DATA COMMUNICATOIN NETWORKING

Instructor: Ouldooz Baghban Karimi

Course Book & Slides:
Computer Networking, A Top-Down Approach
By: Kurose, Ross

Course Overview

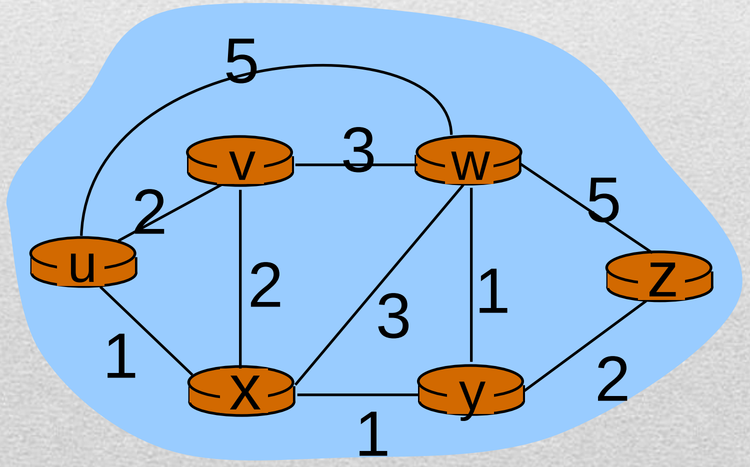
- **Basics of Computer Networks**
  - Internet & Protocol Stack
  - Application Layer
  - Transport Layer
  - Network Layer
  - Data Link Layer

- **Advanced Topics**
  - Case Studies of Computer Networks
  - Internet Applications
  - Network Management
  - Network Security

Graph Abstraction

- **Graph Abstraction of Networks**
  - Graph: G = (N,E)
  - N = set of routers = { u, v, w, x, y, z }
  - E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

- **Other Context**
  - P2P
    - N: Set of peers
    - E: Set of TCP Connections

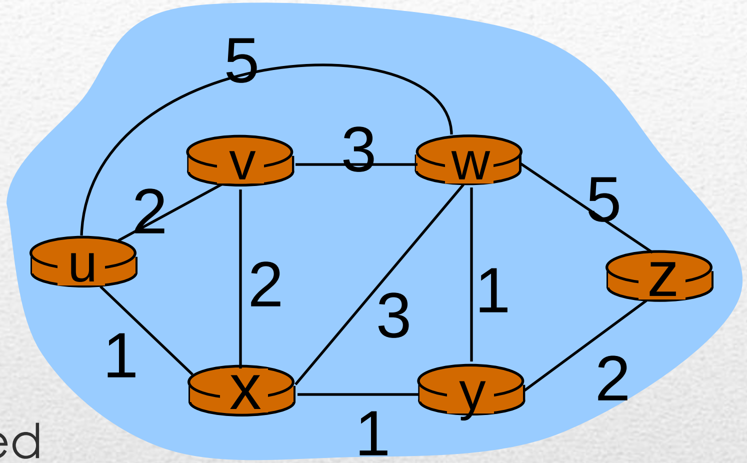


Graph Abstraction

- c(w,z) = cost of link (w,z)

  e.g., c(w,z) = 5

- Cost could always be 1, or inversely related to bandwidth, or inversely related to congestion



cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

What is the least-cost path between u and z ?

Routing algorithm: Algorithm that finds that least cost path

Routing Algorithm Classification

- **Centralized vs. Decentralized**
  - Centralized
    - All routers have complete topology, link cost info
    - **Link State** algorithms
  - Decentralized
    - Router knows physically-connected neighbors, link costs to neighbors
    - Iterative process of computation, exchange of info with neighbors
    - **Distance Vector** algorithms
- **Static vs. Dynamic**
  - Static
    - Routes change slowly over time
  - Dynamic
    - Routes change more quickly
    - Periodic update
    - In response to link cost changes

A Link State Algorithm

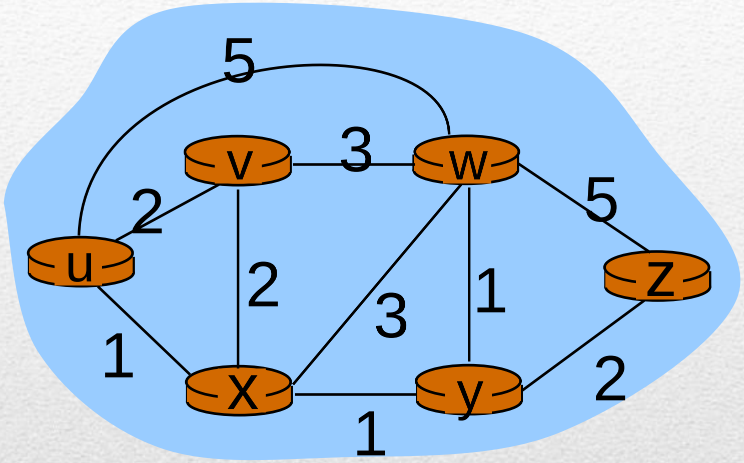
- **Dijkstra's algorithm**
  - Net topology and link costs known to all nodes
    - Accomplished via Link State Broadcast
    - All nodes have the same information
  - Computes least cost paths from one node to all other nodes
    - Gives forwarding table for that node
  - Iterative: after k iterations, know least cost path to k destinations

- **Notation**
  - c(x,y): link cost from node x to y; $= \infty$ if not direct neighbors
  - D(v): current value of cost of path from source to dest. v
  - p(v): predecessor node along path from source to v
  - N': set of nodes whose least cost path definitively known

Dijkstra's Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
until all nodes in N'
```



Distance Vector Algorithms

Bellman-Ford equation (dynamic programming)

Let $d_x(y)$:= cost of least-cost path from x to y

Then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

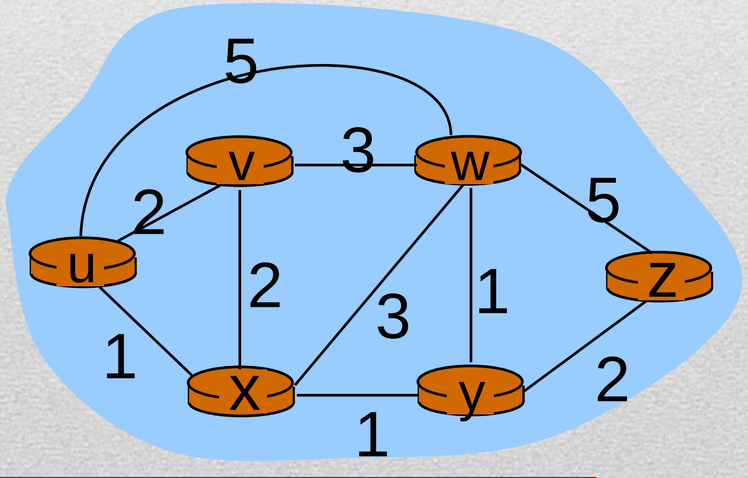
min taken over all neighbors v of x

Bellman Ford Example

$d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$
\begin{aligned}
d_u(z) = \min \{ & c(u,v) + d_v(z), \\
& c(u,x) + d_x(z), \\
& c(u,w) + d_w(z) \} \\
= \min \{ & 2 + 5, \\
& 1 + 3, \\
& 5 + 3\} = 4
\end{aligned}
$$



Node achieving minimum is next
hop in shortest path, used in forwarding table

Distance Vector Algorithms

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$

- Node x:
  - Knows cost to each neighbor v: c(x,v)
  - Maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance Vector Algorithms

Key Idea

- From time-to-time, each node sends its own distance vector estimate to neighbors
- When x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

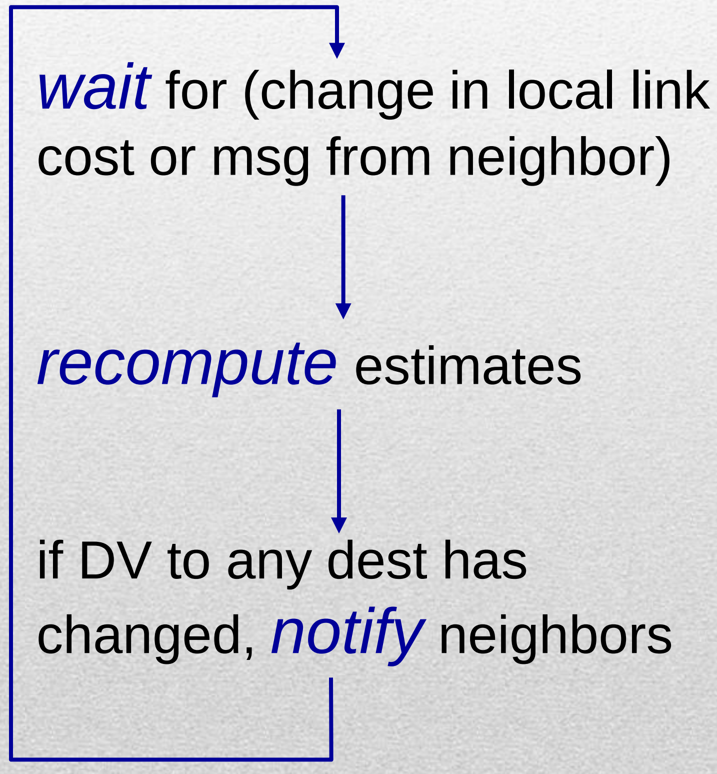
$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithms

- **Iterative, asynchronous**
  - Each local iteration caused by
    - Local link cost change
    - DV update message from neighbor
- **Distributed**
  - Each node notifies neighbors only when its DV changes
  - Neighbors then notify their neighbors if necessary

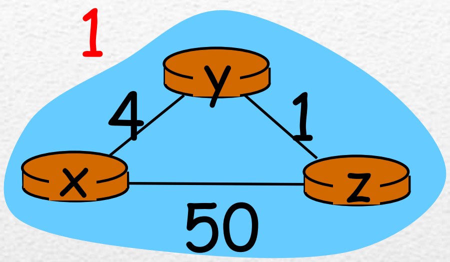
Each node:



Distance Vector

Link Cost Changes

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors



t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*' s update, updates its distance table. *y*' s least costs do *not* change, so *y* does *not* send a message to *z*.

Distance Vector & Link State

- **Message Complexity**
  - ***LS:*** With n nodes, E links, O(nE) msgs sent
  - ***DV:*** Exchange between neighbors only
    - Convergence time varies

- **Speed of Convergence**
  - ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
    - May have oscillations
  - ***DV:*** convergence time varies
    - May be routing loops
    - Count-to-infinity problem

- **Robustness:** hat happens if router malfunctions?
  - ***LS:*** Node can advertise incorrect *link* cost
    - Each node computes only its *own* table
  - ***DV:*** DV node can advertise incorrect *path* cost
    - Each node's table used by others
    - Error propagate thru network

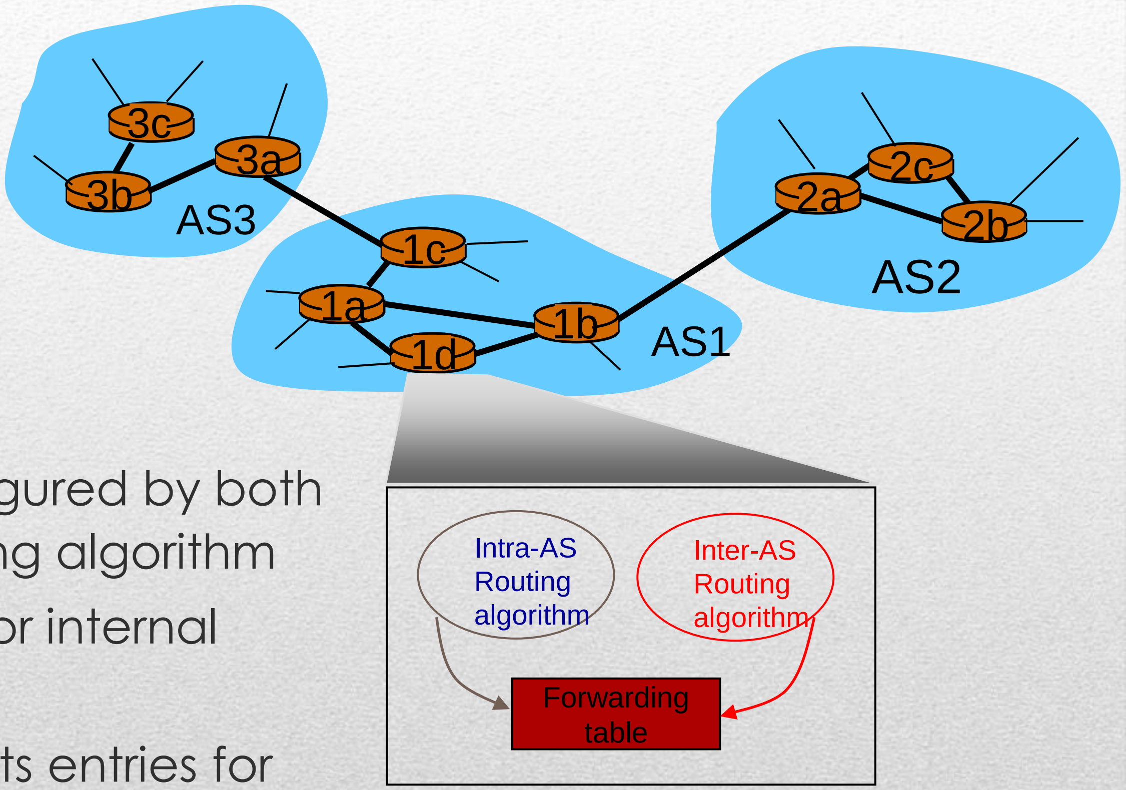
Hierarchical Routing

- **Ideal Model**
  - All routers identical
  - Flat Network
- **Practice**
  - 600 million destinations
  - Cannot store all destinations in routing tables
  - Administrative Autonomy
    - Internet=network of networks
    - Each network admin may want to control routing in its own network

Hierarchical Routing

- Aggregate routers into regions, Autonomous Systems(AS)
- Routers in same AS run same routing protocol
  - Intra-AS routing protocol
  - Routers in different AS can run different intra-AS routing protocol
- **Gateway Router**
  - Edge of its own AS
  - Has link to router in another AS

Interconnected ASs



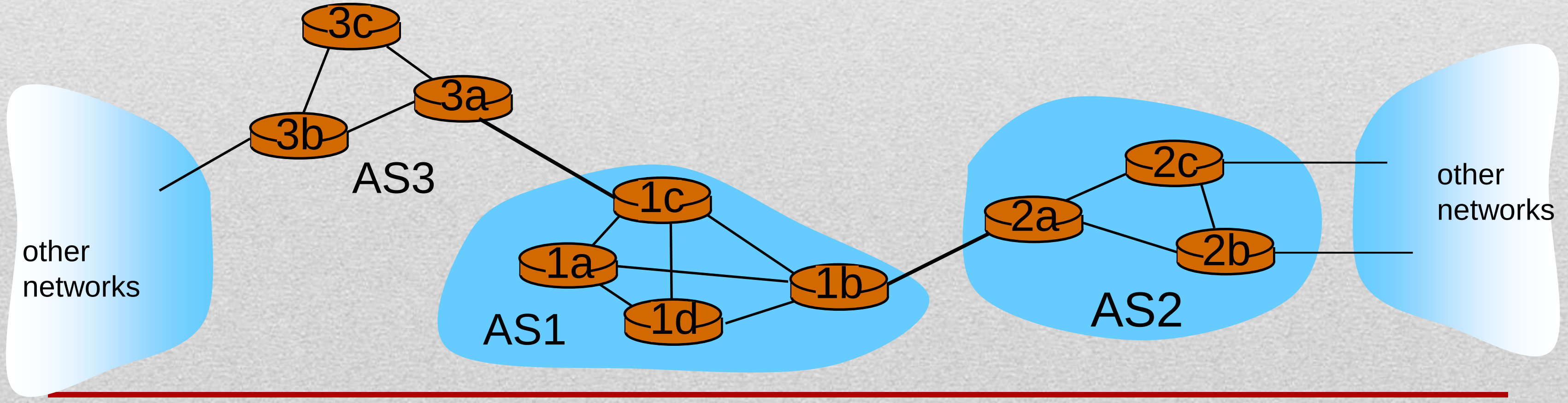
- Forwarding table configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal destinations
  - Inter-AS & intra-AS sets entries for external destinations

Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router, but which one?

- AS1 must
  - Learn which dests are reachable through AS2, which through AS3
  - Propagate this reachability info to all routers in AS1

Job of inter-AS routing!

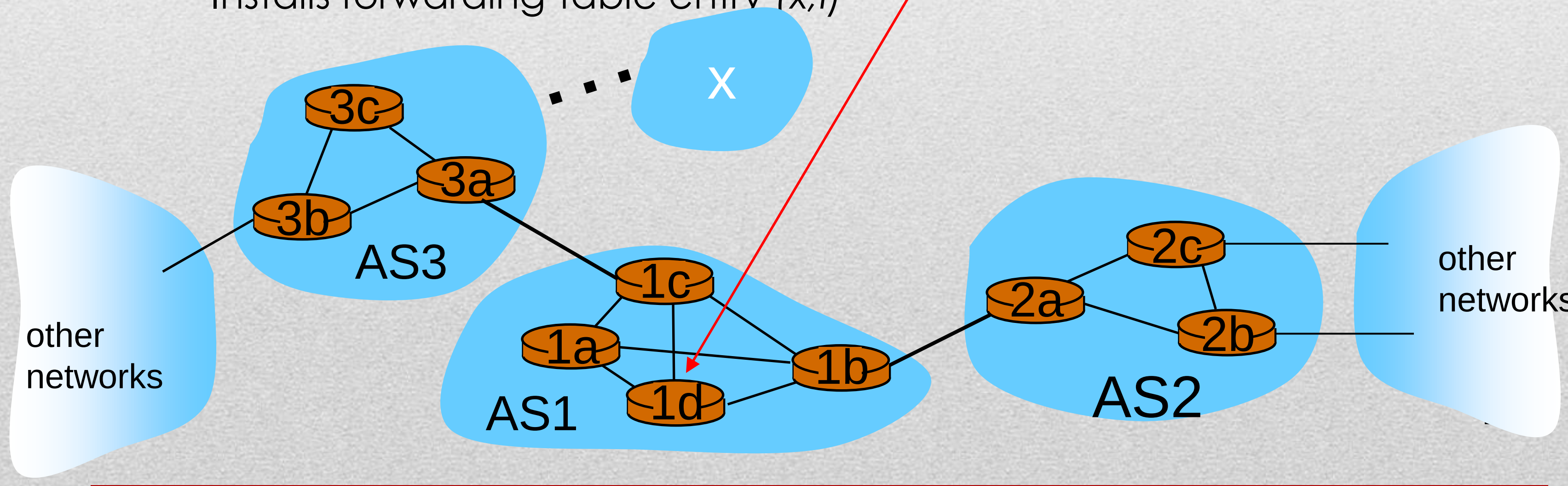


Example

Setting Forwarding Table In Router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet x reachable via AS3 (gateway 1c), but not via AS2
  - Inter-AS protocol propagates reachability info to all internal routers

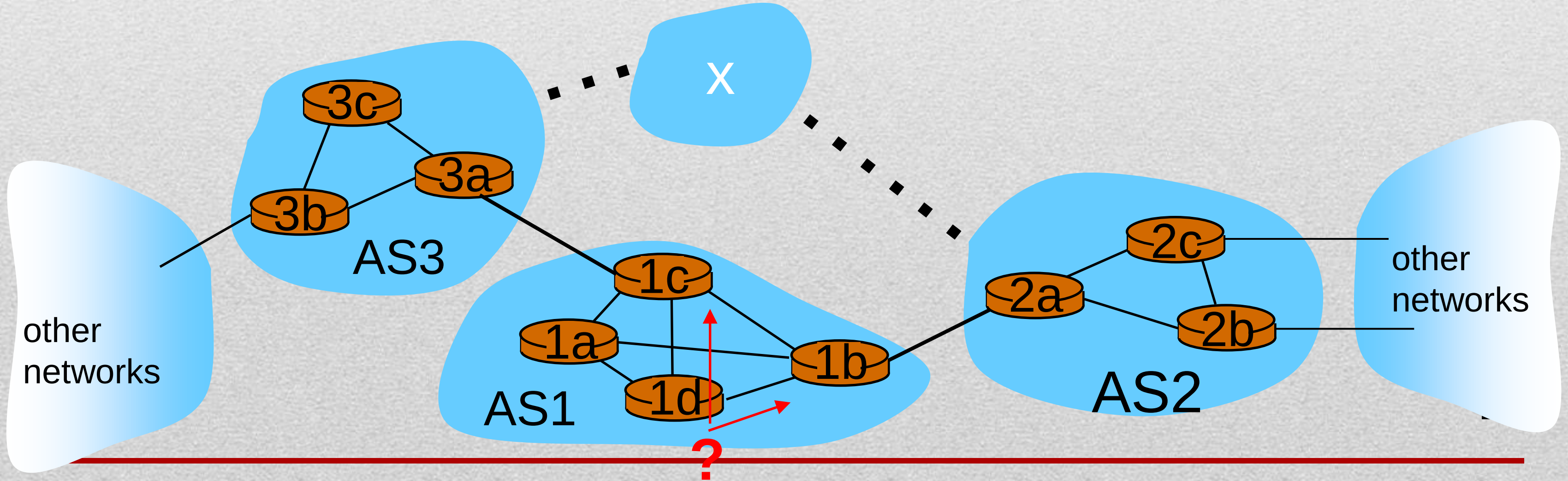
- Router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - Installs forwarding table entry *(x,I)*



Example

Choosing Among Multiple Ass

- Now suppose AS1 learns from inter-AS protocol that subnet X is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine which gateway it should forward packets towards for destination X
  - This is also job of inter-AS routing protocol!



Example

Choosing Among Multiple Ass

- Now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 and from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x
  - This is also job of inter-AS routing protocol!
- Hot potato routing: send packet towards closest of two routers.