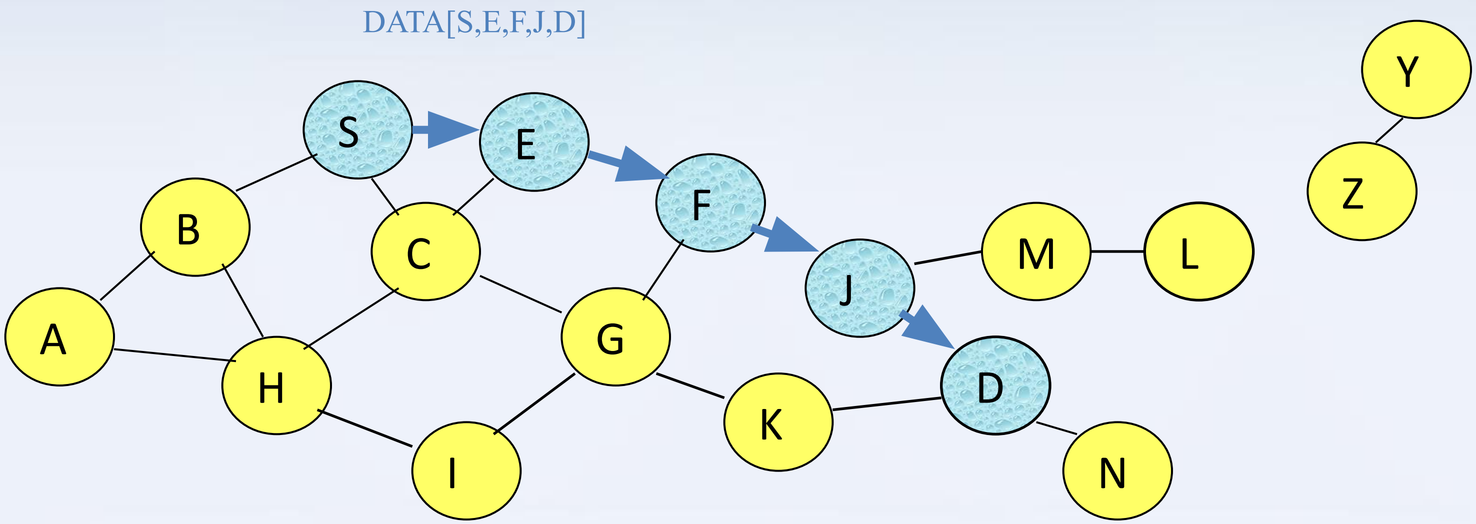
# Proactive vs. Reactive

- Latency of route discovery
  - Proactive protocols have lower latency since routes are maintained all times
  - Reactive protocols have higher latency because a node needs to find a route when it has data to send
- Overhead of route maintenance
  - Reactive protocols have lower overhead since routes are maintained only if they are needed
  - Proactive protocols have higher overhead due to continuous route updating

# Reactive Protocols

- Dynamic Source Routing (DSR)
- Ad Hoc On-Demand Distance Vector (AODV)
- Temporally Ordered Routing Algorithm(TORA)

# Dynamic Source Routing (DSR)




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

- Intermediate nodes use the source route embedded in the packet's header to determine to whom the packet should be forwarded

# DSR

- Two basic mechanisms
  - Route Discovery
    - Route Request (RREQ)
    - Route Reply (RREP)
  - Route Maintenance
    - Route Error (RERR)
- Key optimization
  - Each node maintains a route cache

# Route Discovery

- Every route request packet (RREQ) contains
  <Dest ID, Src ID, route record, request ID>

- Each node maintains a list of the < Src address, request ID>

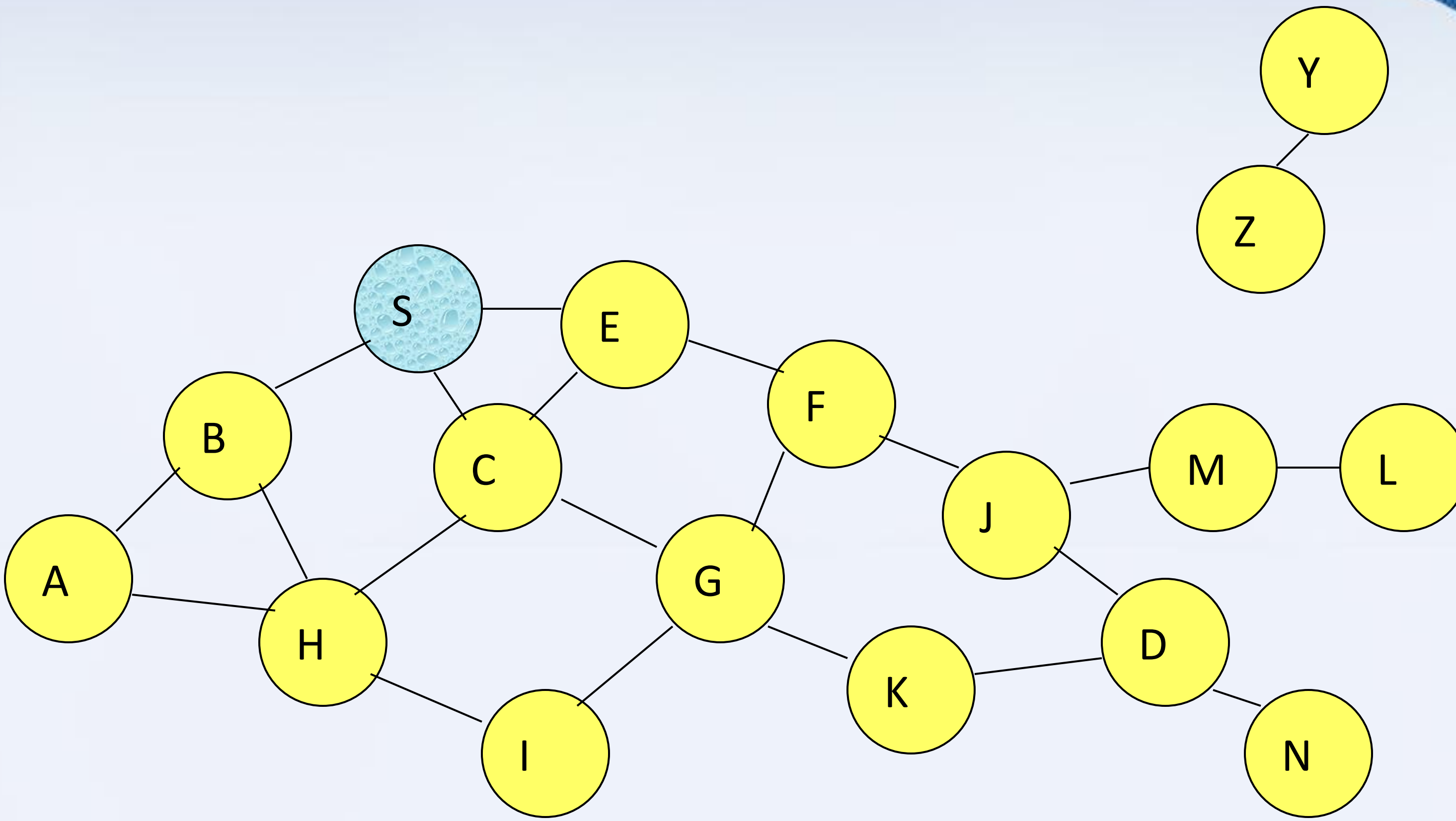
- When a node S receives a RREQ

  - Discards the route request packet
    - if < Src ID, request ID> is in its list

  - Return a route reply packet which contains a route from S to D
    - If D is dest
    - If D has an entry in its route cache for a route to dest

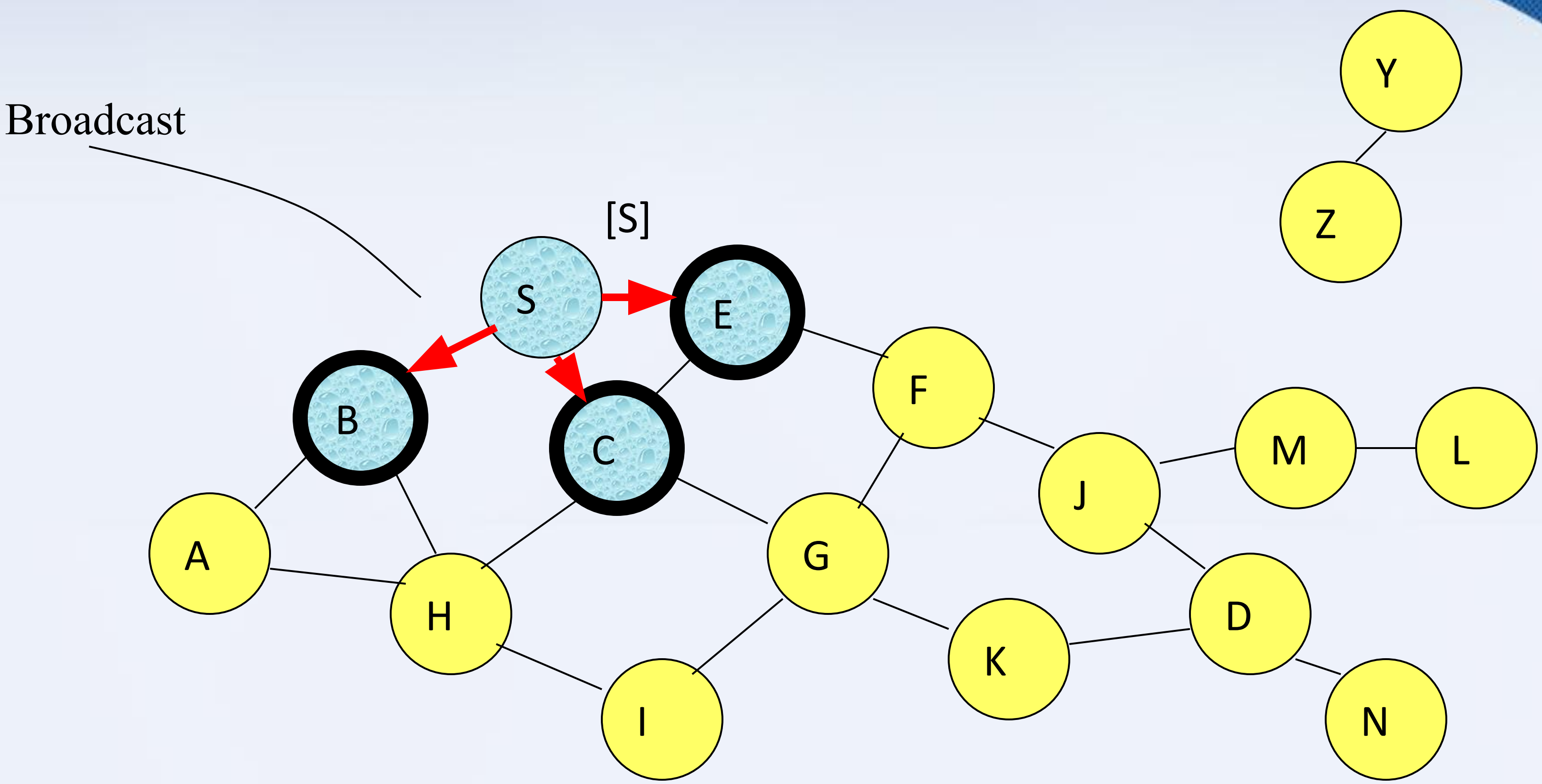
  - Append itself address to the route record in RREQ and re-broadcast RREQ

# Route Discovery in DSR




Represents a node that has received RREQ for D from S

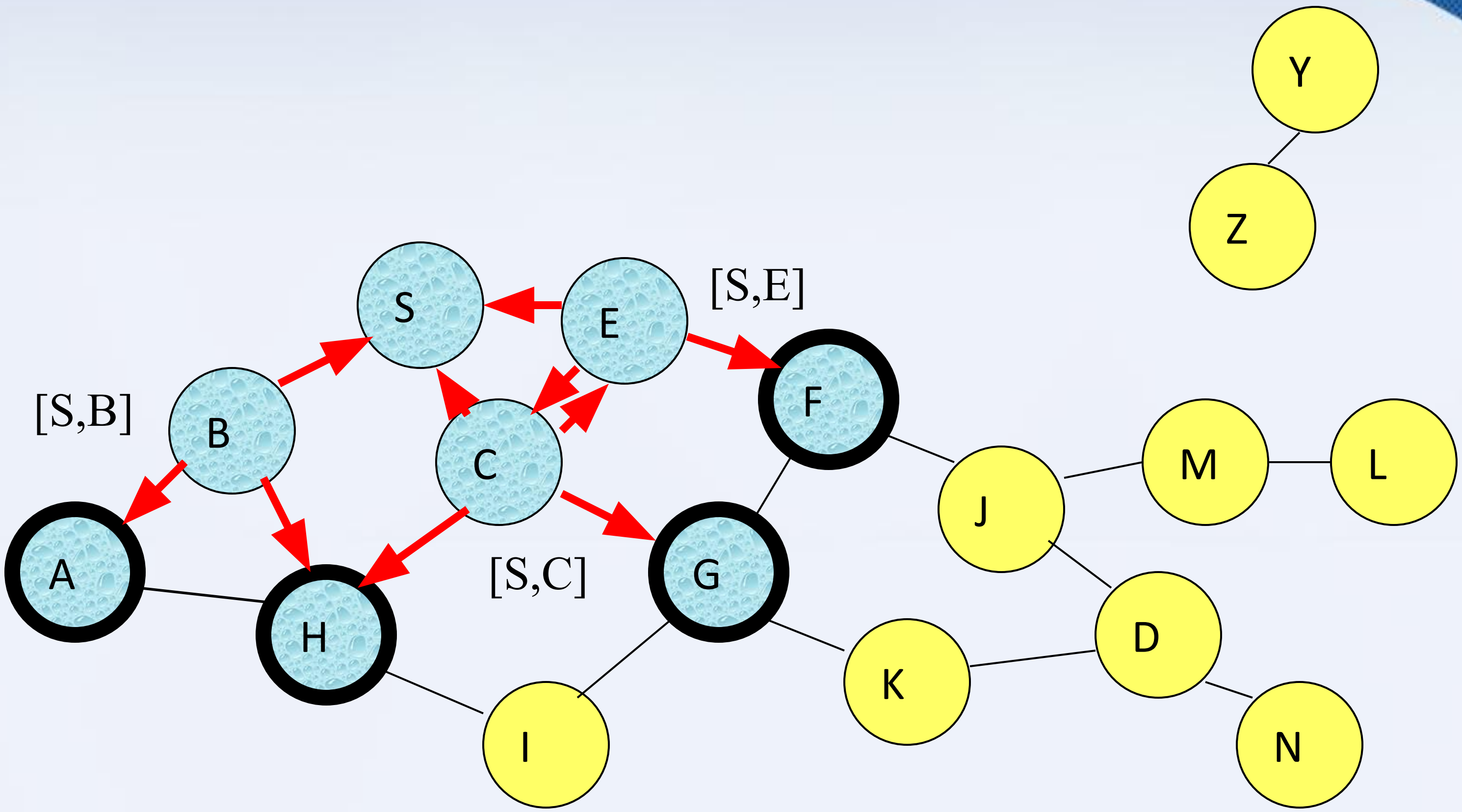
# Route Discovery in DSR




Represents transmission of RREQ

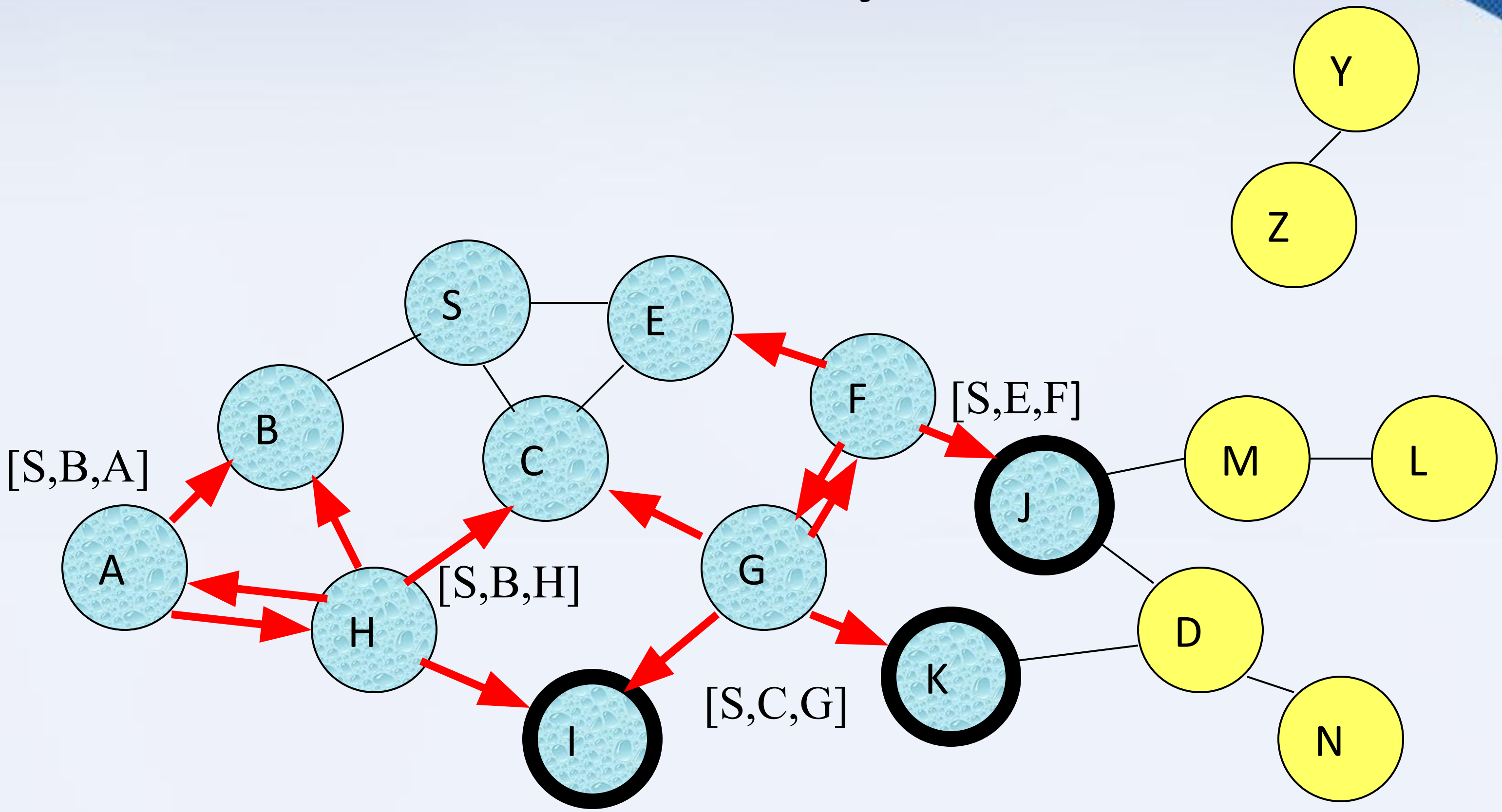
[X,Y] Represents route record stored in RREQ

# Route Discovery in DSR



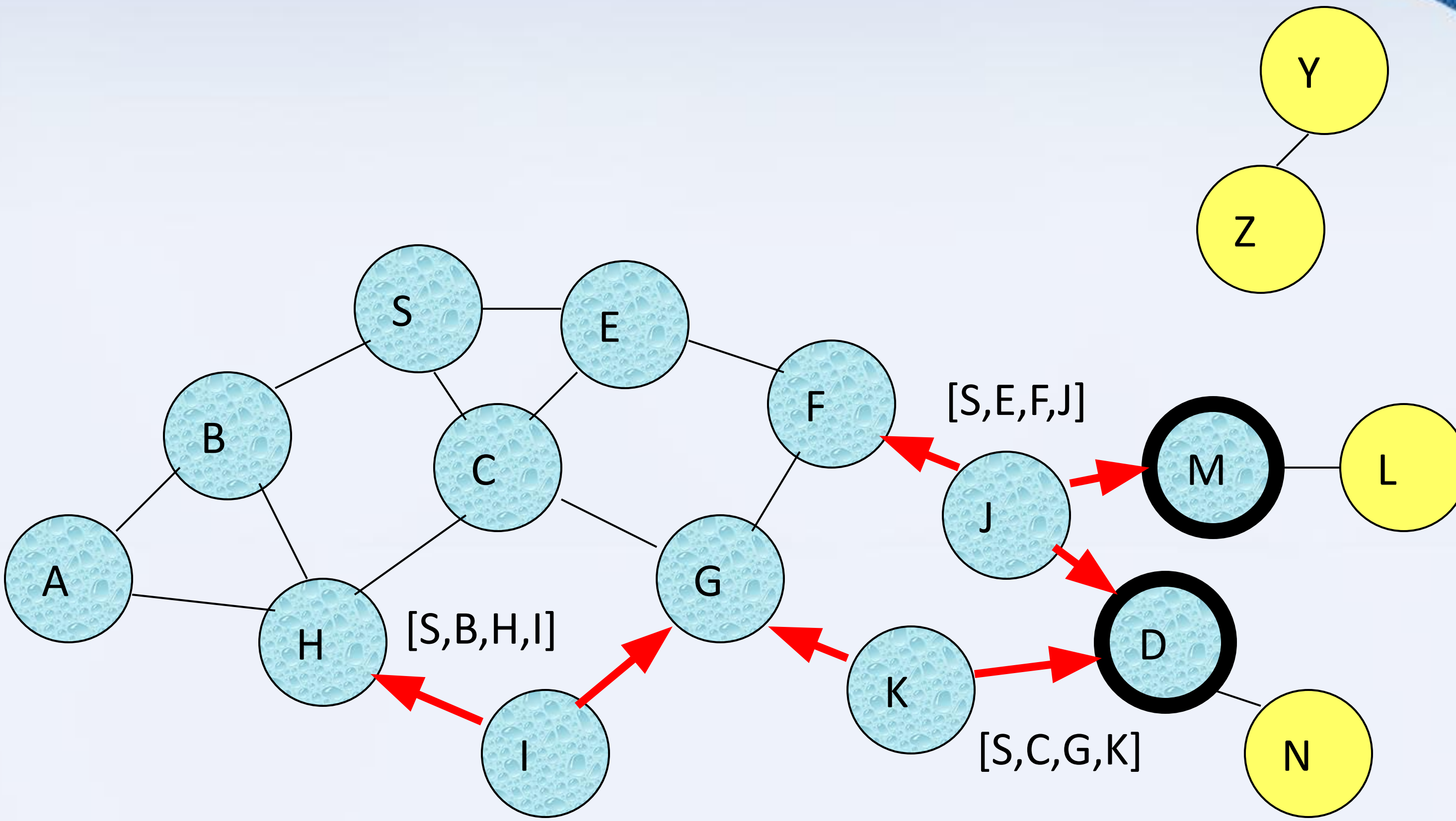

- Node H receives packet RREQ from two neighbors: potential for collision

# Route Discovery in DSR



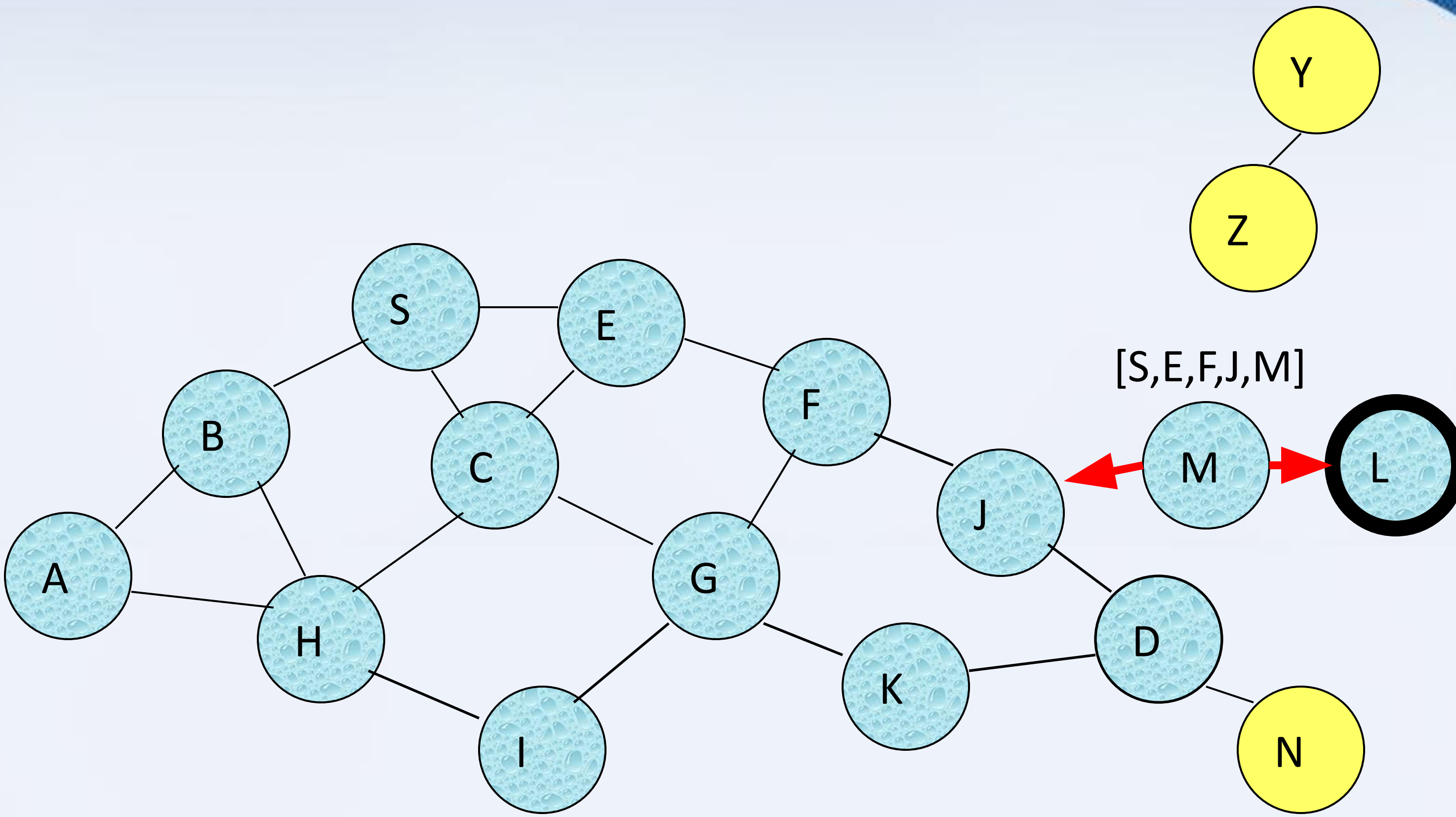

- C receives RREQ from G and H, but does not forward it again, because C has already forwarded RREQ once

# Route Discovery in DSR




J and K both broadcast RREQ to D
Their transmissions may collide at D

# Route Discovery in DSR

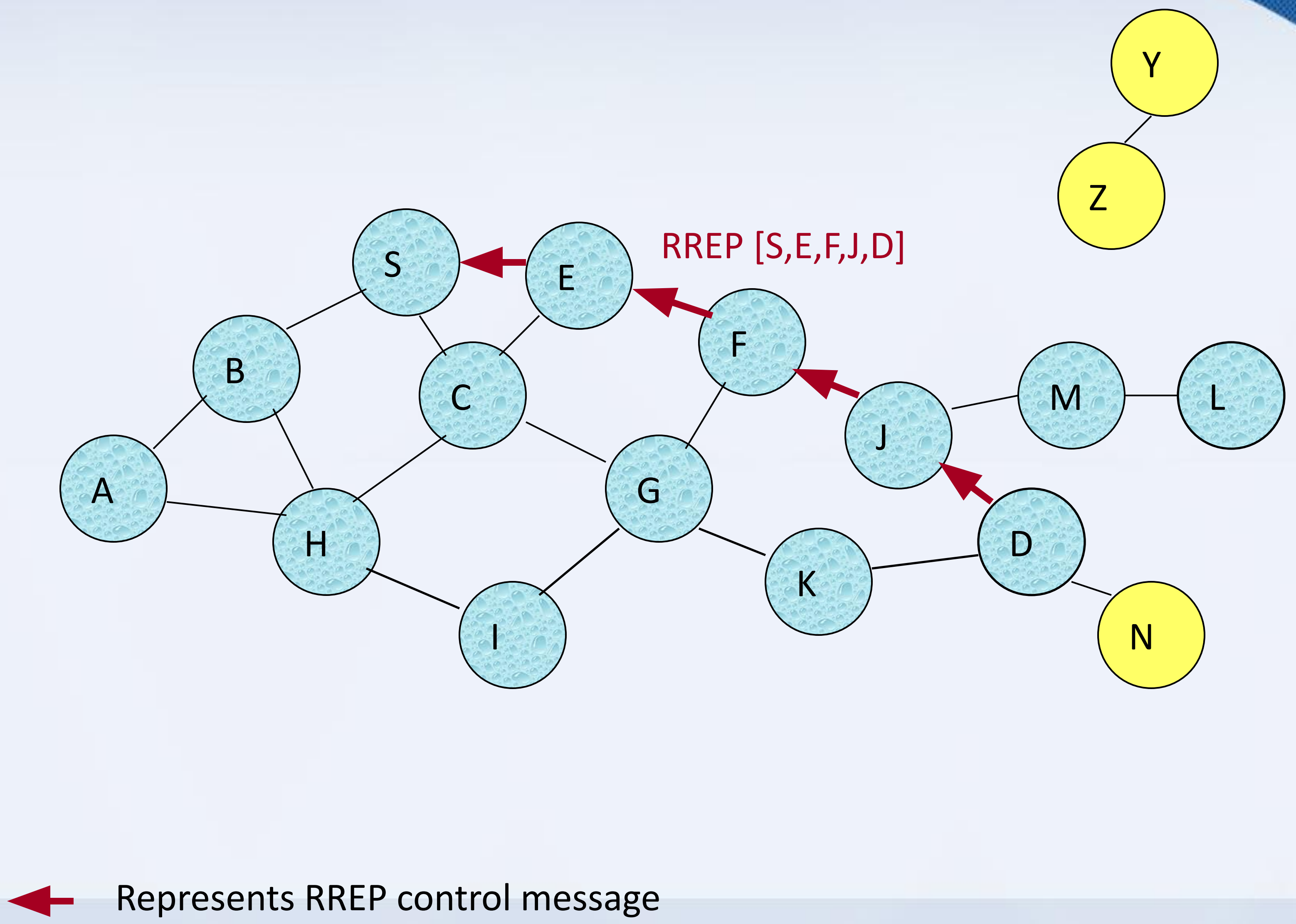



D does not forward RREQ, because D is the intended target

# Route Reply in DSR

- Destination D on receiving the first RREQ, sends a Route Reply (RREP)
- RREP includes the route from S to D
- Route Reply can be sent by reversing the route in Route Request (RREQ)
    - If links are bi-directional

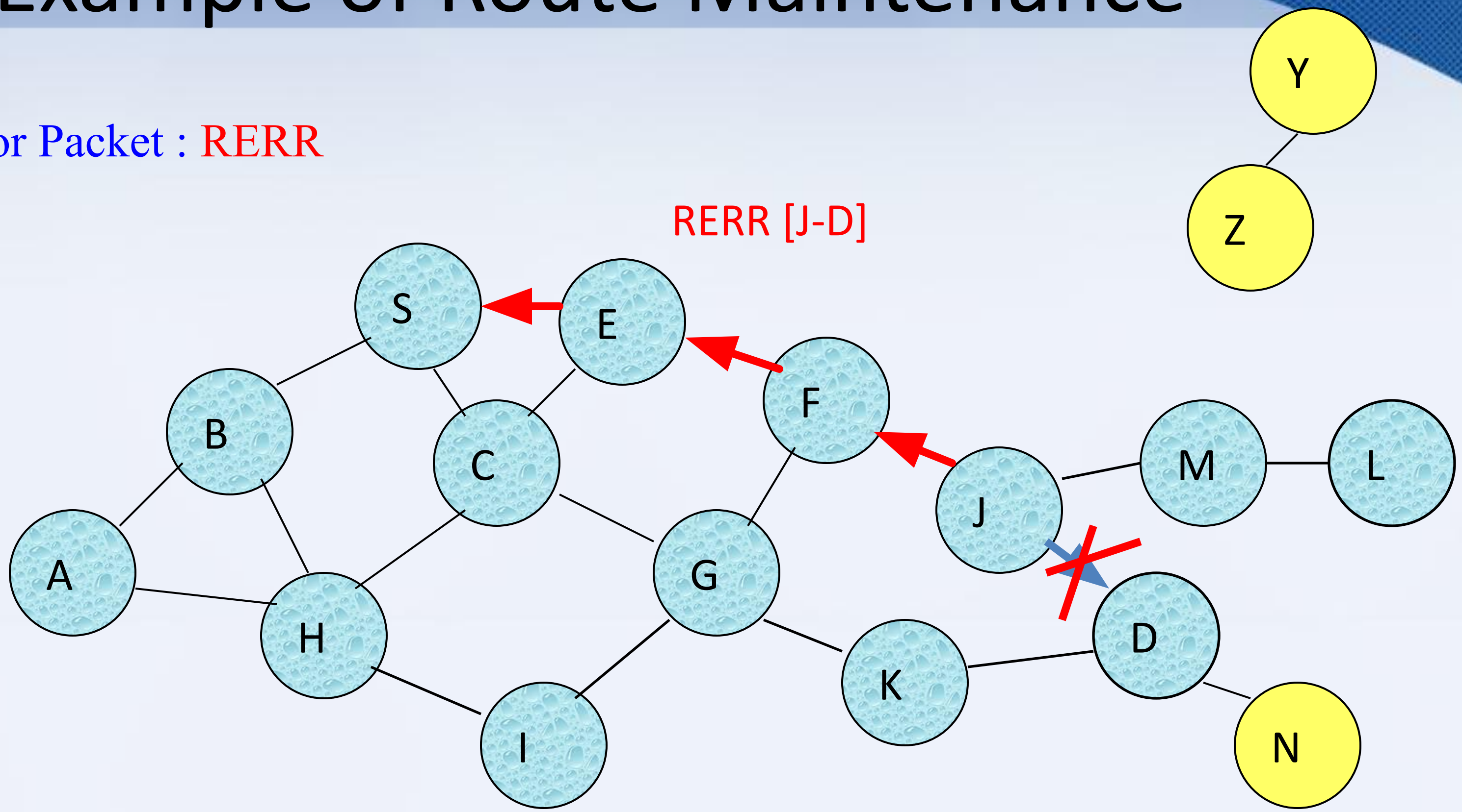
# Route Reply in DSR




Represents RREP control message

# An Example of Route Maintenance

Route Error Packet : RERR

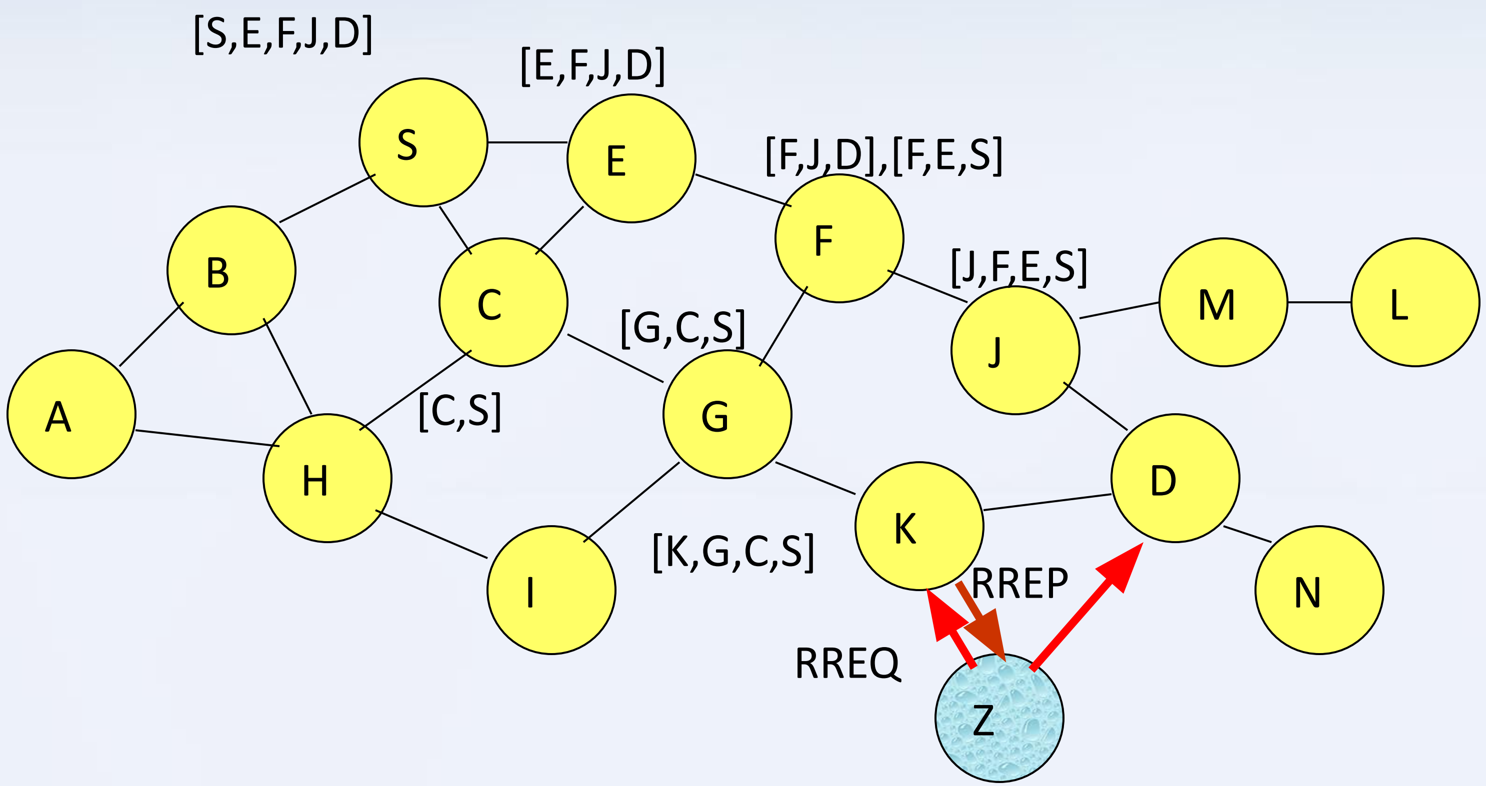



J sends a route error to S along route J-F-E-S when it finds link [J-D] broken

Nodes hearing RERR update their route cache to remove all invalid routes related with link J-D

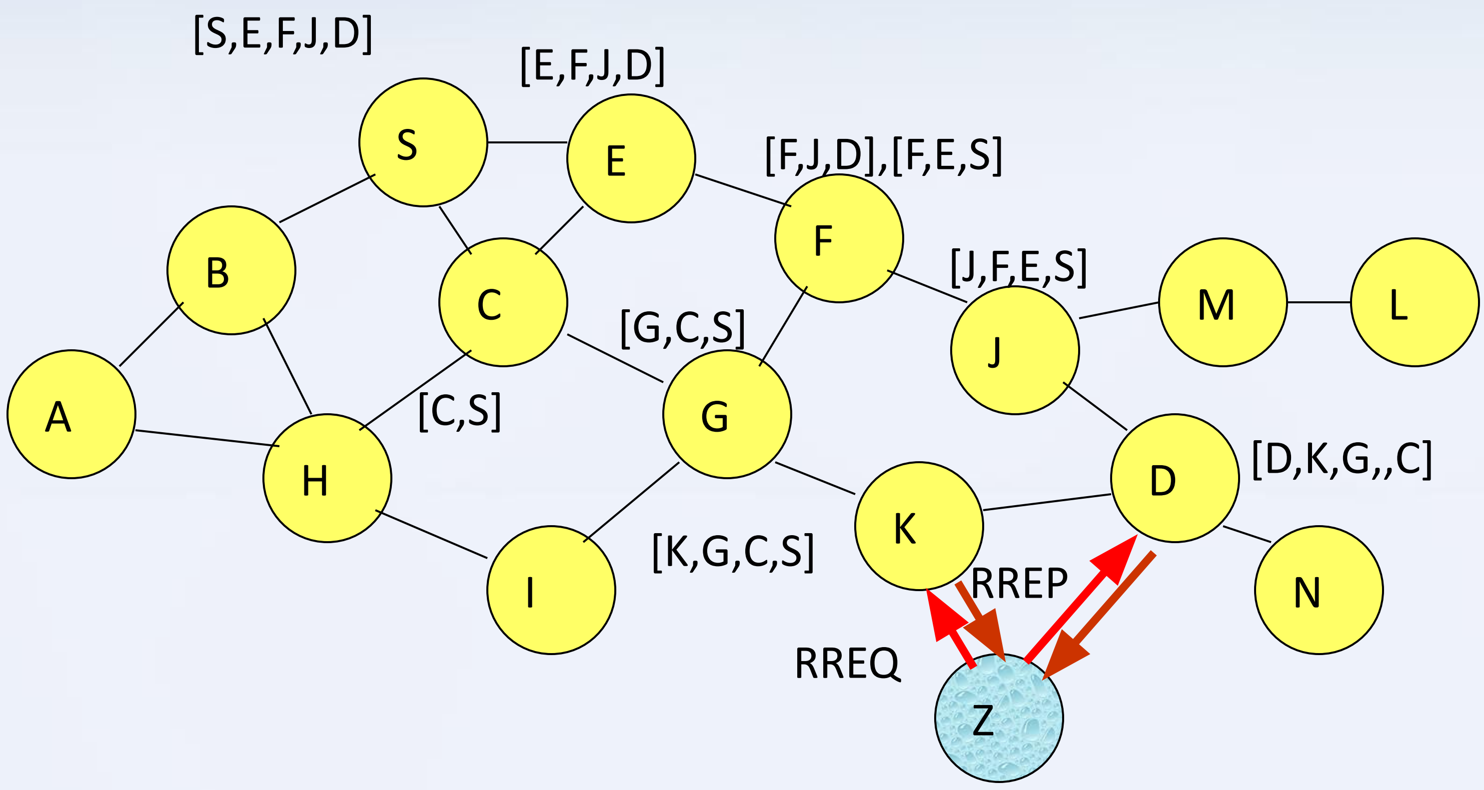
# Use of Route Caching

## Can Speed up Route Discovery




When node Z sends a route request for node C, node K sends back a route reply [Z,K,G,C] to node Z using a locally cached route

# Use of Route Caching
# Can Reduce Propagation of Route Requests




Route Replies (RREP) from node K and D limit flooding of RREQ.

# DSR

- The diameter of an ad-hoc network will not be too larger
    - Packet header will be bigger than payload if route is very longer

# Ad Hoc On-Demand Distance Vector Routing (AODV) Protocol

- The Ad hoc On-Demand Distance Vector protocol is both an on-demand and a table-driven protocol.

- The packet size in AODV is uniform unlike DSR. Unlike DSDV, there is no need for system-wide broadcasts due to local changes.

# AODV

- Each route has a lifetime after which the route expires if it is not used.
- A route is maintained only when it is used and hence old and expired routes are never used.

# AODV

- DSR includes source routes in packet headers
  - Resulting large headers can sometimes degrade performance.
- AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes.
- AODV retains the desirable feature of DSR that routes are maintained only between nodes which need to communicate.

# AODV

- Like DSR, this protocols uses two types of messages, route request (RREQ) and route reply (RREP).
- Like DSDV, we use sequence numbers to keep track of recent routes. Every time a node sends a new message, it uses a new sequence number which increases monotonically.

# Route Request (RREQ) Message

- When node S wants to send a message to node D, S searches its route table for a route to D.
- If there is no route, S initiates a RREQ message with the following components :
  - The IP addresses of S and D
  - The current sequence number of S and the last known sequence number of D
  - A broadcast ID from S. This broadcast ID is incremented each time S sends a RREQ message.

# Processing a RREQ Message

- The <broadcast ID, IP address> pair of the source S forms a unique identifier for the RREQ.

- Suppose a node P receives the RREQ from S. P first checks whether it has received this RREQ before.

- Each node stores the <broadcast ID, IPaddress> pairs for all the recent RREQs it has received.

# Processing a RREQ Message

- If P has seen this RREQ from S already, P discards the RREQ. Otherwise, P processes the RREQ :
  - P sets up a reverse route entry in its route table for the source S.
  - This entry contains the IP address and current sequence number of S,  number of hops to S and the address of the neighbour from whom P got the RREQ.

# Lifetime of a Route-Table Entry

- A lifetime is associated with the entry in the route table.

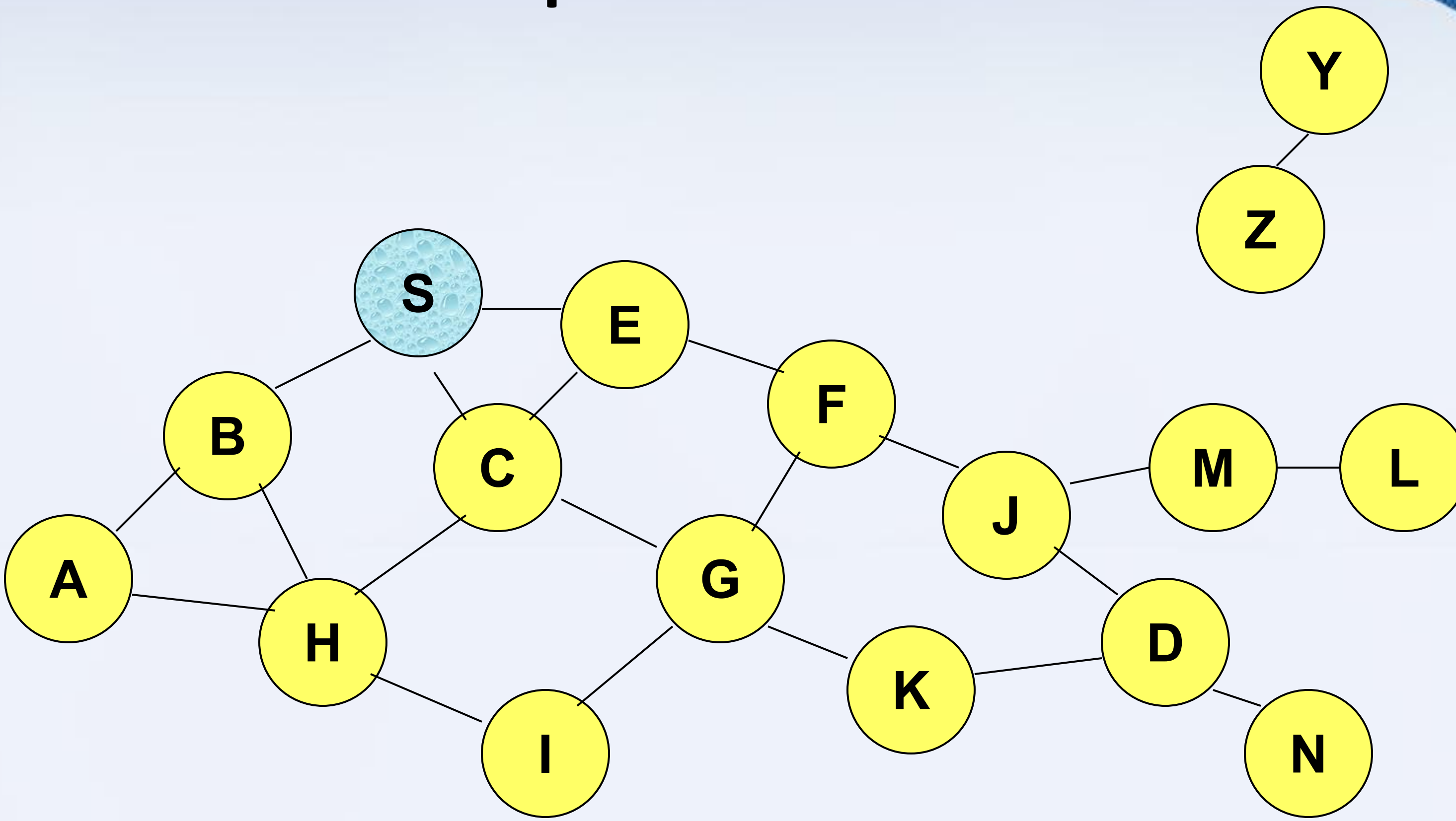
- This is an important feature of AODV. If a route entry is not used within the specified lifetime, it is deleted.

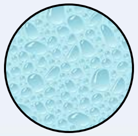
- A route is maintained only when it is used. A route that is unused for a long time is assumed to be stale.

# Handling More than one RREP

- An intermediate node P may receive more than one RREP for the same RREQ.
- P forwards the first RREP it receives and forwards a second RREP later only if :
  - The later RREP contains a greater sequence number for the destination, or
  - The hop-count to the destination is smaller in the later RREP
  - Otherwise, it does not forward the later RREPs. This reduces the number of RREPs propagating towards the source.

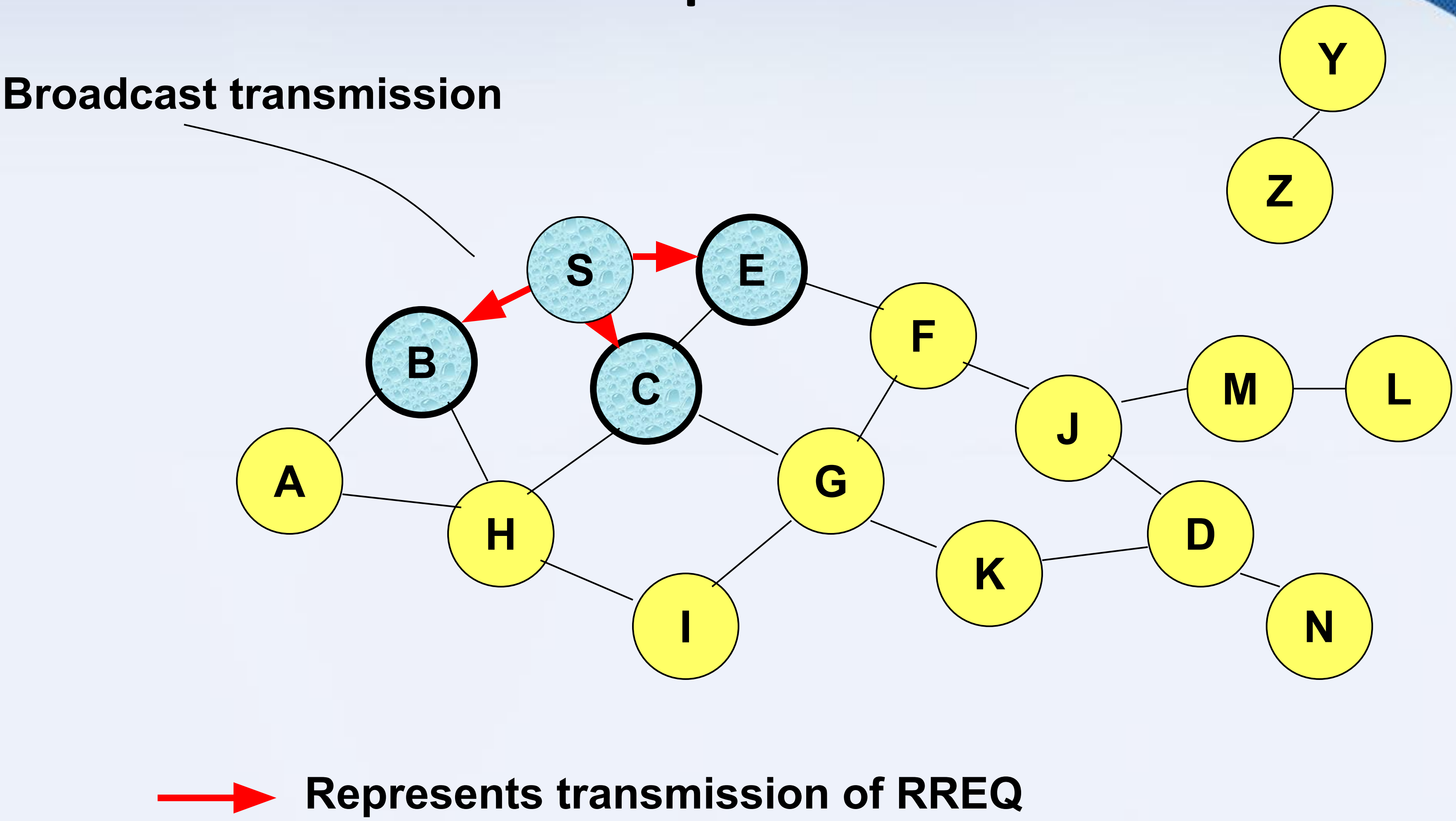
# Route Requests in AODV



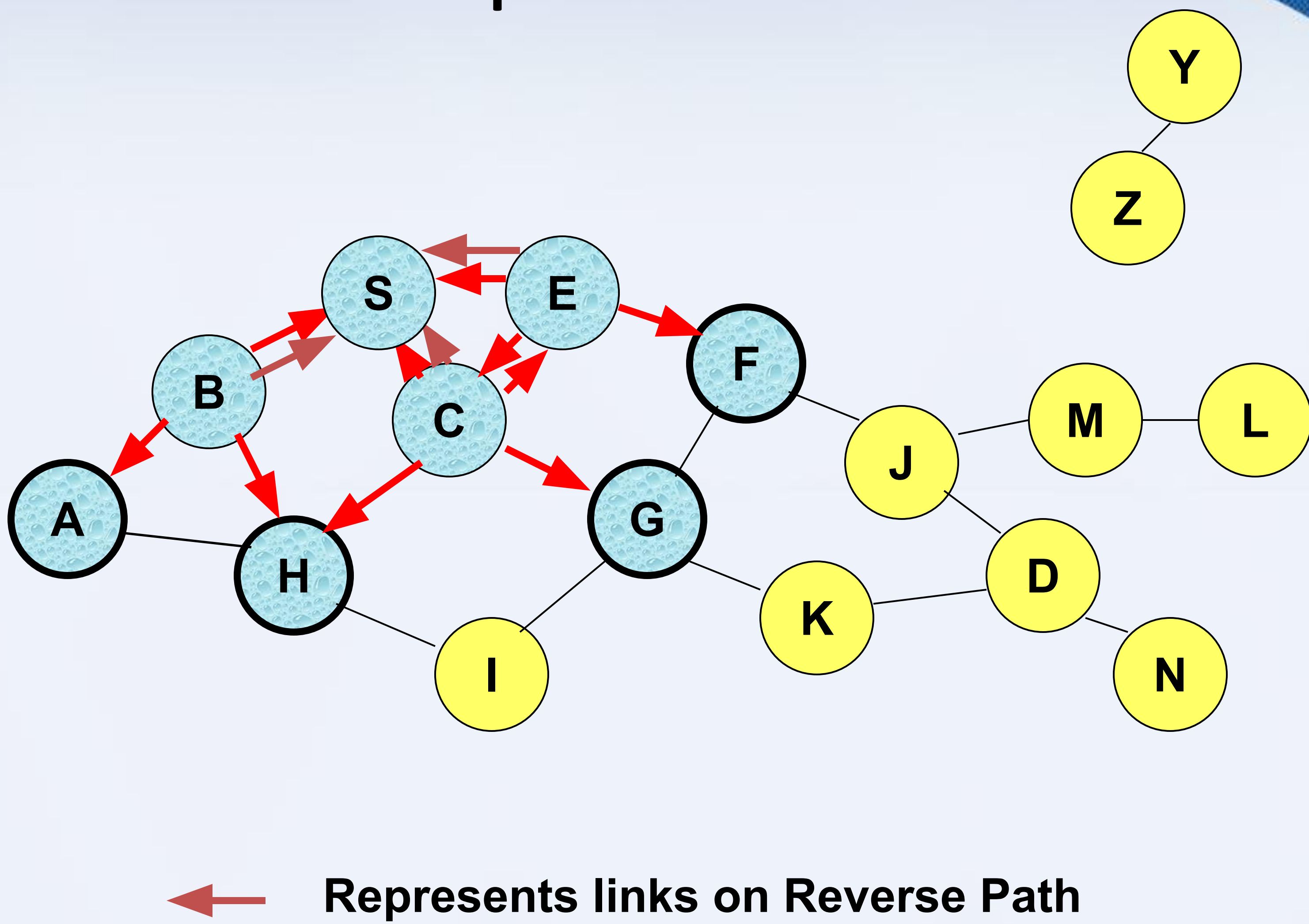



**Represents a node that has received RREQ for D from S**

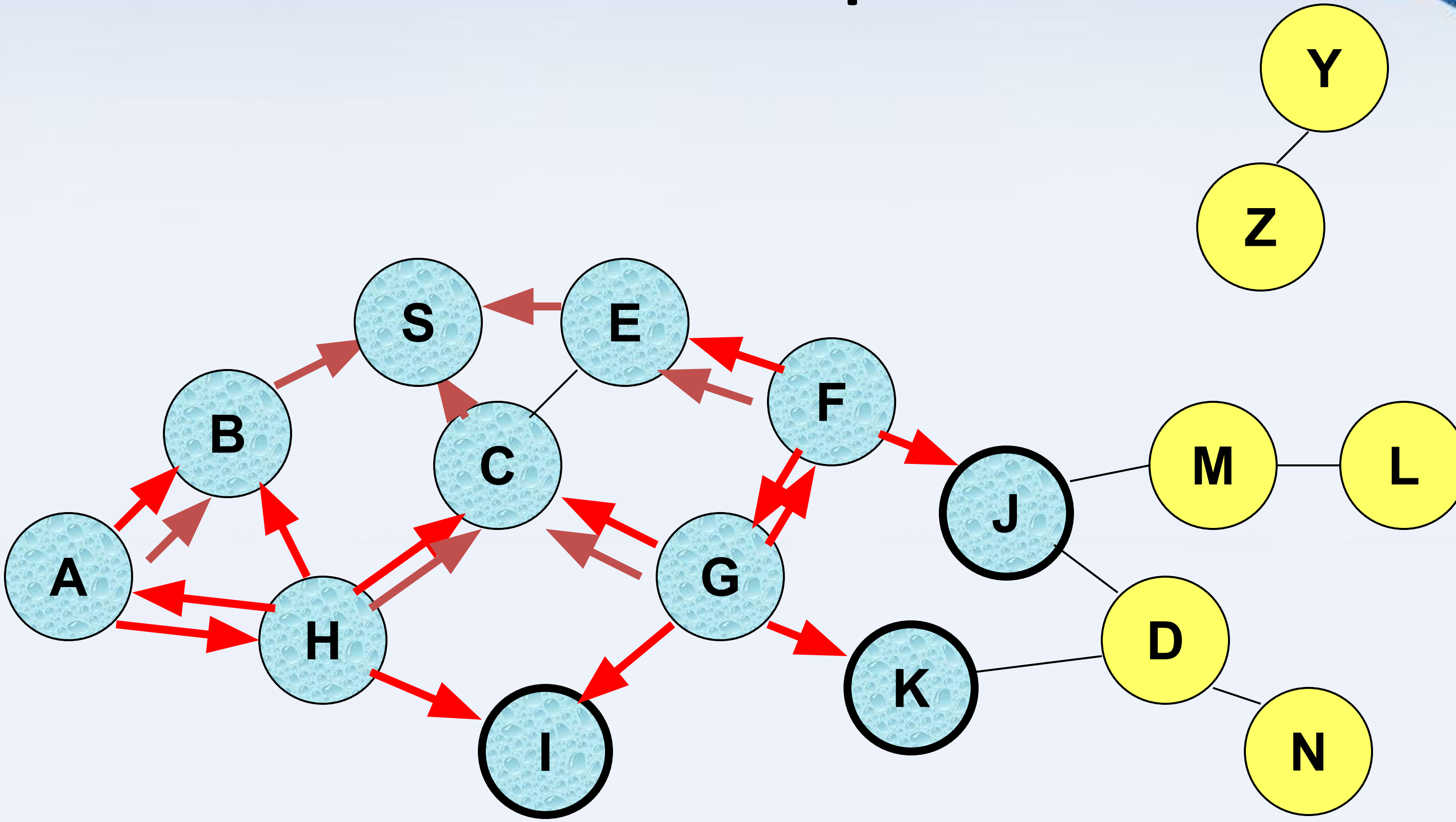
# Route Requests in AODV



# Route Requests in AODV

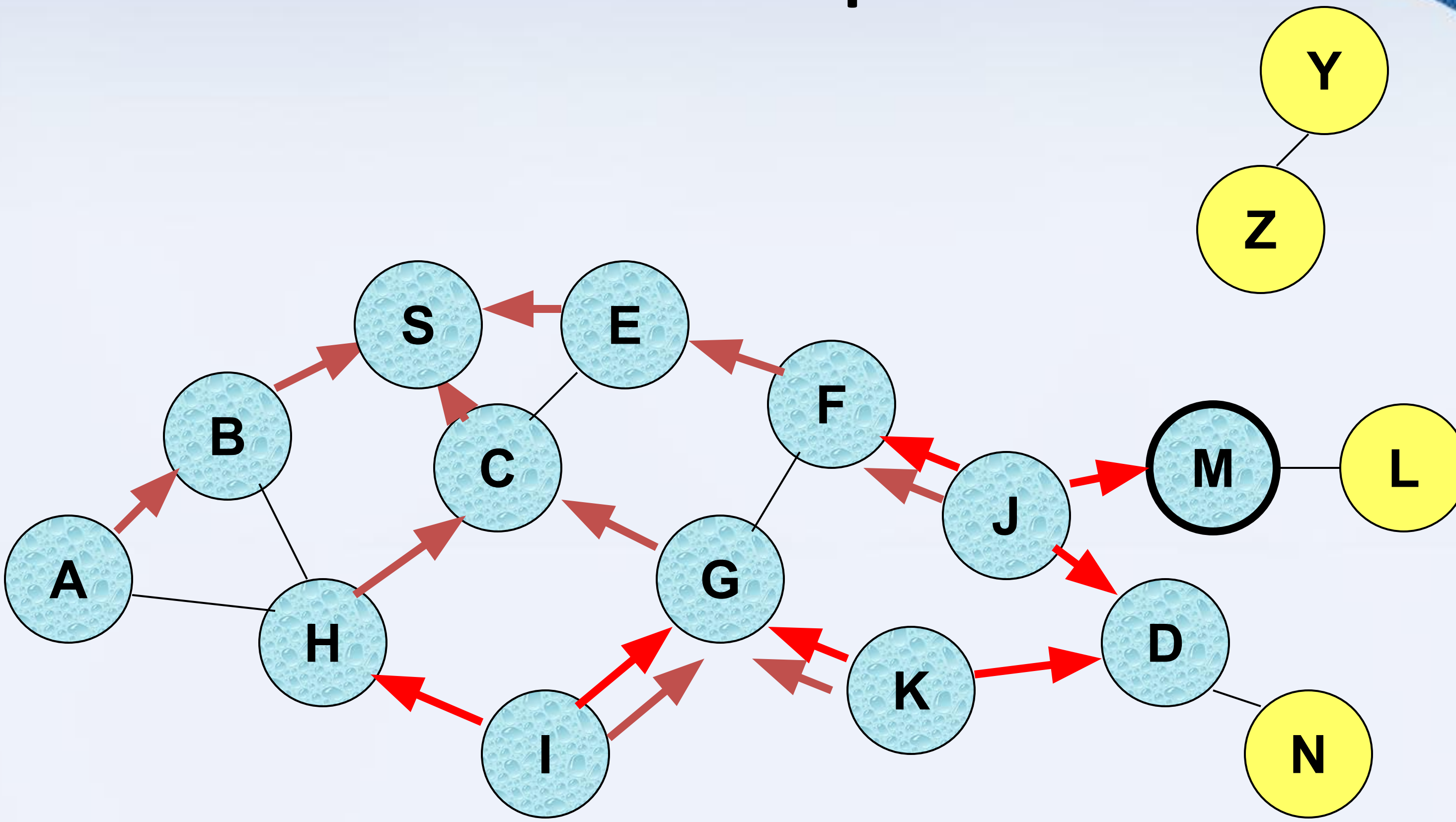


# Reverse Path Setup in AODV

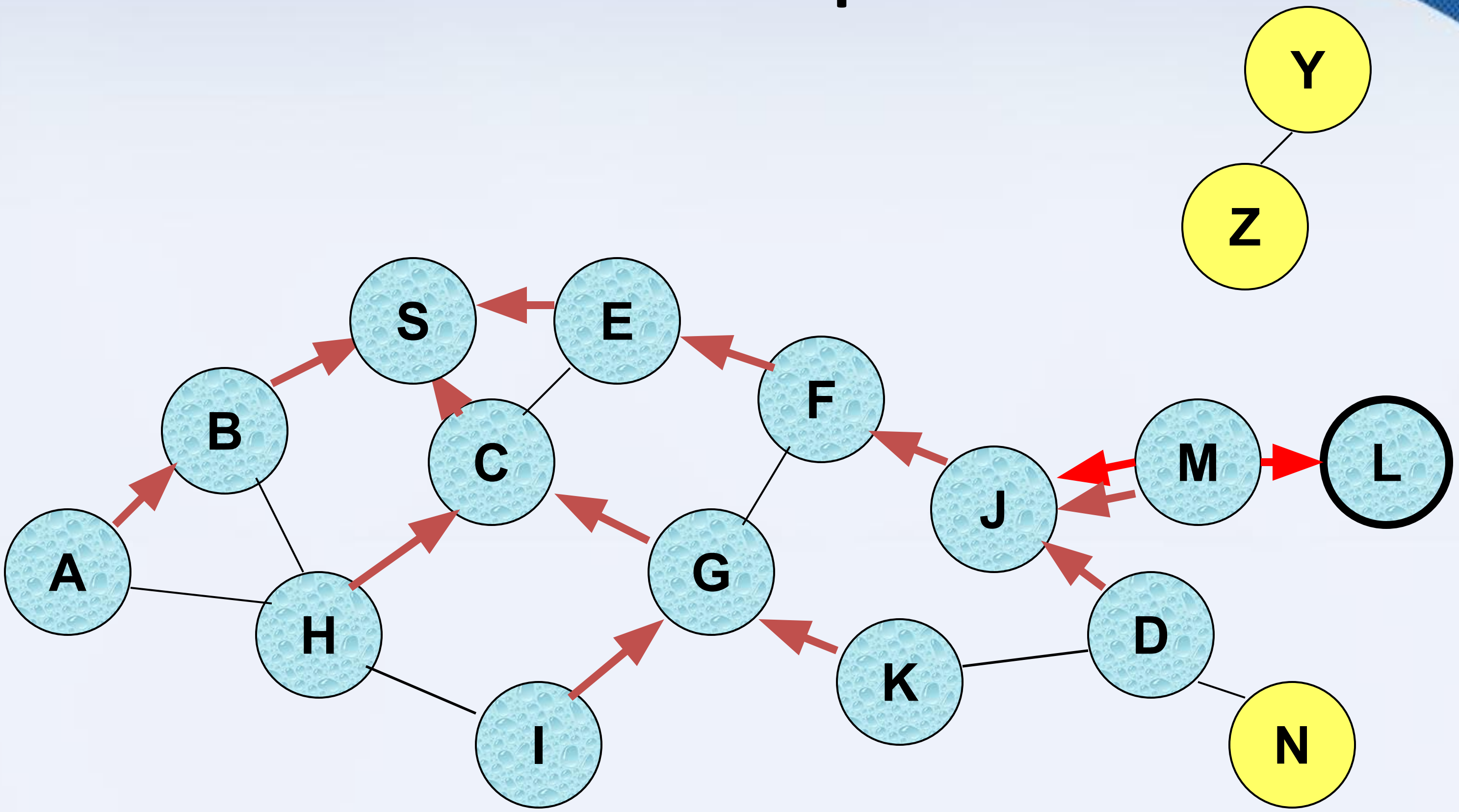



- Node C receives RREQ from G and H, but does not forward it again, because node C has **already forwarded RREQ** once

# Reverse Path Setup in AODV

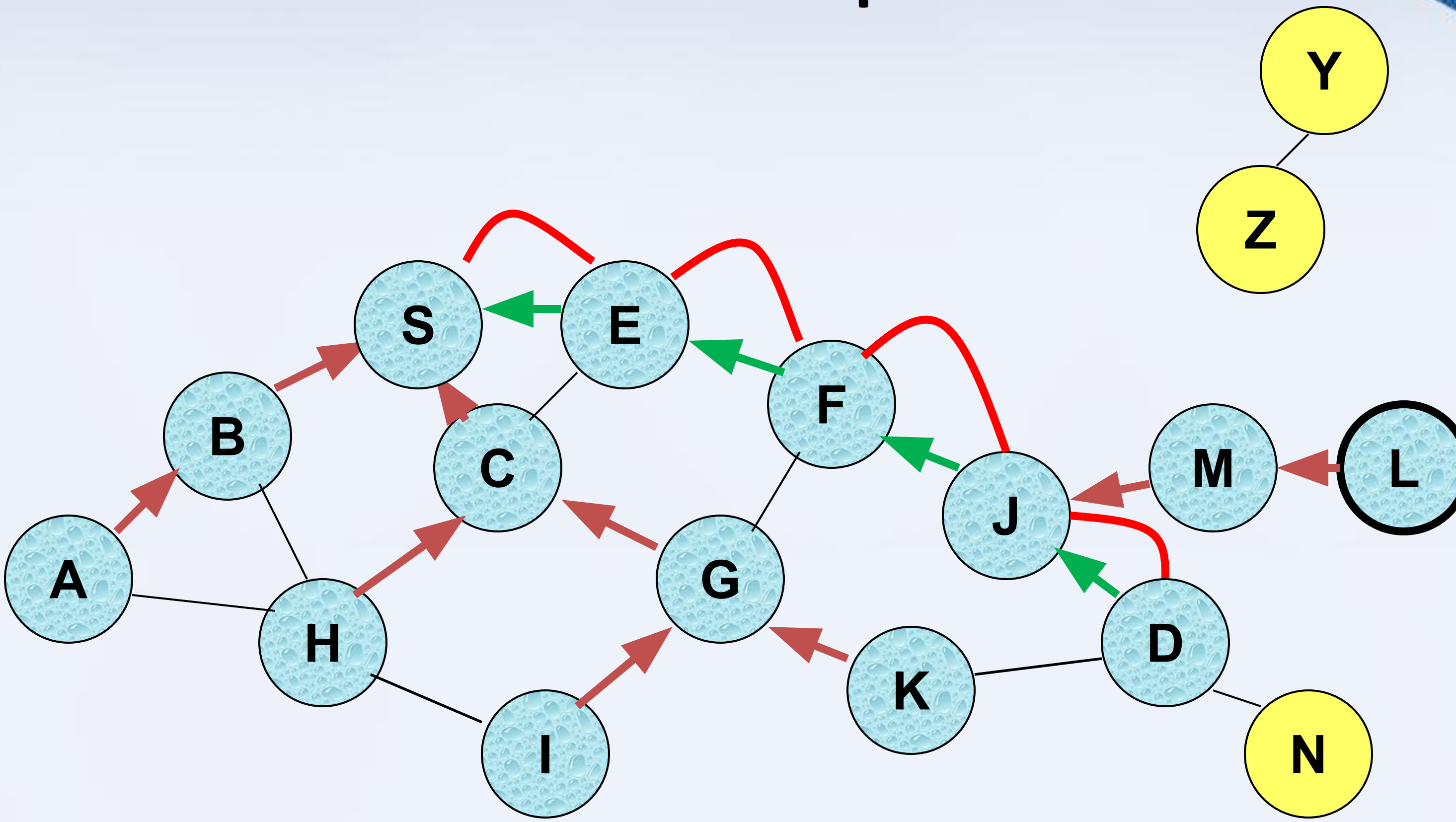


# Reverse Path Setup in AODV




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

# Forward Path Setup in AODV



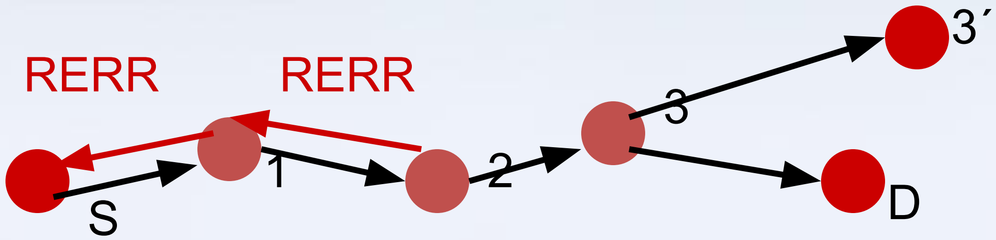

Forward links are setup when RREP travels along the reverse path

Represents a link on the forward path

# Route Maintenance

- Once a unicast route has been established between two nodes S and D, it is maintained as long as S (source node) needs the route.
- If S moves during an active session, it can reinitiate route discovery to establish a new route to D.
- When D or an intermediate node moves, a route error (RERR) message is sent to S.

# Route Maintenance




- The link from node 3 to D is broken as 3 has moved away to a position 3´.

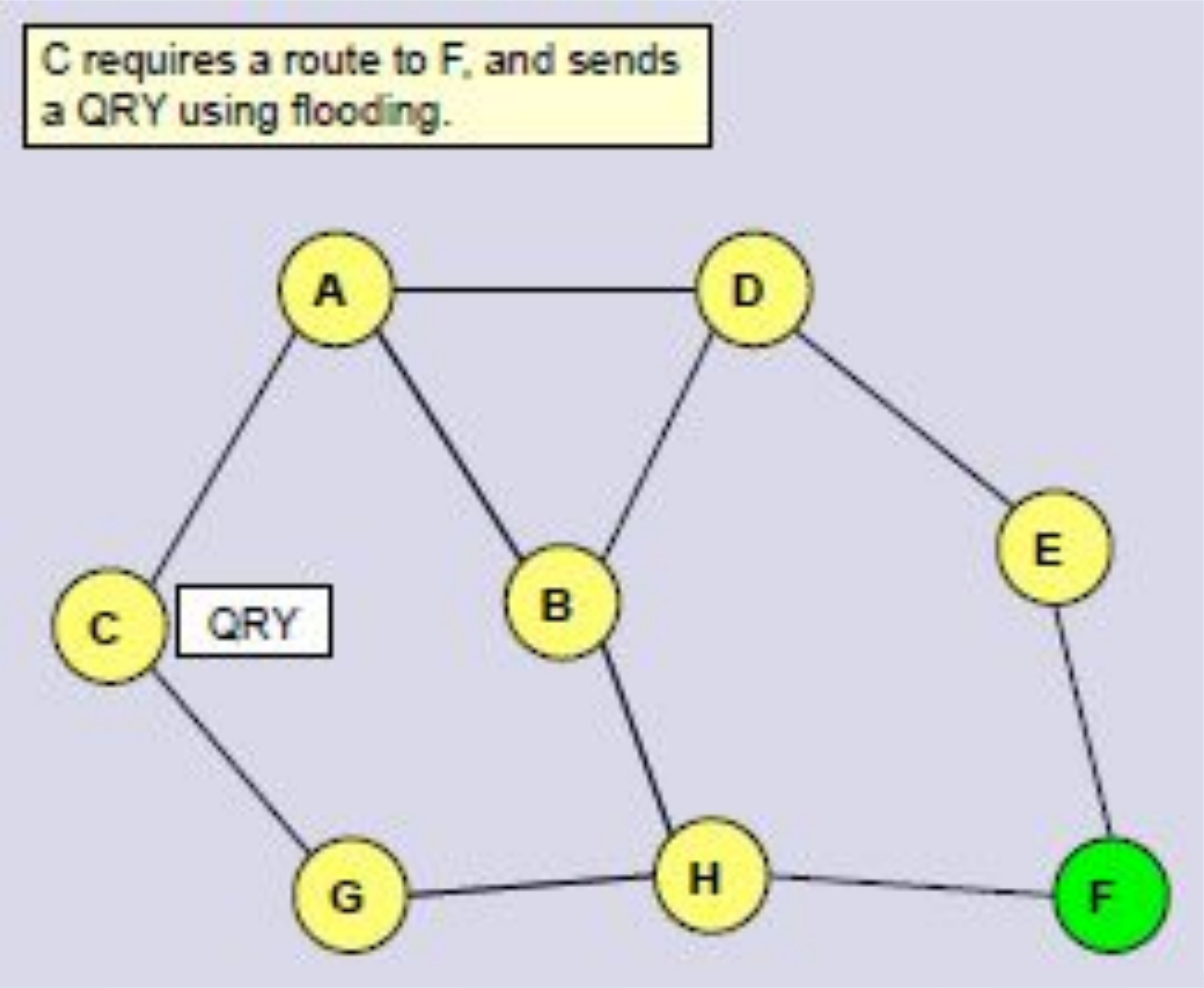
- Node 2 sends a RERR message to 1 and 1 sends the message in turn to S.

- S initiates a route discovery if it still needs the route to D.

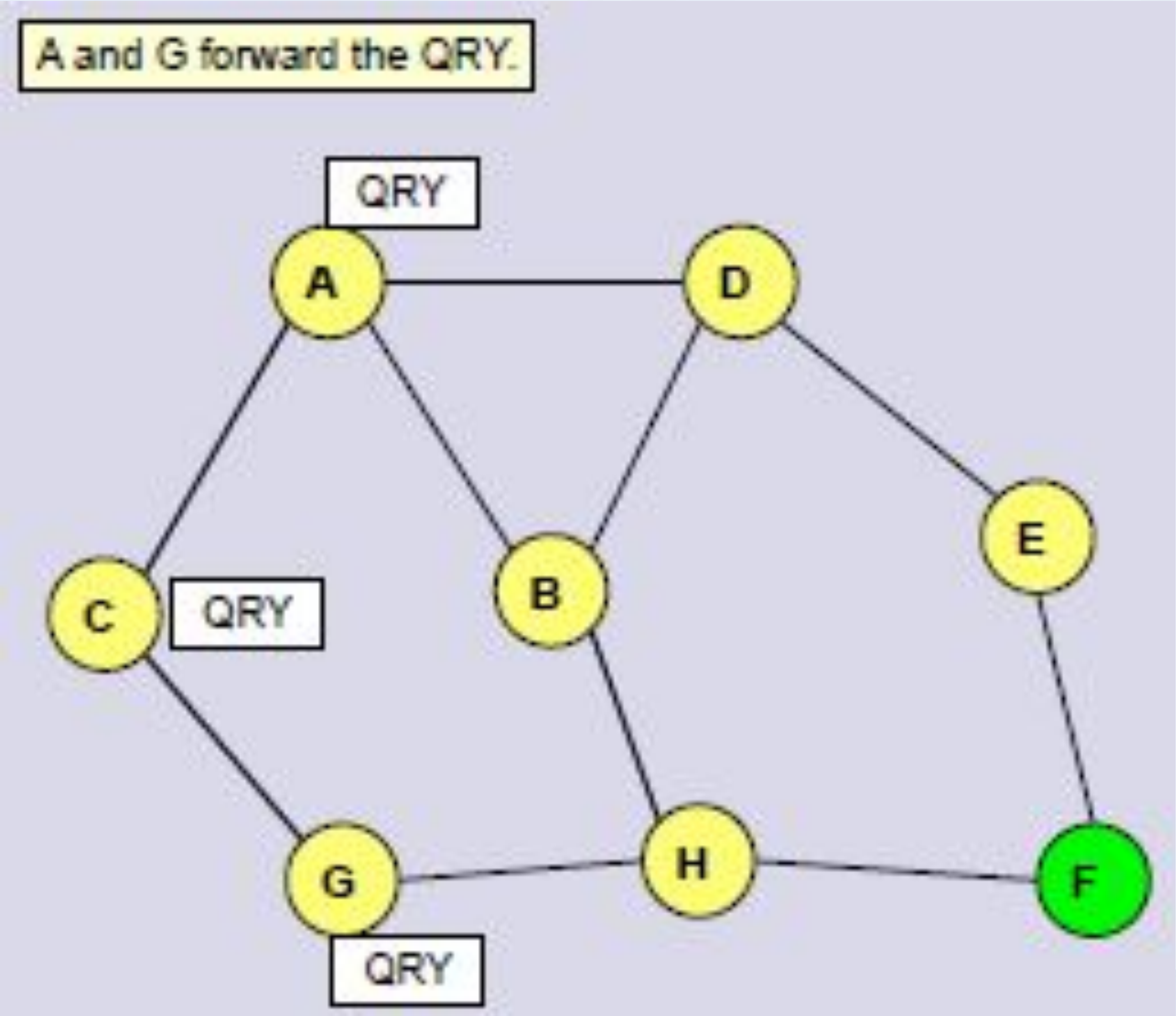
# Temporally Ordered Routing Algorithm(TORA)

- Based on a destination oriented Directed Acyclic Graph (DAG)
- The protocol has three basic functions :
  - Route creation (QRY)
  - Route maintenance (UPD)
  - Route erasure (CLR)

# TORA

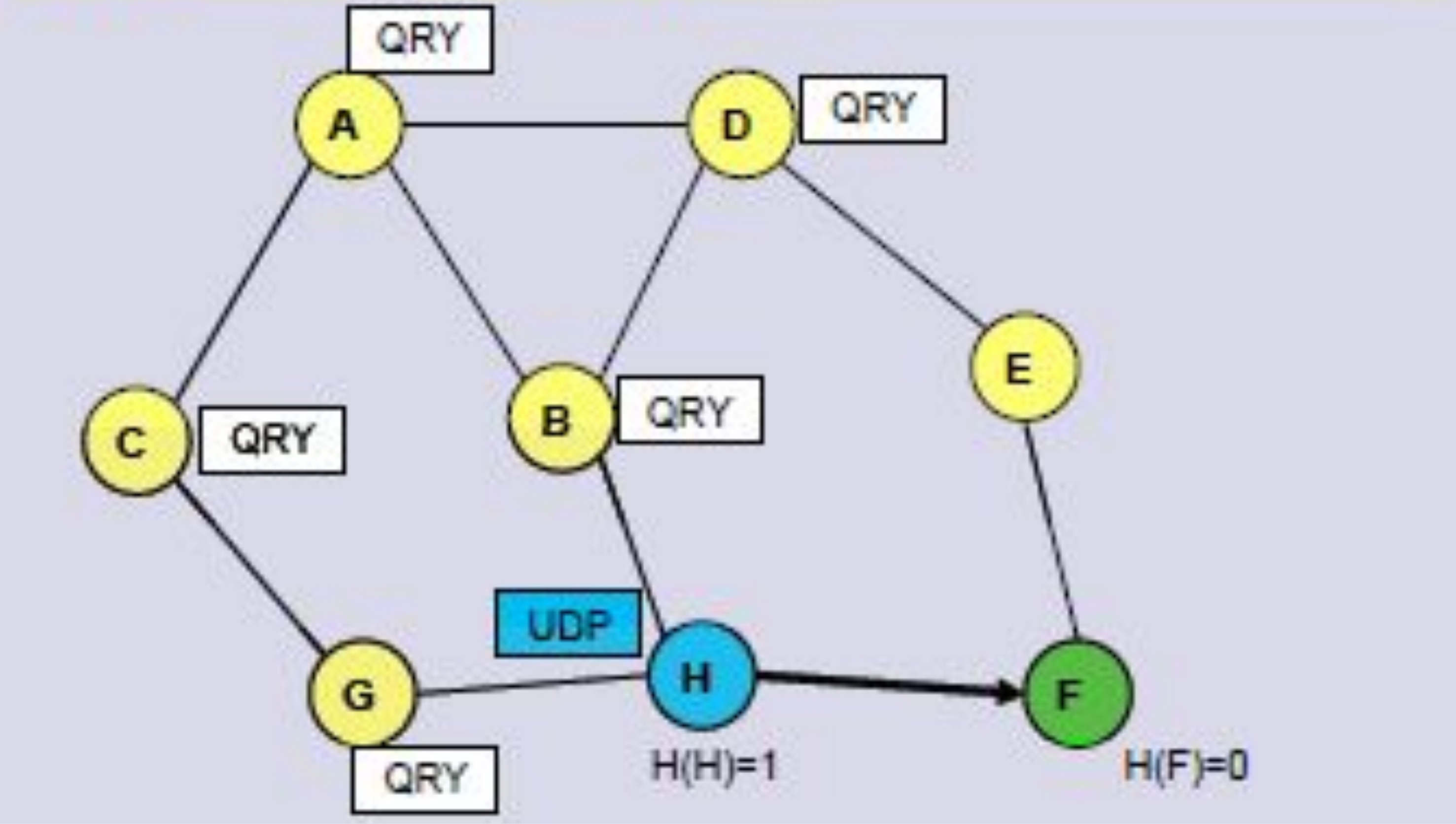

C requires a route to F, and sends a QRY using flooding.
A
D
E
C
QRY
B
G
H
F

# TORA

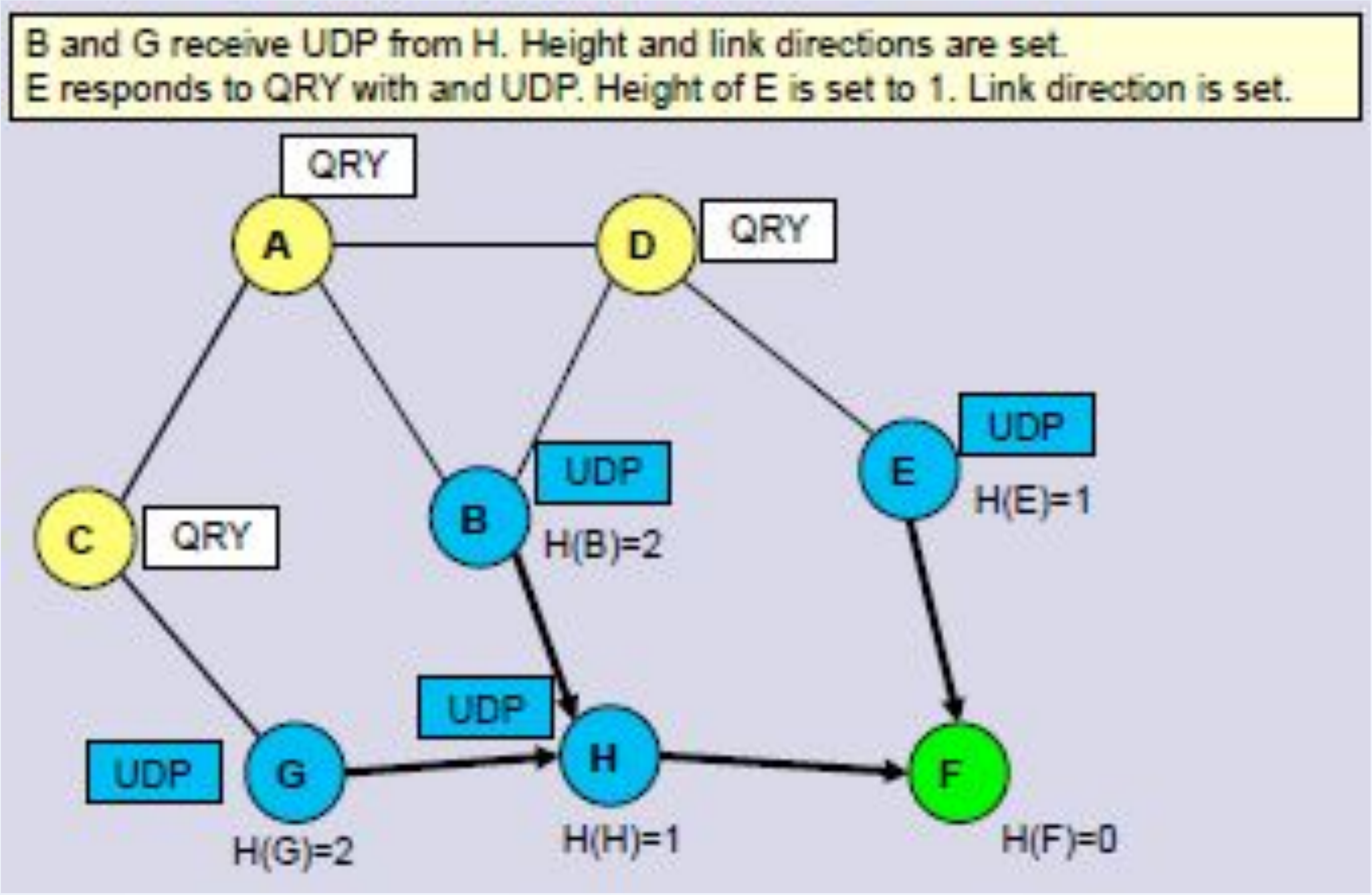


# TORA

B and D forward the QRY.
H responds to QRY with and UDP. Height of H is set to 1.
Height of F is set to 0. Link direction is set.

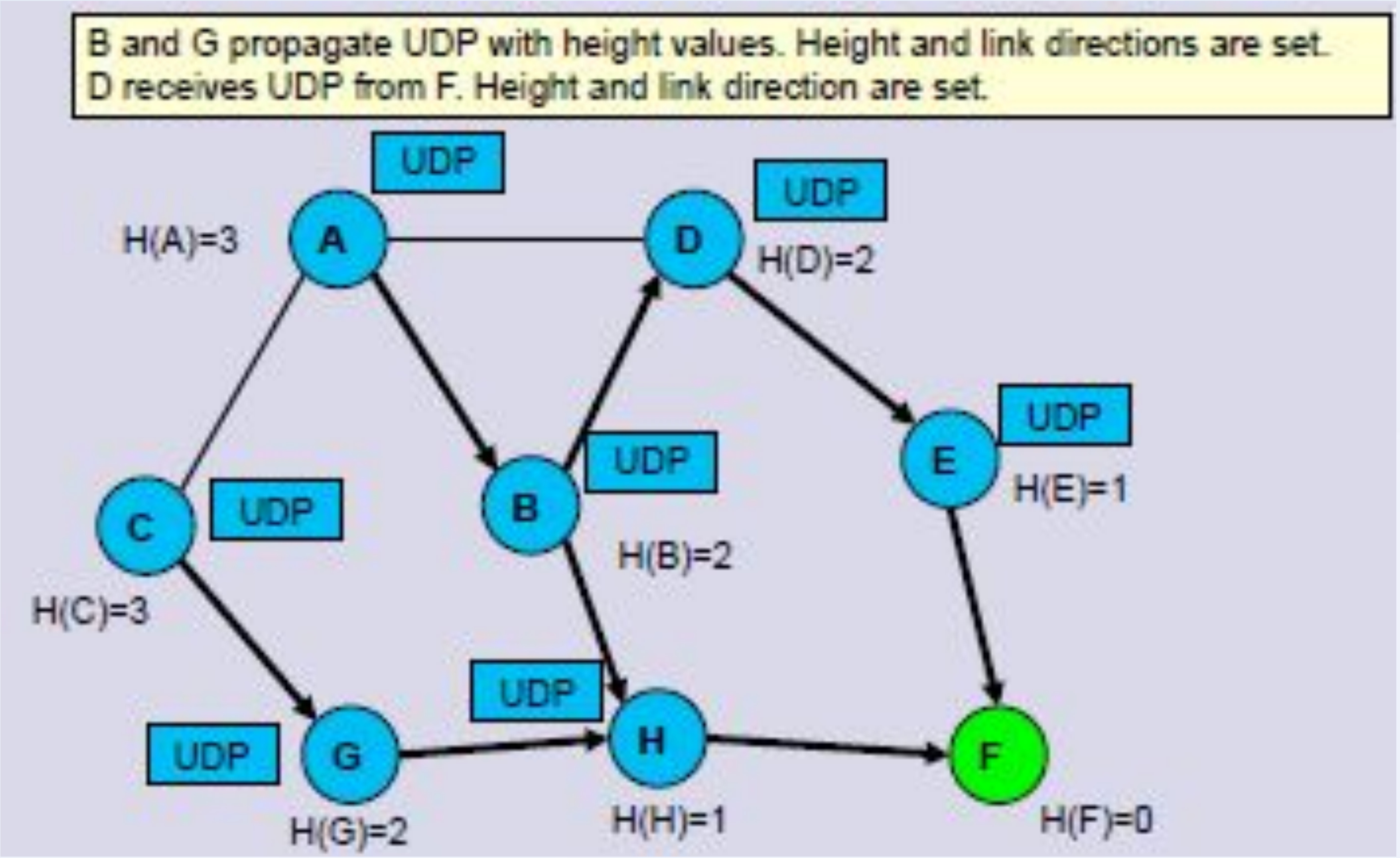


# TORA

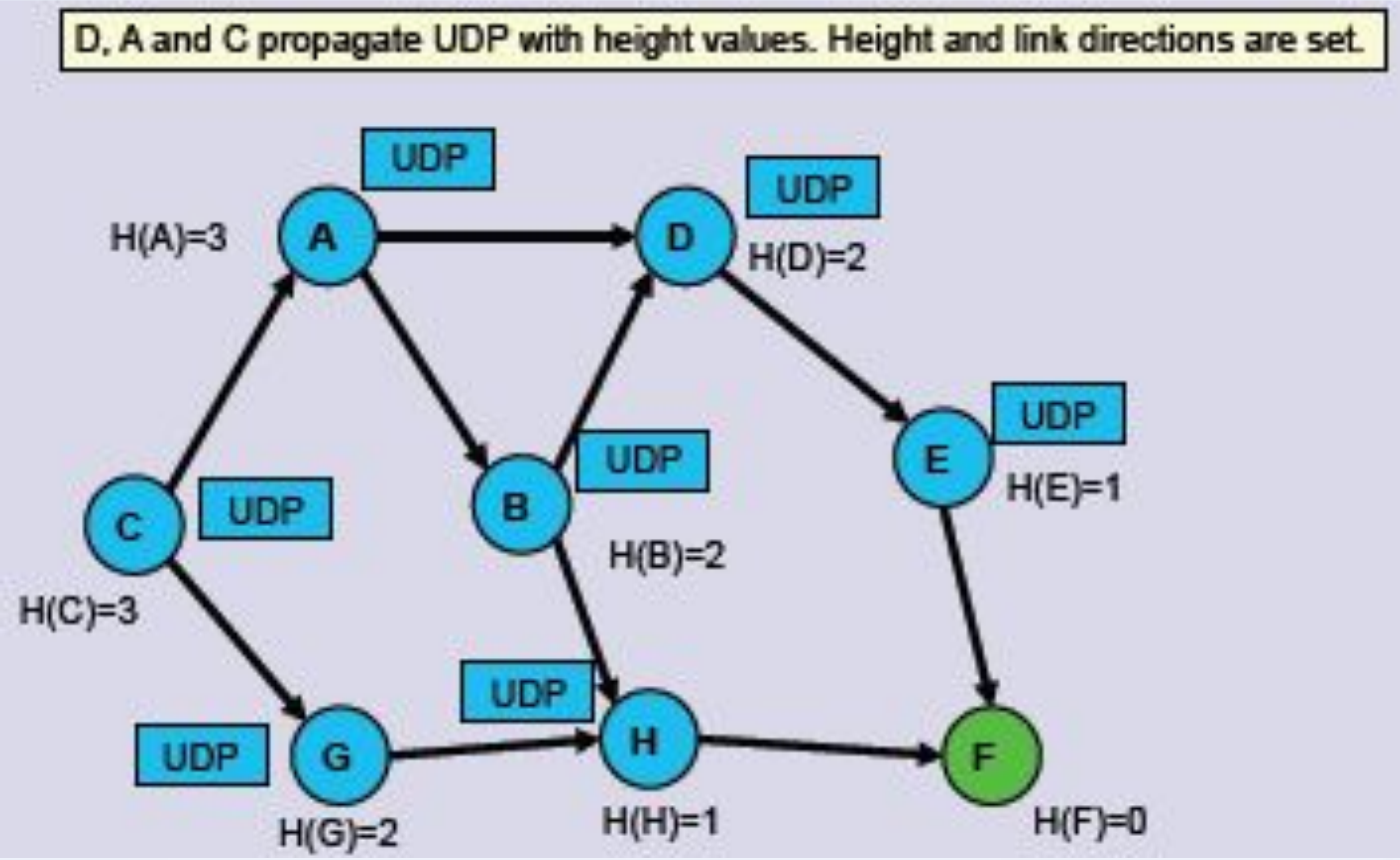


B and G receive UDP from H. Height and link directions are set.
E responds to QRY with and UDP. Height of E is set to 1. Link direction is set.
QRY
A
D
QRY
UDP
E
H(E)=1
UDP
B
H(B)=2
C
QRY
UDP
UDP
G
H
F
H(G)=2
H(H)=1
H(F)=0

# TORA


B and G propagate UDP with height values. Height and link directions are set.
D receives UDP from F. Height and link direction are set.
UDP
H(A)=3
A
D
UDP
H(D)=2
UDP
E
H(E)=1
UDP
B
H(B)=2
C
UDP
H(C)=3
UDP
H
UDP
G
F
H(G)=2
H(H)=1
H(F)=0

# TORA


D, A and C propagate UDP with height values. Height and link directions are set.
UDP
H(A)=3
A
D
UDP
H(D)=2
UDP
E
H(E)=1
UDP
B
H(B)=2
C
UDP
H(C)=3
UDP
H
UDP
G
F
H(G)=2
H(H)=1
H(F)=0

# TORA

Each node maintains height values of neighbours for a destination.
C has several options to reach F → either though A or G.
Height values may be used to favor routing over links with shorter distances.