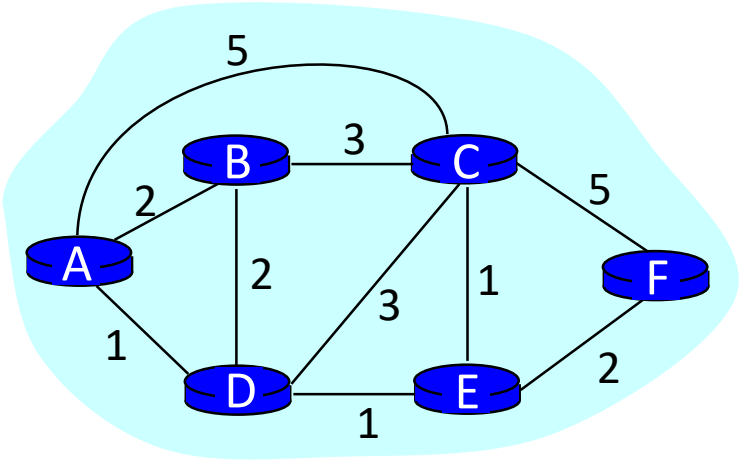
Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.



Graph abstraction for routing algorithms:

- graph *nodes* are routers
- graph *edges* are physical links
  - link cost: delay, $ cost, or congestion level

- "good" path:
  - typically means minimum cost path
    - Cost for *whom?*
  - other definitions possible

Routing Algorithm classification

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

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes
    - People with backhoes!

A Link-State Routing Algorithm

Dijkstra's algorithm

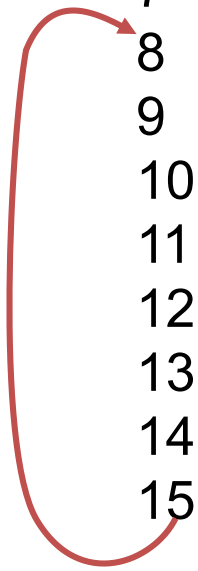
- net topology, link costs known to all nodes
  - accomplished via *link state broadcast*
  - all nodes have *same* info
- computes least cost paths from one node ("source") to all other nodes
  - gives routing table for that node
- **iterative:**
  - after ***k*** iterations, know least cost path to ***k*** dest.'s

Notation:

- 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 dest. V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

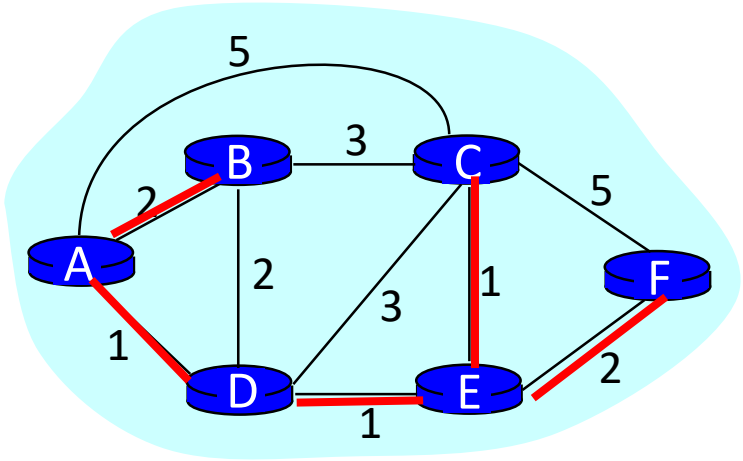
Dijsktra's Algorithm

```
1  Initialization:
2    N = {A}
3    for all nodes v
4      if v adjacent to A
5        then D(v) = c(A,v)
6        else D(v) = ∞
7
8  Loop
9     find w not in N such that D(w) is a minimum
10    add w to N
11    update D(v) for all v adjacent to w and not in N:
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N
```



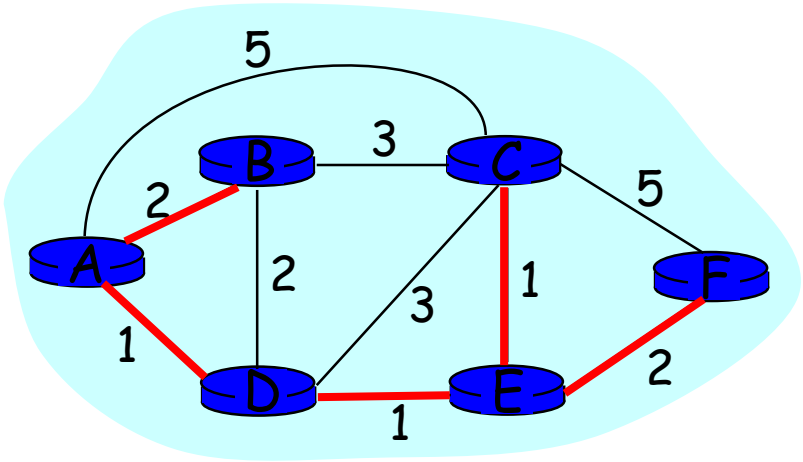
Dijkstra's algorithm: example

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



Dijkstra's algorithm: example

Step	start N	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	2,A	4,D		2,D	∞
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



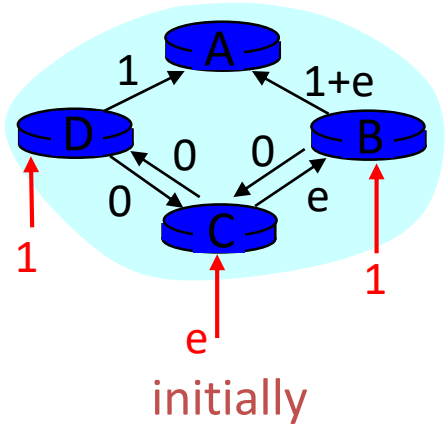
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

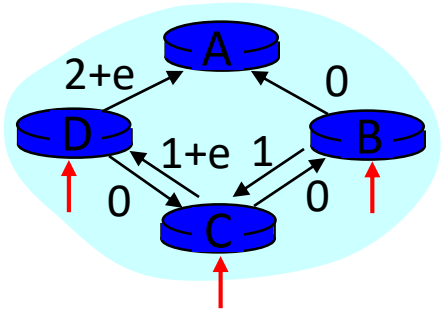
- each iteration: need to check all nodes, w, not in N
- $n*(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n\log n)$

Oscillations possible:

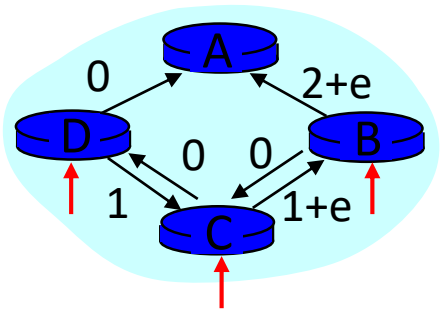
- e.g., link cost = amount of carried traffic



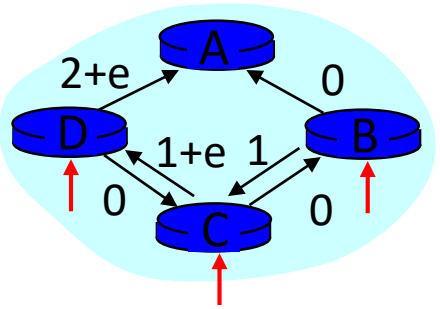
initially



… recompute routing



… recompute

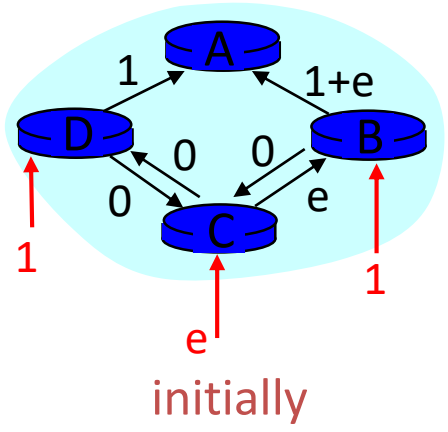


… recompute

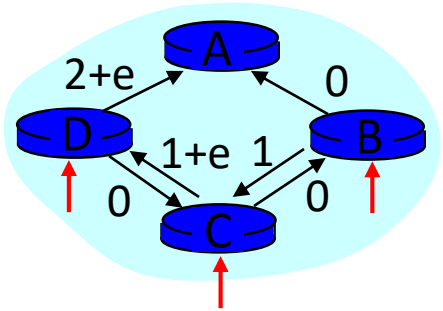
Dijkstra's algorithm, discussion

How can we fix it?

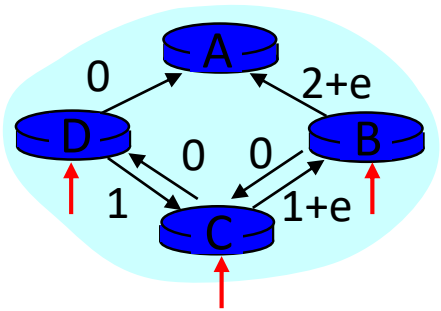
- Make sure all the re-computations don't happen at the same time
- Unfortunately routing updated tend to happen at the same time..
- Don't use traffic as the cost in Link State
- OSFP is a link state routing algo – what cost is used? (number of hops)



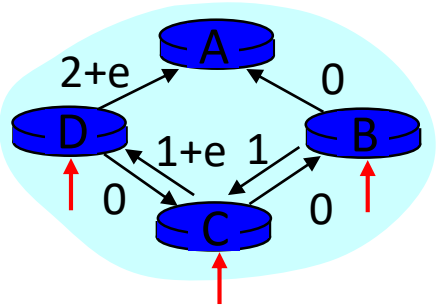
initially



… recompute routing



… recompute



… recompute