Chapter 4: Network Layer

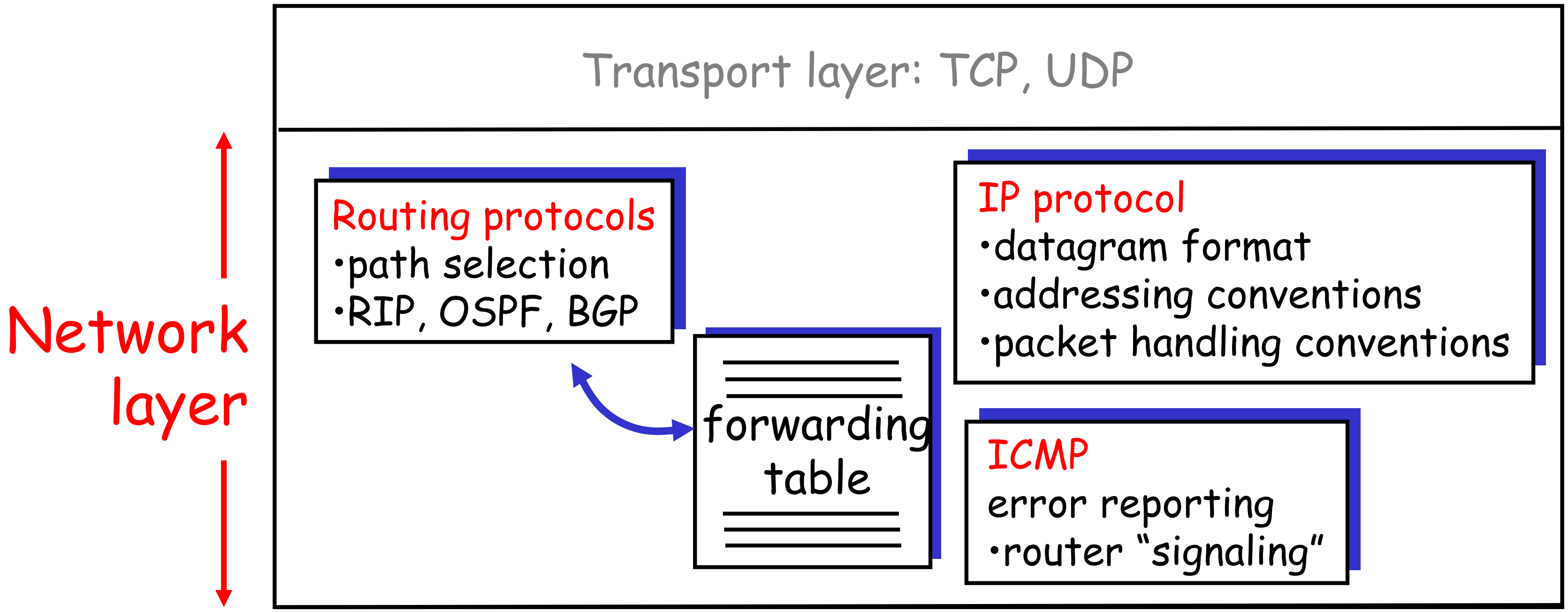
Routing Algorithm

Instructor: HOU, Fen

2025

The Internet Network layer

Three main components: IP protocol, Routing protocol, other supporting protocol.

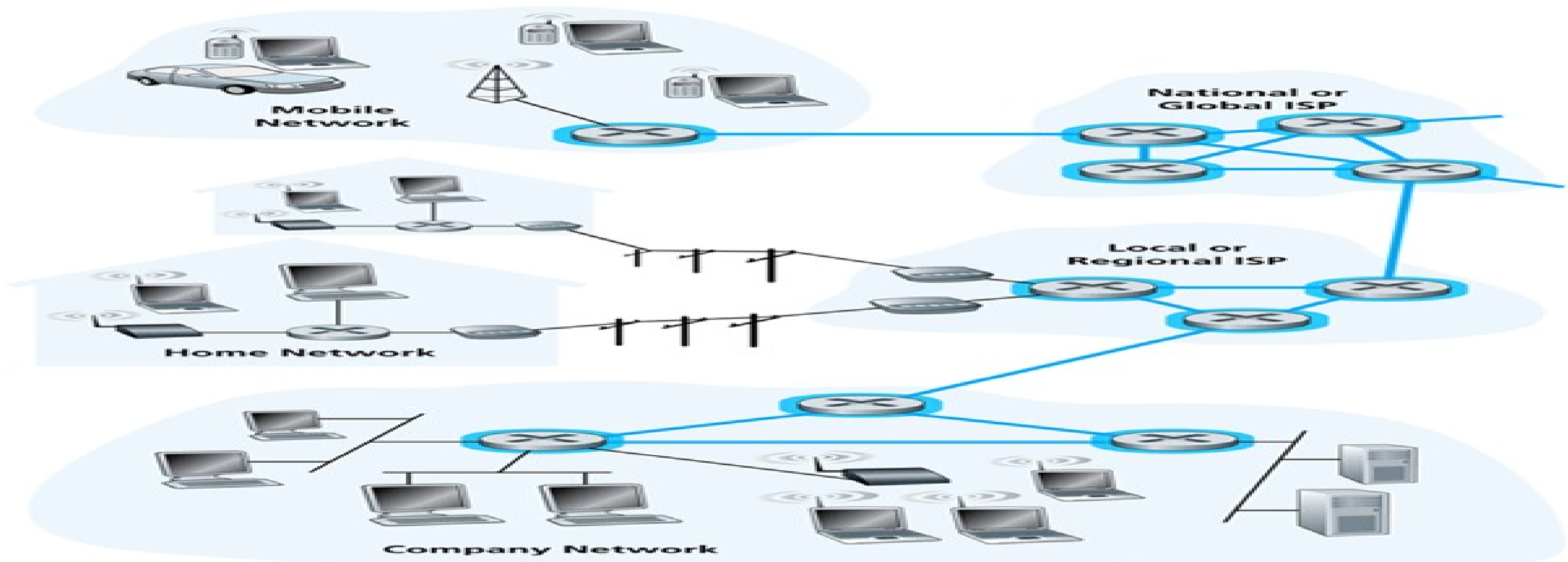


Forwarding: Moving a packet from a router's input link to the appropriate output link.

Routing: Determining the route between source and destinations

Router

- Default router for a host: the first-hop router for the host.
- Source router: the default router of the source host.
- Destination router: the default router of the destination host.

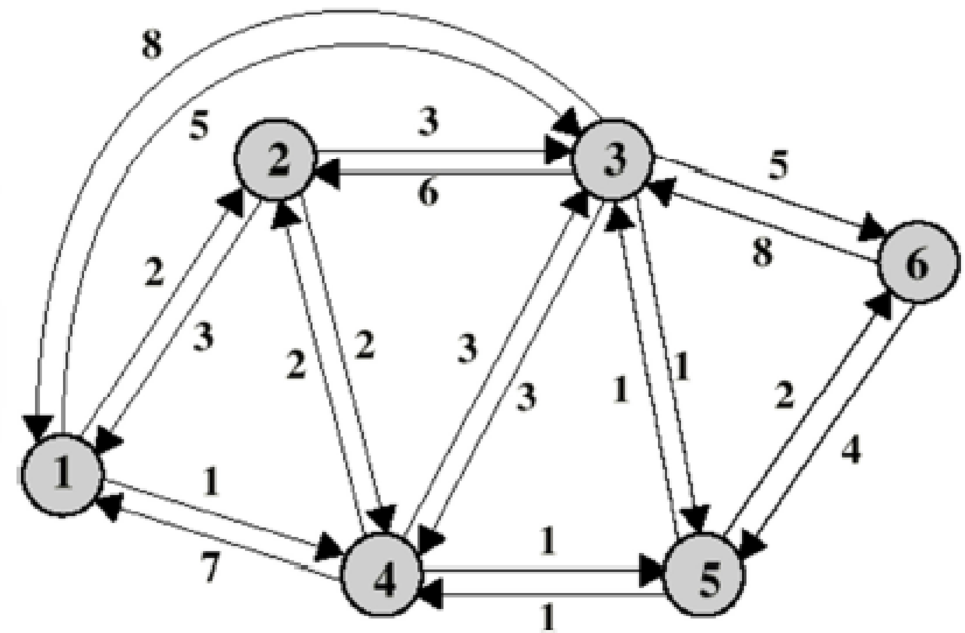


Routing

- Routing algorithm: Given a set of routers, with links connecting the routers, a routing algorithm is to find a 'good' path from source router to destination router.
- Complex, crucial aspect of packet switched networks
- Characteristics required
  - Correctness
  - Simplicity
  - Stability
  - Fairness
  - Optimality
  - Efficiency

Performance Criteria

- Minimum hops
- Least cost
- Short delay
- High throughput



Routing Strategies

- Fixed
- Flooding
- Random
- Adaptive

Fixed Routing

- Single permanent route for each source to destination pair
- Determine routes using the least cost algorithm
- Route fixed, at least until a change in network topology

Fixed Routing

- Central routing directory

From Node

To Node	1	2	3	4	5	6
1	—	1	5	2	4	5
2	2	—	5	2	4	5
3	4	3	—	5	3	5
4	4	4	5	—	4	5
5	4	4	5	5	—	5
6	4	4	5	5	6	—

- Routing table at each router

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

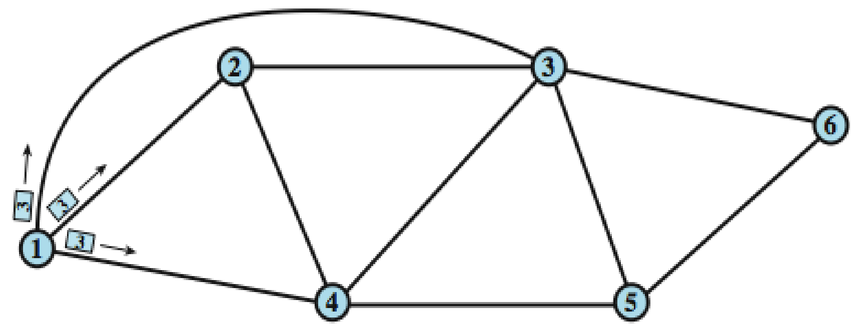
Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

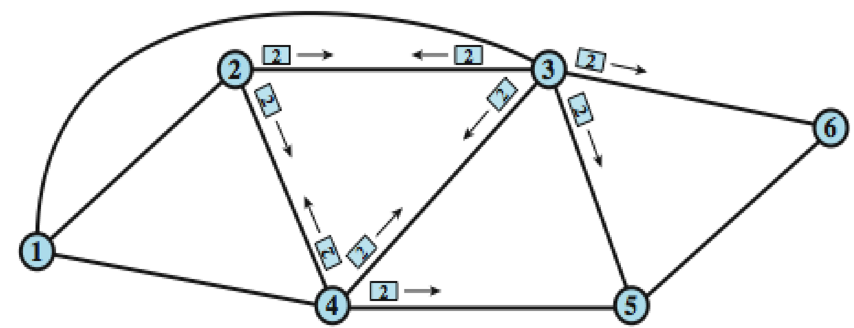
Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Flooding

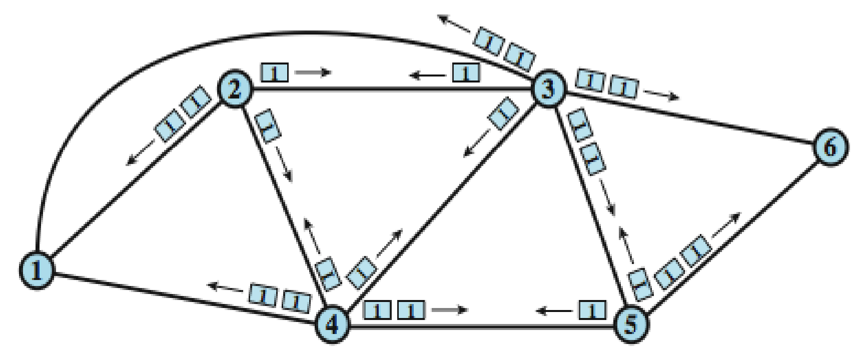
- No network information is required, packet sent by node to every neighbor.
- Incoming packets are retransmitted on every outgoing link except the incoming link



(a) First hop



(b) Second hop



(c) Third hop

Flooding

- Eventually a number of copies will arrive at destination
- Each packet is uniquely numbered so duplicates can be discarded
- Nodes can remember packets that are already forwarded to keep network load in bounds
- Can include a hop count in packets

Properties of Flooding

- All possible routes are tried
  - Very robust
- At least one packet will have taken minimum hop count route
  - Can be used to set up virtual circuit
- All nodes that are directly or indirectly connected to the source node are visited
  - Useful to distribute information

Random Routing

- Node selects one outgoing link to forward the incoming packet
- Selection can be random or round robin
- No network information is needed => simple
- Route is typically not least cost nor minimum hop

$$P_i = \frac{R_i}{\sum_j R_j}$$

where

P_i = probability of selecting link i

R_i = data rate on link i

Adaptive Routing

- Used by almost all packets switched networks
- Routing decision changes as the network condition changes
  - Failure: when a node or trunk fails, it can no longer be used as a part of a route
  - Congestion: when a portion of the network is congested, it is desirable to route packets around, rather than through, the congested area
- Requires information about network
- Decisions are more complex
- Tradeoff between quality and network information and overhead

Adaptive Routing Example

- From node 1 to node 6, based on queue lengths and the values of bias for outgoing link, the minimum value of Q+B is 4,on the link to node 3.

Node 4's Bias Table for Destination 6

Next Node	Bias
1	9
2	6
3	3
5	0

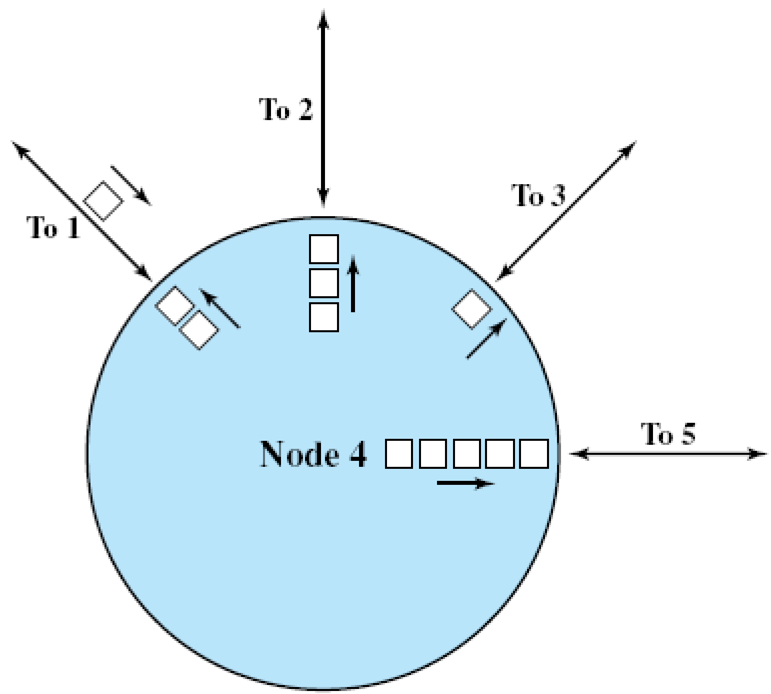
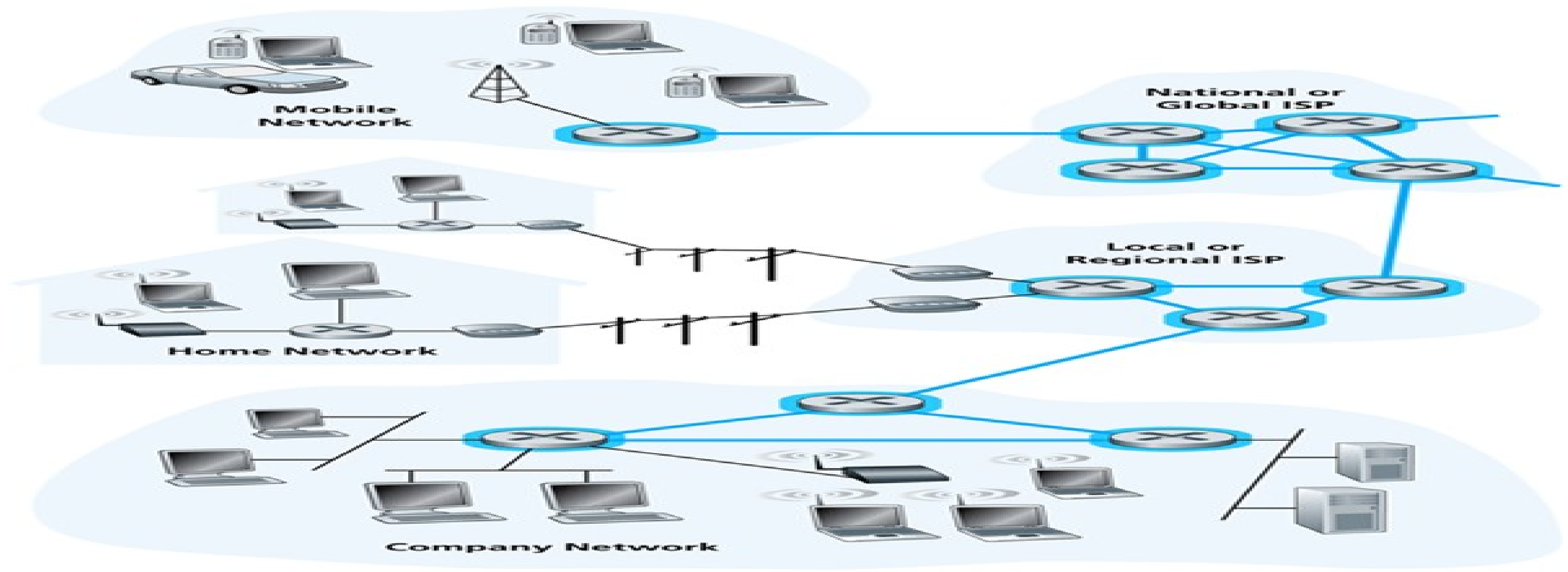


Figure 12.4 Example of Isolated Adaptive Routing

Routing Algorithms

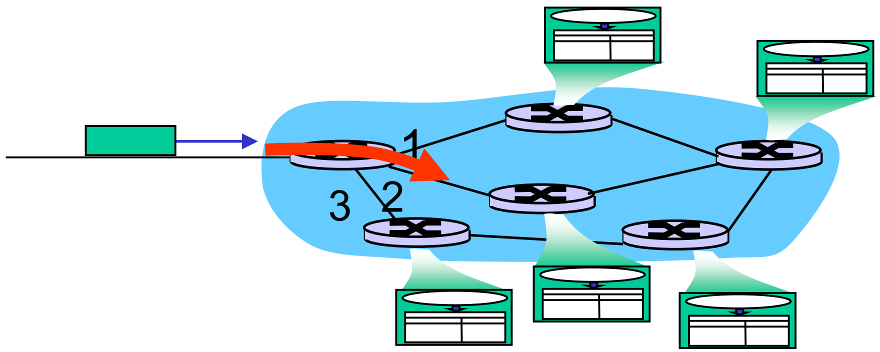
Routing algorithm

- Routing algorithm: Given a set of routers, with links connecting the routers, a routing algorithm is to find a 'good' path from source router to destination router.



Routing algorithm

- The purpose of a routing algorithm: given a set of routers, with links connecting the routers, a router algorithm is to find a "good" path from source router and destination router.
- Routing algorithm operates in network routers to exchange and compute the information that is used to configure these forwarding tables.



Routing Algorithm Classification

Global vs. decentralized

Global routing algorithms:

- all routers have complete information about the node connectivity (i.e., network topology), and link cost information. Global routing algorithms use these complete, global information to compute the least-cost path between a source router to a destination router.
- "link state" algorithm : a global routing algorithm is called as a "link state" algorithm since it must be aware of the states of all links in the network.

Decentralized routing algorithms:

- At the beginning, each router knows the cost of its directly connected links. Using the iterative calculation and information exchange with the neighboring nodes, a router gradually obtains the least-cost path to a destination router.

Routing Algorithm Classification

Static vs. dynamic

Static routing algorithm:

- routes change slowly over time

Dynamic routing algorithm:

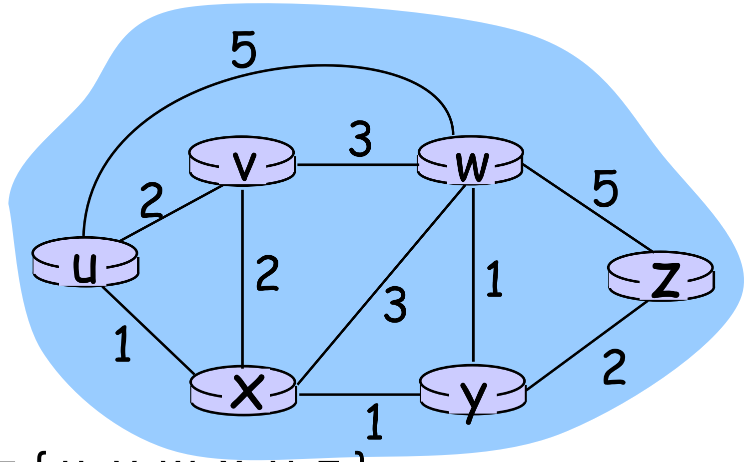
- routes change more quickly
  - periodic update in response to traffic load changes, topology changes, or link cost changes

Dijkstra's Algorithm

Dijkstra's algorithm:

- It is to compute the least cost paths from one source node (i.e., one vertex in the graph) to any other nodes.
- All nodes know the complete information about the network topology and cost of all links in the graph. These information can be obtained via "link state broadcast"

Graph abstraction



Graph: G = (N,E)

N = set of vertexes=set of nodes=set of routers = { u, v, w, x, y, z }

E = set of edges=set of physical links between these routers
={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Neighbor: a node y is said to be a neighbor of node x if (y,x) belongs to E.

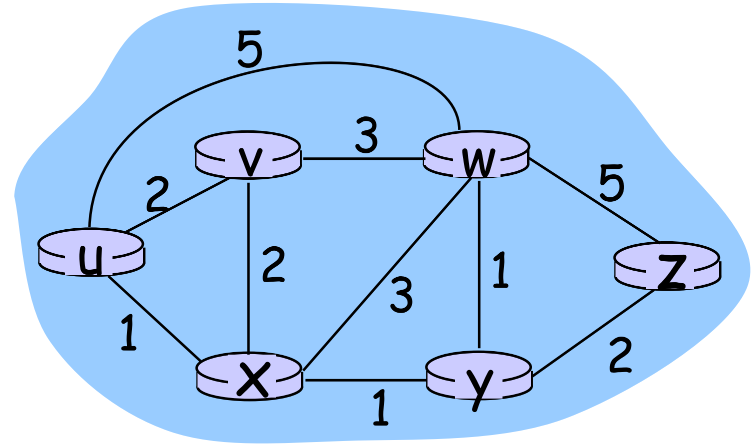
Cost: cost of the edge (x, y) is represented by c(x, y), The cost is related to several factors such as distance, bandwidth, congestion level, etc.

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

- if (x, y) does not belong to E (i.e., there doesn't exist link between router x and router y), we set the cost c(x, y)=∞.

- We consider the undirected graphs (i.e., edges do not have a direction). Therefore c(x,y)=c(y,x)

Graph abstraction



Path: a path in a graph G=(N, E) is s sequence of nodes $(x_1,x_2,\ldots,x_p)$ such that each of the consecutive pairs $(x_1,x_2),(x_2,x_3),\ldots(x_{p-1},x_p)$ are edges in E.

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

Least-cost path: a path with the least cost.
Least-cost path problem: find the least-cost path between the source router and destination Router.

Shortest path: The path with the smallest number of links.
Shortest path problem: find the shortest path between the source router and destination router.

If all edges in the graph have the same cost, the least-cost problem is also the shortest path problem.

Dijkstra's Algorithm

Notation:

- $c(x,y)$: link cost from node x to y; Note that if x and y are not direct neighbors, $c(x,y) = \infty$.
- $D(v)$: current value of the cost of the path from the source node to the node v
- T: The set of nodes whose least cost path are definitively known.

Dijsktra's Algorithm

```
Initialization:
  T= {u}
  for all nodes v
    if v is neighbor to u
       then D(v) = c(u,v)
    else D(v) = ∞
 Loop
   find a node w without the set T and with the minimum cost D(w).
  add w to T
  update D(v) for all nodes v which are neighboring nodes of w,
   but not in the set T :

       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 */
   /* for nodes that are not adjacent to w, we don't need to
   do any update*/
 until |T|=N   /*all nodes are in T*/
```

- Iterative process: after k iterations, a source node knows the least cost path to k nodes.

Dijkstra's algorithm

1 iteration: Initialization

Source nodes: u (start with source router), T={u}

For cost of other nodes:

For adjacent nodes of u: D(x)= cost of the link (u,x) where x is the neighbor of node u.

For all other nodes: the distance is = ∞

2 to N iterations:

Step 1: Get next node

Compare all nodes without the set T to find the node with the minimum cost D, then add this node into the set T.

Step 2: Update least-cost

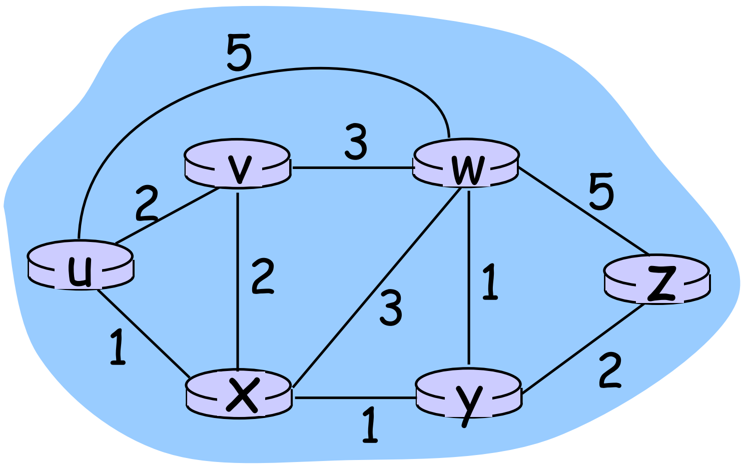
Update the cost for the nodes without the set T:

For notes adjacent to the newly added node (denoted as x) using $D(v) = \min \{ D(v), D(x) + c(x,v) \}$

For all other nodes without the set T: Keep D(z) be the same as that in the previous iteration.

T is the set of nodes whose least cost path are known

Dijkstra's algorithm



Example: Find the least cost path from the node u to any other nodes.

Dijkstra's algorithm: the 1st iteration

The 1st iteration: Initialization process

Source nodes: u (start with source router), T={u}

For cost of other nodes:

For adjacent nodes of u: D(v)=2, D(x)=1, D(w)=5

For all other nodes: the distance is = ∞

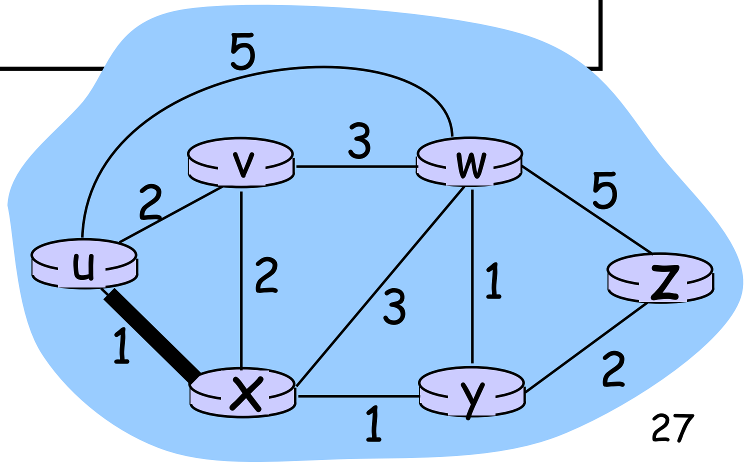
The 2nd iteration:

Compare all nodes without the set T to find the node with the minimum cost D, that is x, then add node x into the set T.

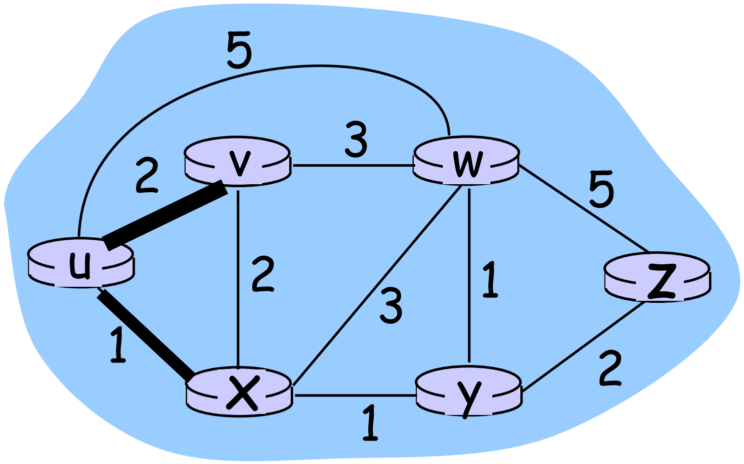
Update the cost for the nodes without the set T

For adjacent nodes of the newly added node x: D(v)=2, D(w)=4, D(y)=2

For all other nodes without the set T: D(z) = ∞



Dijkstra's algorithm



The 3rd iteration:

T={u,x}, the distances of nodes: D(v)=2, D(w)=4, D(y)=2, D(z) = ∞.

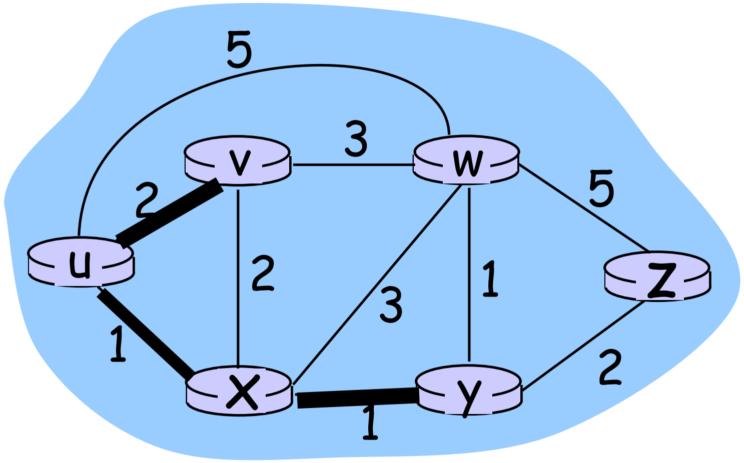
Compare all nodes without the set T to find the node with the minimum cost D, then add node v into the set T. T={u,x,v}

Update the cost for the nodes without the set T, we have

For adjacent nodes of the newly added node v : D(w)=4,

For all other nodes without the set T: D(y)=2, D(z)=∞.

Dijkstra's algorithm



The 4th iteration:

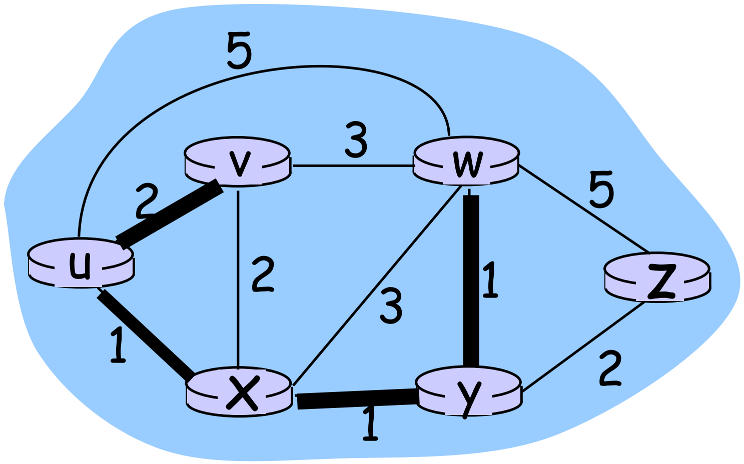
T={u,x,v}, the distances of nodes without T: D(w)=4, D(y)=2, D(z)=∞

Compare all nodes without the set T to find the node with the minimum cost D, then add node y into the set T. T={u,x,v,y}

Update the cost for the nodes without the set T, we have

For adjacent nodes of the newly added node y: D(w)=3, D(z)=4

Dijkstra's algorithm:



The 5th iteration:

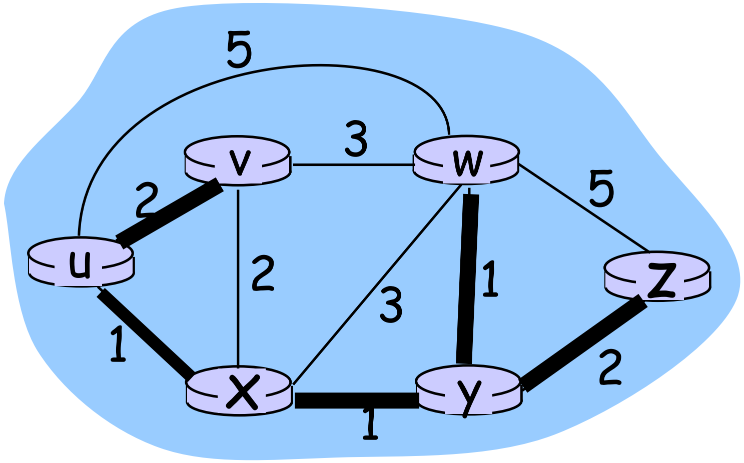
T={u,x,v,y}, the distances of nodes without T: D(w)=3, D(z)=4

Compare all nodes without the set T to find the node with the minimum cost D, then add node w into the set T. T={u,x,v,y,w}

Update the cost for the nodes without the set T, we have

For adjacent nodes of the newly added node w: D(z)=4

Dijkstra's algorithm

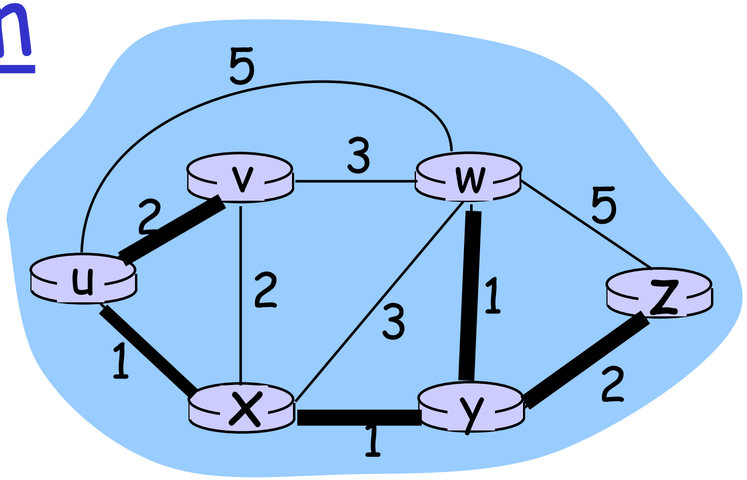


The 6th iteration:
T={u,x,v,y,w}, the distances of nodes without T: D(z)=4

Compare all nodes without the set T to find the node with the minimum cost D, then add node z into the set T. T={u,x,v,y,w,z}

Finally, we find the least cost path from the node u to any other nodes, which is is called the "shortest-path tree", or "sink tree," for node u.

Dijkstra's algorithm

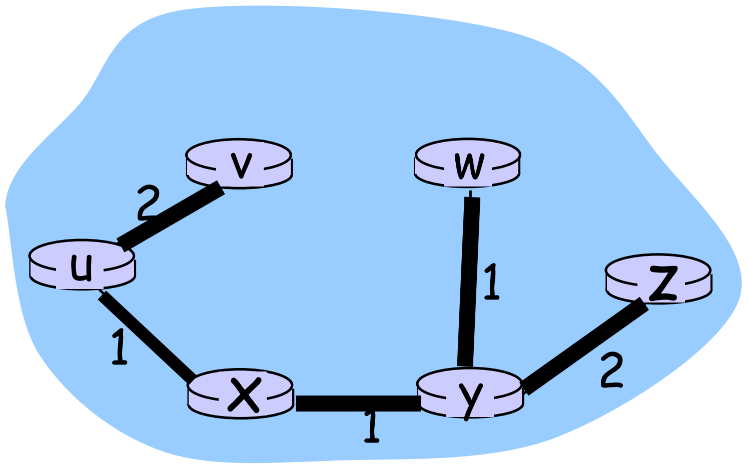


Iteration	T	D(X)	Path	D(V)	Path	D(Y)	Path	D(W)	Path	D(Z)	Path
1	{u}	1	u-x	2	u-v	∞	-	5	u-w	∞	-
2	{u,x}	1	u-x	2	u-v	2	u-x-y	4	u-x-w	∞	-
3	{u,x,v}	1	u-x	2	u-v	2	u-x-y	4	u-x-w	∞	-
4	{u,x,v,y}	1	u-x	2	u-v	2	u-x-y	3	u-x-y-w	4	u-x-y-z
5	{u,x,v,y,w}	1	u-x	2	u-v	2	u-x-y	3	u-x-y-w	4	u-x-y-z
6	{u,x,v,y,w,z}	1	u-x	2	u-v	2	u-x-y	3	u-x-y-w	4	u-x-y-z

T is the set of nodes whose least cost path are known;
D(x) is the least cost for the source node to the node x

Dijkstra's algorithm

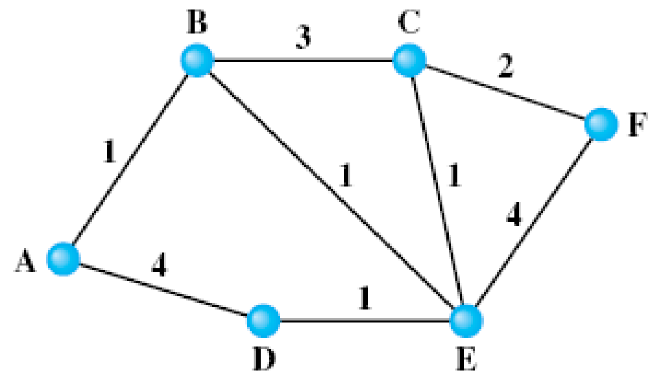
Resulting shortest-path tree from u:



Resulting forwarding table in u:

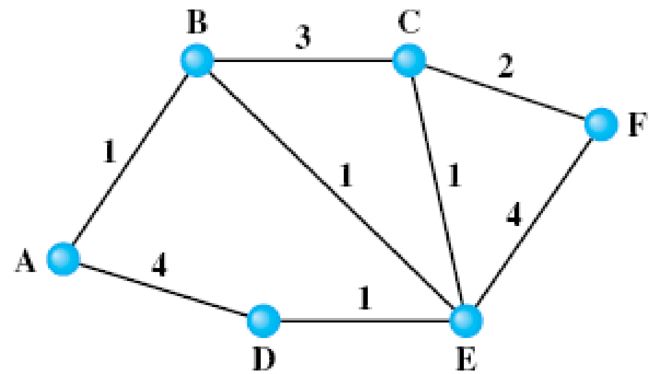
destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Exercise for Dijkstra's Algorithm



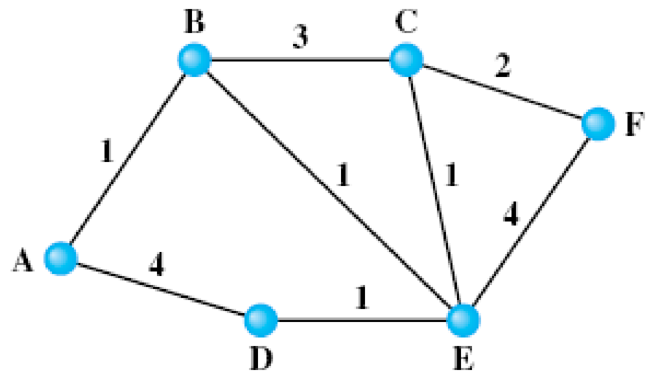
Iteration T	D(B) Path	D(D) Path	D(C) Path	D(E) Path	D(F) Path
1 {A}					
2					
3					
4					
5					
6					

Exercise for Dijkstra's Algorithm



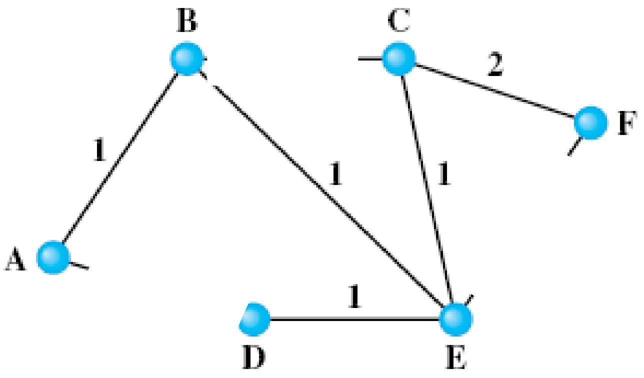
Iteration	T	D(B)	Path	D(D)	Path	D(C)	Path	D(E)	Path	D(F)	Path
1	{A}	1	A-B	4	A-D	∞	-	∞	-	∞	-
2	{A,B}	1	A-B	4	A-D	4	A-B-C	2	A-B-E	∞	-
3	{A,B,E}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	6	A-B-E-F
4	{A,B,E,D}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	6	A-B-E-F
5	{A,B,E,D,C}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	5	A-B-E-C-F
6	{A,B,E,D,C,F}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	5	A-B-E-C-F

Exercise for Dijkstra's Algorithm



Iteration	T	D(B)	Path	D(D)	Path	D(C)	Path	D(E)	Path	D(F)	Path
1	{A}	1	A-B	4	A-D	∞	-	∞	-	∞	-
2	{A,B}	1	A-B	4	A-D	4	A-B-C	2	A-B-E	∞	-
3	{A,B,E}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	6	A-B-E-F
4	{A,B,E,D}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	6	A-B-E-F
5	{A,B,E,D,C}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	5	A-B-E-C-F
6	{A,B,E,D,C,F}	1	A-B	3	A-B-E-D	3	A-B-E-C	2	A-B-E	5	A-B-E-C-F

Resulting shortest-path tree from A to all other nodes



Resulting forwarding table at node A:

destination	link
B	(A,B)
C	(A,B)
D	(A,B)
E	(A,B)
F	(A,B)