# CS 352
# Routing Algorithms: Intro

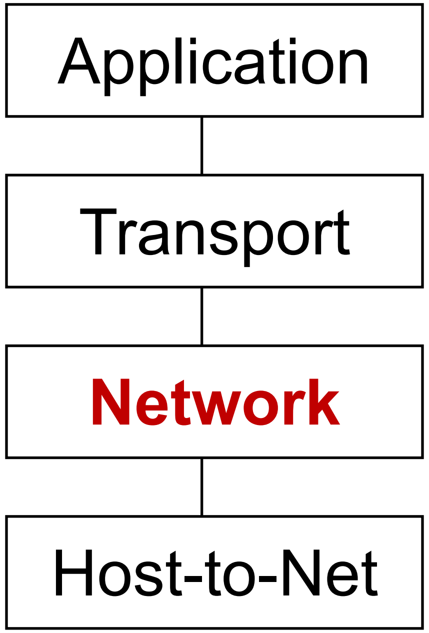
CS 352, Lecture 18.1

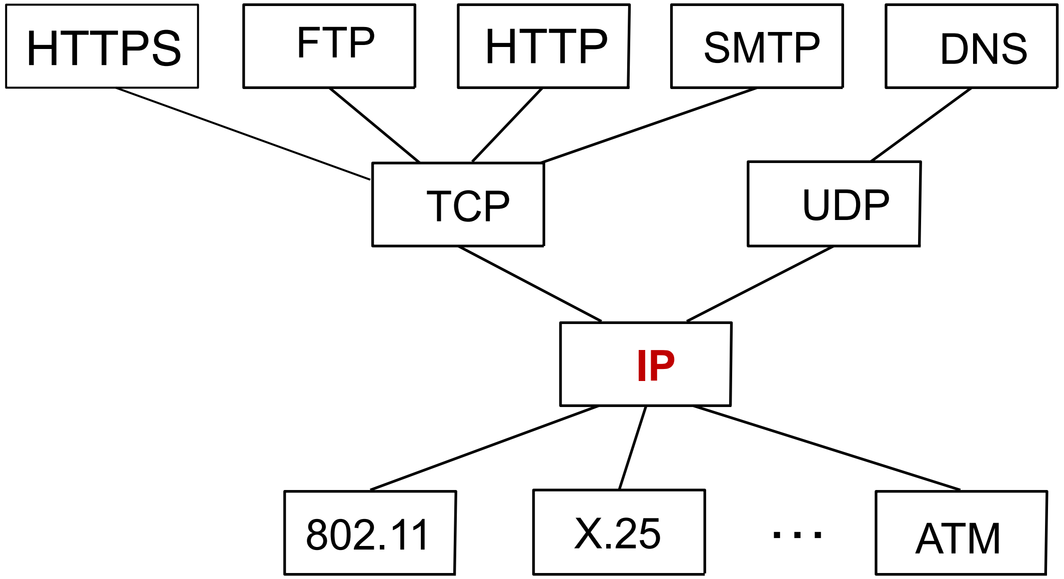
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

# Network




The main function of the network layer is to
move packets from one endpoint to another.

How would one design a "Google Maps" for the Internet?

# Review: Network layer functions

- Forwarding: move packets from router's input to appropriate router output

- Routing: determine route taken by packets from source to destination
  - routing algorithms

- The network layer solves the routing problem.

- Data Plane

- Control Plane

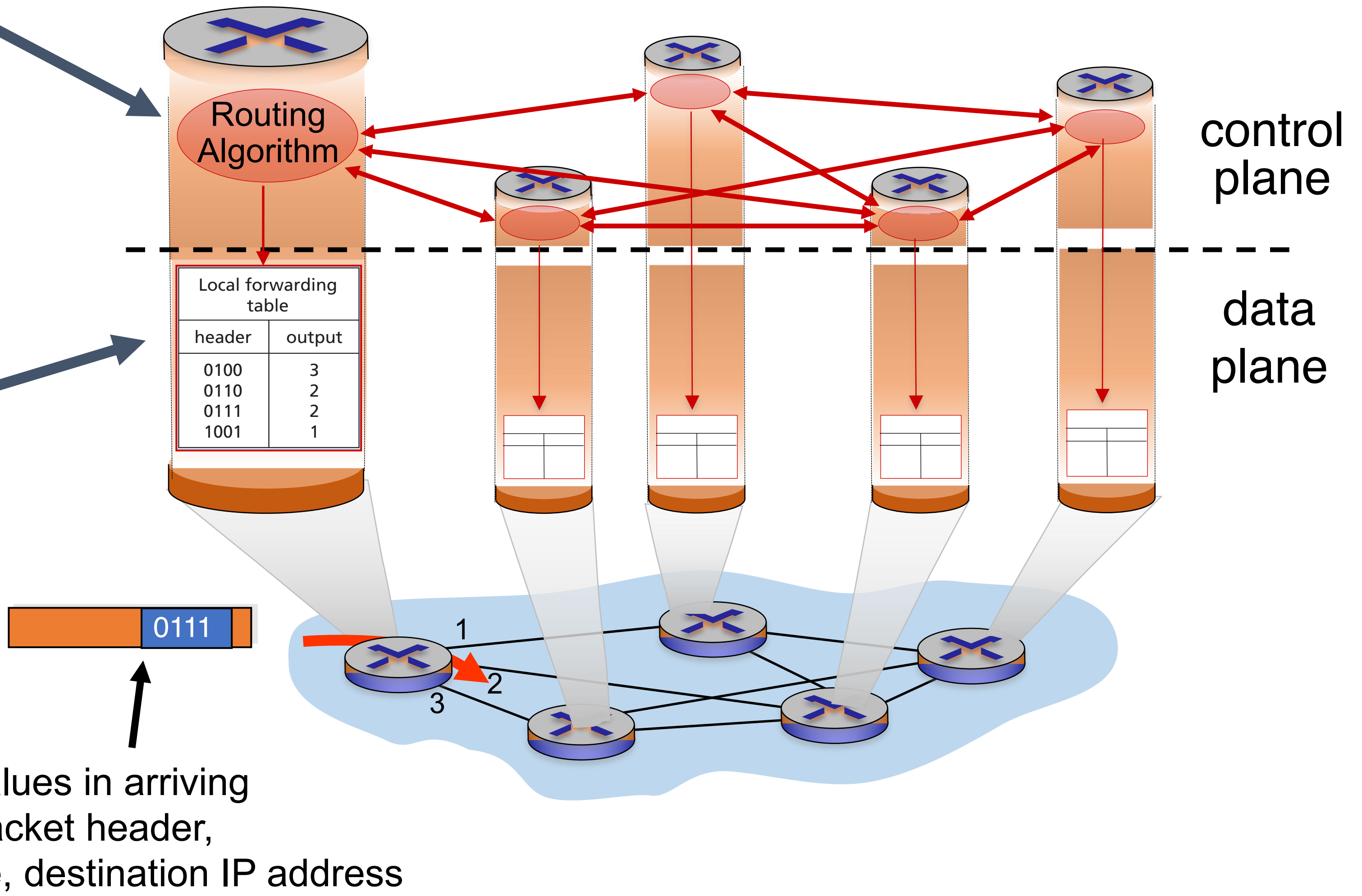
- Two kinds of control planes:
  - Distributed per-router control
  - Logically centralized

The next 2 lectures

# Review: 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



# Goal of Routing Algorithms

- 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)

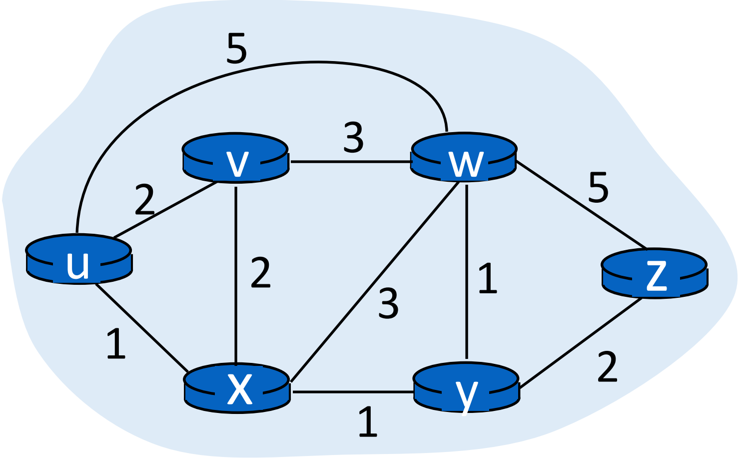
- Routing is a fundamental problem in networking.

# 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
…



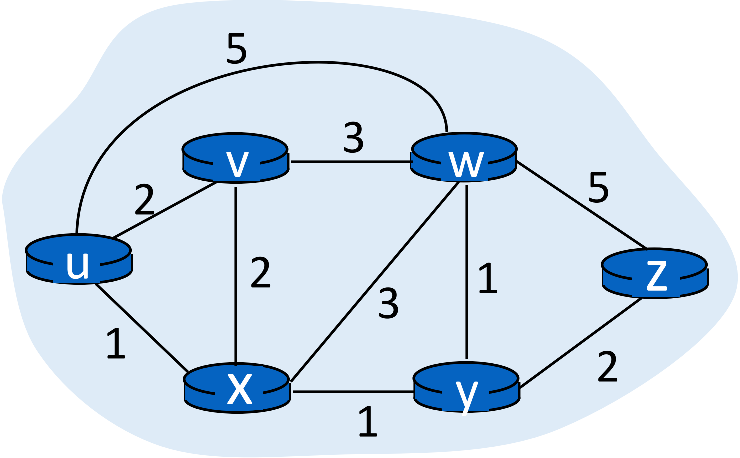

- Each router is a node in a graph
- Each link is an edge in the graph
- Edges have weights (also called link metrics). Set by admin

# The graph abstraction

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

Ex: Rutgers campus

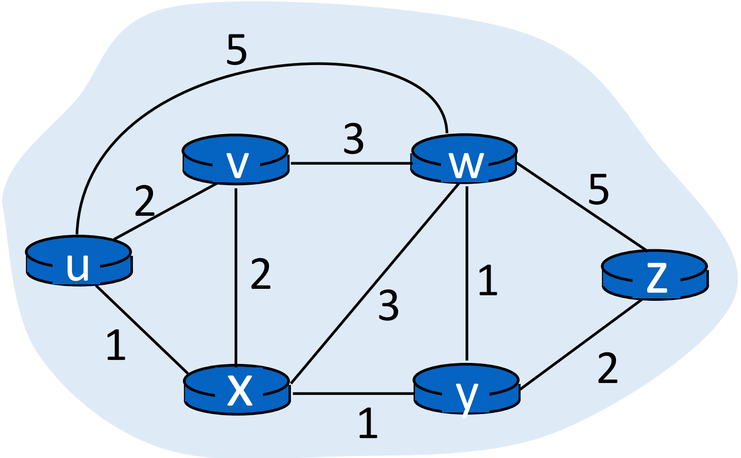
u: Computer Science
v: School of Engineering
…




- 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 = summation 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 algorithm should each node run to compute the least cost path to every node?
- Q2: What information should nodes exchange with each other to enable this computation?

# The rest of this lecture

Routing protocols

## Link state protocols

Each router has complete information of the graph

Information shared by flooding over the network

Message exchanges expensive

## Distance vector protocols

Each router only maintains distances to other routers

Messages are exchanged over each link and stay within the link

Message exchanges cheap

# CS 352
# Link State Protocols

CS 352, Lecture 18.2

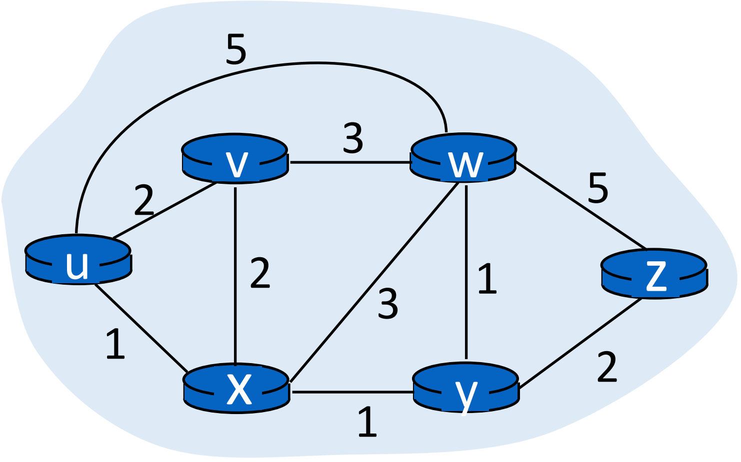
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

# Review: Routing & Link State Algorithms

- Distributed routing protocols
- Goal of routing algorithms: find least cost path in a graph abstraction of the network
- Link state algorithm: Each router has full visibility of the graph, i.e., the “states” of all links
- Q1: what algorithm runs at each node?
- Q2: what information is exchanged?



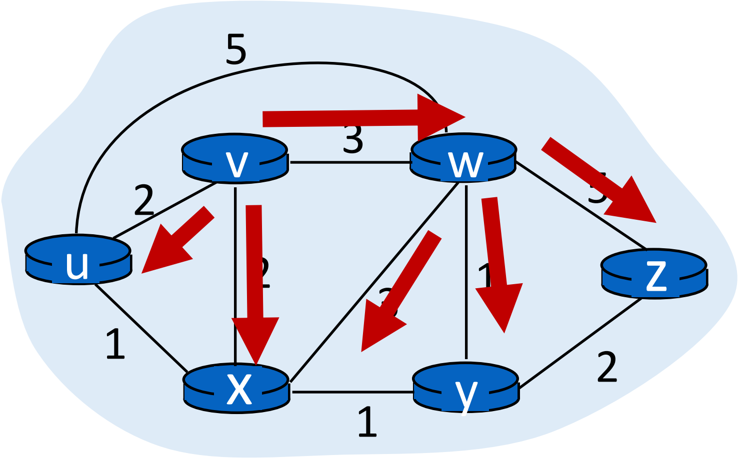

Routing protocols

Link state protocols

Distance vector protocols

# Q2: 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

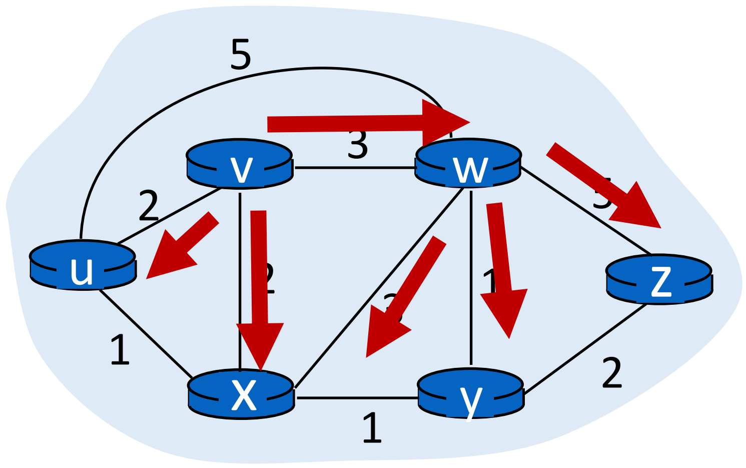


# Q2: Information exchange

- Eventually, the entire network receives LSAs originated by each router

- 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



# Q1: 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; $= \infty$ 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.

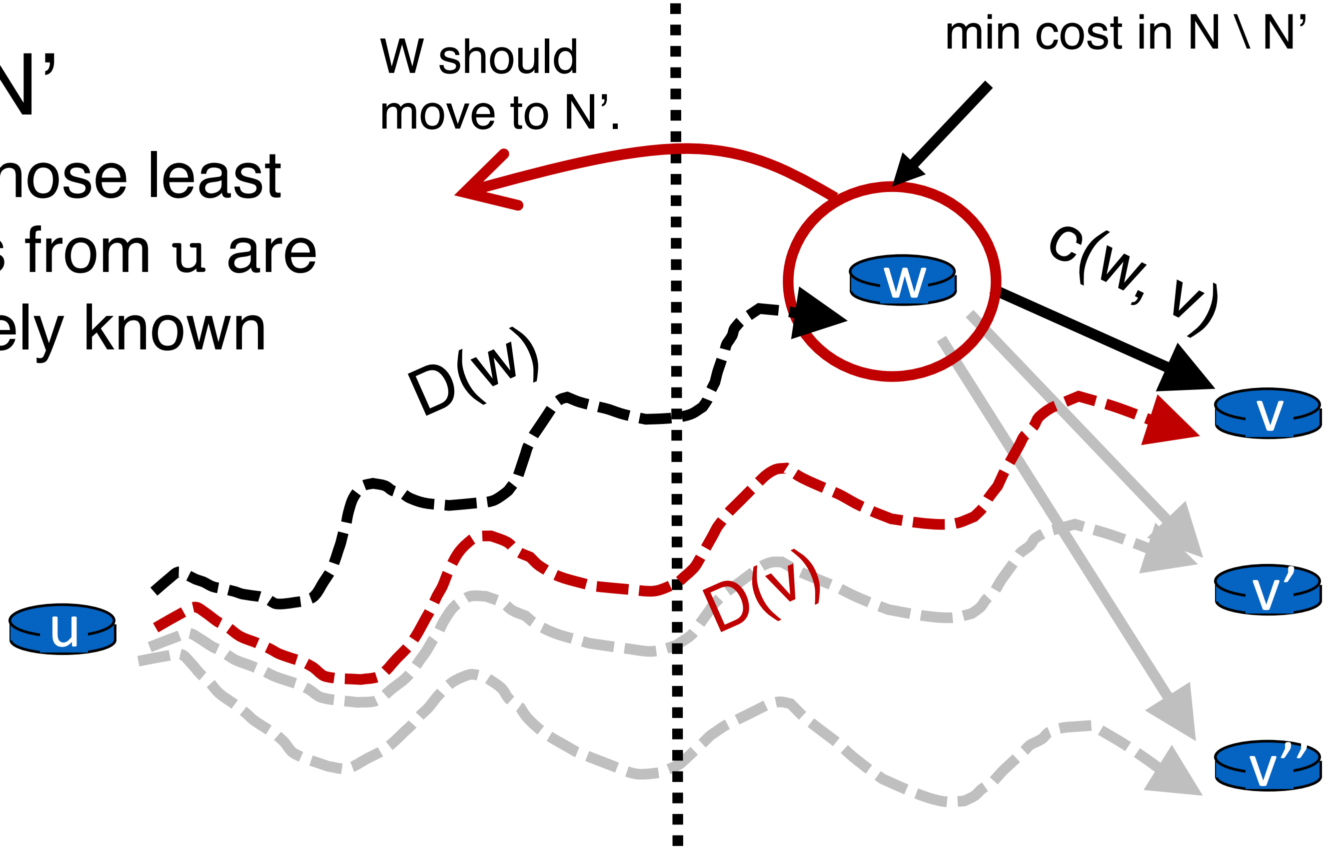
Least cost node among all estimates. This cost cannot decrease further.

Relaxation

# Visualization

**N'**

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




**N \ N'**

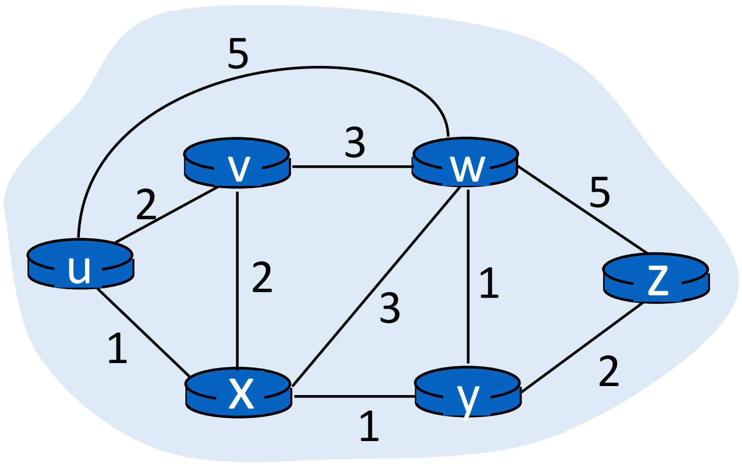
Nodes with estimated least path costs, not definitively known

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 | | | | | |



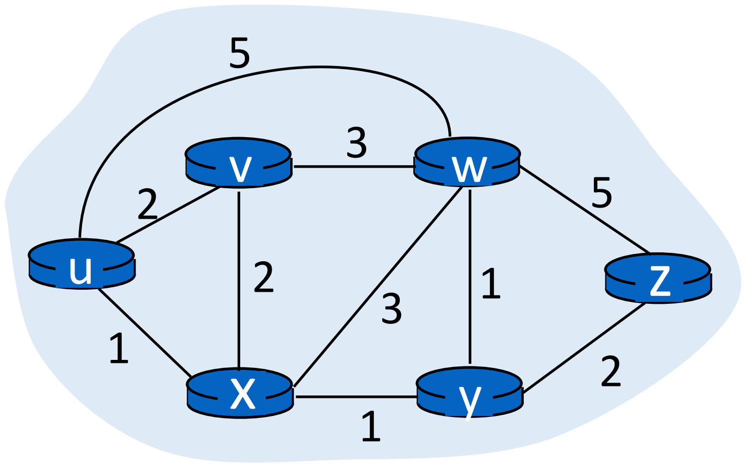
# Constructing the forwarding table

- 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

# CS 352
# Distance Vector Protocols

CS 352, Lecture 18.3

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

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

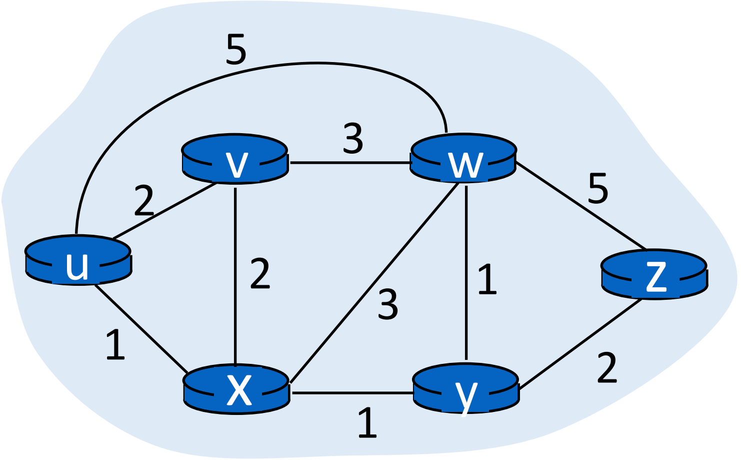
# Review: Routing & Dist Vector Algorithms

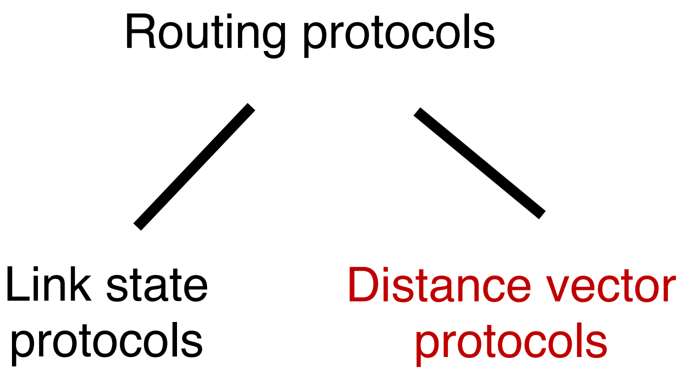
- Distributed routing protocols

- Goal of routing algorithms: find least cost path in a graph abstraction of the network

- DV proto: Each router maintains a vector of distances to all other routers; not the graph.

- Q1: what algorithm runs at each node?
- Q2: what information is exchanged?



# Q2: Exchanged info = Distance Vectors

- Nodes exchange distance vectors with their neighbors
  - No flooding unlike link state protocols. Message not propagated further
- $D_x(y)$ = 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)

# Q1: 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 $d_x(y) = \min_v \{c(x,v) + d_v(y) \}$

minimum taken over all neighbors v of x

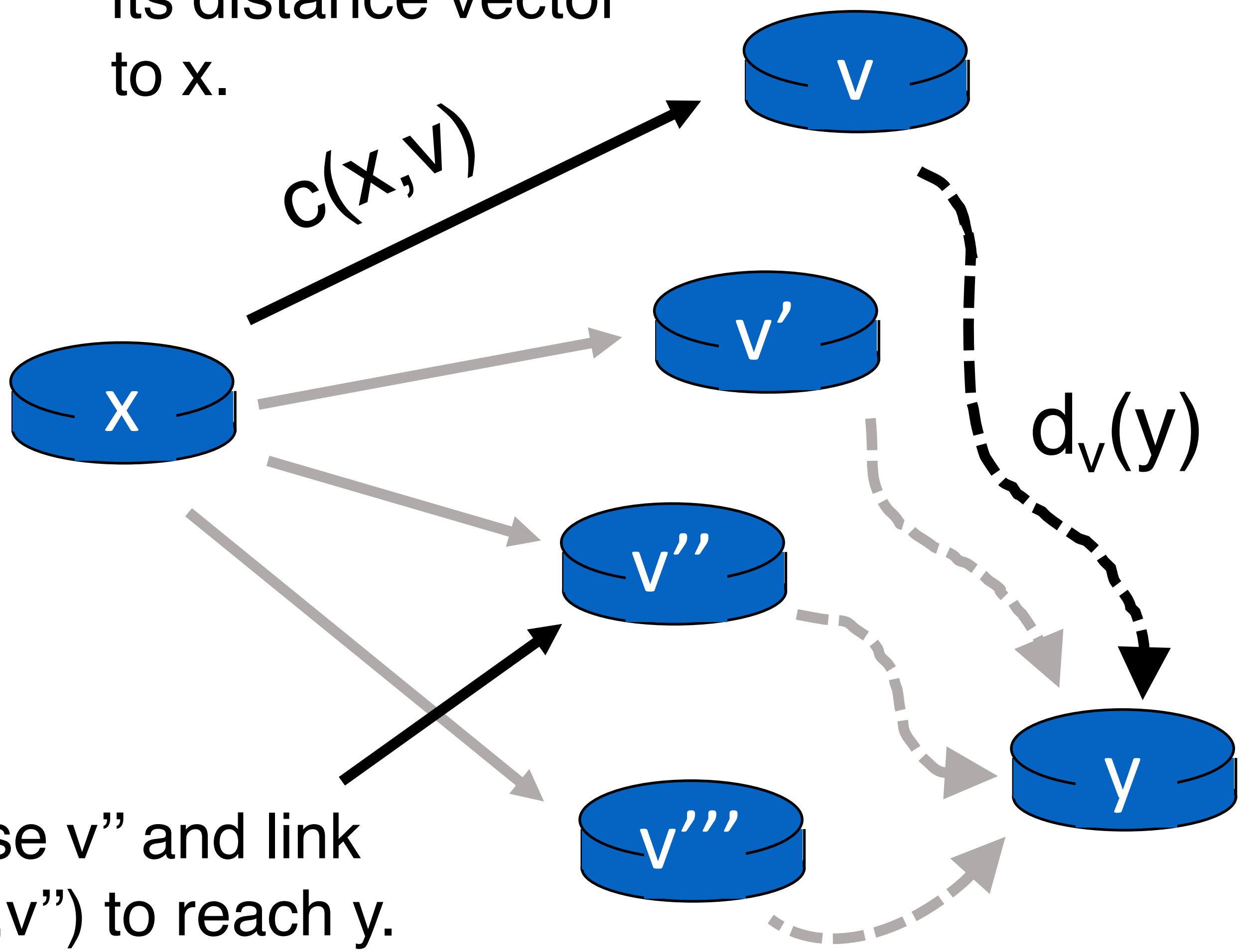
cost to reach neighbor v directly from x

cost of path from neighbor v to destination y

# 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 for the packet.

Neighbor v sends its distance vector to x.




Use v'' and link (x,v'') to reach y.

# Q1: Algorithm

## Bellman-Ford algorithm

- By iteratively performing Bellman-Ford iterations, under some conditions, the estimate $d_x(y)$ converges to the true cost of the least cost path from x to y.

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

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

**node x table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | ∞ | ∞ | ∞ |
| z | ∞ | ∞ | ∞ |

cost to

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

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node y table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node z table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |

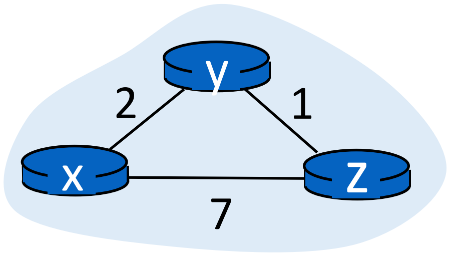
cost to

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

cost to

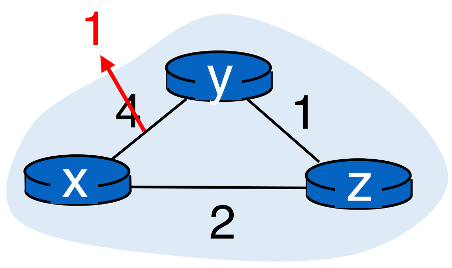
| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

time



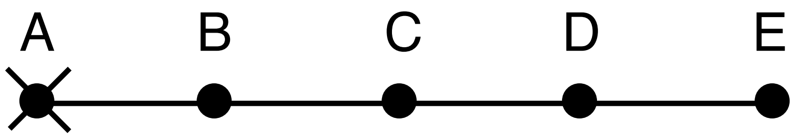
# Good news with distance vector protocols

- 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 with distance vector protocols

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

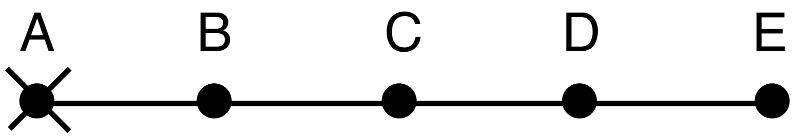



| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | Initially |
| 3 | 2 | 3 | 4 | After 1 exchange |
| 3 | 4 | 3 | 4 | After 2 exchanges |
| 5 | 4 | 5 | 4 | After 3 exchanges |
| 5 | 6 | 5 | 6 | After 4 exchanges |
| 7 | 6 | 7 | 6 | After 5 exchanges etc… to infinity |

Count to infinity problem

# Bad news travels slowly

- Reacting appropriately to bad news requires information that only other routers have.




- B needs to know that C has no other path to A other than via B.

- **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.)
- Fundamentally, DV protocols must exchange more information to react robustly to network changes and router errors.

# Summary: Comparison of LS and DV

## Link State Algorithms

- Nodes have full visibility into the network's graph
- Message complexity is high: each LSA is flooded over the whole network
- In general, robust to network changes. Scope of incorrect info is limited to bad LSAs.

## Distance Vector Algorithms

- Only distances and neighbors are visible
- Message complexity is low: DVs are exchanged among neighbors only
- Brittle against bad news and router bugs: incorrect info can propagate throughout a network

# Deployed routing protocols

- The algorithms we've seen are widely deployed in real ISPs
- Link-state protocols
  - OSPF (Open Shortest Path First)
  - IS-IS: (Intermediate System to Intermediate System)
- Distance-vector protocols
  - RIP: Routing Information Protocol
  - IGRP: Interior Gateway Routing Protocol
- You're likely watching this video over a network running one of these protocols to determine how data should reach your machine.