ELEC / COMP 177 – Fall 2012

Computer Networking
→ Routing Protocols (1)

Some slides from Kurose and Ross, *Computer Networking*, 5th Edition

Schedule

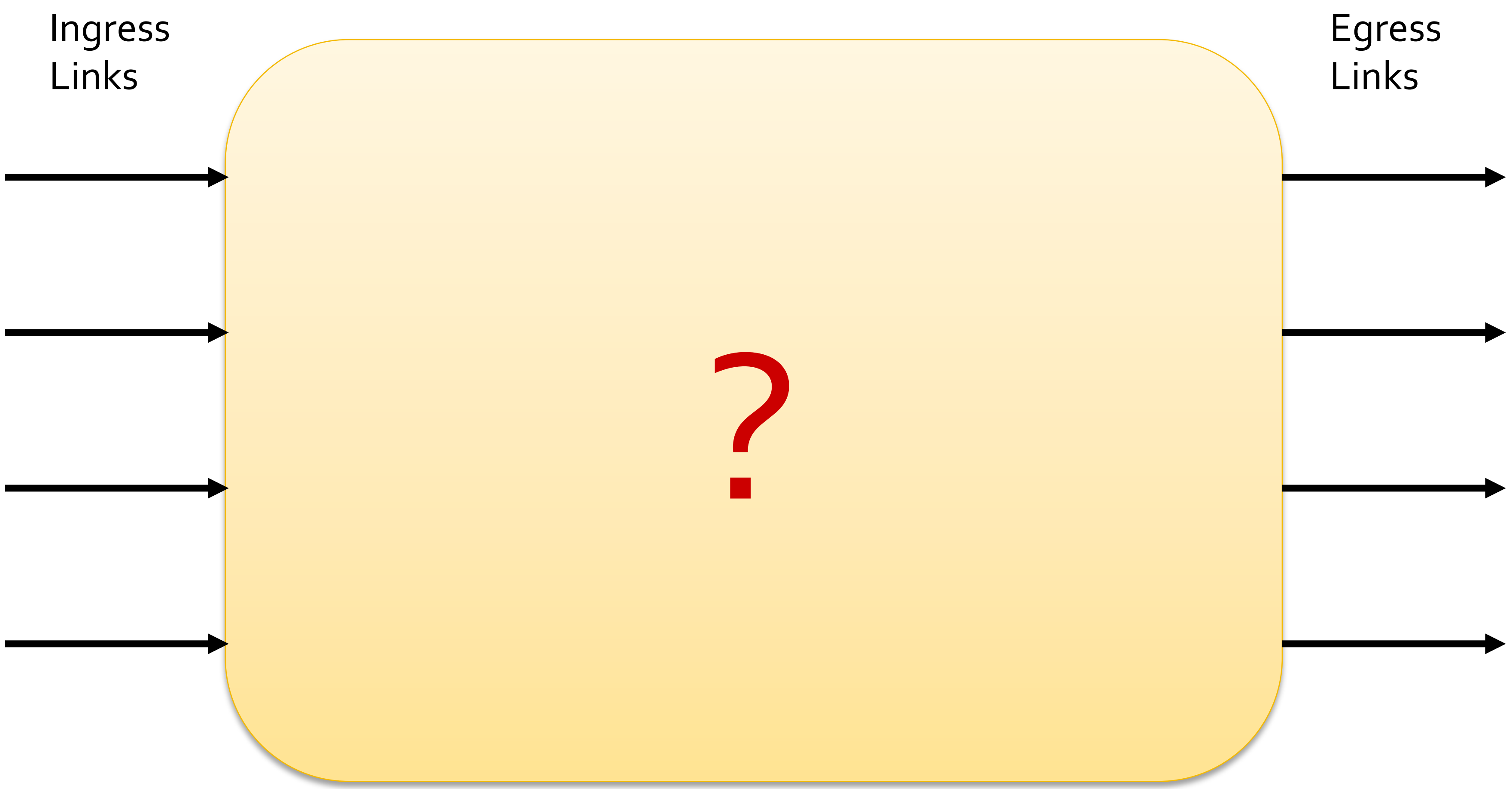
- **Homework #4** – Due Thursday, Nov 1st
- **Project #2** – Due Tuesday, Nov 6th
- ***Later this semester:***
  - ***Homework #5*** – *Due Thursday, Nov 13th*
  - ***Homework #6 - Presentation on security/privacy***
    - ***Topic selection*** – *Due Tuesday, Nov 20th*
    - **Slides** – Due Monday, Nov 26th
    - **Present!** – Tuesday, Nov 27th (and Thursday)
  - ***Project #3*** – *Due Tue, Dec 4th*

Today

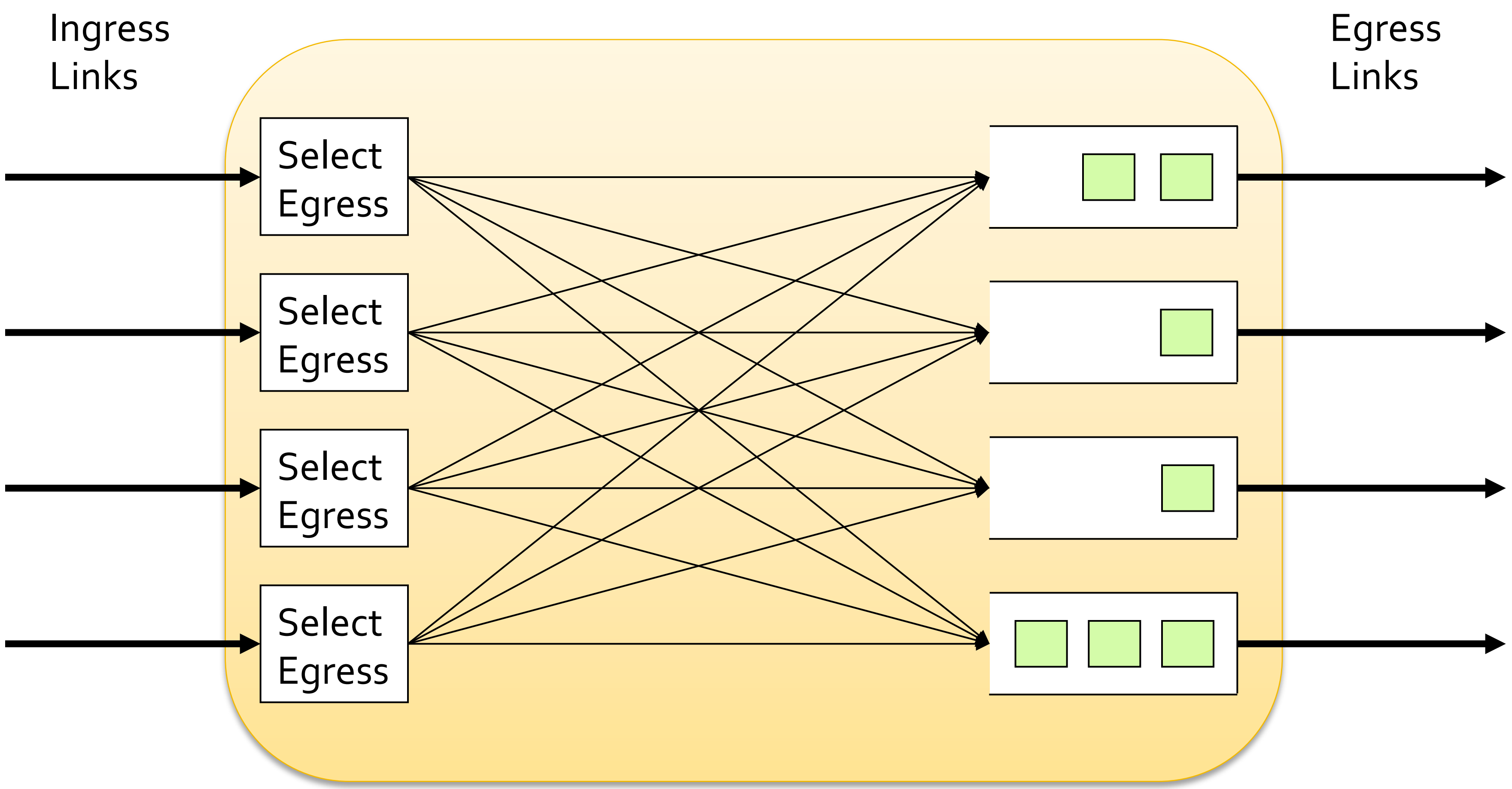
- Discuss what routers do internally
- Discuss the “other piece” of the network layer besides IP: **Routing algorithms**
  - How do routers decide what port to forward a packet out on?
    - *Beyond just having the administrator enter all routes manually like you did in the lab before RIP was enabled…*

Router Operation

What's inside a router?

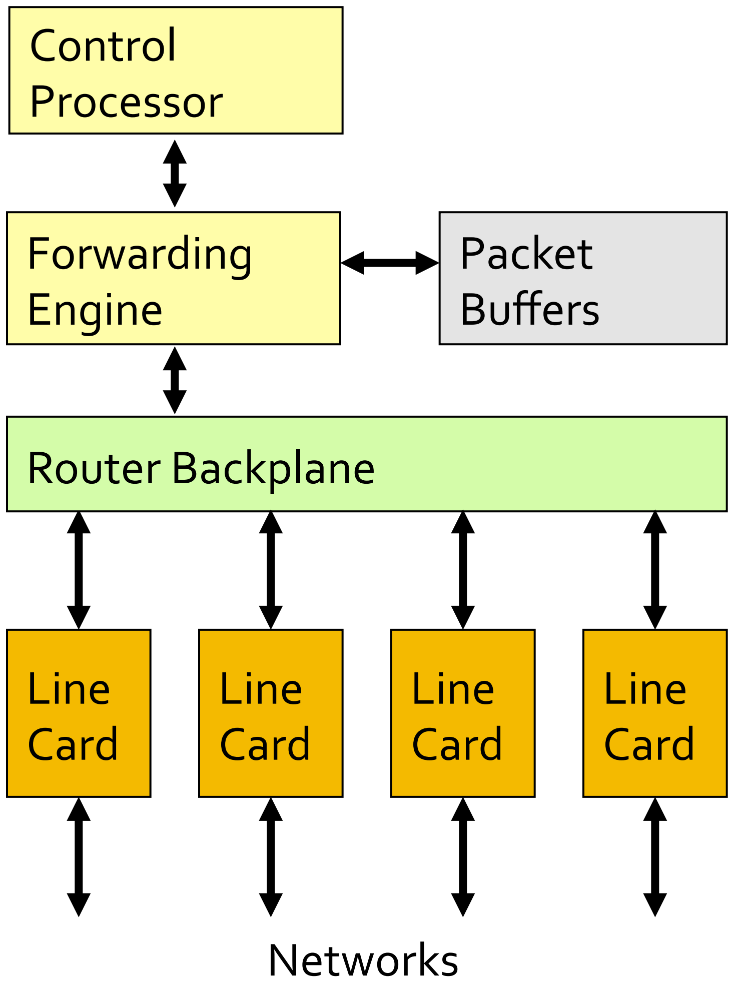


Simplified model of a router



Basic Router Components

- Key Modules
  - Network Interface
  - Packet processing
  - Packet buffering
  - Packet switching
- Processing and buffering can be centralized or decentralized



Packet Processing

- What does a router need to do?

- Driven by protocols
  - Ethernet
  - IP
  - ARP
  - ICMP
  - Transport: TCP, UDP, etc.

On packet arrival…

- Processing
  - Buffer packet?
  - Determine protocol (e.g., IP vs. ARP)
  - Verify checksum, validate the packet, etc.
  - Collect statistics?
- What's next in the "common" (valid IP packet) case?
  - Select egress link

Selecting an Egress Link

- Forwarding table lookup
  - Longest prefix match
  - Determine next hop IP address and egress link
- **What if no match?**
- **Is this sufficient to route the packet to an output queue?**

Prefix	Next Hop	Port
63/8	128.34.12.1	3
128.42/16	128.34.12.1	3
156.3/16	128.36.21.1	2
156.3.224/19	128.36.129.1	1
128.42.96/20	128.37.37.1	4
128.42.128/24	128.36.129.1	1
128.42.160/24	128.36.21.1	2

Updating the Destination Address

- ARP table lookup
  - Exact match on next hop IP address
  - Determine next hop MAC address
- **What if no match?**

IP	MAC
128.34.12.1	0C:FF:63:82:44:01
128.36.21.1	04:32:11:44:82:60
128.36.21.18	10:44:82:82:44:07
128.37.37.37	08:82:82:44:16:32
128.34.12.14	20:33:71:28:15:70
128.36.21.42	14:93:29:22:15:28

Generating ARP Requests

- Broadcast on output port
  - Ask for MAC address of next hop IP address
- Wait for reply
  - **What do you do with the packet?**
  - **How long should you wait? (tradeoffs?)**
- Receive reply
  - Update ARP table
  - Packet continues along forwarding path

Receiving ARP Requests

- Does the IP address match the IP address of the interface that received the ARP request?
  - Another system is trying to determine your MAC address
  - Respond with the appropriate ARP reply on the same interface
- **Should ARP requests be forwarded if they aren't for the router?**

Updating Packets

- Select egress link
- Update MAC address
- **Is it now OK to forward packet to output queue?**

- IP packet header must be modified
  - **What needs to be modified?**
  - **When should it be modified?**

Buffering

- Why do packets need to be buffered?
  - Waiting for access to a resource (lookup table, switch, etc.)
  - Waiting for an ARP reply
  - ...
- What happens when buffers get full?
  - Packets have to be dropped
- How large do buffers need to be?
  - Statistical multiplexing

Error Handling

- ICMP Messages
  - Notify sender of errors
- Common error types
  - Host/network unreachable
    - No ARP response
  - Time exceeded
    - TTL decremented to zero
  - No route to host
    - No entry in routing table

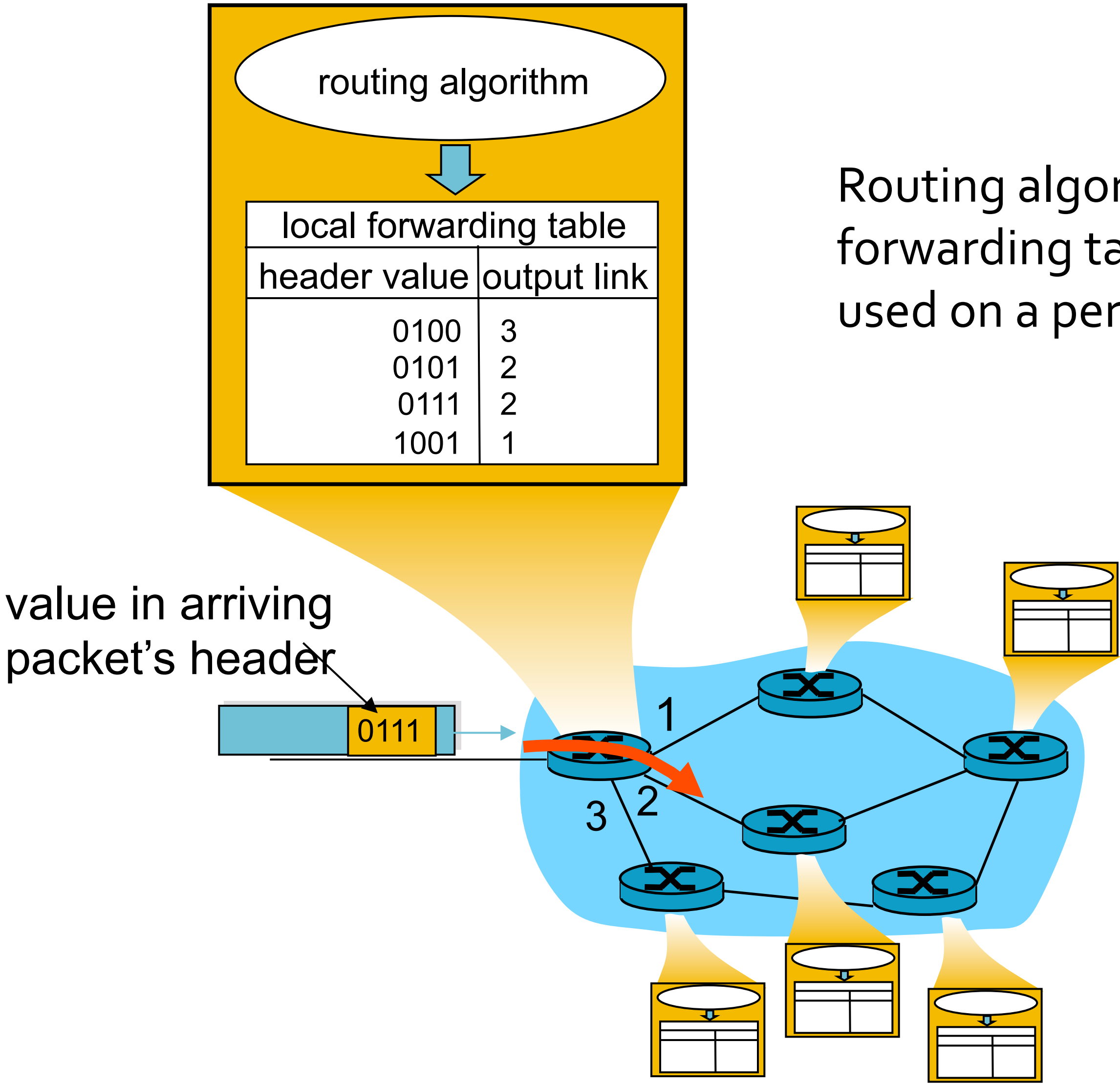
Routing Algorithms

Two Key Network-Layer Functions

- **Forwarding**
  - Move packets from router's input to appropriate router output
  - *Forwarding table*
- **Routing**
  - Determine path (route) taken by packets from source to destination
  - *Routing algorithms*

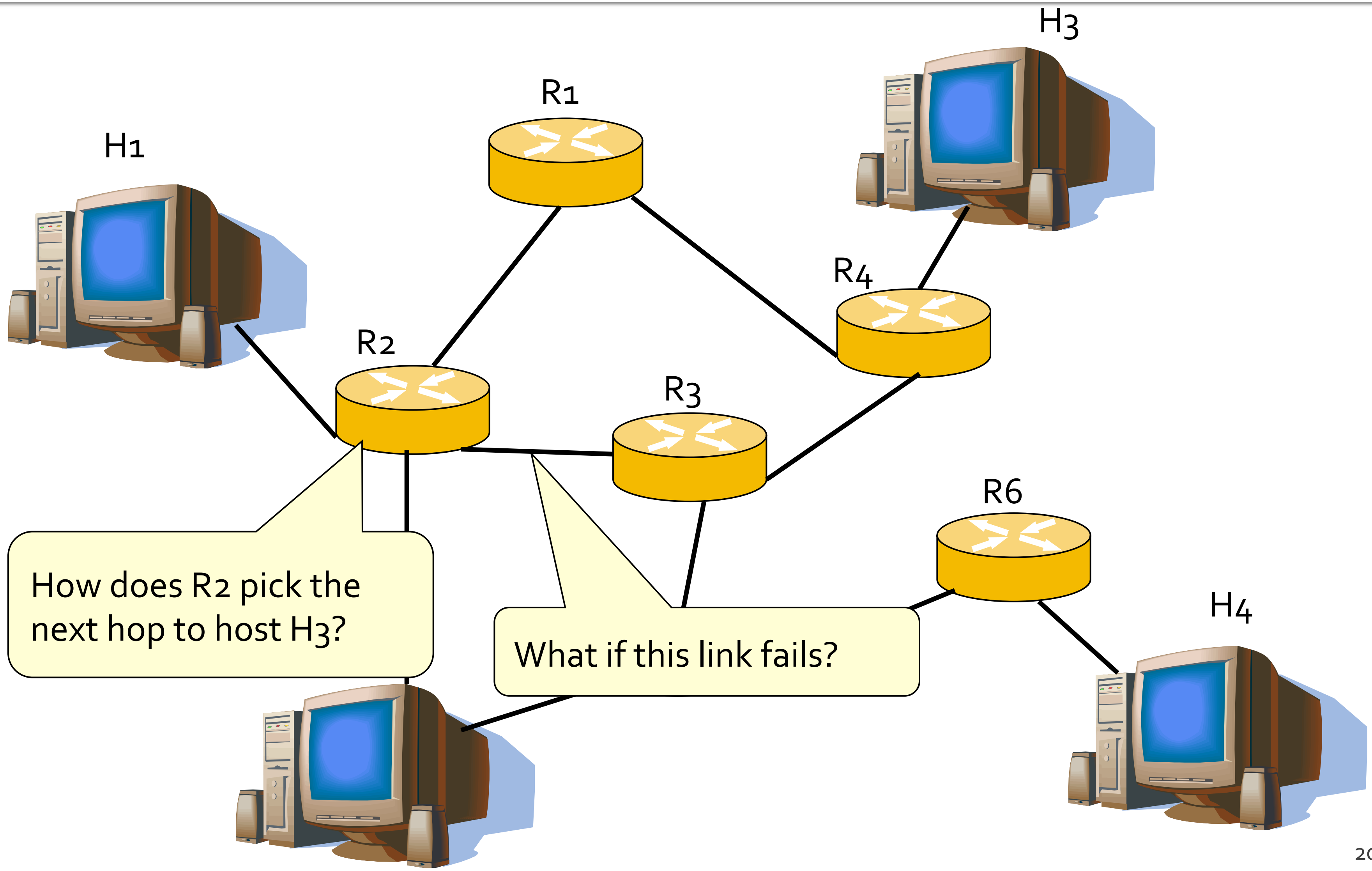
- Road trip analogy:
  - **Forwarding**: process of getting through single interchange
  - **Routing**: process of planning trip from source to destination

Routing versus Forwarding



Routing algorithm *creates* the forwarding table, which is used on a per-packet basis

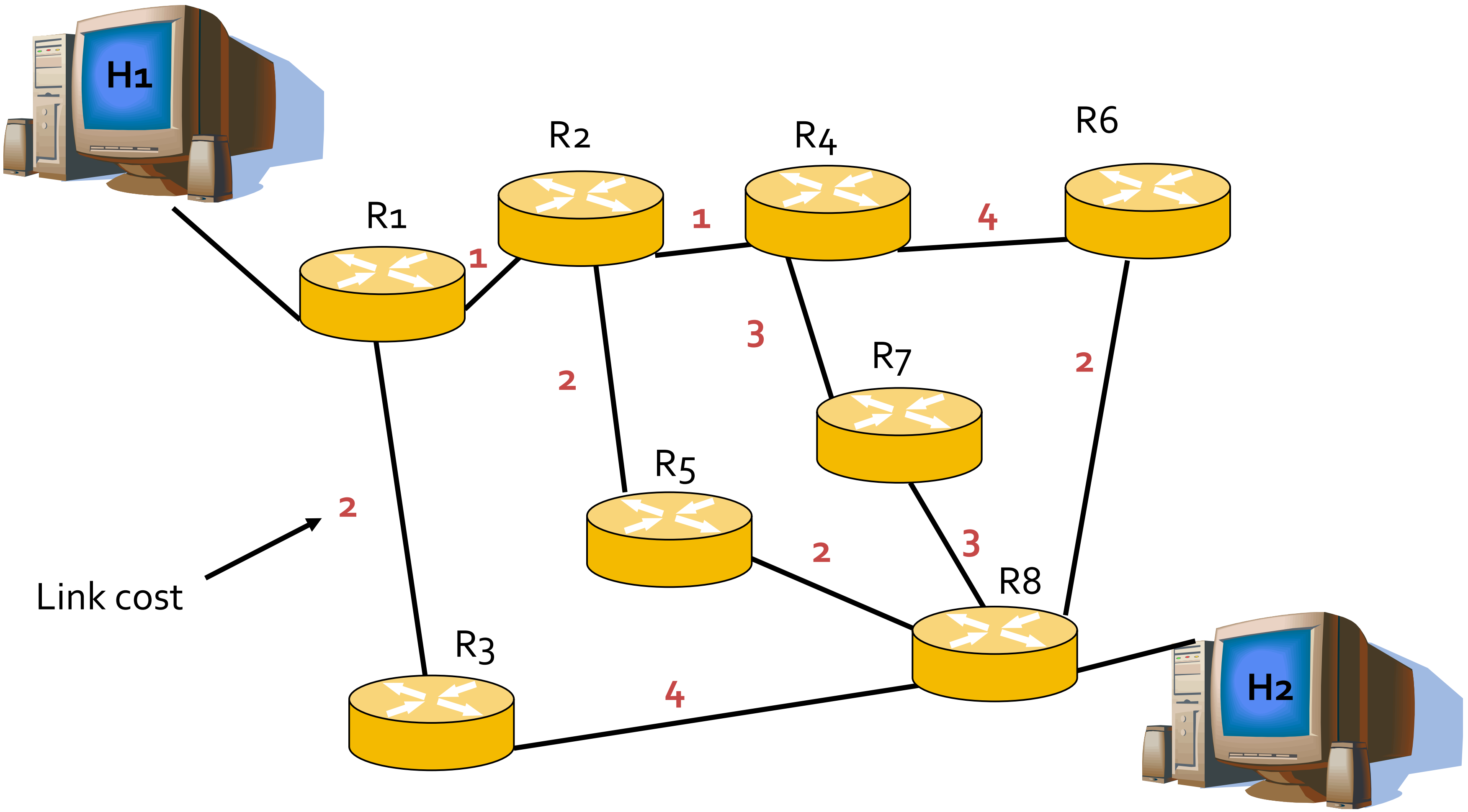
Forwarding Table Entries



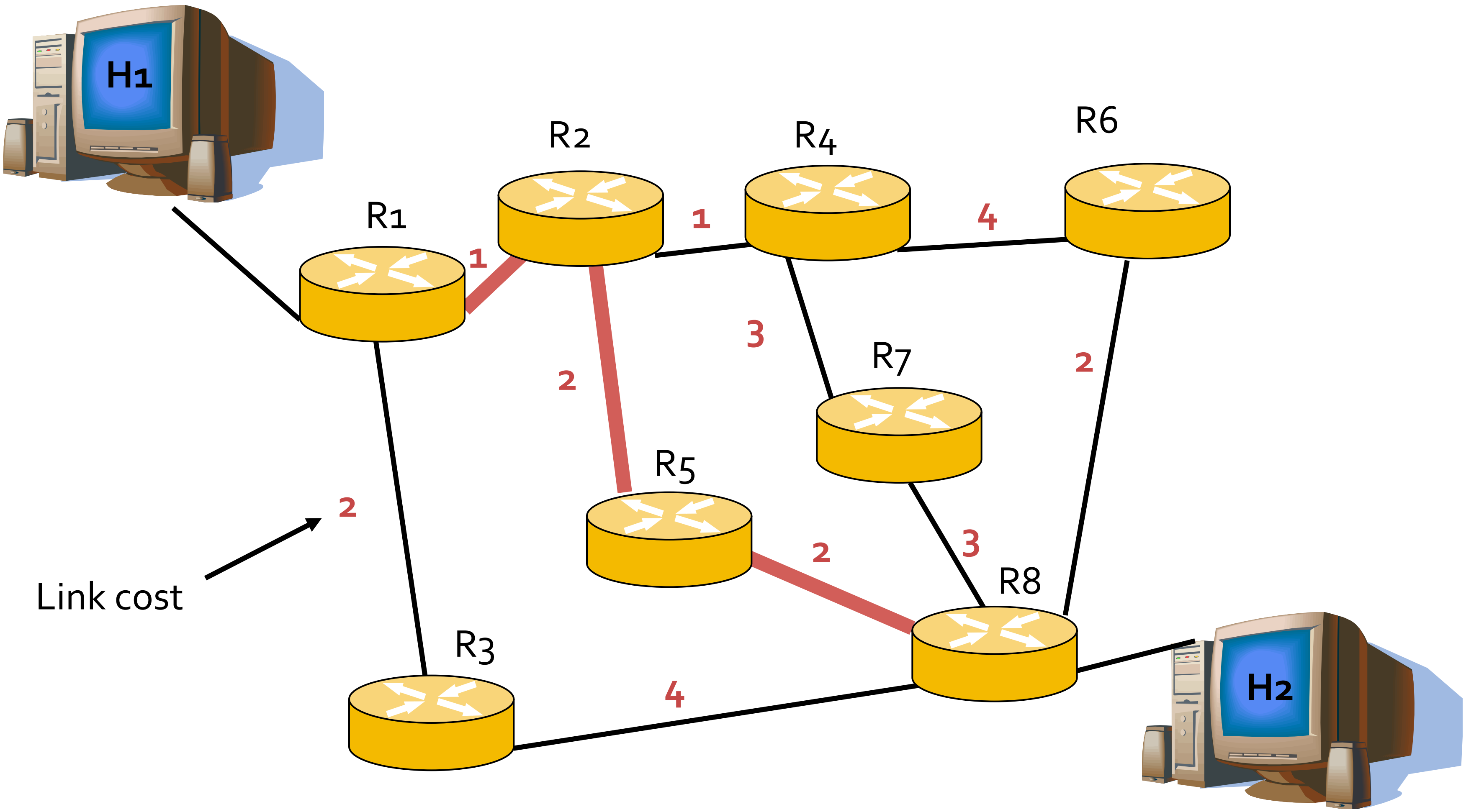
Generating/Updating Routes

- So far, we have assumed forwarding tables are populated statically by an administrator
- In reality, they are dynamically updated
  - Faster reaction to changing network conditions
- **What makes a good route?**
  - Low delay
  - High bandwidth
  - Low link utilization
  - High link stability
  - Low cost
    - (cheaper to use ISP A than ISP B)

Example Network

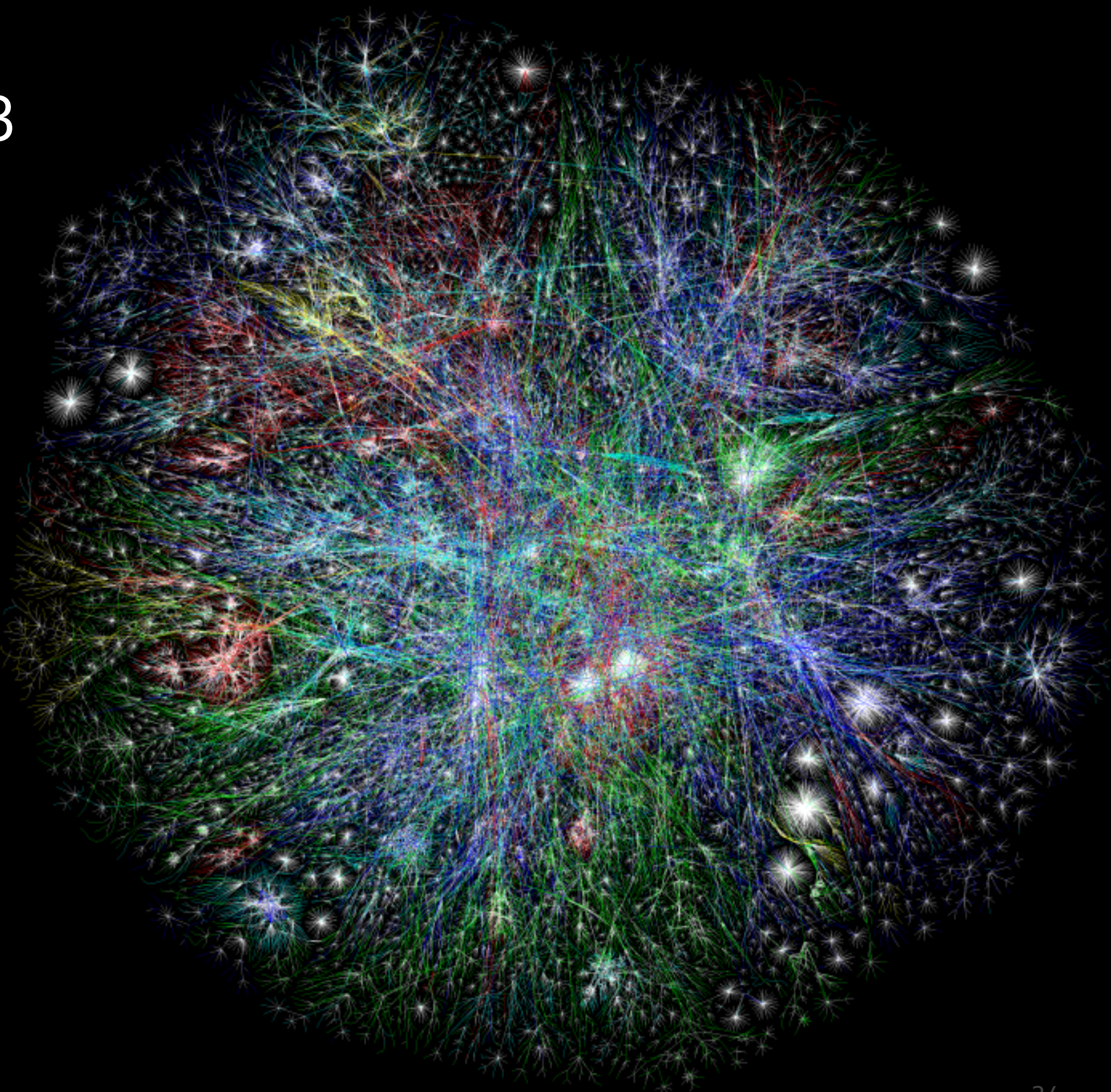


"Best" Path

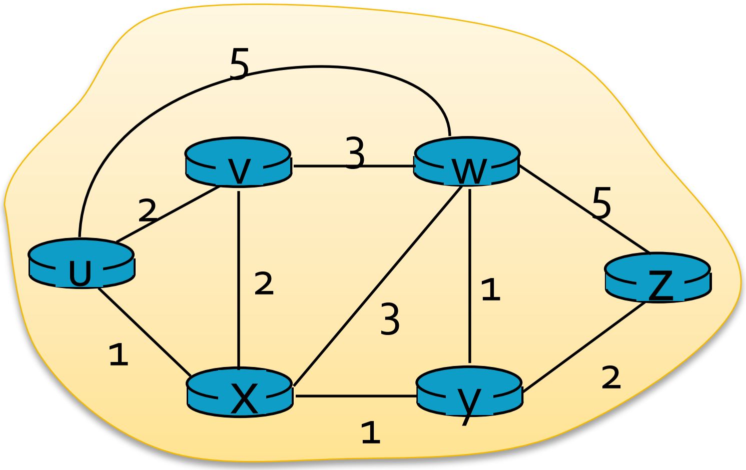


Real Networks Are Complicated

- The Internet in 2003
  - http://www.opte.org/maps/



Graph Abstraction



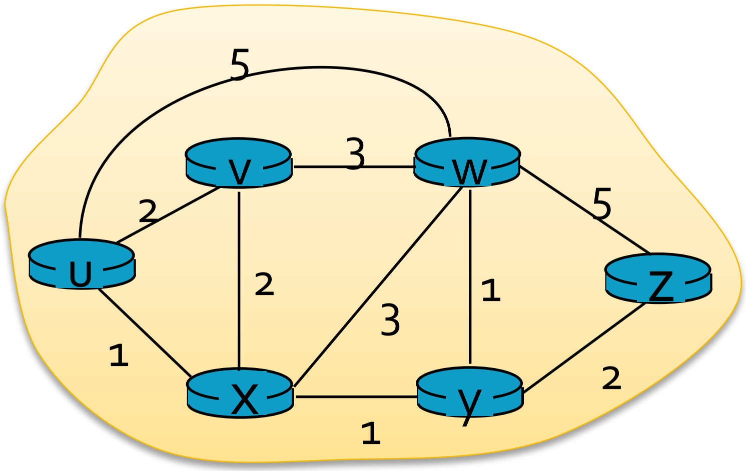
Graph: G = (N,E)

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) }

Graph abstraction is useful in other network contexts.
Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- $c(x,x')$ = cost of link (x,x')
  - e.g., $c(w,z) = 5$

- How do we set cost?
  - Fixed, i.e. always 1
  - Inversely related to bandwidth
  - Related to congestion

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

Question: What's the least-cost path between u and z ?

Routing algorithms find the least-cost path

Routing Algorithm Classification

GLOBAL OR DECENTRALIZED?

- **Global Information**
  - All routers have complete topology, link cost info
  - **"link state" algorithms**
- **Decentralized**:
  - Router knows physically-connected neighbors and link costs to neighbors
  - Iterative process of computation, exchange of info with neighbors
  - **"distance vector" algorithms**

STATIC OR DYNAMIC?

- **Static**
  - Routes change slowly over time
- **Dynamic**
  - Routes change more quickly
  - Periodic update
  - In response to link cost changes

Link-State Routing

Dijkstra's Algorithm

Dijkstra's Algorithm

- Network topology and link costs are **known to all nodes**
  - Accomplished via "link state broadcast"
  - **All nodes have same info**
- Computes least cost paths from one node (source) to all other nodes
  - Produces **forwarding table** for that node
- Iterative: after *k* iterations, know least cost path to *k* destinations

Notation:

- u: the source ("you")
- c(x,y): link cost from node x to y; = ∞ 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

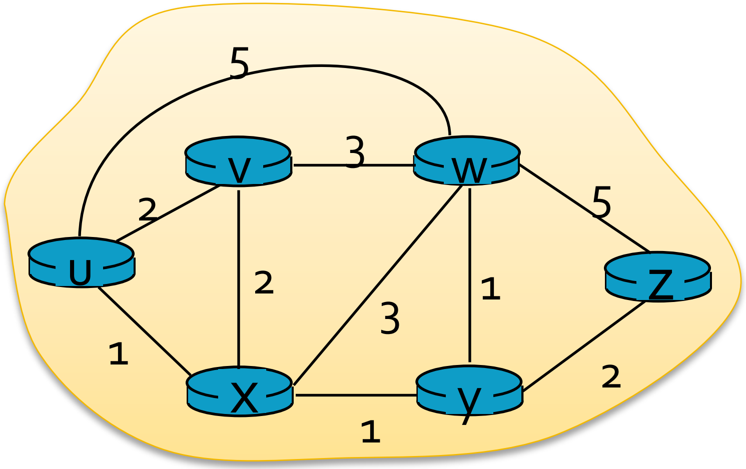
Dijkstra's Algorithm

```
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'
```

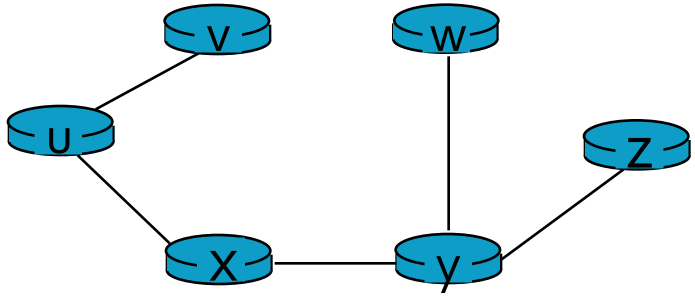
Dijkstra's Algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's Algorithm: example

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Distance-Vector Routing

Bellman – Ford Algorithm

Bellman-Ford Equation

Distributed! No global knowledge needed!

Define:

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

Then:

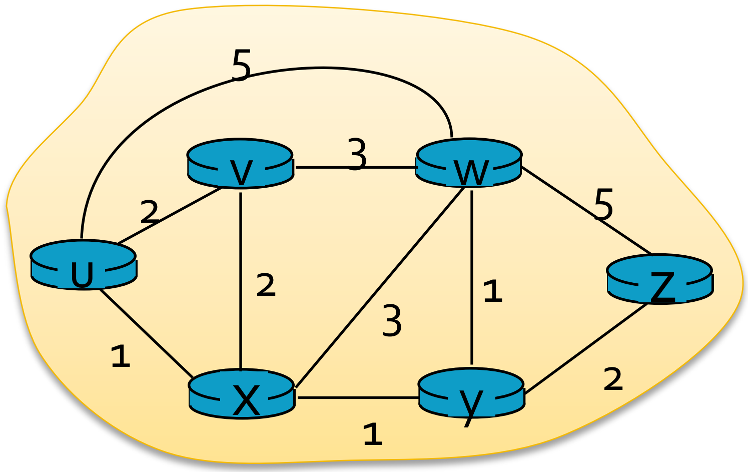
Something I know...

Something my neighbor told me...

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

where min is taken over all neighbors v of x

Bellman-Ford Example



From the figure: $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z) \}$$

$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = \mathbf{4}$$ *(by way of x!)*

The node that provides the minimum cost is entered in the router forwarding table as the next hop

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains distance vector

  $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains

    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance Vector Basics

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

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

Distance Vector Algorithm

Iterative, asynchronous:

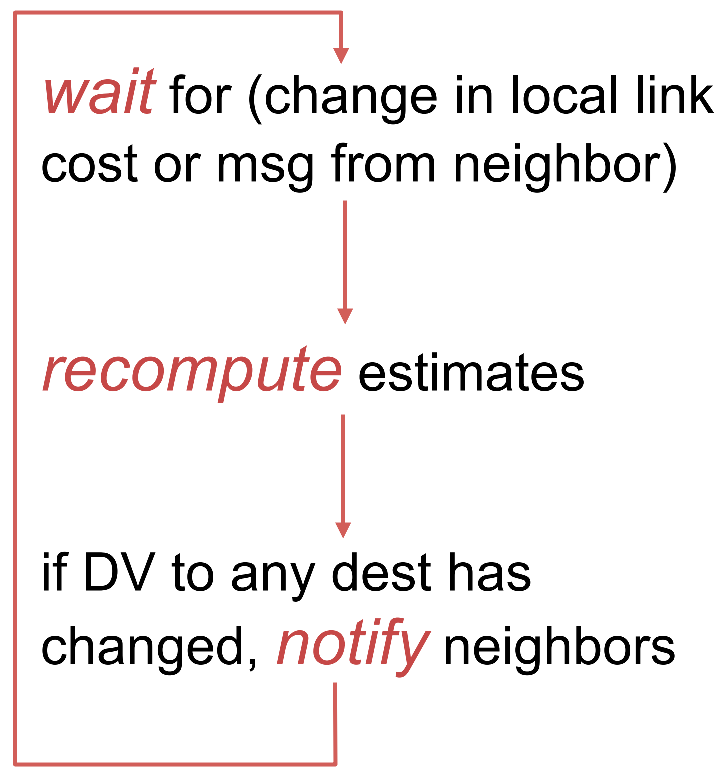
each local iteration caused by:

- 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:



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$
Found a shorter path!

node x table

cost to (from rows):

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to (from rows):

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

cost to (from rows):

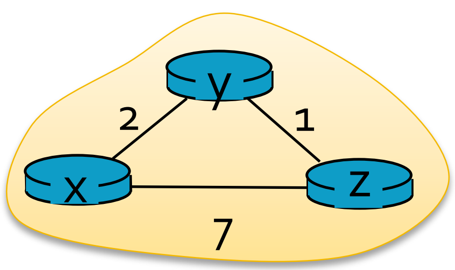
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to (from rows):

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

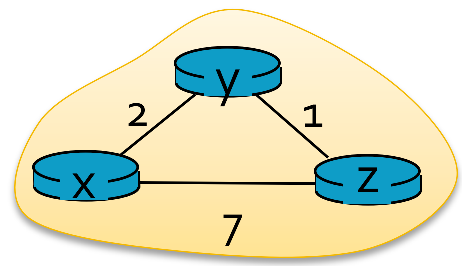
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time →



Hierarchical Routing

Hierarchical Routing

- Our routing discussion thus far has been idealized
  - All routers are identical
  - The network is “flat”
- This is not true in practice!

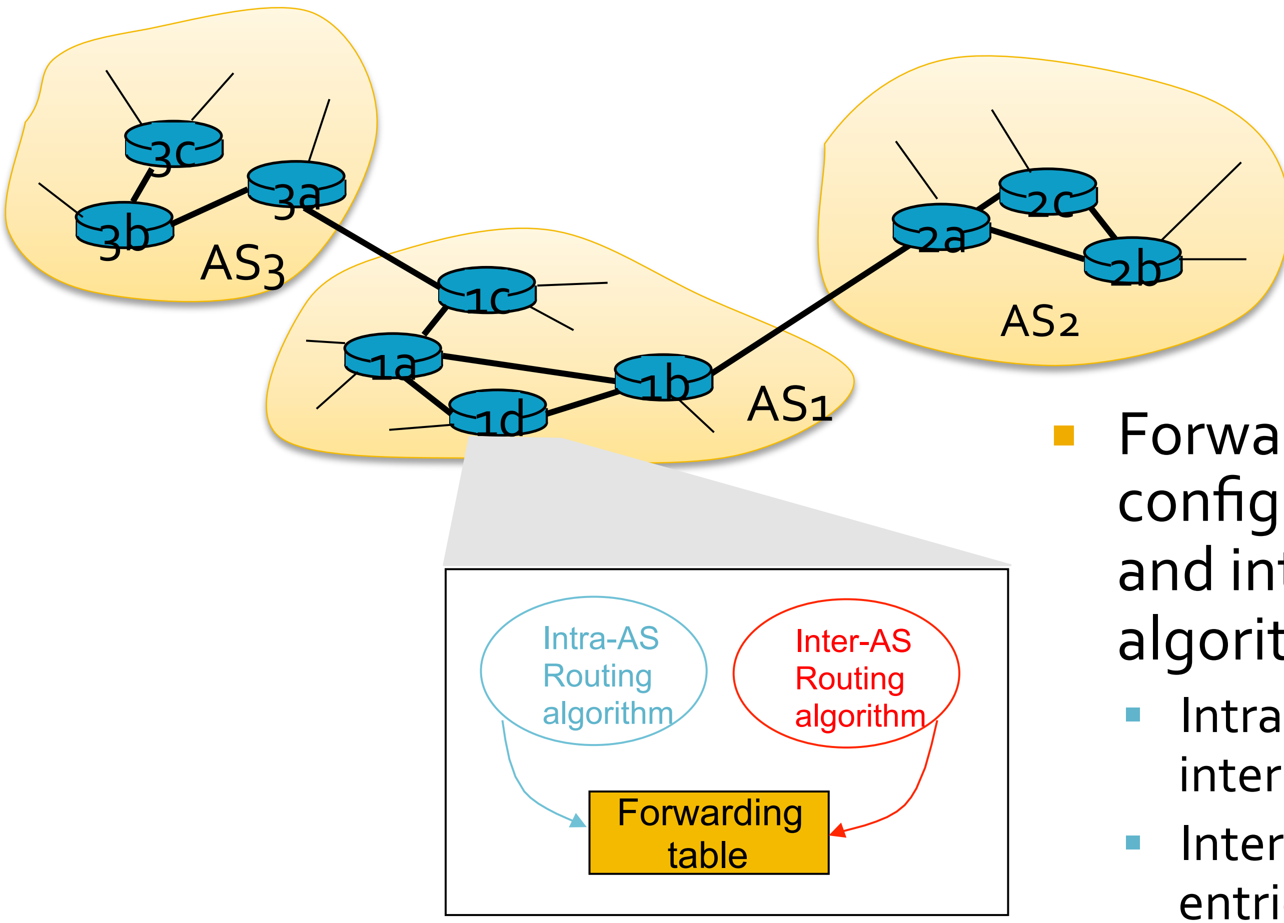
- Problem 1 – **Scale**
  - Hundreds of millions of destinations:
  - Can’t store all destinations in routing tables!
  - Routing table exchange would swamp links!
  - Distance-vector would never converge

- Problem 2 - **Administrative autonomy**
  - Internet = network of networks
  - Each network admin wants to control routing in his/her own network

Hierarchical Routing

- Aggregate routers into regions
  - aka "**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
- Border router
  - Direct link to router in another AS

Interconnected Autonomous Systems

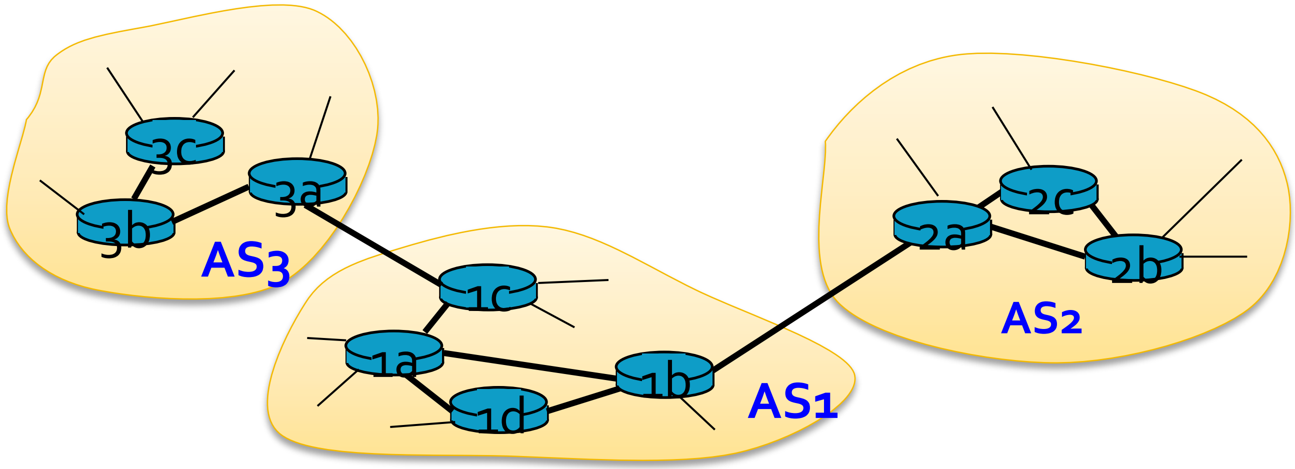


- Forwarding table configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal destinations
  - Inter-AS & intra-As sets entries for external destinations

Inter-AS tasks

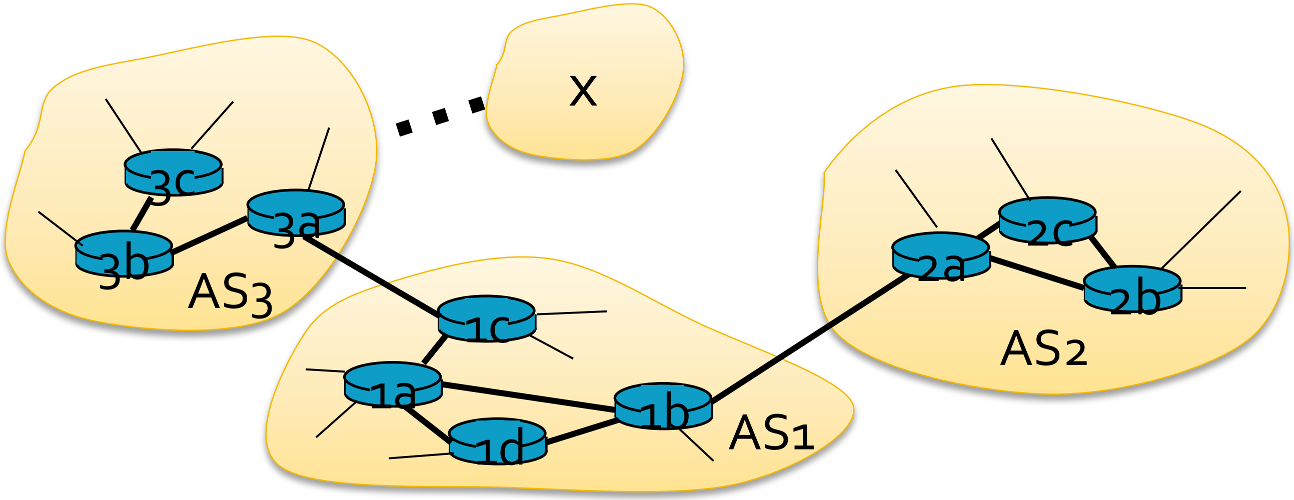
- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to border router, but which one?

- AS1 must:
  1. Learn which dests are reachable through AS2 versus AS3
  2. 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* is reachable only via AS3 (gateway 1c) and not via AS2.
  - Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *n* is on the least cost path to 1c.
  - Installs forwarding table entry *(x,n)*



Example: Choosing Among Multiple Autonomous Systems

- 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 dest *x*.
  - This is also job of inter-AS routing protocol!