Routing

Network layer: data plane, control plane

Data plane (Forwarding)

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

Control plane (Routing)

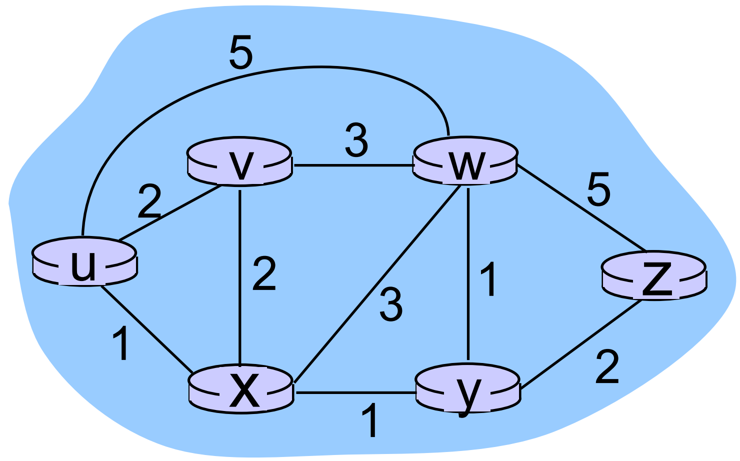
- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

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 will traverse in going from given initial source host to given final destination host**
- **"good": least "cost", "fastest", "least congested"**

Graph abstraction of the network



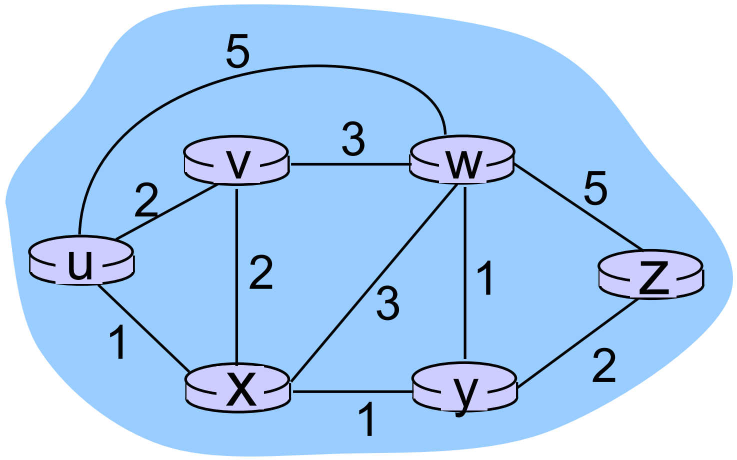
graph: G = (N,E)

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



$c(x,x')$ = 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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?

routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

global:

- **all routers have complete topology, link cost info**
- **"link state" algorithms**

decentralized:

- **router knows physically-connected neighbors, link costs to neighbors**
- **iterative process of computation, exchange of info with neighbors**
- **"distance vector" algorithms**

Q: static or dynamic?

static:

- **routes change slowly over time**

dynamic:

- **routes change more quickly**
  - periodic update
  - in response to link cost changes

Analytical Background on Route Optimization

- **Network model:**
  - Network is a graph of nodes (routers) and links.
  - Each link k has a capacity: $c(k)$.
  - Traffic matrix: $M(i,j)$ = rate of traffic from node i to node j.
  - Proportion of traffic from node i to node j traversing link k: $P(i,j,k)$. If only one path from i to j, then $P(i,j,k)$ is either 1 or 0, depending on whether link k is on the path.
- **Classical algorithmic problem.**

In practice: Distributed, Dynamic Routing Protocols

- **Distributed because in a dynamic network, no single, centralized node “knows” the whole “state” of the network.**
- **Dynamic because routing must respond to “state” changes in the network for efficiency.**
- **Two types of protocols: Link State and Distance Vector.**
  - Link State uses Dijkstra’s shortest path algorithm, but makes up a distributed version.
  - Distance Vector uses a distributed version of Bellman Ford algorithm.

Routing uses standard shortest path algorithms

- **Derived from classic algorithms (e.g., Prims)**
  - Djistra's shortest path
  - Bellman-Ford
- **The key idea is to relax the distance as more information is provided**

Link State versus Distance Vector

- **Link state**
  - Send information to everyone in the network
  - Each will compute the shortest path

- **Distance vector**
  - Send information to neighbors only
  - Each neighbor determines shortest path based on one-hop information

Link State Protocol

- **Each node "floods" the network with link state packets (LSP) describing the cost of its own (outgoing) links.**

- **Each node maintains a LSP database of all LSPs it received.**
  - Only the recent most LSP is maintained for a link.
  - The LSP database describes this node's view of the "state" of the network.
  - It is expected that all nodes "see" the same state (but of course not guaranteed)

Flooding Mechanism

- **Flooding is a basic routing service. Used by many protocols in some form.**
- **The originator generates LSPs periodically, or when some link costs changes significantly.**
  - The originator transmits LSP on all its interfaces.

A link-state routing algorithm

Dijkstra's algorithm

- **net topology, link costs known to all nodes**
  - accomplished via "link state broadcast"
- **computes least cost paths from one node ('source") to all other nodes**
  - gives *forwarding table* for that node

notation:

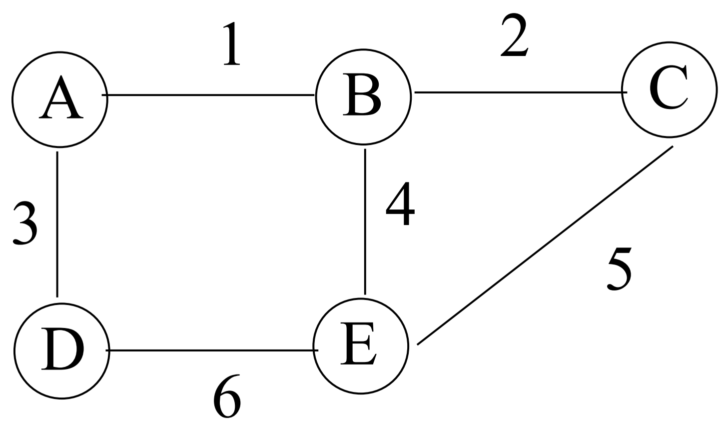
- **c(x,y): link cost from node x to y; = ∞ if not direct neighbors**
- **D(v): current value of cost of path from source to dest. v**
- **p(v): predecessor node along path from source to v**
- **N': set of nodes whose least cost path definitively known**

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'
```

Link State (using Djikstra's shortest path)



LSP database on a node

A - B 1

A - D 3

B - E 4

…..

……

- Important property of link state: Every node has "all" information about every other node.
- In class example.

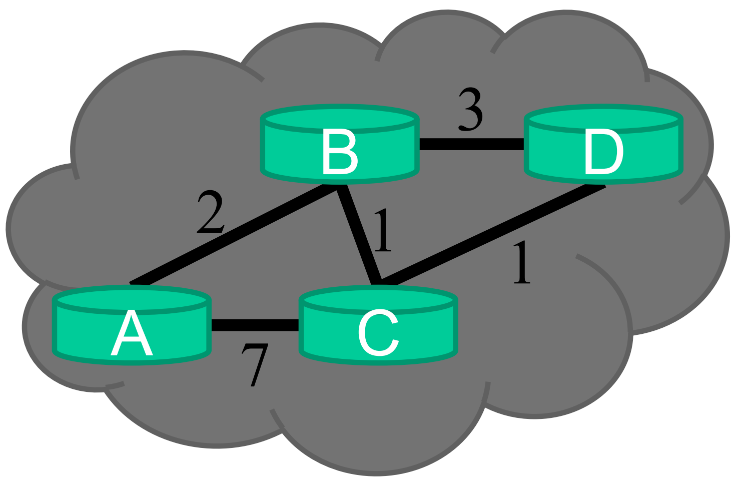
Distance Vector Routing

- **What is a distance vector?**
  - Current best known cost to reach a destination
- **Idea: exchange vectors among neighbors to learn about lowest cost paths**

Distance Vector Routing Algorithm

1. Wait for change in local link cost or message from neighbor
2. Recompute distance table
3. If least cost path to any destination has changed, notify neighbors

Distance Vector Initialization



1. **Initialization:**
2. **for all** neighbors V **do**
3. **if** V adjacent to A
4. $D(A, V) = c(A,V)$;
5. **else**
6. $D(A, V) = \infty$;

…

Node A

Dest.	Cost	Next
B	2	B
C	7	C
D	∞	

Node B

Dest.	Cost	Next
A	2	A
C	1	C
D	3	D

Node C

Dest.	Cost	Next
A	7	A
B	1	B
D	1	D

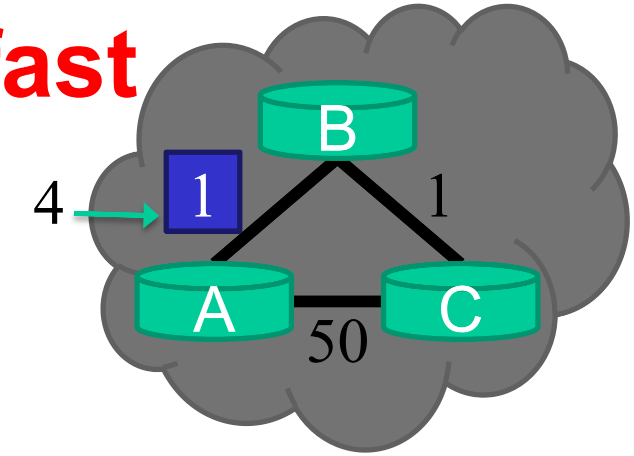
Node D

Dest.	Cost	Next
A	∞	
B	3	B
C	1	C

Distance Vector (DV) Routing

- See more details in class notes (loaded to resources section)
- Unlike link state, DV only knows the shortest path route to a destination from its neighbors.
- Distributed variation of Bellman-Ford algorithm.
- # of rounds to converge is roughly the length of the network.

Good news travels fast



Link Cost Changes, Algorithm Starts

Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

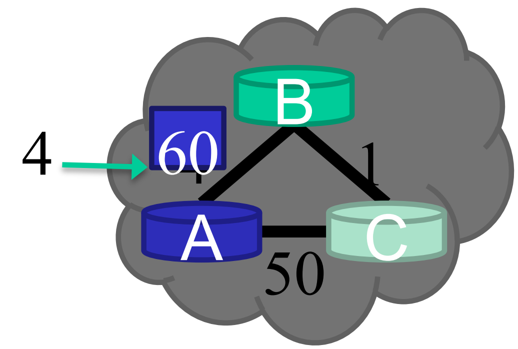
D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

D	C	N
A	2	B
B	1	B

Time

Bad news travels slowly: Count to Infinity Problem



Node B

D	C	N
A	4	A
C	1	C

D	C	N
A	6	C
C	1	C

D	C	N
A	6	C
C	1	C

D	C	N
A	8	C
C	1	C

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	7	B
B	1	B

D	C	N
A	7	B
B	1	B

Time

Count to infinity problem

- **Because of cycles**
- **There are some ways to improve this, we will not cover them**

Distance Vector (RIP) vs. Link State (OSPF)

- RIP uses UDP to exchange information
- OSPF uses IP packets to exchange information

Which is best?
In practice, it depends.
In general, link state is more popular.