

Lecture 5

Routing

Routing in Packet Networks
Shortest Path Routing



Lecture 5

Routing

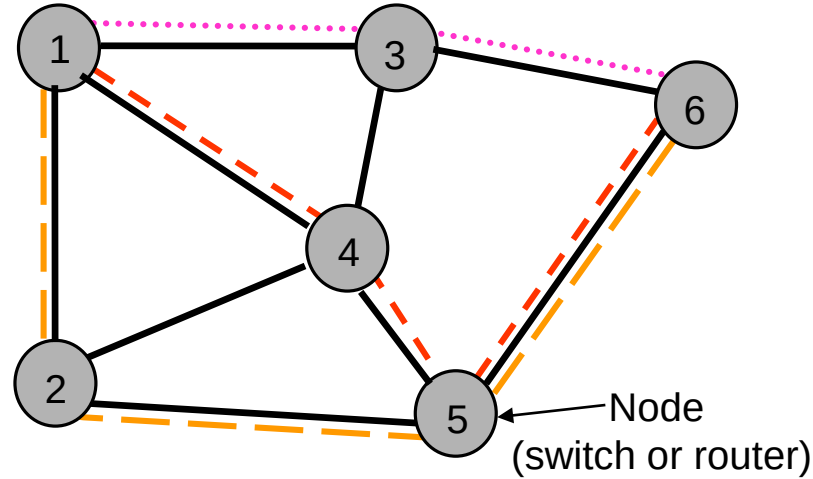
Routing in Packet Networks



Network Layer

- **Network Layer: the most complex layer**
 - Requires the coordinated actions of multiple, geographically distributed network elements (switches & routers)
 - Must be able to deal with very large scales
 - Billions of users (people & communicating devices)
- **Biggest Challenges**
 - Addressing: where should information be directed to?
 - Routing: what path should be used to get information there?

Routing in Packet Networks



- Three possible (loopfree) routes from 1 to 6:
 - 1-3-6, 1-4-5-6, 1-2-5-6
- Which is "best"?
 - Min delay? Min hop? Max bandwidth? Min cost? Max reliability?

Centralized vs Distributed Routing

- **Centralized Routing**
 - All routes determined by a central node
 - All state information sent to central node
 - Problems adapting to frequent topology changes
 - Does not scale
- **Distributed Routing**
 - Routes determined by routers using distributed algorithm
 - State information exchanged by routers
 - Adapts to topology and other changes
 - Better scalability

Specialized Routing

- **Flooding**
 - Useful in starting up network
 - Useful in propagating information to all nodes

Flooding (1)

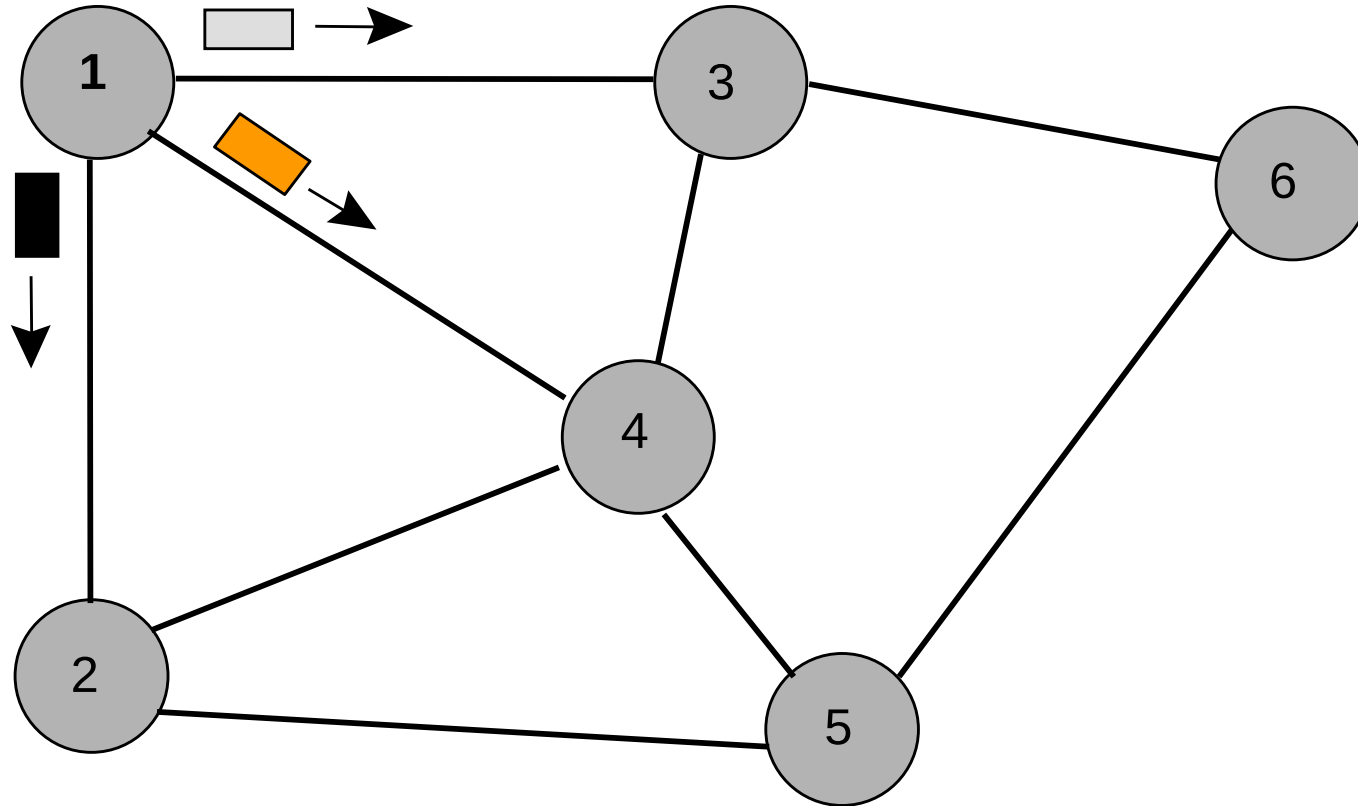
Send a packet to all nodes in a network

- **No routing tables available**
- **Need to broadcast packet to all nodes (e.g. to propagate link state information)**

Approach

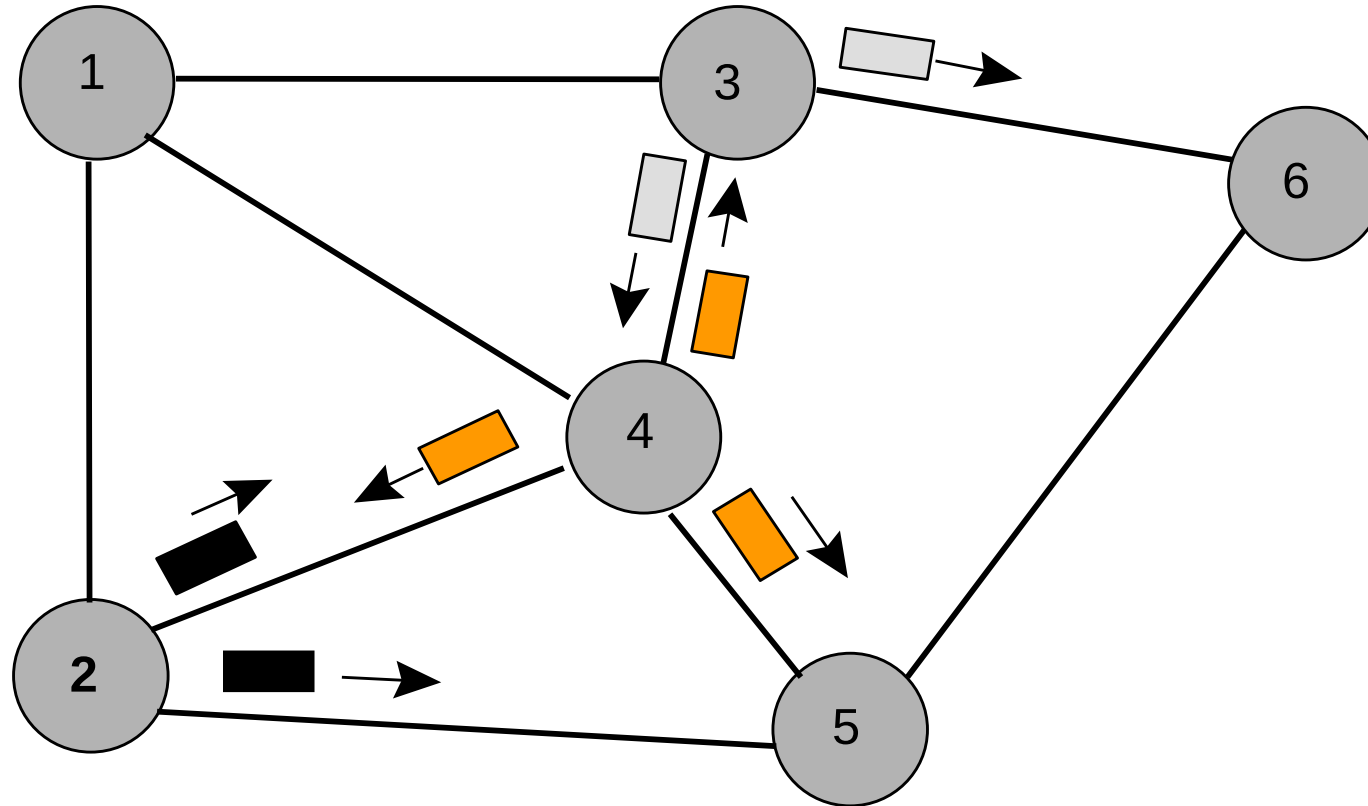
- **Send packet on all ports except one where it arrived**
- **Exponential growth in packet transmissions**

Flooding (2)



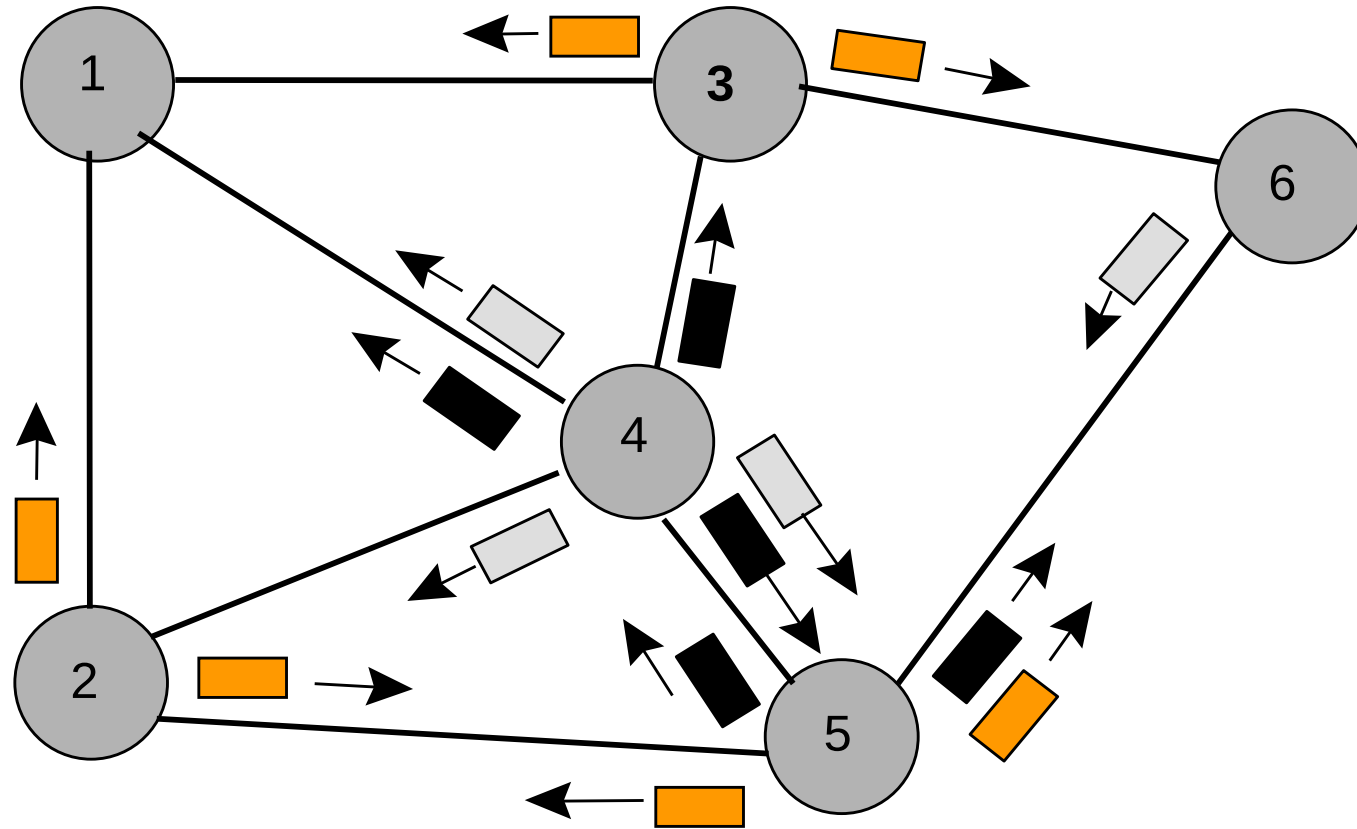
Flooding is initiated from Node 1: Hop 1 transmissions

Flooding (3)



Flooding is initiated from Node 1: Hop 2 transmissions

Flooding (4)



Flooding is initiated from Node 1: Hop 3 transmissions

Limited Flooding

- Time-to-Live field in each packet limits number of hops to certain diameter
- Each switch adds its ID before flooding; discards repeats
- Source puts sequence number in each packet; switches records source address and sequence number and discards repeats

Lecture 5

Routing

Routing in Packet Networks
Shortest Path Routing



Lecture 5

Routing

Shortest Path Routing



Shortest Paths & Routing

- Many possible paths connect any given source and to any given destination
- Routing involves the selection of the path to be used to accomplish a given transfer
- Typically it is possible to attach a cost or distance to a link connecting two nodes
- Routing can then be posed as a shortest path problem

Routing Metrics

Means for measuring desirability of a path

- Path Length = sum of costs or distances
- Possible metrics
 - Hop count: rough measure of resources used
 - Reliability: link availability; BER
 - Delay: sum of delays along path; complex & dynamic
 - Bandwidth: “available capacity” in a path
 - Load: Link & router utilization along path
 - Cost: \$\$\$

Shortest Path Approaches

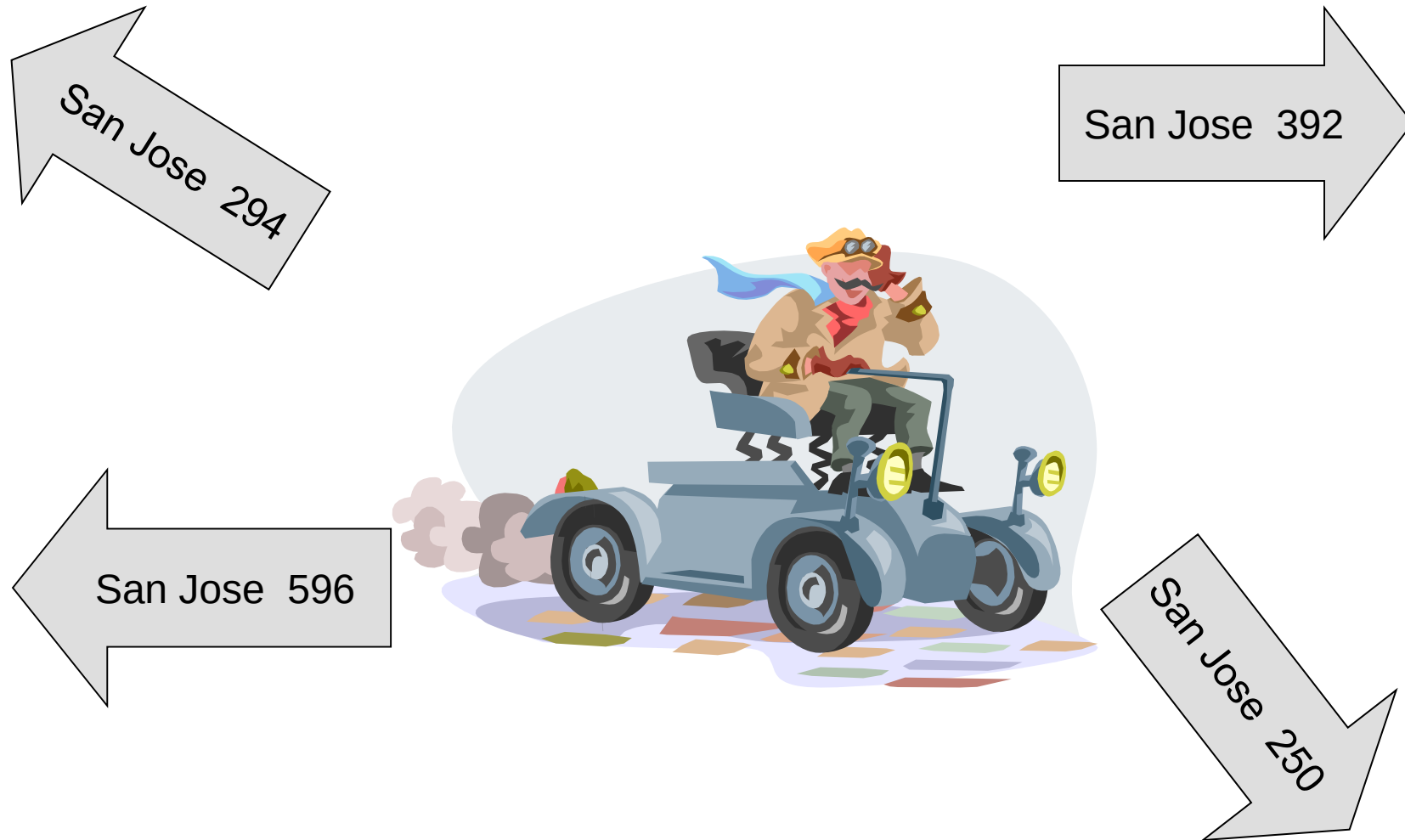
Distance Vector Protocols

- Neighbors exchange list of distances to destinations
- Best next-hop determined for each destination
- Ford-Fulkerson (distributed) shortest path algorithm

Link State Protocols

- Link state information flooded to all routers
- Routers have complete topology information
- Shortest path (& hence next hop) calculated
- Dijkstra (centralized) shortest path algorithm

Distance Vector **Do you know the way to San Jose?**



Distance Vector

Local Signpost

- Direction
- Distance

Routing Table

For each destination list:

- Next Node
- Distance

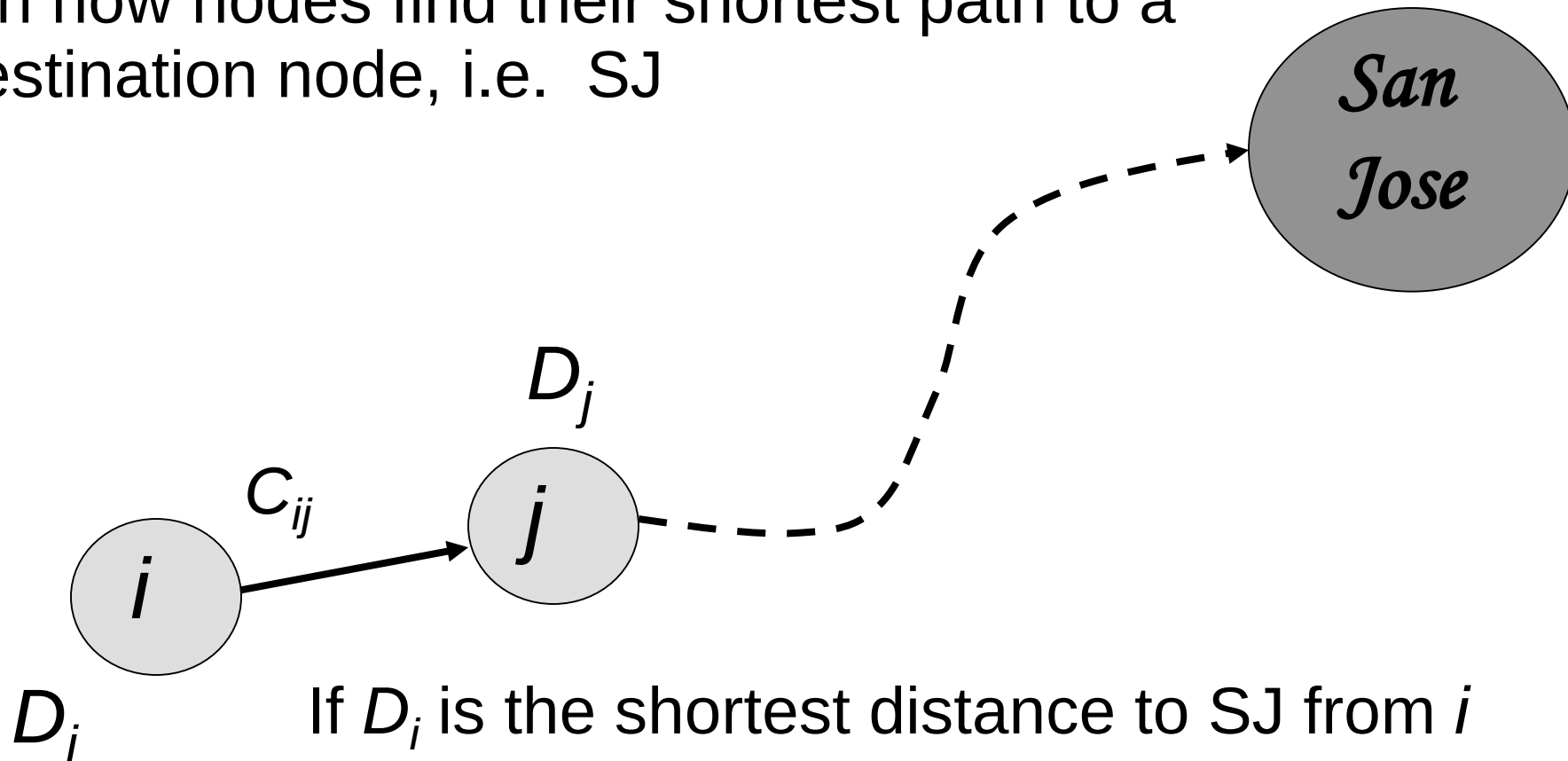
Table Synthesis

- Neighbors exchange table entries
- Determine current best next hop
- Inform neighbors
 - Periodically
 - After changes

dest	next	dist

Shortest Path to SJ

Focus on how nodes find their shortest path to a given destination node, i.e. SJ

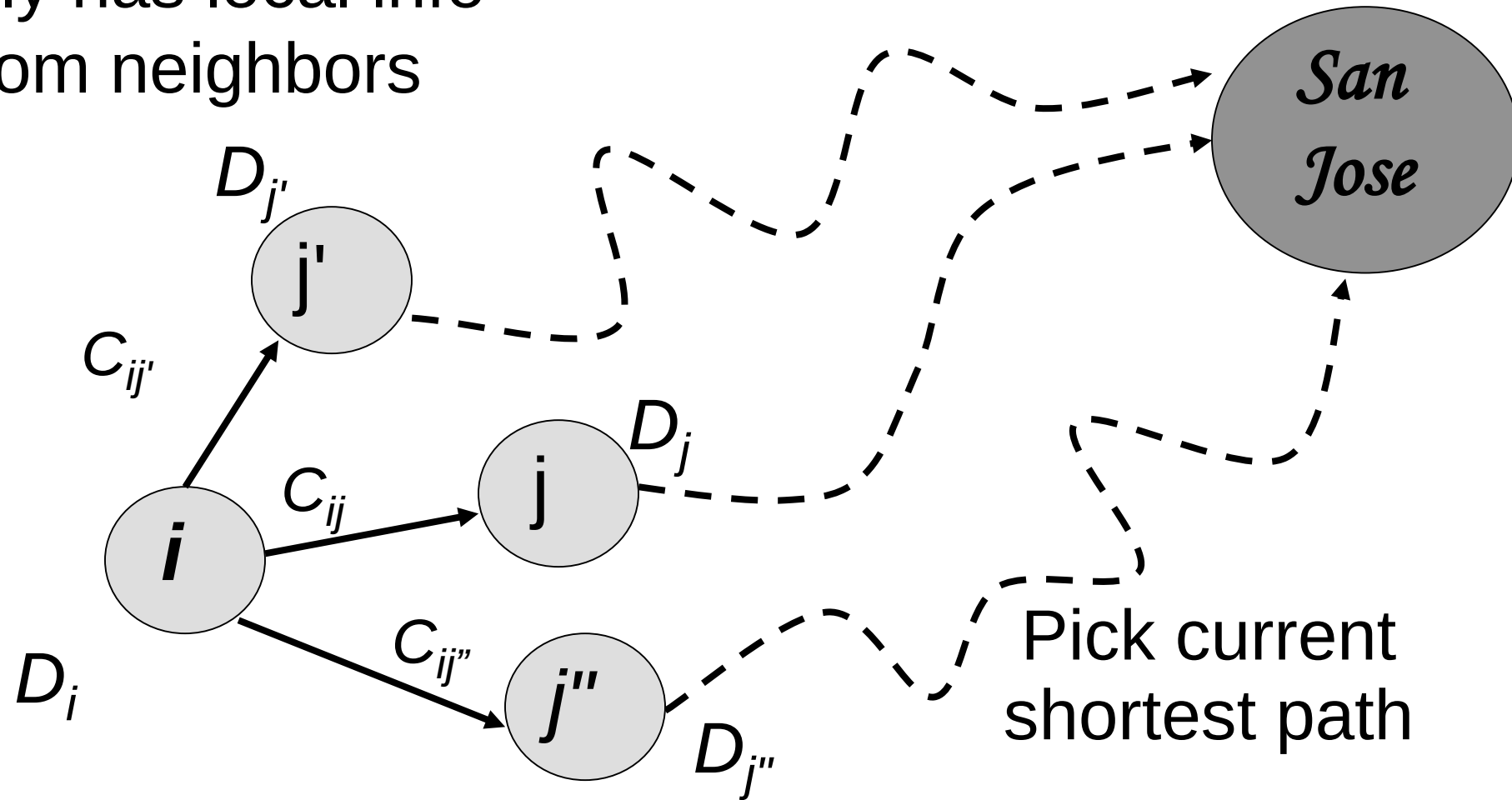


If D_i is the shortest distance to SJ from i and if j is a neighbor on the shortest path, then

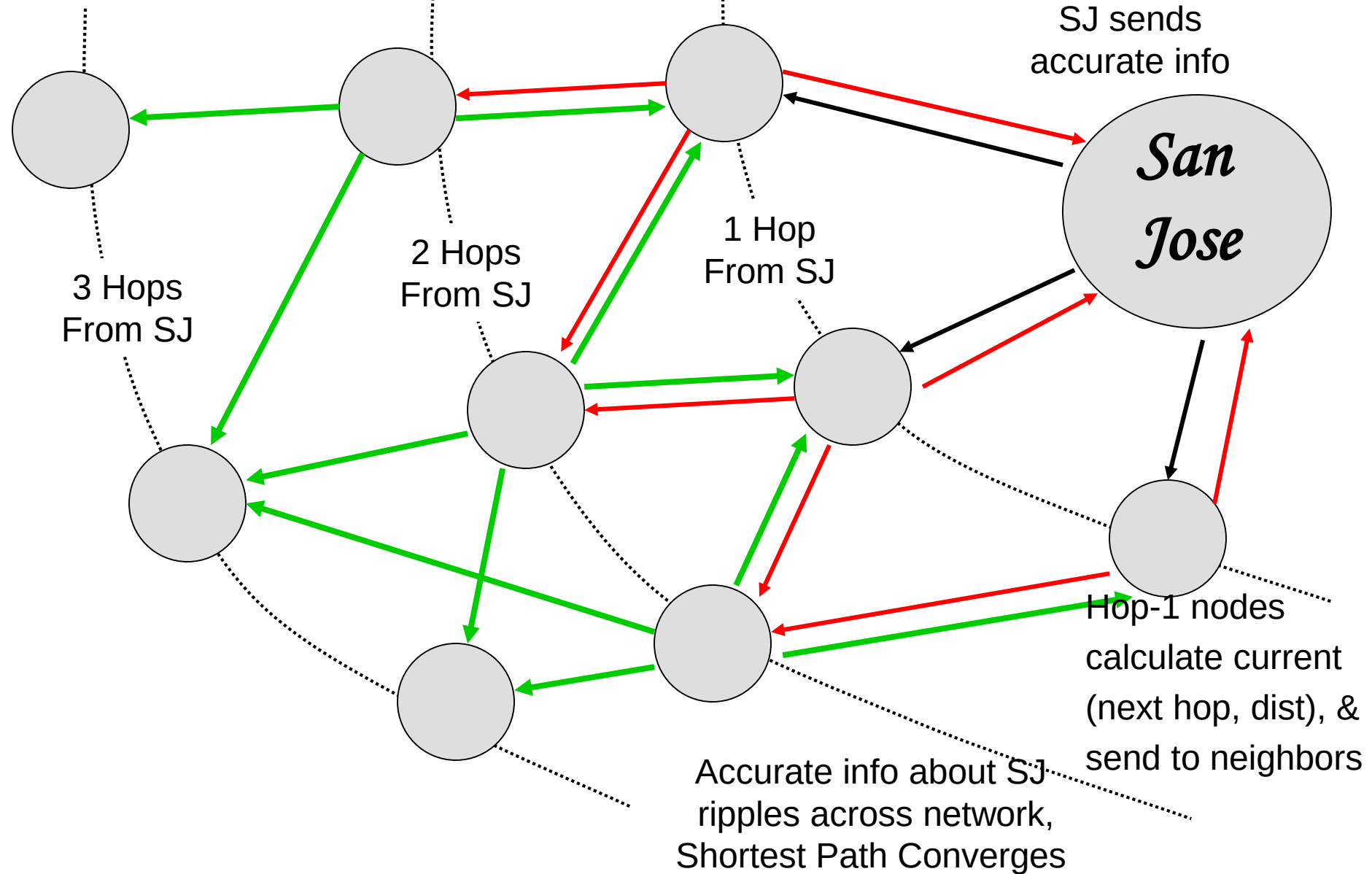
$$D_i = C_{ij} + D_j$$

But we don't know the shortest paths

i only has local info
from neighbors



Why Distance Vector Works



Bellman–Ford Algorithm

- *Consider computations for one destination d*
- *Initialization*
 - Each node table has 1 row for destination d
 - Distance of node d to itself is zero: $D_d=0$
 - Distance of other node j to d is infinite: $D_j=\infty$, for $j \neq d$
 - Next hop node $n_j = -1$ to indicate not yet defined for $j \neq d$
- *Send Step*
 - Send new distance vector to immediate neighbors across local link
- *Receive Step*
 - At node j , find the next hop that gives the minimum distance to d ,
 - $\text{Min}_j \{ C_{ij} + D_j \}$
 - Replace old $(n_j, D_j(d))$ by new $(n_j^*, D_j^*(d))$ if new next node or distance
 - Go to send step

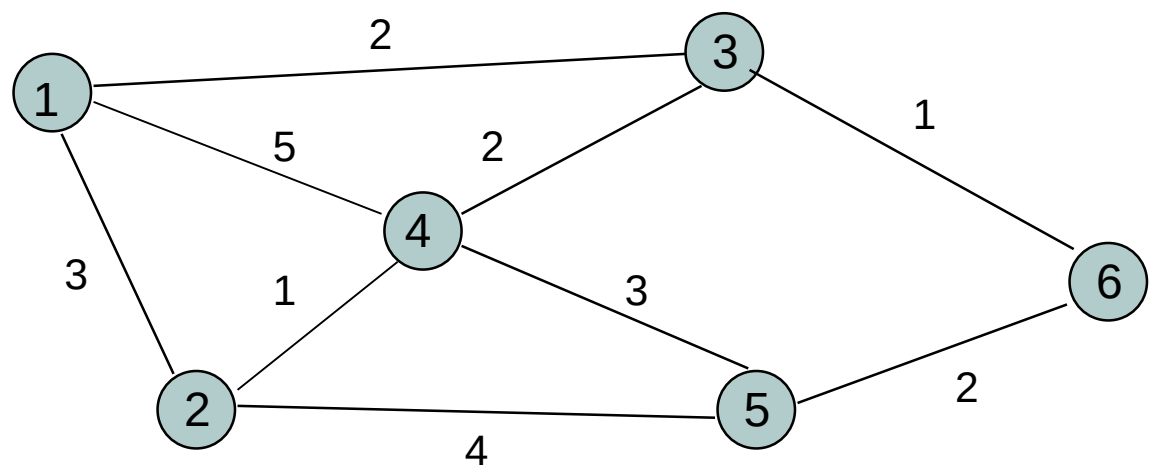
Bellman–Ford Algorithm

- *Now consider parallel computations for all destinations d*
- *Initialization*
 - Each node has 1 row for each destination d
 - Distance of node d to itself is zero: $D_d(d)=0$
 - Distance of other node j to d is infinite: $D_j(d)=\infty$, for $j \neq d$
 - Next node $n_j = -1$ since not yet defined
- *Send Step*
 - Send new distance vector to immediate neighbors across local link
- *Receive Step*
 - For each destination d , find the next hop that gives the minimum distance to d ,
 - $\text{Min}_j \{ C_{ij} + D_j(d) \}$
 - Replace old $(n_j, D_j(d))$ by new $(n_j^*, D_j^*(d))$ if new next node or distance found
 - Go to send step

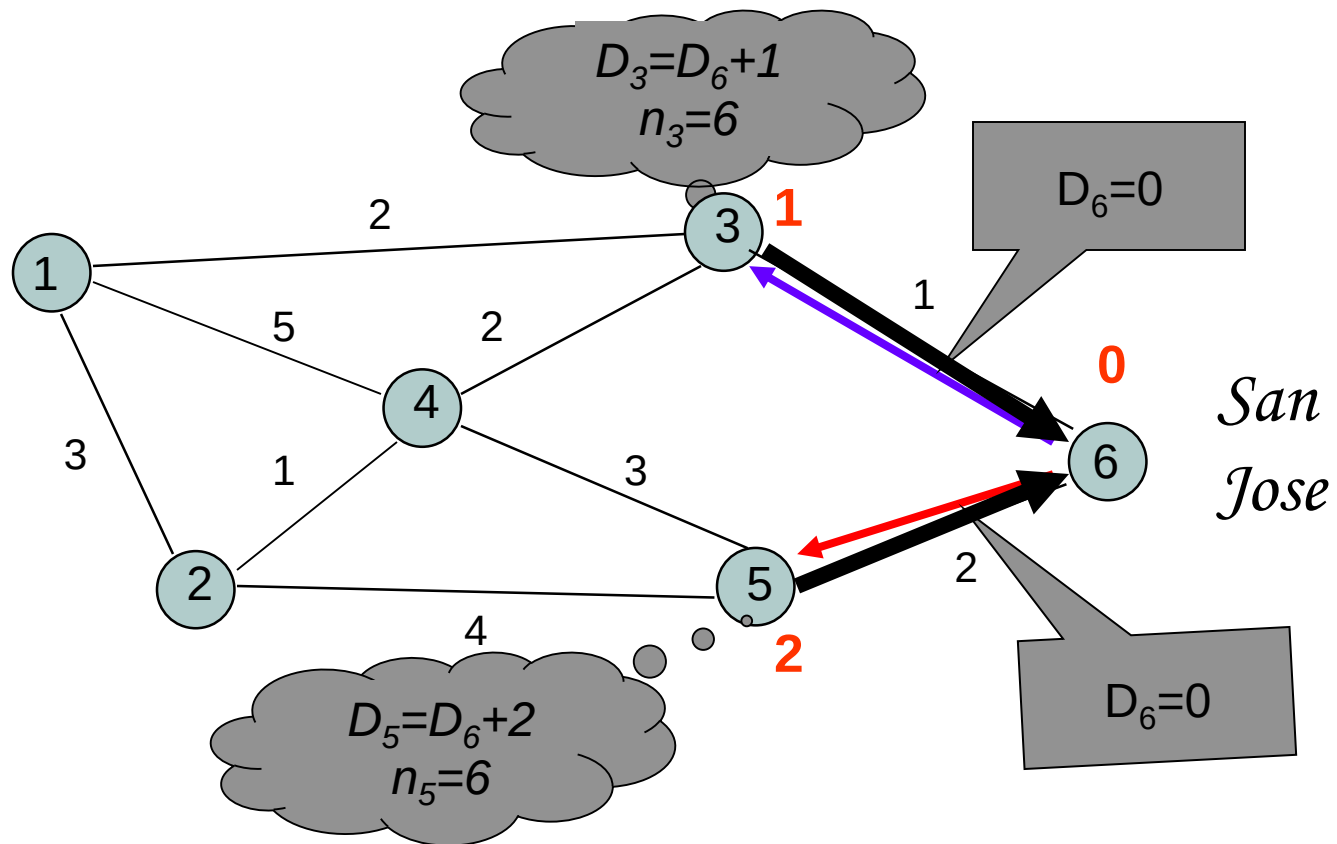
Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$
1					
2					
3					

Table entry
@ node 1
for dest SJ

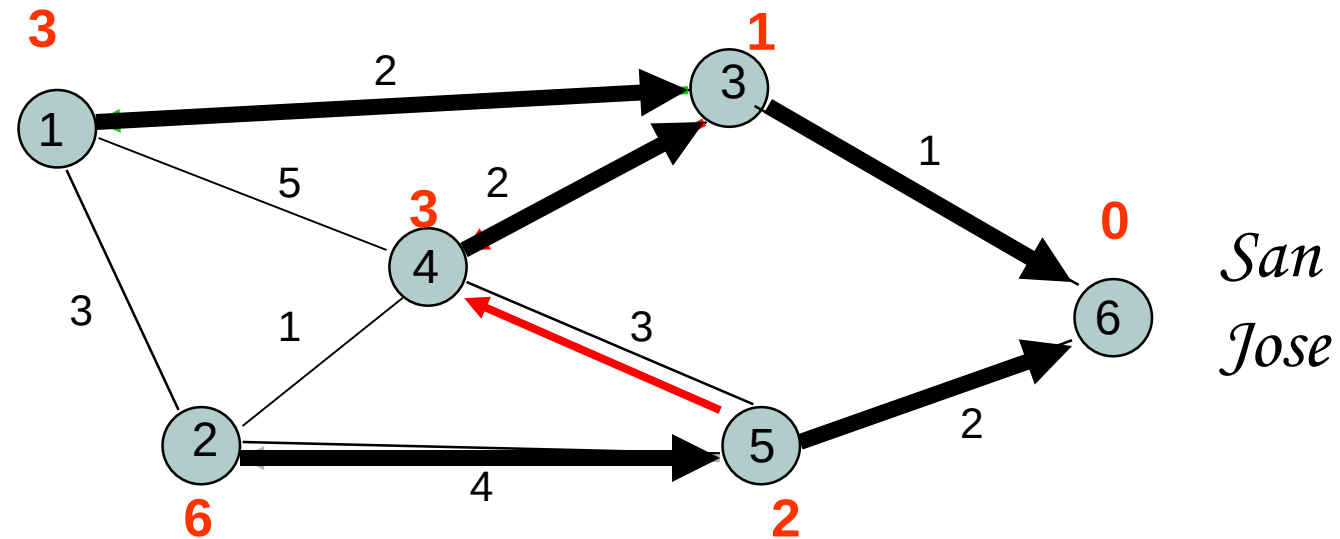
Table entry
@ node 3
for dest SJ



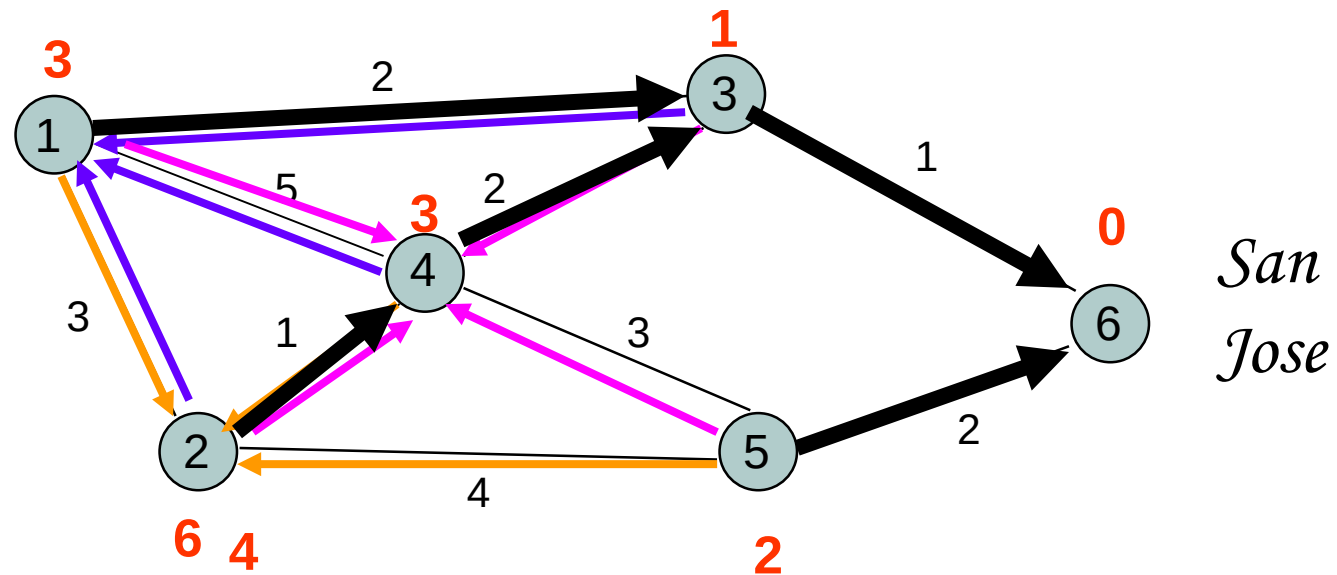
Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$
1	$(-1, \infty)$	$(-1, \infty)$	(6,1)	$(-1, \infty)$	(6,2)
2					
3					



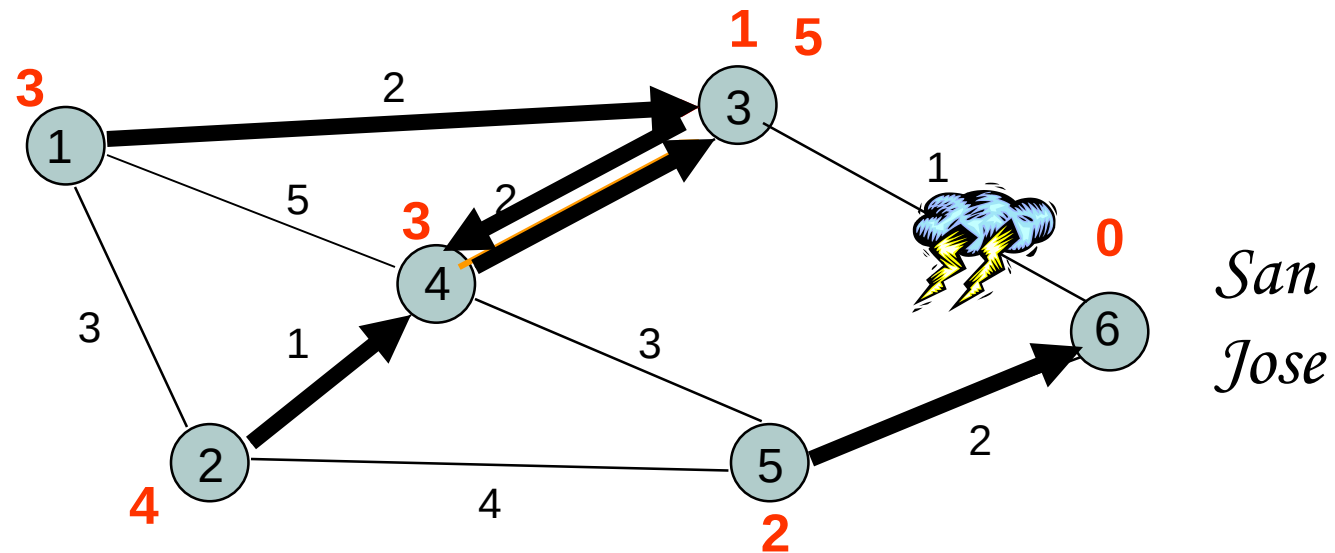
Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$
1	$(-1, \infty)$	$(-1, \infty)$	$(6, 1)$	$(-1, \infty)$	$(6, 2)$
2	$(3, 3)$	$(5, 6)$	$(6, 1)$	$(3, 3)$	$(6, 2)$
3					



Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$
1	$(-1, \infty)$	$(-1, \infty)$	$(6, 1)$	$(-1, \infty)$	$(6, 2)$
2	$(3, 3)$	$(5, 6)$	$(6, 1)$	$(3, 3)$	$(6, 2)$
3	$(3, 3)$	$(4, 4)$	$(6, 1)$	$(3, 3)$	$(6, 2)$

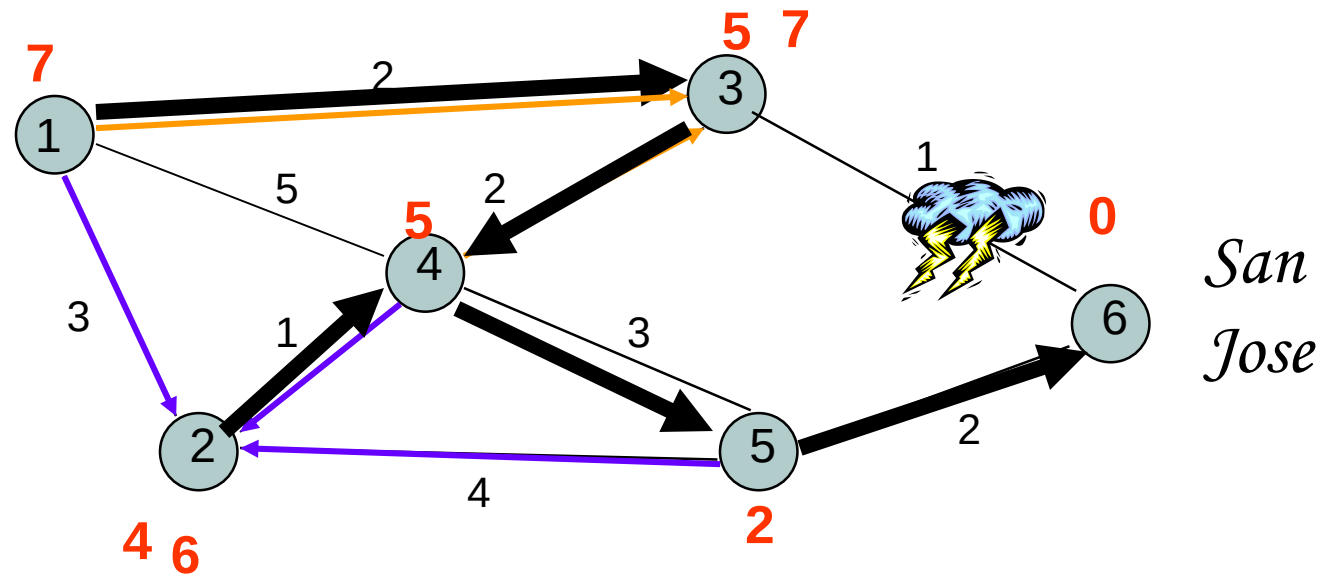


Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2					
3					



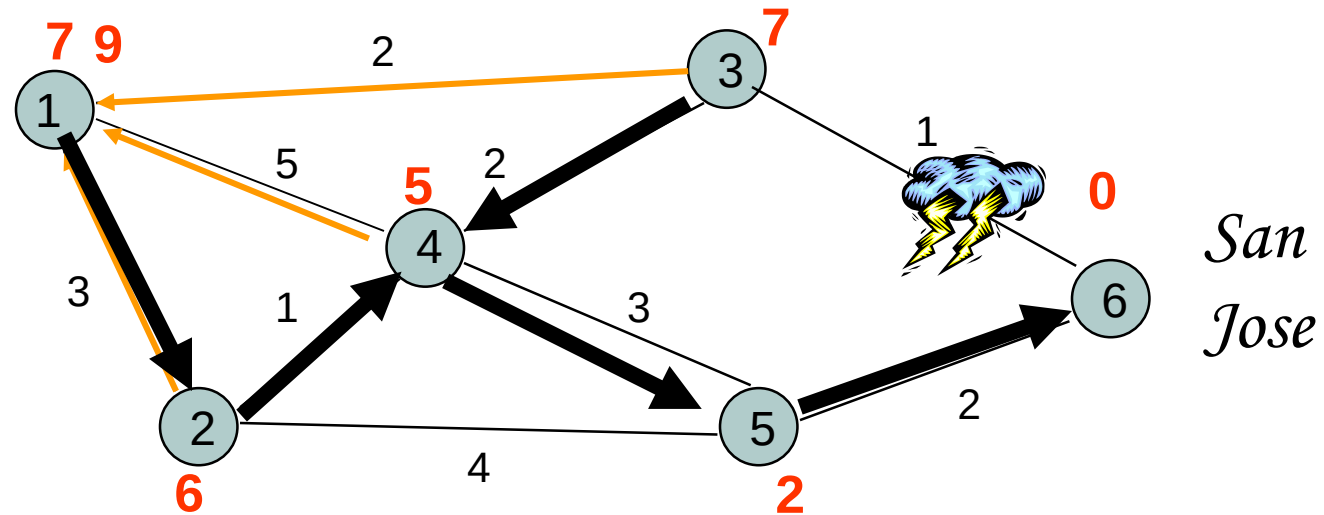
Network disconnected; Loop created between nodes 3 and 4

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2	(3,7)	(4,4)	(4, 5)	(5,5)	(6,2)
3	(3,7)	(4,6)	(4, 7)	(5,5)	(6,2)



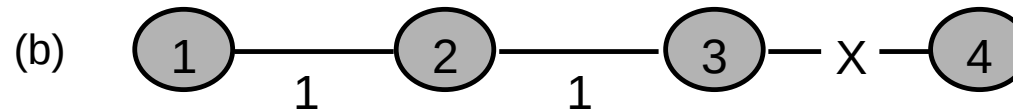
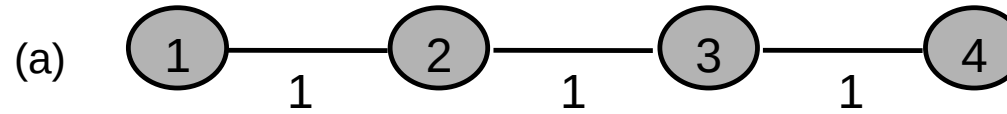
Node 2 could have chosen 5 as next node because of tie

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2	(3,7)	(4,4)	(4, 5)	(2,5)	(6,2)
3	(3,7)	(4,6)	(4, 7)	(5,5)	(6,2)
4	(2,9)	(4,6)	(4, 7)	(5,5)	(6,2)



Node 1 could have chose 3 as next node because of tie

Counting to Infinity Problem



Nodes believe best path is through each other

(Destination is node 4)

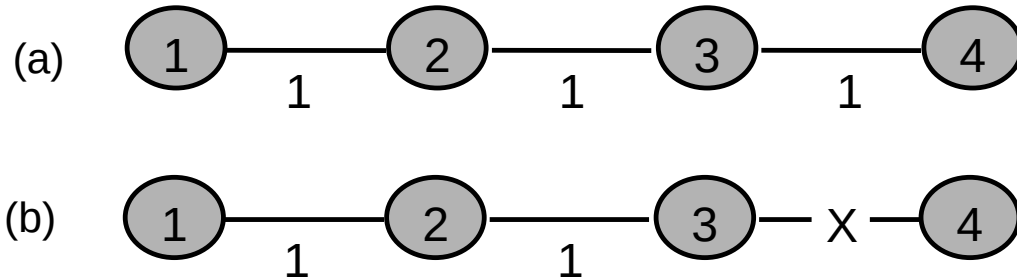
Update	Node 1	Node 2	Node 3
Before break	(2,3)	(3,2)	(4, 1)
After break	(2,3)	(3,2)	(2,3)
1	(2,3)	(3,4)	(2,3)
2	(2,5)	(3,4)	(2,5)
3	(2,5)	(3,6)	(2,5)
4	(2,7)	(3,6)	(2,7)
5	(2,7)	(3,8)	(2,7)
...	...	...	...

Problem: Bad News Travels Slowly

Remedies

- **Split Horizon**
 - Do not report route to a destination to the neighbor from which route was learned
- **Poisoned Reverse**
 - Report route to a destination to the neighbor from which route was learned, but with infinite distance
 - Breaks erroneous direct loops immediately
 - Does not work on some indirect loops

Split Horizon with Poison Reverse



Nodes believe best path is through each other

Update	Node 1	Node 2	Node 3	
Before break	(2, 3)	(3, 2)	(4, 1)	
After break	(2, 3)	(3, 2)	$(-1, \infty)$	Node 2 advertizes its route to 4 to node 3 as having distance infinity; node 3 finds there is no route to 4
1	(2, 3)	$(-1, \infty)$	$(-1, \infty)$	Node 1 advertizes its route to 4 to node 2 as having distance infinity; node 2 finds there is no route to 4
2	$(-1, \infty)$	$(-1, \infty)$	$(-1, \infty)$	Node 1 finds there is no route to 4

Link-State Algorithm

- Basic idea: two step procedure
 - Each source node gets a map of all nodes and link metrics (link state) of the entire network
 - Find the shortest path on the map from the source node to all destination nodes
- Broadcast of link-state information
 - Every node i in the network broadcasts to every other node in the network:
 - ID's of its neighbors: $\mathcal{N}_i$ =set of neighbors of i
 - Distances to its neighbors: $\{C_{ij} \mid j \in \mathcal{N}_i\}$
 - Flooding is a popular method of broadcasting packets

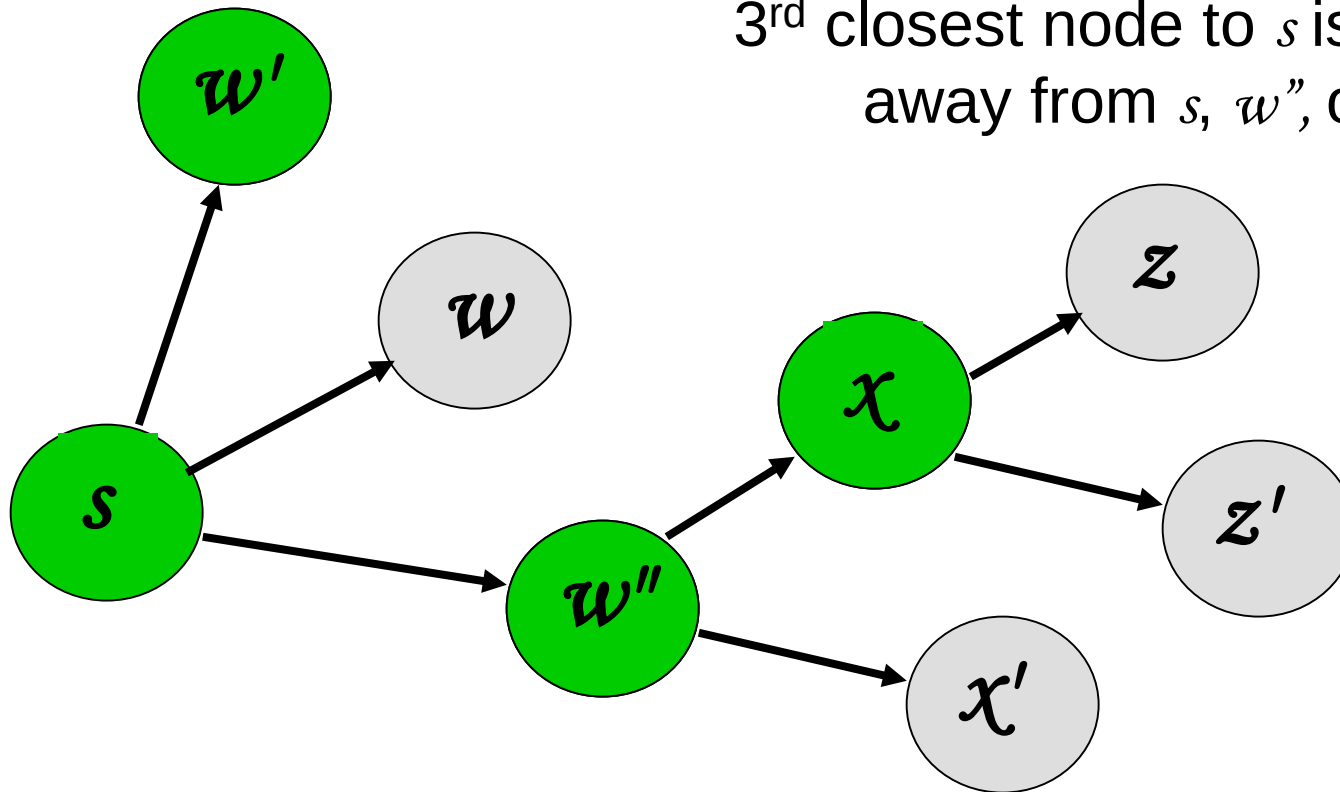
Dijkstra Algorithm: Finding shortest paths in order

Find shortest paths from source s to all other destinations

Closest node to s is 1 hop away

2nd closest node to s is 1 hop away from s or w''

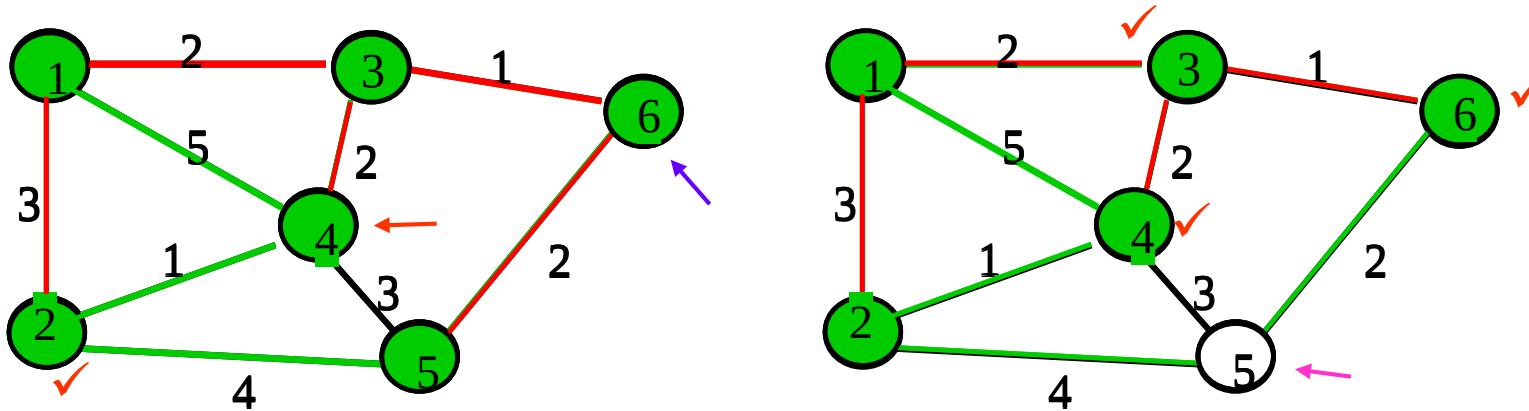
3rd closest node to s is 1 hop away from s , w'' , or χ



Dijkstra's algorithm

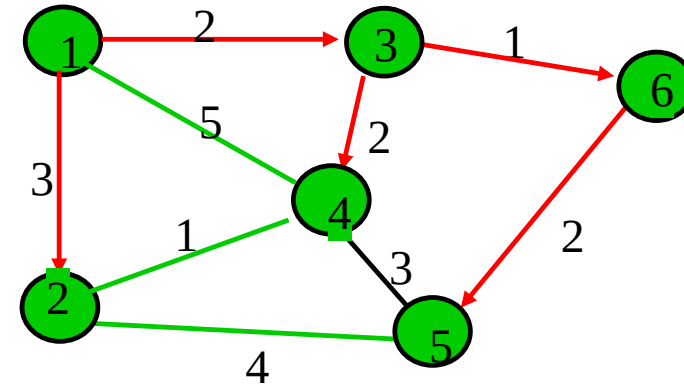
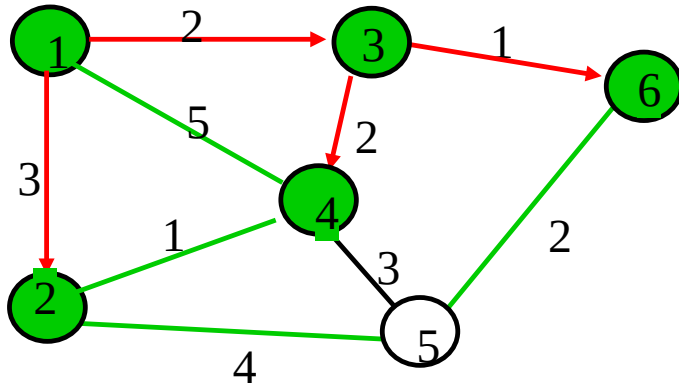
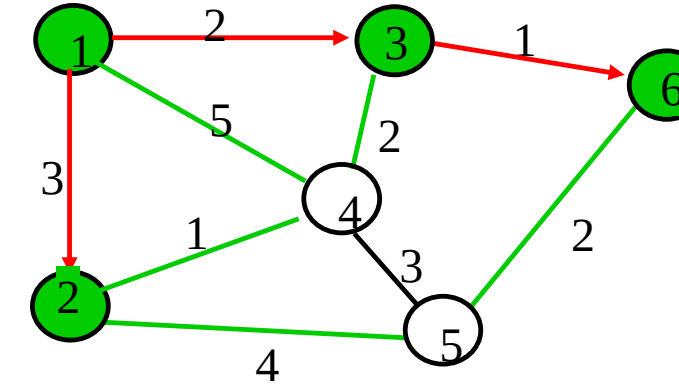
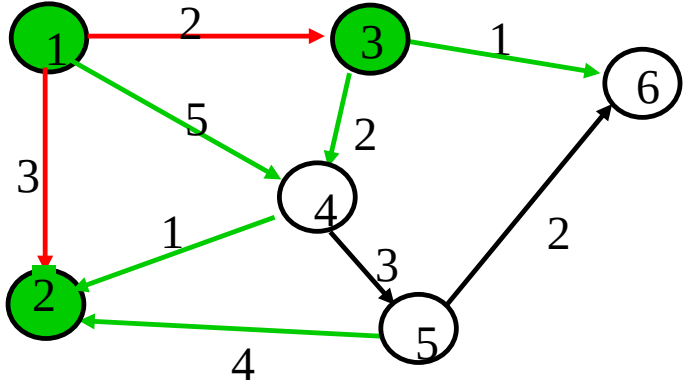
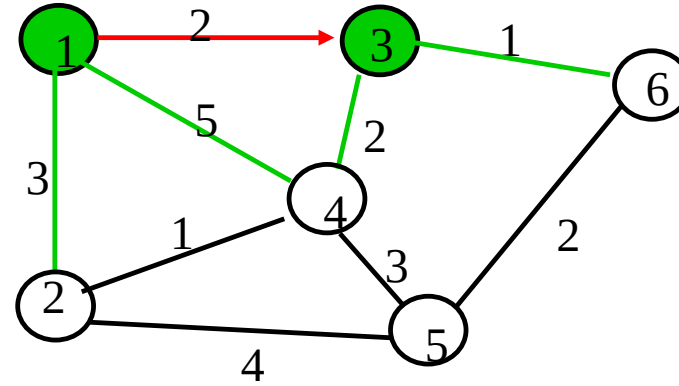
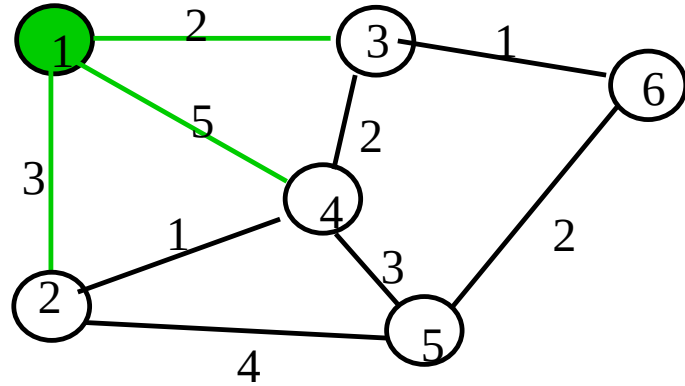
- N : set of nodes for which shortest path already found
- Initialization: (*Start with source node s*)
 - $N = \{s\}$, $D_s = 0$, " s is distance zero from itself"
 - $D_j = C_{sj}$ for all $j \neq s$, distances of directly-connected neighbors
- Step A: (*Find next closest node i*)
 - Find $i \notin N$ such that
 - $D_i = \min D_j$ for $j \notin N$
 - Add i to N
 - If N contains all the nodes, stop
- Step B: (*update minimum costs*)
 - For each node $j \notin N$
 - $D_j = \min (D_j, D_i + C_{ij})$  *Minimum distance from s to j through node i in N*
 - Go to Step A

Execution of Dijkstra's algorithm



Iteration	N	D_2	D_3	D_4	D_5	D_6
Initial	{1}	3	2 ✓	5	∞	∞
1	{1,3}	3 ✓	2	4	∞	3
2	{1,2,3}	3	2	4	7	3 ✓
3	{1,2,3,6}	3	2	4 ✓	5	3
4	{1,2,3,4,6}	3	2	4	5 ✓	3
5	{1,2,3,4,5,6}	3	2	4	5	3

Shortest Paths in Dijkstra's Algorithm



Routing table at node 1

Destination	Next node	Cost
2	2	3
3	3	2
4	3	4
5	3	5
6	3	3

Reaction to Failure

- If a link fails,
 - Router sets link distance to infinity & floods the network with an update packet
 - All routers immediately update their link database & recalculate their shortest paths
 - Recovery very quick
- But watch out for old update messages
 - Add time stamp or sequence # to each update message
 - Check whether each received update message is new
 - If new, add it to database and broadcast
 - If older, send update message on arriving link