Debugging; Routing

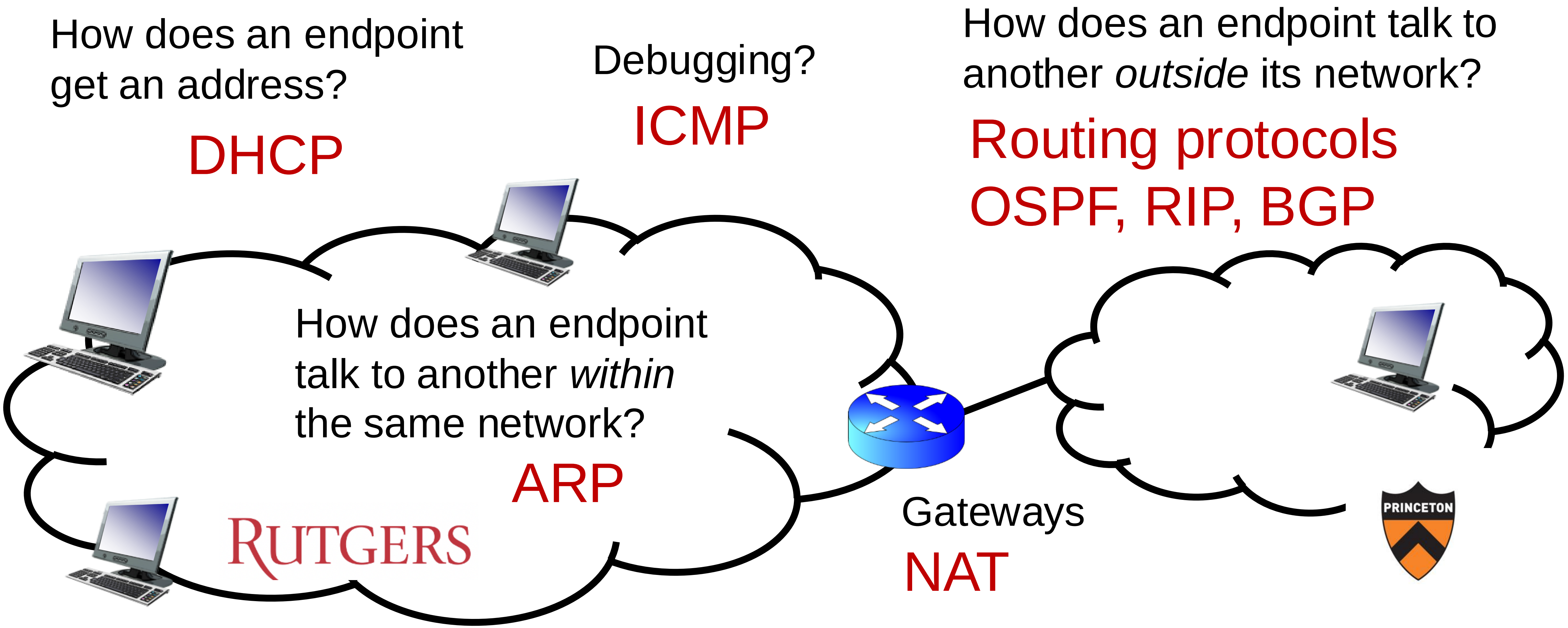
Lecture 23

http://www.cs.rutgers.edu/~sn624/352-F24

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

The network layer enables reachability. We'll see protocols that solve subproblems.



Review: Internet Control Message Protocol (ICMP)

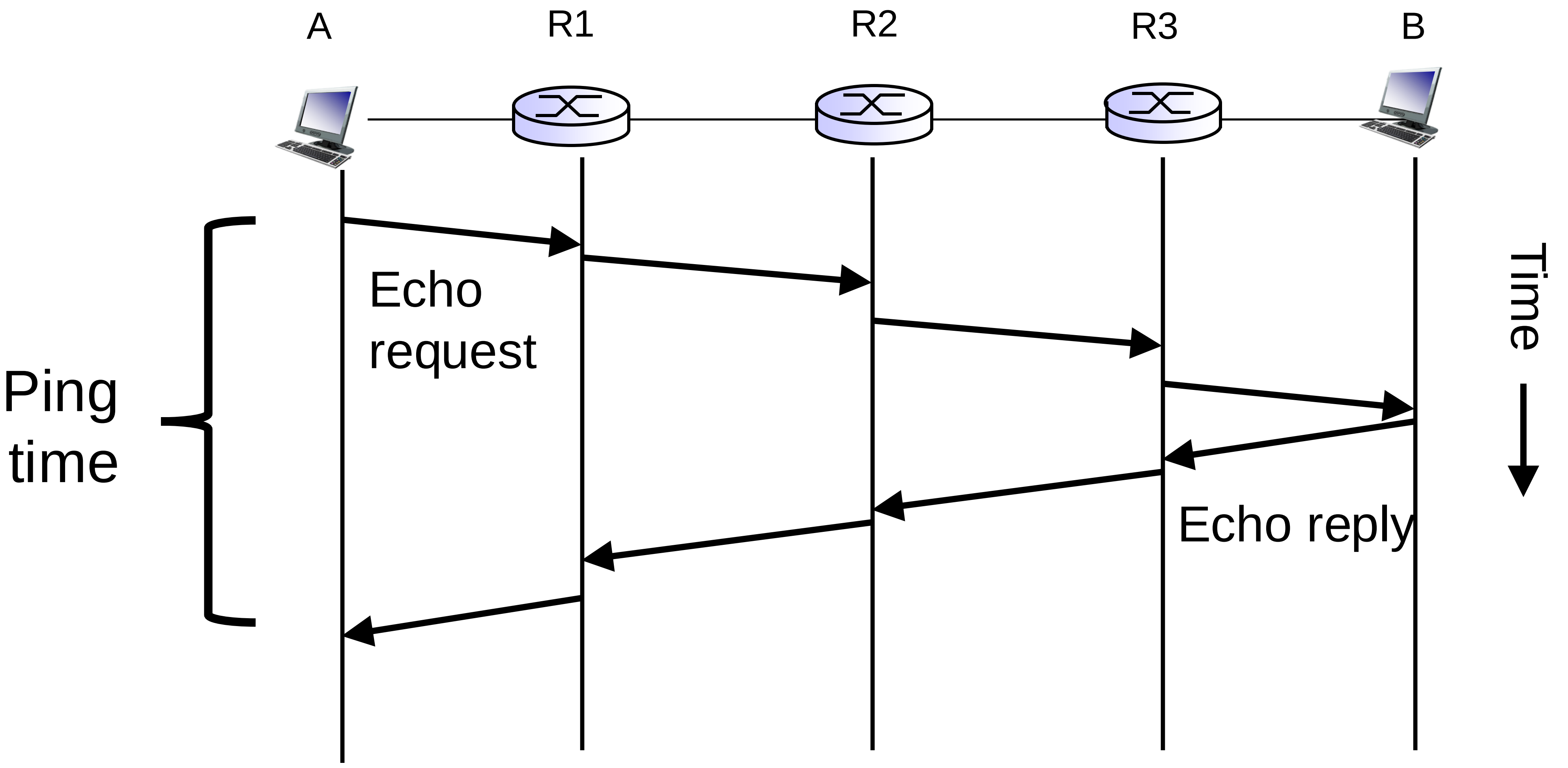
- A protocol for troubleshooting and diagnostics
- Works over IP: unreliable delivery of packets
- Some functions of ICMP:
  - Determine reachability (ping) and provide unreachability errors
  - Specify that packets have been in the network for too long (traceroute)

ICMP header Message type, Code, Checksum, ICMP data
IP header

Ping

- Uses ICMP echo request (type=8, code=0) and reply (type=0, code=0)
- Source sends ICMP echo request message to dst address
- Destination network stack replies with an ICMP echo reply message
- Source can calculate round trip time (RTT) of packets
- If no echo reply comes back, then the destination is unreachable
- Don’t need to have a server program running on the other side
  - In general, the remote endpoint can be completely outside your control

Ping



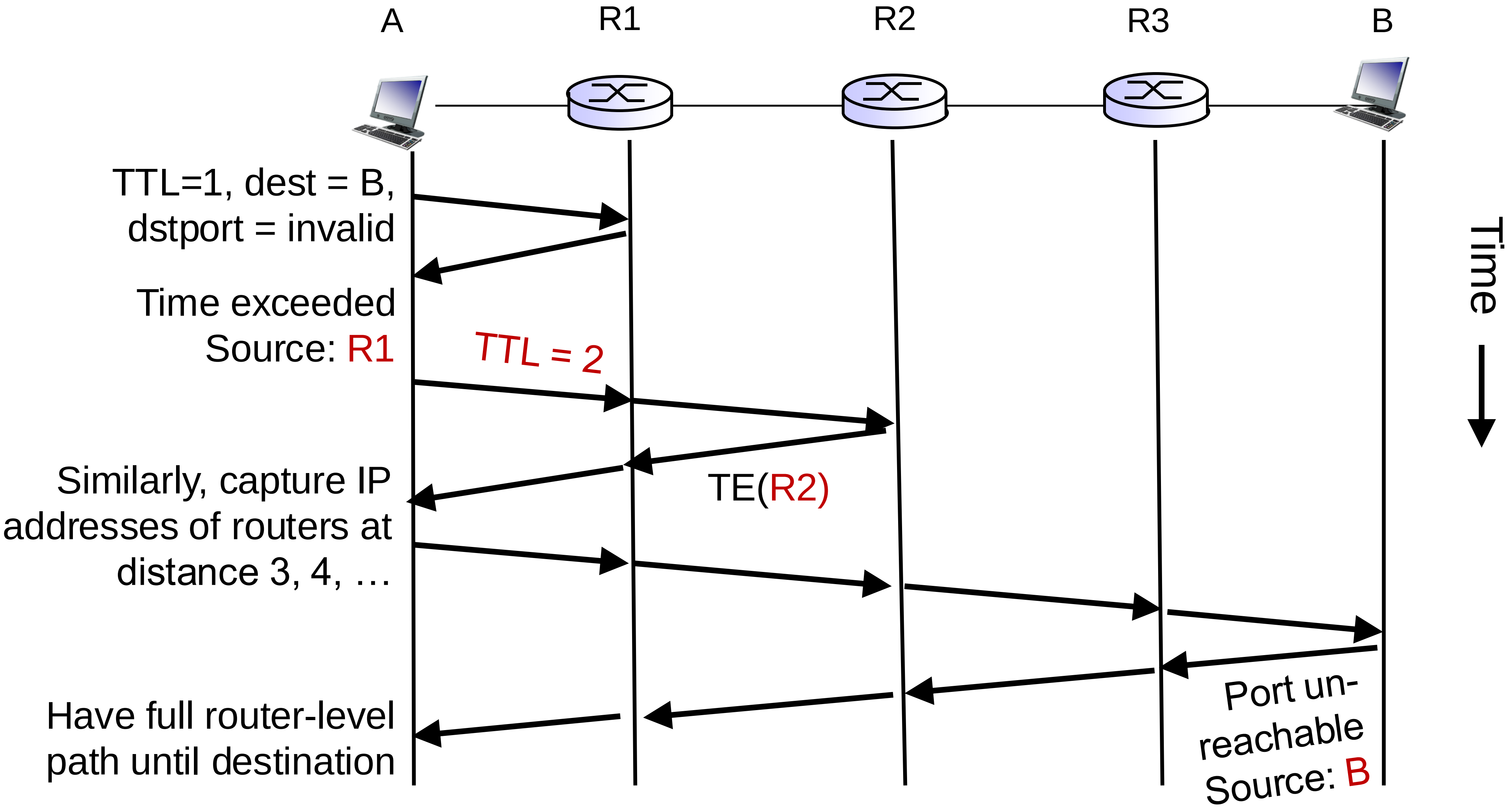
Traceroute

- A tool that can record the router-level path taken by packets
- A clever use of the IP time-to-live (TTL) field
- In general, when a router receives an IP packet, it decrements the TTL field on the packet
  - A failsafe mechanism to ensure packets don't keep taking up network resources for too long
- If a router receives a packet with TTL=0, it sends an ICMP time exceeded message (type=11, code=0) to the source endpoint

Traceroute

- Traceroute sends multiple packets to a destination endpoint
- But it progressively increases the TTL on those packets: 1, 2, ...
- Every time a time exceeded message is received, record the router's IP address
- Process repeated until the destination endpoint is reached
- If the packet reaches the destination endpoint (i.e.: TTL is high enough), then the endpoint sends a port unreachable message (type=3, code=3)

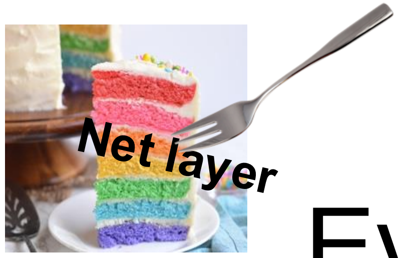
Traceroute



A
R1
R2
R3
B
TTL=1, dest = B,
dstport = invalid
Time exceeded
Source: R1
TTL = 2
Similarly, capture IP
addresses of routers at
distance 3, 4, …
TE(R2)
Time
Have full router-level
path until destination
Port un-
reachable
Source: B

Summary of ICMP

- A protocol for network diagnostics and troubleshooting
- Two useful tools: ping and traceroute
- Ping: test connectivity to a machine totally outside your control
  - Use ICMP echo request and reply
- Traceroute: determine router-level path to a remote endpoint
  - A smart use of the TTL field in the IP header

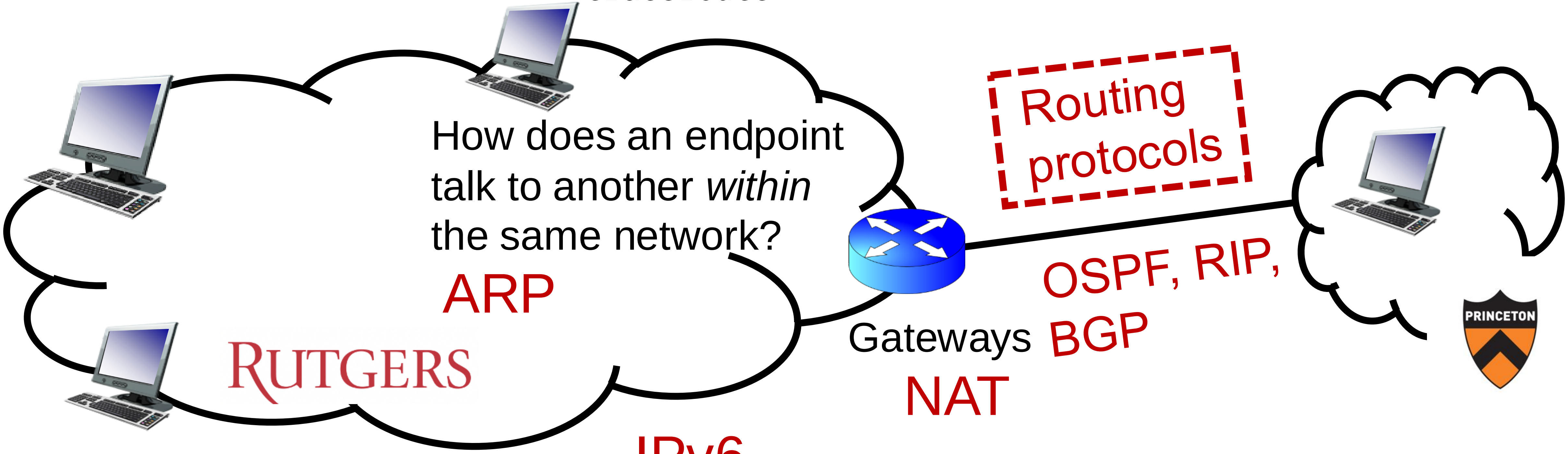


The network layer enables reachability. Every protocol below solves a sub-problem.

How does an endpoint get an address?
DHCP

Debugging
ICMP
`ping`
`traceroute`

How does an endpoint talk to another *outside* its network?



IPv6

Routing is a fundamental problem in networking.

How would one design a “Google Maps”
to navigate the Internet?

Goals of Routing Protocols #1

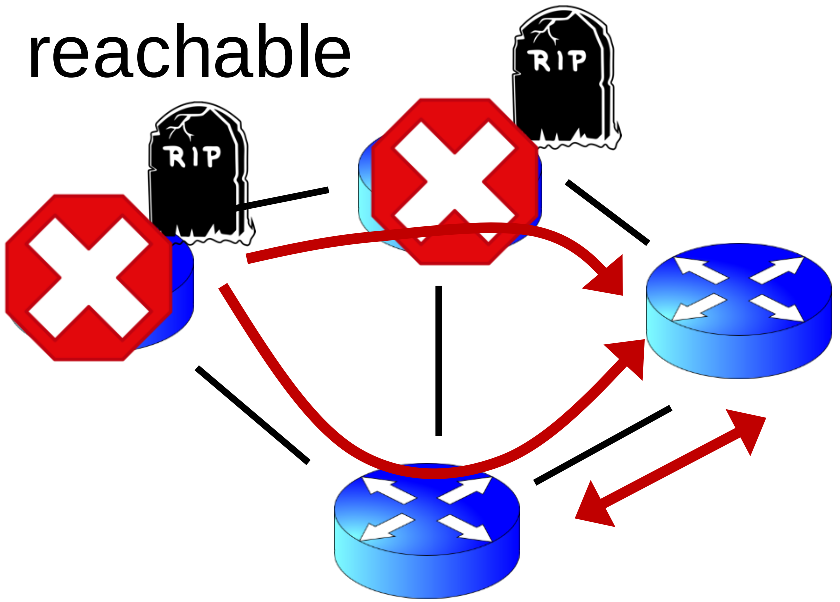
- Determine good paths from source to destination

- “Good” = least cost
  - Least propagation delay
  - Least cost per unit bandwidth (e.g., $ per Gbit/s)
  - Least congested (workload-driven)

- “Path” = a sequence of router ports (links)

Goals of Routing Protocols #2

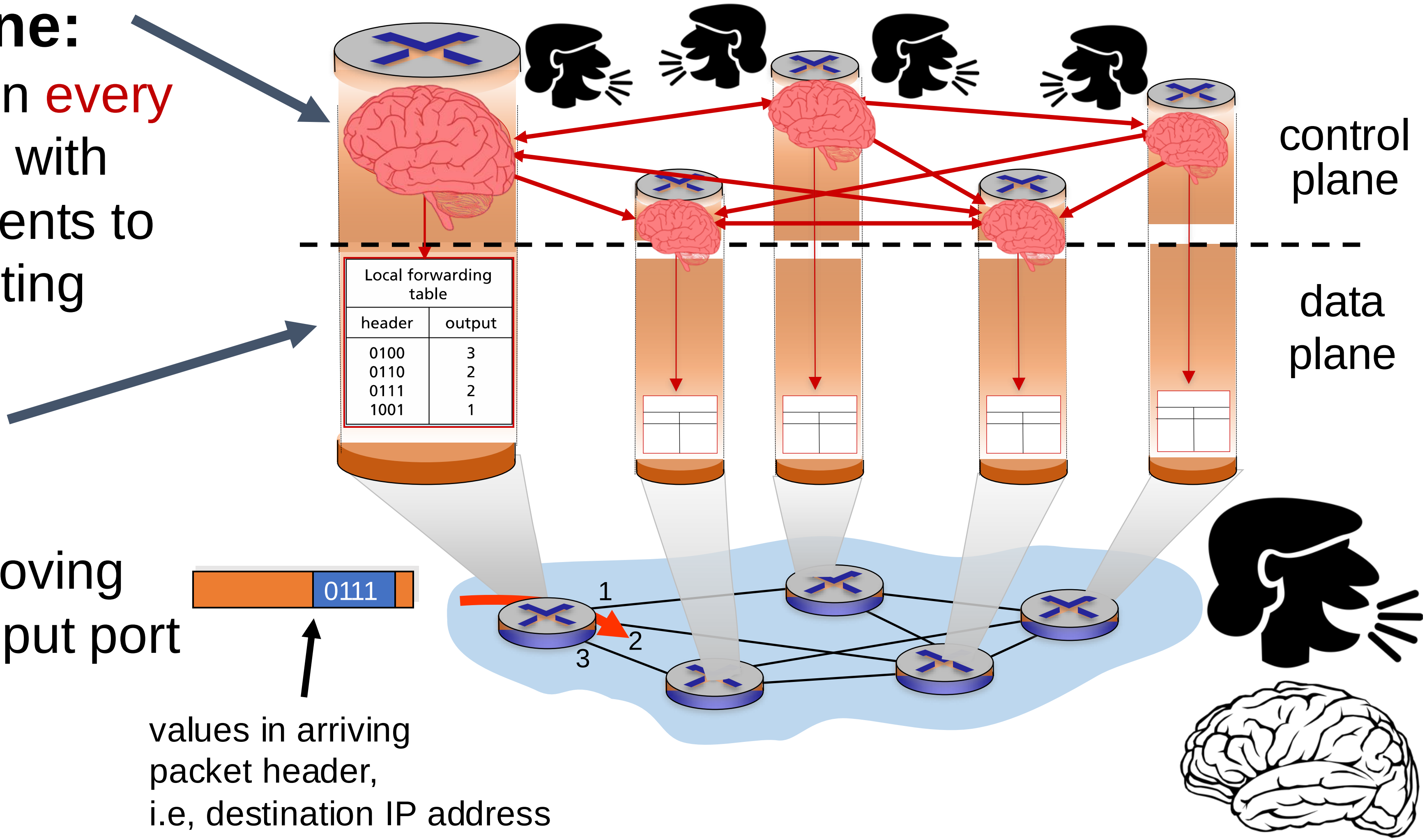
- Make networks resilient to failures
- Routers & links can fail without taking down the entire network
- Entire subsets can be unreachable; rest still reachable
- Hence, the protocol must be distributed



Per-router control plane

Distributed control plane:
Components in every router interact with other components to produce a routing outcome.

Data plane
per-packet processing, moving packet from input port to output port



Routing protocol

Q1. What info exchanged?

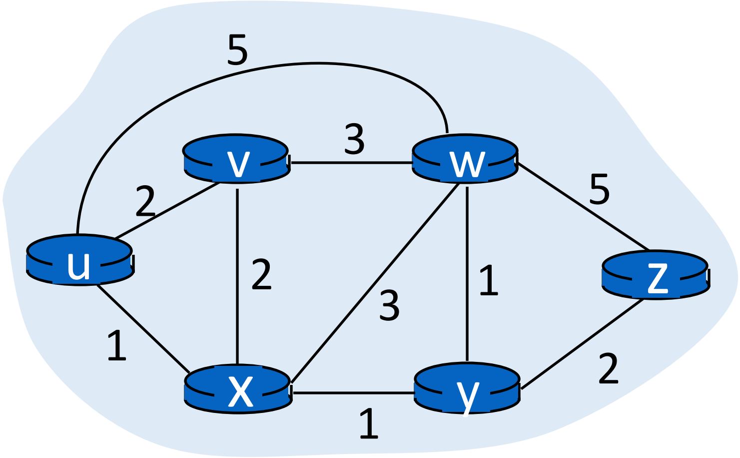
Q2. What computation?

The graph abstraction

- Routing algorithms work over an abstract representation of a network: the graph abstraction

Ex: Rutgers campus

u: Computer Science
v: School of Engineering
…



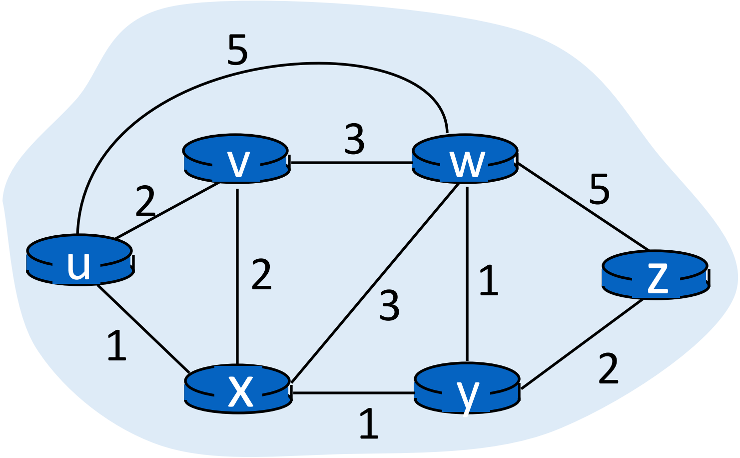
- Each router is a node in a graph
- Each link is an edge in the graph
- Edges have weights (also called link metrics). Set by netadmin

The graph abstraction

- Routing algorithms work over an abstract representation of a network: the graph abstraction

Ex: Rutgers campus

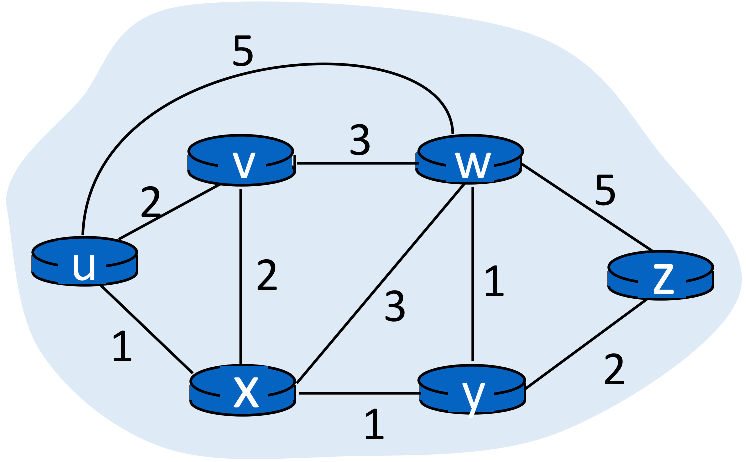
u: Computer Science
v: School of Engineering
…



- G = (N, E)
- N = {u, v, w, x, y, z}
- E = { (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

The graph abstraction

- Cost of an edge: c(x, y)
  - Examples: c(u, v) = 2, c(u, w) = 5
- Cost of a path = sum of edge costs
  - c(path x → w → y → z) = 3 + 1 + 2 = 6



- Outcome of routing: each node should determine the least cost path to every other node
- Q1: What information should nodes exchange with each other to enable this computation?
- Q2: What algorithm should each node run to compute the least cost path to every node?

Coming up next

Routing protocols

Link state protocols

Each router has complete information of the graph

Messages exchanged by flooding all over the network

Communication expensive, but complete

Distance vector protocols

Each router only maintains distances & next hop to others

Messages are exchanged only between neighbors

Communication cheap, but incomplete

Link State Protocols

Link state protocol

- Each router knows the state of all the links and routers in the network

- Every router performs an independent computation on globally shared knowledge of network's complete graph representation

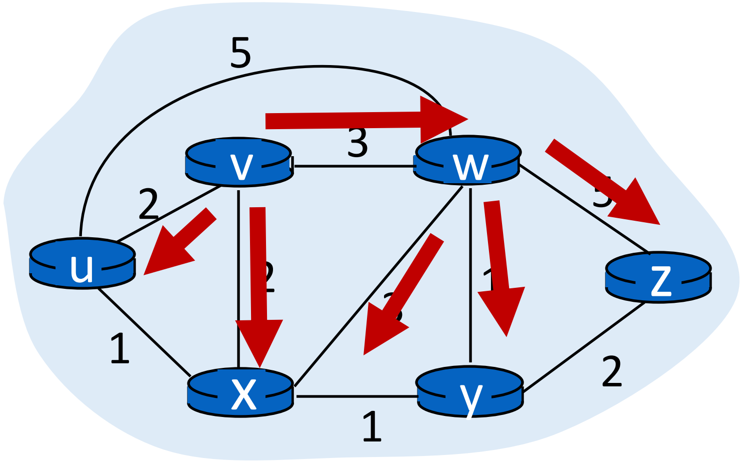
Routing protocols

Link state protocols

Distance vector protocols

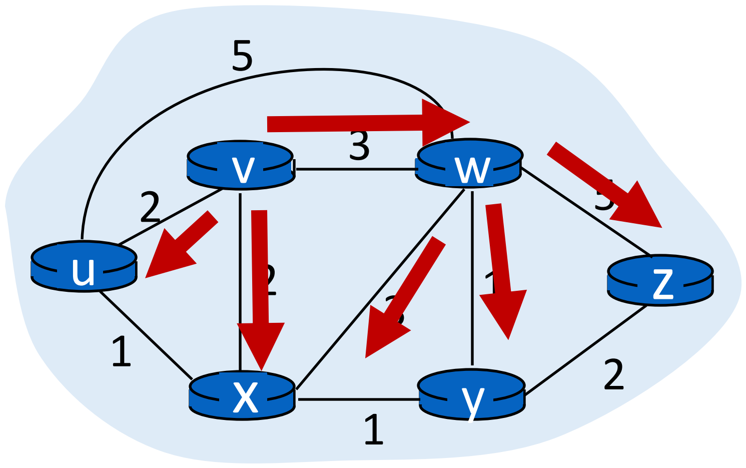
Q1: Information exchange

- Link state flooding: the process by which neighborhood information of each network router is transmitted to all other routers
- Each router sends a link state advertisement (LSA) to each of its neighbors
- LSA contains the router ID, the IP prefix owned by the router, the router's neighbors, and link cost to those neighbors
- Upon receiving an LSA, a router forwards it to each of its neighbors: flooding

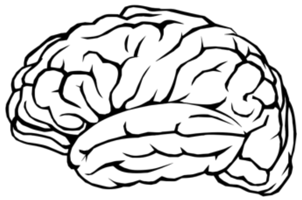


Q1: Information exchange

- Eventually, the entire network receives LSAs originated by each router
- LSAs put into a link state database
- LSAs occur periodically and whenever the graph changes
  - Example: if a link fails
  - Example: if a new link or router is added
- The routing algorithm running at each router can use the entire network's graph to compute least cost paths



Q2: The algorithm



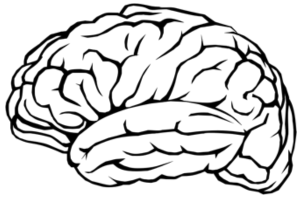
Dijkstra's algorithm

- Given a network graph, the algorithm computes the least cost paths from one node (source) to all other nodes
- This can then be used to compute the forwarding table at that node
- Iterative algorithm: maintain estimates of least costs to reach every other node. After k iterations, each node definitively knows the least cost path to k destinations

Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current estimate of cost of path from source to destination v
- p(v): (predecessor node) the last node before v on the path from source to v
- N': set of nodes whose least cost path is definitively known

Dijsktra’s Algorithm



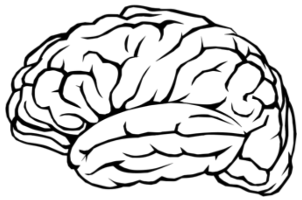
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

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'
```

Initial estimates of distances are just the link costs of neighbors.

Least cost node among all estimates. This cost cannot decrease further.

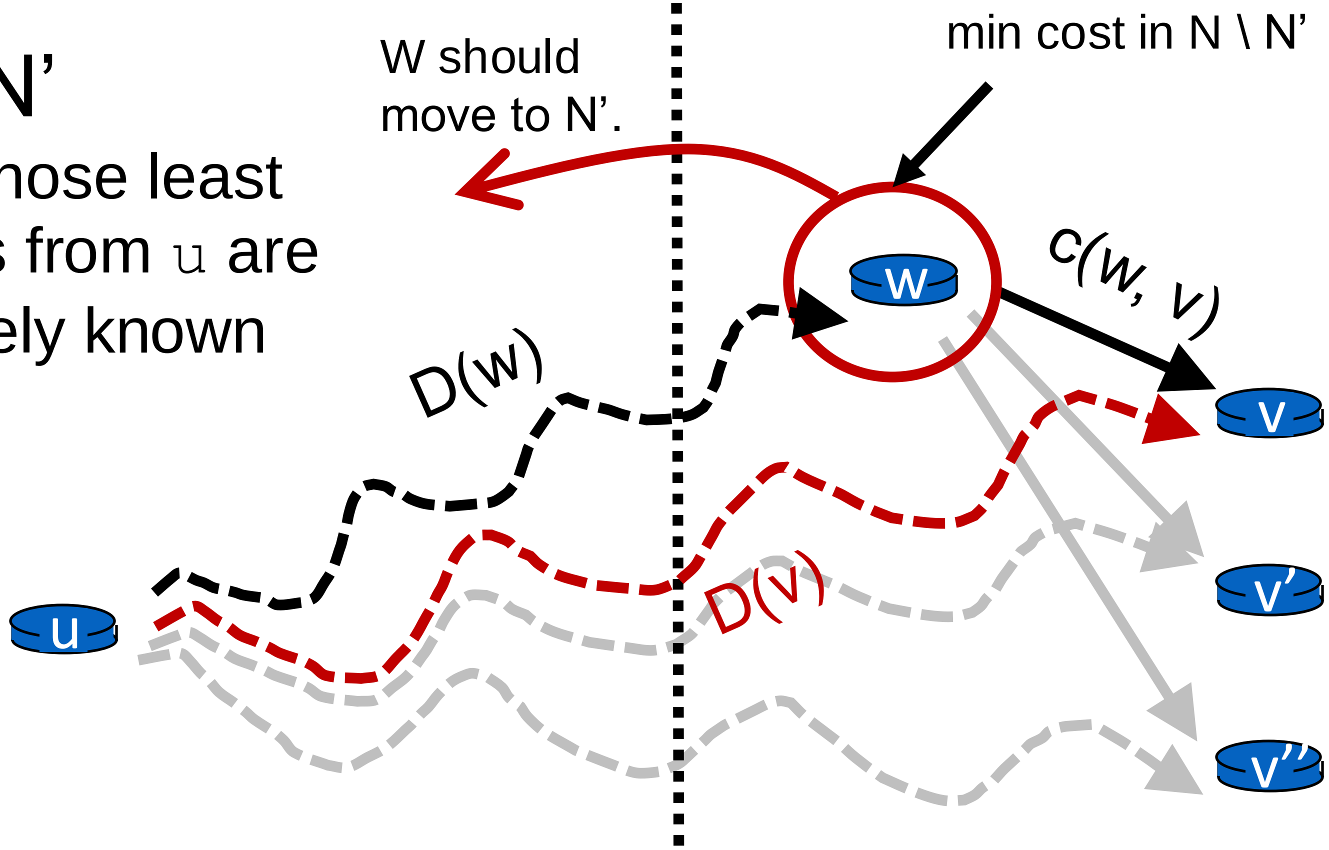
Relaxation

Visualization



N’

nodes whose least cost paths from `u` are definitively known



N \ N’

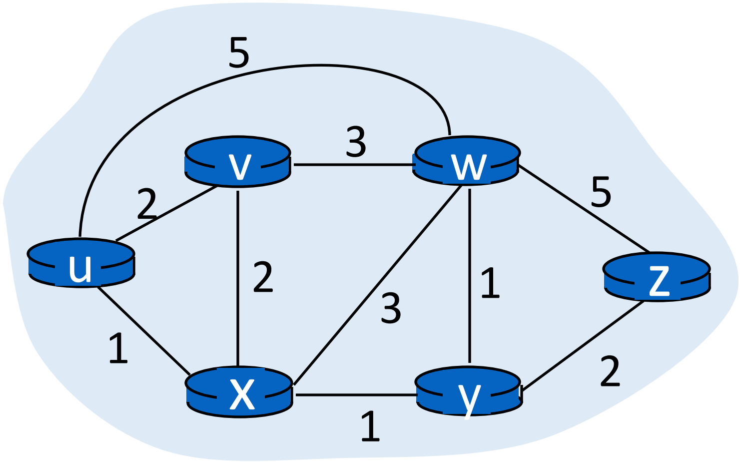
Nodes with estimated least path costs, not definitively known to be smallest possible

Relaxation: for each v in N \ N’, is the cost of the path via w smaller than known least cost path to v?
If so, update D(v)
Predecessor of v is w.

Cost of path via w: D(w) + c(w,v)
Cost of known best path: D(v)

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					



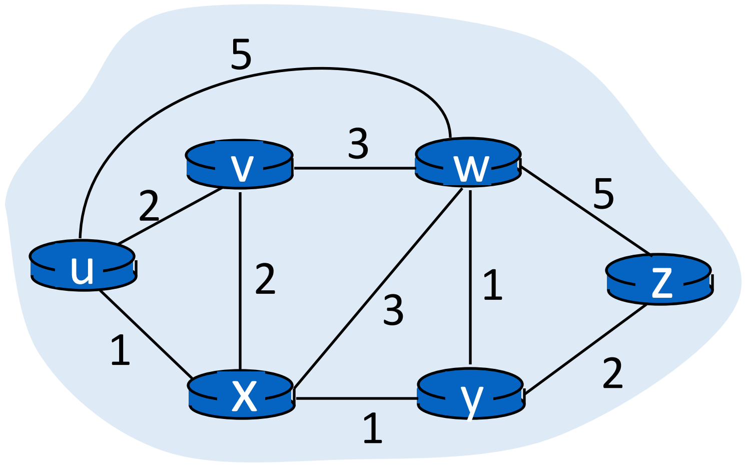
Constructing the forwarding table

- To find the router port to use for a given destination (router), find the predecessor of the node iteratively until reaching an immediate neighbor of the source `u`

- The port connecting `u` to this neighbor is the output port for this destination

Constructing the forwarding table

- Suppose we want forwarding entry for z.



D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
2,u	3,y	1,u	2,x	4,y

z: p(z) = y
y: p(y) = x
x: p(x) = u
x is an immediate neighbor of u

Forwarding table at `u`:

destination	link
z	(u,x)

Summary of link state protocols

- Each router announces link state to the entire network using flooding

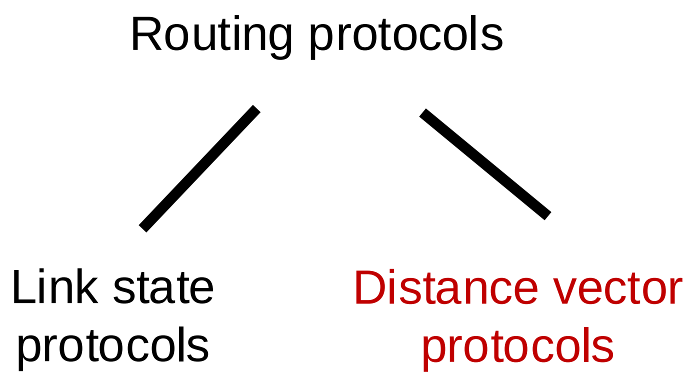
- Each node independently computes least cost paths to every other node using the full network graph

- Dijkstra’s algorithm can efficiently compute these best paths
  - Easy to populate the forwarding table from predecessor information computed during the algorithm

Distance Vector Protocols

Distance Vector Protocol

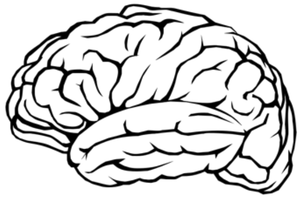
- Each router only exchanges a distance vector with its neighbors
  - Distance: how far the destination is
  - Vector: a value for each destination
- DVs are only exchanged between neighbors; not flooded
- Use incomplete view of graph derived from neighbors' distance vectors to compute the shortest paths



Q1: Distance Vectors

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x knows cost of edge to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$
- Nodes exchange distance vector periodically and whenever the local distance vector changes (e.g., link failure, cost changes)

Q2: Algorithm



Bellman-Ford algorithm

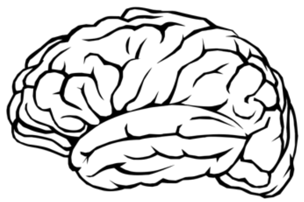
- Each node initializes its own distance vector (DV) to edge costs
- Each node sends its DVs to its neighbors
- When a node `x` receives new DV from a neighbor `v`, it updates its own DV using the Bellman-Ford equation:
- Given $d_x(y)$:= estimated cost of the least-cost path from x to y
- Update $d_x(y) = \min_v \{c(x,v) + d_v(y)\}$

cost of path from neighbor v to destination y

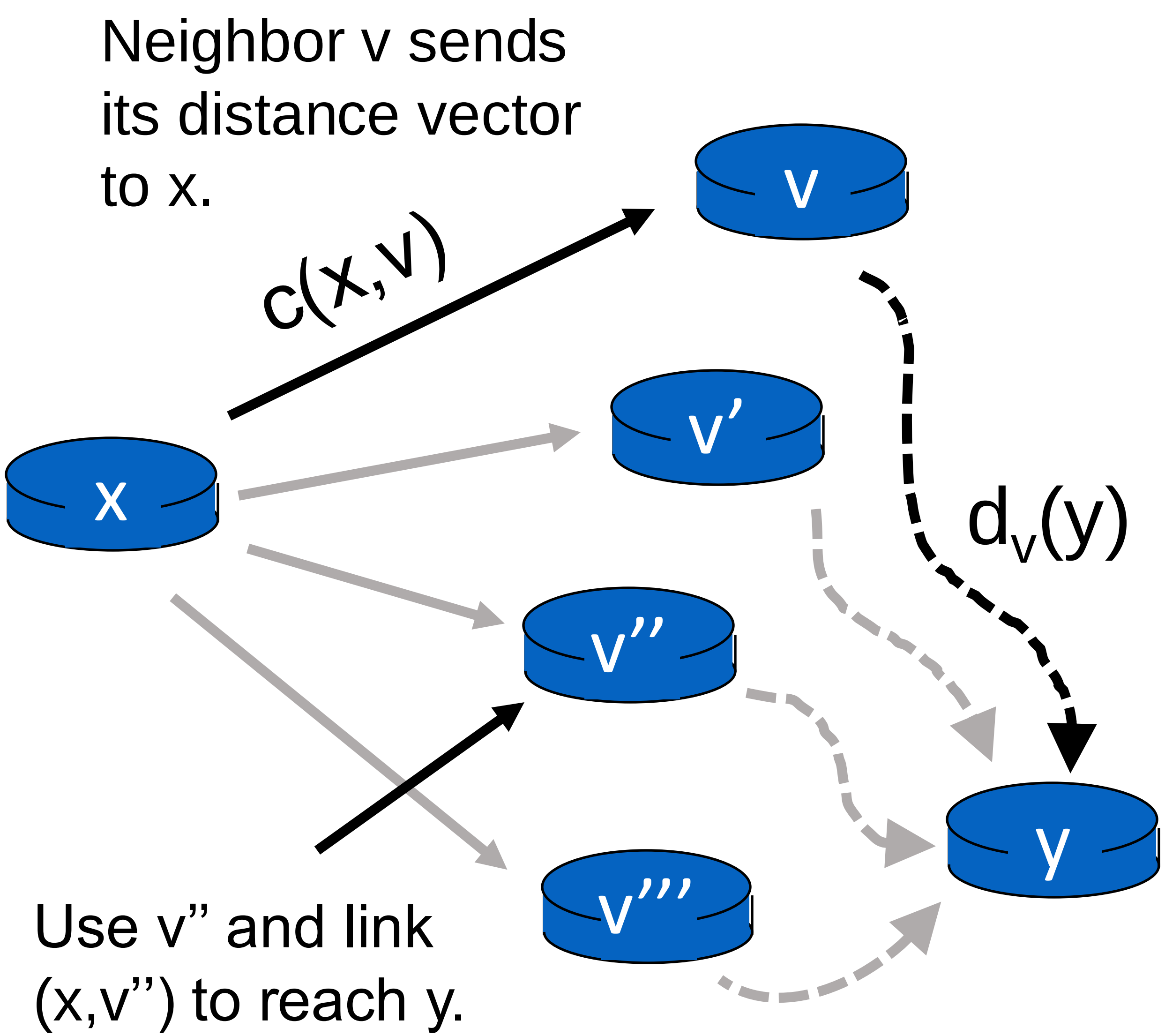
minimum taken over all neighbors v of x

cost to reach neighbor v directly from x

Visualization



- Which neighbor v offers the current best path from x to y?
- Path through neighbor v has cost c(x,v) + d_v(y)
- Choose min-cost neighbor
- Remember min-cost neighbor as the one used to reach node y
- This neighbor determines the output port!



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

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

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

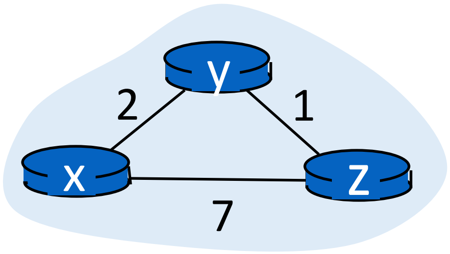
node z table

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

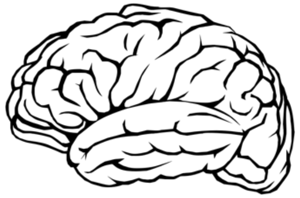
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

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

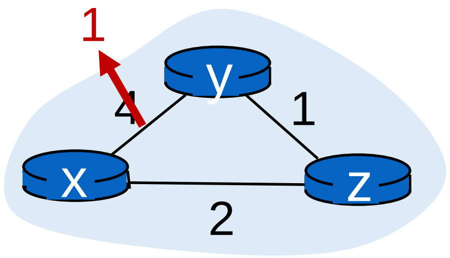
time



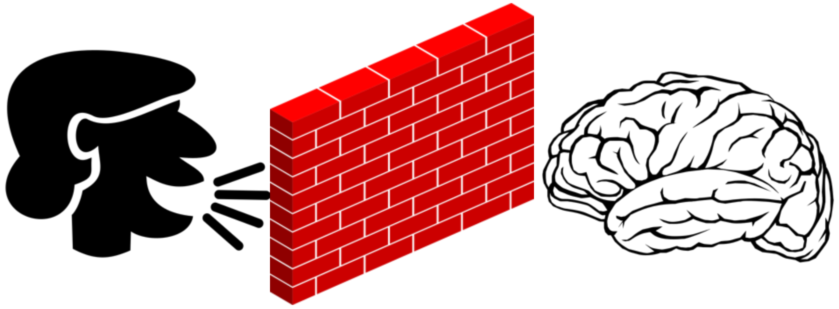
Good news travels fast



- Suppose the link cost reduces or a new better path becomes available in a network.
- The immediate neighbors of the change detect the better path immediately
- Since their DV changed, these nodes notify their neighbors immediately.
  - And those neighbors notify still more neighbors
  - … until the entire network knows to use the better path
- Good news travels fast through the network
- This is despite messages only being exchanged among neighbors



Bad news travels slowly



- If router goes down, could be a while before network realizes it.

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges

B still thinks it can reach A through C… bad!

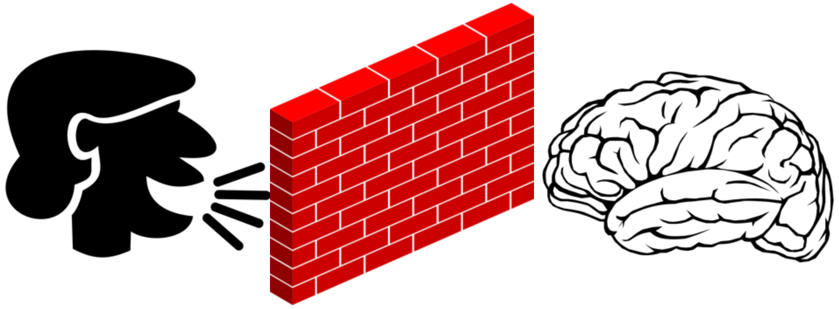
C thinks it can reach A through B… worse!

B, D think they can reach A through C… ugly!

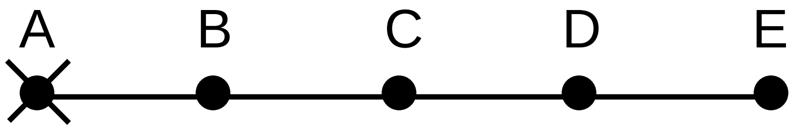
Count to infinity problem

etc… to infinity

Bad news travels slowly



- Reacting appropriately to bad news requires information that only other routers have. DV does not exchange sufficient info.



- B needs to know that C has no other path to A other than via B.
- DV does not exchange paths; just distances!
- Poisoned reverse: if X gets its route to Y via Z, then X will announce $d_X(Y) = \infty$ in its message to Z
  - Effect: Z won't use X to route to Y
  - However, this won't solve the problem in general (think why.)

Summary: Comparison of LS and DV

Link State Algorithms

- Nodes have full visibility into the network's graph
- Copious message exchange: each LSA is flooded over the whole network
- Robust to network changes and failures

OSPF
Open Shortest Path First
(v2 RFC 2328)

Distance Vector Algorithms

- Only distances and neighbors are visible
- Sparse message exchange: DVs are exchanged among neighbors only
- Brittle to router failures. Incorrect info may propagate all over net

EIGRP
Enhanced Interior Gateway Routing Protocol
(RFC 7868)