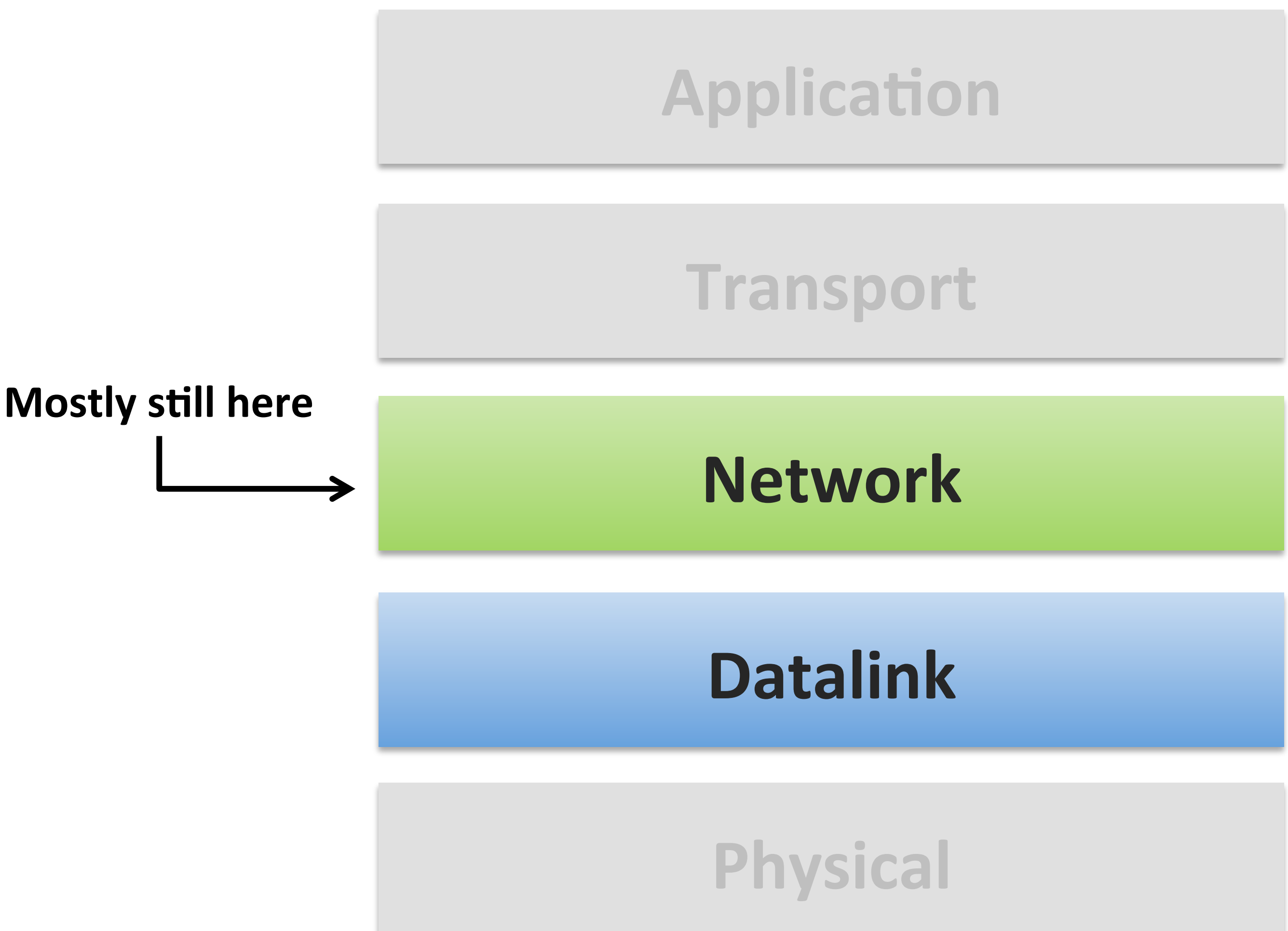
Routing

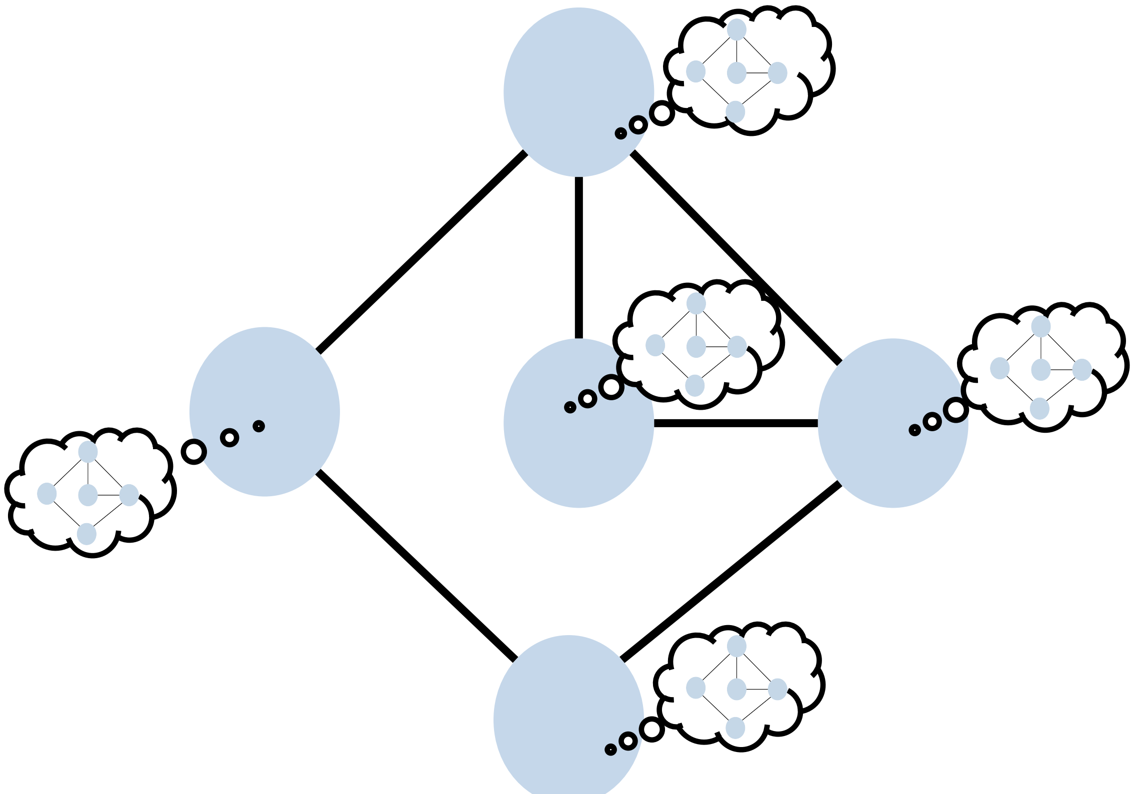
Link-State, Distance-Vector

CS168 Section 2

First off, where are we?



Link-State Routing



Every node has **global knowledge** of the entire topology and does **local route computation** using **Djikstra's Algorithm.**

Link-State Routing

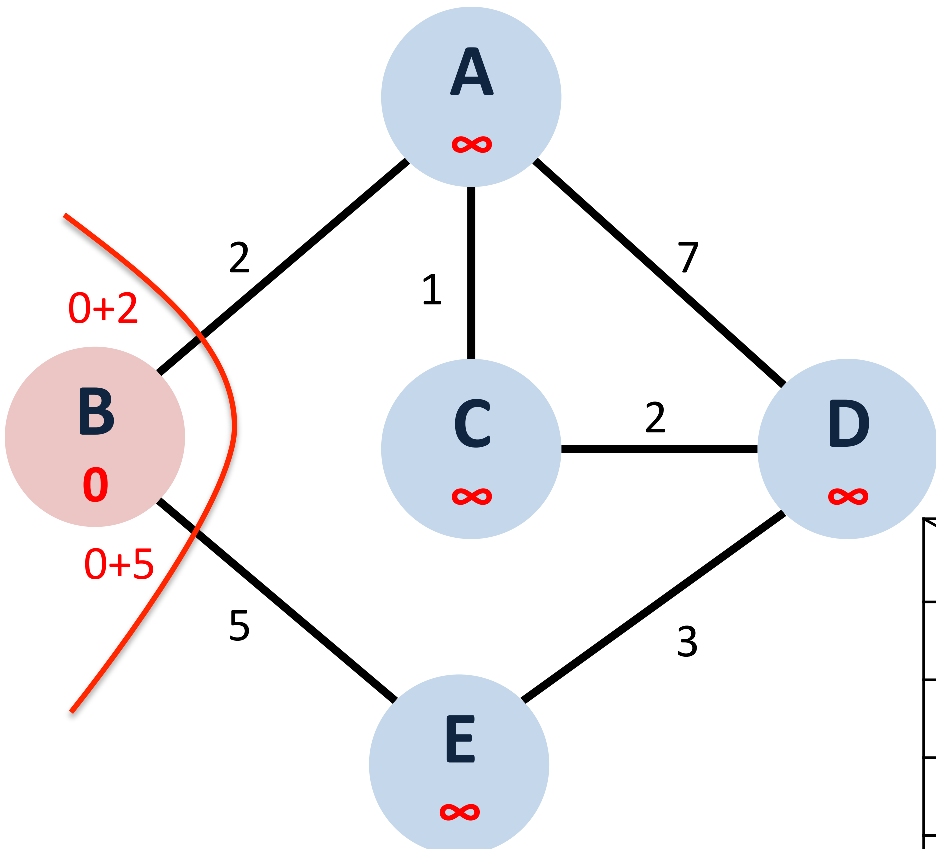
Djikstra's Algorithm

```
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:
    if D(w) + c(w,v) < D(v) then
      // w gives us a shorter path to v than we've found so far
      D(v) = D(w) + c(w,v); p(v) = w;
until all nodes in S;
```

- c(i,j): link cost from node *i* to *j*
- D(v): current cost source → *v*
- p(v): *v*'s predecessor along path from source to *v*
- S: set of nodes whose least cost path definitively known

Link-State Routing



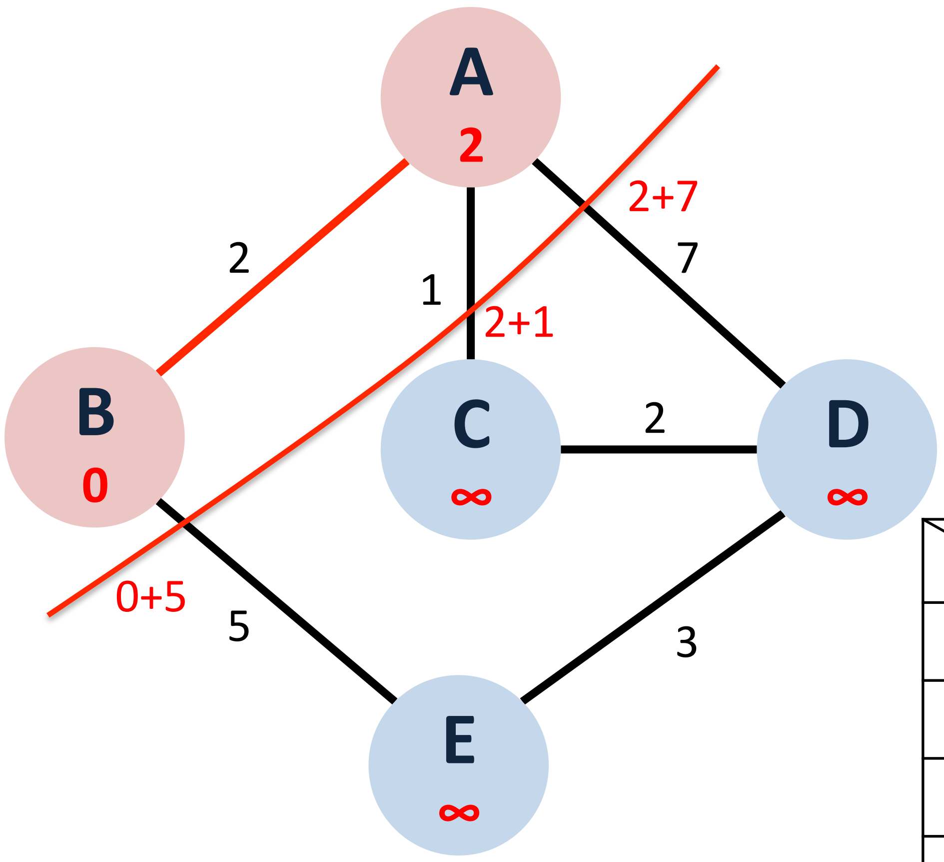
S: set of nodes whose least cost path definitively known

Node B

Dest / *i*	*A*	*C*	*D*	*E*
1				
2				
3				
4				

S

Link-State Routing

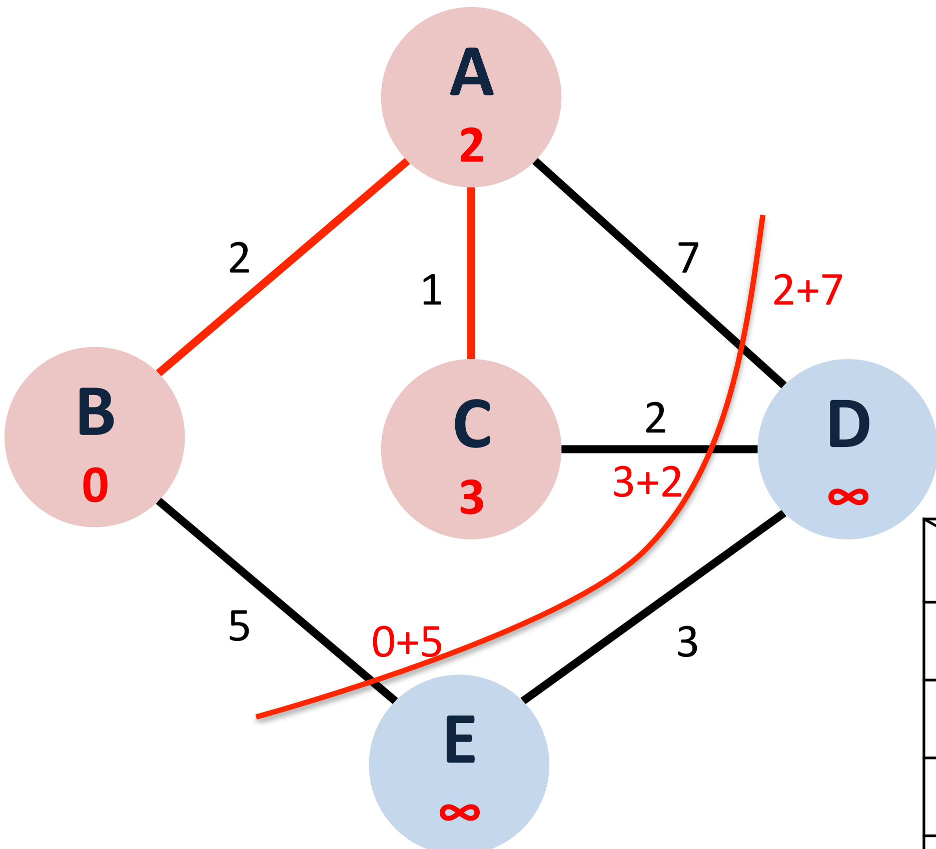


S: set of nodes whose least cost path definitively known

Node B

Dest / *i*	*A*	*C*	*D*	*E*	*S*
1	(2, B)	∞	∞	(5, B)	BA
2					
3					
4					

Link-State Routing

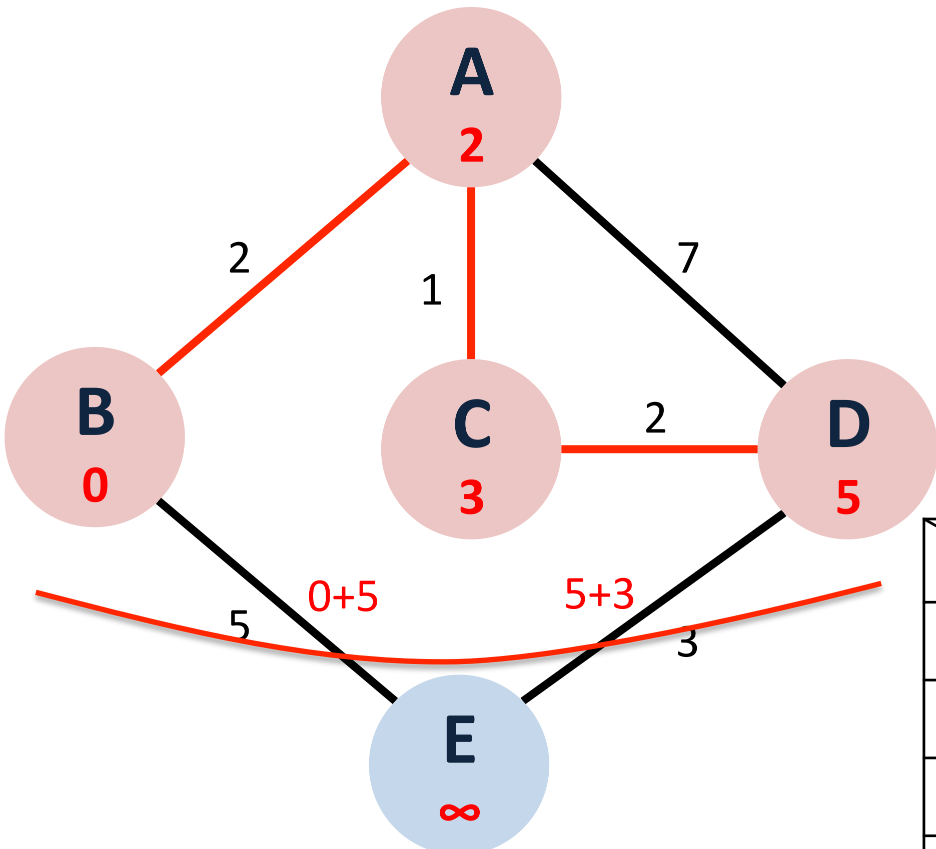


S: set of nodes whose least cost path definitively known

Node B

Dest / i	A	C	D	E	S
1	(2, B)	∞	∞	(5, B)	BA
2	(2, B)	(3, A)	(9, A)	(5, B)	BAC
3					
4					

Link-State Routing

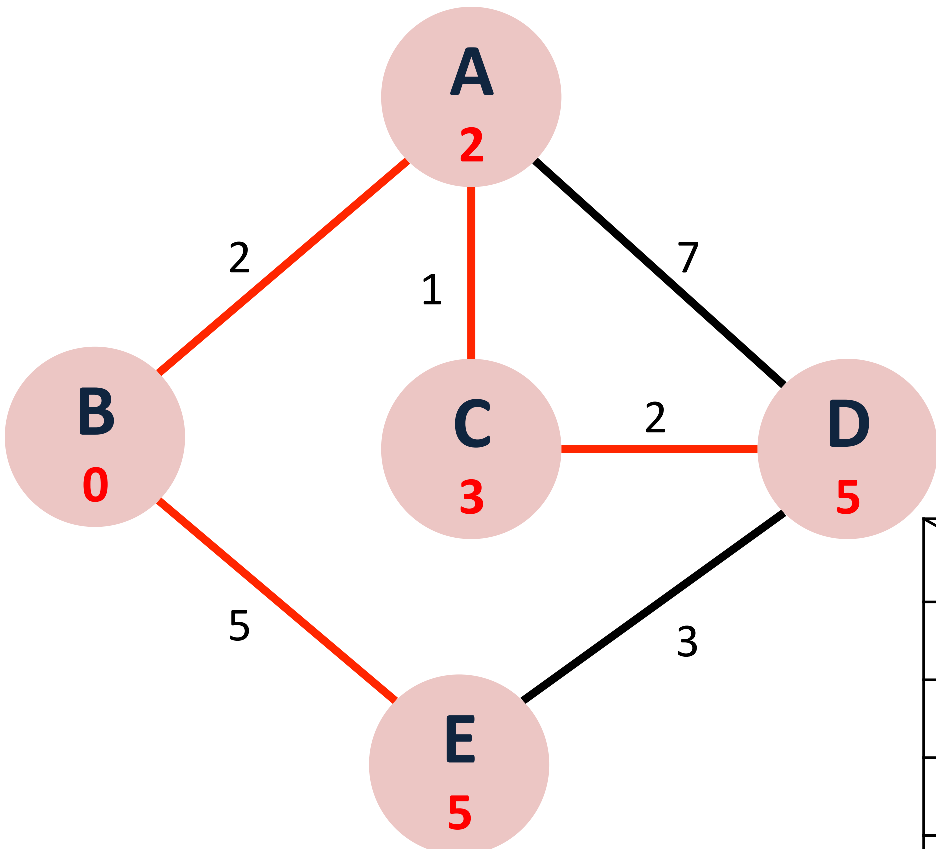


S: set of nodes whose least cost path definitively known

Node B

Dest / i	A	C	D	E	S
1	(2, B)	∞	∞	(5, B)	BA
2	(2, B)	(3, A)	(9, A)	(5, B)	BAC
3	(2, B)	(3, A)	(5, C)	(5, B)	BACD
4					

Link-State Routing

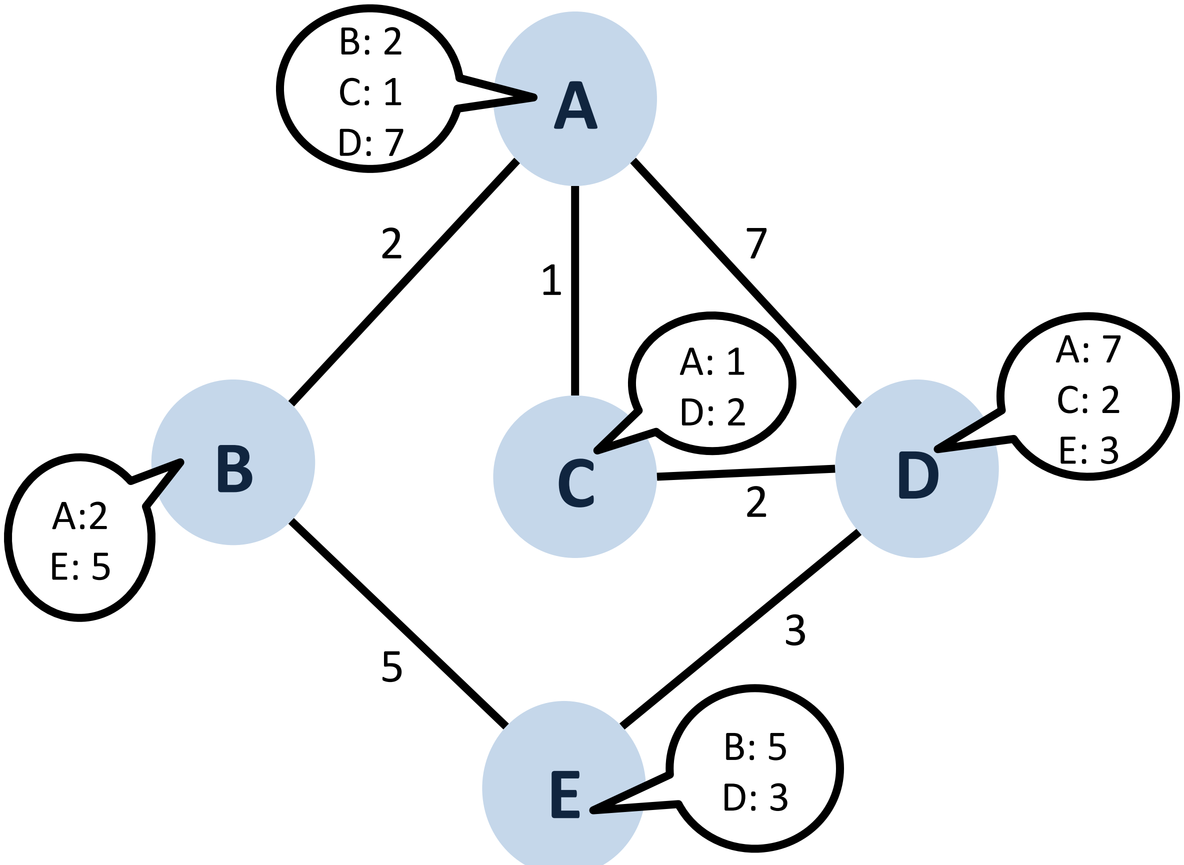


S: set of nodes whose least cost path definitively known

Node B

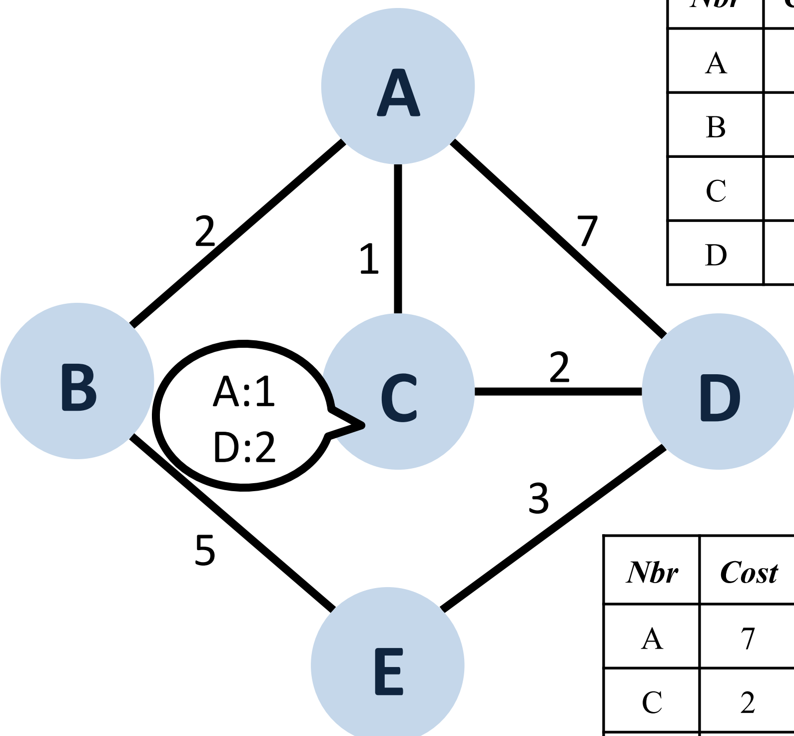
Dest / i	A	C	D	E	S
1	(2, B)	∞	∞	(5, B)	BA
2	(2, B)	(3, A)	(9, A)	(5, B)	BAC
3	(2, B)	(3, A)	(5, C)	(5, B)	BACD
4	(2, B)	(3, A)	(5, C)	(5, B)	BACDE

Distance-Vector Routing



Every node has **local knowledge** about its neighbors and does **global route computation** using **Bellman-Ford's Algorithm.**

Distance-Vector Routing



Node A

Nbr	Cost
A	0
B	2
C	1
D	7

To / From	A	B	C	D
A	0, A	2, A	1, A	7, A
B	-	0	-	-
C	-	-	0	-
D	-	-	-	0

⇨

To / From	A	B	C	D
A	0, A	2, A	1, A	3, C
B	-	0	-	-
C	1	-	0	2
D	-	-	-	0

Node D

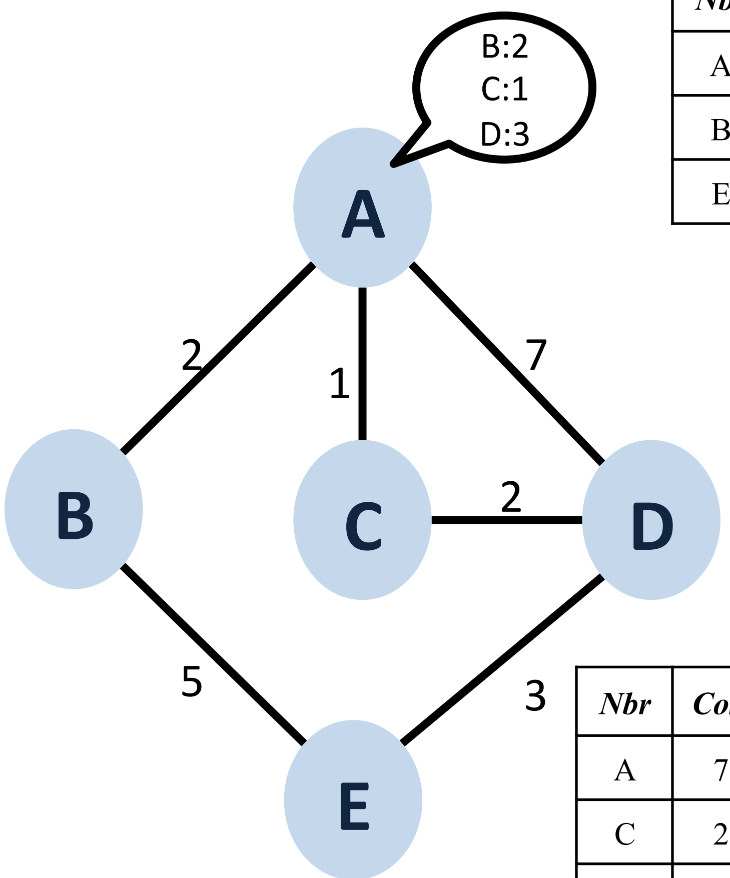
Nbr	Cost
A	7
C	2
D	0
E	3

To / From	A	C	D	E
A	0	-	-	-
C	-	0	-	-
D	7, D	2, D	0, D	3, D
E	-	-	-	0

⇨

To / From	A	C	D	E
A	0	-	-	-
C	1	0	2	-
D	3, C	2, D	0, D	3, D
E	-	-	-	0

Distance-Vector Routing



Node B

Nbr	Cost
A	2
B	0
E	5

From \ To	A	B	E
A	0	-	-
B	2, B	0, B	5, B
E	-	-	0

⇨

From \ To	A	B	C	D	E
A	0	2	1	3	-
B	2, B	0, B	3, A	5, A	5, B
E	-	-	-	-	0

Node C

Nbr	Cost
A	1
C	0
D	2

From \ To	A	C	D
A	0	-	-
C	1, C	0, C	2, C
D	-	-	0

⇨

From \ To	A	B	C	D
A	0	2	1	3
C	1, C	3, A	0, C	2, C
D	-	-	-	0

Node D

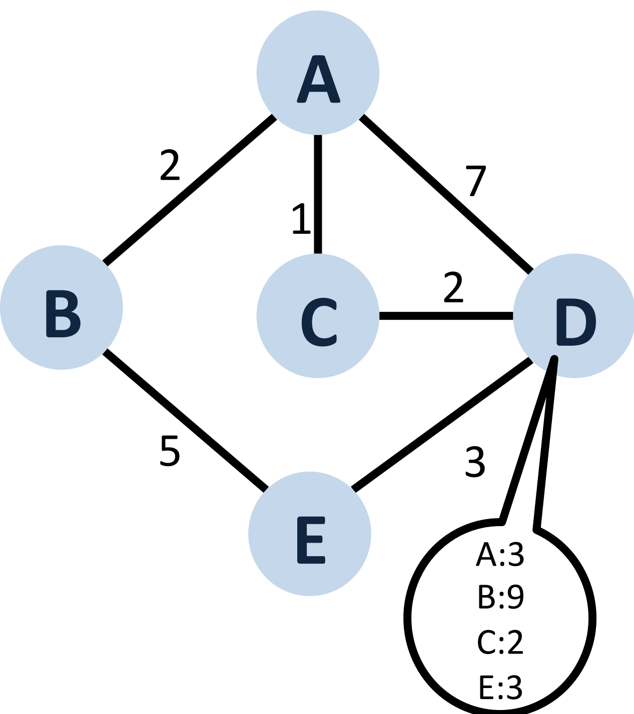
Nbr	Cost
A	7
C	2
D	0
E	3

From \ To	A	C	D	E
A	0	-	-	-
C	1	0	2	-
D	3, C	2, D	0, D	3, D
E	-	-	-	0

⇨

From \ To	A	B	C	D	E
A	0	2	1	3	-
C	1	-	0	2	-
D	3, C	9, A	2, D	0, D	3, D
E	-		-	-	0

Distance-Vector Routing



Node A

Nbr	*Cost*
A	0
B	2
C	1
D	7

To / From	*A*	*B*	*C*	*D*
A	0, A	2, A	1, A	3, C
B	-	0	-	-
C	1	-	0	2
D	-	-	-	0

⇨

To / From	*A*	*B*	*C*	*D*	*E*
A	0, A	2, A	1, A	3, C	10, D
B	-	0	-	-	-
C	1	-	0	2	-
D	3	9	2	0	3

Node C

Nbr	*Cost*
A	1
C	0
D	2

To / From	*A*	*B*	*C*	*D*
A	0	2	1	3
C	1, C	3, A	0, C	2, C
D	-	-	-	0

⇨

To / From	*A*	*B*	*C*	*D*	*E*
A	0	2	1	3	-
C	1, C	3, A	0, C	2, C	5, D
D	3	9	2	0	3

Node E

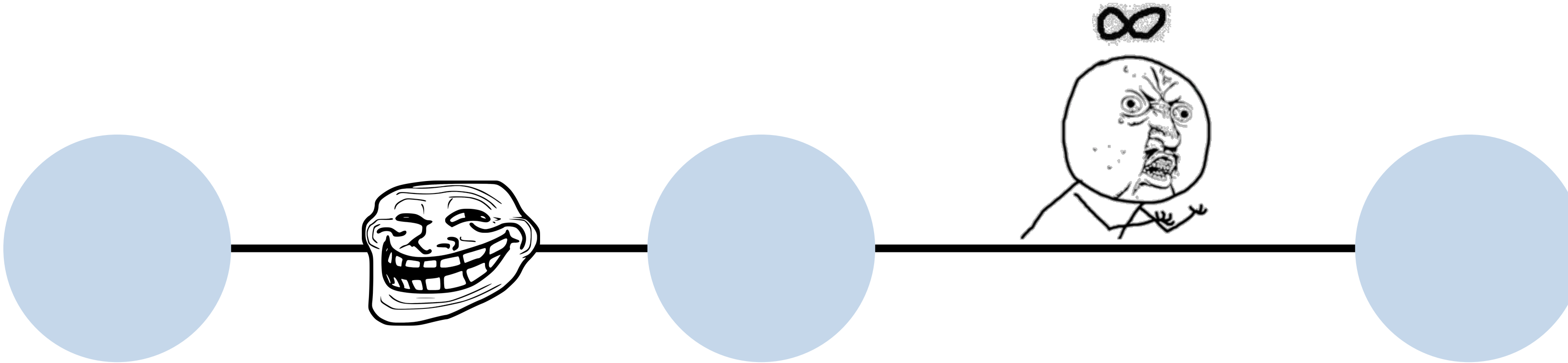
Nbr	*Cost*
B	5
D	3
E	0

To / From	*B*	*D*	*E*
B	0	-	-
D	-	0	-
E	5, E	3, E	0, E

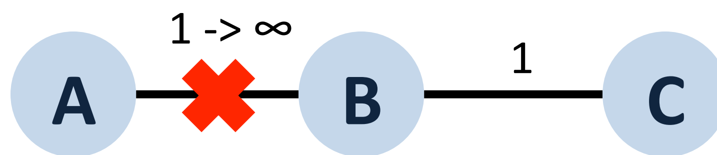
⇨

To / From	*A*	*B*	*C*	*D*	*E*
B	-	0	-	-	-
D	3	9	2	0	3
E	6, D	5, E	5, D	3, E	0, E

Count-To-Infinity Problem



Count-To-Infinity Problem



Node A

Nbr	Cost
A	0
B	1

From \ To	A	B	C
A	0, A	1, A	2, B
B	1	0	1

Node B

Nbr	Cost
A	1
B	0
C	1

From \ To	A	B	C
A	0	1	2
B	1, B	0, B	1, B
C	2	1	0

Nbr	Cost
A	∞
B	0
C	1

From \ To	A	B	C
B	3, C	0, B	1, B
C	2	1	0

From \ To	A	B	C
B	5, C	0, B	1, B
C	4	1	0

Node C

Nbr	Cost
B	1
C	0

From \ To	A	B	C
B	1	0	1
C	2, B	1, C	0, C

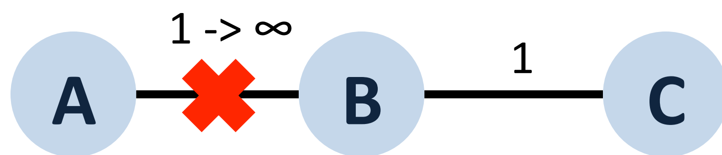
Nbr	Cost
B	1
C	0

From \ To	A	B	C
B	3	0	1
C	4, B	1, C	0, C

……

Time

Poison Reverse



Node A

Nbr	Cost
A	0
B	1

From \ To	A	B	C
A	0, A	1, A	2, B
B	1	0	1

Node B

Nbr	Cost
A	1
B	0
C	1

From \ To	A	B	C
A	0	1	∞
B	1, B	0, B	1, B
C	∞	1	0

Nbr	Cost
A	∞
B	0
C	1

From \ To	A	B	C
B	∞	0, B	1, B
C	∞	1	0

From \ To	A	B	C
B	∞	0, B	1, B
C	∞	1	0

Node C

Nbr	Cost
B	1
C	0

From \ To	A	B	C
B	1	0	1
C	2, B	1, C	0, C

Nbr	Cost
B	1
C	0

From \ To	A	B	C
B	∞	0	1
C	∞	1, C	0, C

Time