UNIT-IV

Ad hoc On Demand Distance Vector (AODV) Routing Protocol

MADHESWARI.K

AP/CSE

Introduction: DSDV vs. AODV

DSDV	AODV
DSDV broadcasts every change in the network to every node	In AODV such broadcasts are not necessary
When two neighbors enter communication range of each other ❍ This results in a network wide broadcast	If a link breakage does not affect on going transmission -> no global broadcast occurs
Similarly when two nodes drift apart from each other's range –> link breakage ❍ Also results in a network wide broadcast ❐ Local movements have global effects	Only affected nodes are informed ❐ Local movements of nodes have local effects ❐ AODV reduces the network wide broadcasts to the extent possible ❐ Significant reduction in control overhead as compared to DSDV

Ad Hoc On Demand DistanceVector (AODV) Routing

- Reactive or on Demand

❒ Descendant of DSDV

❒ Uses bi-directional links

❒ Route discovery cycle used for route
finding

❒ Maintenance of active routes

❒ Sequence numbers used for loop prevention and as route freshness criteria

❒ Provides unicast and multicast communication

Ad Hoc On Demand DistanceVector (AODV) Routing

- Whenever routes are not used -> get expired -> Discarded
  ❍ Reduces stale routes
  ❍ Reduces need for route maintenance

  ❒ Minimizes number of active routes between an active source and destination

  ❒ Can determine multiple routes between a source and a destination, but implements only a single route, because

  ❍ Difficult to manage multiple routes between same source/destination pair
  ❍ If one route breaks, its difficult to know whether other route is available
  ❍ Lot of book-keeping involved

AODV Properties

AODV discovers routes as and when necessary

1. Does not maintain routes from every node to every other

2. Routes are maintained just as long as necessary

3. Every node maintains its monotonically increasing sequence number -> increases every time the node notices change in the neighborhood topology

AODV Properties (2)

- AODV utilizes routing tables to store routing information
  1. A Routing table for unicast routes
  2. A Routing table for multicast routes

- The route table stores: <destination addr, next-hop addr destination sequence number, life_time>

- For each destination, a node maintains a list of precursor nodes, to route through them
  - Precusor nodes help in route maintenance (more later)

- Life-time updated every time the route is used
  - If route not used within its life time -> it expires

AODV - Route Discovery (1)

- When a node wishes to send a packet to some destination -
  - It checks its routing table to determine if it has a current route to the destination
    - If Yes, forwards the packet to next hop node
    - If No, it initiates a route discovery process
- Route discovery process begins with the creation of a Route Request (RREQ) packet -> source node creates it
- The packet contains - source node's IP address, source node's current sequence number, destination IP address, destination sequence number

AODV - Route Discovery (2)

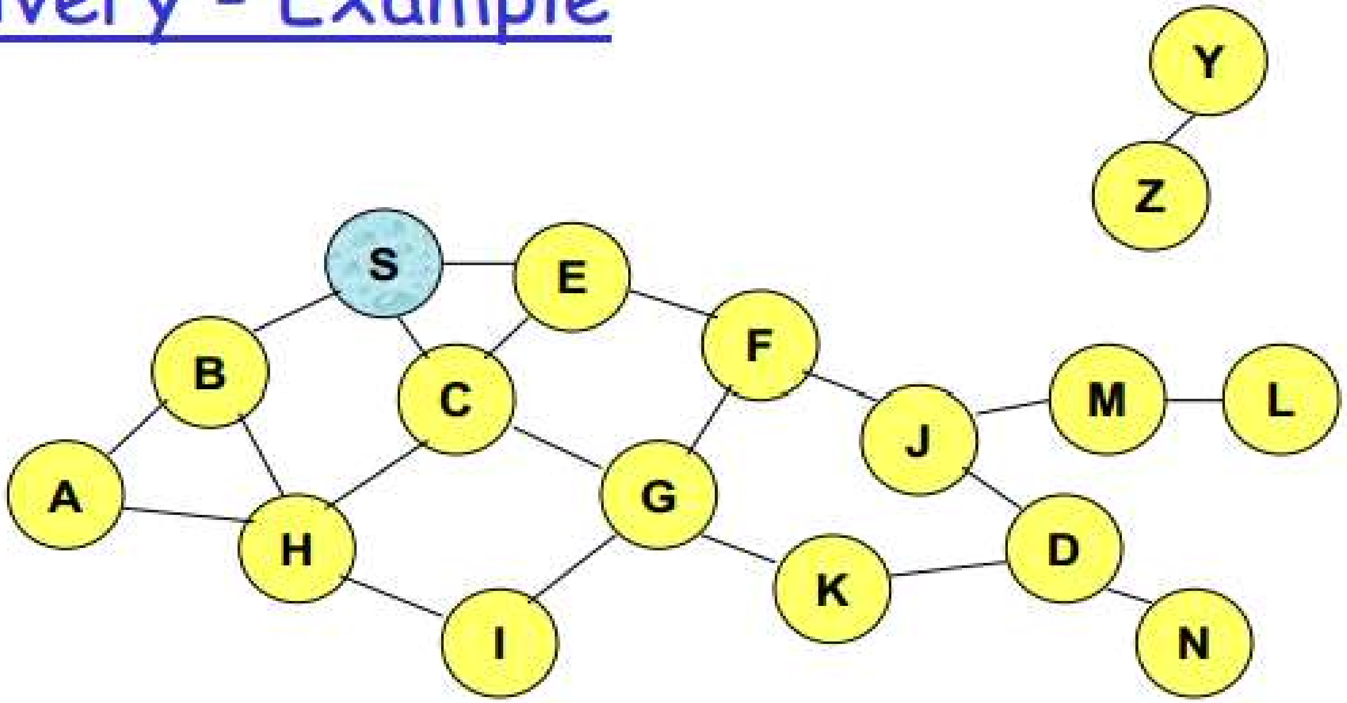
- Packet also contains broadcast ID number
  - Broadcast ID gets incremented each time a source node uses RREQ
  - Broadcast ID and source IP address form a unique identifier for the RREQ

- Broadcasting is done via Flooding

Flooding for Control Packet Delivery

- Sender S broadcasts a control packet P to all its neighbors
- Each node receiving P forwards P to its neighbors
- Sequence numbers help to avoid the possibility of forwarding the same packet more than once
- Packet P reaches destination D provided that D is reachable from sender S
- Node D does not forward the packet

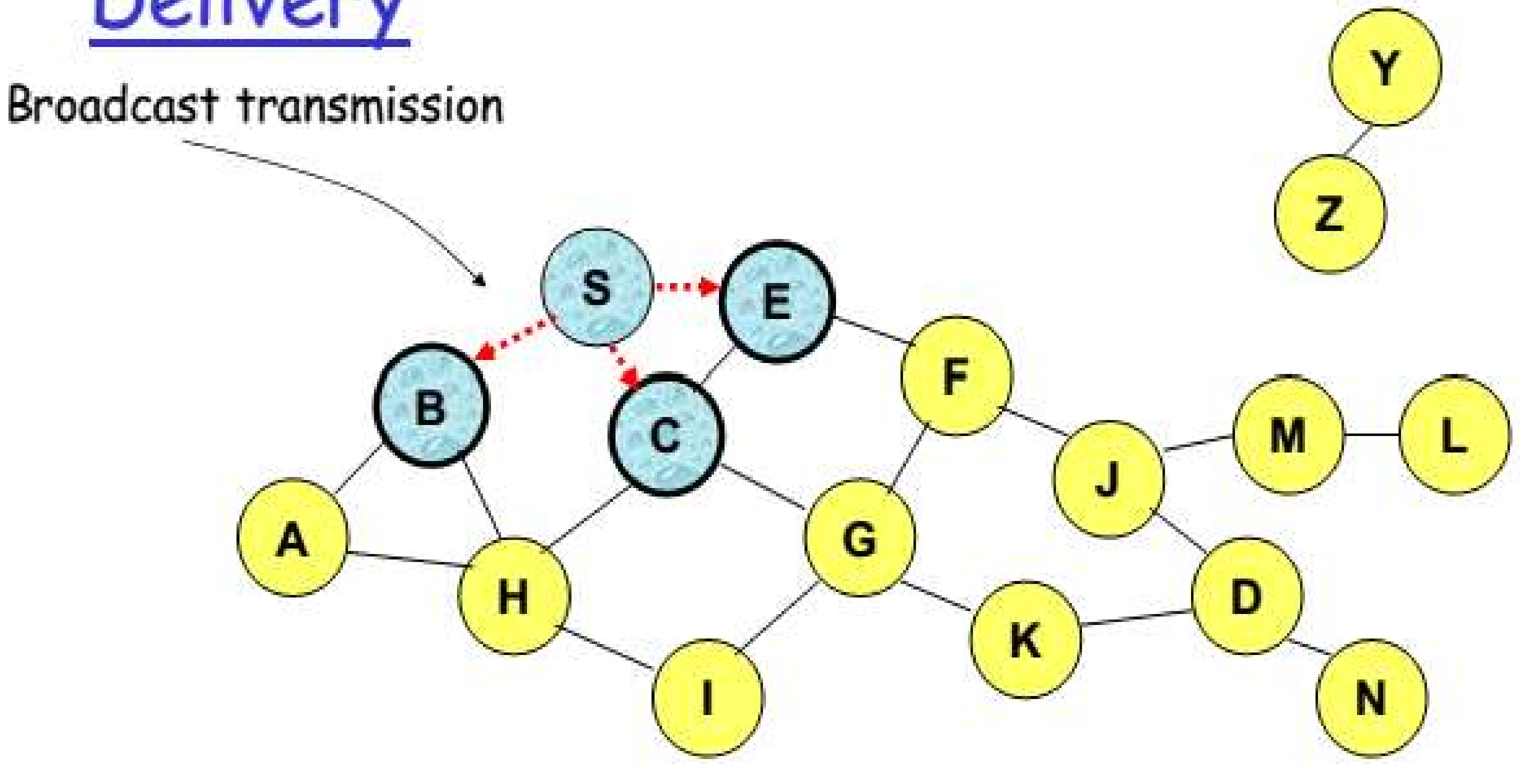
Flooding for Control Packet Delivery - Example



● Represents a node that has received packet P

— Represents that connected nodes are within each other's transmission range

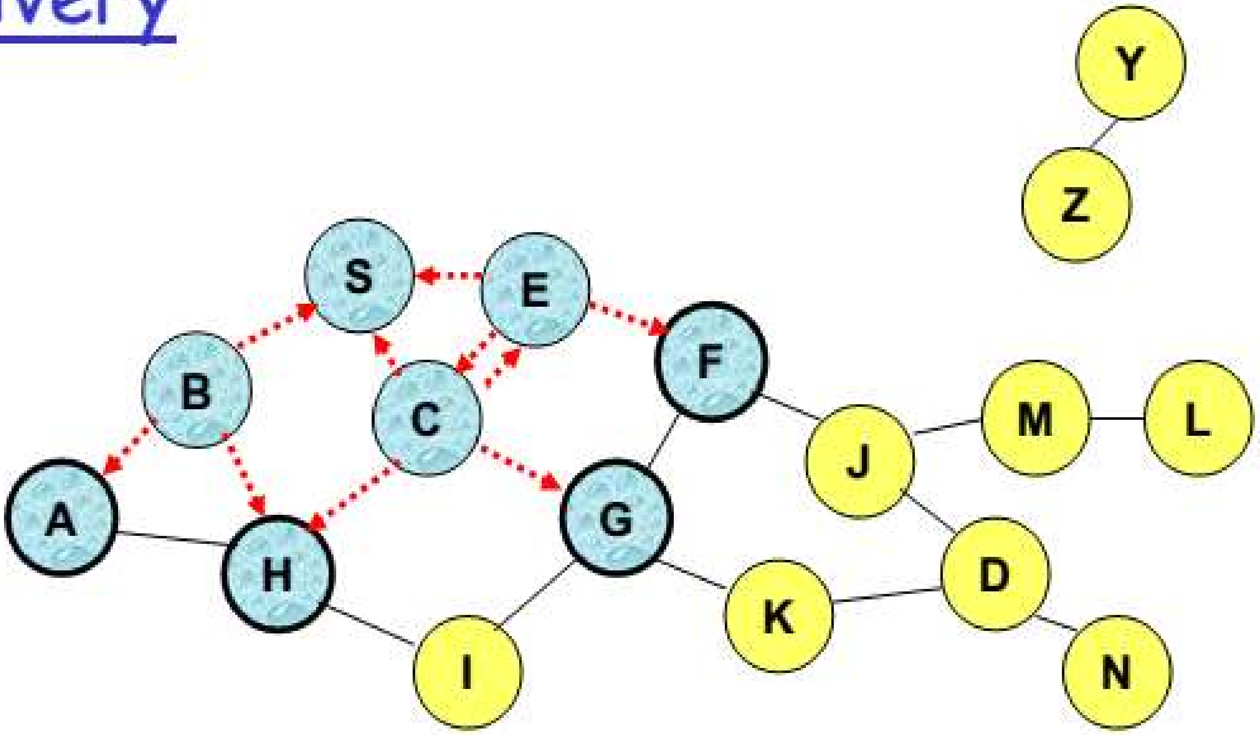
Flooding for Control Packet Delivery



Represents a node that receives packet P for the first time

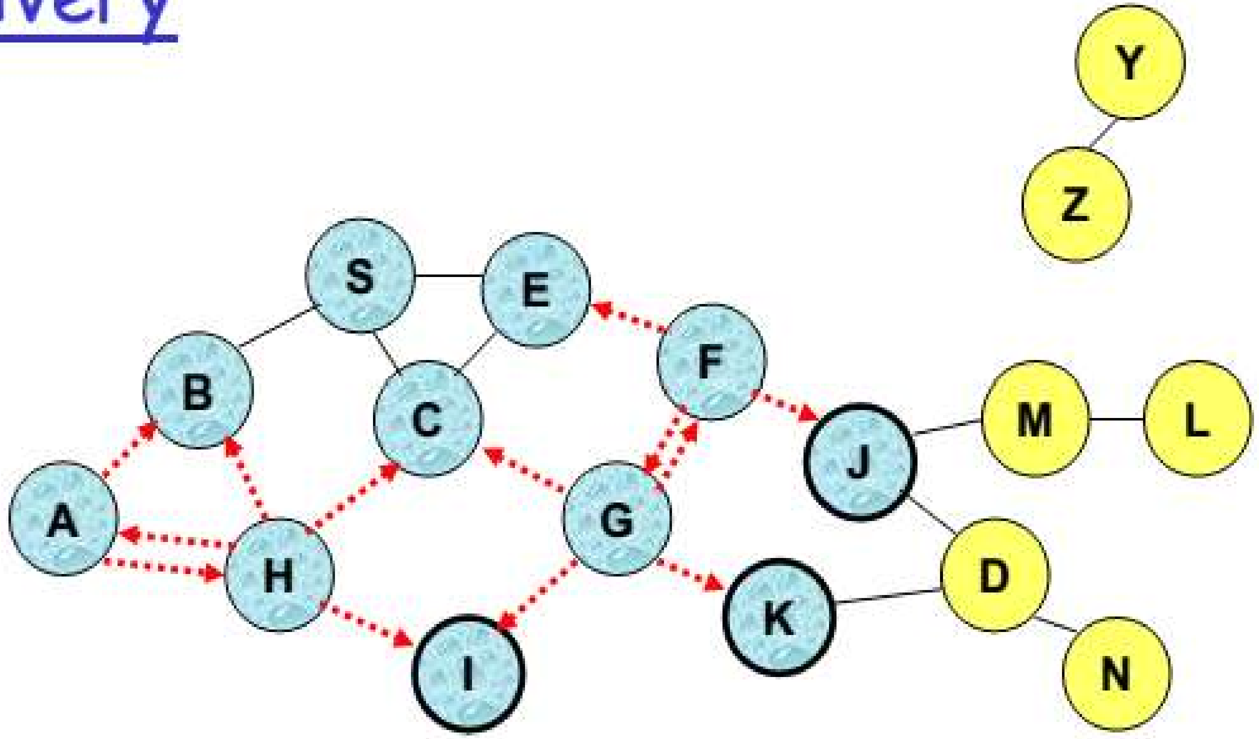
Represents transmission of packet P

Flooding for Control Packet Delivery



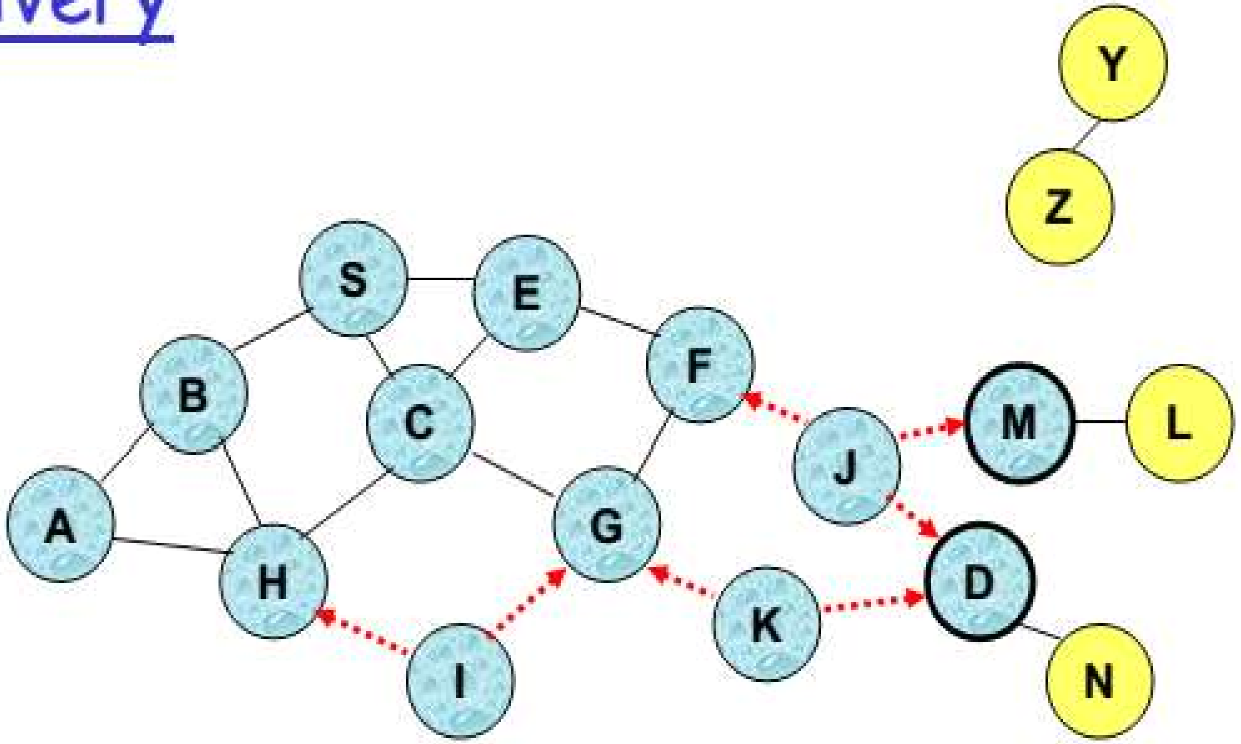
- Node H receives packet P from two neighbors: potential for collision

Flooding for Control Packet Delivery



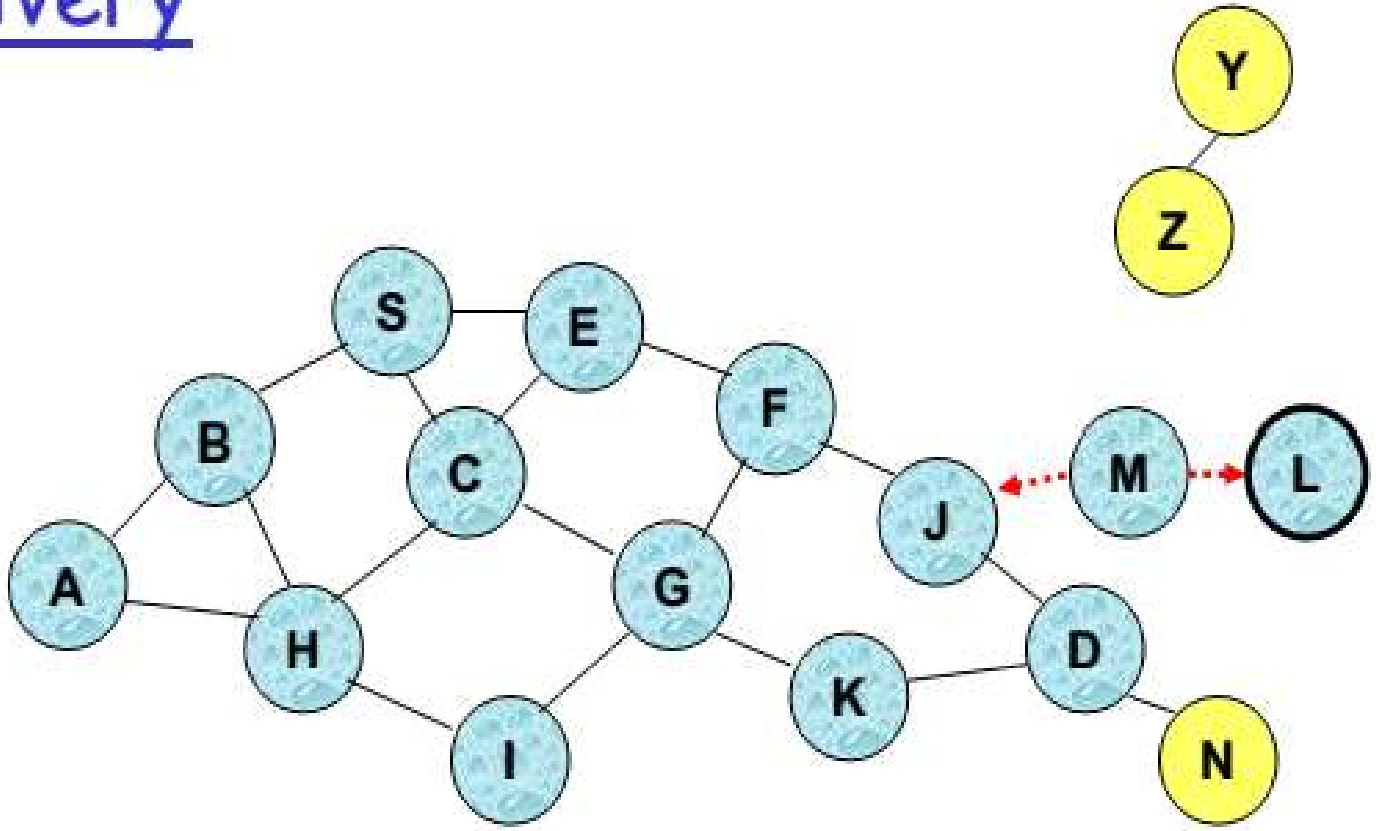
- Node C receives packet P from G and H, but does not forward it again, because node C has already forwarded packet P once

Flooding for Control packet Delivery



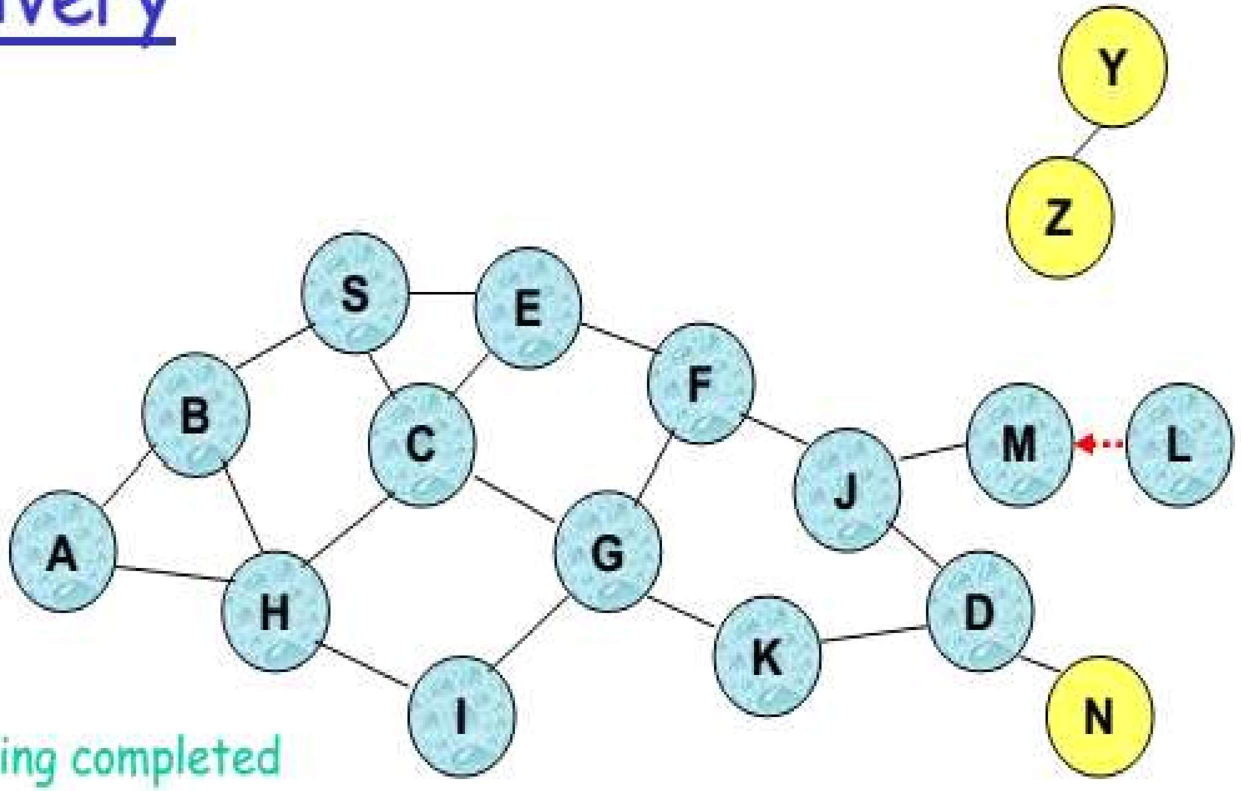
- Nodes J and K both broadcast packet P to node D
- Since nodes J and K are hidden from each other, their transmissions may collide
  - => Packet P may not be delivered to node D at all, despite the use of flooding

Flooding for Control Packet Delivery



- Node D does not forward packet P, because node D is the intended destination of packet P

Flooding for Control Packet Delivery



- Flooding completed
- Nodes unreachable from S do not receive packet P (e.g., node Z)
- Nodes for which paths go through the destination D also do not receive packet P (example: node N)

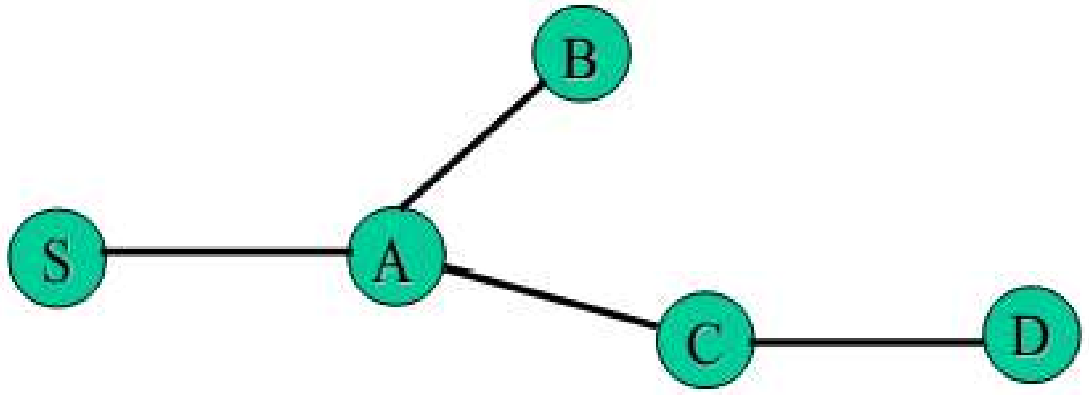
Route Discovery (Contd.)

- Once an intermediate node receives a RRREQ, the node sets up a reverse route entry for the source node in its route table
  - Reverse route entry consists of <Source IP address, Source seq. number, number of hops to source node, IP address of node from which RREQ was received>
  - Using the reverse route a node can send a RREP (Route Reply packet) to the source
    - Reverse route entry also contains – life time field
- RREQ reaches destination -> In order to respond to RREQ a node should have in its route table:
  1. Unexpired entry for the destination
  2. Seq. number of destination at least as great as in RREQ (for loop prevention)

Route Discovery (Contd.)

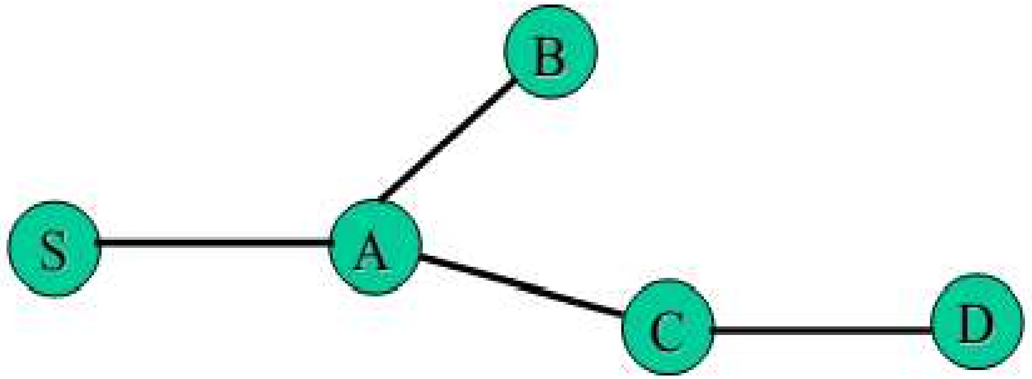
- RREQ reaches destination (contd.)
  - If both conditions are met & the IP address of the destination matches with that in RREQ → the node responds to RREQ by sending a RREP back using unicasting and not flooding to the source using reverse path
  - If conditions are not satisfied, then node increments the hop count in RREQ and broadcasts to its neighbors
- Ultimately the RREQ will make to the destination

AODV – Route Discovery - Example



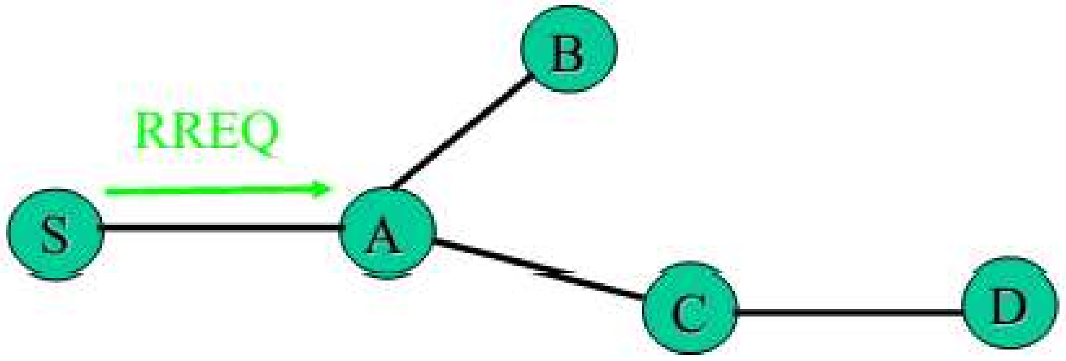
1. Node S needs a route to D

AODV – Route Discovery



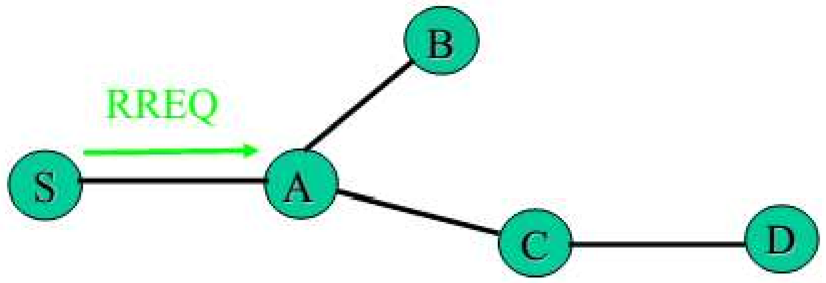
1. Node S needs a route to D
2. Creates a Route Request (RREQ)
   - Enters D's IP addr, seq#, S's IP addr, seq#, hopcount (=0)

AODV – Route Discovery



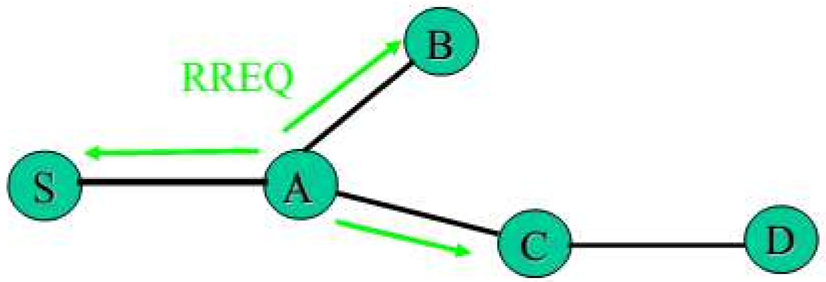
1. Node S needs a route to D
2. Creates a Route Request (RREQ)
   - Enters D's IP addr, seq#, S's IP addr, seq#, hopcount (=0)
3. Node S broadcasts RREQ to neighbors

AODV – Route Discovery



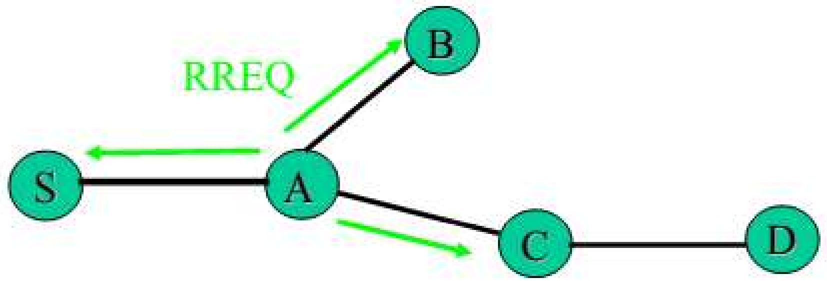
4. Node A receives RREQ
   - Makes a reverse route entery for S
     dest=S, nexthop=S, hopcount=1
   - It has no routes to D, so it rebroadcasts RREQ

AODV – Route Discovery



4. Node A receives RREQ
   - Makes a reverse route entery for S
     dest=S, nexthop=S, hopcount=1
   - It has no routes to D, so it rebroadcasts RREQ

AODV – Route Discovery



5. Node C receives RREQ
   - Makes a reverse route entry for S
     dest=S, nexthop=A, hopcount=2
   - It has a route to D, and the seq# for route to D is >= D's seq# in RREQ

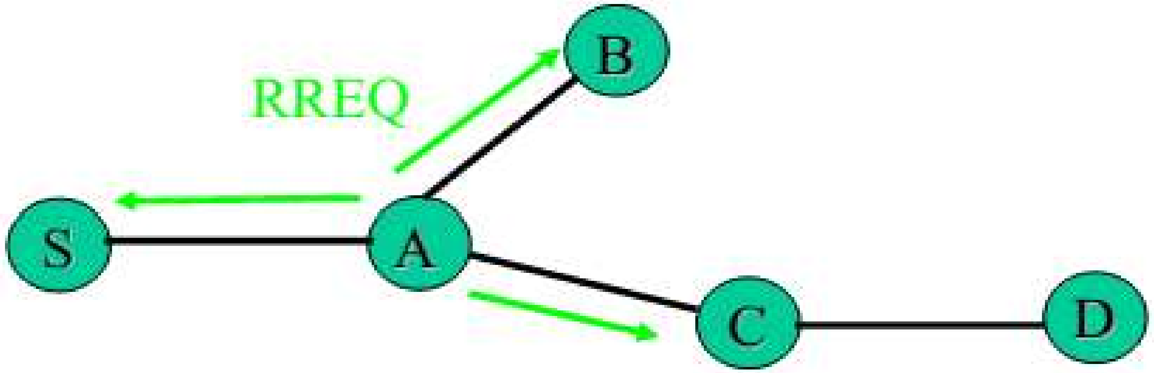
Route Reply in AODV - Comments

- An intermediate node (not the destination) may also send a Route Reply (RREP) provided that it knows a more recent path than the one previously known to sender S

- To determine whether the path known to an intermediate node is more recent, *destination sequence numbers* are used

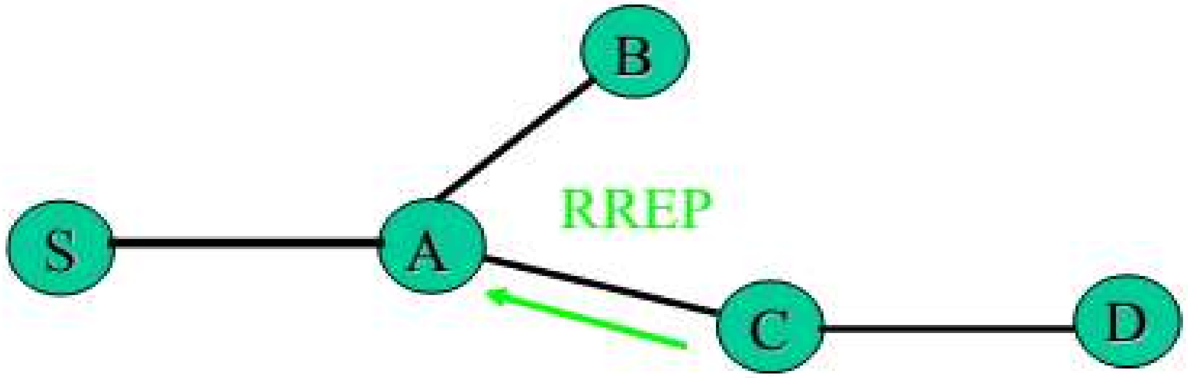
- The likelihood that an intermediate node will send a Route Reply when using AODV not as high as DSR (Later)
  - A new Route Request by node S for a destination is assigned a higher destination sequence number. An intermediate node which knows a route, but with a smaller sequence number, cannot send Route Reply

AODV - Route Discovery



5. Node *C* receives RREQ (Cont.)
   - *C* creates a Route Reply (RREP)
     Enters D's IP addr, *seq#*, *S*'s IP addr, hopcount to D (=1)
   - Unicasts RREP to *A*

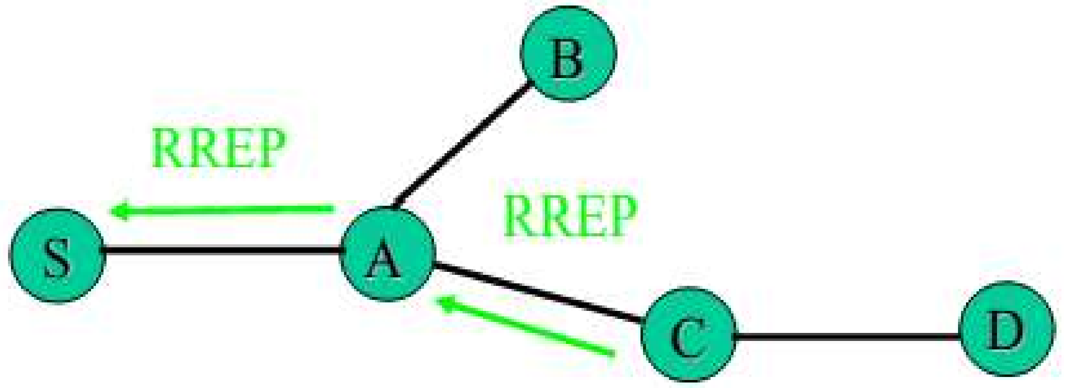
AODV - Route Discovery



5. Node *C* receives RREQ (Cont.)

- *C* creates a Route Reply (RREP)
  Enters D's IP addr, *seq#*, *S*'s IP addr, hopcount to D (=1)
- Unicasts RREP to *A*

AODV - Forward Path Setup



5. Node A receives RREP
   - Makes a forward route entry to D
     dest=D, nexthop=C, hopcount=2
   - Unicasts RREP to S

Forward Path Setup (1)

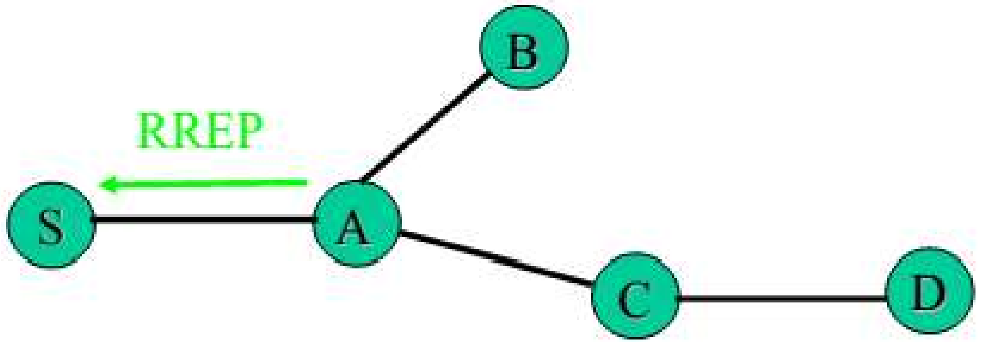
- When a node determines that it has a current route to respond to RREQ i.e. has a path to destination – It creates RREP (Route Reply)
- RREP contains <IP address of source and destination>
  - If RREP is being sent by destination
    - The RREP will also contain the <current sqn # of destination, hop-count=0, life-time>
  - If RREP is sent by an intermediate node
    - RREP will contain its record of the <destination sequence number, hop-count=its distance to destination, its value of the life-time>

Forward Path Setup (2)

- When an intermediate node receives the RREP, it sets up a forward path entry to the destination in its route table
  - Forward path entry contains

    <IP Address of destination, IP address of node from which the entry arrived, hop-count to destination, life-time>
  - To obtain its distance to destination i.e. hop-count, a node increments its distance by 1
  - If route is not used within the life time, its deleted
- After processing the RREP, the node forwards it towards the source

AODV - Forward Path Setup

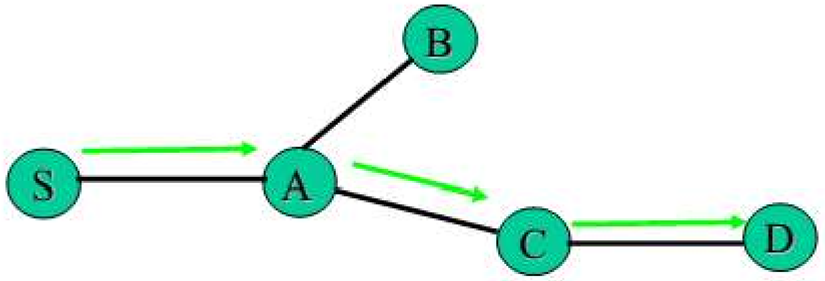


5. Node S receives RREP
   - Makes a forward route entry to D
     dest=D, nexthop =A, hopcount = 3

Receipt of Multiple RREP

- A node may receive multiple RREP for a given destination from more than one neighbor
  - The node only forwards the first RREP it receives
  - May forward another RREP if that has greater destination sequence number or a smaller hop count
  - Rest are discarded -> reduces the number of RREP propagating towards the source
- Source can begin data transmission upon receiving the first RREP

AODV – Data Delivery



5. Node S receives RREP
   - Makes a forward route entry to D
     dest=D, nexthop =A, hopcount = 3
   - Sends data packet on route to D

Link Failure Reporting

- A neighbor of node X is considered active for a routing table entry if the neighbor sent a packet within *active_route_timeout* interval and was forwarded using that entry

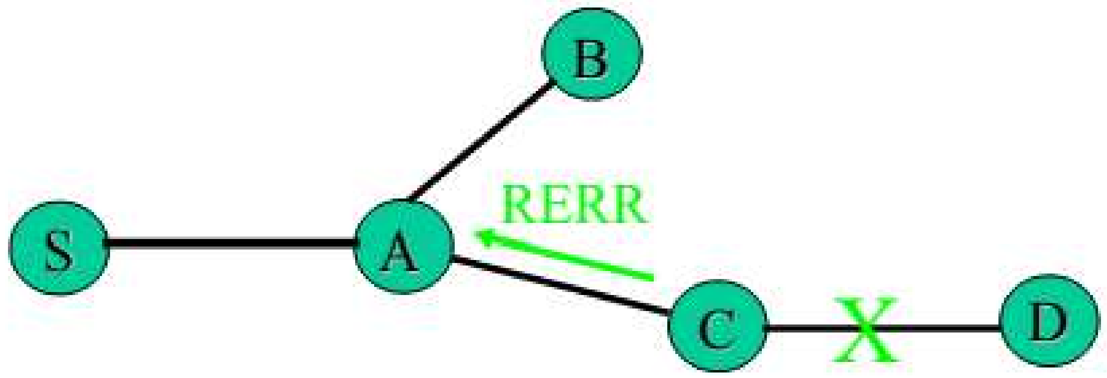
- If a source node moves, a new route discovery process is initiated

- If intermediate nodes or the destination move ->
  - The next hop links break resulting in link failures
  - Routing tables are updated for the link filaures
  - All active neighbors are informed by RERR message

Route Maintenance - RERR

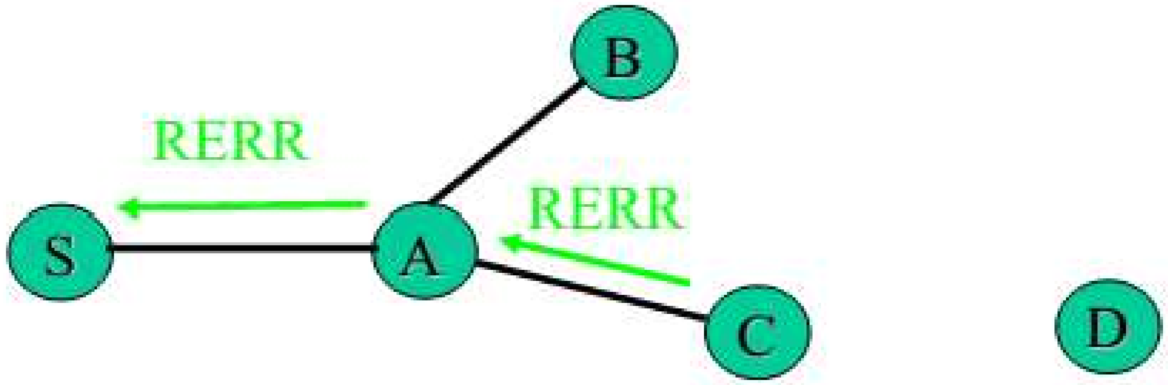
- RERR is initiated by the node upstream (closer to the source) of the break
  - Its propagated to all the affected destinations
  - RERR lists all the nodes affected by the link failure -> Nodes that were using the link to route messages (precursor nodes)
  - When a node receives an RERR, it marks its route to the destination as invalid -> Setting distance to the destination as infinity in the route table
- When a source node receives an RRER, it can reinitiate the route discovery

AODV - Route Maintenance- Example



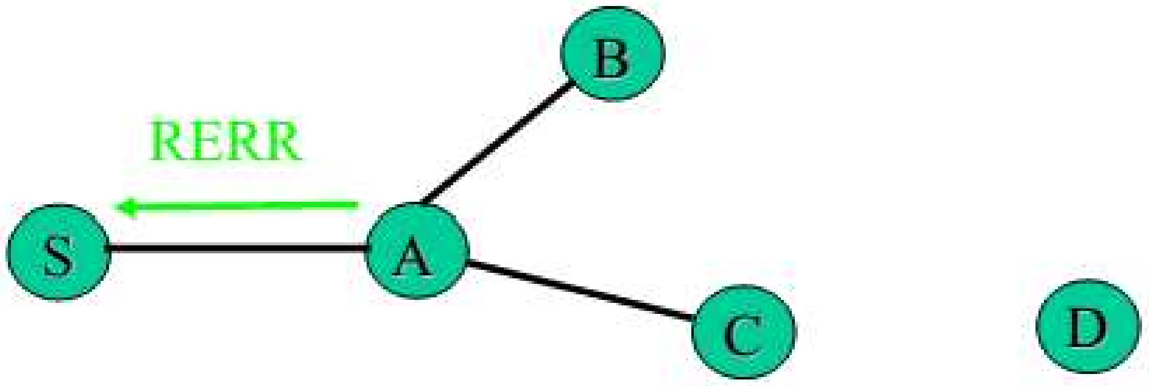
1. Link between C and D breaks
2. Node C invalidates route to D in route table
3. Node C creates Route Error message
   - ❒ Lists all destinations that are now unreachable
   - ❒ Sends to upstream neighbors

AODV - Route Maintenance - Example



4. Node A receives RERR
   - Checks whether C is its next hop on route to D
   - Deletes route to D (makes distance -> infinity)
   - Forwards RERR to S

AODV - Route Maintenance-Example



5. Node S receives RERR
   - Checks whether A is its next hop on route to D
   - Deletes route to D
   - Rediscovers route if still needed