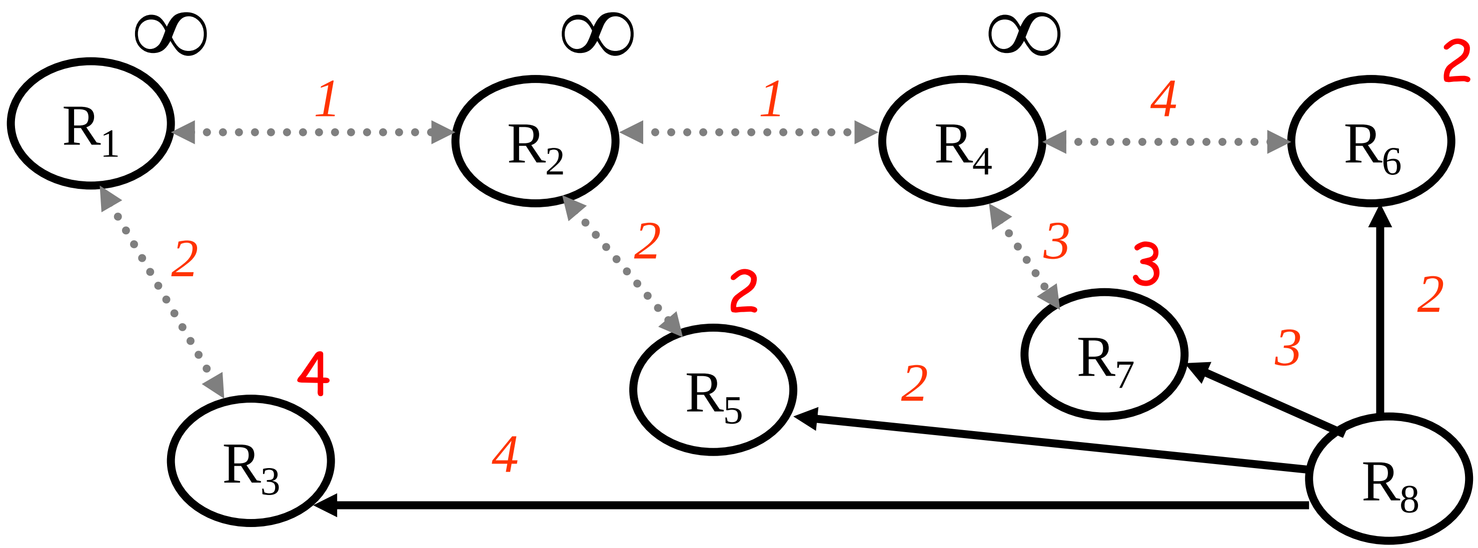


<u>Initial State:</u> All routers except R8 set their route length to ∞. R8 sets its route length to 0.

Distance Vector Routing

Example

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



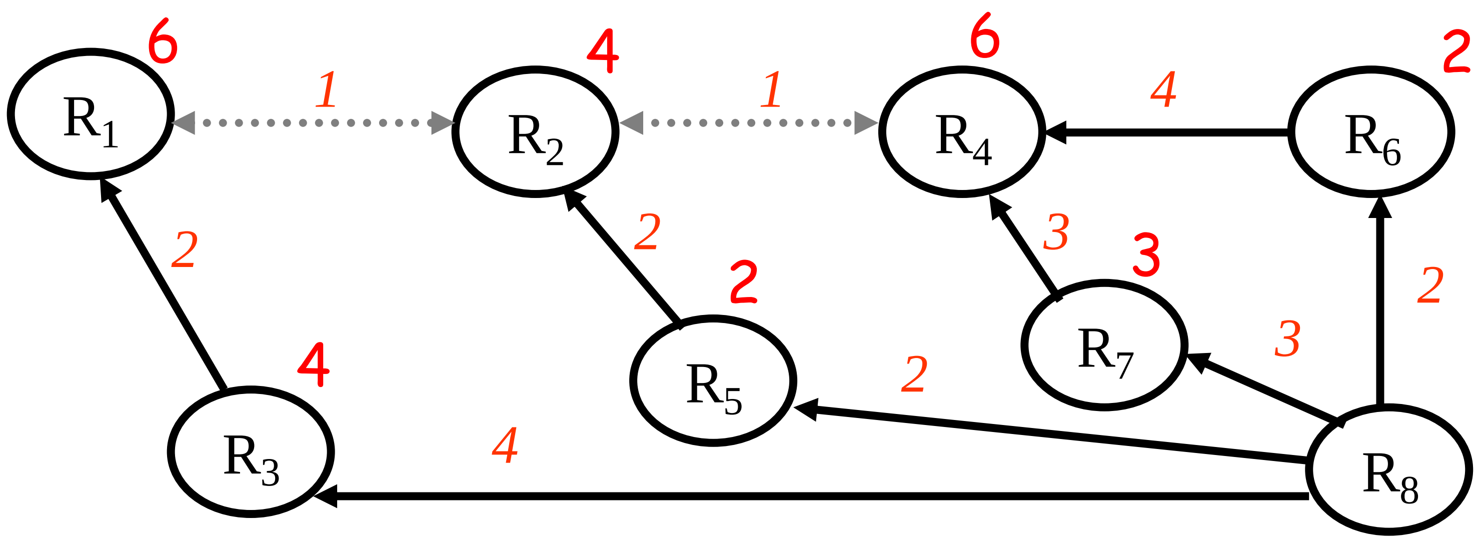
- Every T seconds, Router *i* tells its neighbors about its current lowest-cost path to R8
- Each router updates its distance as *min(current distance, received distance + link weight)*

Note, routing tables have both the next-hop and the distance

Distance Vector Routing

Example

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

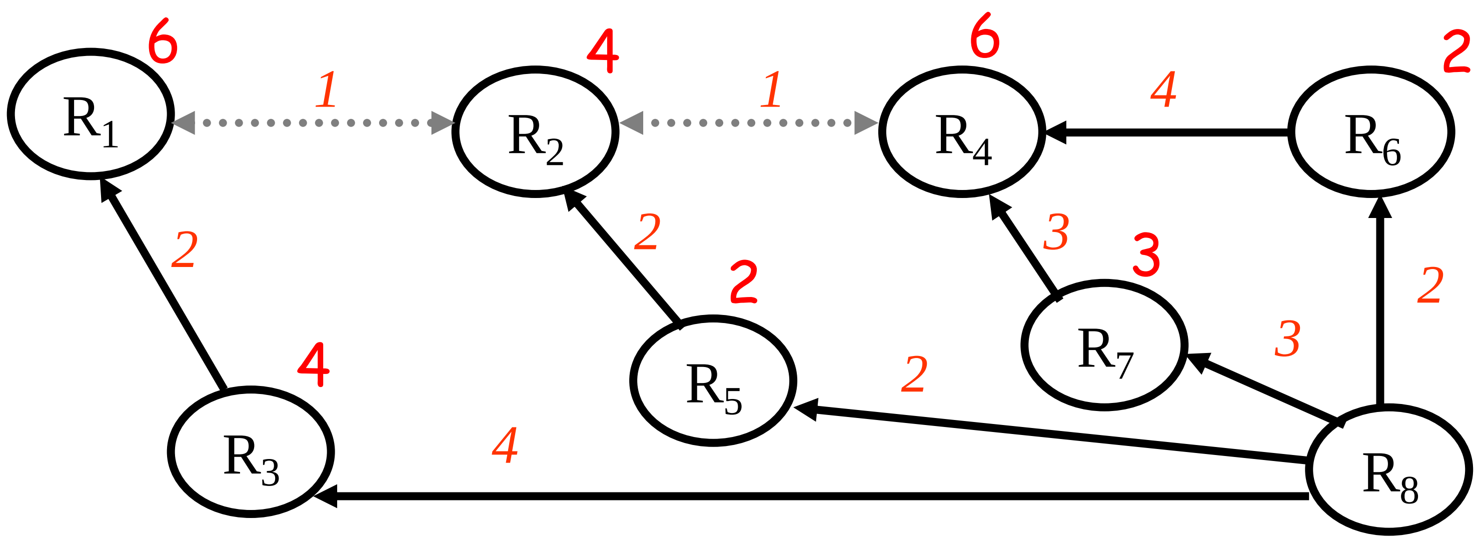


Repeat until no distance change

Distance Vector Routing

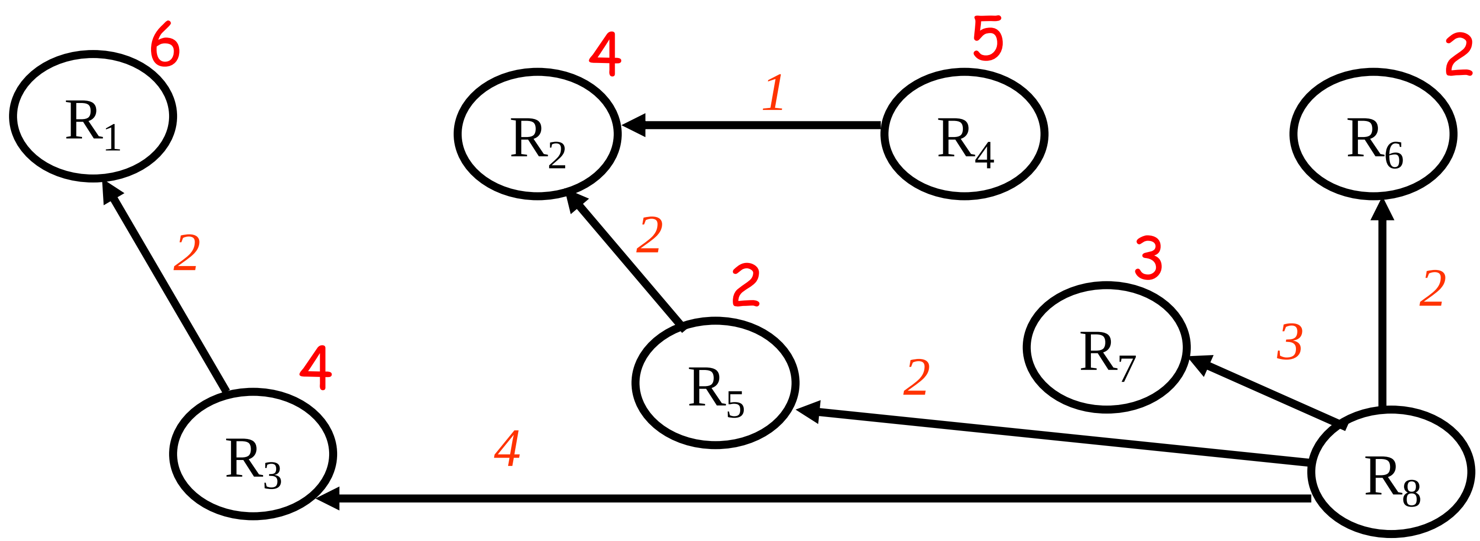
Example

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

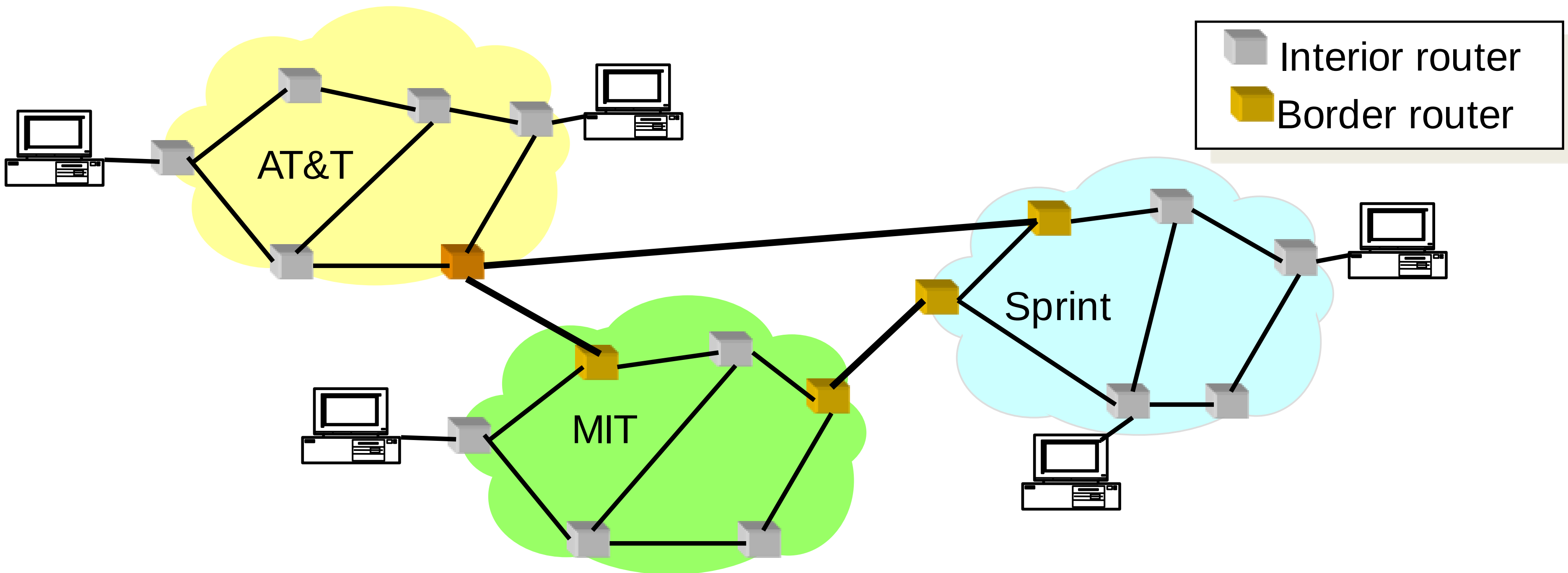


Distance Vector however cannot scale to the whole Internet

Requirements of Internet-Wide Routing

- Scalability
  - Small routing tables: Cannot have an entry per machine → causes large look up delay
  - Small message overhead and fast convergence: A link going up or down should not cause routing messages to spread to the whole Internet
- Policy-compliant
  - Shortest path is not the only metric; Internet Service Providers (ISPs) want to maximize revenues!

Scaling Routing to the Internet



- The Internet is a network of Domains of Autonomous Systems (ASs)
  - E.g., MIT, AT&T, Stanford, …
- Internally, each AS runs its own routing protocol (e.g., Distance Vector) → Autonomy
- Across ASs, we run a different routing protocol (called BGP)

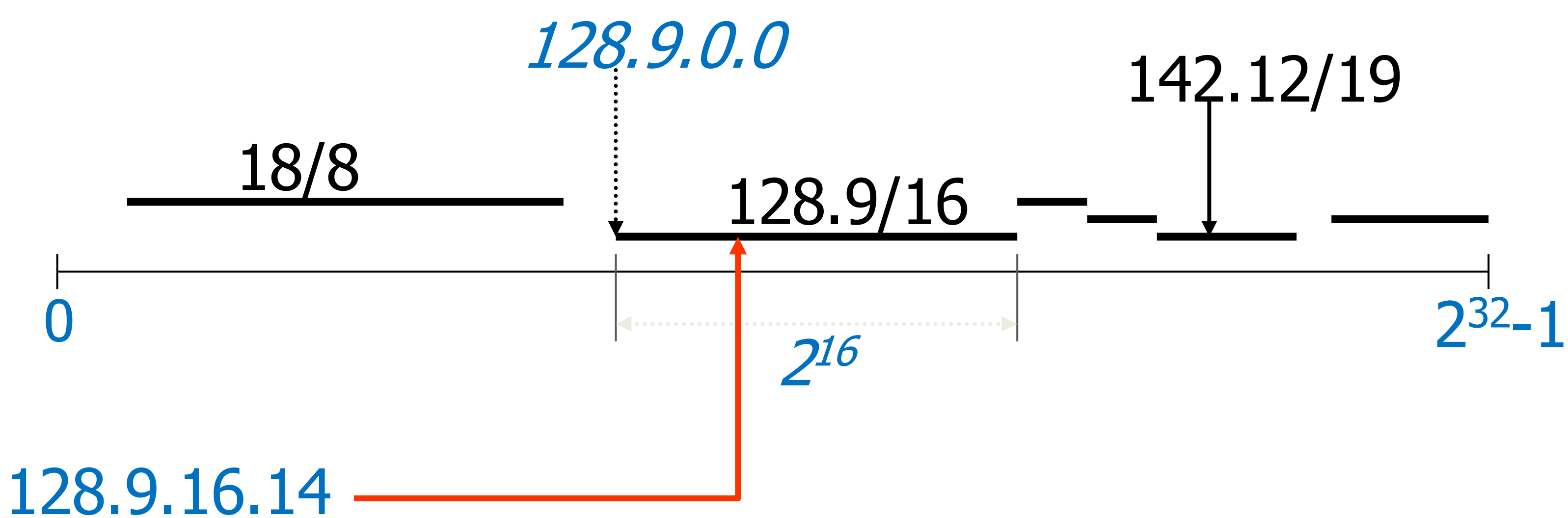
Idea for Scaling

- Need less information with increasing distance to destination

→ Hierarchical Routing and Addressing

Hierarchical Addressing

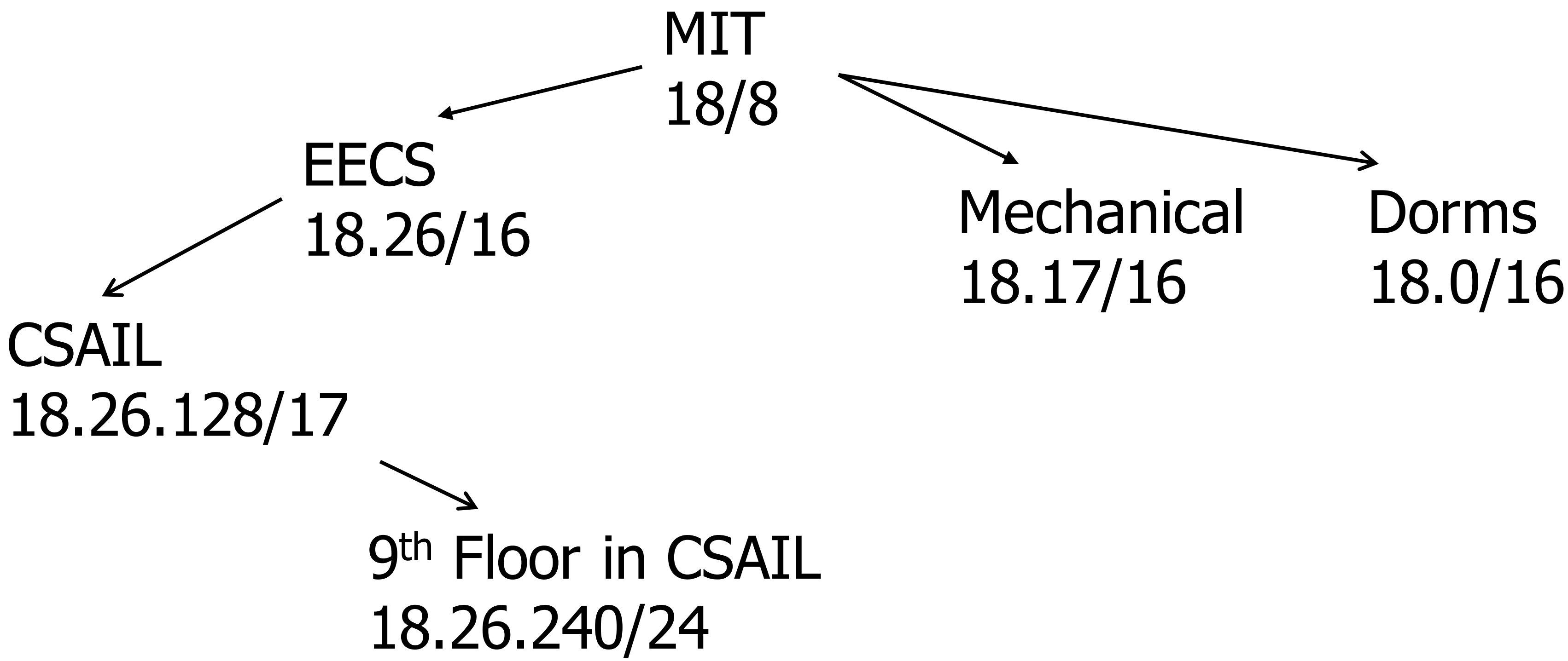
- Each IP address is 4 bytes, e.g., 18.0.1.2
- The IP address space is divided into segments of contiguous chunk of addresses; each such segment is described by a *prefix.*
- A prefix is of the form x/y where x is the prefix of all addresses in the segment, and y is the length of the segment in bits
- e.g. the prefix 128.9/16 represents the segment containing addresses in the range: 128.9.0.0 … 128.9.255.255.



Hierarchical Address Allocation

- Addresses that start with same prefix are co-located
  - E.g., all addresses that start with prefix 18/8 are in MIT
- Entries in the routing/forwarding table are for IP prefixes → shorter routing tables

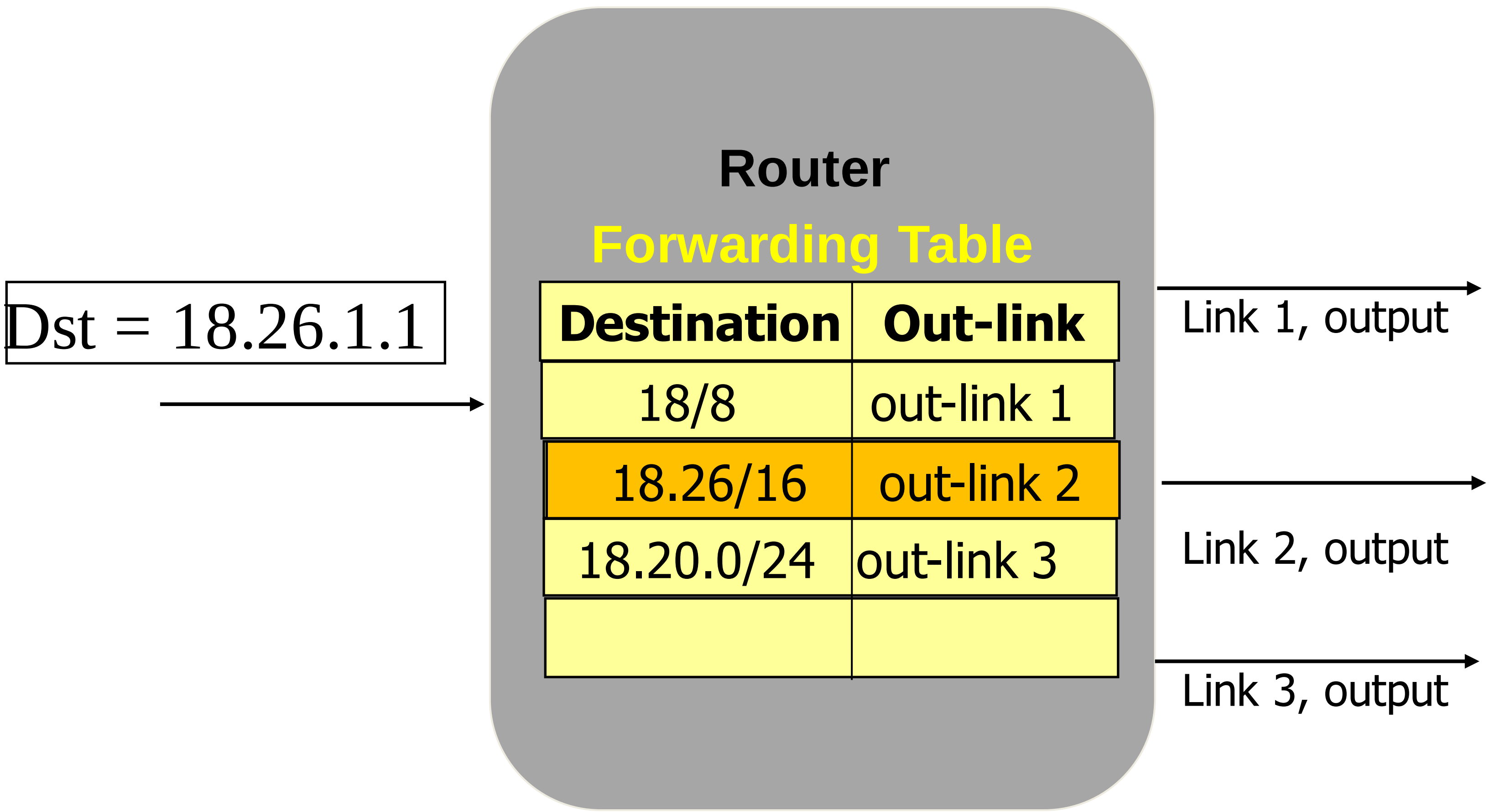
Hierarchical Addressing



- Forwarding tables in Berkeley can have one entry for all MIT's machines. E.g., (18/8, output-link)
- Forwarding tables in Mechanical Engineering have one entry for all machines in EECS
- But, a switch on the 9th floor subnet knows about all machines on its subnet

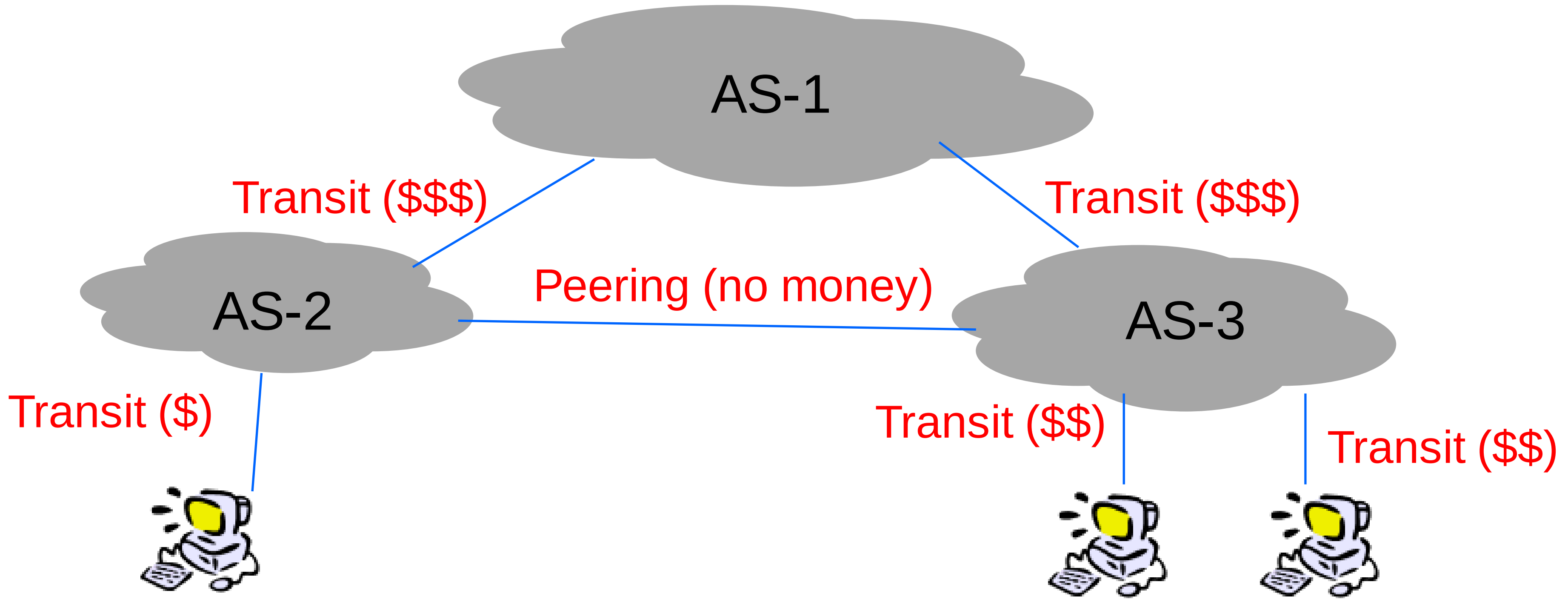
Longest Prefix Match

A Router forwards a packet according to the entry in the forwarding table that has the longest matching prefix



- Hierarchical addressing and routing give us scalability
- Still need to tackle policies

Inter-AS Relationship: Transit vs. Peering

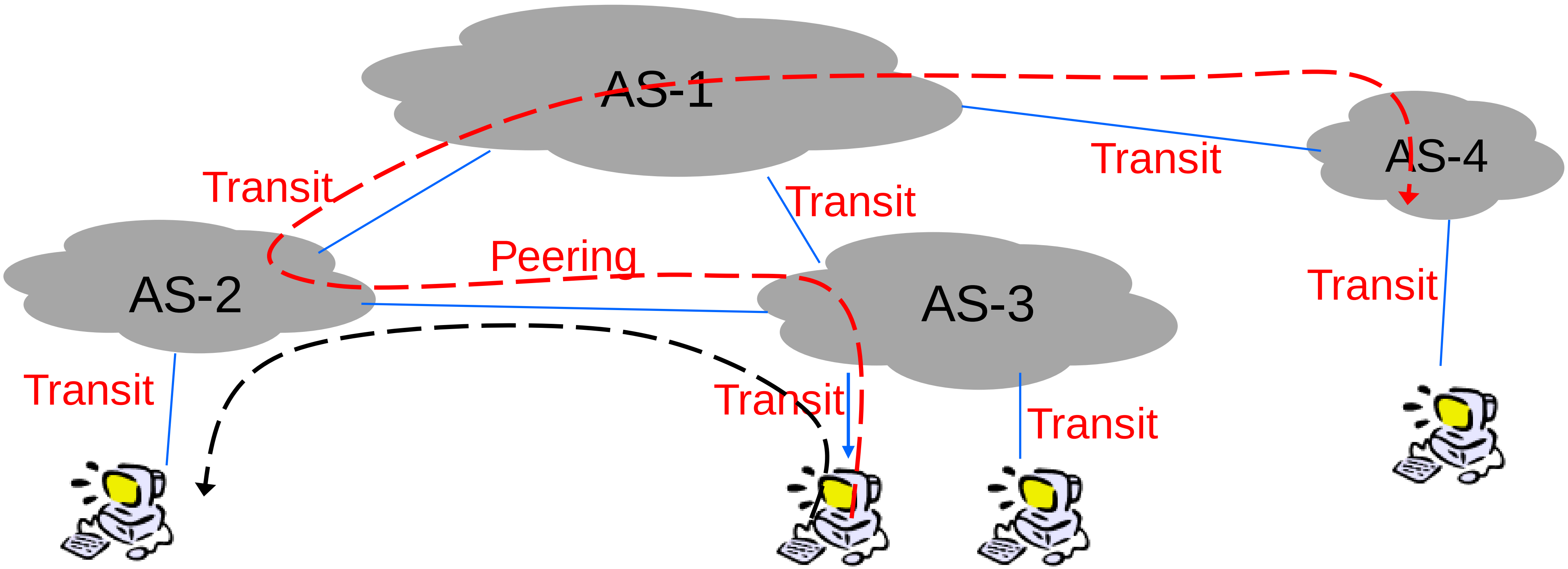


- Transit relationship
  - One AS is a customer of the other AS, who is the provider; customer pays provider both for sending and receiving packets
- Peering relationship
  - Two ASs forward packets for each other without exchanging money

Policy-Based Routing

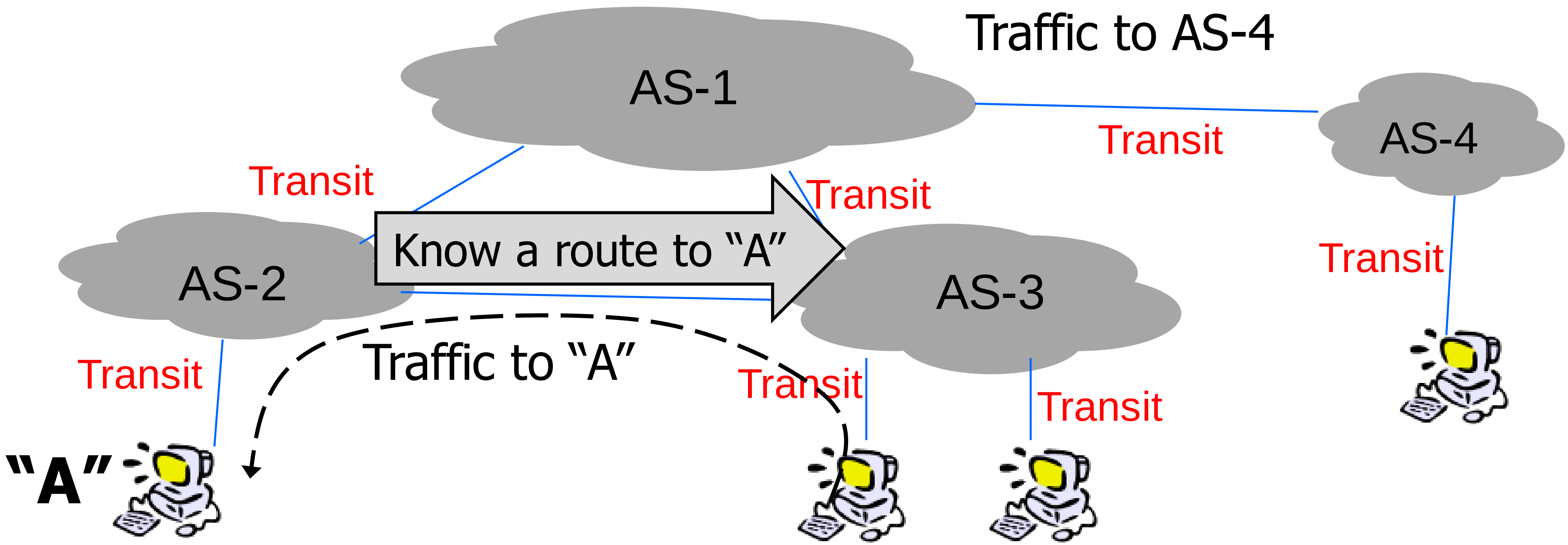
- Main Rule:
  - An AS does not accept transit traffic unless it makes money of it
- Rule translates into incoming and outgoing routing policies

Desirable Incoming Policies



- AS-2 likes AS-3 to use the peering link to exchange traffic between their customers → saves money because it bypasses AS-1
- But, AS-2 does not want to forward traffic between AS-3 and AS-4 because this makes AS-2 pay AS-1 for traffic that does not benefit its own customers

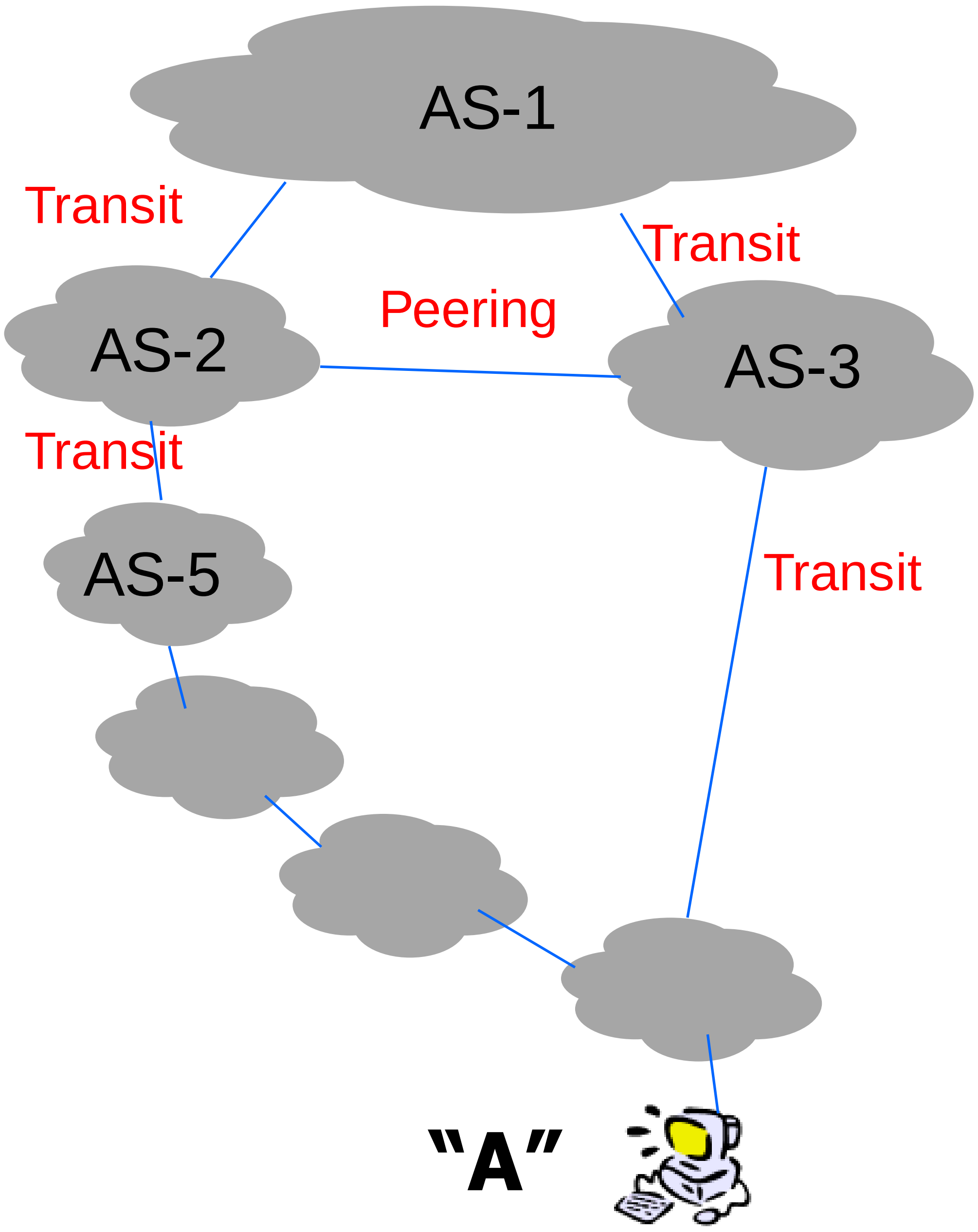
How Does AS-2 Control Incoming Traffic?



- AS-2 advertises to AS-3 a route to its customer's IP prefix
- AS-2 does not tell AS-3 that it has a route to AS-4, i.e., it does not tell AS-3 routes to non-customers IP-prefixes

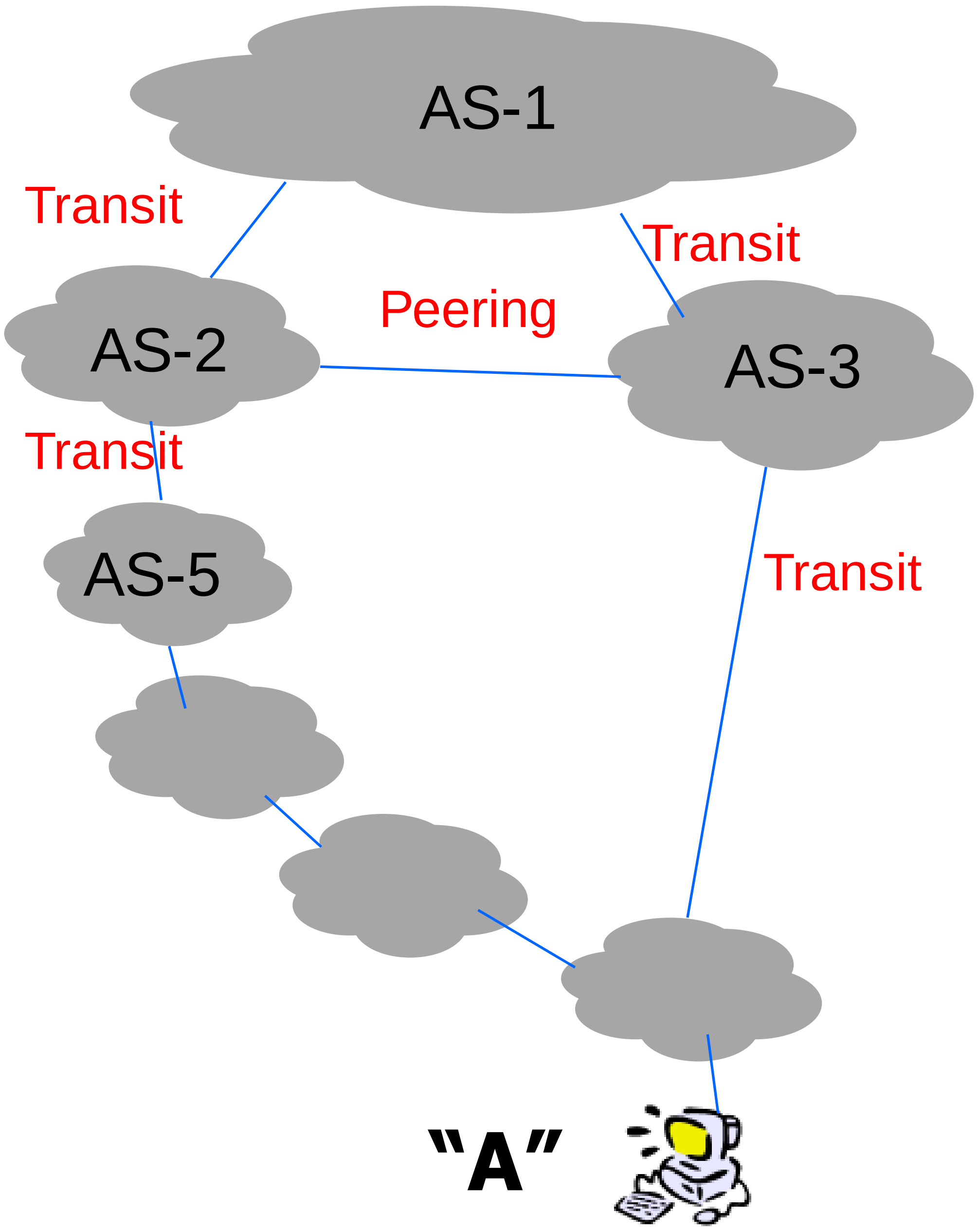
Desirable Outgoing Policies

- AS-2 prefers to send traffic to "A" via its customer AS-5 rather than its provider or peer despite path being longer



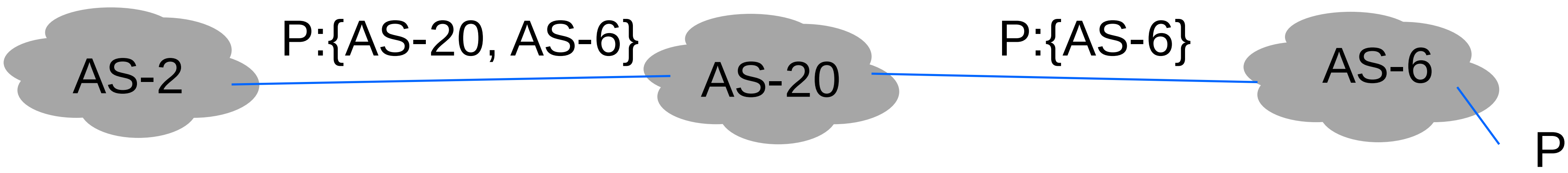
How Does AS-2 Control Outgoing Traffic?

- AS-1, AS-3, and AS-4 advertise their routes to “A” to AS-2
- But AS-2 uses only AS-5’s route (i.e., it inserts AS-5’s route and the corresponding output link into its forwarding table)



BGP: Border Gateway Protocol

1. Advertize whole path



- Loop detection→ an AS checks for its own AS number in advertisement and rejects route if it has its own AS number

2. Incremental updates

- AS sends routing updates only when its preferred route changes (Messages are reliably delivered using TCP)
- Two types of update messages: advertisements, e.g., "P:{AS-20, AS-6}" and withdrawals "withdraw P"

Enforcing Policies (i.e., making money) Using BGP

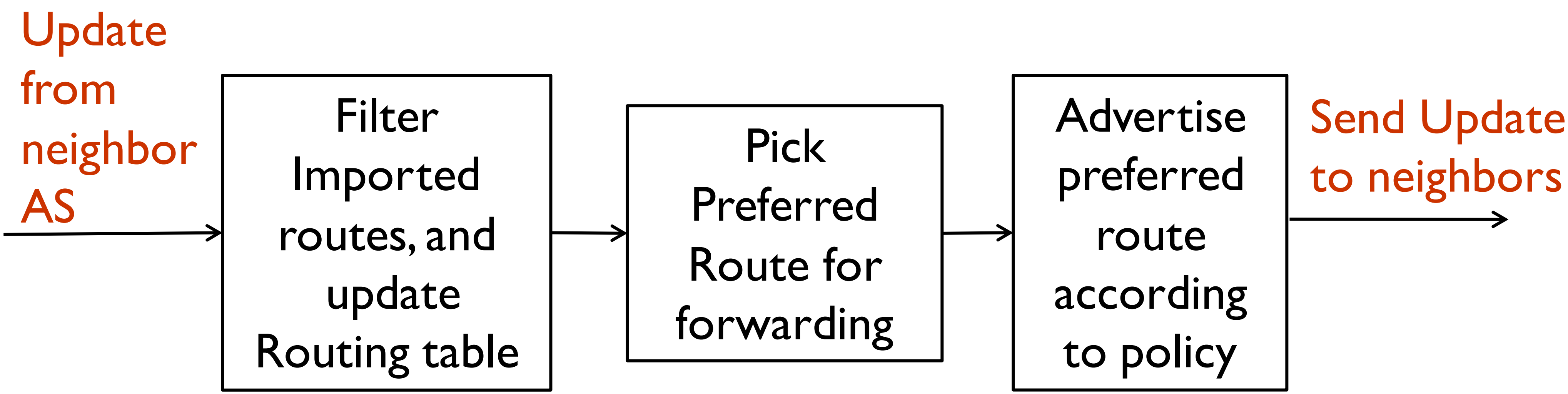
Route Export: controls incoming traffic

- AS advertises its customers (and internal prefixes) to all neighbors
- AS advertises all routes it uses to its customers (and internally)

Route Import: controls outgoing traffic

- For each dest. prefix, AS picks its preferred route from those in its routing table as follows:
  - Prefer route from **Customer > Peer > Provider**
  - Then, prefer route with shorter AS-Path

BGP



Update from neighbor AS
Filter Imported routes, and update Routing table
Pick Preferred Route for forwarding
Advertise preferred route according to policy
Send Update to neighbors

BGP Update Message Processing

When AS receives an advertisement,

For each destination prefix,

- Learn paths from neighbors
- Ignore loopy paths and keep the rest in your routing table
- Order paths according to AS preferences
  - Customers > peers > providers
  - Path with shorter AS hops are preferred to longer paths
- Insert the most preferred path into your forwarding table
- Advertise the most preferred path to a neighbor according to policies

When AS receives a withdrawal

- If withdrawn path not used/preferred, remove from routing table
- If withdrawn path is used –i.e., preferred
  - Remove the path from forwarding table and routing table
  - insert the next preferred path from the routing table into forwarding table
  - For each neighbor decide whether to tell him about the new path based on policies
    - If yes, advertise the new path which implicitly withdraws the old path for the corresponding prefix
    - If no, withdraw old path

Summary

- Hierarchical addressing and hierarchical routing improve scalability
- Inter-domain routing is policy-based not shortest path
  - An AS forwards transit traffic only if it makes money from it
- BGP is a path vector routing algorithm that implements policy-based routing