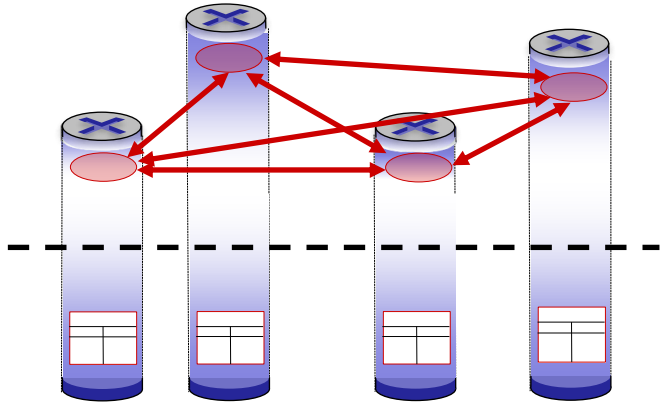


Computer Networks II

Routing Algorithms

Link State Routing

Amitangshu Pal

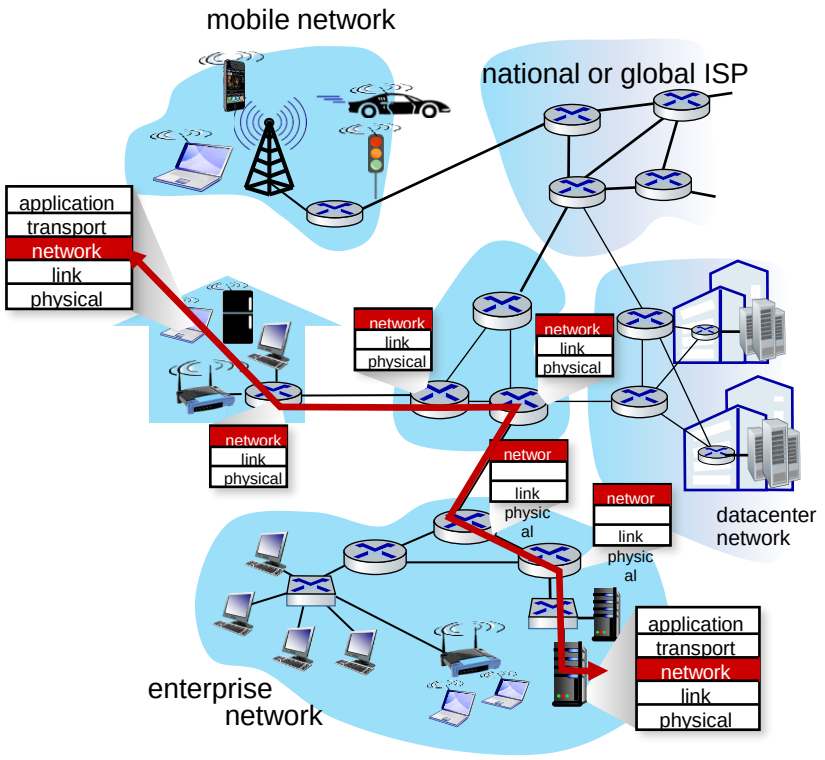
Computer Science and Engineering

IIT Kanpur

Routing Protocols

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- Path: Sequence of routers packets traverse from given initial source host to final destination host
- "Good": least "cost", "fastest", "least congested", "shortest"



Routing Algorithm Classification

Global (centralized): All routers have complete topology, link cost info

- "Link state" algorithms

How fast do routes change?

Static: routes change slowly over time

Dynamic: routes change more quickly

- periodic updates or in response to link cost changes

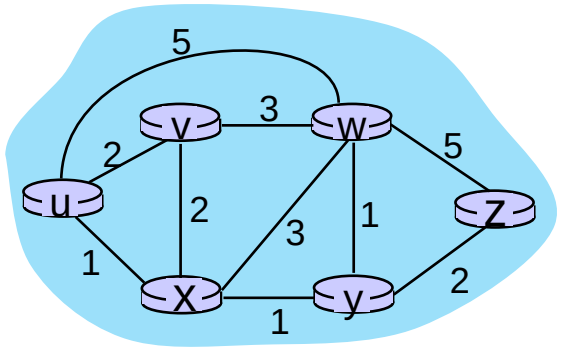
Decentralized: iterative process of computation, exchange of info with neighbors

- Routers initially only know link costs to attached neighbors
- "Distance vector" algorithms

Global or decentralized information?

Link State Routing Protocol

Graph Abstraction: Link Costs



$c_{a,b}$: cost of direct link connecting a and b

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

cost defined by network operator: could always be 1, or inversely related to bandwidth, or related to congestion

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) }

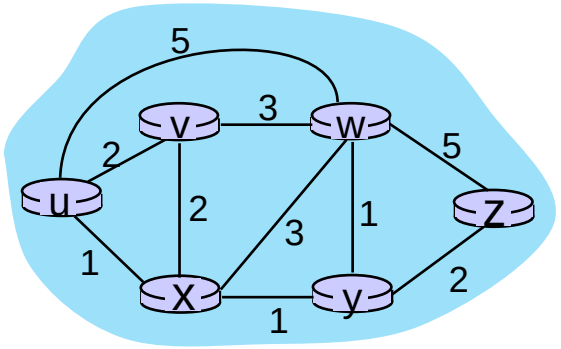
Link-state Routing Algorithm

- Runs in two steps

- Step1: Learning the network topology
  - Nodes flood the information about their neighbors and their link costs in Link State Packets
  - All nodes learn the entire network topology

- Step2: Finding the shortest path
  - Each node runs Dijkstra's algorithm to find out the shortest path (or least cost path) corresponding to each destination

Learning the Network Topology



LS packet for u

Neighbor	Cost
v	2
x	1
w	5

LS packet for v

Neighbor	Cost
u	2
x	2
w	3

LS packet for w

Neighbor	Cost
u	5
v	3
x	3
y	1
z	5

LS packet for x

Neighbor	Cost
u	1
v	2
w	3
y	1

LS packet for y

Neighbor	Cost
w	1
x	1
z	2

LS packet for z

Neighbor	Cost
w	5
y	2

Dijkstra's Link-state Routing Algorithm

- Centralized: network topology, link costs known to all nodes

- 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 destinations

Notation

- $c_{x,y}$: direct link cost from node x to y; = ∞ if not direct neighbors
- D(v): current estimate of cost of least-cost-path from source to destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least-cost-path definitively known

Dijkstra's Link-state Routing Algorithm

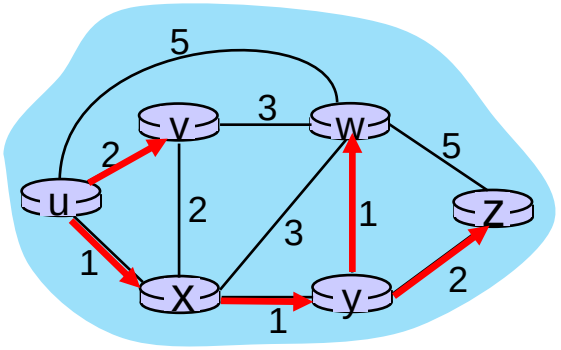
```
1  Initialization:
2    N' = {u}                    /* compute least cost path from u to all other nodes */
3    for all nodes a
4      if a adjacent to u        /* u initially knows direct-path-cost only to  direct neighbors */
5          then D(a) = c_u,a     /* but may not be minimum cost */
6      else D(a) = ∞

7    Loop
8      find a not in N' such that D(a) is minimum
9      add a to N'
10     update D(b) for all b adjacent to a and not in N' :
11         D(b) = min ( D(b),  D(a) + c_a,b )
12     /* new least-path-cost to b is either old least-cost-path to b or known least-cost-path to a plus
          direct-cost from a to b */

13   until all nodes in N'
```

Notation

- $c_{x,y}$: direct link cost from node x to y; = ∞ if not direct neighbors
- D(v): current estimate of cost of least-cost-path from source to destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least-cost-path definitively known

Dijkstra's Algorithm: An 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					



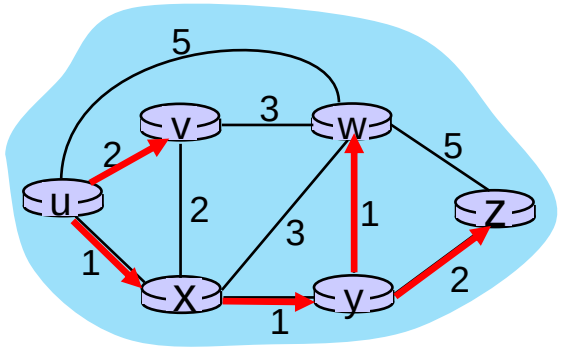
Initialization (step 0): For all **a**: if **a** adjacent to **u**, then D(**a**) = $c_{u,a}$

find **a** not in N' such that D(**a**) is a minimum
add **a** to N'
update D(**b**) for all **b** adjacent to **a** and not in N' :
D(b) = min (D(b), D(a) + $c_{a,b}$)

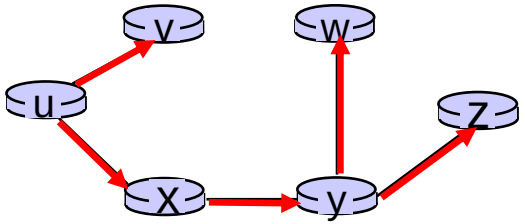
Notation

- $c_{x,y}$: direct link cost from node x to y; = ∞ if not direct neighbors
- D(v): current estimate of cost of least-cost-path from source to destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least-cost-path definitively known

Dijkstra's Algorithm: An Example



Resulting least-cost-path tree from u:



Resulting forwarding table in u:

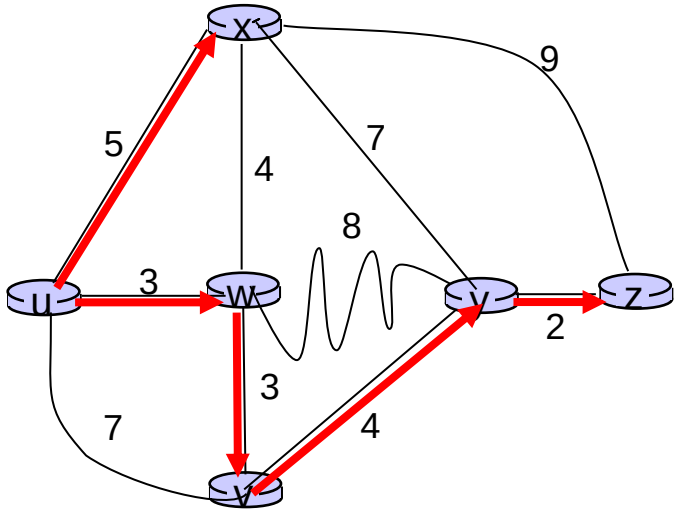
Destination	Outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

route from u to v directly

route from u to all other destinations via x

Dijkstra's Algorithm: Another Example

Step	N'	v D(v), p(v)	w D(w), p(w)	x D(x), p(x)	y D(y), p(y)	z D(z), p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Dijkstra's Algorithm: Discussion

Algorithm complexity: n nodes

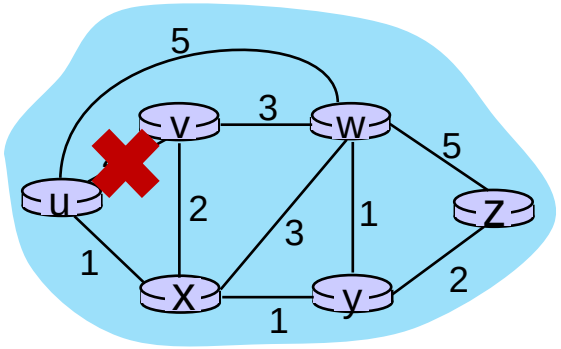
- Each of n iteration: need to check all nodes, a, not in N'
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- More efficient implementations possible: O(nlogn)
- Memory requirement: If each nodes has k neighbors → memory required to store the link states is O(nk)

Message complexity:

- Each router must broadcast its link state information to other n routers
- Efficient broadcast algorithms: O(n) link crossings to disseminate a broadcast message from one source
- Each router's message crosses O(n) links: overall message complexity: $O(n^2)$

This means that the message travels through each link in the network only once in the best-case scenario, which is O(n).

Handling Failures



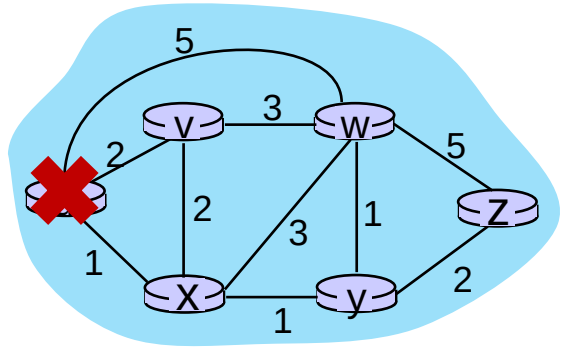
Link failure

LS packet for u

Neighbor	Cost
v	∞
x	1
w	5

LS packet for v

Neighbor	Cost
u	∞
x	2
w	3



Node failure

LS packet for x

Neighbor	Cost
u	∞
v	2
w	3
y	1

LS packet for v

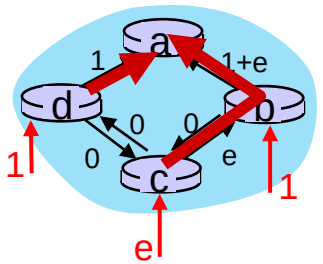
Neighbor	Cost
u	∞
x	2
w	3

LS packet for w

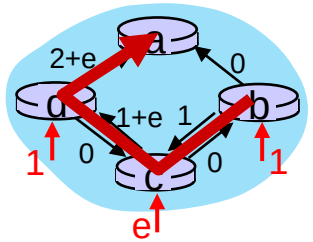
Neighbor	Cost
u	∞
v	3
x	3
y	1
z	5

Dijkstra's Algorithm: Oscillations

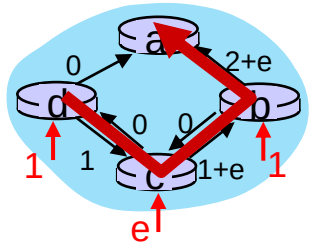
- When link costs depend on traffic volume, route oscillations possible
- Sample scenario:
  - Routing to destination a, traffic entering at b, c, d with rates 1, e (<1), 1
  - Link costs are directional, and volume-dependent



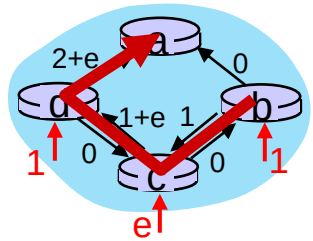
Initially



Given these costs, find new routing…. resulting in new costs



Given these costs, find new routing…. resulting in new costs



Given these costs, find new routing…. resulting in new costs

Summary

- Routing algorithms:
  - Centralized vs decentralized
  - Static vs dynamic

- Link state routing algorithm:
  - Limitations: may lead to oscillations