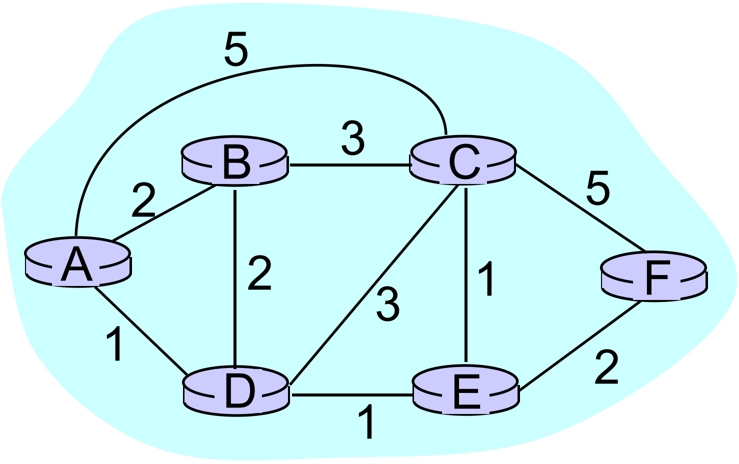
COMP/ELEC 429
Introduction to Computer Networks

Lecture 10: Intra-domain routing

Slides from Edward W. Knightly, T. S. Eugene Ng, Ion Stoica, Hui Zhang

Viewing Routing as a Policy

- Given multiple alternative paths, how to route information to destinations should be viewed as a policy decision
- What are some possible policies?
  - Shortest path (RIP, OSPF)
  - Most load-balanced
  - QoS routing (satisfies app requirements)
  - etc

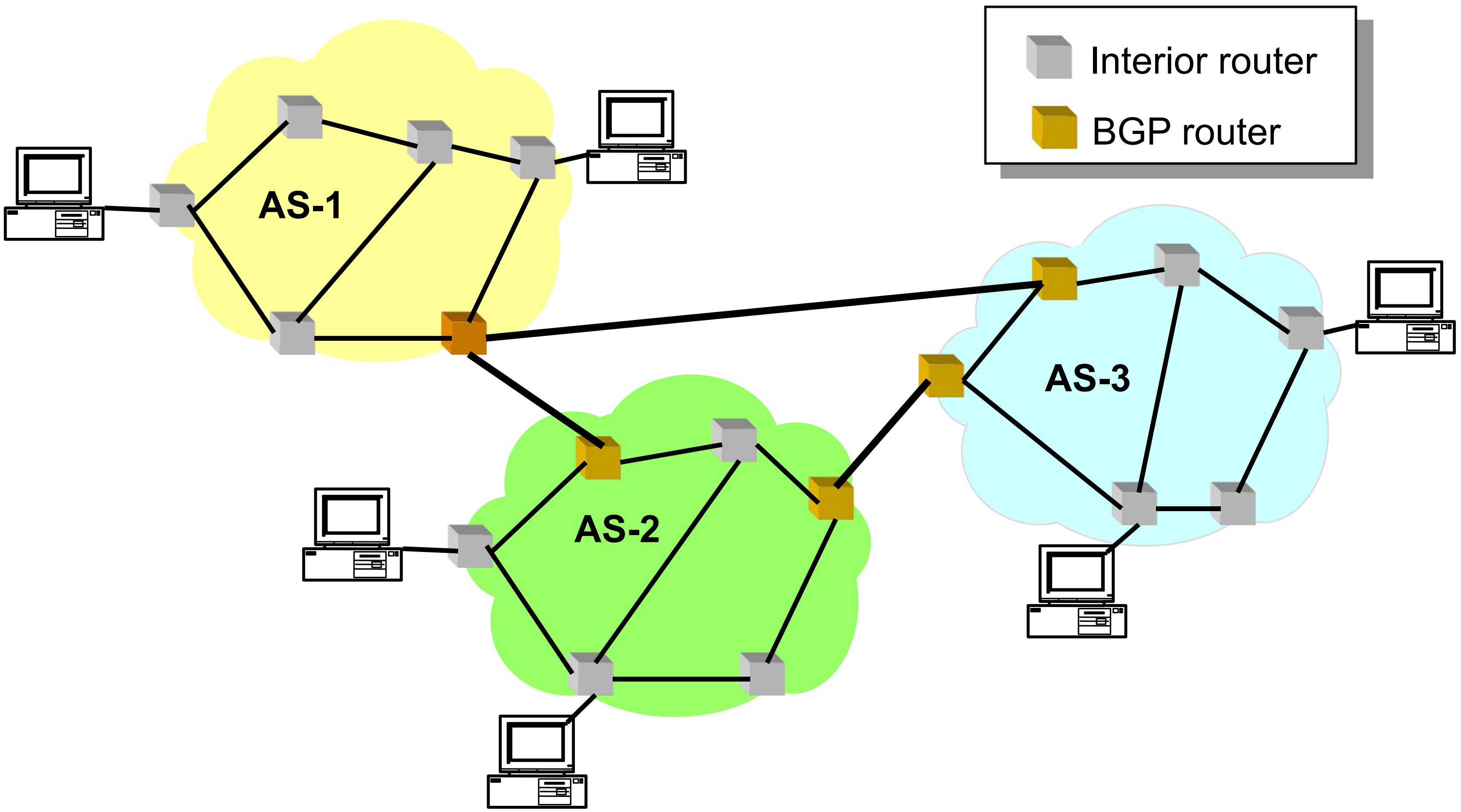


Internet Routing

- Internet topology roughly organized as a two level hierarchy
- First lower level – autonomous systems (AS's)
  - AS: region of network under a single administrative domain
- Each AS runs an intra-domain routing protocol
  - Distance Vector, e.g., Routing Information Protocol (RIP)
  - Link State, e.g., Open Shortest Path First (OSPF)
  - Possibly others

- Second level – inter-connected AS's
- Between AS's runs inter-domain routing protocols, e.g., Border Gateway Routing (BGP)
  - De facto standard today, BGP-4

Example

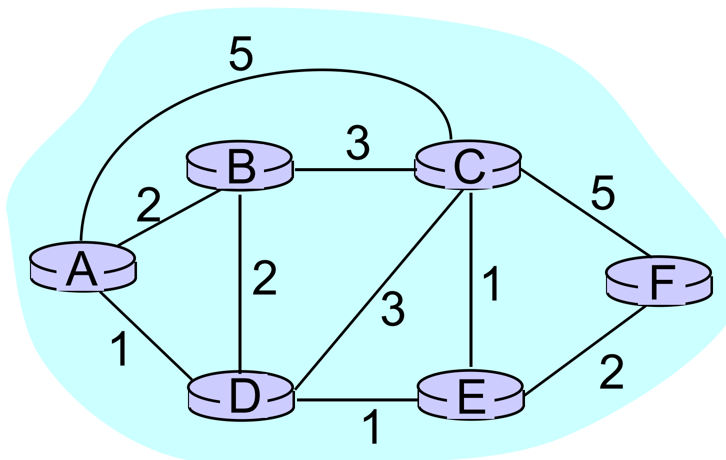


Intra-domain Routing Protocol

- Based on unreliable datagram delivery
- Link state
  – Open Shortest Path First (OSPF), based on Dijkstra's algorithm
  – Each router periodically floods *immediate* reachability information to other routers
  – Fast convergence, but high communication and computation overhead

Routing on a Graph

- Goal: determine a "good" path through the network from source to destination
  - Good often means the shortest path
- Network modeled as a graph
  - Routers ☾ nodes
  - Link ☾ edges
    - Edge cost: delay, congestion level,…

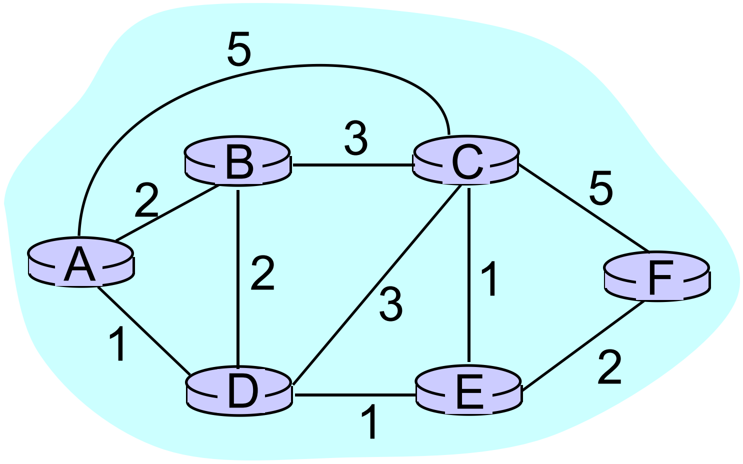


Link State Routing (OSPF): Flooding

- Each node knows its connectivity and cost to a direct neighbor
- Every node tells every other node this local connectivity/cost information
  - Via flooding
- In the end, every node learns the complete topology of the network

- E.g. A floods message

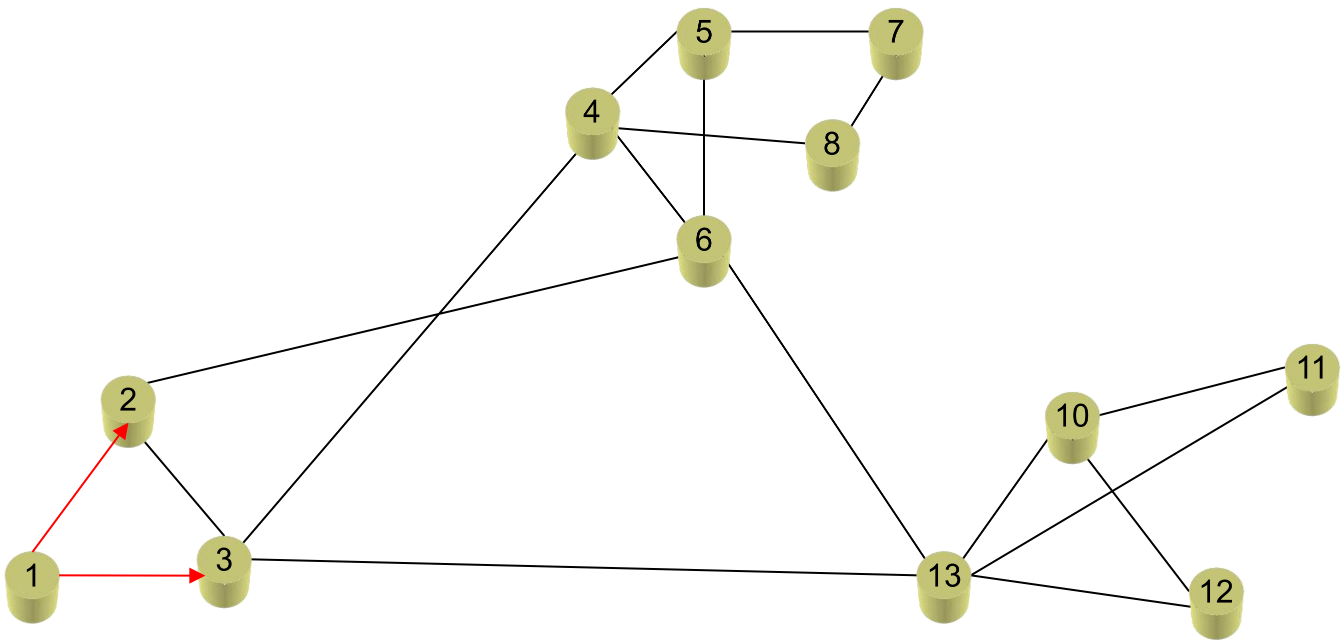
A connected to B cost 2
A connected to D cost 1
A connected to C cost 5



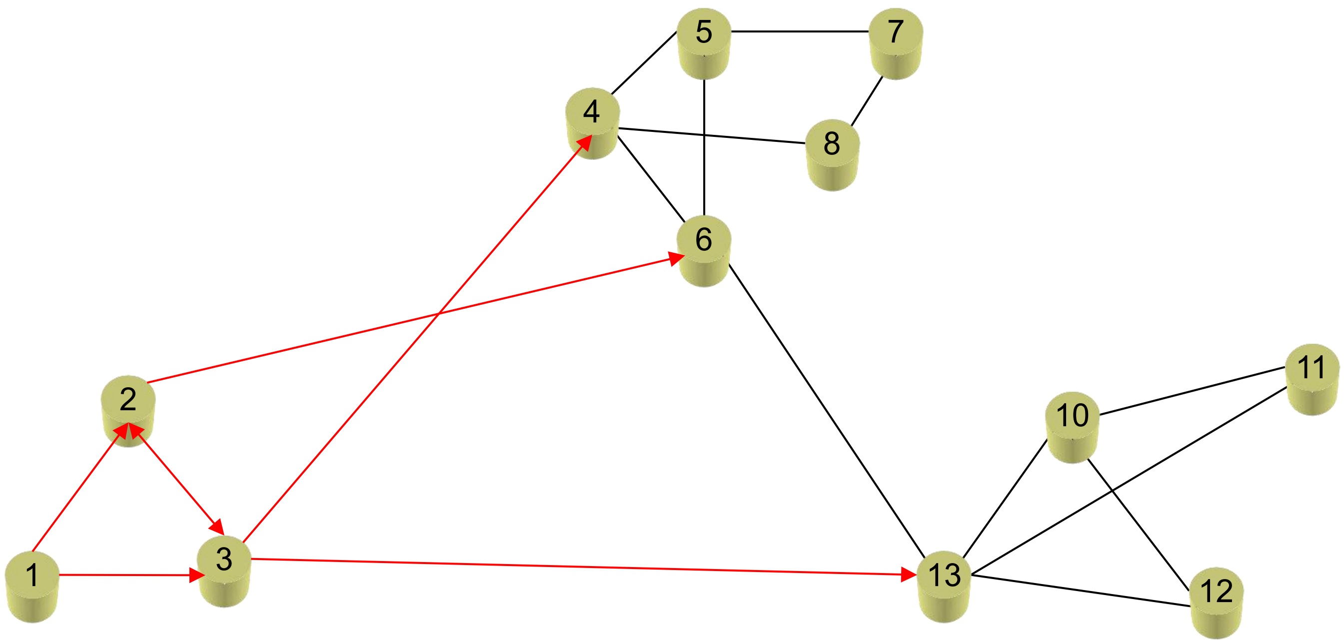
Flooding Details

- Each node periodically generates Link State Packet (LSP) contains
  - ID of node created LSP
  - List of direct neighbors and costs
  - Sequence number (64 bit, assume to never wrap around)
  - Time to live
- Flood is reliable
  - Use acknowledgement and retransmission
- Sequence number used to identify *newer* LSP
  - An older LSP is discarded
- Receiving node flood LSP to all its neighbors except the neighbor where the LSP came from
- LSP is also generated when a link's state changes (failed or restored)

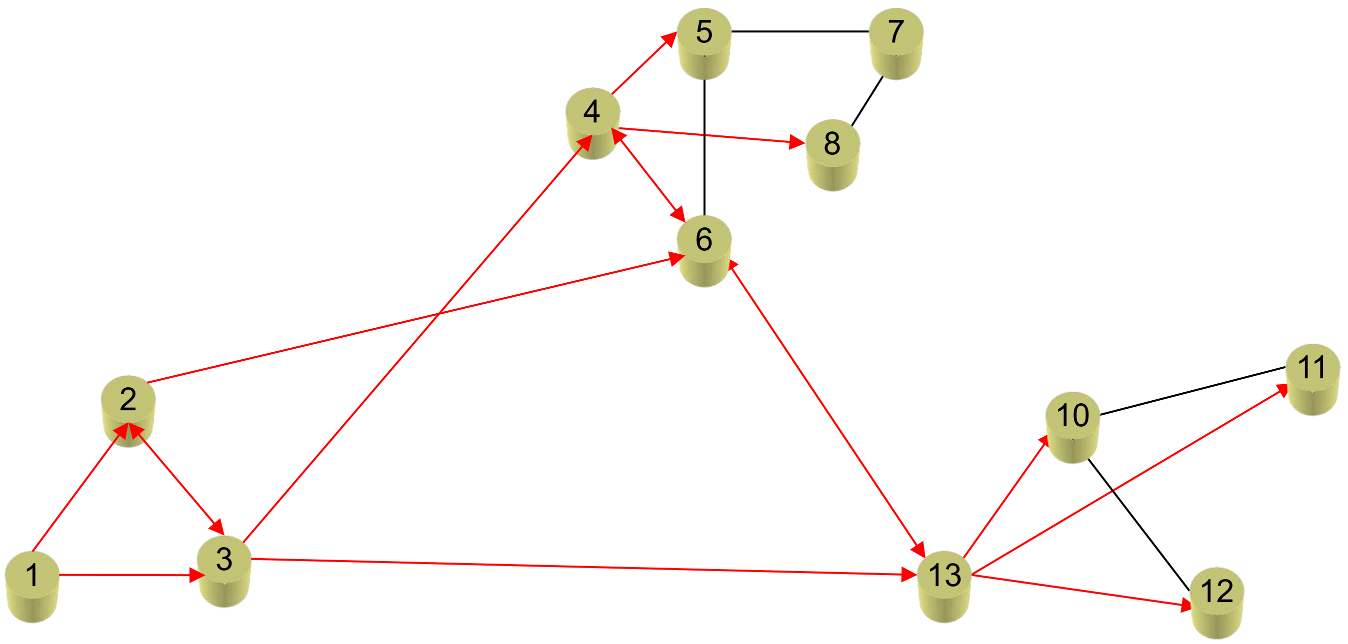
Link State Flooding Example



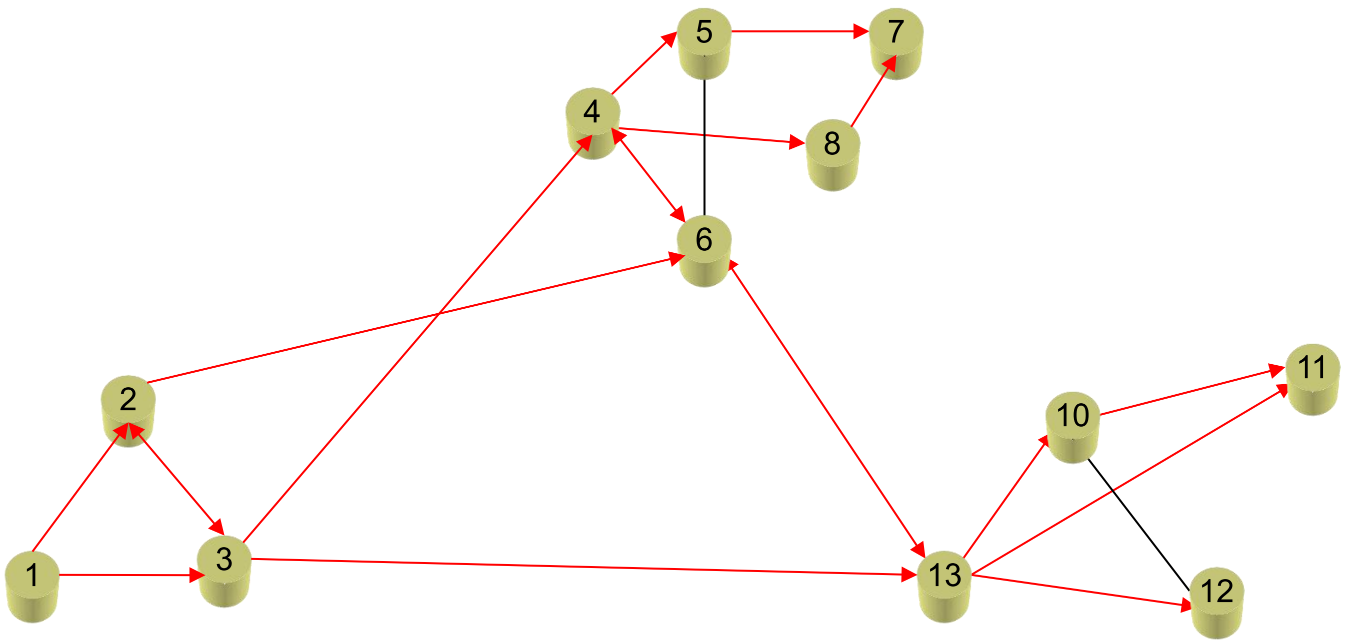
Link State Flooding Example



Link State Flooding Example



Link State Flooding Example



A Link State Routing Algorithm

Dijkstra's algorithm

- Net topology, link costs known to all nodes
  - Accomplished via "link state flooding"
  - All nodes have same info
- Compute least cost paths from one node ('source") to all other nodes
- Repeat for all sources

Notations

- c(i,j): link cost from node *i* to *j*; cost infinite if not direct neighbors
- D(v): current value of cost of path from source to node *v*
- p(v): predecessor node along path from source to *v*, that is next to *v*
- S: set of nodes whose least cost path definitively known

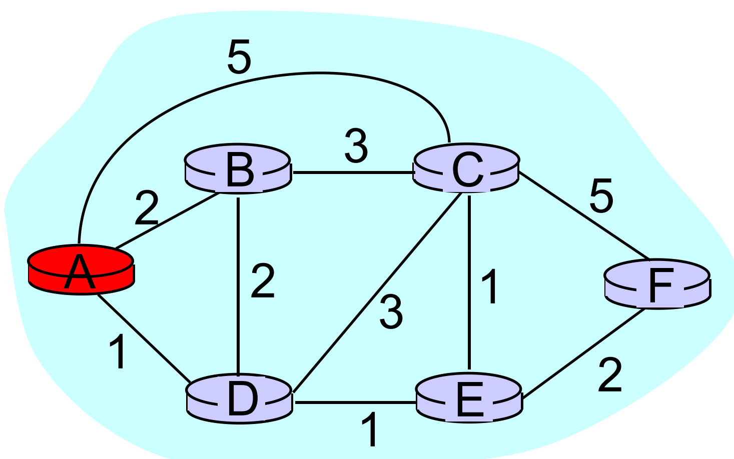
Dijsktra's Algorithm (A "Greedy" Algorithm)

```
Initialization:
  S = {A};
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v);
      else D(v) = ∞;

Loop
    find w not in S such that D(w) is a minimum;
    add w to S;
    update D(v) for all v adjacent to w and not in S:
       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 S;
```

Example: Dijkstra's Algorithm

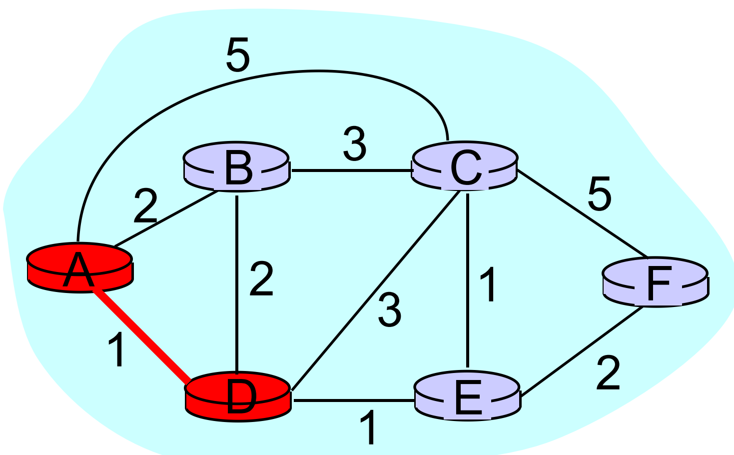
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						



Initialization:
  S = {A};
  for all nodes *v*
    if *v* adjacent to *A*
      then D(v) = c(A,v);
      else D(v) = ∞;
…

Example: Dijkstra's Algorithm

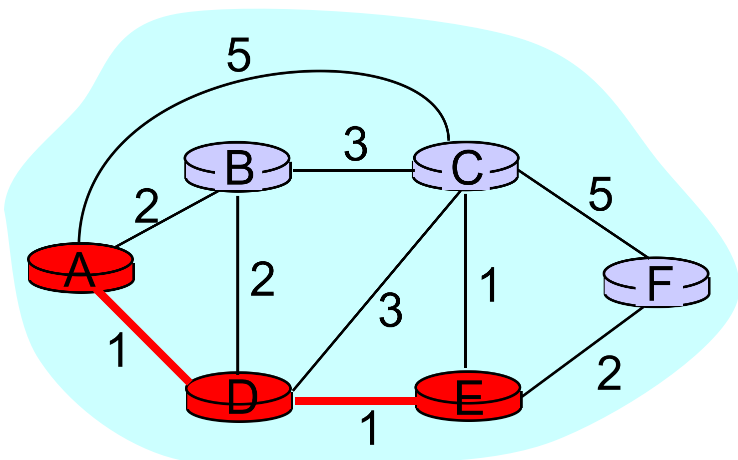
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						




```
…
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
   update D(v) for all v adjacent
   to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

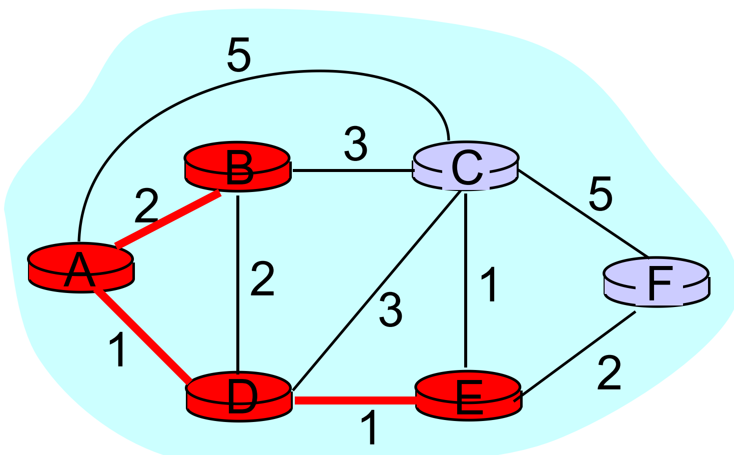
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						




```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
 update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

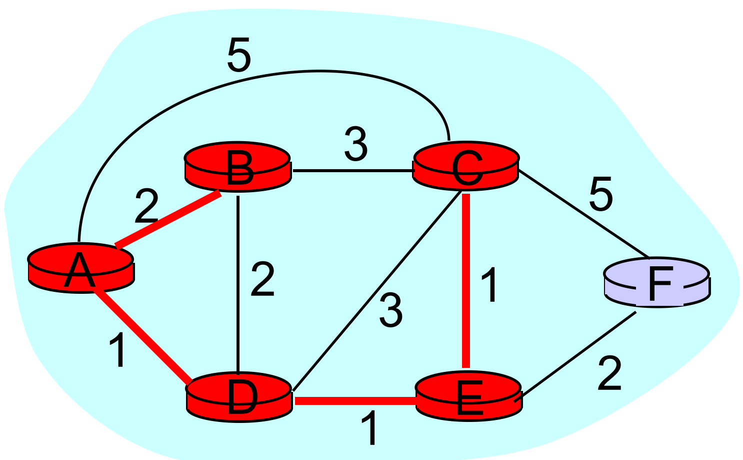
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						




```
…
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
  update D(v) for all v adjacent
   to w and not in S:
     D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

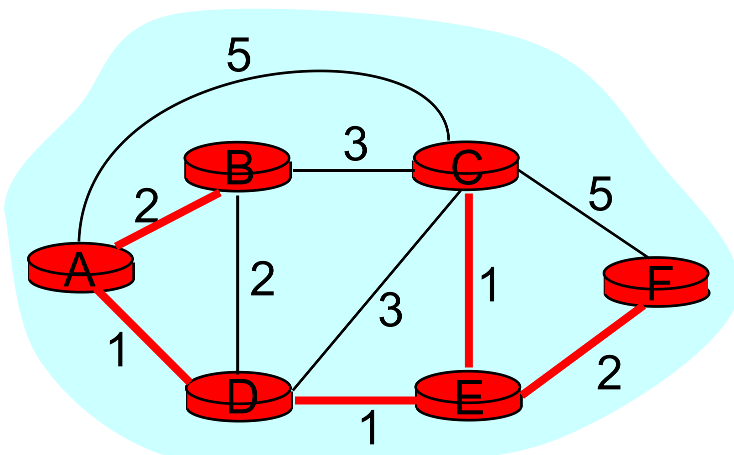
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						




```
...
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
  update D(v) for all v adjacent
   to w and not in S:
     D(v) = min( D(v), D(w) + c(w,v) );
 until all nodes in S;
```

Example: Dijkstra's Algorithm

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					




```
…
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
  update D(v) for all v adjacent
   to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
   until all nodes in S;
```

Overview of Assignment

Given a set of N routers, the goal is for EACH router to:

(a) exchange HELLO packets with neighbours

(b) create Link State Advertisement (LSA) packets based on neighboring nodes' info

(c) broadcast the LSA packets to all other routers in the network

(d) construct the network topology based on the LSA packets received from other routers,

(e) determining the routing table entries based on this topology, by using Dijkstra' algorithm

(single source –all nodes shortest paths)

If multiple equal-cost paths exist, any one of them can be reported.

Overview of Assignment

- The HELLO packets will be exchanged every x seconds
- LSA updates will be sent every y seconds
- Routing table computation will be done every z seconds

./ospf -i id -f infile -o outfile -h hi -a lsai -s spfi

The values specified in the command line are:

- -i id: Node identifier value(i)
- -f infile: Input file
- -o outfile: Output file
- -h hi: HELLO INTERVAL (in seconds)
- -a lsai: LSA INTERVAL (in seconds)
- -s spfi: SPF INTERVAL (in seconds)

Overview of Assignment

5		10	
0	2	2	8
2	3	5	10
3	4	6	20
4	7	4	10
...			

The first entry on the first line specifies the number of routers (N) The node indices go from 0 to (N − 1).

The second entry on the first line specifies the number of links.

Each subsequent row contains the tuple (i, j, MinC ij , MaxC ij). This implies a bidirectional link between nodes i and j .