

# CS 330: Network Applications & Protocols

## Network Layer

---

Galin Zhelezov  
Department of Physical Sciences  
York College of Pennsylvania



# Overview of Network Layer

---

- **Virtual Circuit and Datagram Networks**
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
  - Overview
  - Link state
  - Distance vector
- **Routing in the Internet**
- **Broadcast and multicast routing**

# Network-layer functions

---

*Recall: two network-layer functions:*

- **forwarding:** move packets from router's input to appropriate router output
- **routing:** determine route taken by packets from source to destination

*data plane*

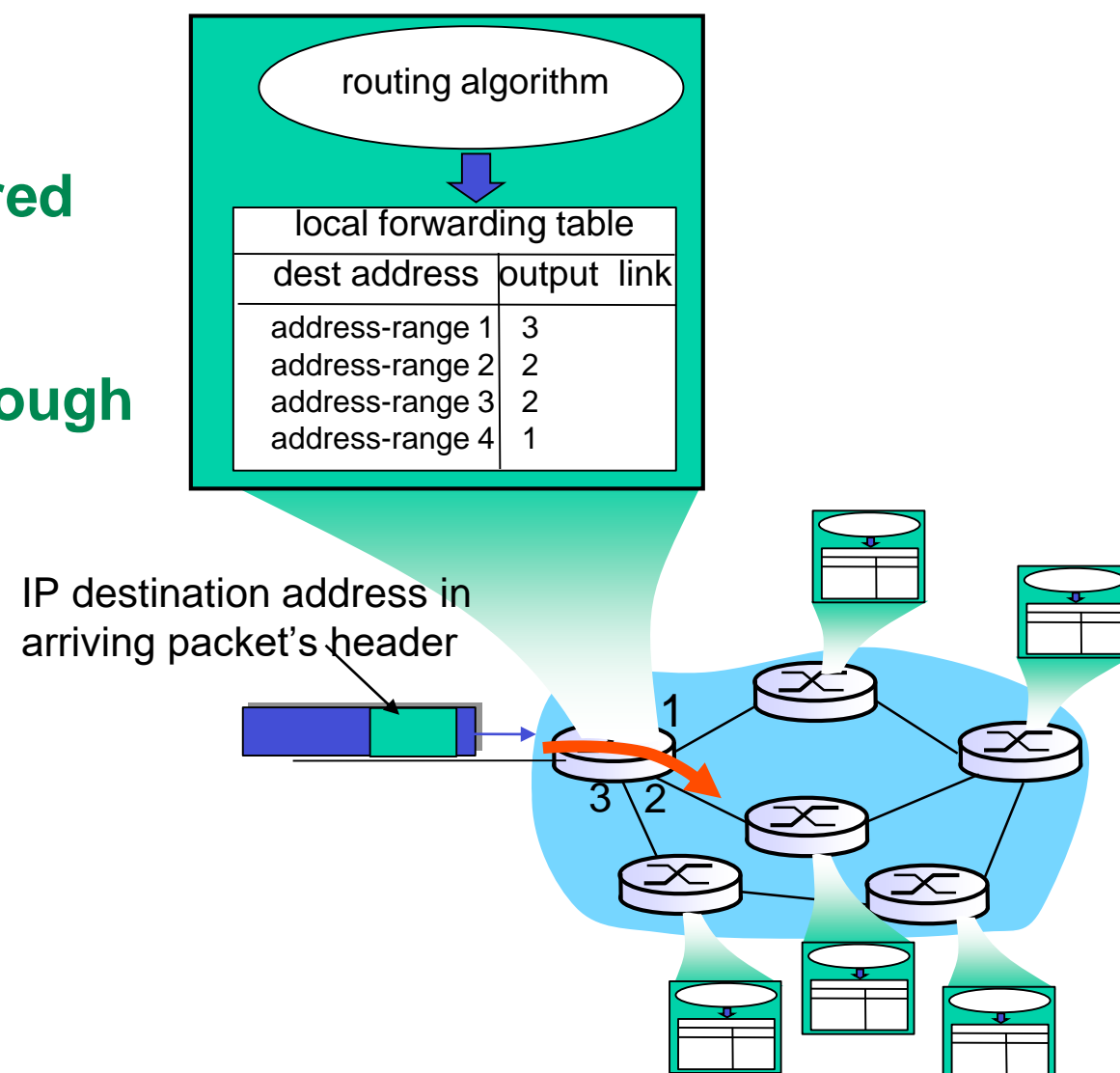
*control plane*

*Two approaches to structuring network control plane:*

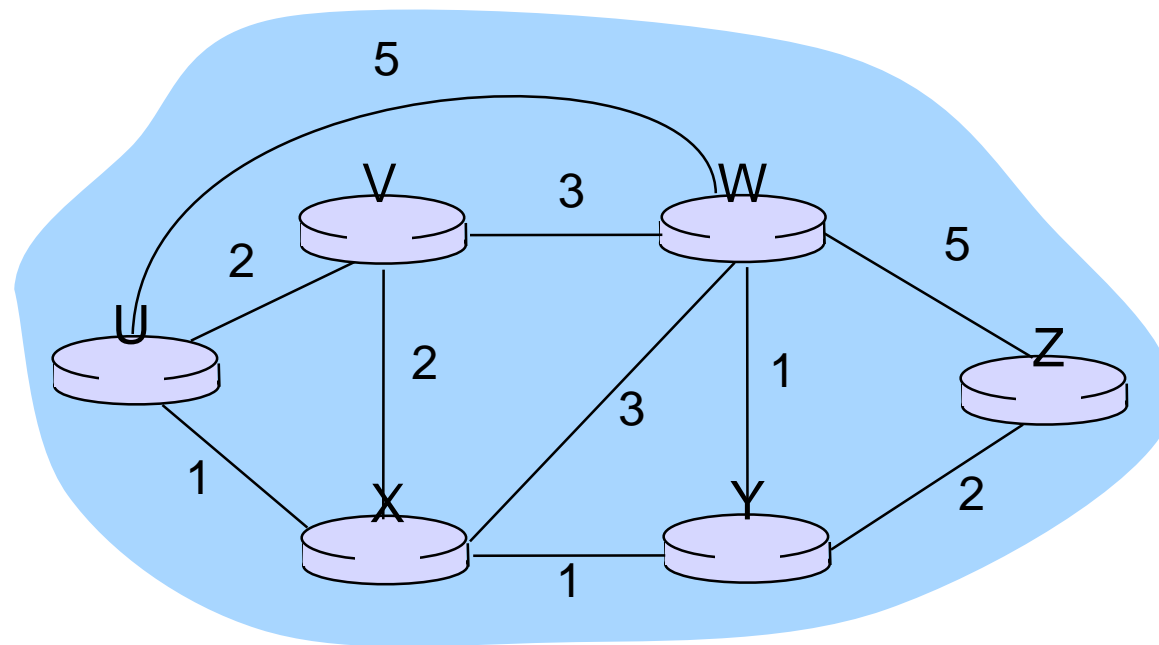
- per-router control (traditional)
- logically centralized control (software defined networking)

# Overview

- Routers use **routing algorithms** to maintain **forwarding tables**
- Routers exchange information with other routers to compute information that is entered into forwarding table
- Routing algorithms determine **best path** through a network from some source to some destination
  - **Least-cost path** is best path
    - Cost of path can depend on: distance, congestion, \$\$ cost, or other factors
  - Networks are represented as graphs
    - Routers are nodes in the graph
    - Links between routers are edges in graph



# Graph abstraction of the network



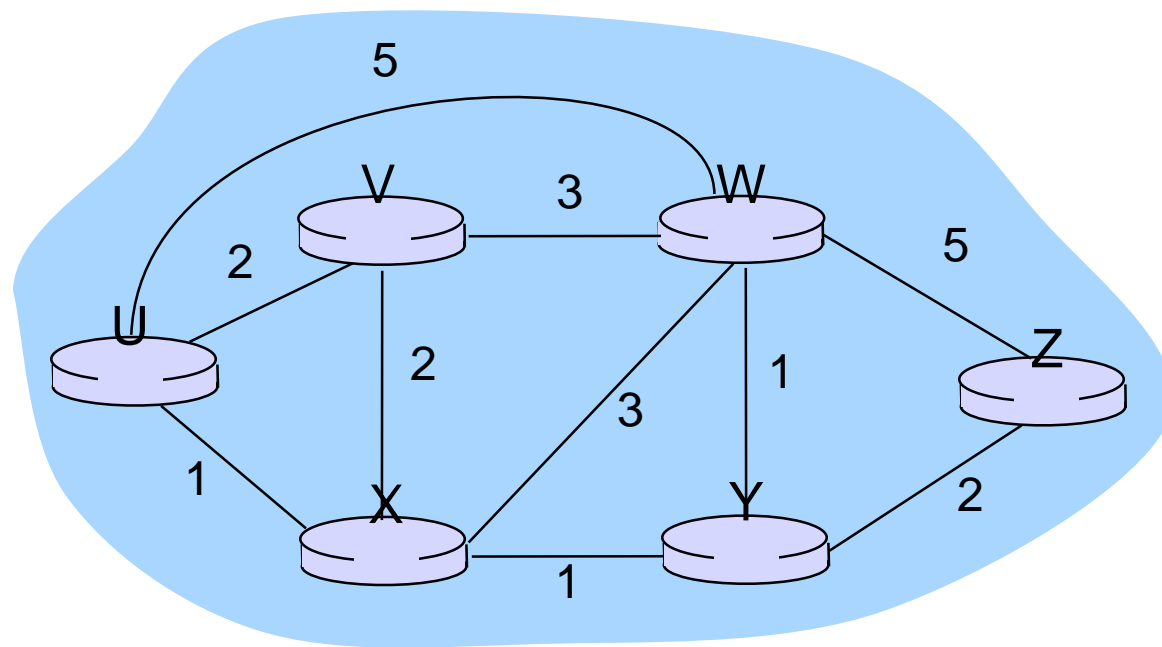
graph:  $G = (N, E)$

$N$  = set of routers =  $\{ u, v, w, x, y, z \}$

$E$  = set of links =  $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

*aside:* graph abstraction is useful in other network contexts, e.g., P2P, where  $N$  is set of peers and  $E$  is set of TCP connections

# Graph abstraction: costs



$c(x, x') = \text{cost of link } (x, x')$   
e.g.,  $c(w, z) = 5$

cost could always be 1, or  
inversely related to bandwidth,  
or inversely related to  
congestion

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

**key question:** what is the least-cost path between u and z ?  
**routing algorithm:** algorithm that finds that least cost path

# Types of Routing Algorithms

---

- **Global Routing Algorithm**

- All routers know complete network topology, and link cost info
- Often called **Link-state algorithms** since the algorithm must know the state (i.e. cost) of all links in the network

- **Decentralized Routing Algorithm**

- No router has complete information about the network or links costs
- Routers know information about physically-connected neighbors, such as link costs to neighbors
  - **Neighbors also exchange information about each others' neighbors**
- Calculation of least-cost path is determined iteratively in a distributed fashion
- An example is the **Distance-vector algorithm**

# Types of Routing Algorithms

---

- **Static Routing Algorithm**

- Routes change slowly over time
- Static routes are often entered by a human into a forwarding table

- **Dynamic Routing Algorithm**

- Routes are not dependent on human input
- Routes are periodically recomputed based on changes in the network
  - A new router is added
  - A router crashes
  - Too much congestion in part of the network
  - etc.
- Must be intelligent to avoid creating routing loops in network



# Overview of Network Layer

---

- **Virtual Circuit and Datagram Networks**
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
  - Overview
  - **Link state**
  - Distance vector
- **Routing in the Internet**
- **Broadcast and multicast routing**

# Link-State Routing Algorithm

---

- **All routers know complete network topology, and link cost info**
  - Periodically, **routers broadcast link-state information** about each of their links to all other routers on the network
  - Network topology and link costs known to all routers
    - **All nodes have same information about the network**
  - Each router computes least-cost path from itself (the source) to *ALL* other routers on the network (the destinations)
    - **Typically done using Dijkstra's algorithm**
    - **Result provides forwarding table for that node**

# Dijkstra's Algorithm

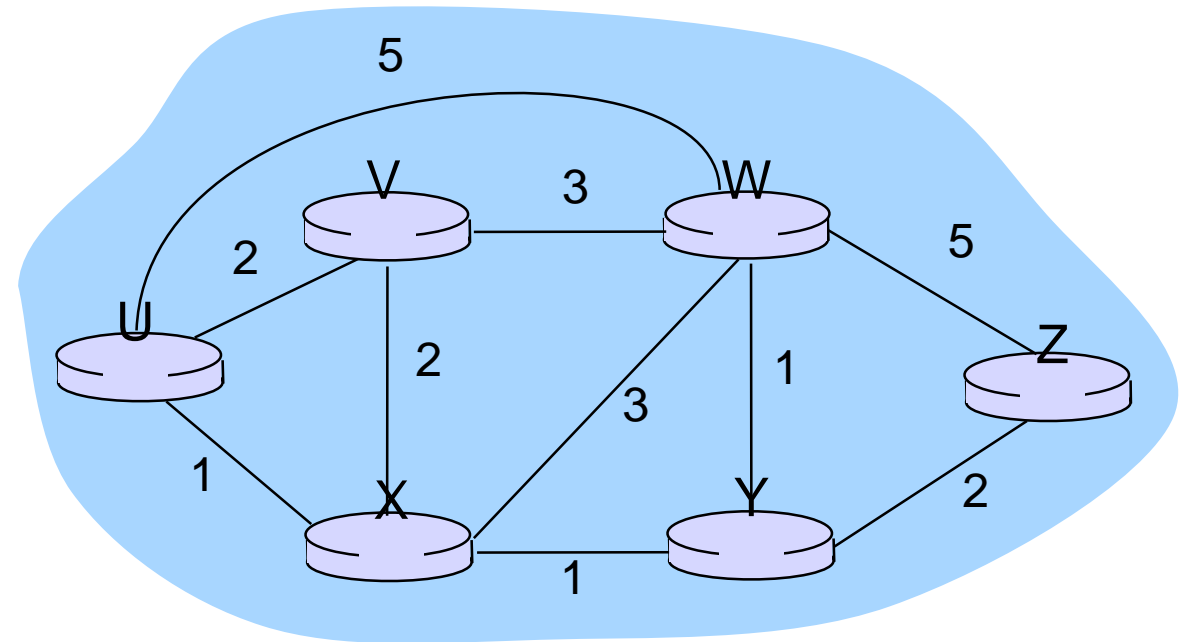
---

- **Algorithm works as follows:**

- Starts by assigning some initial distance value for each node in the graph
  - Distance from node  $s$  to itself is 0
  - Distances from node  $s$  to all other nodes in graph are initialized to INFINITY
- Operates in steps, where at each step the algorithm improves the distance values for nodes in the graph
- At each step the shortest distance from node  $s$  to another node in the graph is determined

# Dijkstra's Algorithm Example

- **Network is represented as a graph**
  - Routers are nodes in graph
  - Links are edges in graph
    - Cost of edge is labeled
- **Each router computes distance to all other routers in network**
  - Example shows computation done by router U



# A link-state routing algorithm

---

## **Dijkstra's algorithm**

- **net topology, link costs 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**
  - gives *forwarding table* for that node
- **iterative: after k iterations, know least cost path to k dest.’s**

## **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

---

1 **Initialization:**

2  $N' = \{u\}$

3 for all nodes  $v$

4 if  $v$  adjacent to  $u$

5 then  $D(v) = c(u, v)$

6 else  $D(v) = \infty$

7

8 **Loop**

9 find  $w$  not in  $N'$  such that  $D(w)$  is a minimum

10 add  $w$  to  $N'$

11 update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$  :

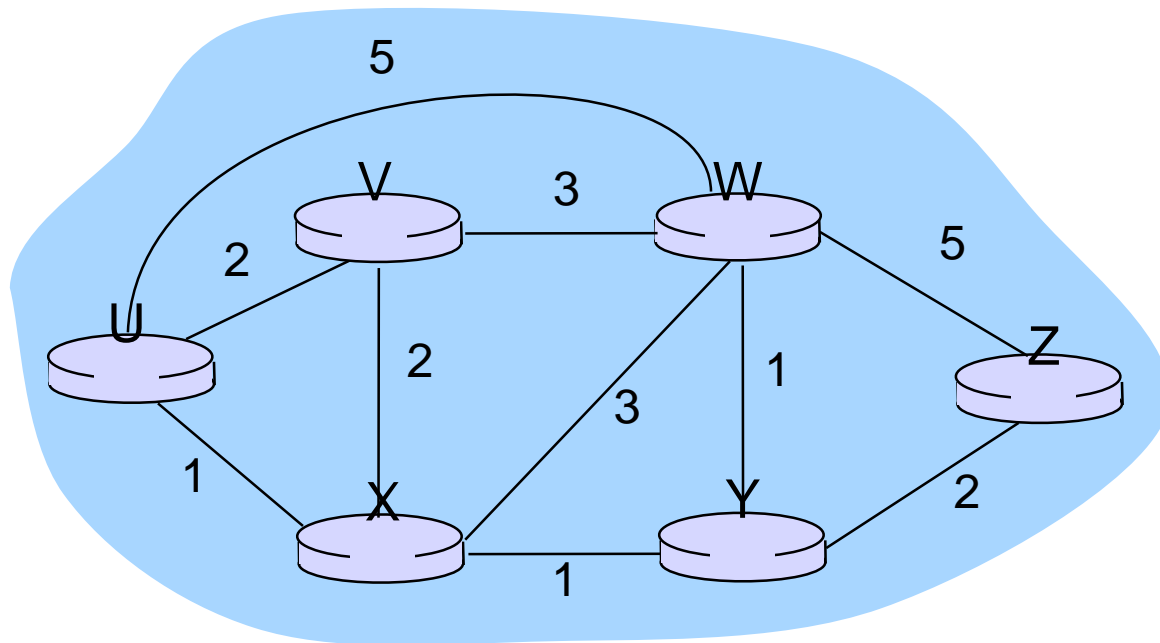
12  **$D(v) = \min( D(v), D(w) + c(w, v) )$**

13 /\* new cost to  $v$  is either old cost to  $v$  or known

14 shortest path cost to  $w$  plus cost from  $w$  to  $v$  \*/

15 **until all nodes in  $N'$**

# Dijkstra's Algorithm Example (Node U)

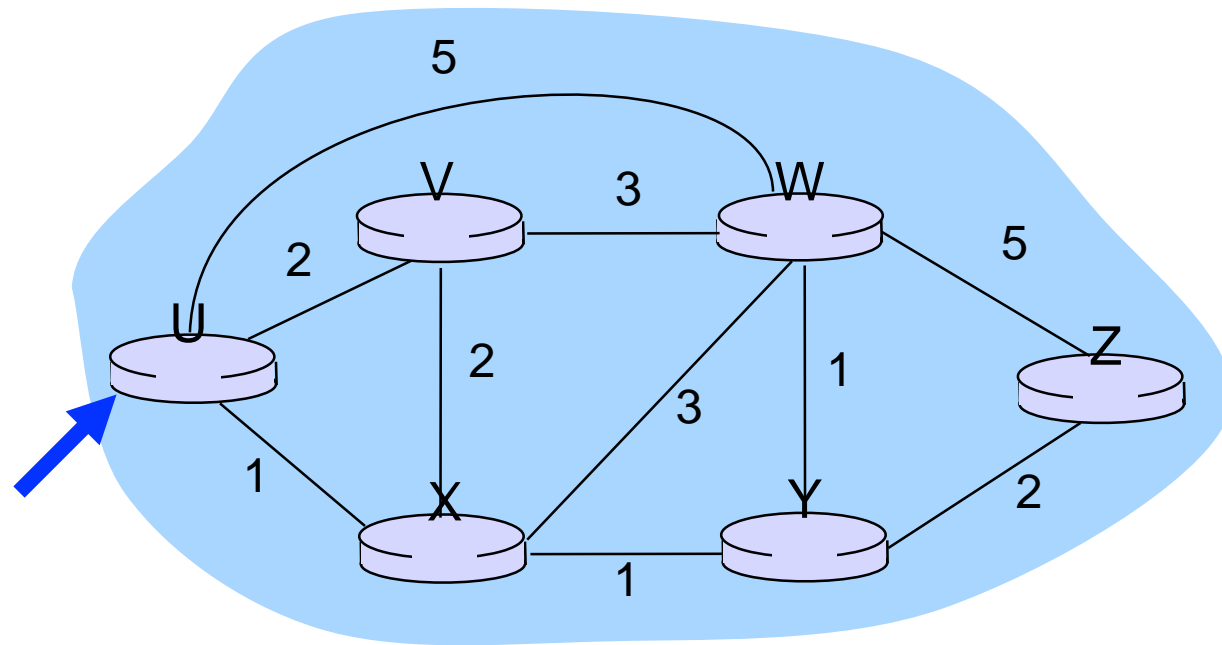


Initialize distances to U:

- Distance to itself is 0
- Distance to all other nodes is  $\infty$

Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

# Dijkstra's Algorithm Example (Node U)



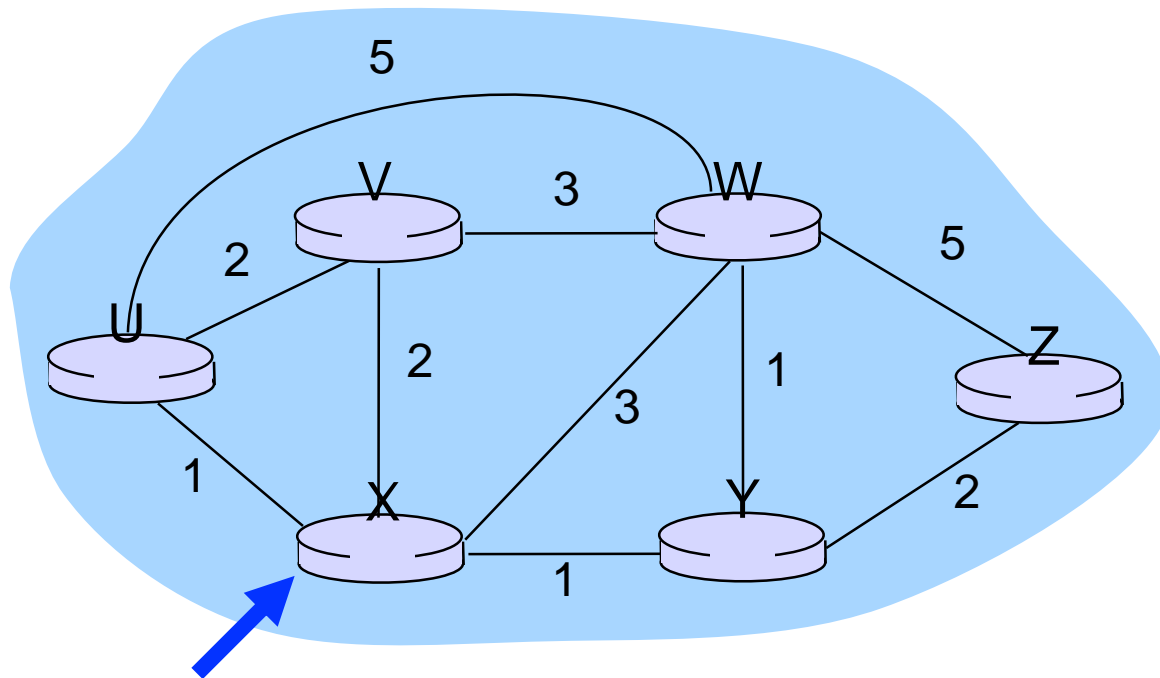
Select node with shortest distance to U (currently U) and determine shortest distance of its neighbors from U

- If node is unreachable it is still  $\infty$
- Record the path

Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

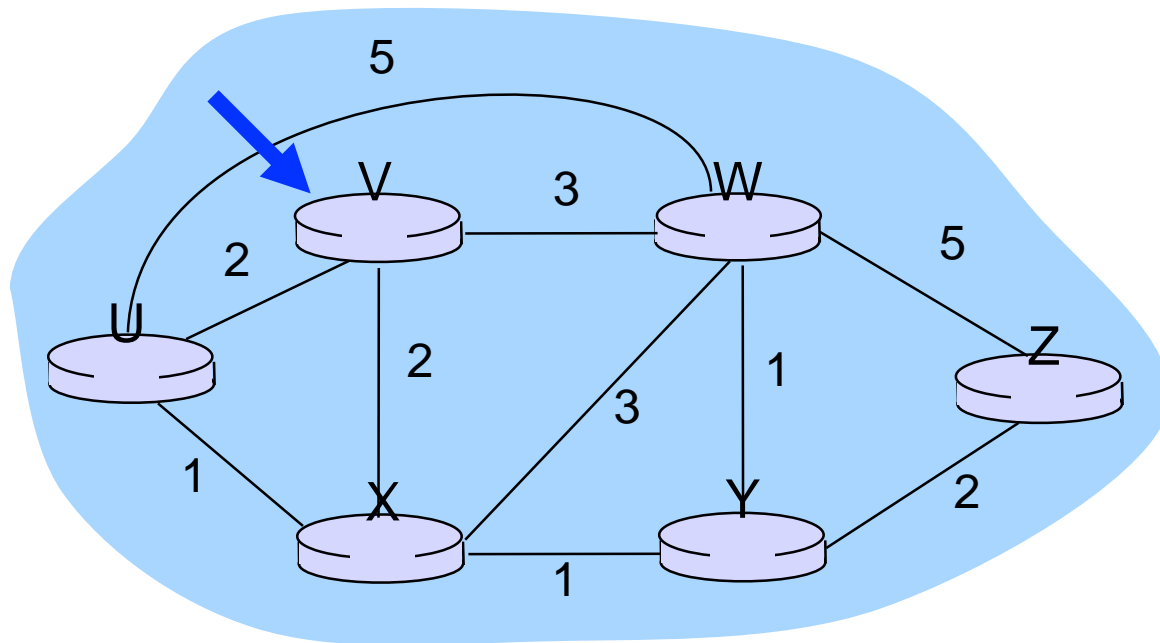


# Dijkstra's Algorithm Example (Node U)



Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

# Dijkstra's Algorithm Example (Node U)

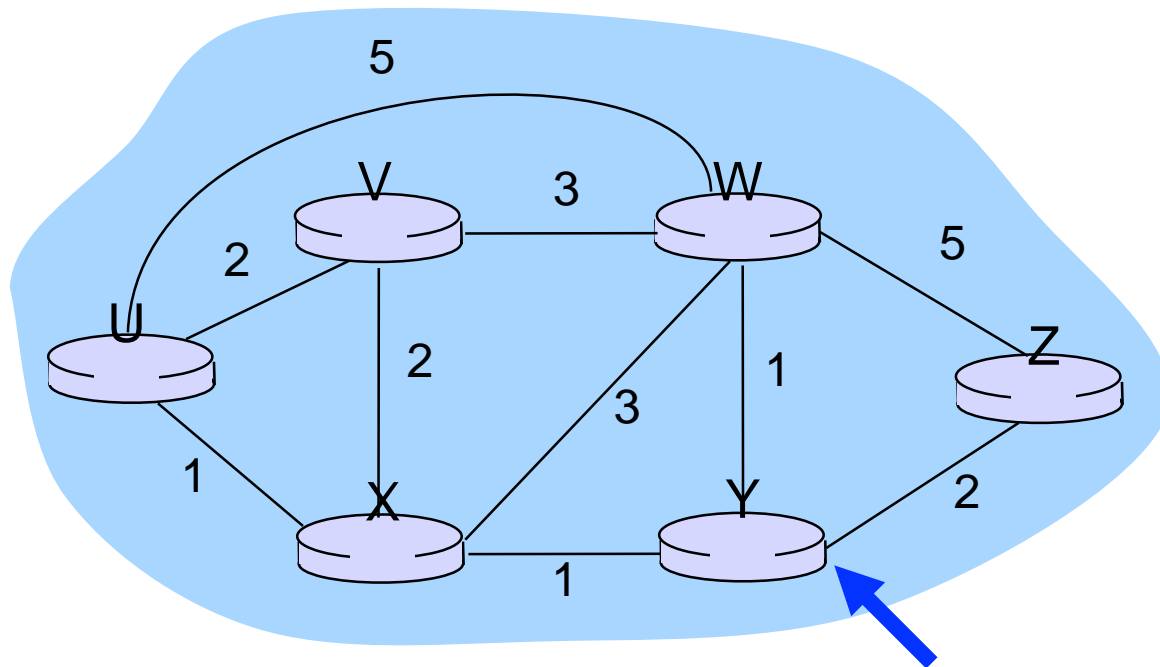


Select node with shortest distance to U (currently X) and determine shortest distance of its neighbors from U

- Node selected is min distance

Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

# Dijkstra's Algorithm Example (Node U)

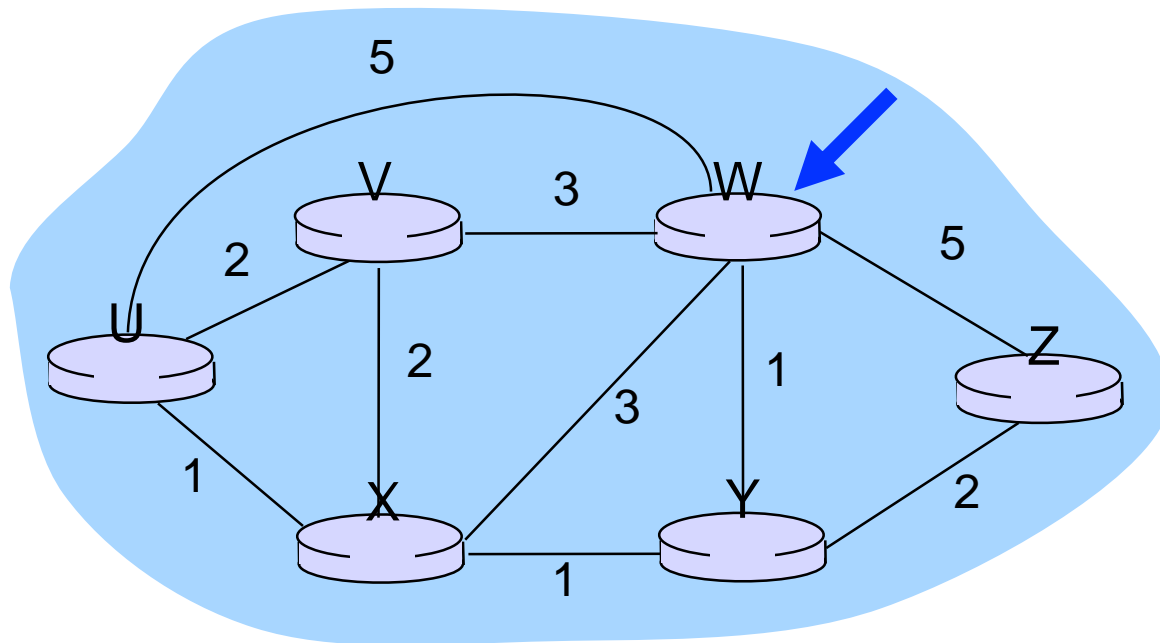


Select node with shortest distance to U (currently Y) and determine shortest distance of its neighbors from U

- Node selected is min distance

Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

# Dijkstra's Algorithm Example (Node U)

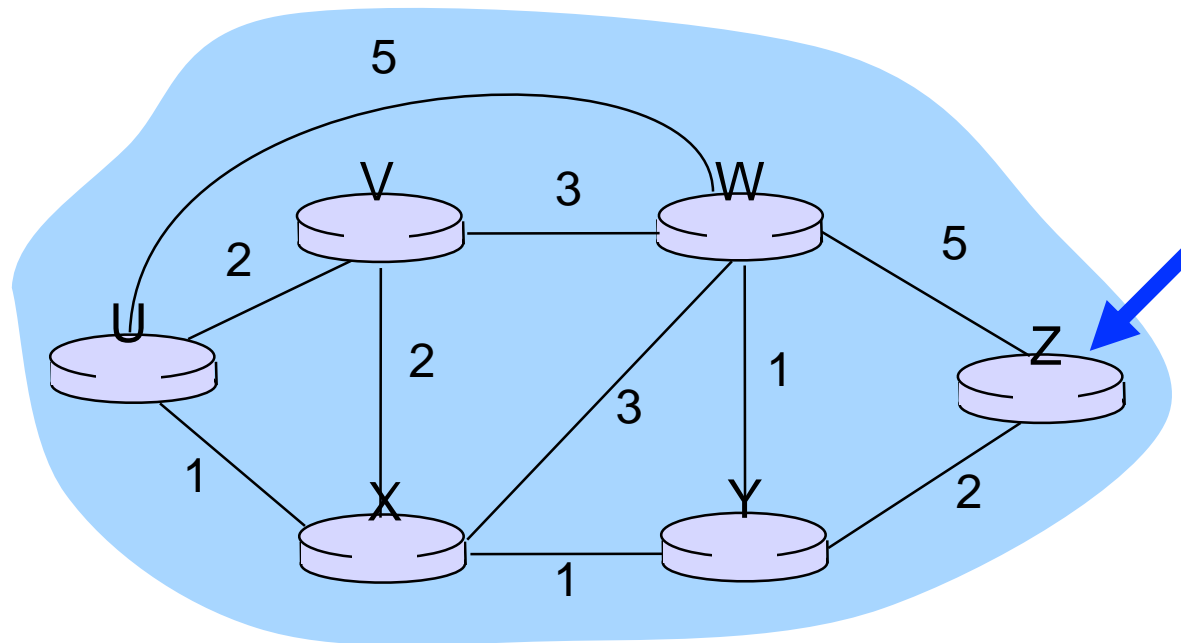


Select node with shortest distance to U (currently W) and determine shortest distance of its neighbors from U

- Node selected is min distance

Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

# Dijkstra's Algorithm Example (Node U)



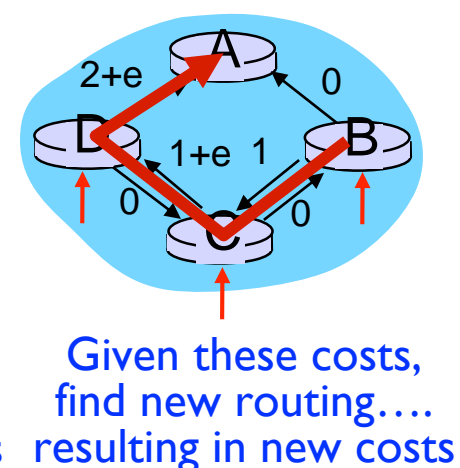
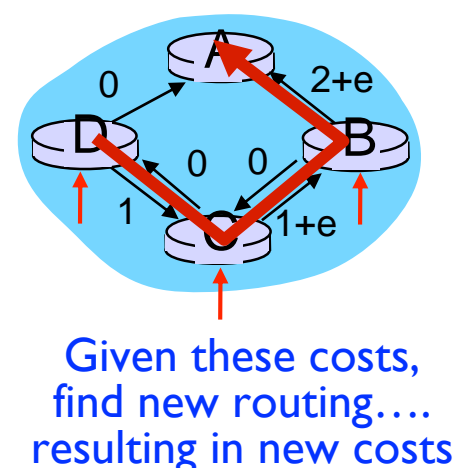
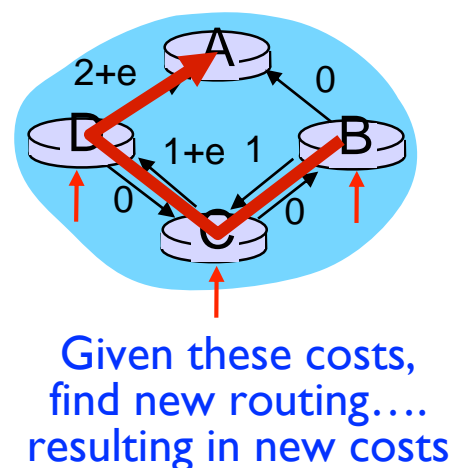
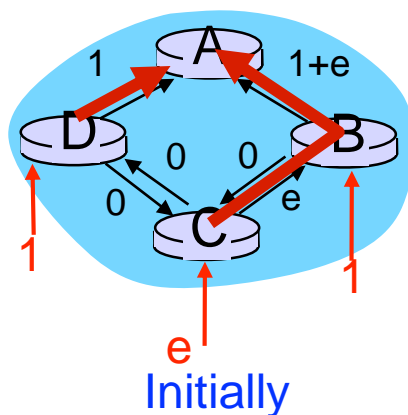
Select node with shortest distance to U (currently Z) and determine shortest distance of its neighbors from U

- All nodes have been accounted for, so terminate

Node	Distance Computation from U						Path
U							
V							
W							
X							
Y							
Z							

# Dijkstra's Algorithm Considerations

- When using Dijkstra's algorithm dynamically, **oscillations are possible**
  - For example, if link cost equals amount of carried traffic
    - Links with no traffic have low cost ..
    - So all traffic is rerouted down those links
    - Then a different set of links will have lower cost



# Overview of Network Layer

---

- **Virtual Circuit and Datagram Networks**
- **Router Architectures**
- **IP: Internet Protocol**
- **Routing algorithms**
  - Overview
  - Link state
  - Distance vector
- **Routing in the Internet**
- **Broadcast and multicast routing**

# Distance Vector Algorithm

---

- No router has complete information about the network or links costs
- Routers know information about physically-connected neighbors, such as link costs to neighbors
- Key idea
  - From time-to-time, each node sends its own distance vector estimate to its neighbors
  - When  $x$  receives new DV estimate from neighbor, it updates its own DV using the **Bellman-Ford algorithm**

Let

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

then

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

$\uparrow$   $\uparrow$   $\uparrow$   
Cost from neighbor  $v$  to destination  $y$   
Cost to neighbor  $v$   
 $\min$  taken over all neighbors  $v$  of  $x$



# Distance vector algorithm

---

## ***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:

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

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

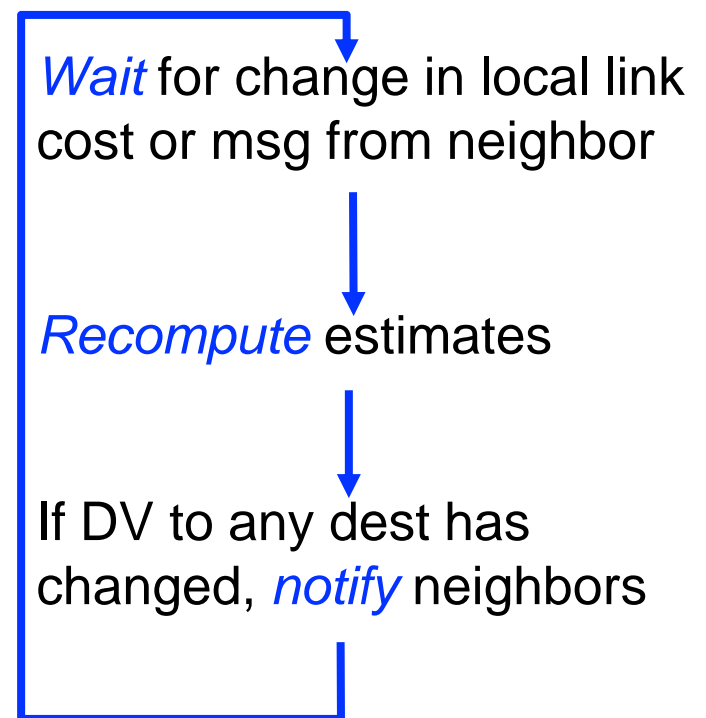
# Distance Vector Algorithm

---

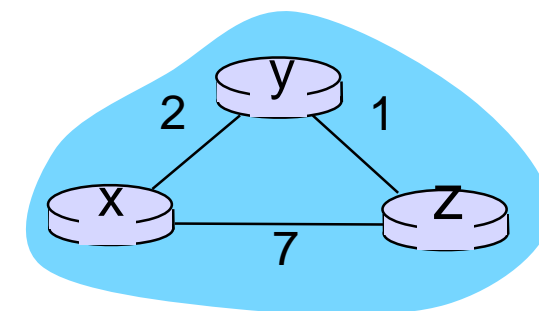
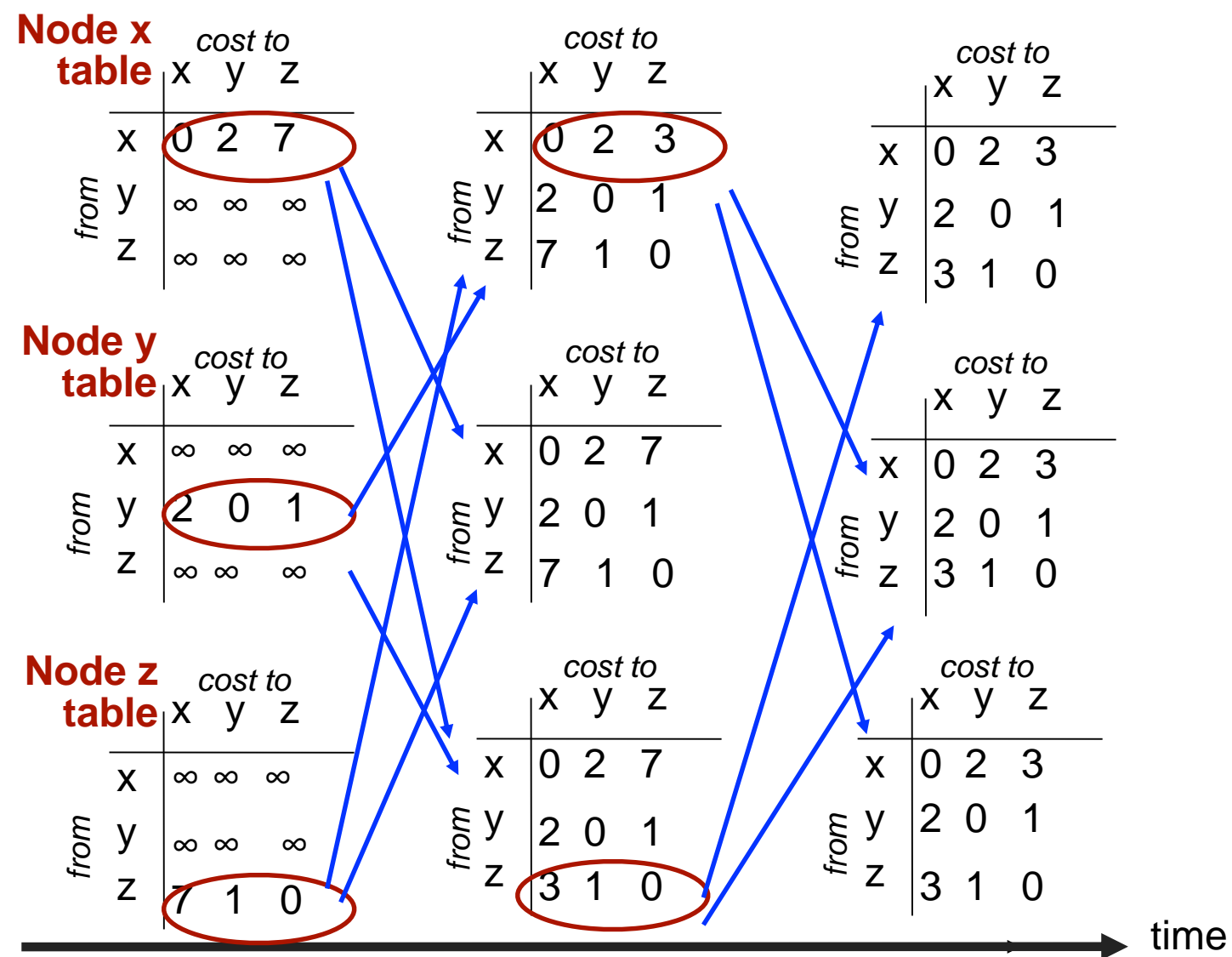
- **Algorithm is:**

- Iterative and 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

## At Each Node:



# Distance Vector Example



# Distance Vector Considerations

---

- **When link costs change**

- Good news travels quickly

- When link cost decreases, notifying neighbors is quick

- Bad news travels slowly

- When link cost increases, can cause **routing loops** that last for long periods of time (see text section 5.5.2)

# Comparison of LS and DV algorithms

## **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: what 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

- **Routing protocols within a single network (an autonomous system) are determined by some network administrator**
  - Provide routing information for hosts *within the network* (i.e. Intra-AS routing)
  - For example, a company or ISP would use some routing protocol for their network
- **Routers within an AS also need a way to route packets to hosts on other networks**
  - Route packets to **gateway router** that will direct packets to the next network (AS)
    - All packets destined for another network are routed to the gateway router
  - What if there are multiple gateways?  
To which gateway should packets get routed?
    - Handled by **inter-AS routing protocols (BGP)**
    - Routers get information from both intra and inter-AS routing protocols and maintain info in forwarding table

