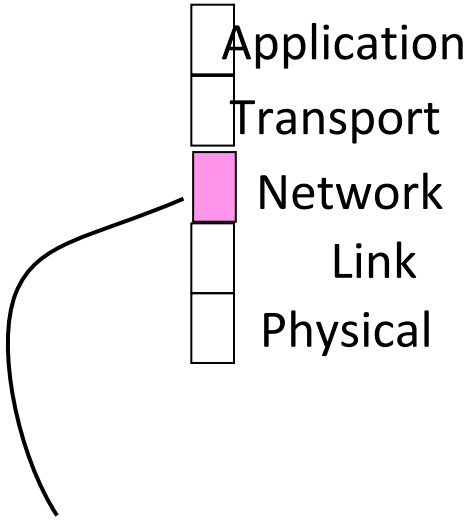
SRM UNIVERSITY AP
Amaravati



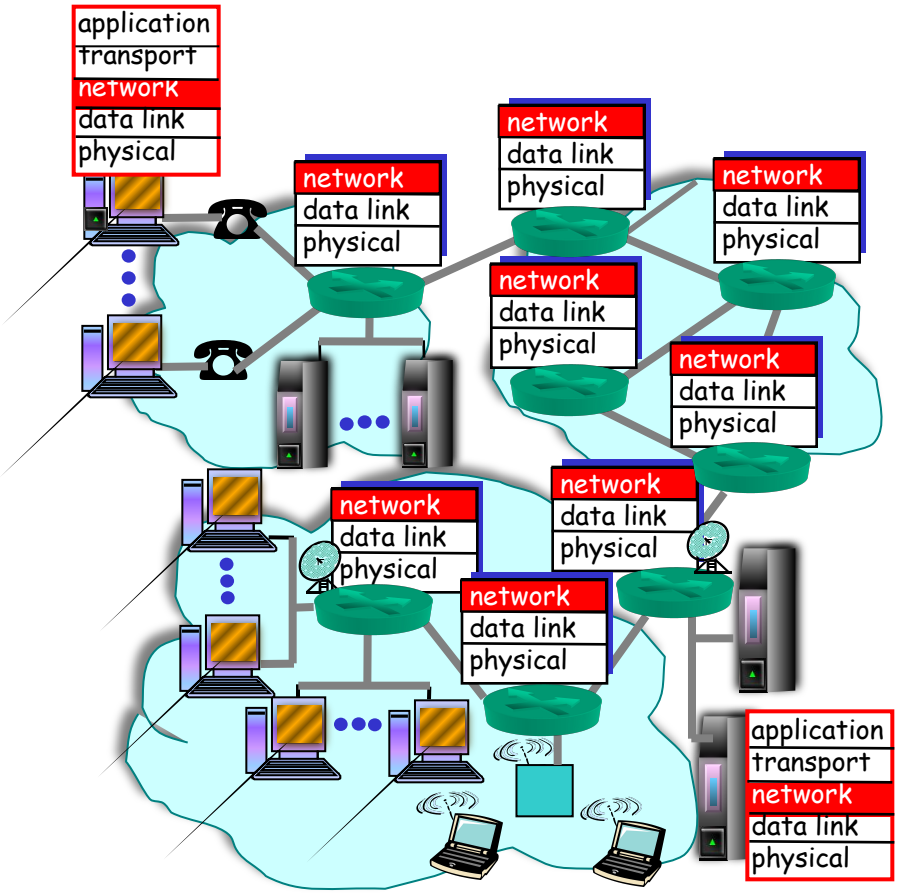
Computer Networks

Network Layer

Dr. Satish Anamalamudi,
Department of CSE
SRM University-AP

Overview

- Functions:
  a) Routing issues
     a) determine "good" path (sequence of routers) thru network from source to dest.
  – Congestion (Not Contention!)
     - More packets enter an area than can be processed
  – Internetworking
     - connecting different network technologies together



- Network layer protocols in *every* host, router

Network Layer Design Issues

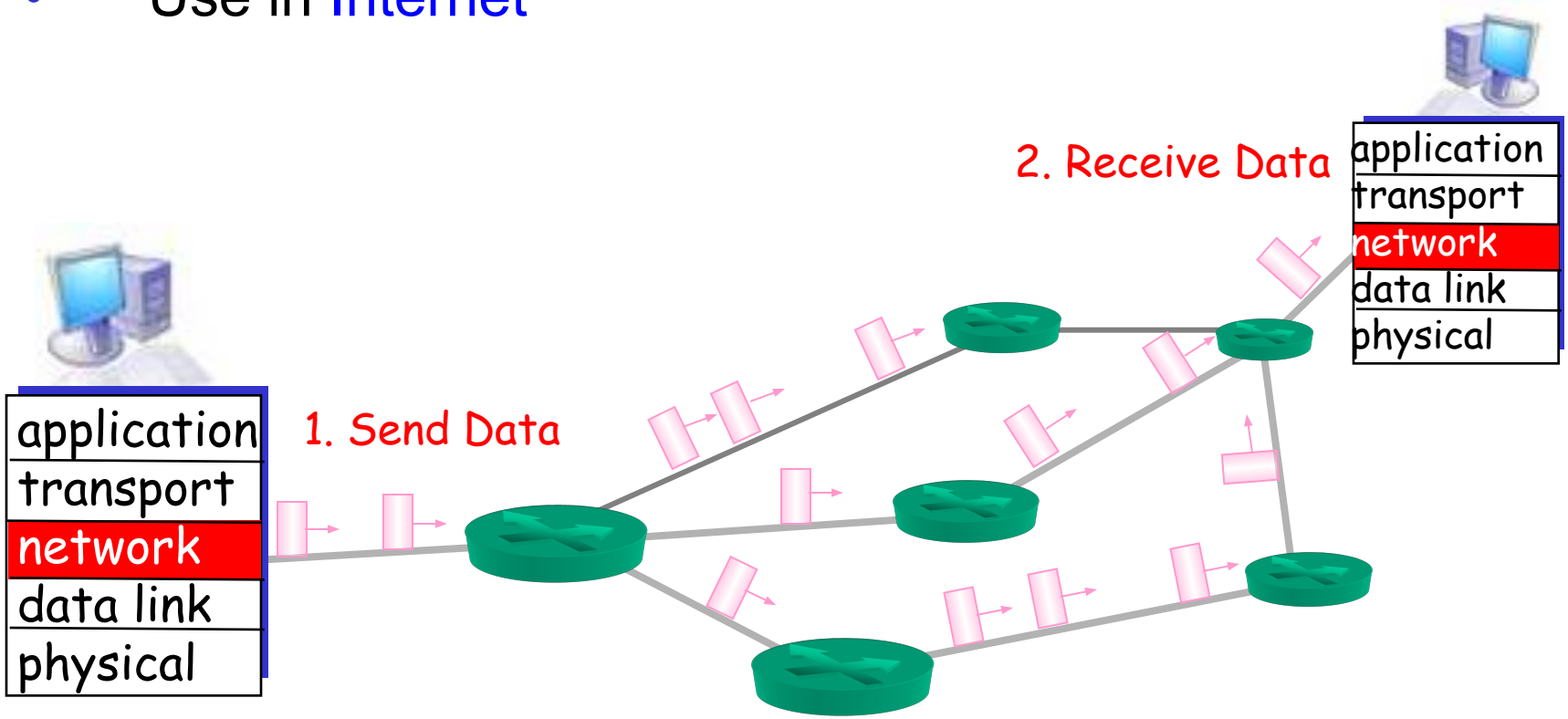
- Services Provided to the Transport Layer
  a) The network layer should shield the transport layer from having to know details of the underlying subnet

- Network Layer Services can:
  a) Connection-Oriented: Provides Virtual Circuit (VC) subnet,
     a) source-to-destination path behaves much like telephone circuit, Avoids choosing a new route for each packet.
     b) A virtual circuit remembers how to send a packet from source to destination.

  b) Connection-less: Provides Datagram subnet,
     a) Each packet sent is routed independently of its predecessors

Connectionless: Datagram (1)

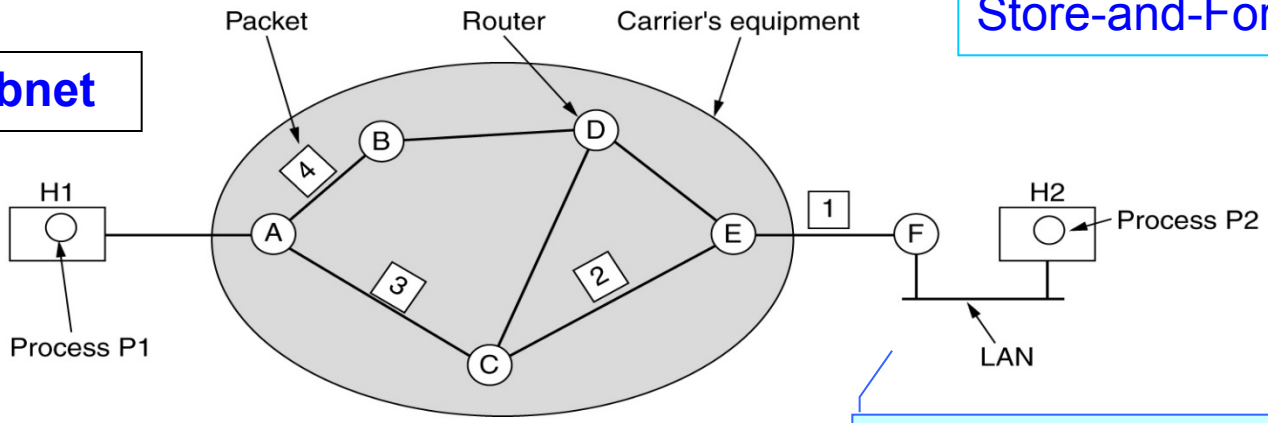
- No call setup at network layer
- Packets forwarded using destination host address
  a) packets between same source-dest. pair may take different paths
- Use in Internet



Connectionless: Datagram (2)

Store-and-Forward packet

Subnet



Routing tables

A's table

initially		later	
A	–	A	–
B	B	B	B
C	C	C	C
D	B	D	B
E	C	E	B
F	C	F	B

Dest. Line

C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

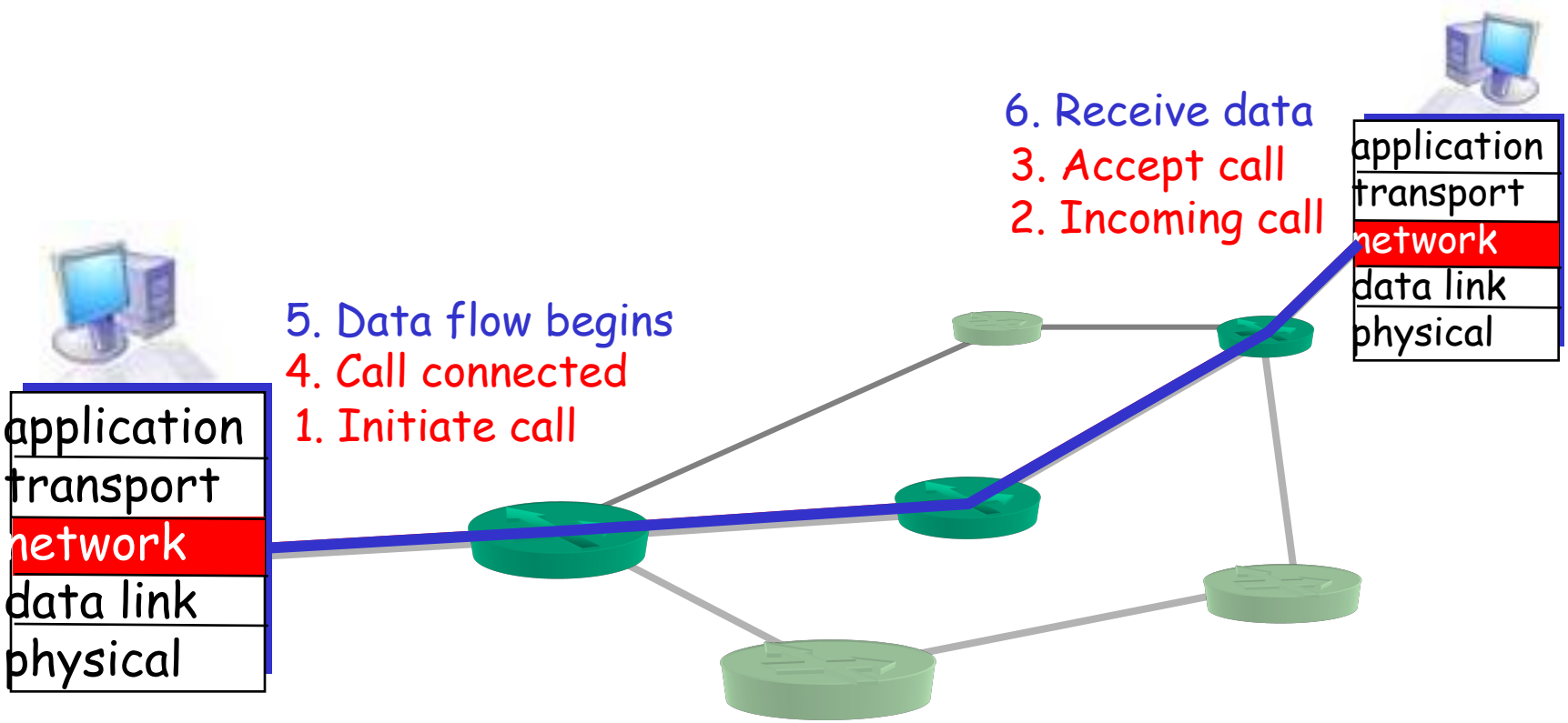
The table of router A is changed because of some reasons!

Management and update this tables for routing = Routing algorithm

a) Routing within a diagram subnet:

Virtual Circuits (1)

- Call setup, do for each call *before* data can flow
- Each packet carries VC identifier
- Used in ATM, frame-relay, X.25



Comparison of Virtual-Circuit and Datagram

	Internet	ATM
Issue	**Datagram subnet**	**Virtual-circuit subnet**
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Quality of Service: QoS

- Factors:
  a) Timing
     a) Connection Establishment Delay
     b) End-To-End Delay
  b) Connection Establishment Failure Probability
  c) Throughput or Bandwidth Guarantee
  d) Ordering Preservation
  e) Congestion Control
  f) Bit-Error rate or Packet-Loss Rate Control
  g) Protection
  h) Priority
  i) …

Routing (1)

- The network layer is responsible for routing packets from the source to destination.
- The routing algorithm is the piece of software that decides where a packet goes next (e.g., which output line, or which node on a broadcast channel).

- For connectionless networks, the routing decision is made for each datagram. For connection-oriented networks, the decision is made once, at circuit setup time.

- The routing algorithm must deal with the following issues:
  a) Correctness, simplicity, stability, fairness and optimality
  b) Mimizing mean packet delay or maximizing total network throughput

- Routing is different from Forwarding!:
  a) Forwarding: Select the output path using routing table
  b) Routing: Management and updating the routing tables

Routing (2)

- There are two types:

a) Static (Non-Adaptive)

   a) routes never update or update slowly over time
   b) Examples: Dijkstra, Flooding algorithm

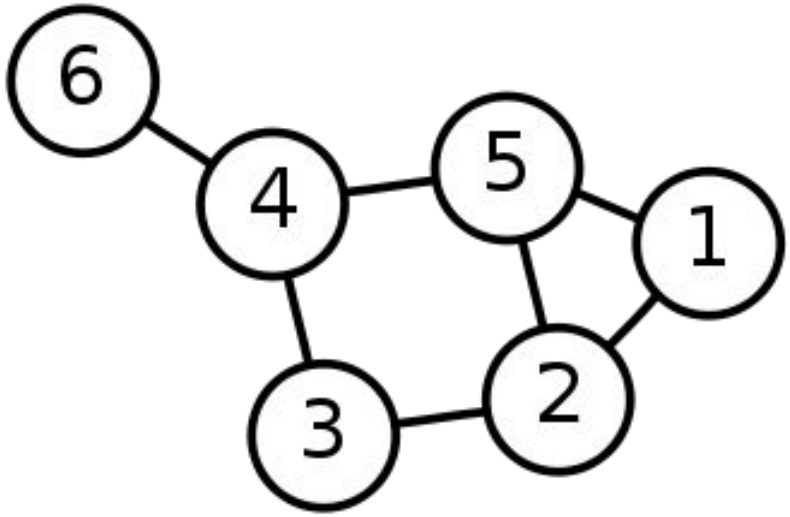
b) Dynamic (Adaptive)

   a) routes update more quickly use dynamic information of current topology such as load, delay, …
   b) Examples: Distance Vector, Link State Routing

Single-Source Shortest Path Problem

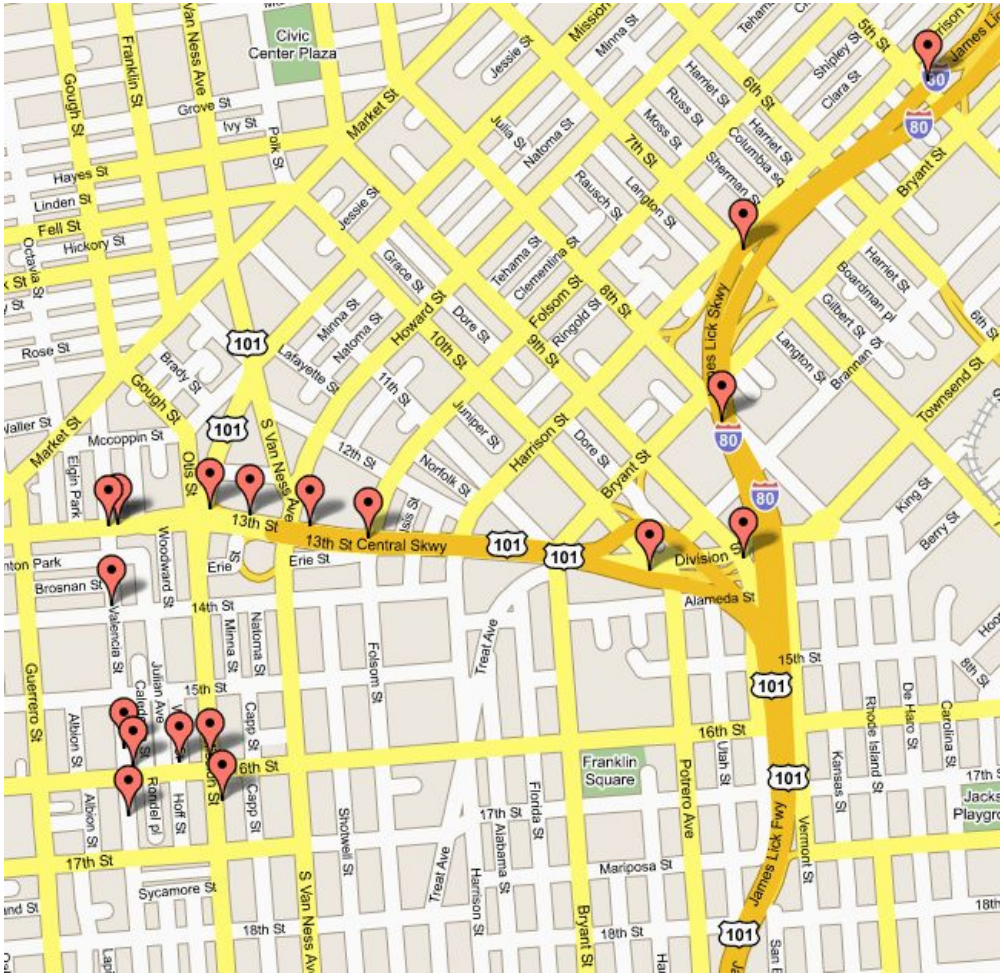
Dijkstra Algorithm

Single-Source Shortest Path Problem - The problem of finding shortest paths from a source vertex *v* to all other vertices in the graph.



Applications

- Maps (Map Quest, Google Maps)
- Routing Systems



Dijkstra's algorithm

Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory.

Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Input: Weighted graph G={E,V} and source vertex $v \in V$, such that all edge weights are nonnegative

Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices

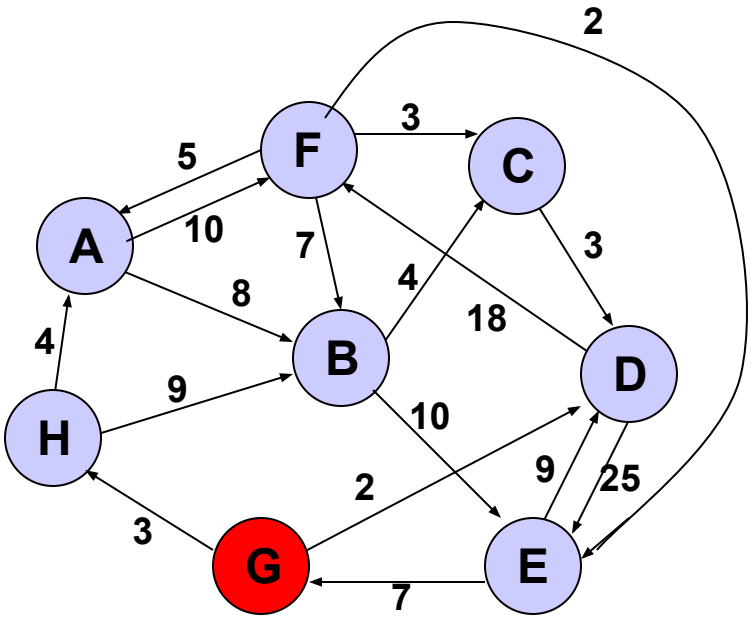
Approach

a) The algorithm computes for each vertex u the distance to u from the start vertex v, that is, the weight of a shortest path between v and u.

b) the algorithm keeps track of the set of vertices for which the distance has been computed.

c) Every vertex has a label D associated with it. For any vertex u, D[u] stores an approximation of the distance between v and u. The algorithm will update a D[u] value when it finds a shorter path from v to u.

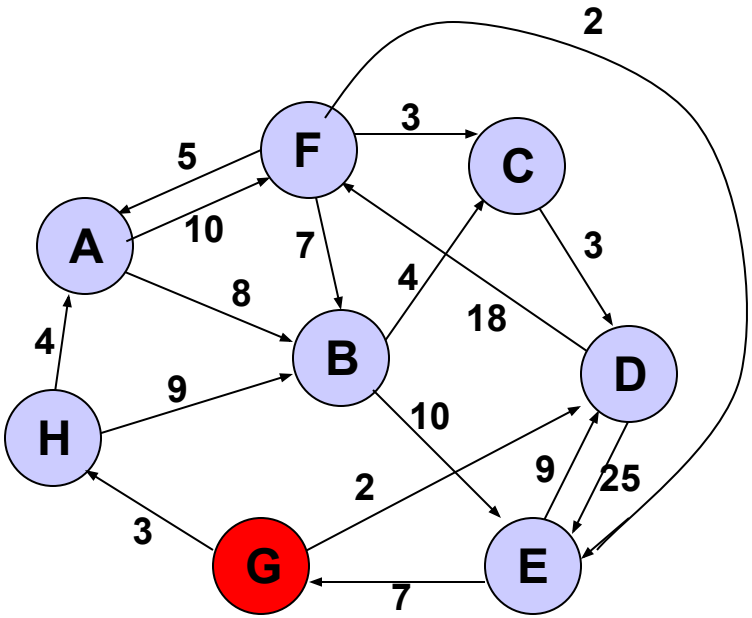
Dijkstra's algorithm



Start with G

	K	d_v	p_v
A			
B			
C			
D			
E			
F			
G	T	0	–
H			

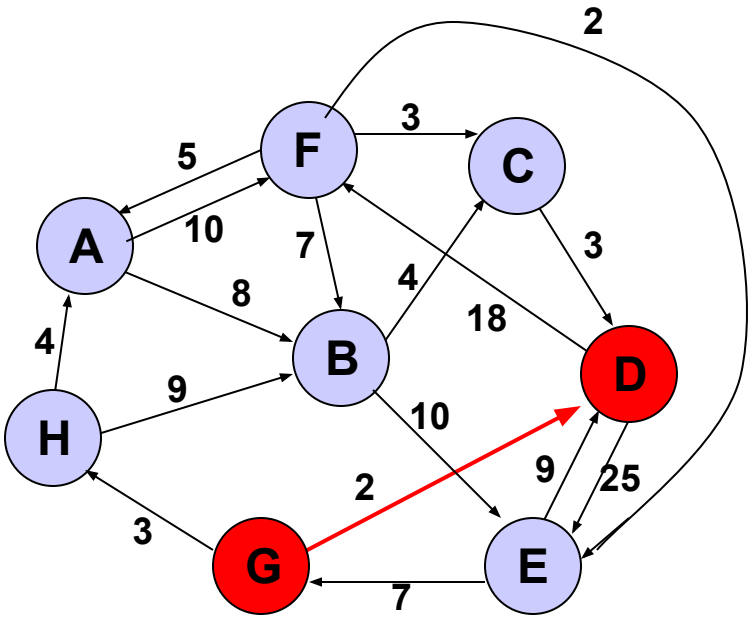
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A			
B			
C			
D		2	G
E			
F			
G	T	0	–
H		3	G

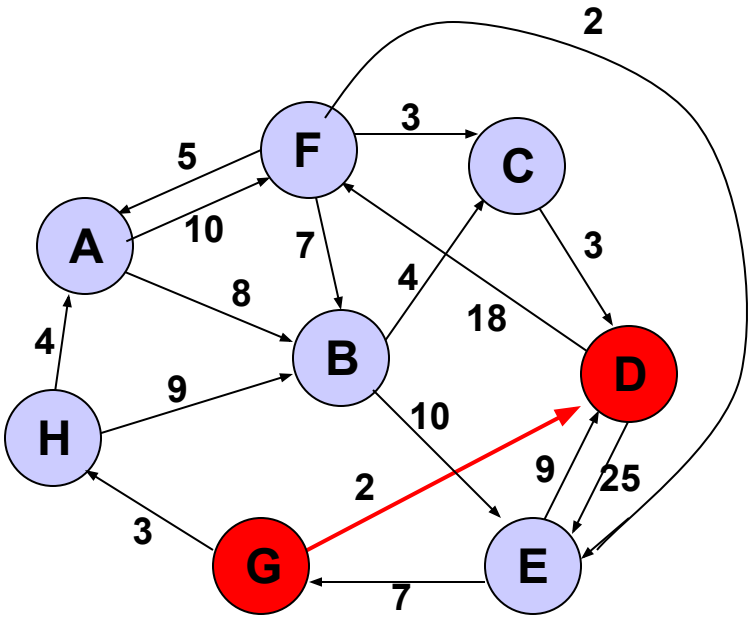
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A			
B			
C			
D	T	2	G
E			
F			
G	T	0	–
H		3	G

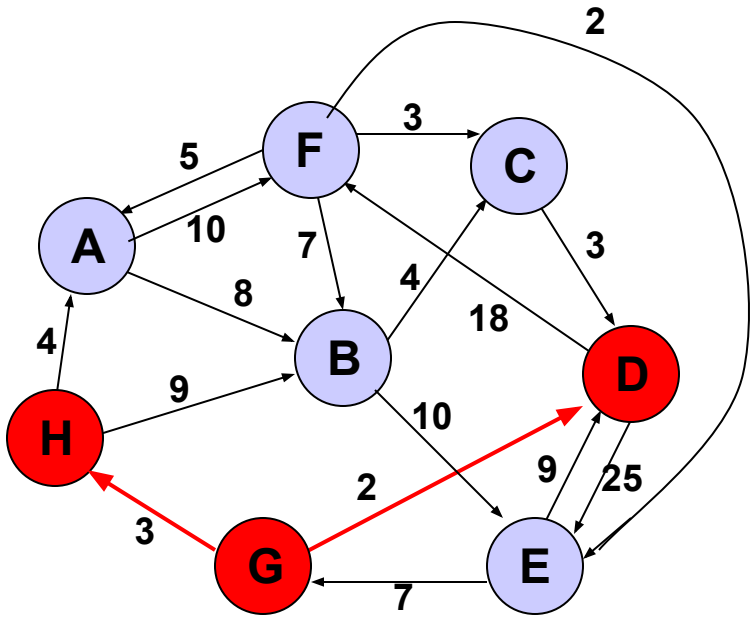
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A			
B			
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H		3	G

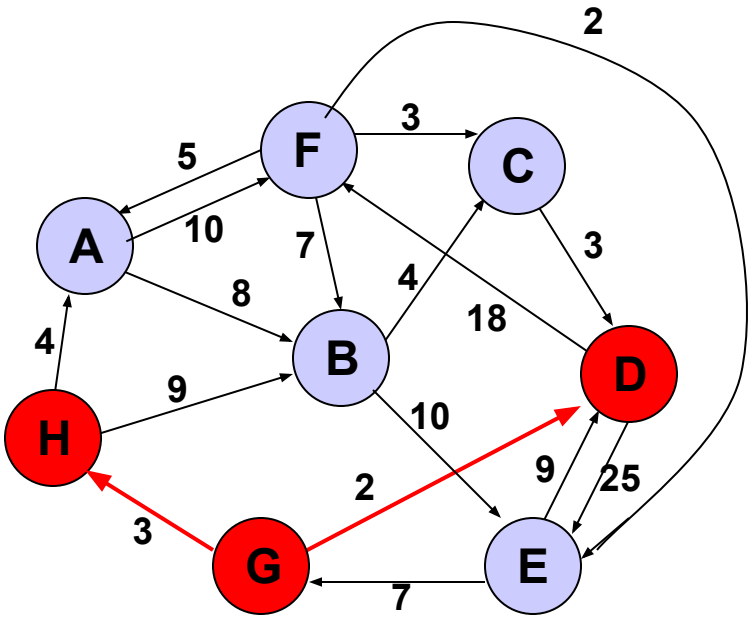
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A			
B			
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H	T	3	G

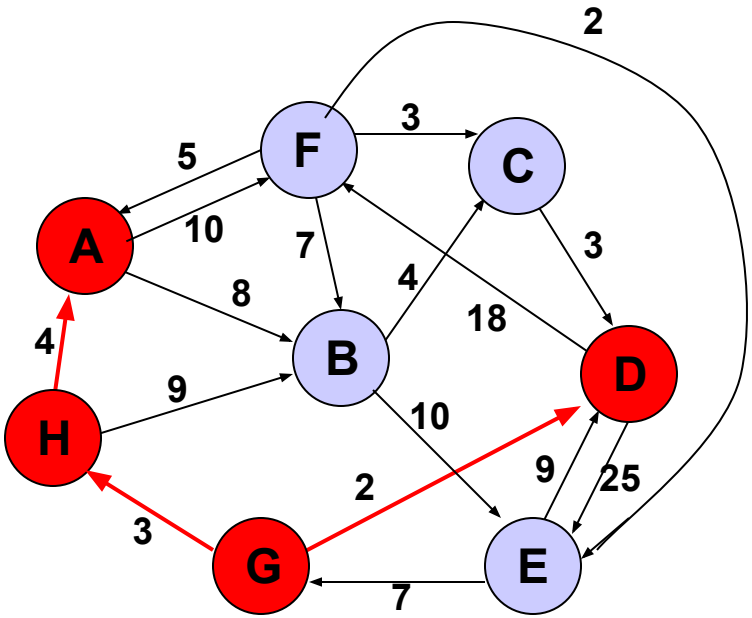
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A		7	H
B		12	H
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H	T	3	G

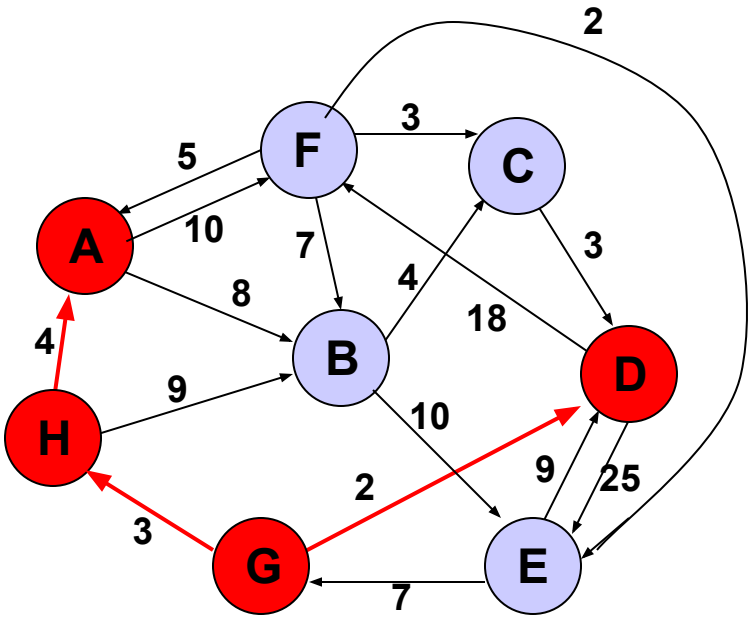
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A	T	7	H
B		12	H
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H	T	3	G

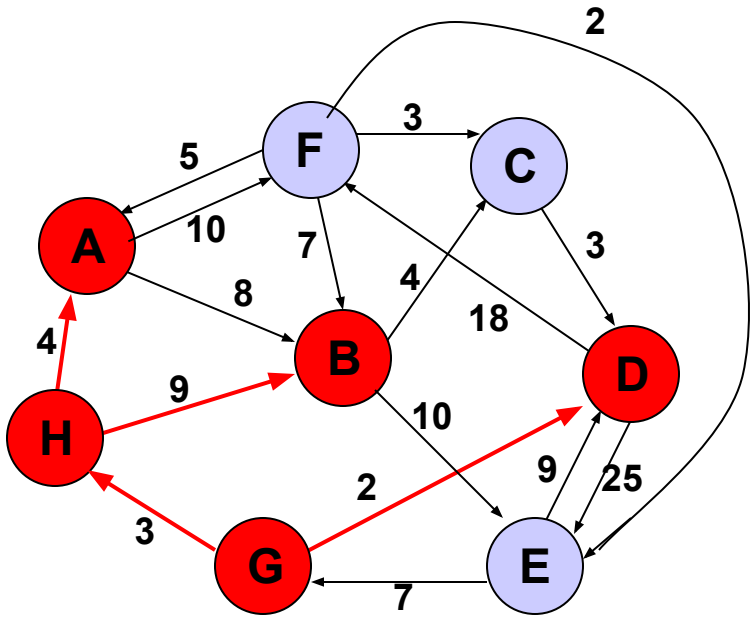
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B		12	H
C			
D	T	2	G
E		27	D
F		17	A
G	T	0	–
H	T	3	G

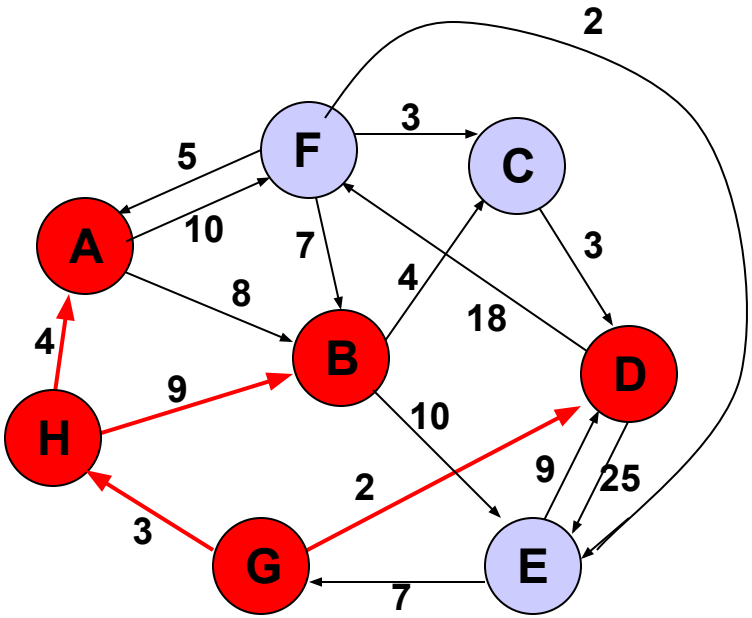
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C			
D	T	2	G
E		27	D
F		17	A
G	T	0	–
H	T	3	G

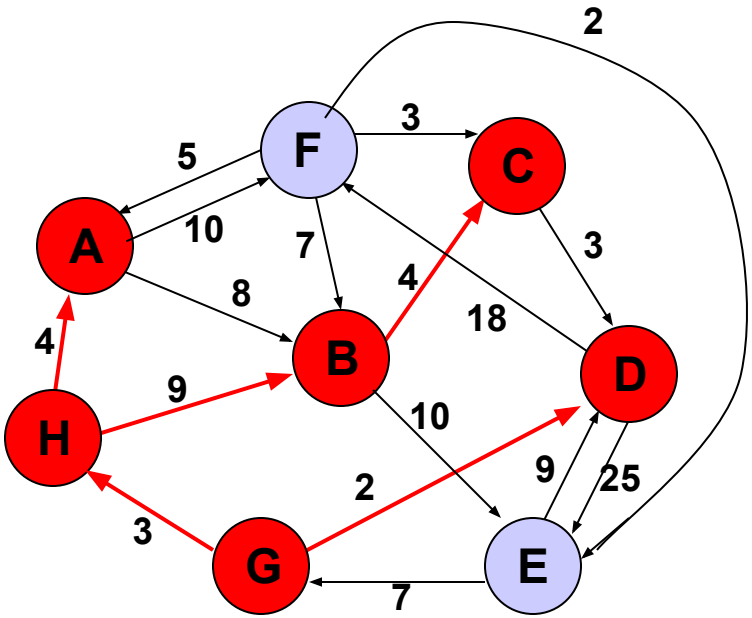
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B	T	12	H
C		16	B
D	T	2	G
E		22	B
F		17	A
G	T	0	–
H	T	3	G

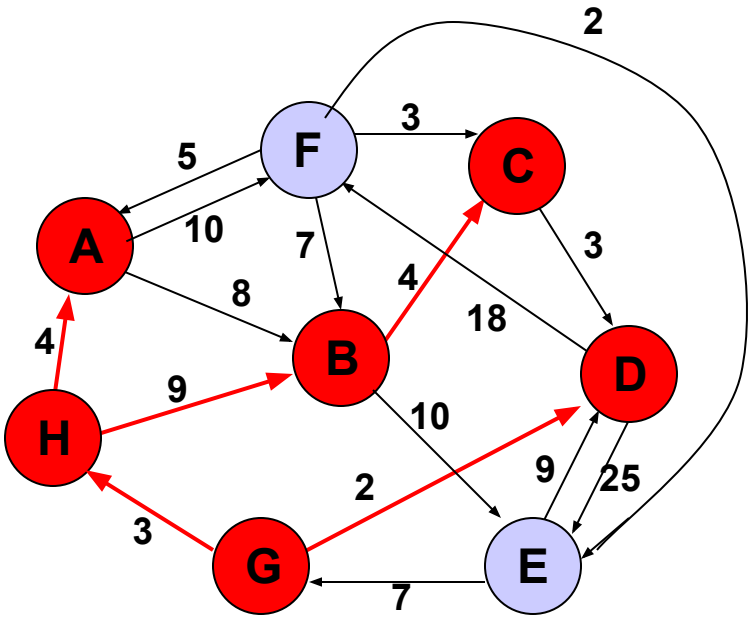
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		22	B
F		17	A
G	T	0	–
H	T	3	G

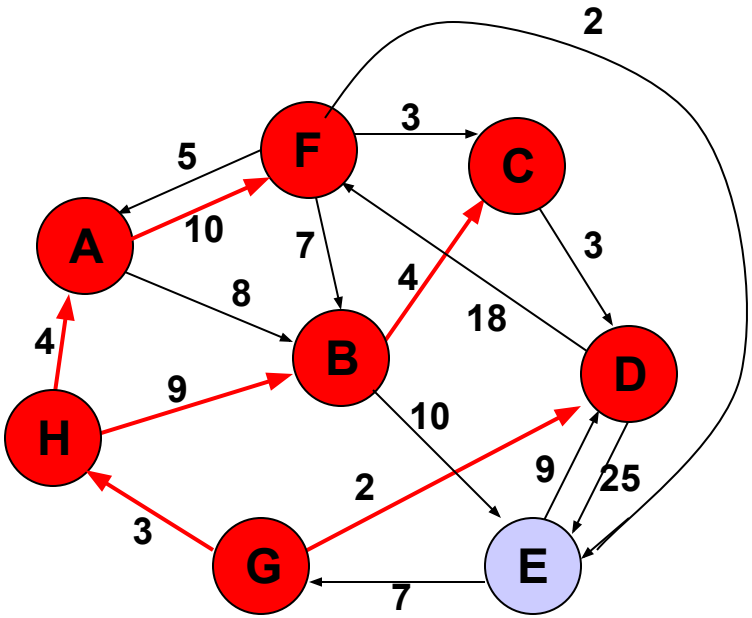
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		22	B
F		17	A
G	T	0	–
H	T	3	G

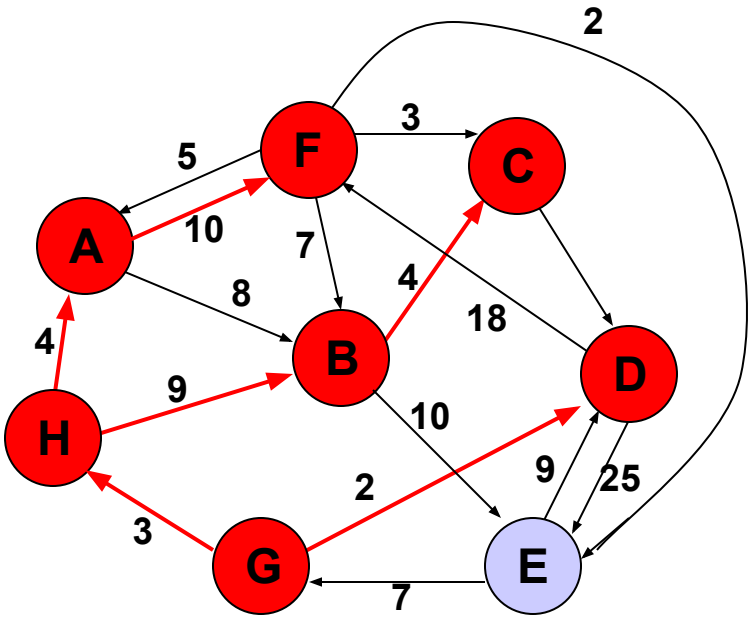
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		22	B
F	T	17	A
G	T	0	–
H	T	3	G

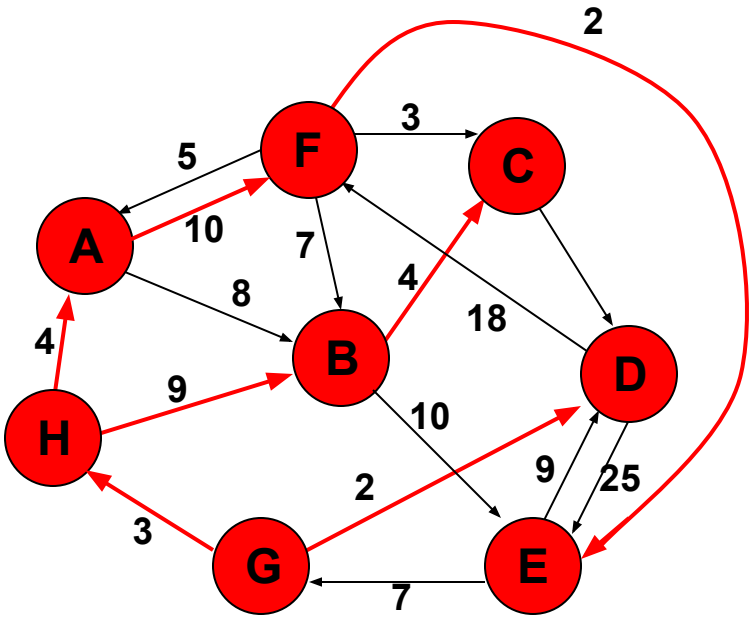
Dijkstra's algorithm



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		19	F
F	T	17	A
G	T	0	–
H	T	3	G

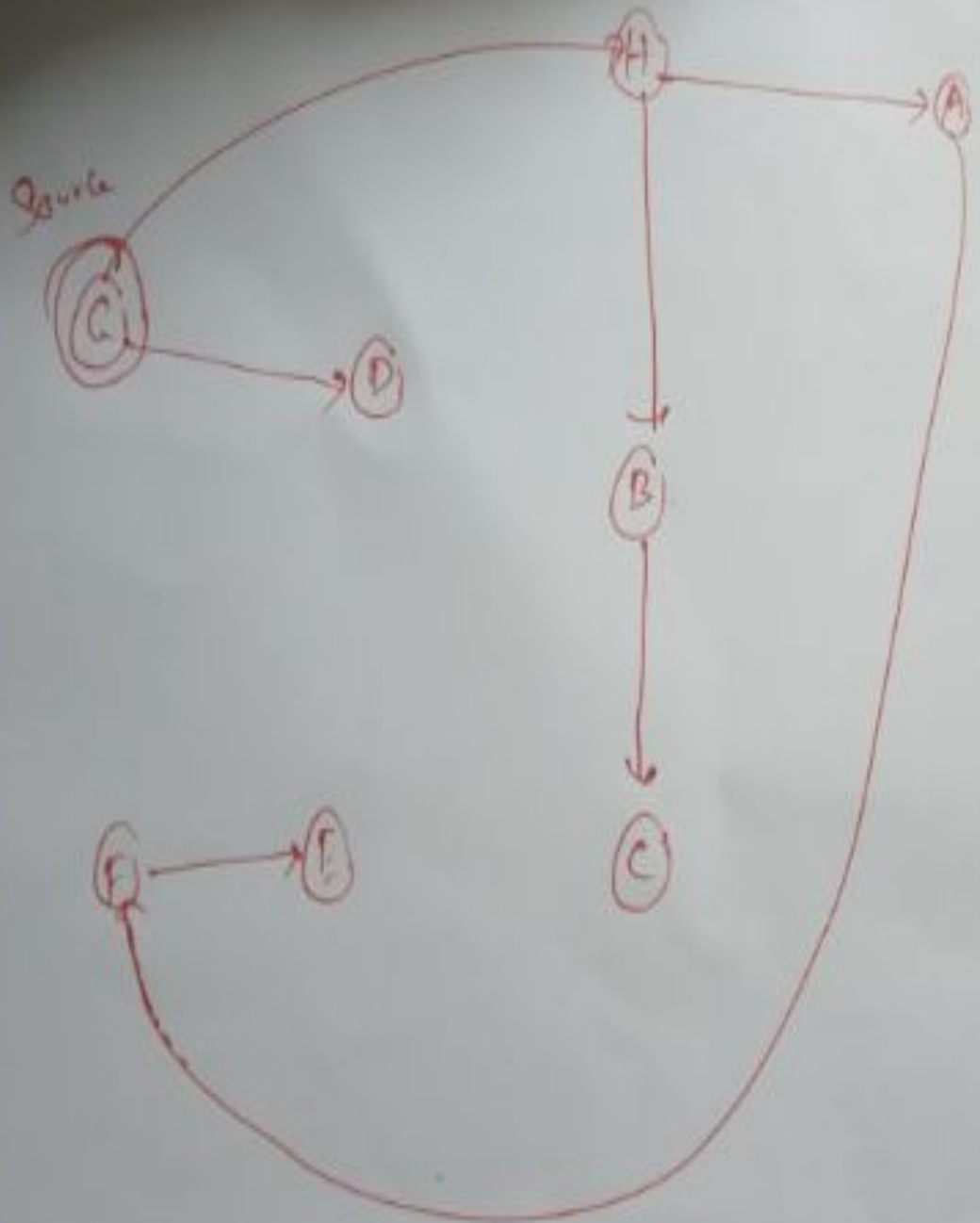
Dijkstra's algorithm



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E	T	19	F
F	T	17	A
G	T	0	–
H	T	3	G

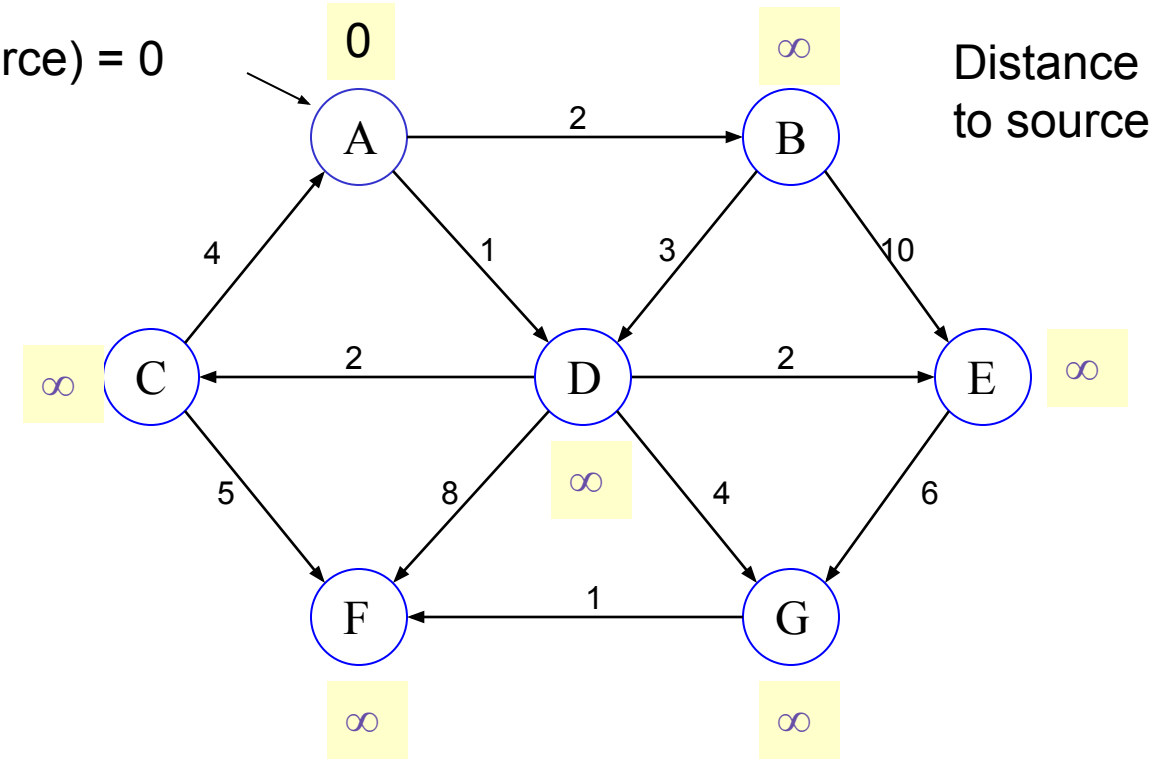
Done



Destination (G)	Dist	Cost
A	(G, H)	7
B	(G, H)	12
C	(G, H)	16
D	(G, D)	2
E	(G, H)	19
F	(G, H)	17
H	(G, H)	3

Dijkstra's-Example

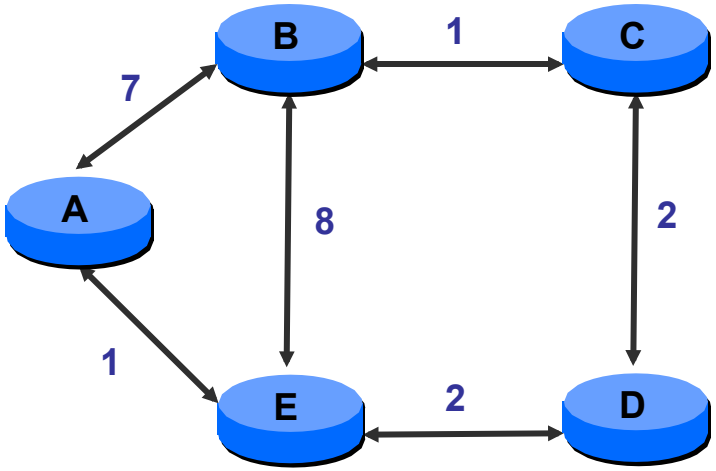
Distance(source) = 0



Distance (all vertices to source) = ∞

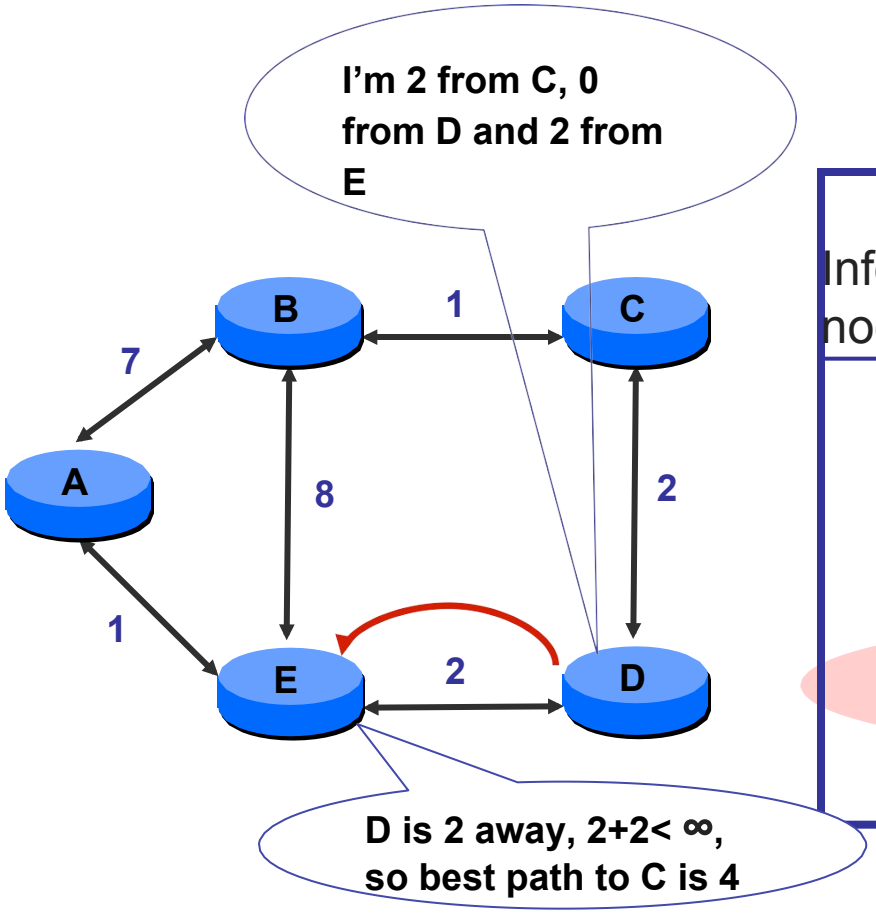
Pick vertex in List with minimum distance.

Distance Vector Routing



Info at node	Distance to Node A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

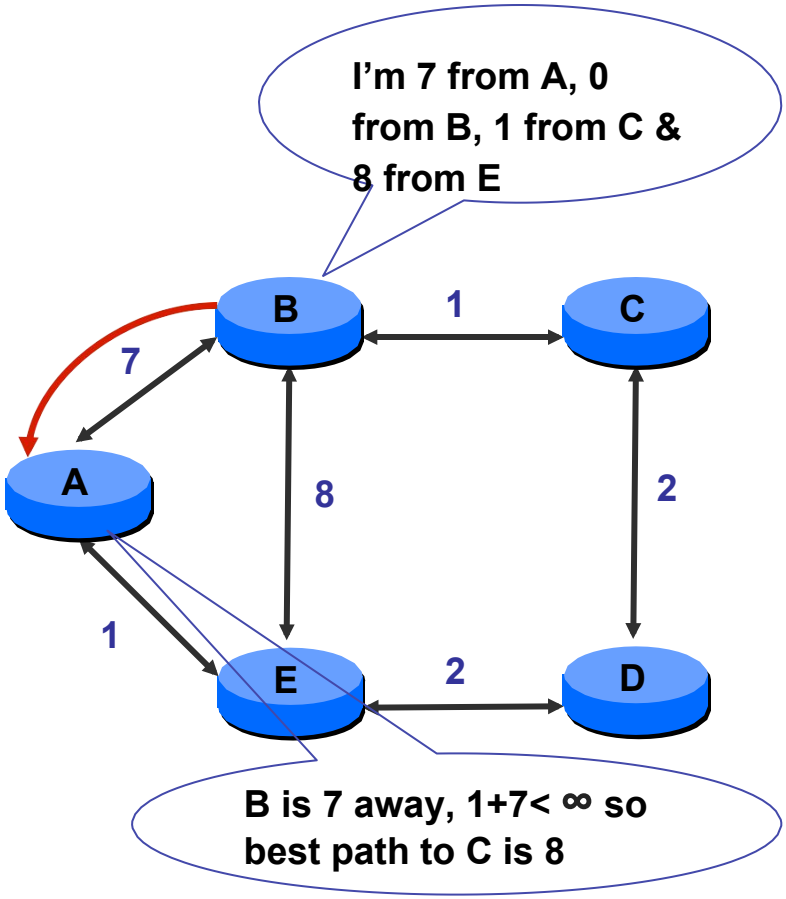
Distance Vector Routing
D sends vector to *E*



Info at node	Distance to Node A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	**4**	2	0

Distance Vector Routing

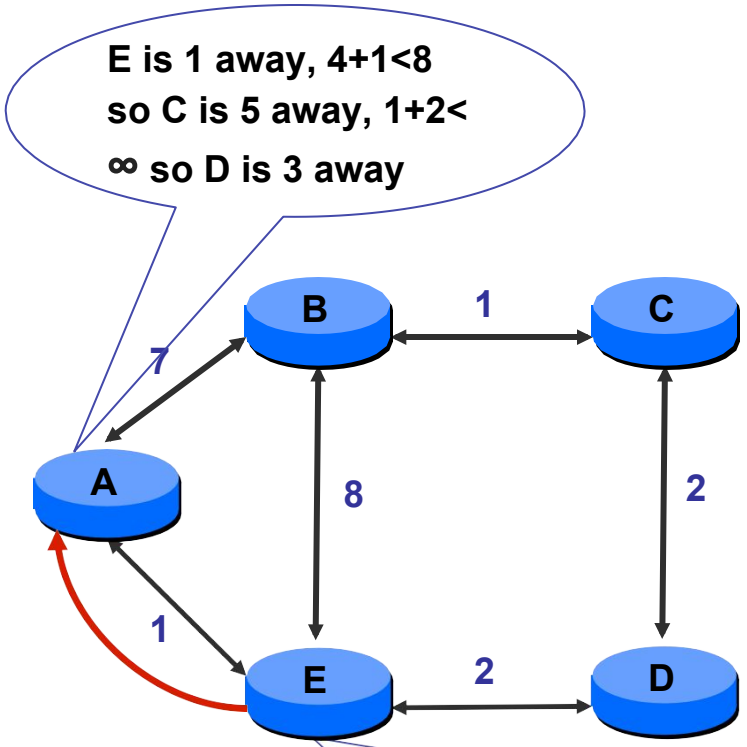
B sends vector to *A*



Info at node	Distance to Node A	B	C	D	E
A	0	7	8	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

Distance Vector Routing

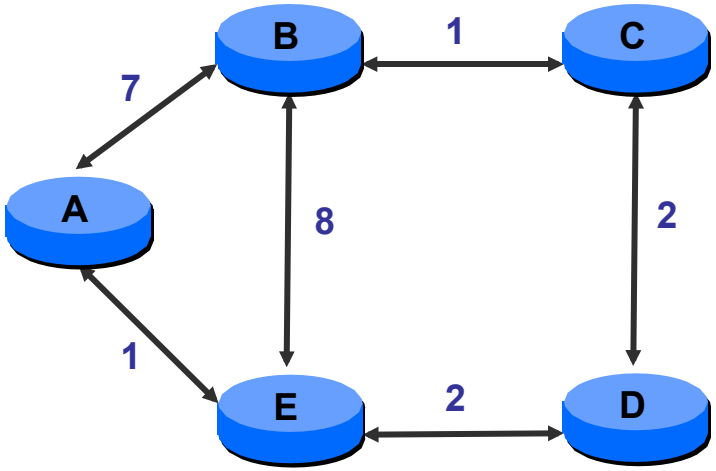
E sends vector to *A*



Info at node	Distance to Node A	B	C	D	E
A	0	7	**5**	**3**	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

Distance Vector Routing

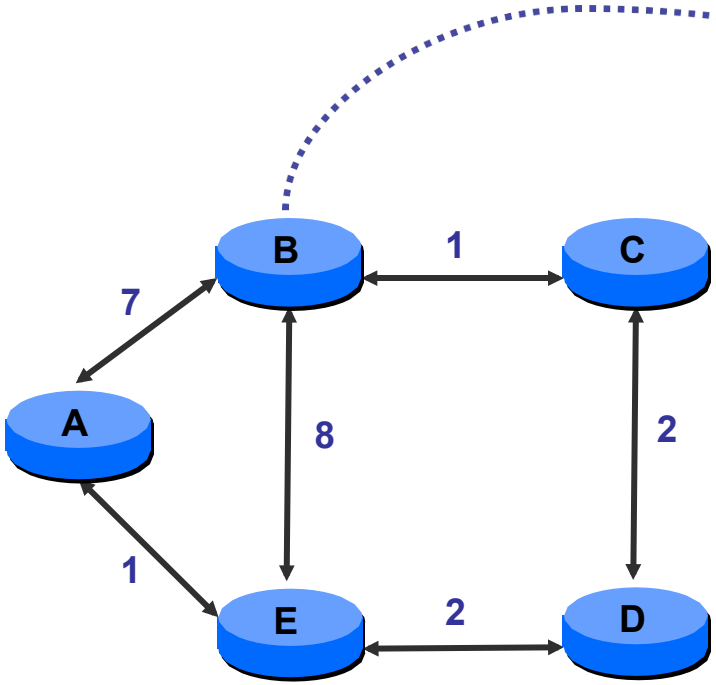
…until Convergence



Info at node	Distance To Node A	B	C	D	E
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

Distance Vector Routing

Node B's distance vectors

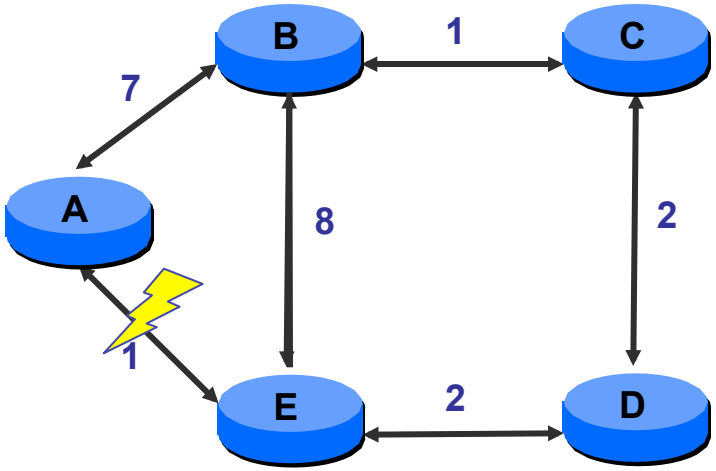


Dest	Next hop A	E	C
A			6
C			1
D			3
E			5

Distance Vector Routing

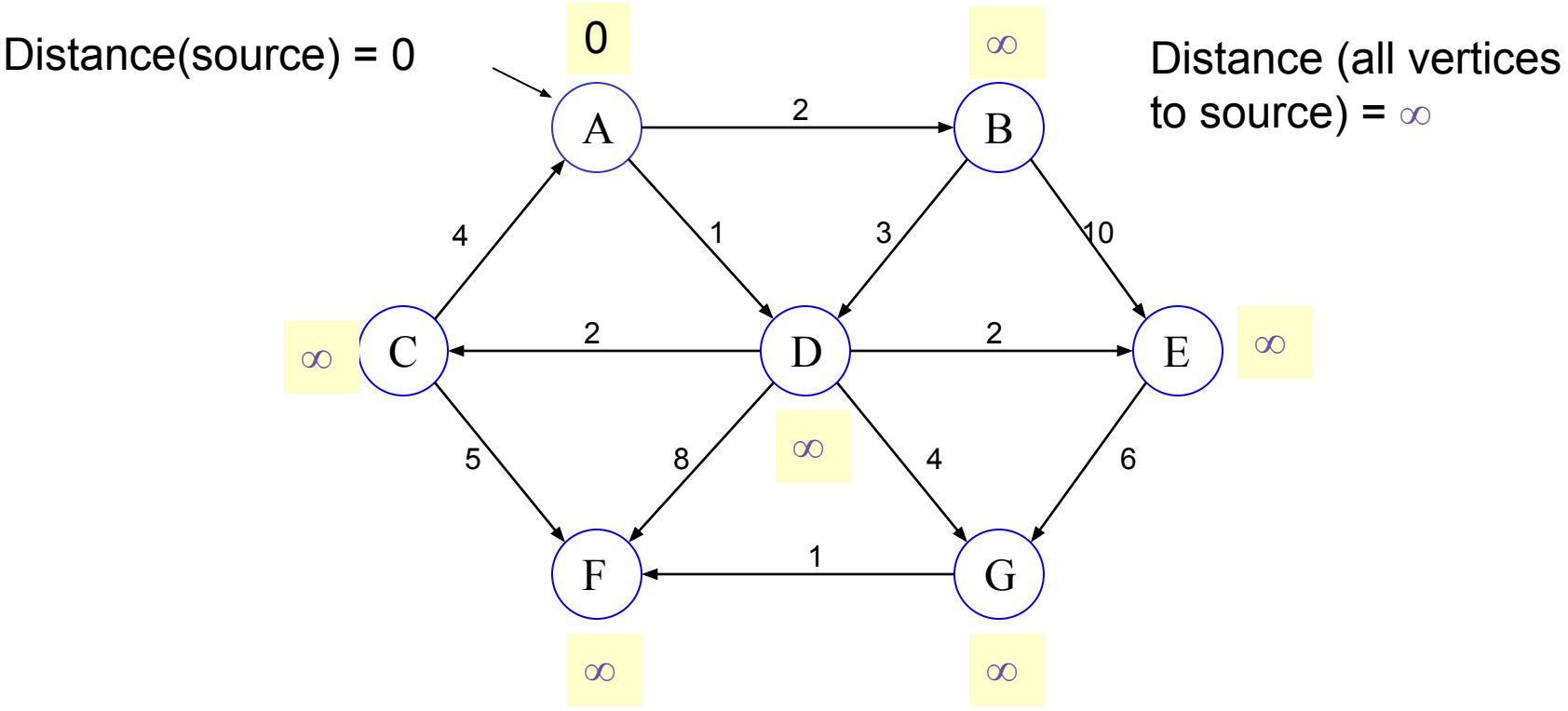
Handling Link Failure

- A marks distance to E as ∞ , and tells B
- E marks distance to A as ∞ , and tells B and D
- B and D recompute routes and tell C, E and E
- etc… until converge



Info at node	Distance to Node A	B	C	D	E
A	0	7	8	10	12
B	7	0	1	3	5
C	8	1	0	2	4
D	10	3	2	0	2
E	12	5	4	2	0

Distance Vector Routing-Example



Pick vertex in List with minimum distance.

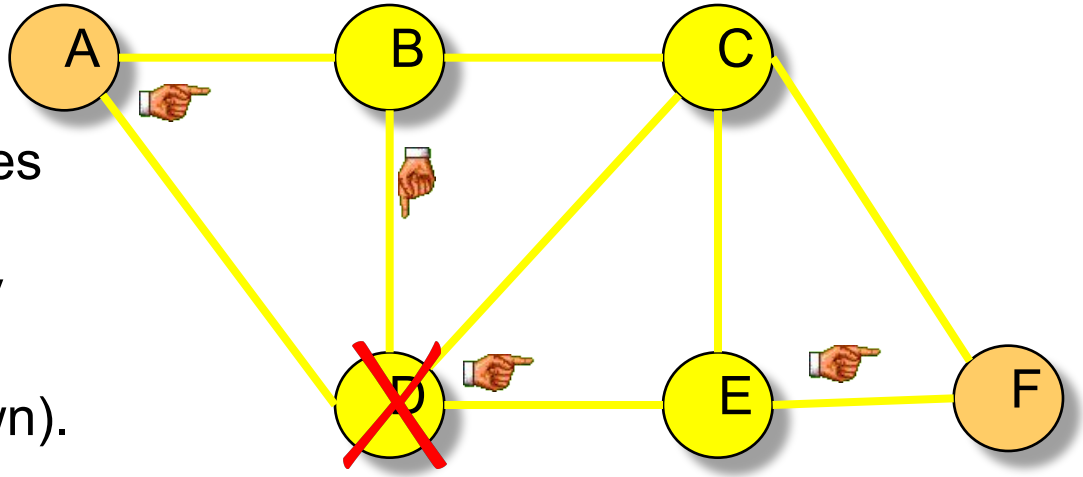
Distance Vector Routing

Distance Vector Routing

a) Each node knows the distance (=cost) to its directly connected neighbors
b) A node sends a list to its neighbors with the current distances to all nodes
c) Idea:
d) Tell neighbors about learned distances to all destinations
e) If all nodes update their distances, the routing tables eventually converge

Distance Vector vs. Link State Routing

- With distance vector routing, each node has information only about the next hop:
  a) Node A: to reach F go to B
  b) Node B: to reach F go to D
  c) Node D: to reach F go to E
  d) Node E: go directly to F

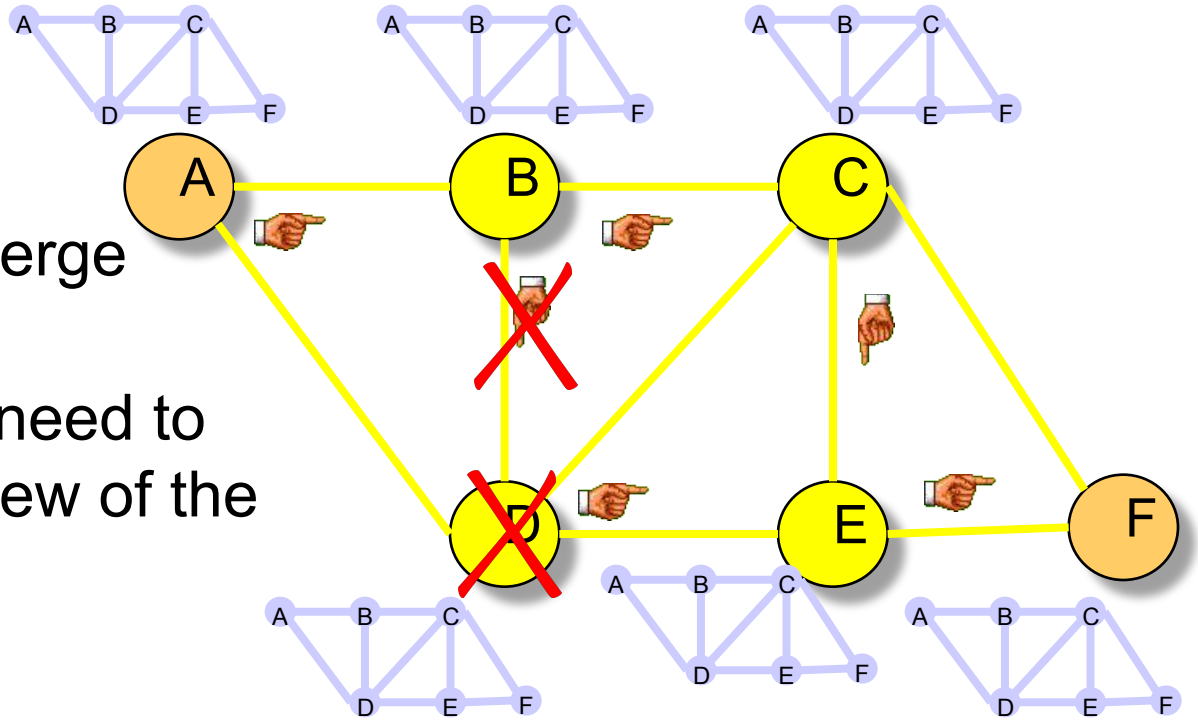


- Distance vector routing makes poor routing decisions if directions are not completely correct
(e.g., because a node is down).

- If parts of the directions incorrect, the routing may be incorrect until the routing algorithms has re-converged.

Distance Vector vs. Link State Routing

- In link state routing, each node has a complete map of the topology (Link state information must be flooded to all nodes).
- If a node fails, each node can calculate the new route.
- Guaranteed to converge
- Difficulty: All nodes need to have a consistent view of the network



Link State Routing

- Each router creates a *link state packet* (LSP) which contains names (e.g. network addresses) and cost to each of its neighbours

a) The LSP is transmitted to *all* other routers, who each update their own records.

b) When a routers receives LSPs from all routers, it can use (collectively) that information to make topology-level decisions

Link State Packets

- LSPs are generated and distributed when:
  a) A time period passes(proactive routing)
  b) New neighbours connect to the router
  c) The link cost of a neighbour has changed
  d) A link to a neighbour has failed (link failure)
  e) A neighbour has failed (node failure)

- Distribution of LSPs can be difficult
  a) Routers themselves are the means for delivering messages
  b) How do routers deliver their own messages, particularly when routers are in an inconsistent state
     a) e.g. During link failure, before each router has been notified of the problem

Link State Packets

- One method for LSP distribution: Flooding
  a) Each LSP received is transmitted to every direct neighbour (except the neighbour where the LSP came from)
  b) This creates an exponential number of packets on the network.
  c) It does, however, guarantee that the LSP will be received by every router
     a) Disadvantage of flooding:
     b) Possibility of count-to-infinity (Infinite Loops)
     c) Routers may receive the same LSP from multiple routers

Link State Packets

- An improvement to the flooding is as follows:
- When an LSP is received, it's sequence number is compared with the stored copy
  a) If it is identical to the stored copy, it is dropped
  b) If it is different, the stored LSP is overwritten with the new LSP and the LSP is transmitted to every direct neighbour (except the source of the LSP)
- This scheme works because if a given router has already received a LSP from another neighbour, it will have also already distributed the LSP to all of its neighbours

Link State Routing Algorithm

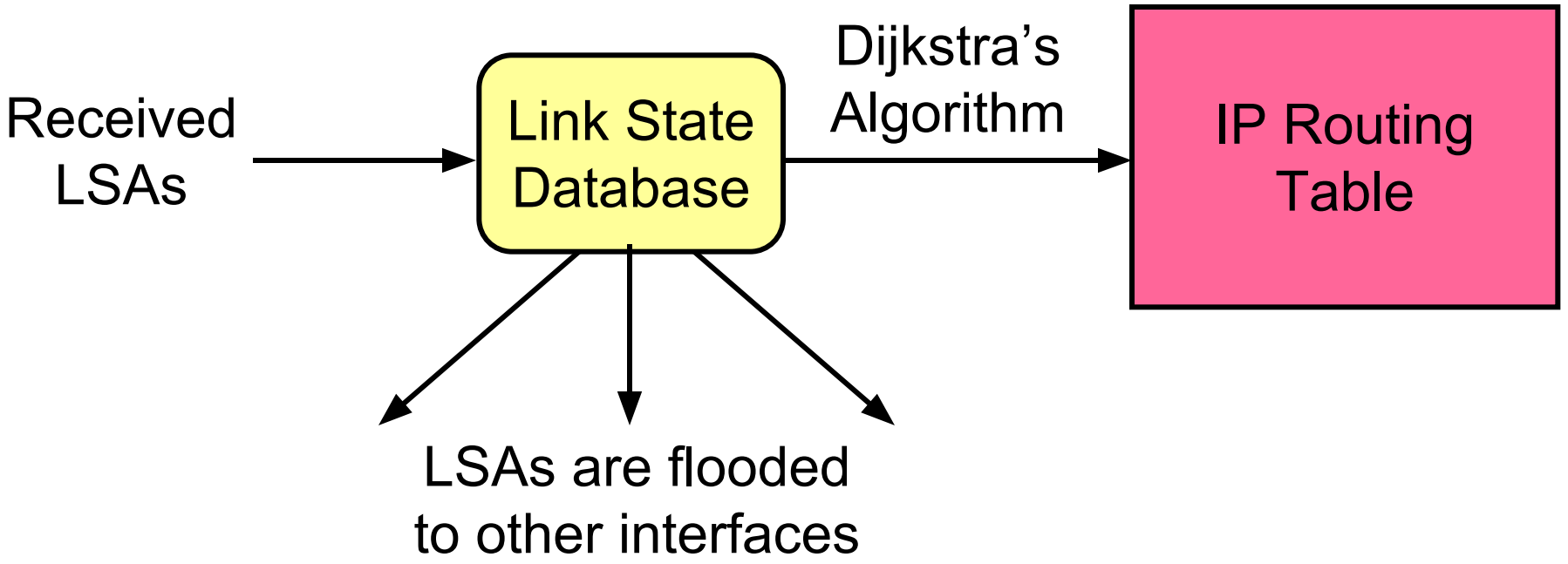
- Ok, now that we know how to distribute LSPs, how are they used to determine routes?

a) The algorithm used was developed by Dijkstra.

b) Essentially, the algorithm runs at each router, computing each possible path to the destination, adding up each cost

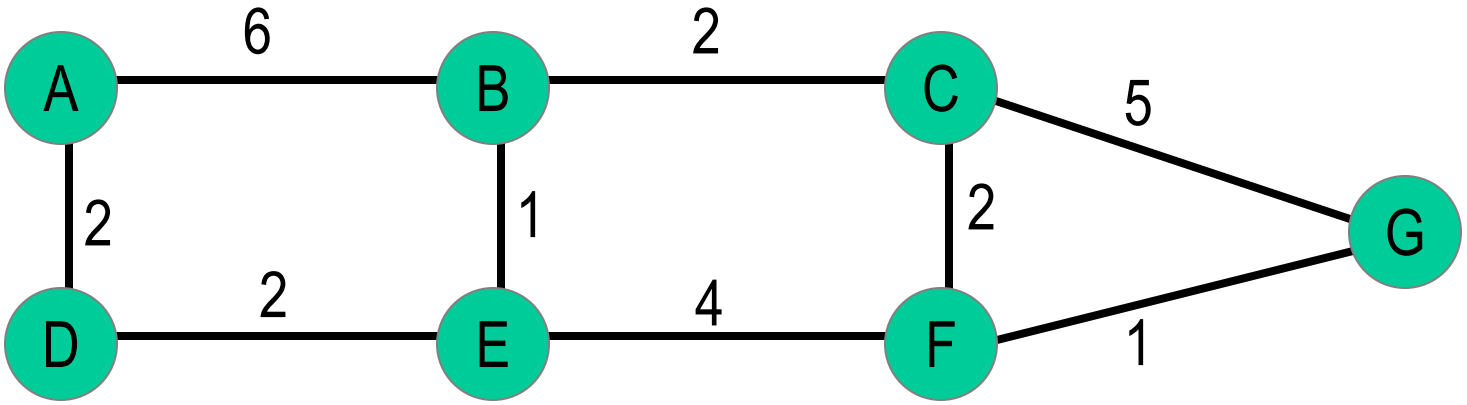
    a) The path with the lowest cost is used

Operation of a Link State Routing protocol



Dijkstra's LSR Algorithm

a) Consider the following network:



Link state database:

A	
B	6
D	2

B	
A	6
C	2
E	1

C	
B	6
F	2
G	5

D	
A	2
E	2

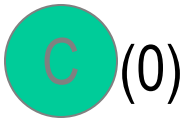
E	
B	1
D	2
F	4

F	
C	2
E	4
G	1

G	
C	5
F	1

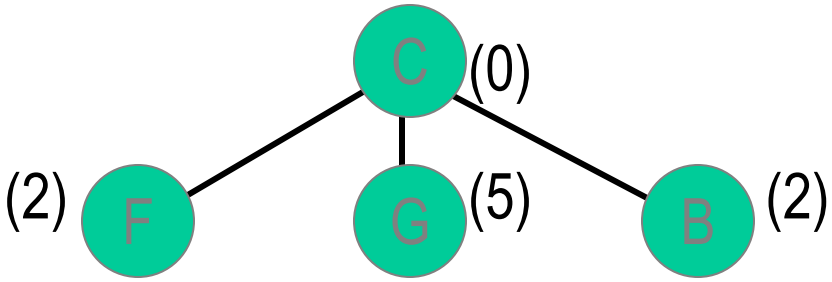
Dijkstra's LSR Algorithm

a) Now, if we want to generate a PATH for C:
   a) First, we add (C,0,0) to PATH



Dijkstra's LSR Algorithm

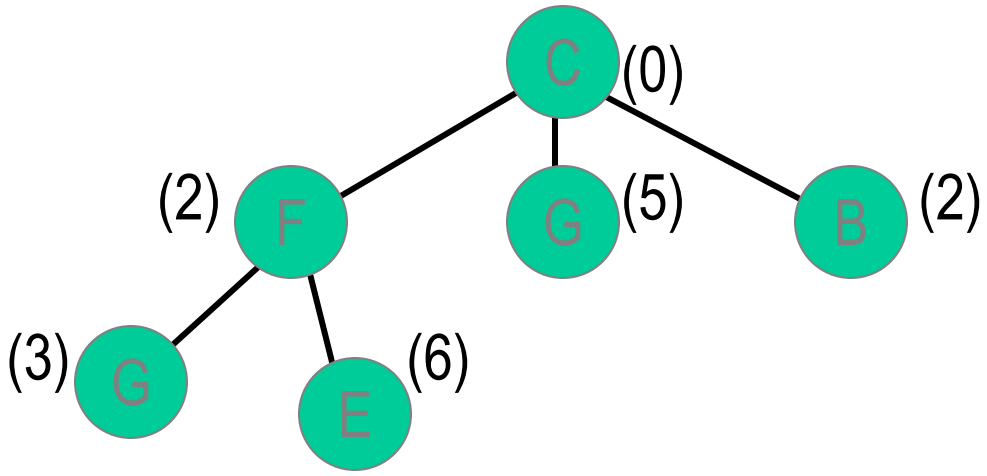
a) Examine C's LSP
   a) Add F, G, and B to TENT



Dijkstra's LSR Algorithm

a) Place F in PATH (shown as solid line)

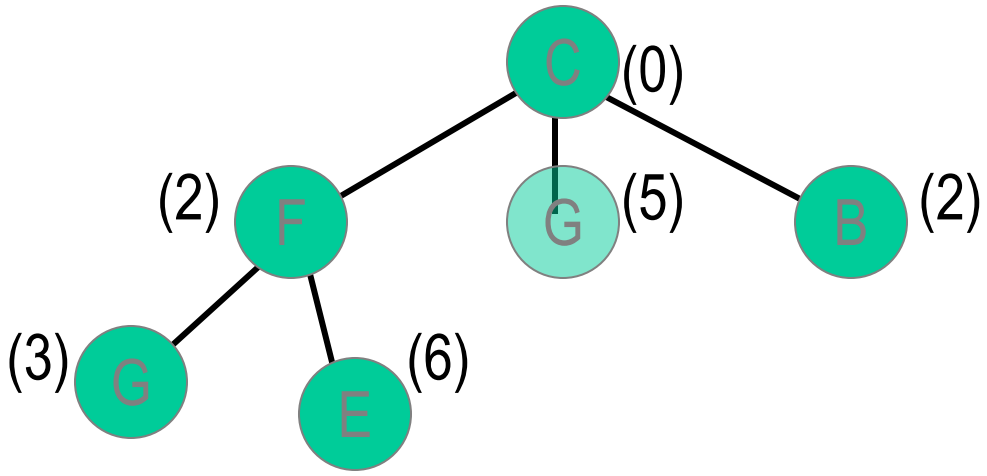
a) Add G and E to TENT (adding costs)



Dijkstra's LSR Algorithm

a) G exists in TENT twice, keep only the best

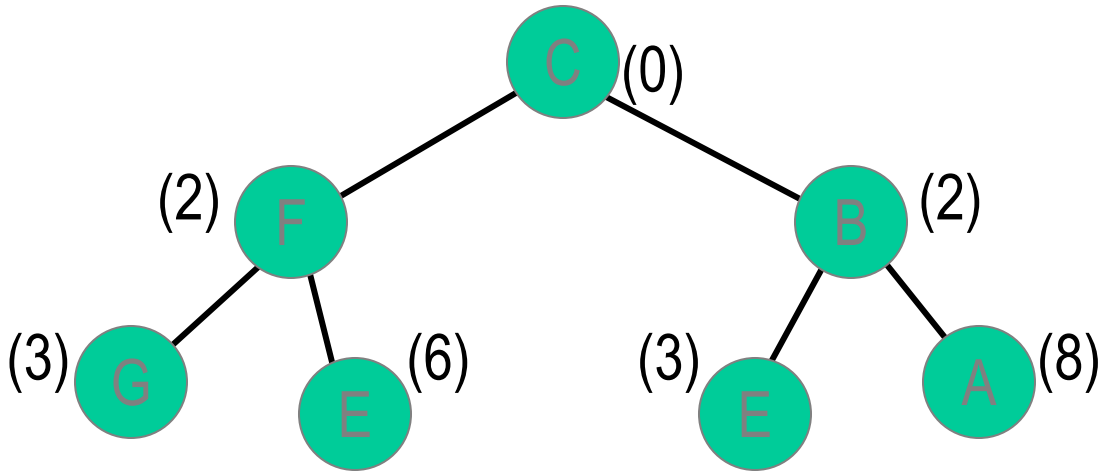
   a) The new G is a better path than the old (3 < 5)



Dijkstra's LSR Algorithm

a) Put B into path (shown as solid line)

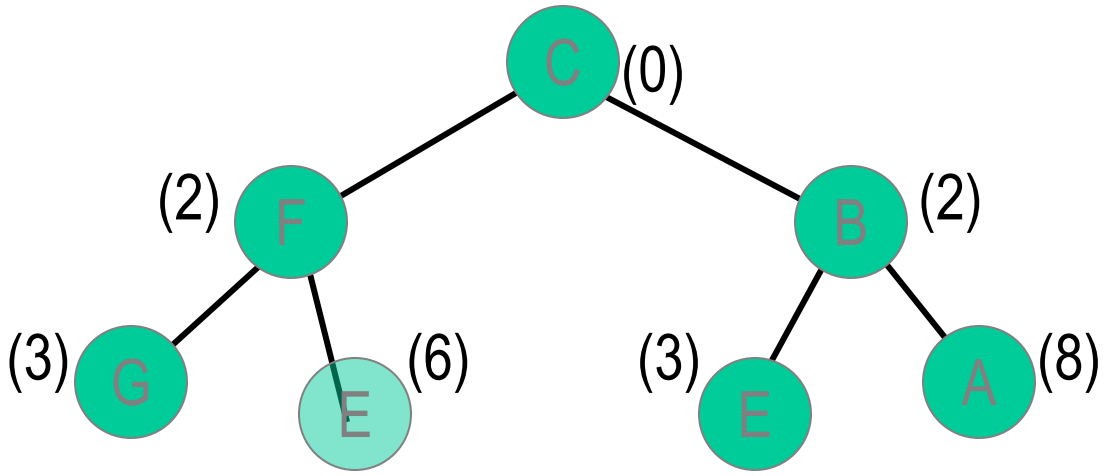
a) Add A and E to TENT



Dijkstra's LSR Algorithm

a) E exists in TENT twice, keep only the best

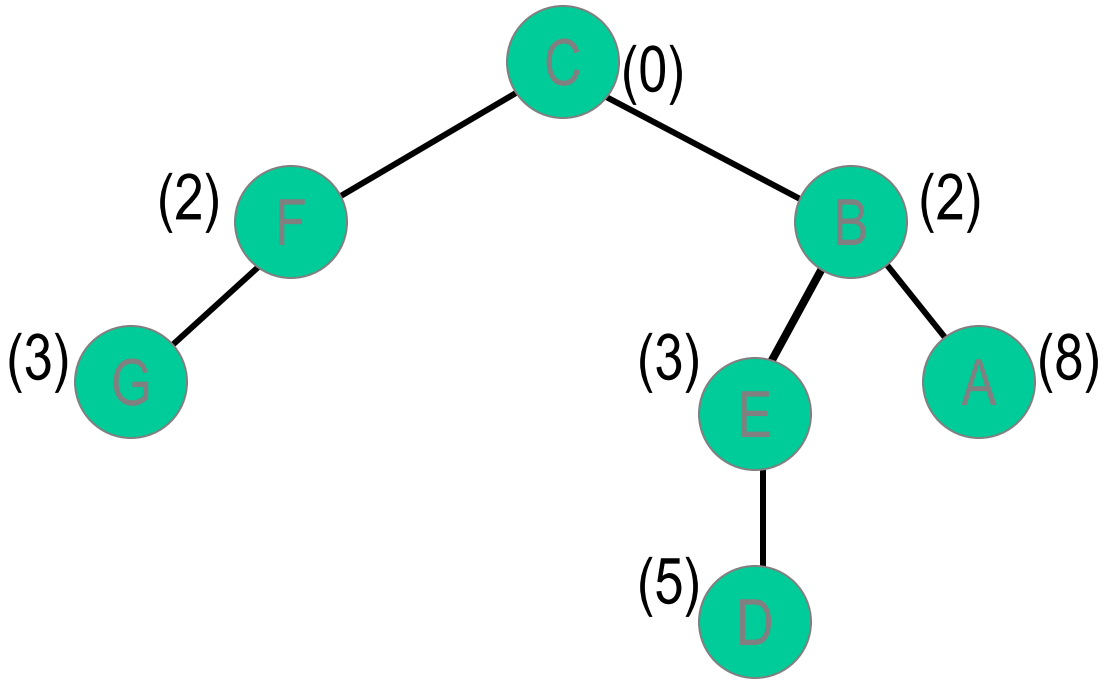
   a) The new E is better than the old (3 < 6)



Dijkstra's LSR Algorithm

a) Place E in PATH (shown as solid line)

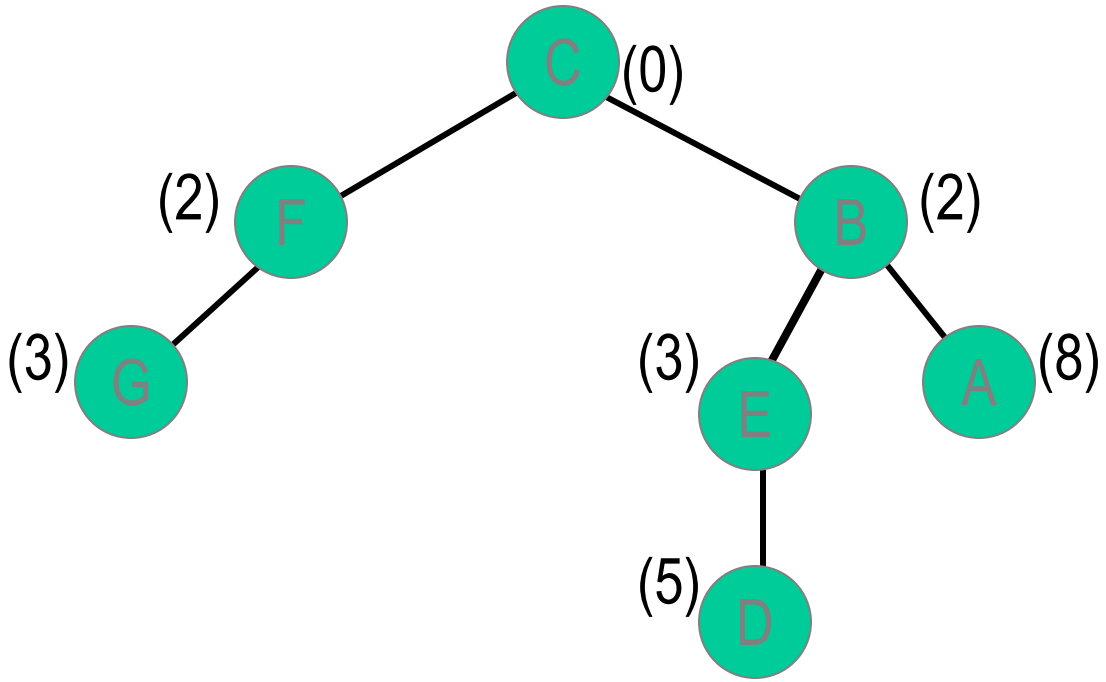
a) Add D to TENT



Dijkstra's LSR Algorithm

a) Place G in PATH (shown as solid line)

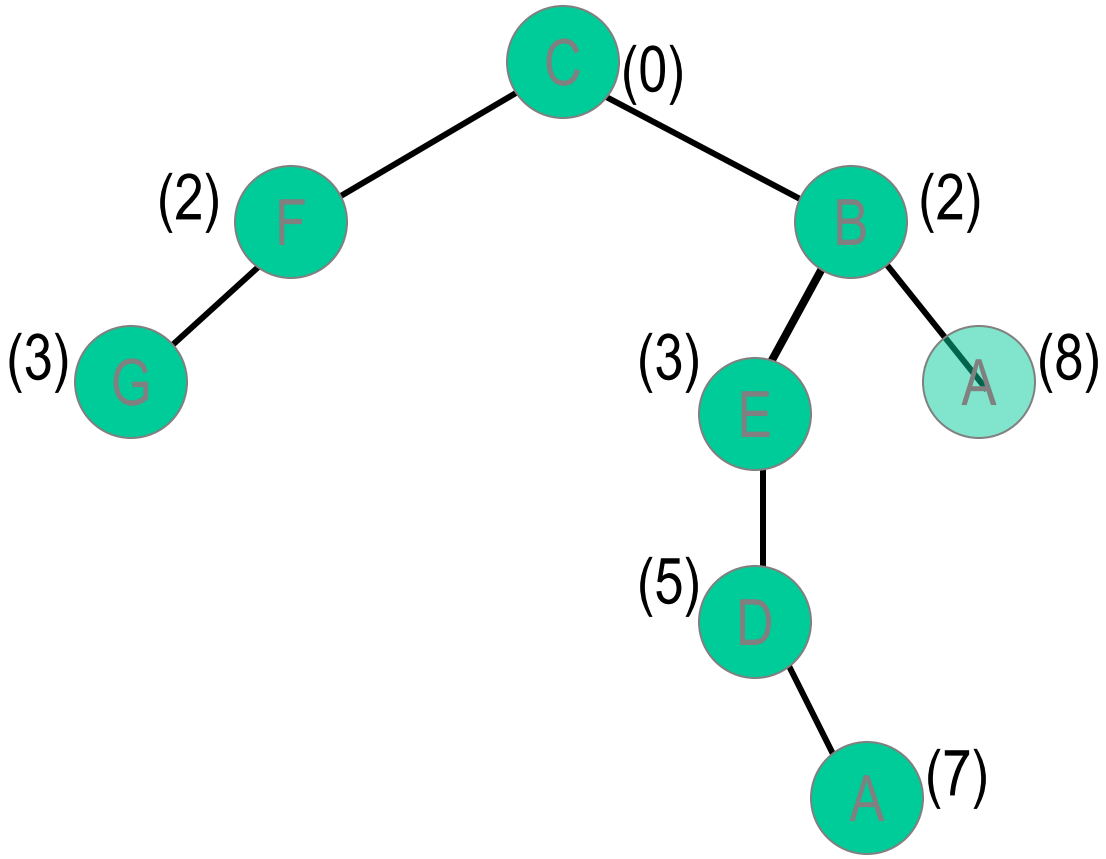
a) All G's LSP elements already exist in TENT



Dijkstra's LSR Algorithm

a) Place D in PATH (shown as solid line)

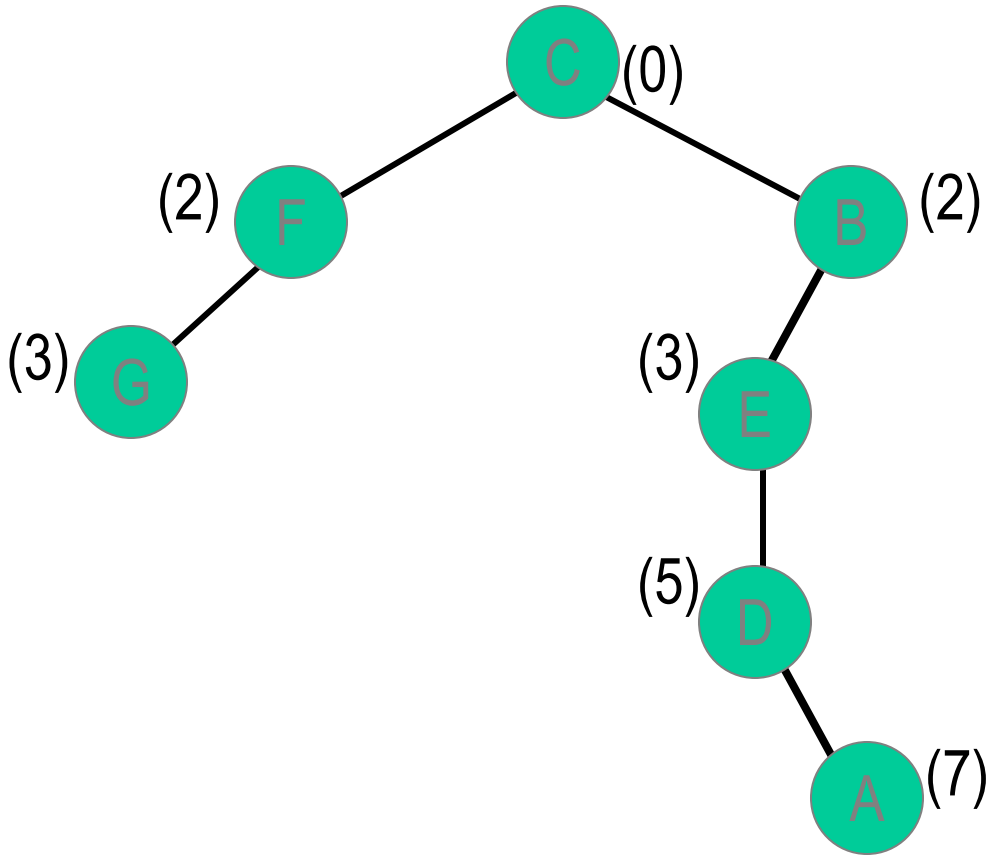
a) Add path to A since it is better than old A



Dijkstra's LSR Algorithm

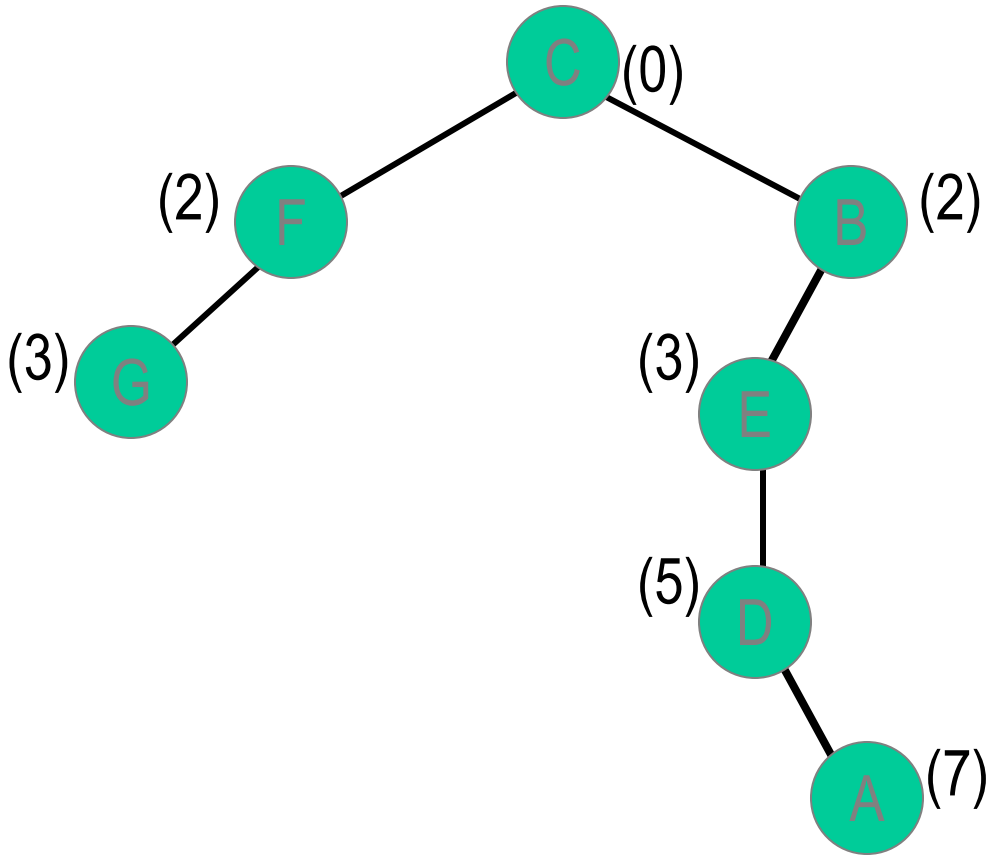
a) Place A in PATH (shown as solid line)

a) All A's LSP elements already exist in PATH



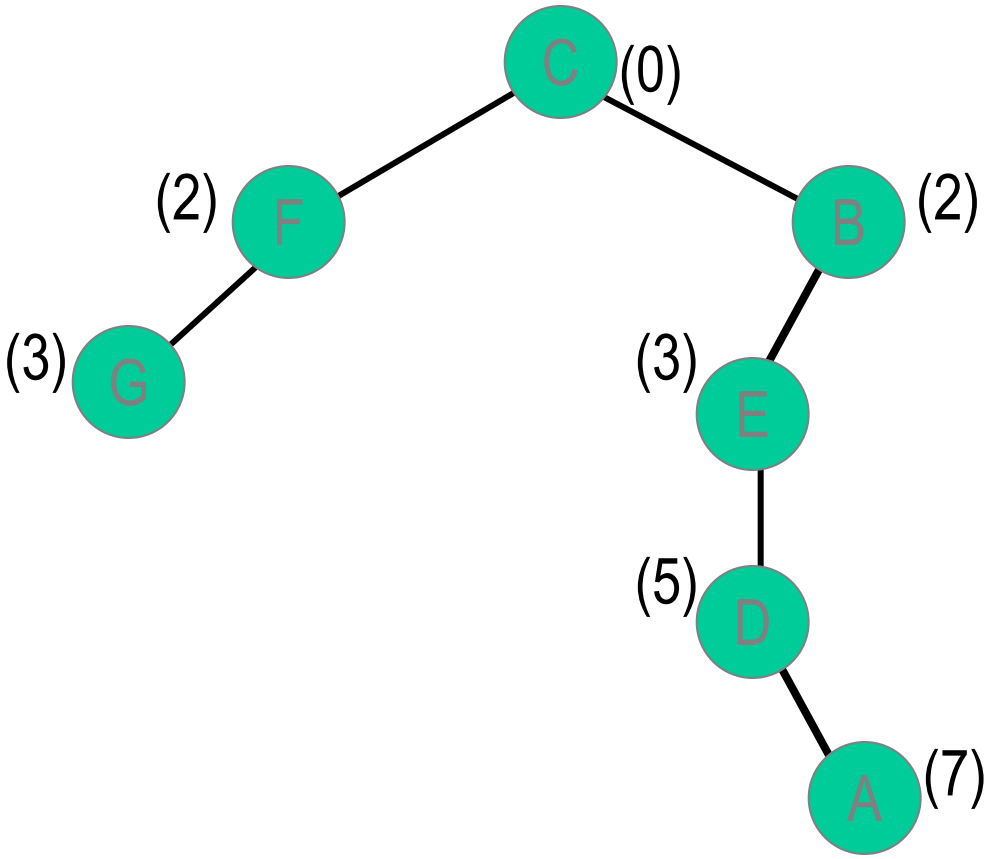
Dijkstra's LSR Algorithm

a) We are done since all routes from TENT were placed into PATH



Dijkstra's LSR Algorithm

- We can now create a forwarding database:



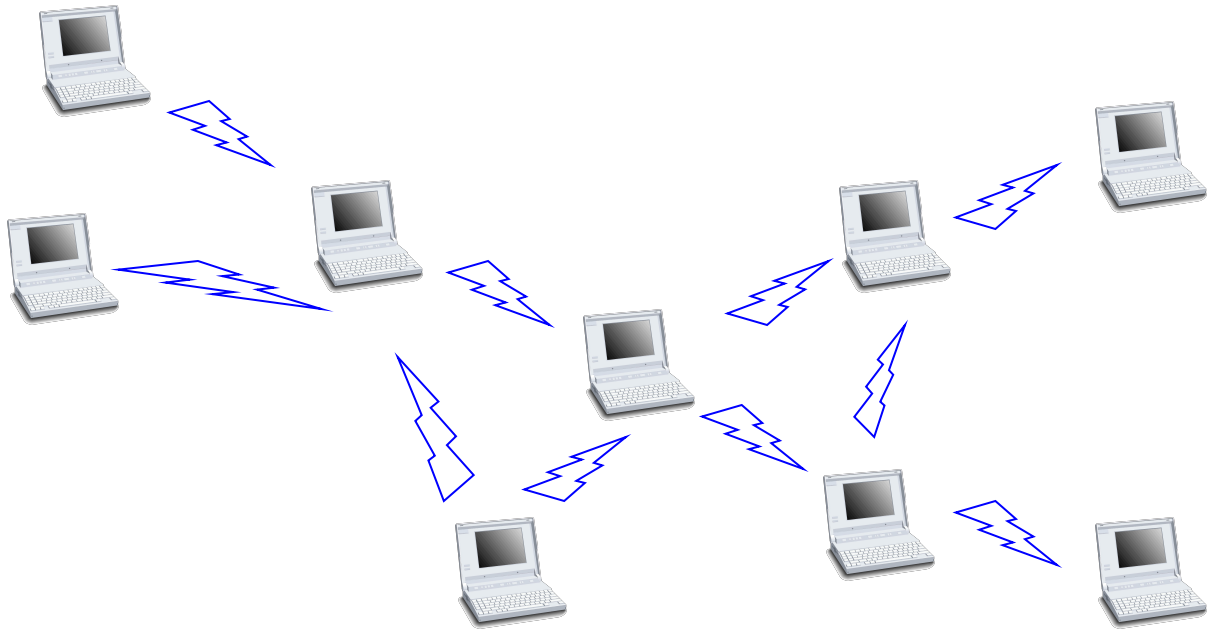
Forwarding Database	
Destination	**Port**
C	C
F	F
G	F
B	B
E	B
D	B
A	B

Wireless Networks

- Two types of wireless networks:
  a) infrastructured network:
     a) base stations are the bridges
     b) a mobile host will communicate with the nearest base station
     c) handoff is taken when a host roams from one base to another
  b) ad hoc network:
     a) infrastructureless: no fixed base stations
     b) without the assistance of base stations for communication
     c) Due to transmission range constraint,
        a) two MHs need multi-hop routing for communication
     d) quickly and unpredictably changing topology

MANET

- MANET = Mobile Ad Hoc Networks
  a) a set of mobile hosts, each with a transceiver
  b) no base stations; no fixed network infrastructure
  c) multi-hop communication
  d) needs a routing protocol which can handle changing topology



Applications of MANET

- battlefields
- nature disaster areas
- fleet in oceans
- historical cites
- festival ground

On-demand vs. Table-driven

- Table-Driven Routing Protocol:
  a) proactive!!
  b) continuously evaluate the routes
  c) attempt to maintain consistent, up-to-date routing information
     a) when a route is needed, one may be ready immediately
  d) when the network topology changes
     a) the protocol responds by propagating updates throughout the network to maintain a consistent view

- Source-Initiated On-Demand Routing Protocol:
  a) reactive!!
  b) on-demand style: create routes only when it is desired by the source node
     a) route discovery: invoke a route-determination procedure
     b) the procedure is terminated when
        a) a route has been found
        b) no route is found after all route permutations are examined
  c) longer delay: sometimes a route may not be ready for use immediately when data packets come

AODV Concepts

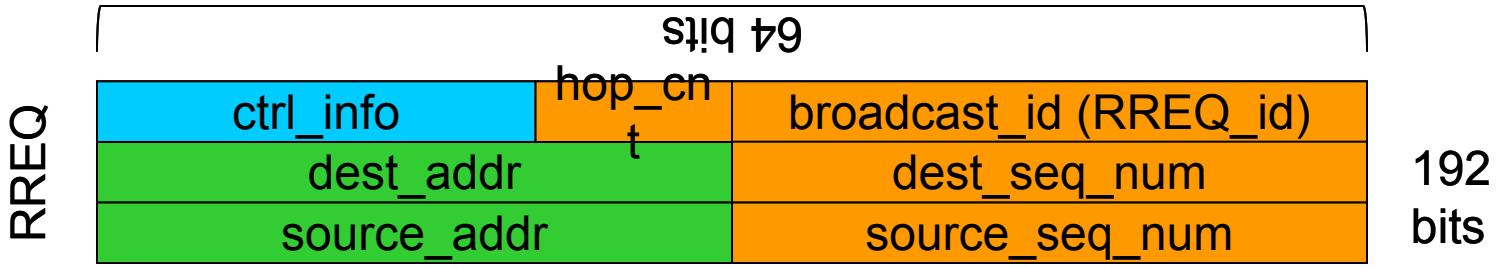
- Pure on-demand routing protocol
  a) A node does not perform route discovery or maintenance until it needs a route to another node or it offers its services as an intermediate node
  b) Nodes that are not on active paths do not maintain routing information and do not participate in routing table exchanges.
  c) Routes are based on dynamic table entries maintained at intermediate nodes

RREQ-Path discovery (1/5)

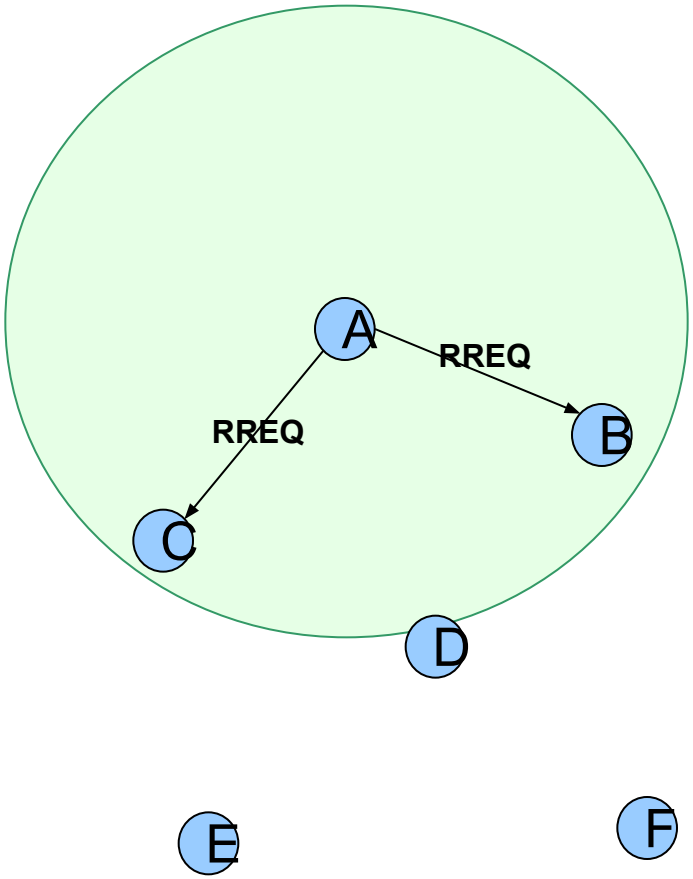
- Initiated, when node needs to communicate with new node (no routing information in table)
- Route Request (RREQ) packet is broadcasted to network
- An expanding ring search should be used

a) TTL (Time to live) parameter in IP-header sets the lifetime in hops for packets

b) TTL is first small and is then increased, if a route is not found until a limit is reached

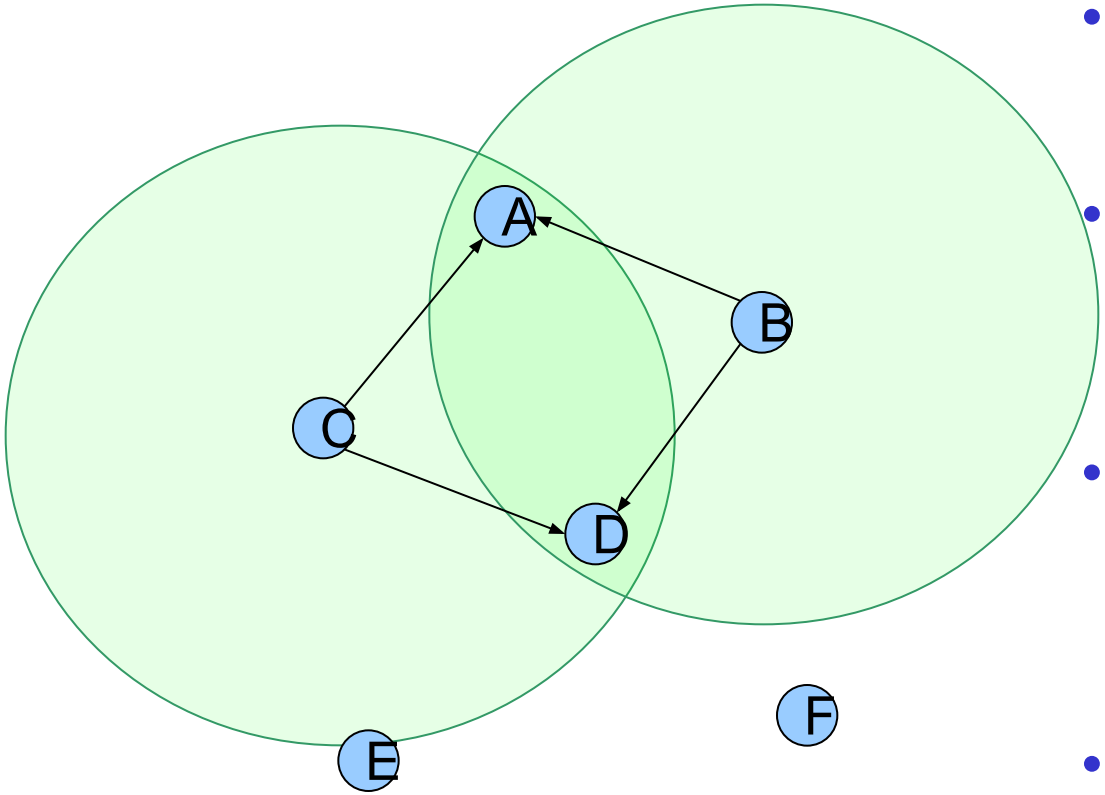


Path discovery (2/5)



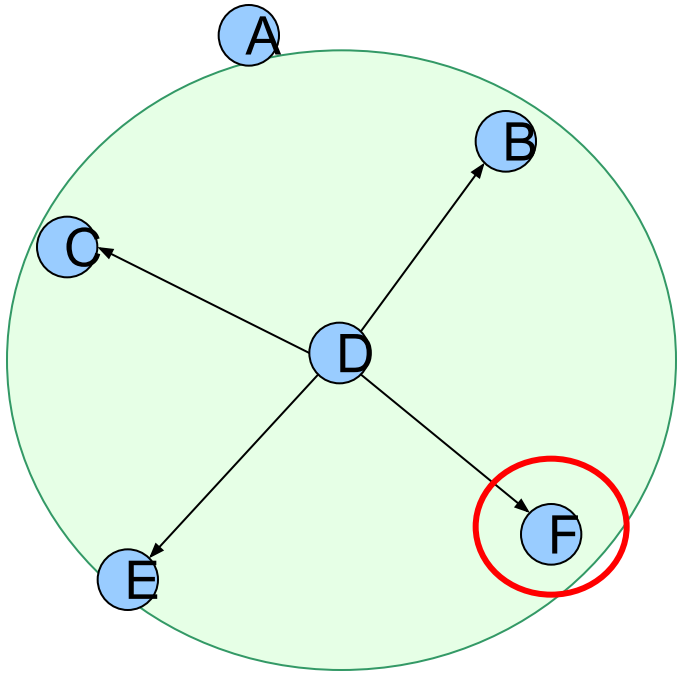
- Example: Node A needs to communicate with F
- RREQ A->F is released to network
- Neighbors C and B receive RREQ and learn route to A

Path discovery (3/5)



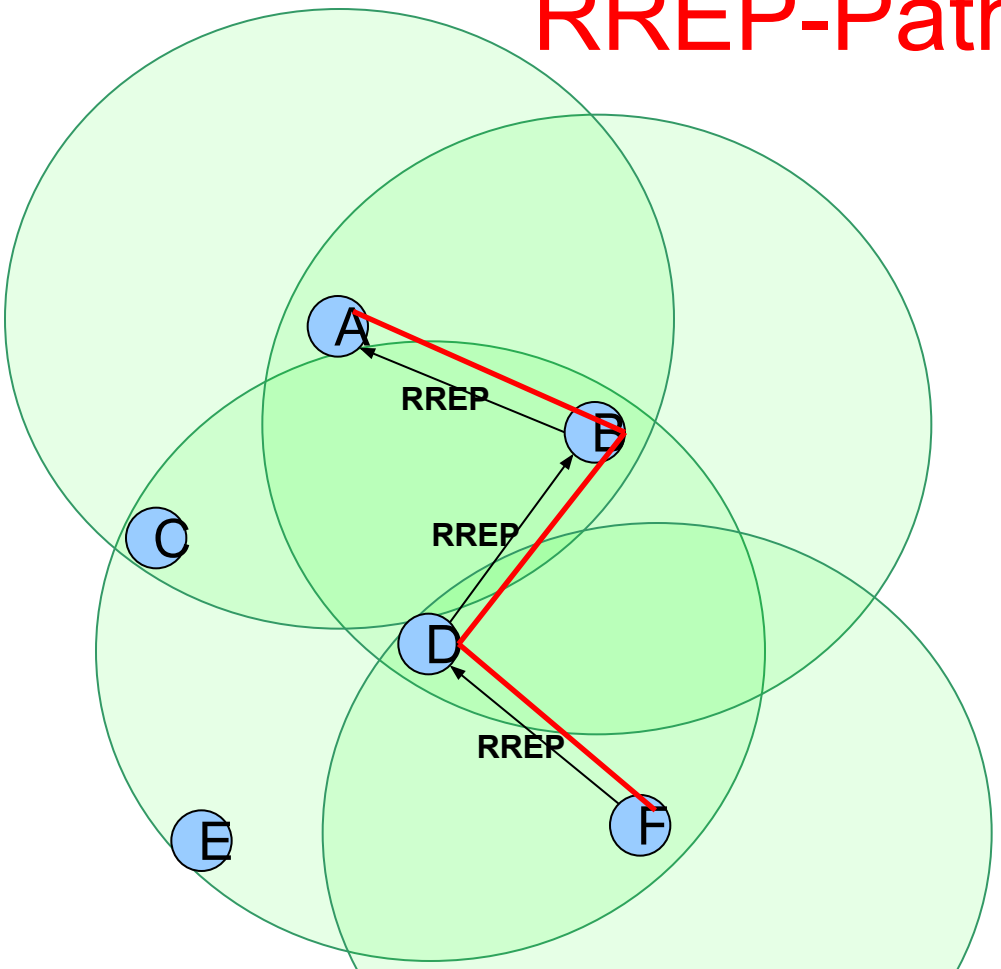
- Intermediate nodes C and B do not have route to F
- RREQ is broadcasted forward with increased *hop count* only if hop limit is not yet reached
- A receives it's own RREQ
  - a) *paths* to B and C are formed
  - b) RREQ is discarded
- Intermediate node D receives multiple copies of RREQ form A
  - a) Direct routes to C and B are formed
  - b) The first arrived RREQ is set used to form route to A (e.g. B here)

Path discovery (4/5)

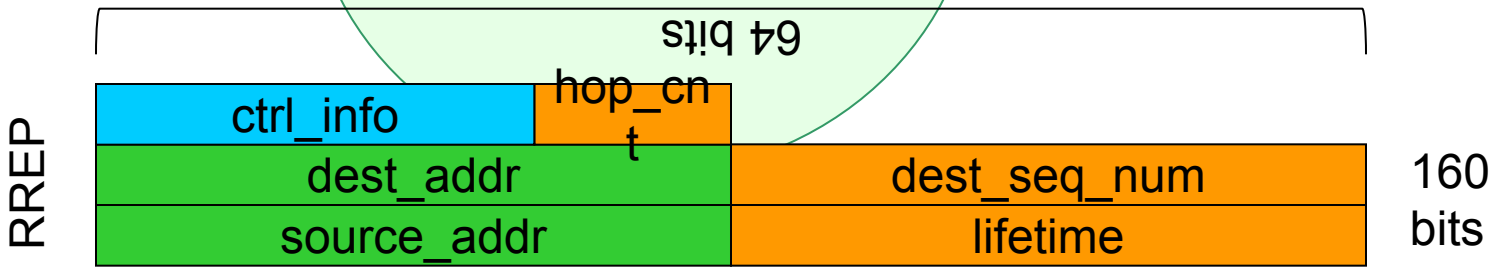


- D forwards RREQ
- B and C discard duplicate RREQ and learn route to D
- Destination node F finally gets RREQ

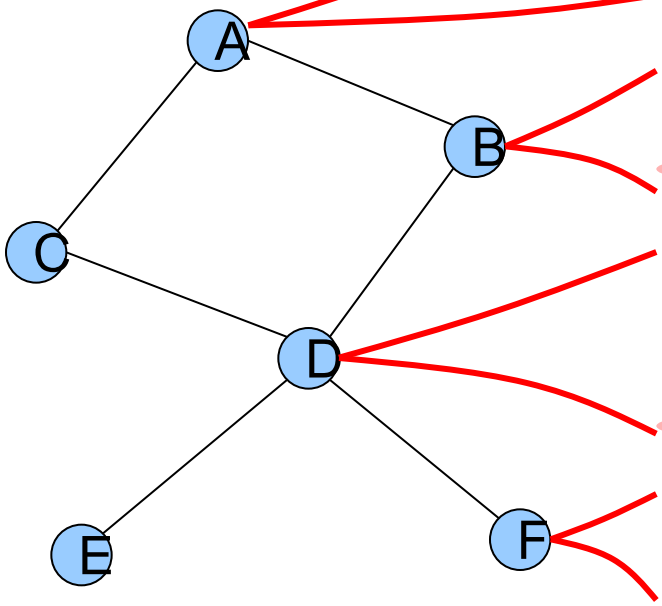
RREP-Path discovery (5/5)



- Route reply packet (RREP) is sent back to node A along reverse route
  a) In fact any node, which has a fresh route to destination can send RREP and therefore end route search
- Active *forward path* from A to F is created
  a) Intermediate nodes also have now active *forward path* to F
- Route is ready for data transmission



Routing tables



Dest	Nex hop	Num hops	Exp time	Seq num dest	Active neighbors
B	B	1	10 s		
C	C	1	3 s		
F	B	3	20 s		

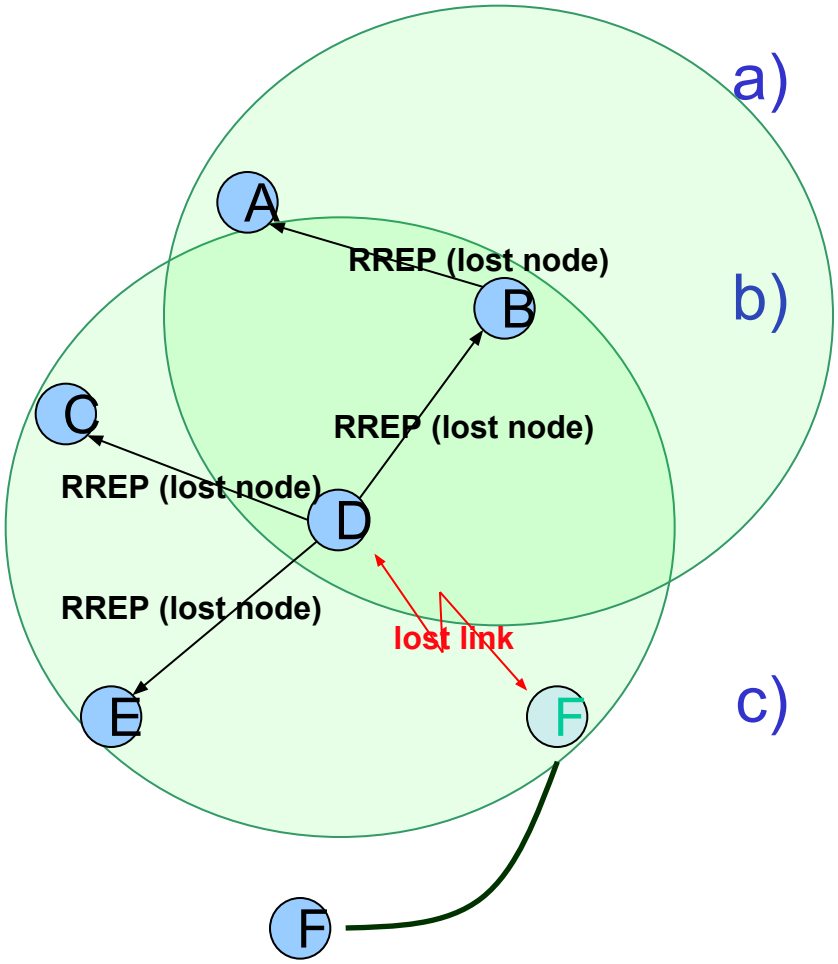
Dest	Nex hop	Num hops	Exp time	Seq num dest	Active neighbors
A	A	1	10 s		
D	D	1	3 s		
F	D	2	20 s		

Dest	Nex hop	Num hops	Exp time	Seq num dest	Active neighbors
C	C	1	3 s		
E	E	1	3 s		
A	B	2	10 s		
B	B	1	10 s		
F	F	1	20 s		

Dest	Nex hop	Num hops	Exp time	Seq num dest	Active neighbors
D	D	1	10 s		
A	D	3	10 s		

- Route expiration times are updated when route is used
- Routes in AODV are in fact virtual routes!
  a) a single node do not know the complete route and therefore "route control" is distributed

Path maintenance (original AODV)



a) Example: Link D<->F is broken due to movement of F

b) A special RREP(Route Error-RERR) is send upstream in path (now B->A)

   a) hop_count to lost destination (now F) is set to ∞

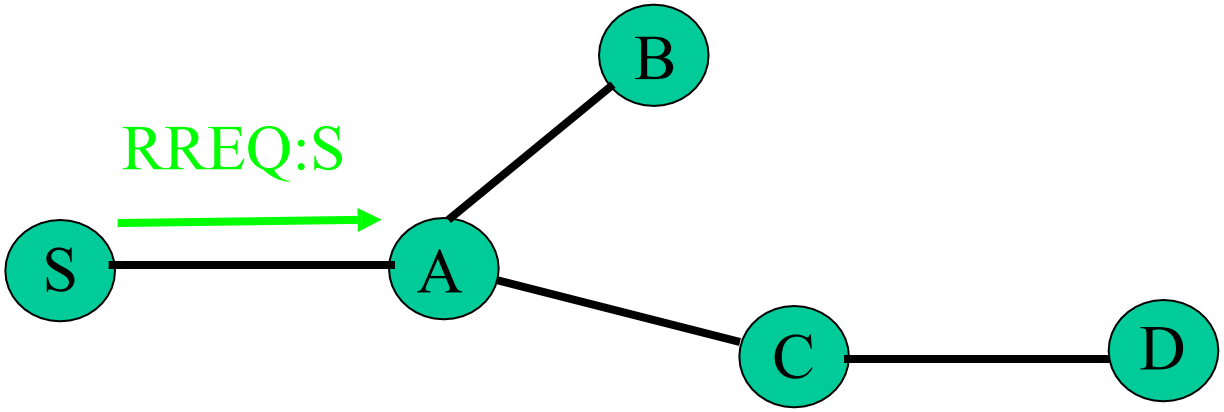
c) From RREP node A sees a broken link towards F

   a) New route is searched as in initial route search (RREQ) if needed

Dynamic Source Routing (DSR) - Introduction

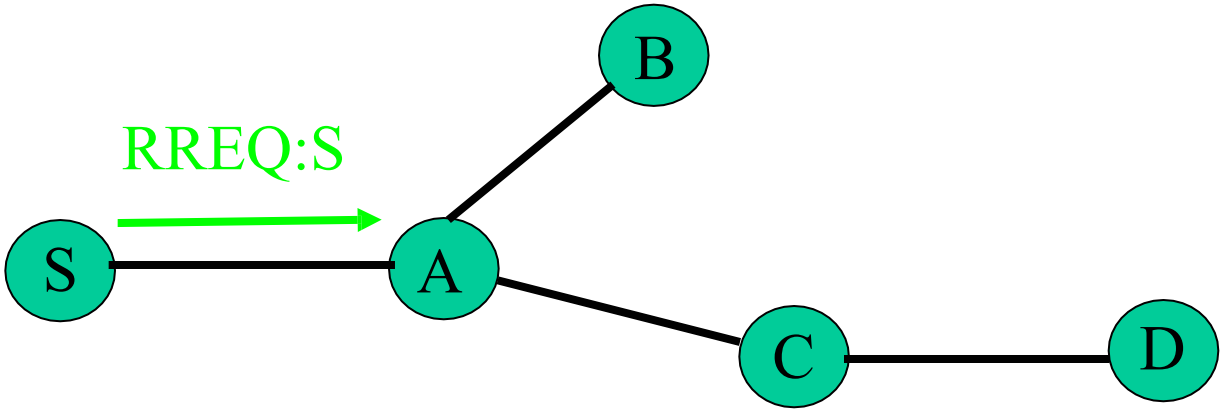
- Reactive or On Demand Developed at CMU in 1996
- Route discovery cycle used for route finding – on Demand
- Utilizes source routing (entire route is part of the header)
- When node S wants to send a packet to node D, but does not know a route to D, node S initiates a route discovery
- Source node S floods Route Request (RREQ)
- Each RREQ, has sender's address, destination's address, and a unique Request ID determined by the sender
- Each node appends own identifier when forwarding RREQ

DSR – Route Discovery



1. Node S needs a route to D
2. Broadcasts RREQ packet

DSR – Route Discovery



1. Node S needs a route to D
2. Broadcasts RREQ packet
3. Node A receives packet, has no route to D
   Rebroadcasts packet after adding its address to source route

Route Discovery – Node Actions

Upon receiving a RREQ, the node takes the following actions:

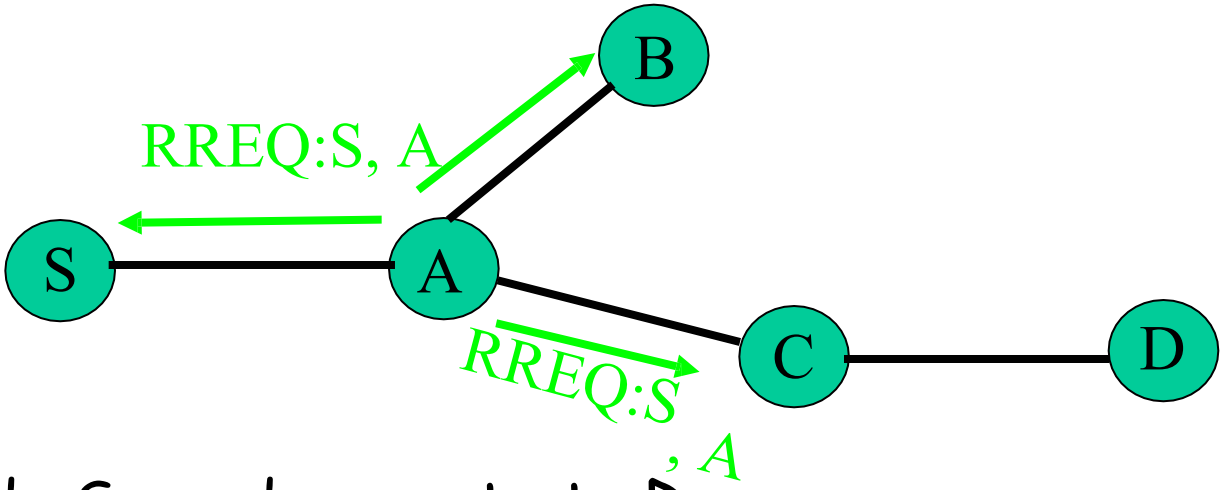
1. if the node is the Target (Destination)

Returns a Route Reply (RREP) message to the sender

Copies the accumulated route record from RREQ into RREP

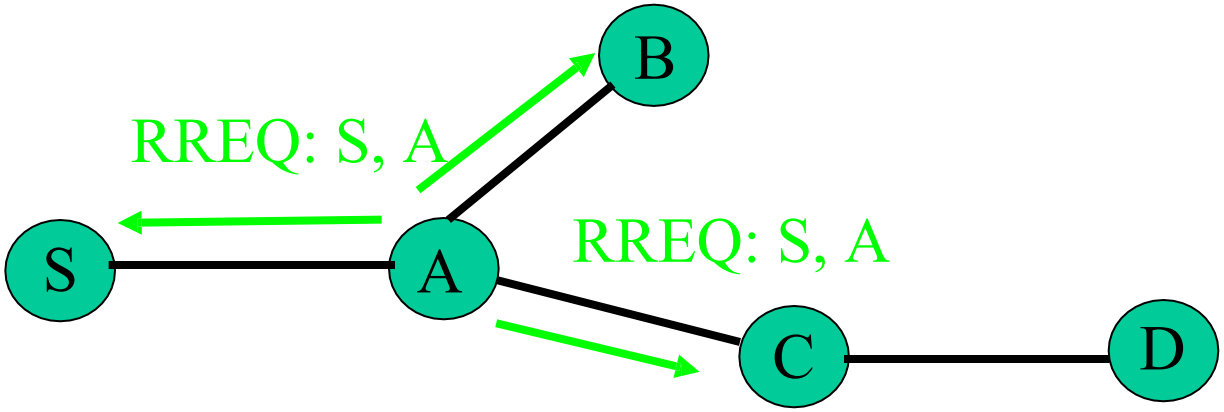
Sender upon receiving RREP, caches the route in its route cache for subsequent routing

DSR – Route Discovery



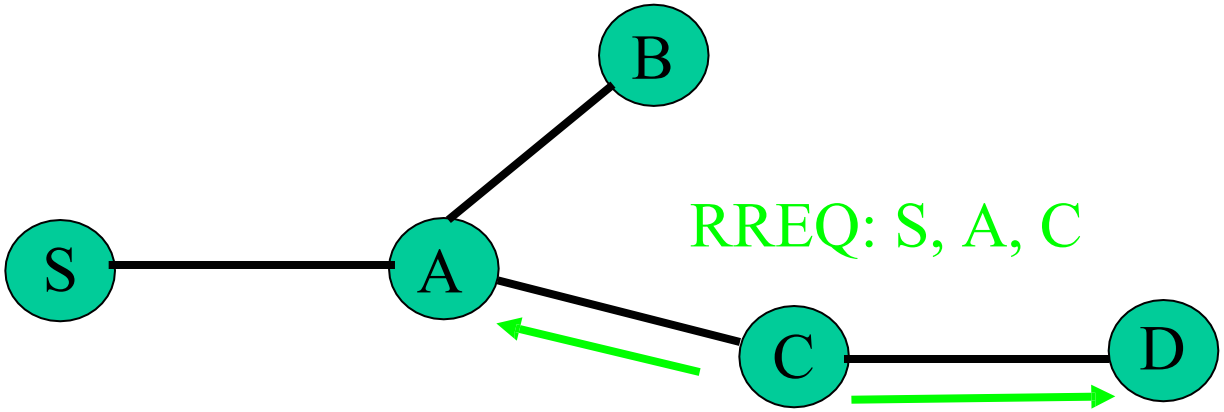
1. Node S needs a route to D
2. Broadcasts RREQ packet
3. Node A receives packet, has no route to D
   Rebroadcasts packet after adding its address to source route

DSR – Route Discovery



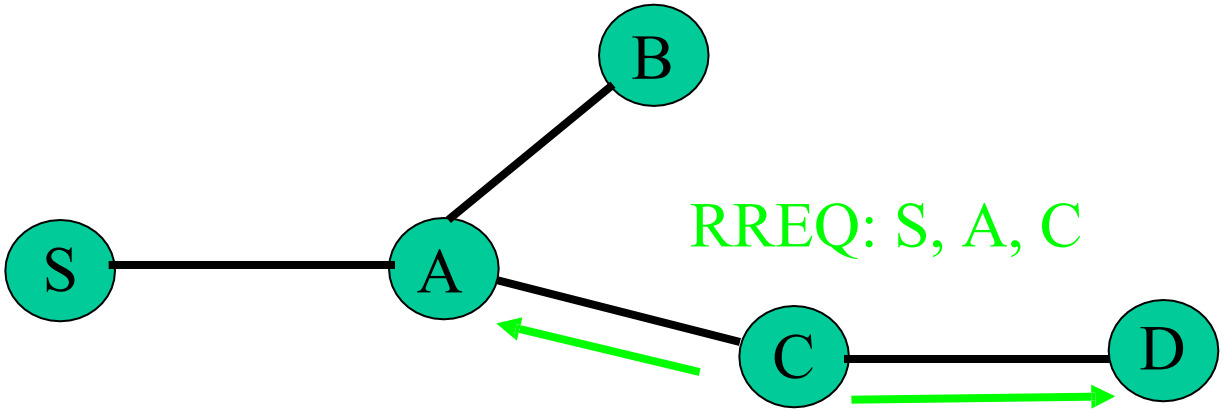
4. Node C receives RREQ, has no route to D
   - Rebroadcasts packet after adding its address to source route

DSR – Route Discovery



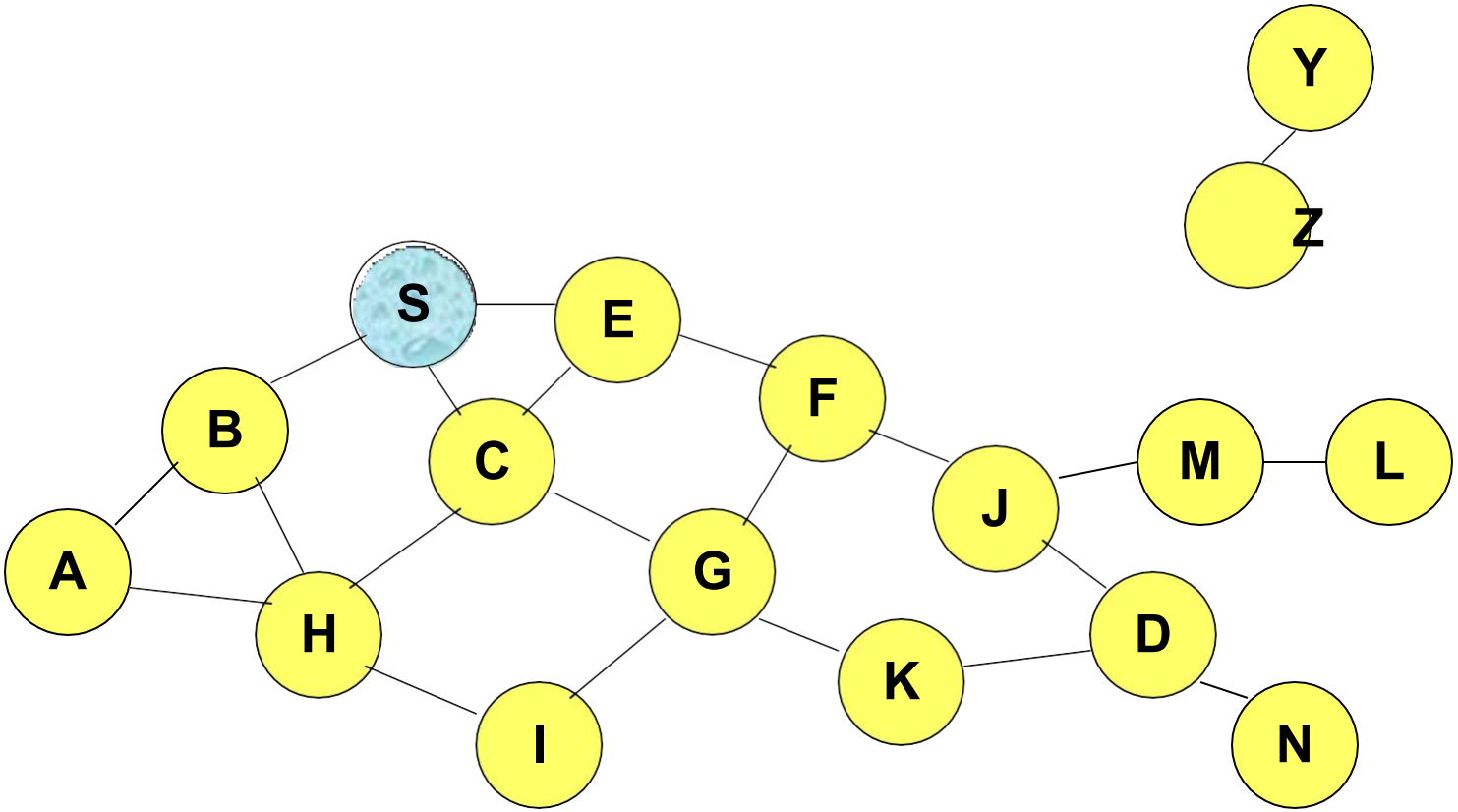
4. Node C receives RREQ, has no route to D
   - Rebroadcasts packet after adding its address to source route

DSR – Route Discovery



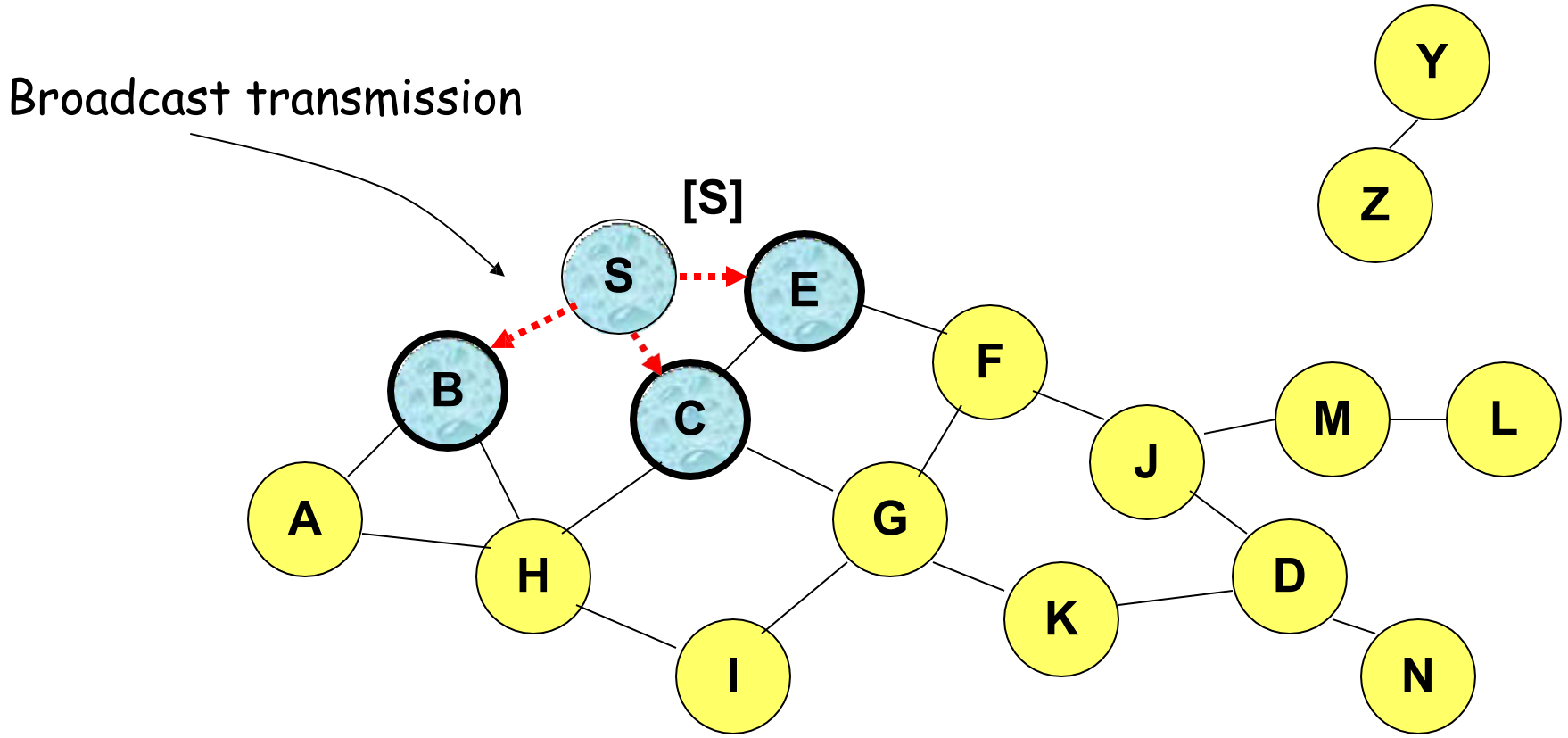
4. Node C receives RREQ, has no route to D
   Rebroadcasts packet after adding its address to source route
5. Node D receives RREQ, unicasts RREP to C
   D puts entire route in RREP message

Route Discovery in DSR



Represents a node that has received RREQ for D from S

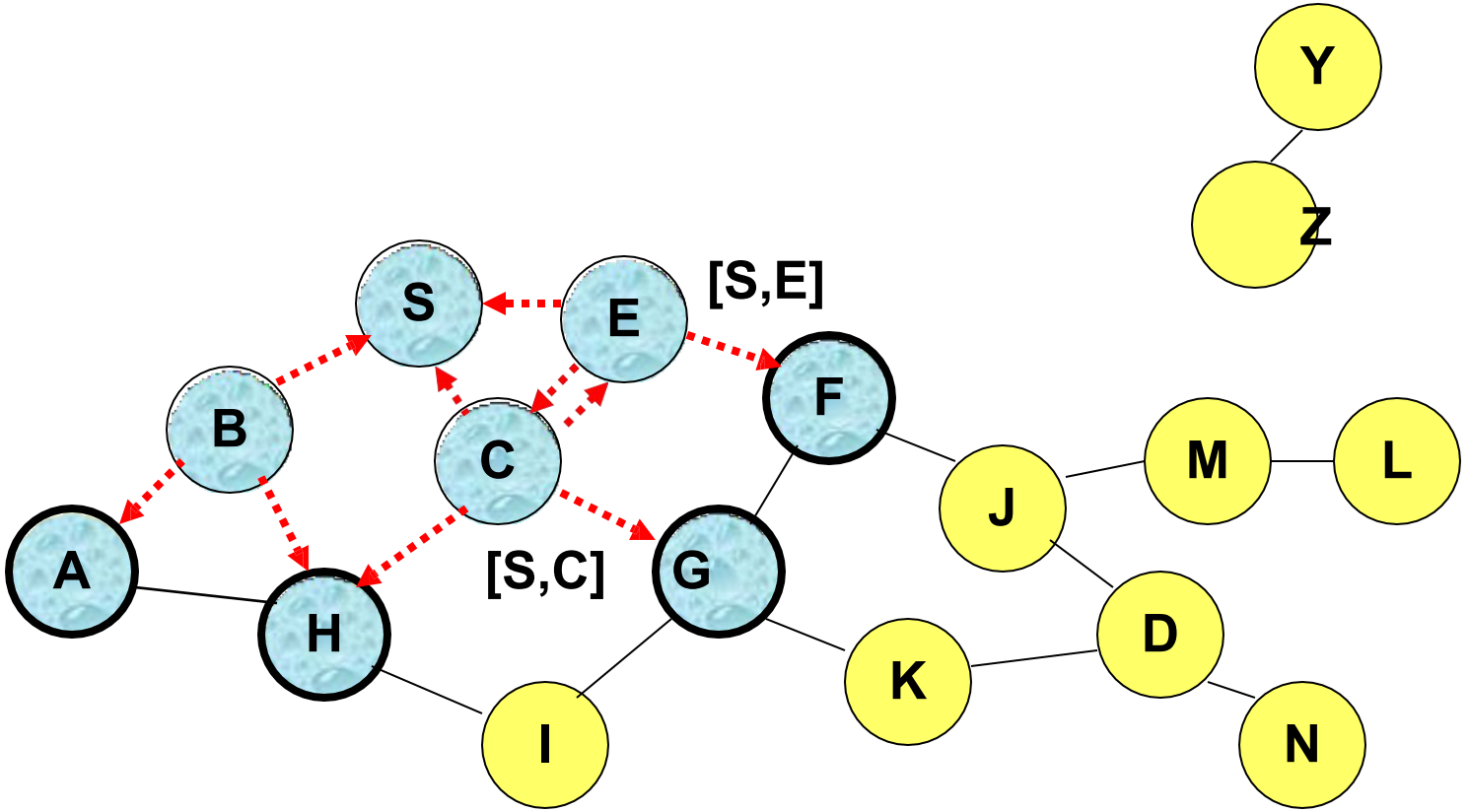
Route Discovery in DSR



Represents transmission of RREQ

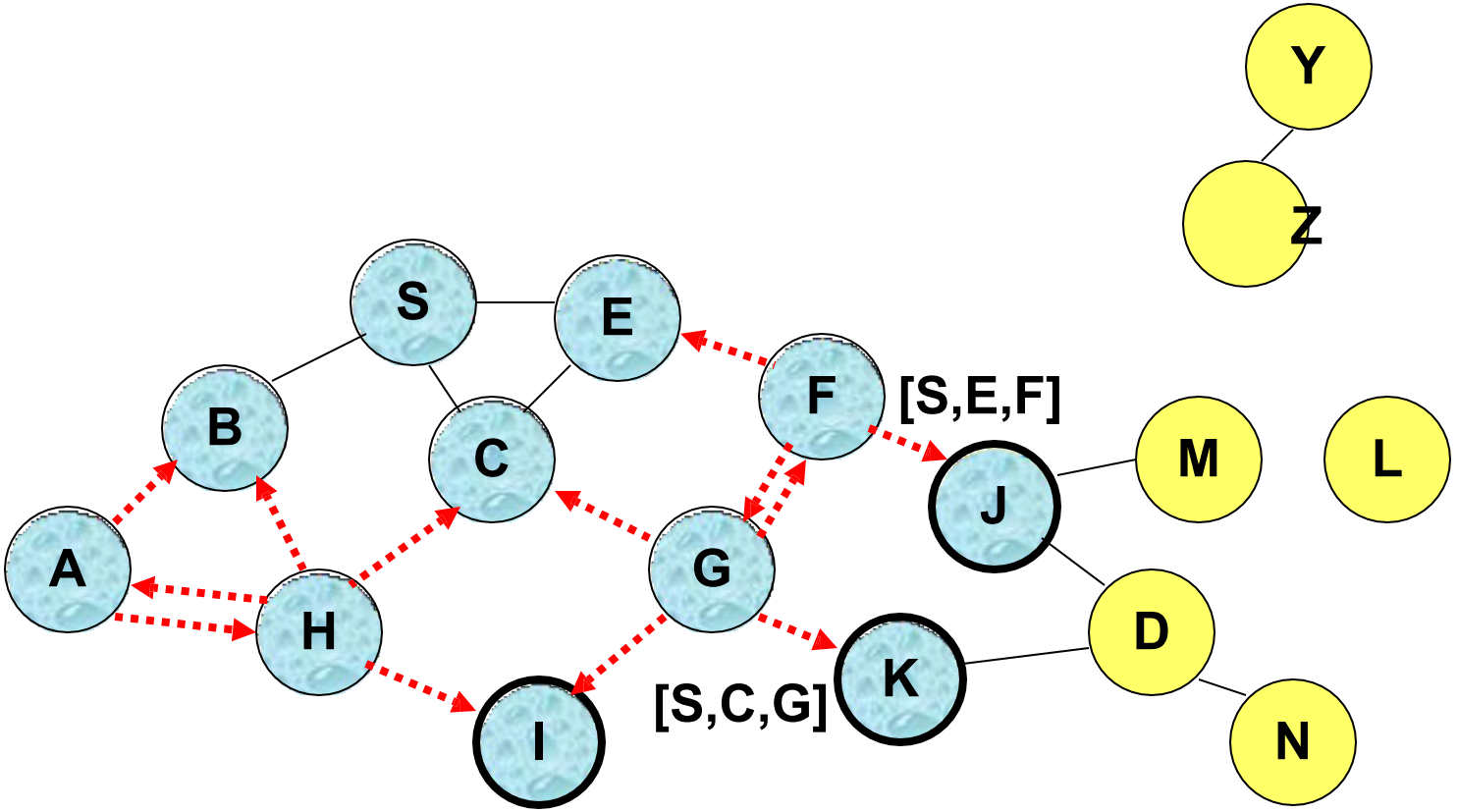
[X,Y] Represents list of identifiers appended to RREQ

Route Discovery in DSR



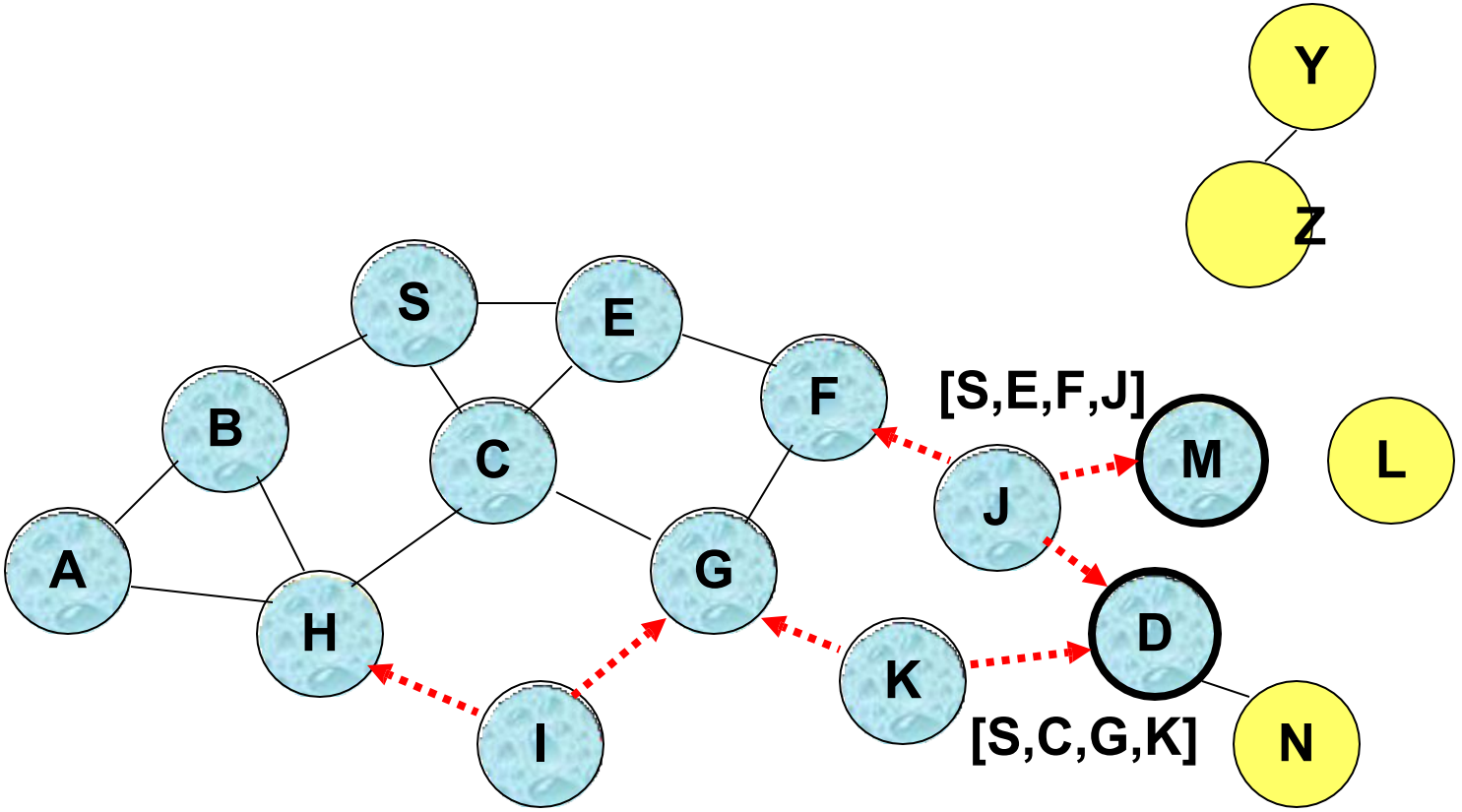
- Node H receives packet RREQ from two neighbors: potential for collision

Route Discovery in DSR



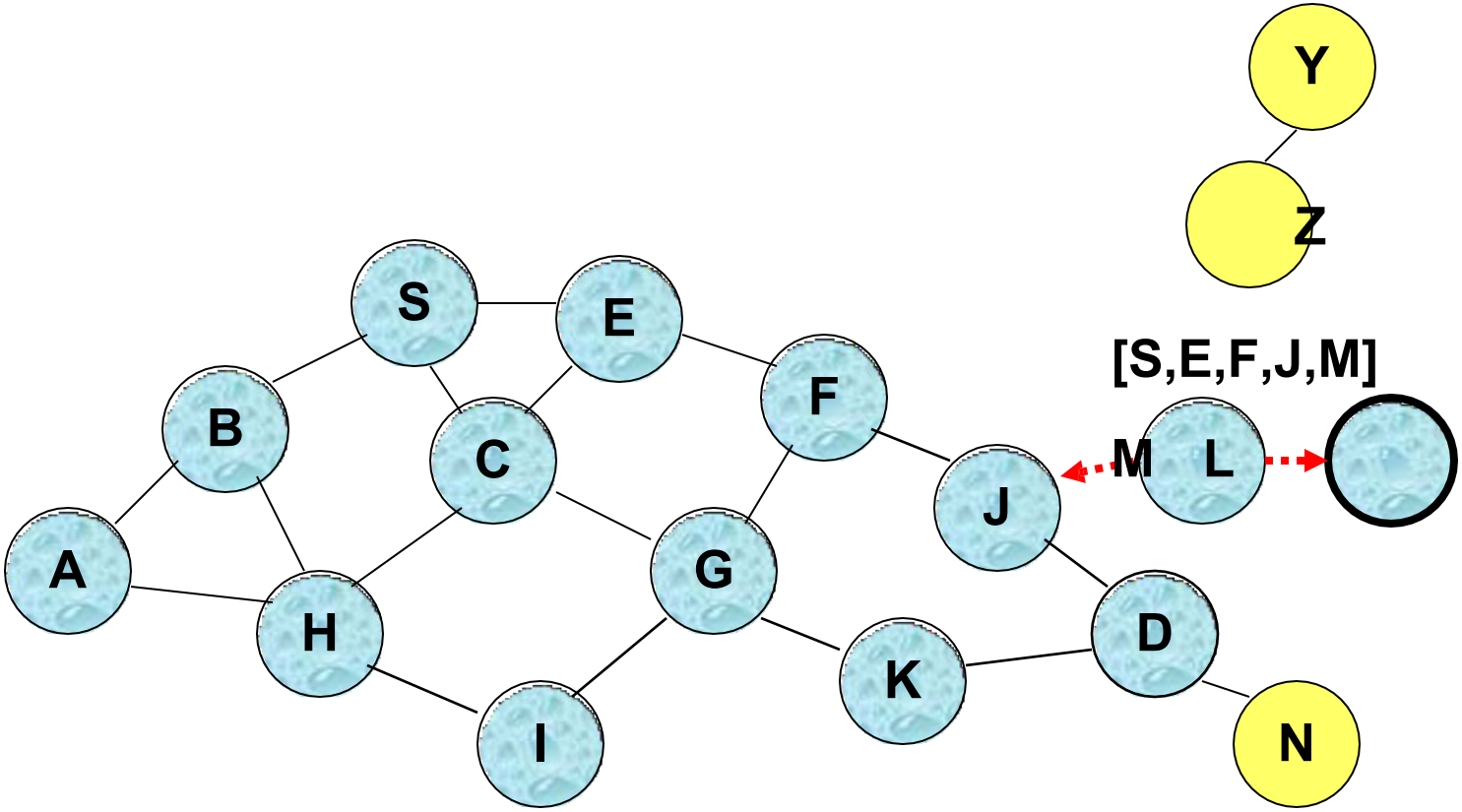
- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

Route Discovery in DSR



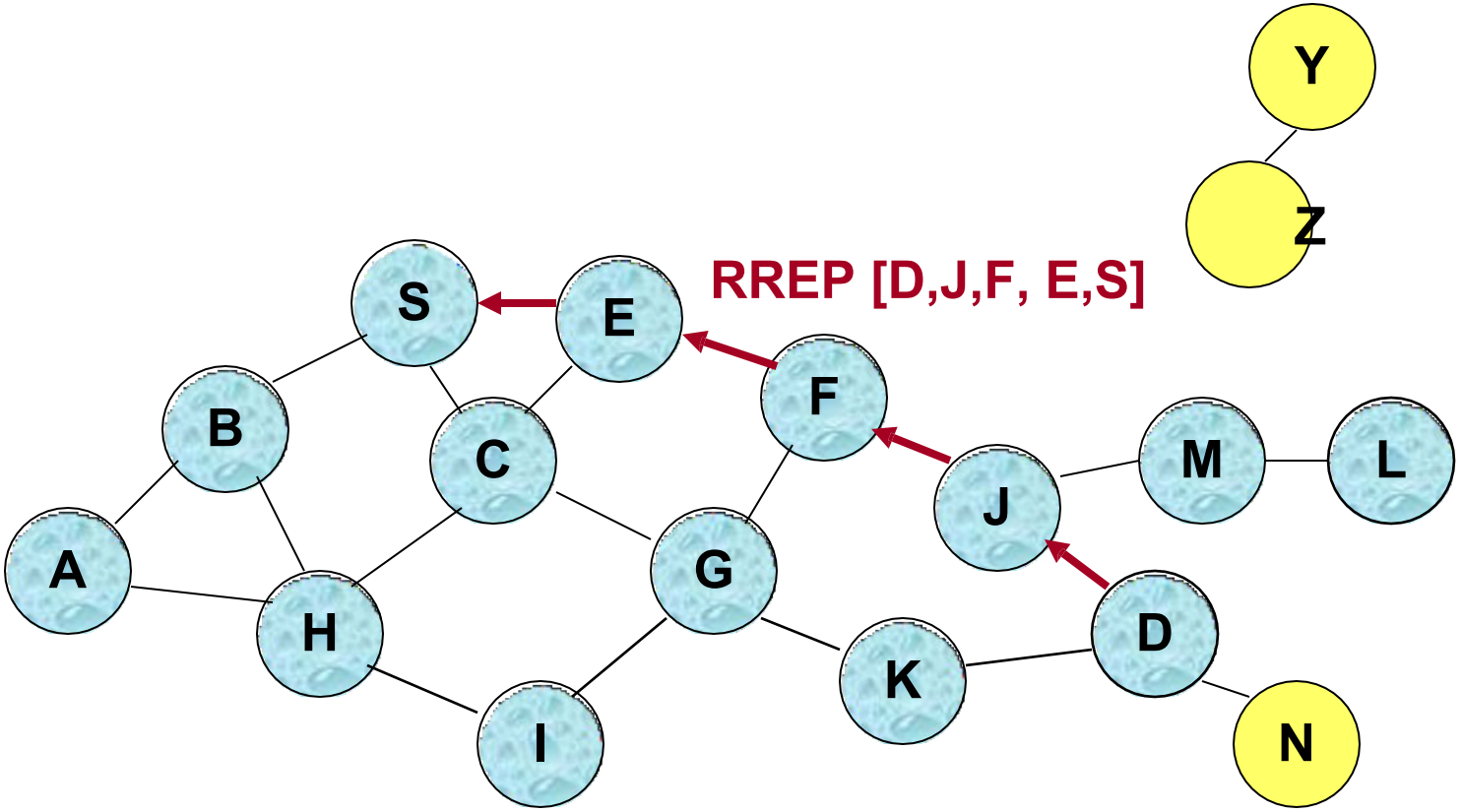
- Nodes J and K both broadcast RREQ to node D

Route Discovery in DSR



- Node D does not forward RREQ, because node D is the intended target of the route discovery

Route Reply in DSR



← Represents RREP control message

Dynamic Source Routing (DSR)

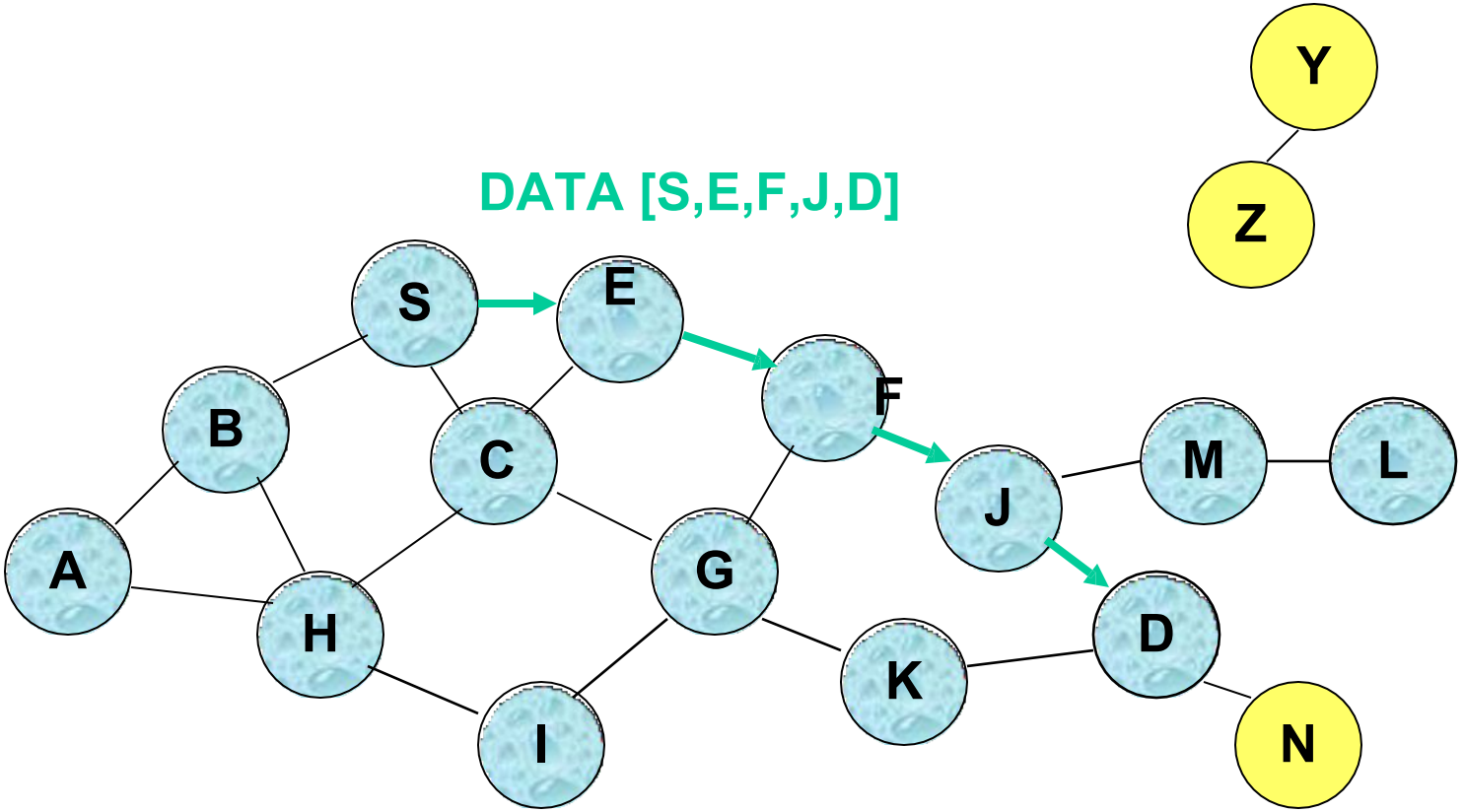
Node S on receiving RREP, caches the route included in the RREP

When node S sends a data packet to D, the entire route is included in the packet header

hence the name source routing

Intermediate nodes use the source route included in a packet to determine to whom a packet should be forwarded

Data Delivery in DSR



- Packet header size grows with route length