Internet Protocols
Fall 2006

Lectures 9-10

Intra-domain Routing

Andreas Terzis

Outline

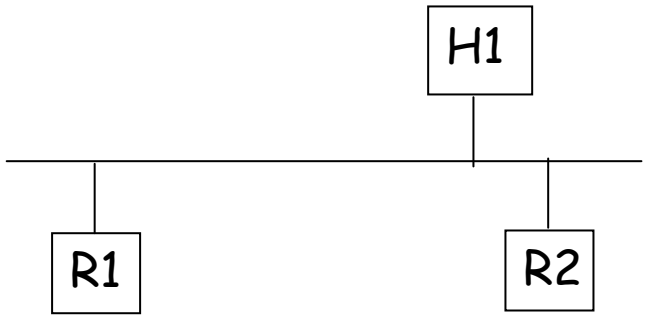
- Internet Intra-Domain Routing
  - Distance Vector
  - Shortest Path

Forwarding Table

```
terzis@peregrine(~)> netstat -nr
Routing Table:
Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
128.220.13.0         128.220.13.85         U        3   8438  le0
224.0.0.0            128.220.13.85         U        3      0  le0
default              128.220.13.1          UG       0  58348
127.0.0.1            127.0.0.1             UH       0    668  lo0
```

- How are entries inserted in fwd table?
  - Statically (`route add`,`route delete`)
  - ICMP Redirects
  - Dynamic Routing Protocols

ICMP Redirects



- Default router informs host that better router exists
  - How does it know that?
- Host adds a new entry in it's routing table
  - Security concerns?

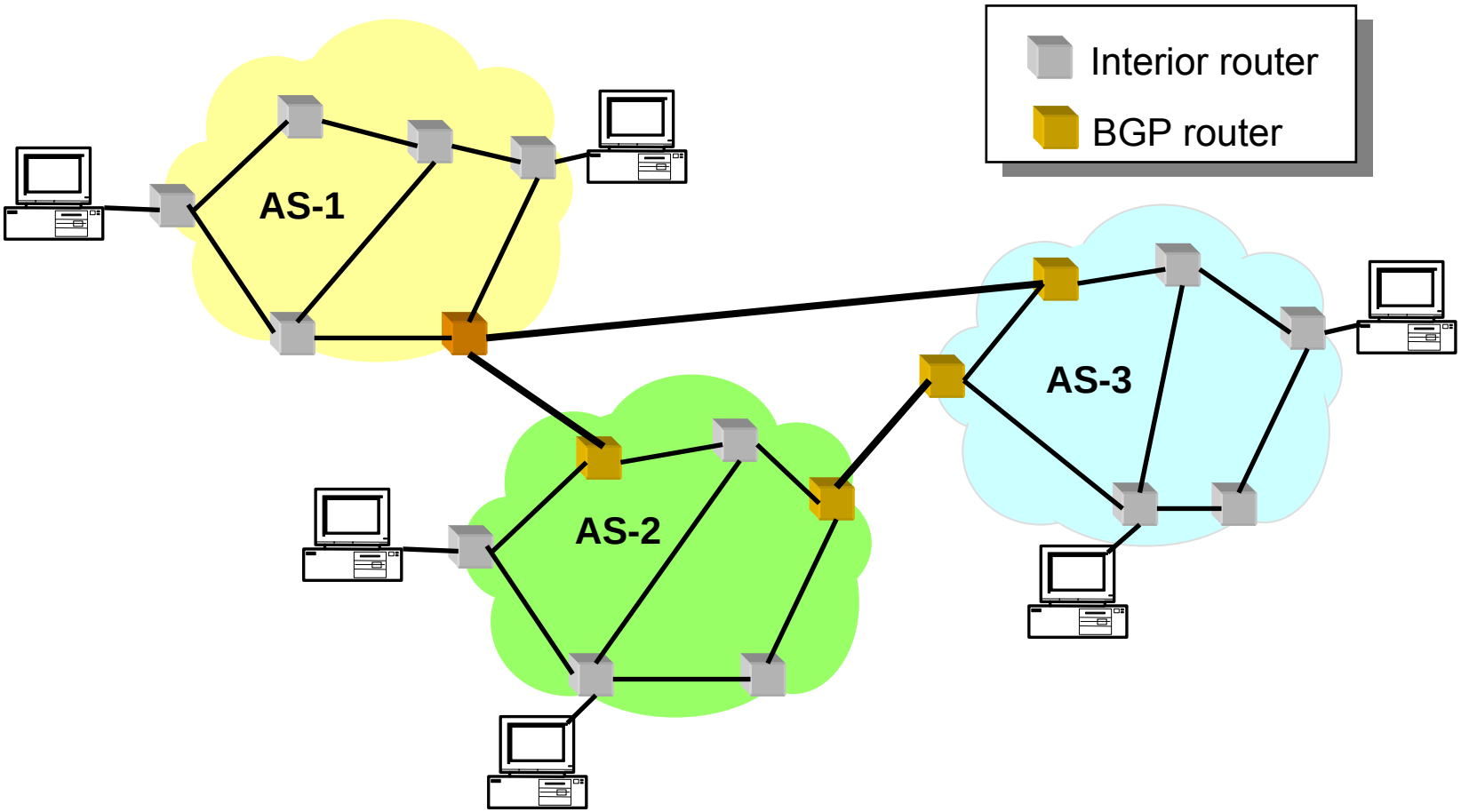
What is Routing?

- Routing is the core function of a network
- It ensures that
  - Information accepted for transfer at a source node
  - Is delivered to the correct set of destination nodes, at reasonable levels of performance.

Internet Routing

- Internet organized as a two level hierarchy
- First level – autonomous systems (AS's)
  - AS – region of network under single administrative control
- AS's run an intra-domain routing protocols
  - Distance Vector, e.g., Routing Information Protocol (RIP)
  - Link State, e.g., Open Shortest Path First (OSPF)
- Between AS's runs inter-domain routing protocols, e.g., Border Gateway Routing (BGP)
  - De facto standard today, BGP-4

Example



Intra-domain Routing Protocols

- Based on unreliable datagram delivery
- Distance vector
  - Routing Information Protocol (RIP), based on Bellman-Ford
  - Each neighbor periodically exchange reachability information to its neighbors
  - Minimal communication overhead, but it takes long to converge, i.e., in proportion to the maximum path length
- Link state
  - Open Shortest Path First (OSPF), based on Dijkstra
  - Each network periodically floods immediate reachability information to other routers
  - Fast convergence, but high communication and computation overhead

Distance Vector Routing Algorithm

- Iterative: continues until no nodes exchange info
- Asynchronous: nodes need *not* exchange info/iterate in lock step
- Distributed: each node communicates *only* with directly-attached neighbors
- Each router maintains
  - Row for each possible destination
  - Column for each directly-attached neighbor to node
  - Entry in row Y and column Z of node X ➔ best known distance from X to Y, via Z as next hop (remember this !)
- *Note: for simplicity in this lecture examples we show only the shortest distances to each destination*

Distance Vector Routing

- Each local iteration caused by:
  - Local link cost change
  - Message from neighbor: its least cost path change from neighbor to destination
- Each node notifies neighbors *only* when its least cost path to any destination changes
  - Neighbors then notify their neighbors if necessary

Each node:

wait for (change in local link cost or msg from neighbor)

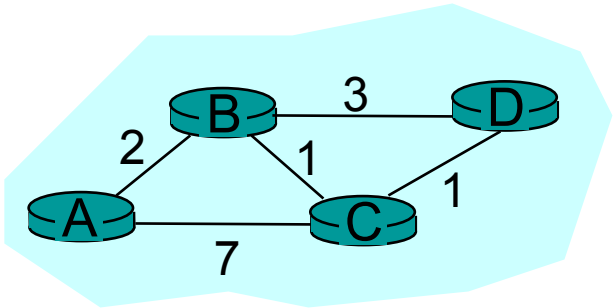
recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Distance Vector Algorithm (cont'd)

```
Initialization:
  for all nodes V  do
    if V adjacent to A
      D(A, V) = c(A,V);
    else
      D(A, V) = ∞;
    loop:
  wait (until A sees a link cost change to neighbor V
          or until A receives update from neighbor V)
  if (D(A,V) changes by d)
     for all destinations Y through V do
        D(A,Y) =  D(A,Y) + d
  else if (update D(V, Y) received from V)
     /* shortest path from V to some Y has changed  */
     D(A,Y) = D(A,V) + D(V, Y);
  if (there is a new minimum for destination Y)
     send D(A, Y) to all neighbors
  forever
```

Example: Distance Vector Algorithm



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

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

Node C

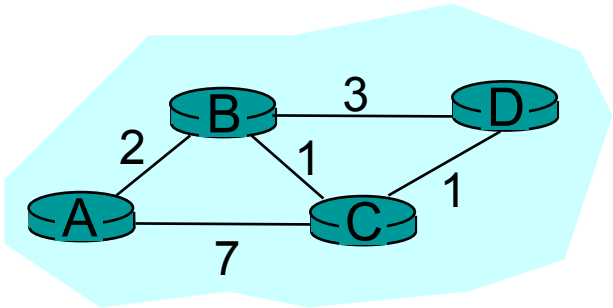
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
Initialization:
  for all nodes V do
    if V adjacent to A
      D(A, V) = c(A,V);
    else
      D(A, V) = ∞;
…
```

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	**8**	C

Node B

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

D(A, D) = D(A, C) + D(C,D) = 7 + 1 = 8

(D(C,A), D(C,B), D(C,D))

Node C

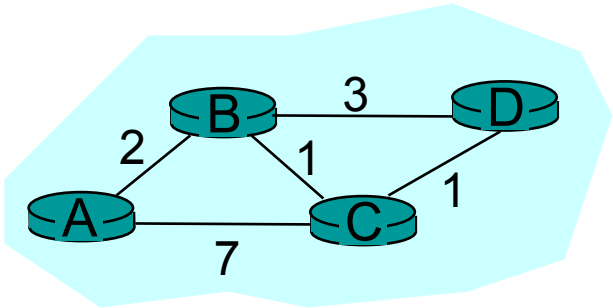
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
7     loop:
    …
13   else if (update D(V, Y) received from V)
14     D(A,Y) = D(A,V) + D(V, Y);
15   if (there is a new min. for destination Y)
16     send D(A, Y) to all neighbors
17  forever
```

Example: 1st Iteration (B→A, C→A)



Node A

Dest.	Cost	NextHop
B	2	B
C	**3**	**B**
D	**5**	**B**

Node B

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

D(A,D) = D(A,B) + D(B,D) = 2 + 3 = 5

D(A,C) = D(A,B) + D(B,C) = 2 + 1 = 3

```
7        loop:
   …
13   else if (update D(V, Y) received from V)
14     D(A,Y) = min(D(A,V), D(A,V) + D(V, Y))
15   if (there is a new min. for destination Y)
16     send D(A, Y) to all neighbors
17  forever
```

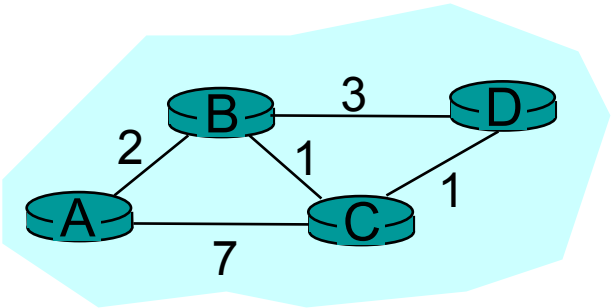
Node C

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

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: End of 1st Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	**3**	**B**
D	**5**	**B**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	**2**	**C**

Node C

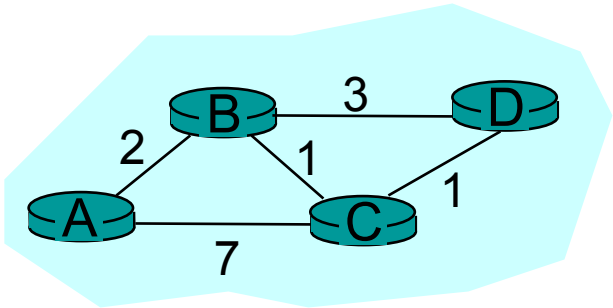
Dest.	Cost	NextHop
A	**3**	**B**
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	**5**	**B**
B	3	B
C	1	C

```
7       loop:
    …
13   else if (update D(V, Y) received from V)
14     D(A,Y) = D(A,V) + D(V, Y);
15   if (there is a new min. for destination Y)
16     send D(A, Y) to all neighbors
17  forever
```

Example: End of 2nd Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	**4**	**B**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

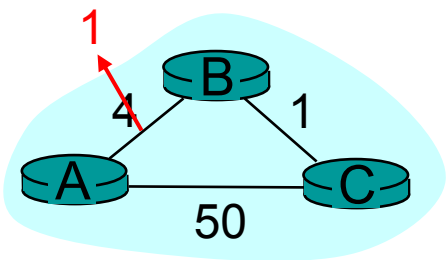
Dest.	Cost	NextHop
A	**4**	**C**
B	3	B
C	1	C

```
    loop:
  …
  else if (update D(V, Y) received from V)
    D(A,Y) = D(A,V) + D(V, Y);
  if (there is a new min. for destination Y)
    send D(A, Y) to all neighbors
forever
```

Nothing changes → algorithm terminates

Distance Vector: Link Cost Changes

```
loop:
  wait (until A sees a link cost change to neighbor V
        or until A receives update from neighbor V)
  if (D(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) = D(A,Y) + d
  else if (update D(V, Y) received from V)
    D(A,Y) = D(A,V) + D(V, Y);
  if (there is a new minimum for destination Y)
    send D(A, Y) to all neighbors
forever
```



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

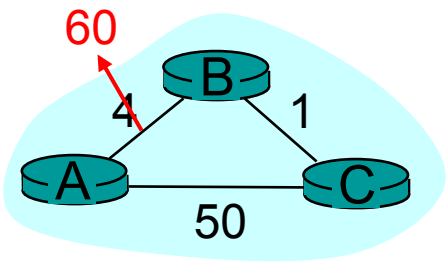
"good news travels fast"

time

Link cost changes here **Algorithm terminates**

Distance Vector: Count to Infinity Problem

```
loop:
  wait (until A sees a link cost change to neighbor V
        or until A receives update from neighbor V)
  if (D(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) = D(A,Y) + d ;
  else if (update D(V, Y) received from V)
    D(A,Y) = D(A,V) + D(V, Y);
  if (there is a new minimum for destination Y)
    send D(A, Y) to all neighbors
forever
```



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	2	B
B	1	B

...

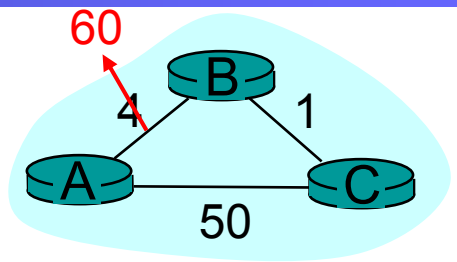
time

"bad news travels slowly"

Link cost changes here; recall from slide 23 that B also maintains shortest distance to A through C, which is 6. Thus D(B, A) becomes 6 !

Distance Vector: Poisoned Reverse

- If C routes through B to get to A:
  - C tells B its (C's) distance to A is infinite (so B won't route to A via C)
  - Will this completely solve count to infinity problem?



Node B

D	C	N
A	4	A
C	1	C

D	C	N
A	**60**	A
C	1	C

D	C	N
A	60	A
C	1	C

D	C	N
A	**51**	**C**
C	1	C

D	C	N
A	51	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	**50**	A
B	1	B

D	C	N
A	50	A
B	1	B

D	C	N
A	50	A
B	1	B

time

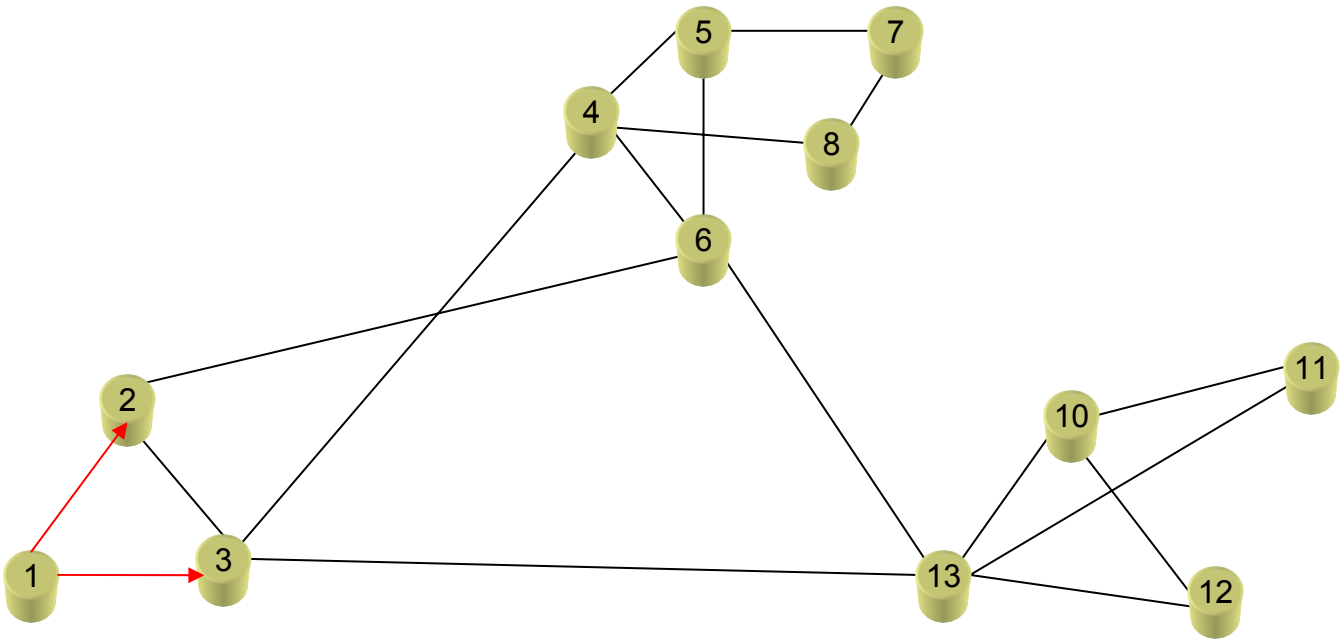
Link cost changes here; B updates D(B, A) = 60 as C has advertised D(C, A) = ∞

Algorithm terminates

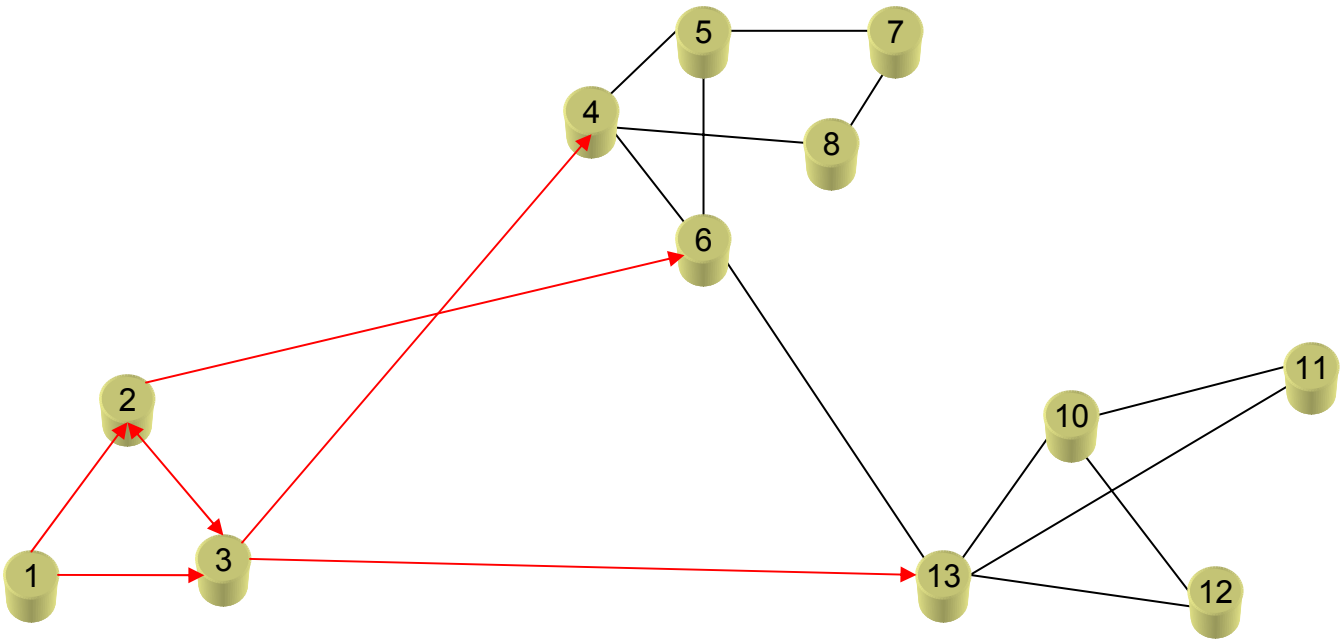
Outline

- Distance Vector
- Link State

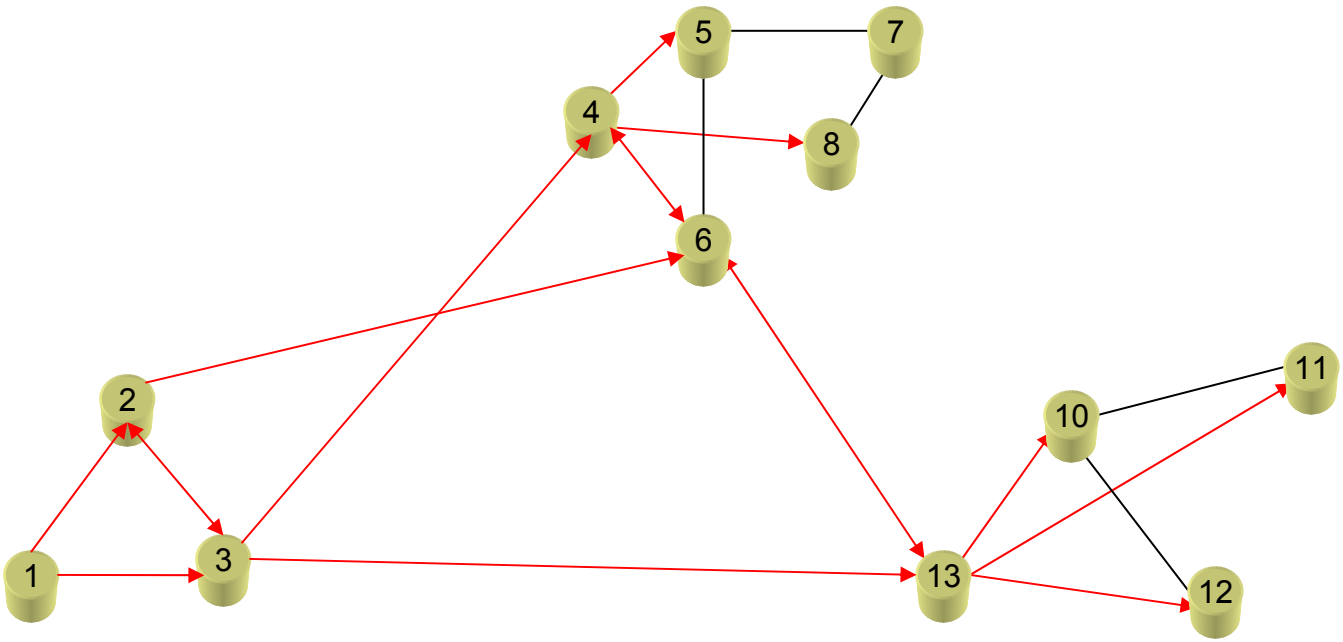
Link State Flooding Example



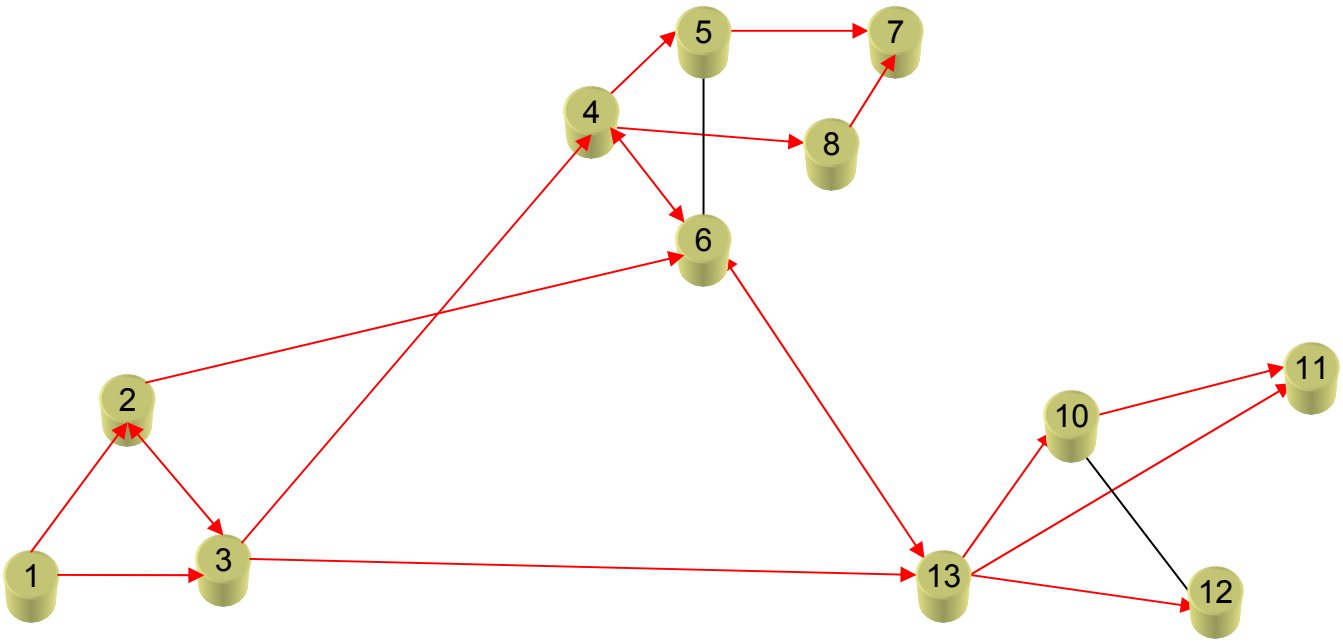
Link State Flooding Example



Link State Flooding Example



Link State Flooding Example



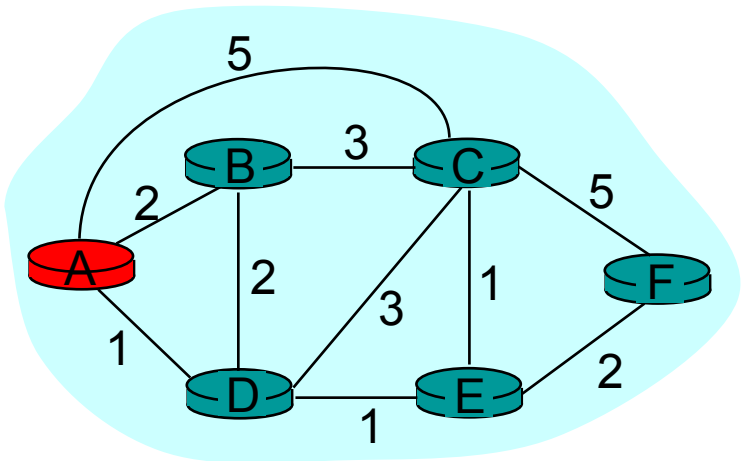
Dijsktra's Algorithm

```
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

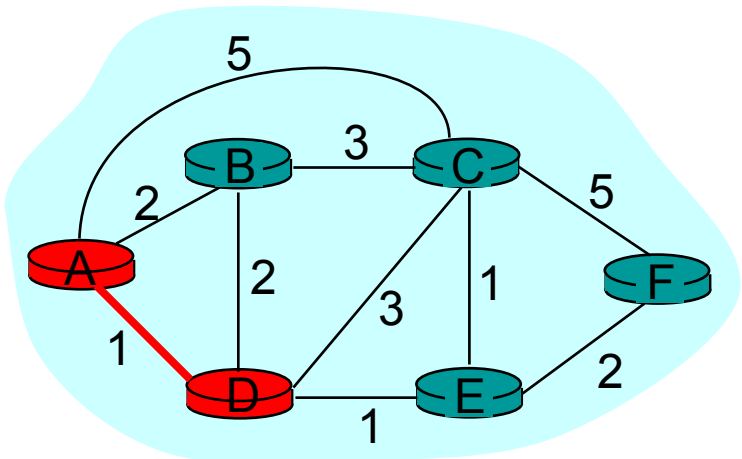
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						




```
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

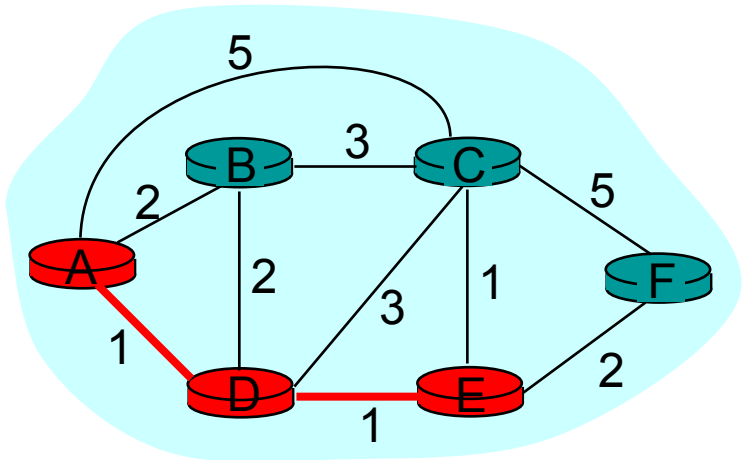
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						




```
…
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

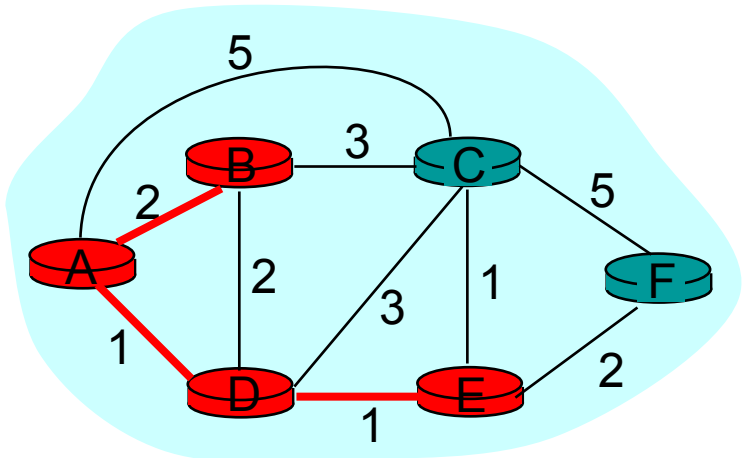
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						




```
…
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

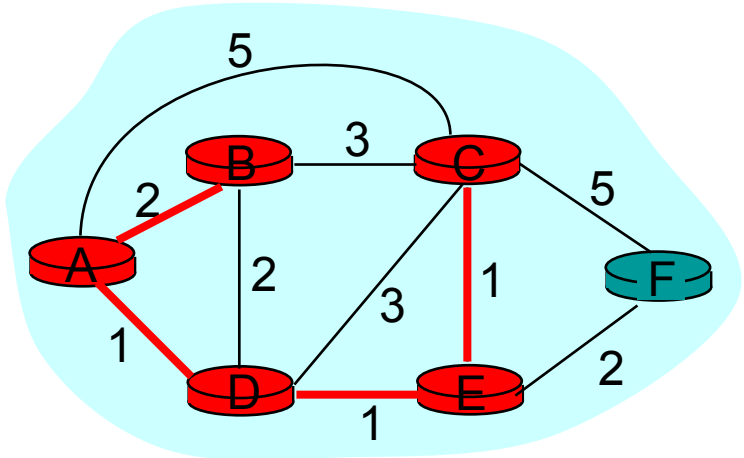
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						




```
…
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

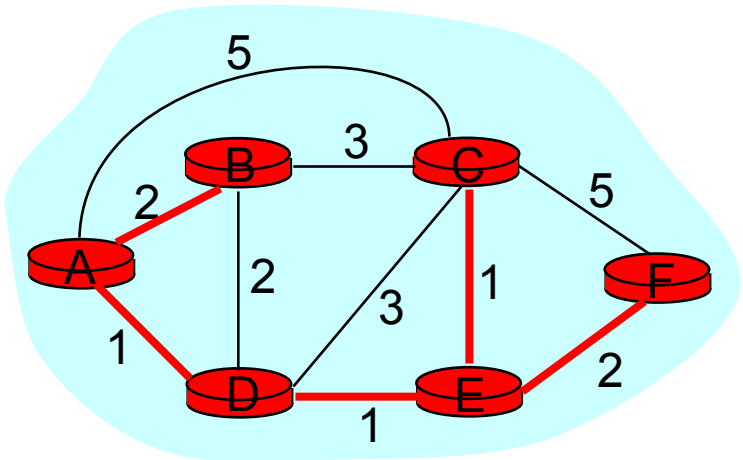
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						




```
…
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;
```

Complexity

- Assume a network consisting of n nodes
  - Each iteration: need to check all nodes, w, not in S
  - n*(n+1)/2 comparisons: $O(n^2)$
  - More efficient implementations possible: O(n*log(n))