Mobility and MANET (II)
Intelligent Transportation Systems

Rolland Vida

MANET routing

- **Proactive routing**
  - The routing table is continuously maintained
    - No matter if there is traffic or not
  - Relatively stable networks
  - DSDV – based on the Bellman-Ford algorithm
- **On demand, reactive routing**
  - Builds a route only if needed, if a packet has to be sent to the destination
  - The routes are temporary, are dismantled if not used
  - AODV
- Hybrid protocols
  - Combining the previous two
- Position-based protocols
  - Makes use of geographical position information for routing

Constraints

- Delay
  - Proactive protocols provide lower delay, as routes are prepared in advance, and always up to date, ready to use
  - Reactive protocols provide large delay, as the route from A to B has to be found, when needed
- Overhead
  - Proactive protocols have a large overhead, too much signaling traffic to build and maintain the routes, even if no real data to send
  - Reactive protocols have lower overhead, useless routes are not maintained

- Each application will choose the best protocol
  - Low mobility -> Proactive protocols
  - High mobility -> Reactive protocols

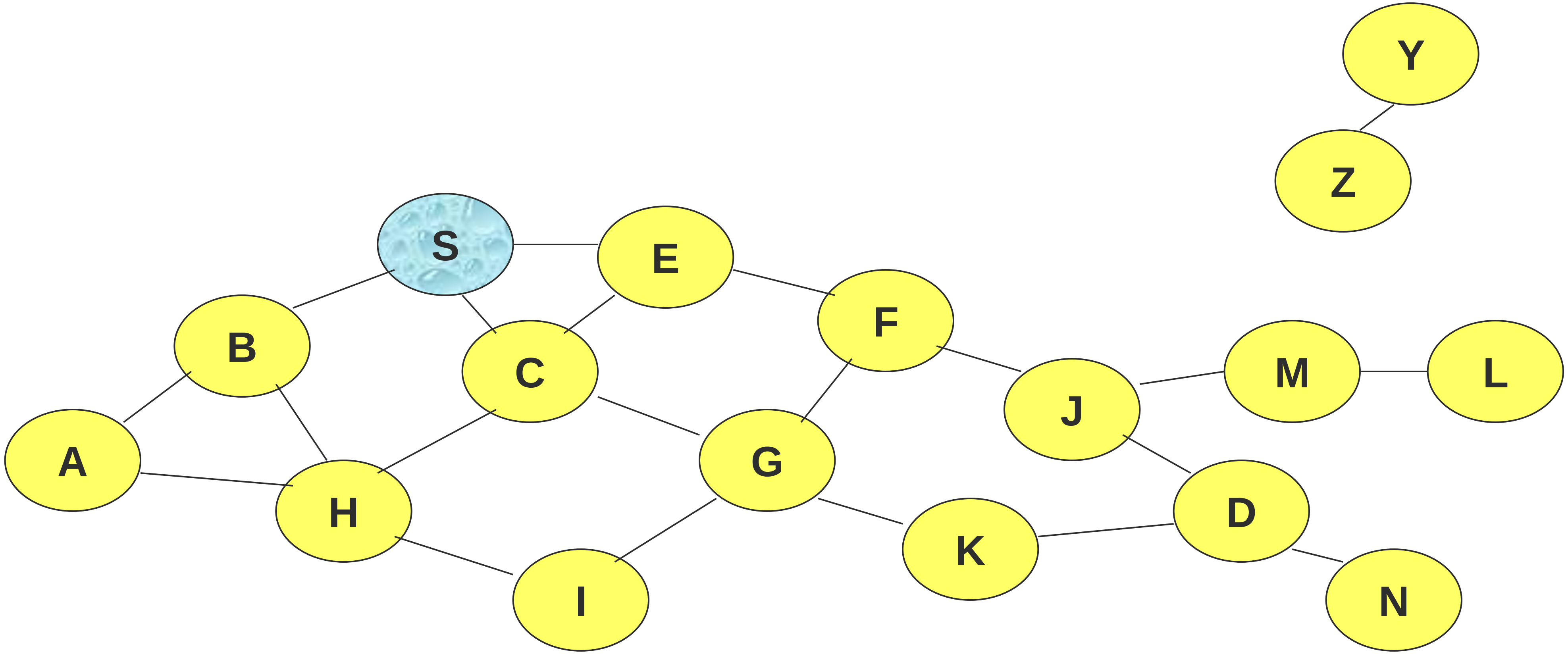
Ad Hoc On-Demand Distance Vector Routing (AODV)

- Reactive protocol
  - Maintains a routing table in each node, no need to store the route in the packet header
  - The route is built and maintained only if it is „active"

AODV

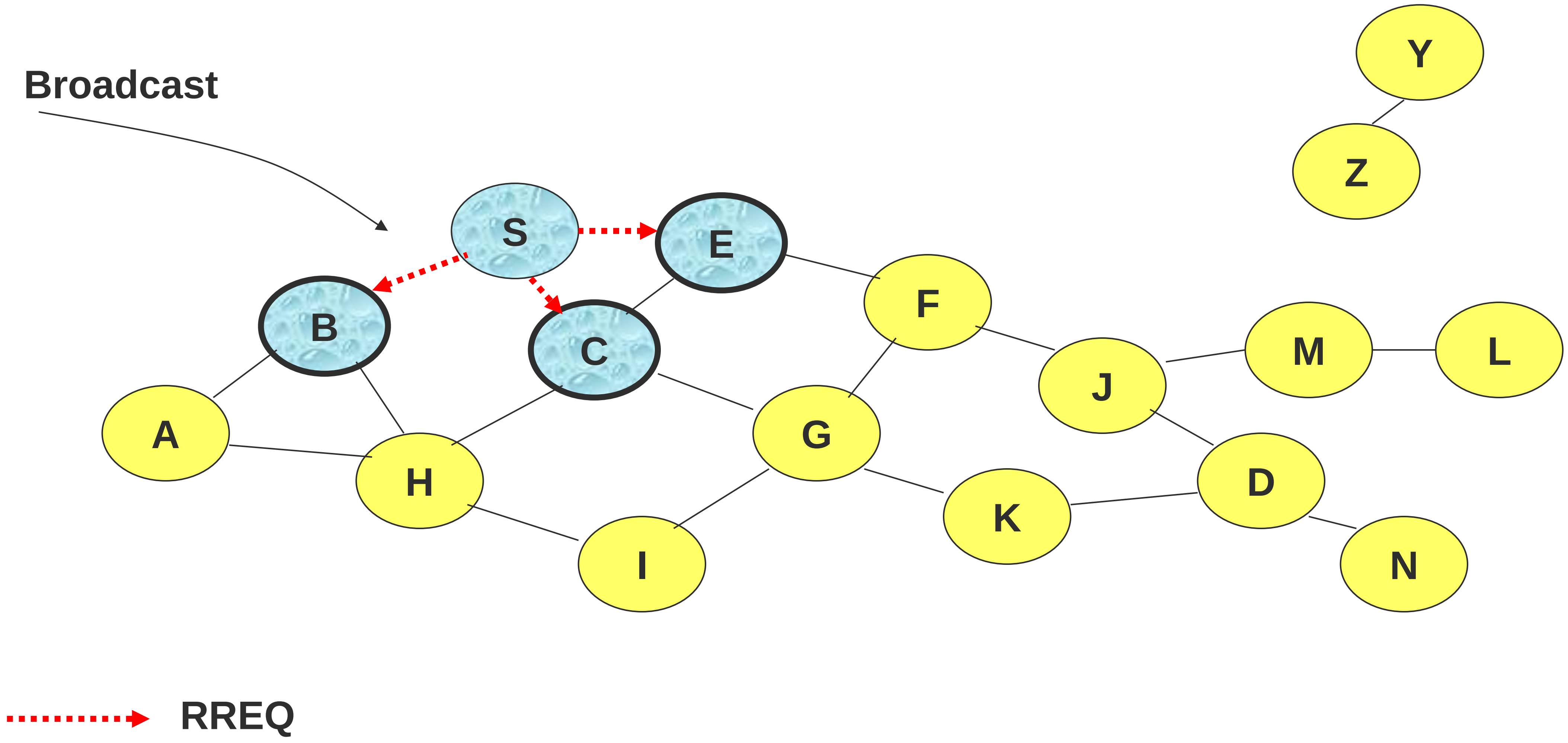
- To discover the route, the source broadcasts a **Route Request (RREQ) message**
- Those who receive it, rebroadcast it
- When a node rebroadcasts a Route Request message, it stores a **reverse path pointer** towards the node from where the request came
  - AODV symmetric (bi-directional) links
  - A small timer ensures that these records time out after a while
- If the RREQ arrives to the destination D, a **Route Reply (RREP)** message is sent back
- It will propagate along the path built from the reverse path pointers

Route Request - AODV

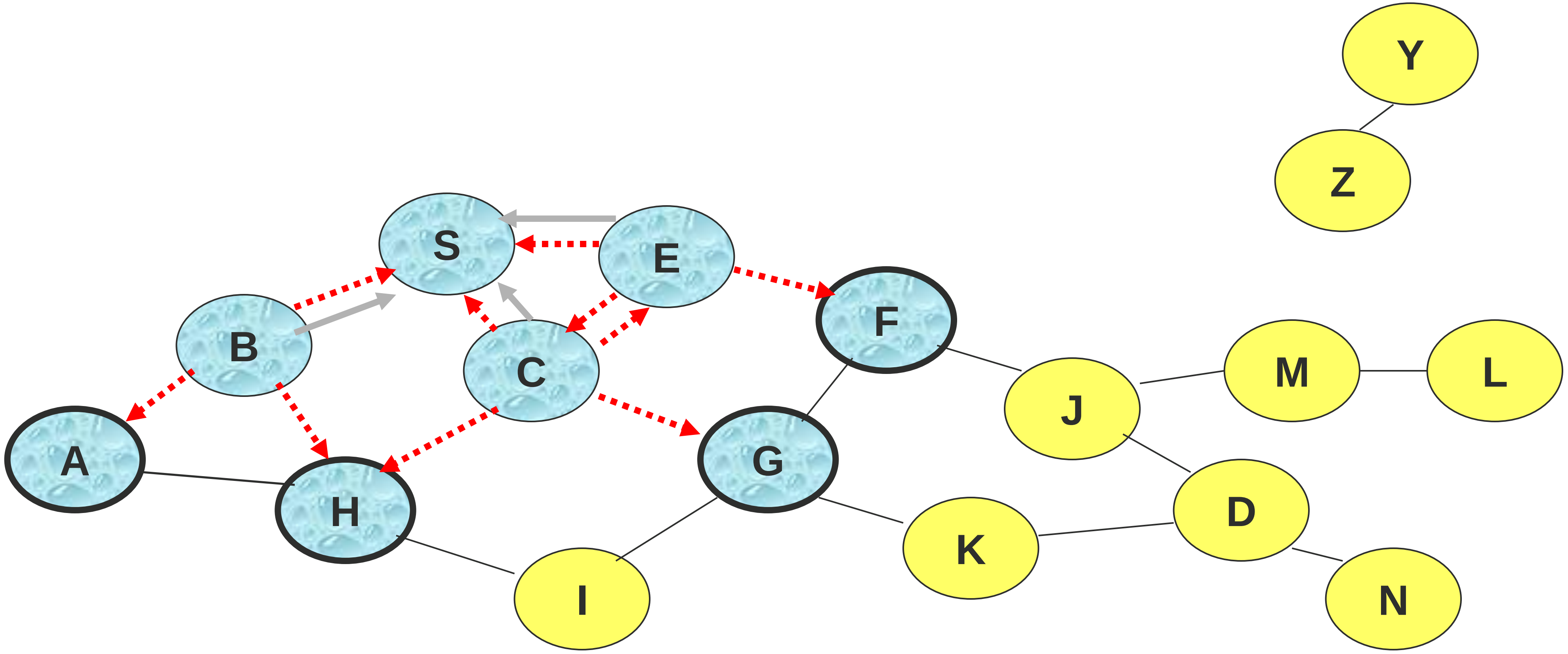


Node that already received a RREQ for D initiated by S

Route Request - AODV

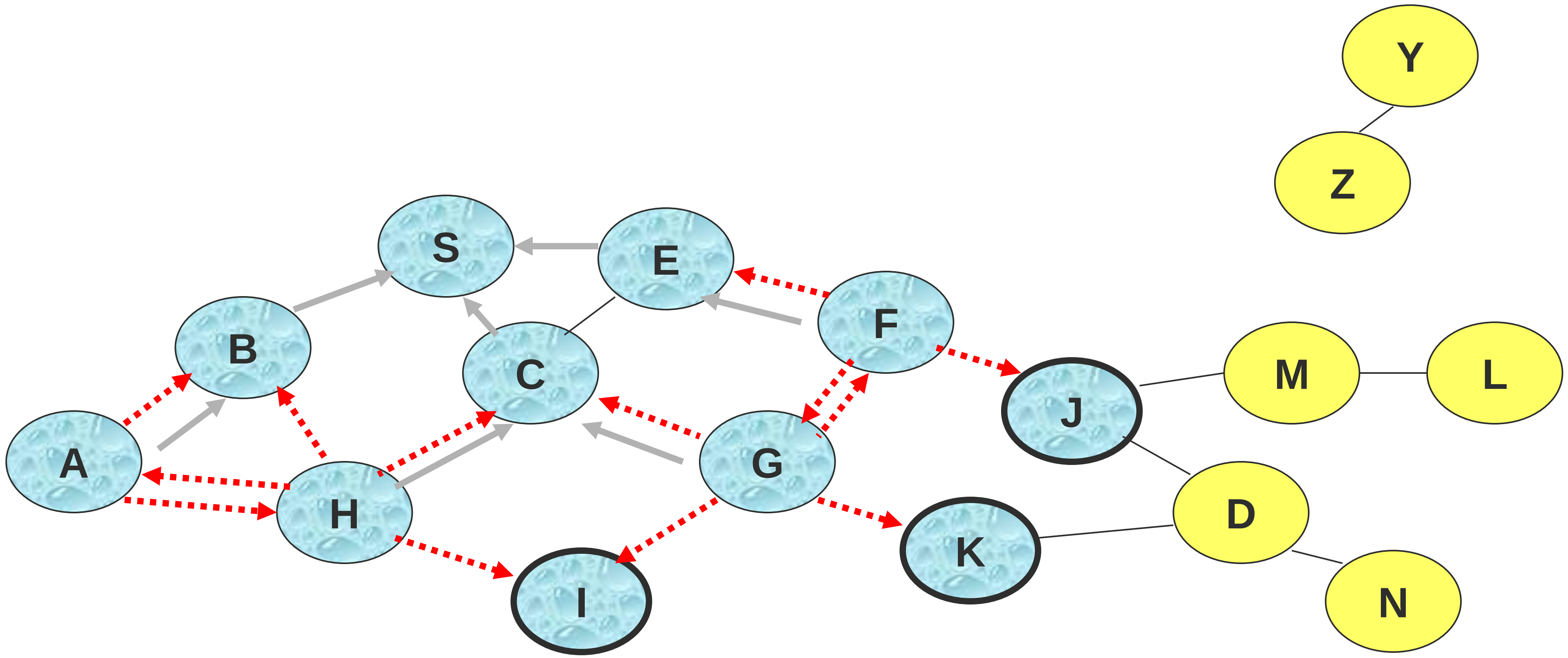


Route Requests - AODV



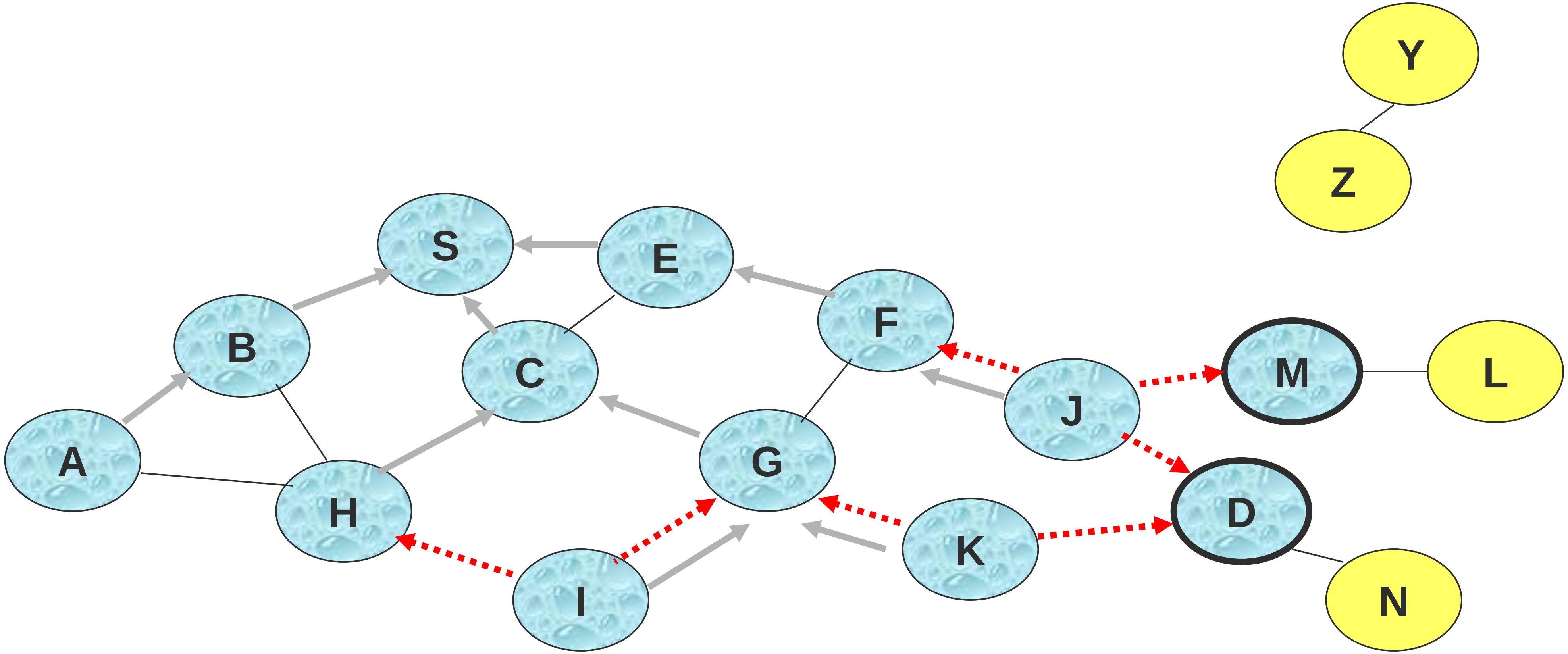
Reverse Path pointer

Reverse Path - AODV

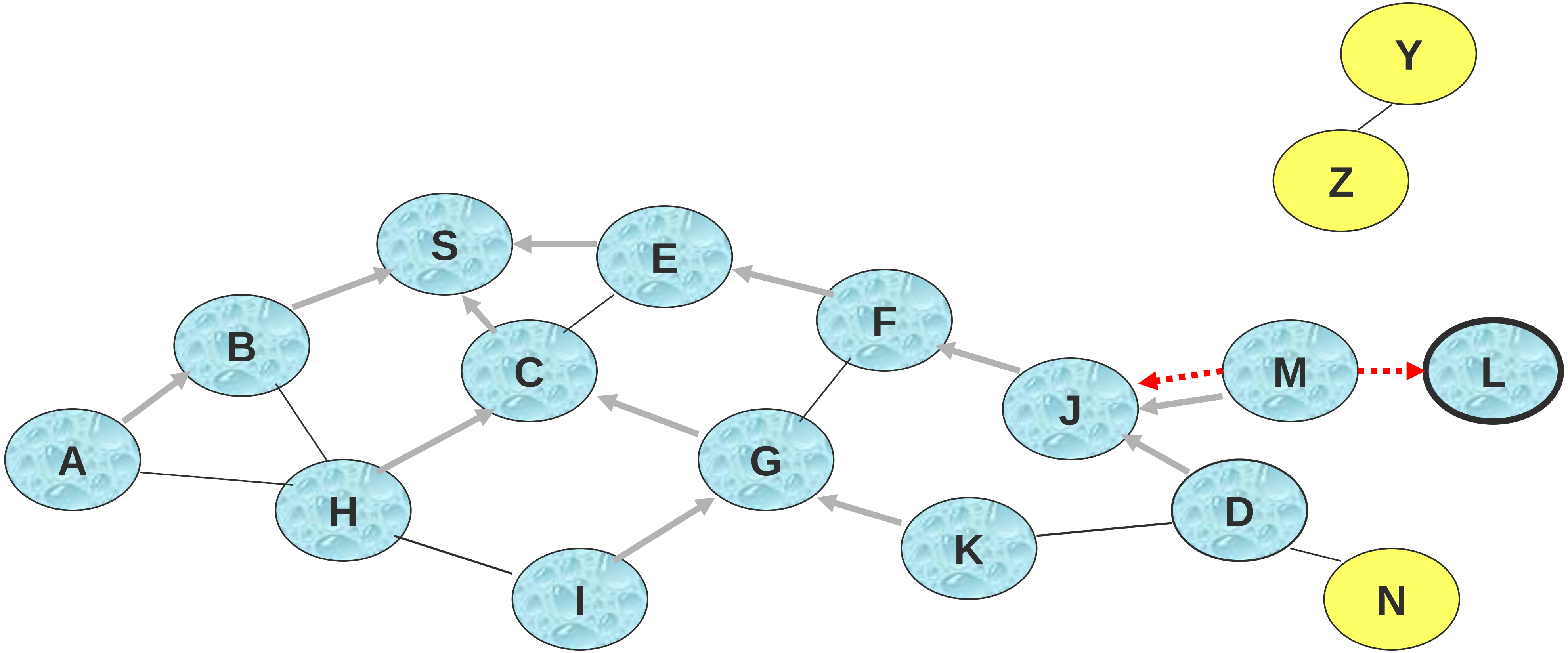


- C receives a RREQ from neighbors (G and H)
But does not rebroadcast it again

Reverse Path - AODV

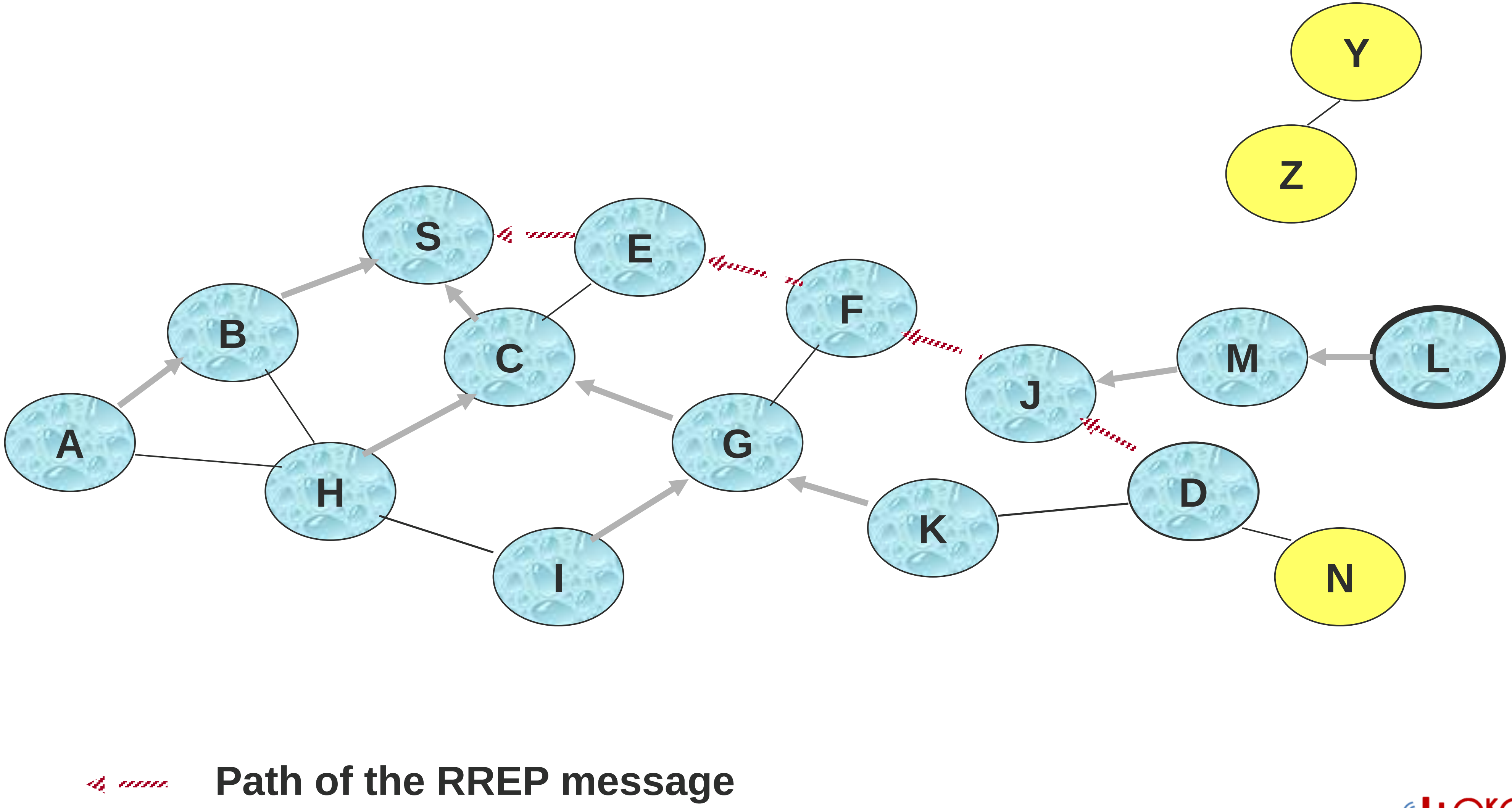


Reverse Path - AODV

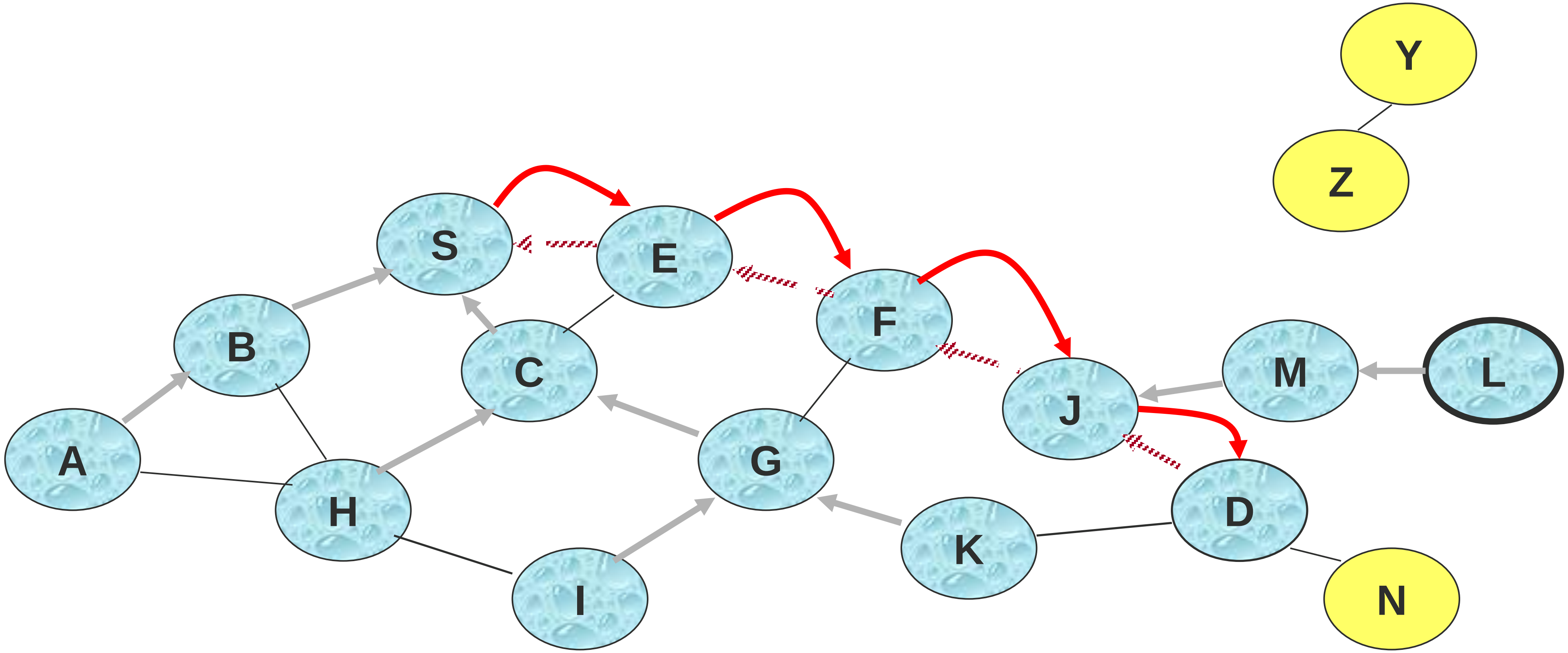


- **node D does not forward anymore the RREQ message, as he is the destination**

Route Reply - AODV



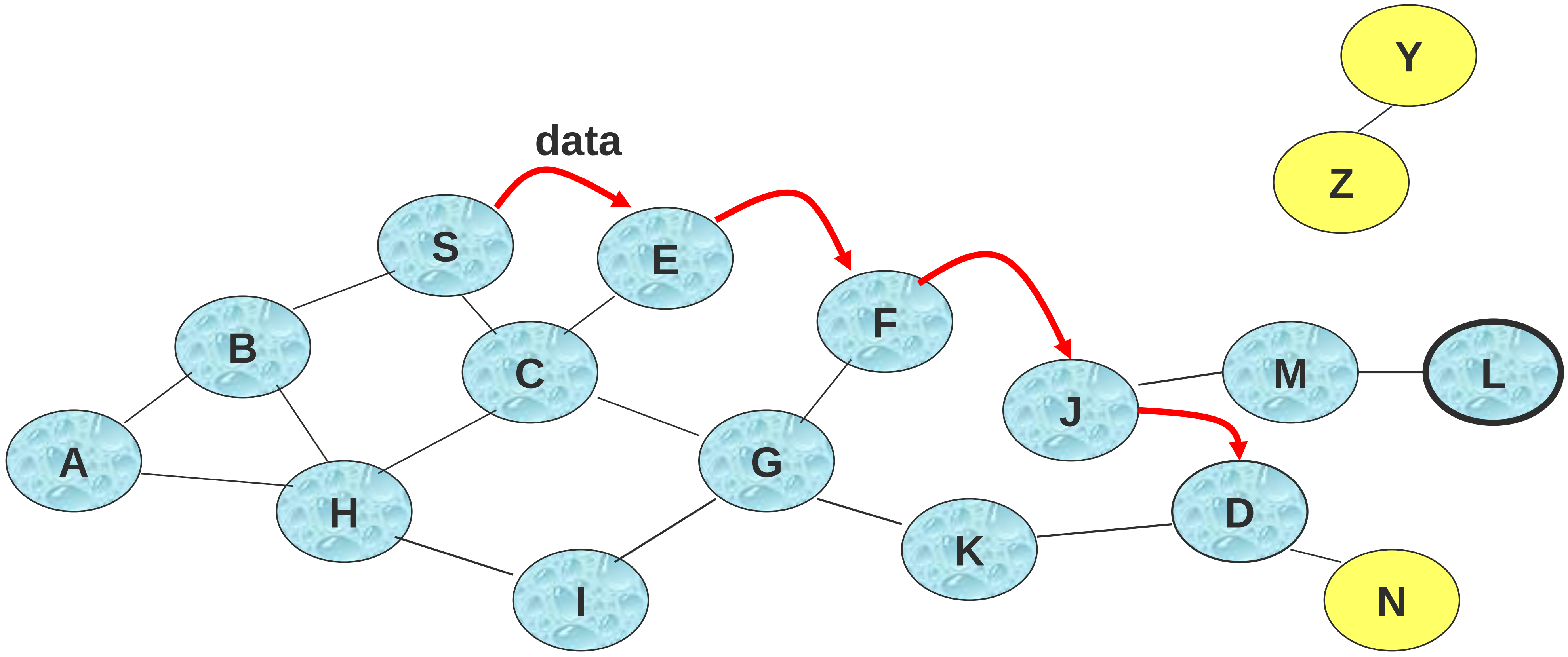
Forward Path - AODV



As the RREP message travels from D to S, forward path pointers are stored in the intermediate nodes

Forward path pointer

Data sending - AODV



For sending the useful data, these forward path pointers are used

The path is not included in the header

Timers

- The reverse path records are deleted after a while from the routing tables
  - We should take into account the specificities of the wireless domain and the size of the network, leave time for the RREP message to propagate back before deleting the record
- The forward path pointer is deleted if it becomes inactive – no traffic
  - *active_route_timeout*
  - If no traffic, the record is deleted, even if the path is still valid

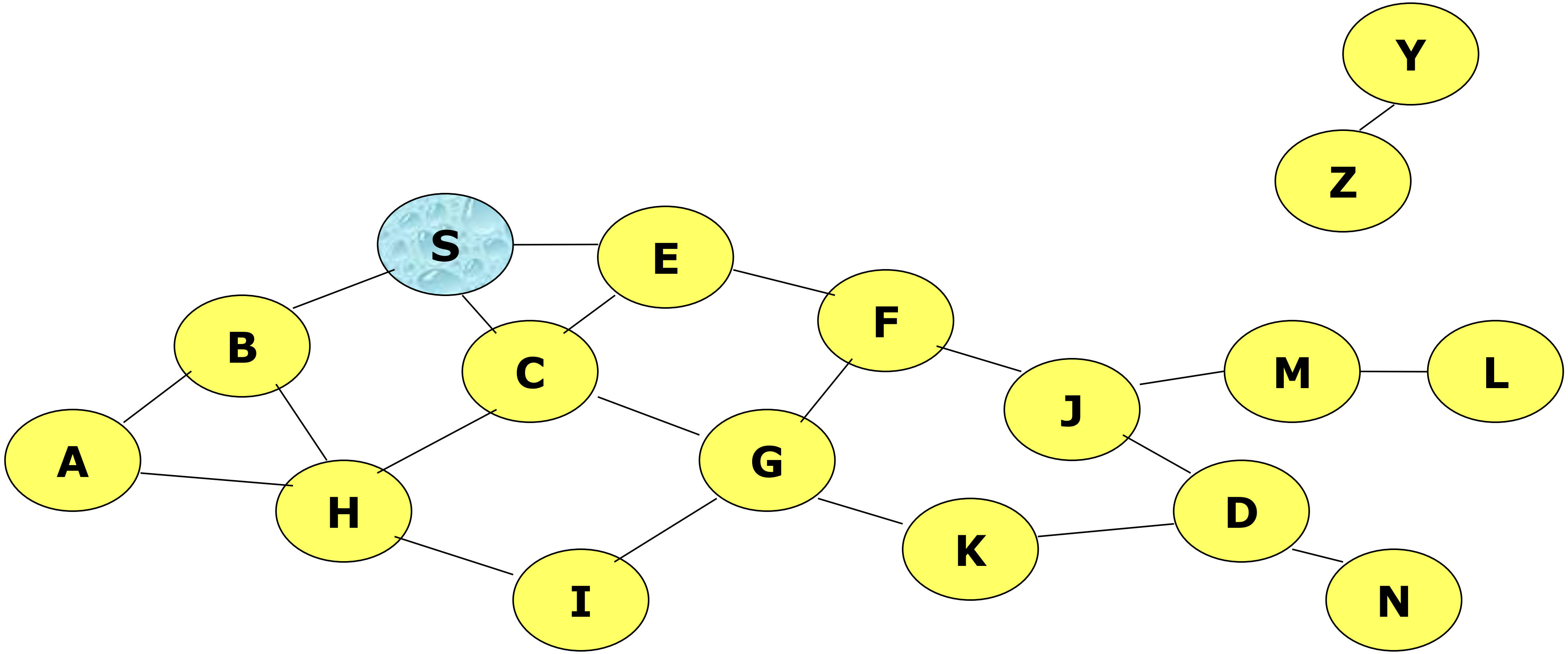
Optimization: Expanding Ring Search

- Searching a expanding territory
- The RREQ messages are first sent out with a small Time-to-Live (TTL) value
  - After each hop the TTL value is decreased
  - If 0, the message is dropped
  - Used in many protocols that are based on flooding
- If no Route Reply until a timer expires, the value of the TTL is increased
  - After a few steps the search will cover the entire topology

Dynamic Source Routing (DSR)

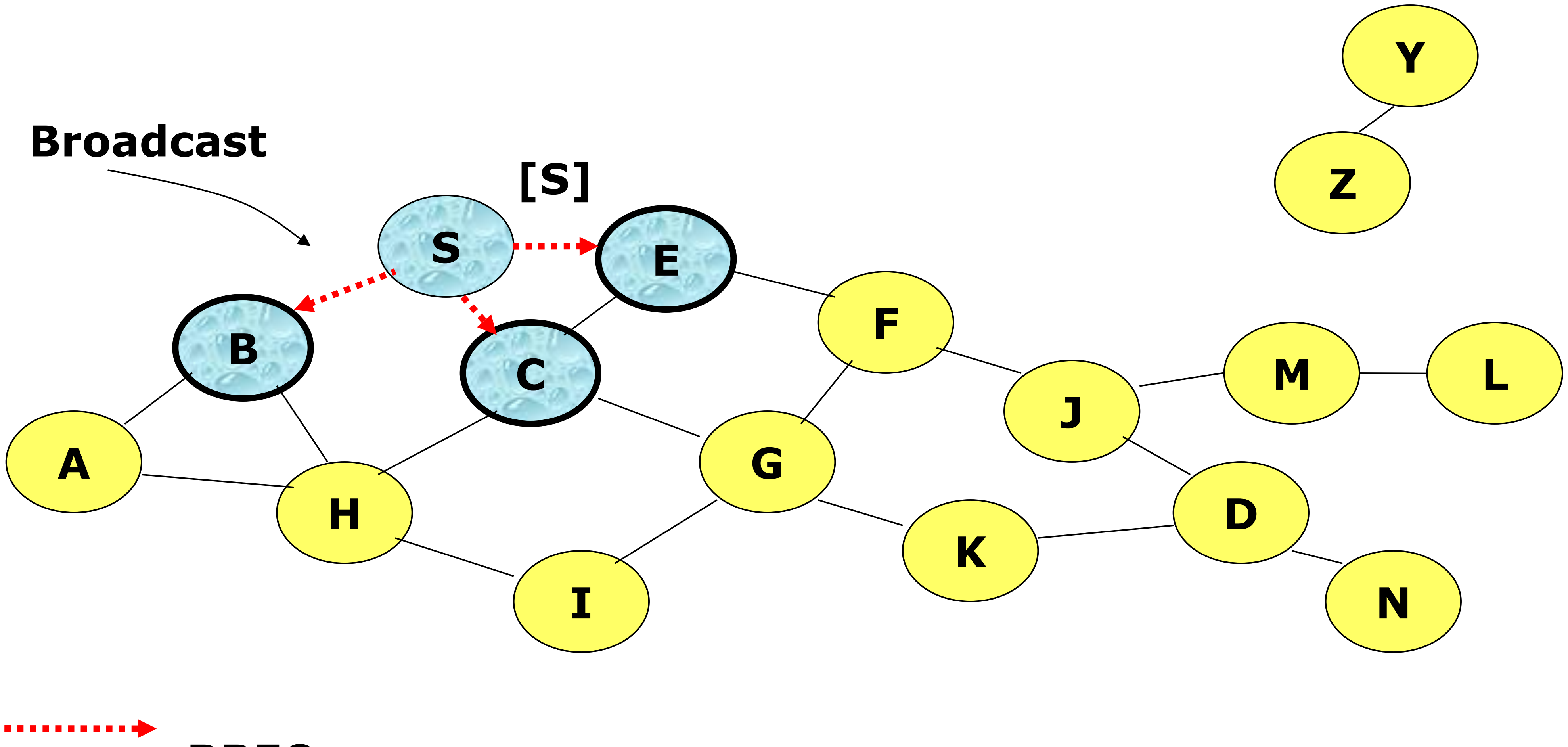
- If the source **S** wants to send something to destination **D,** it initiates route discovery
- **S** floods the network with Route Request (**RREQ**) messages
- Each intermediate node adds his ID to the RREQ before forwarding it

DSR Route Discovery



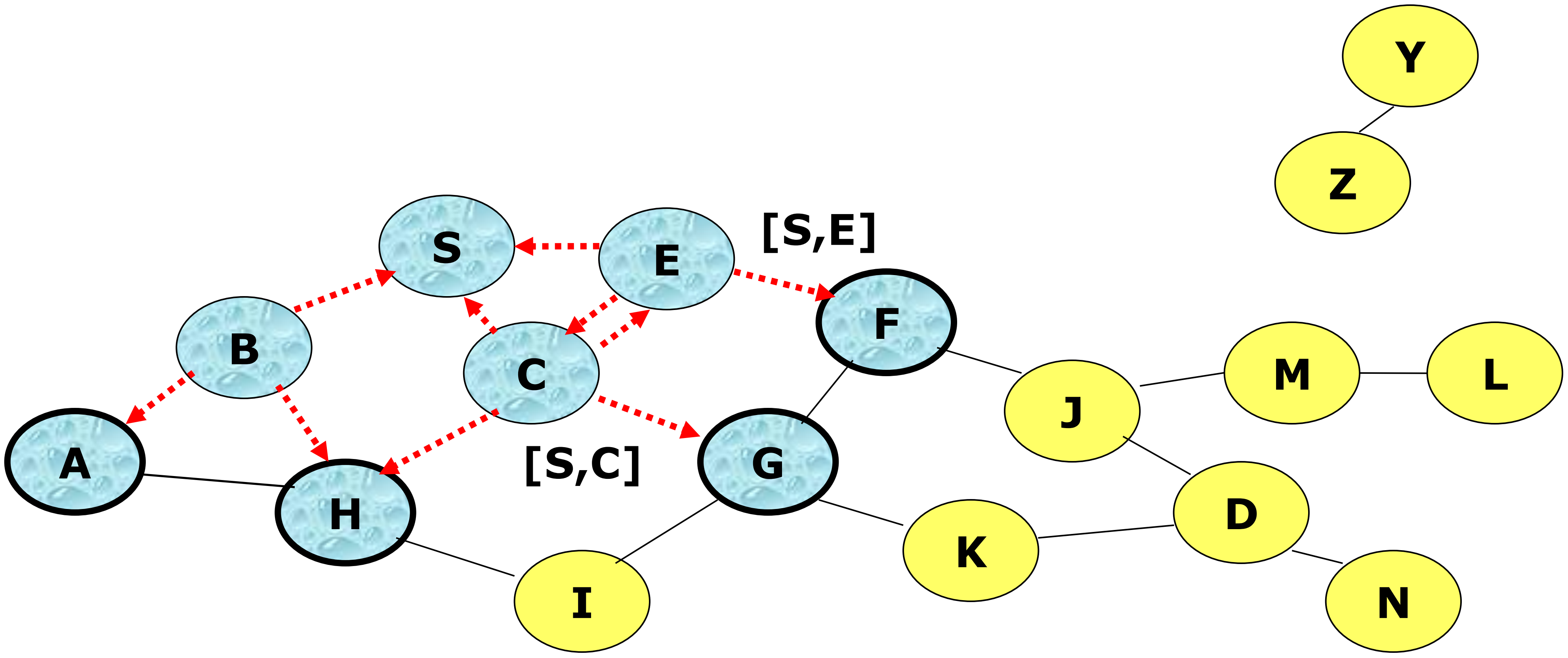
Nodes that already received the RREQ from S, regarding D

DSR Route Discovery

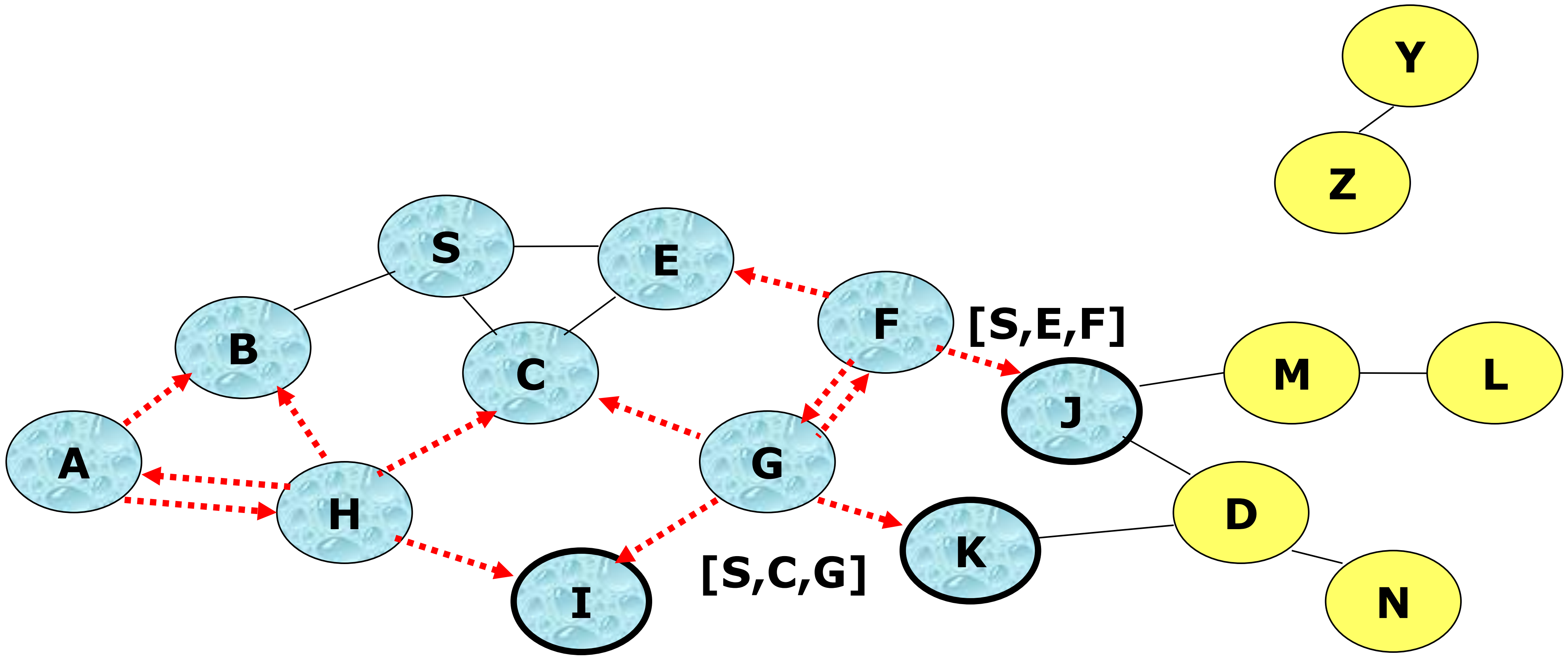


RREQ
[X, ...] intermediate nodes already added to the path

DSR Route Discovery

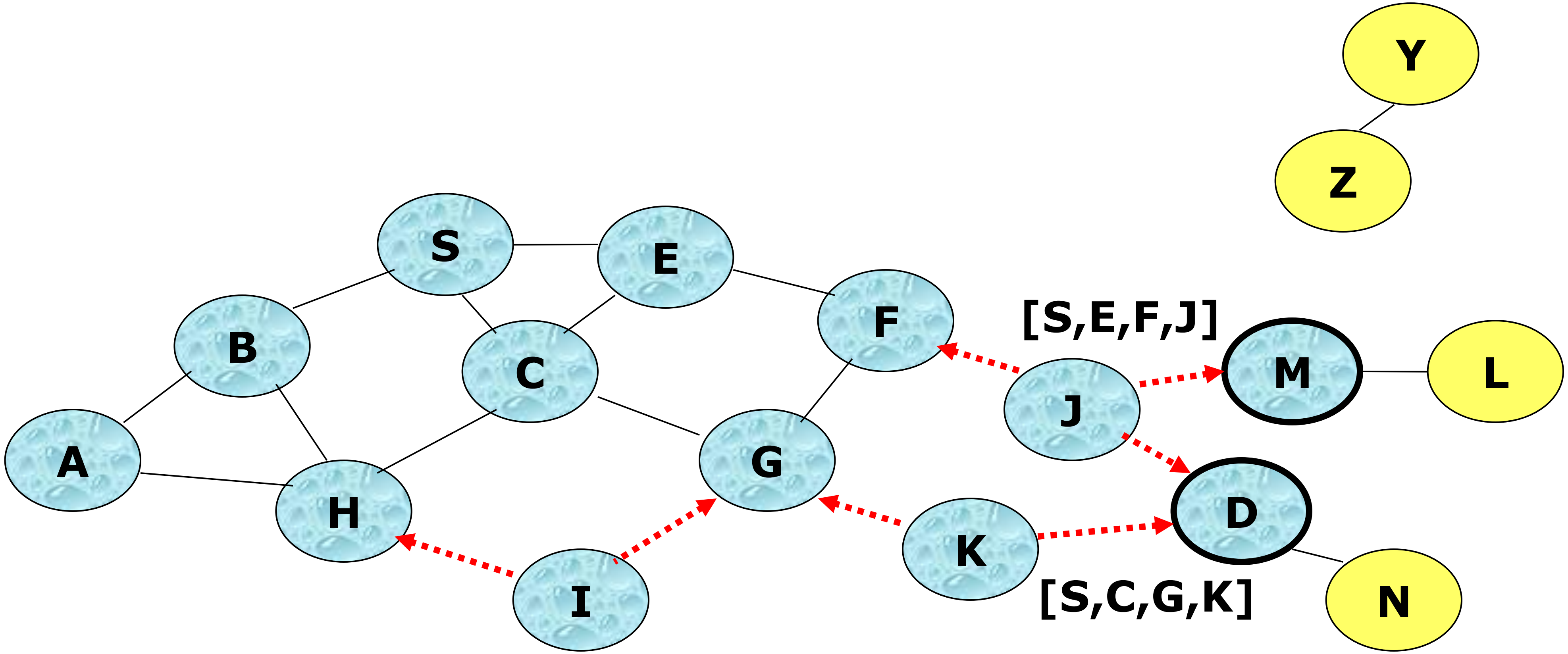


DSR Route Discovery



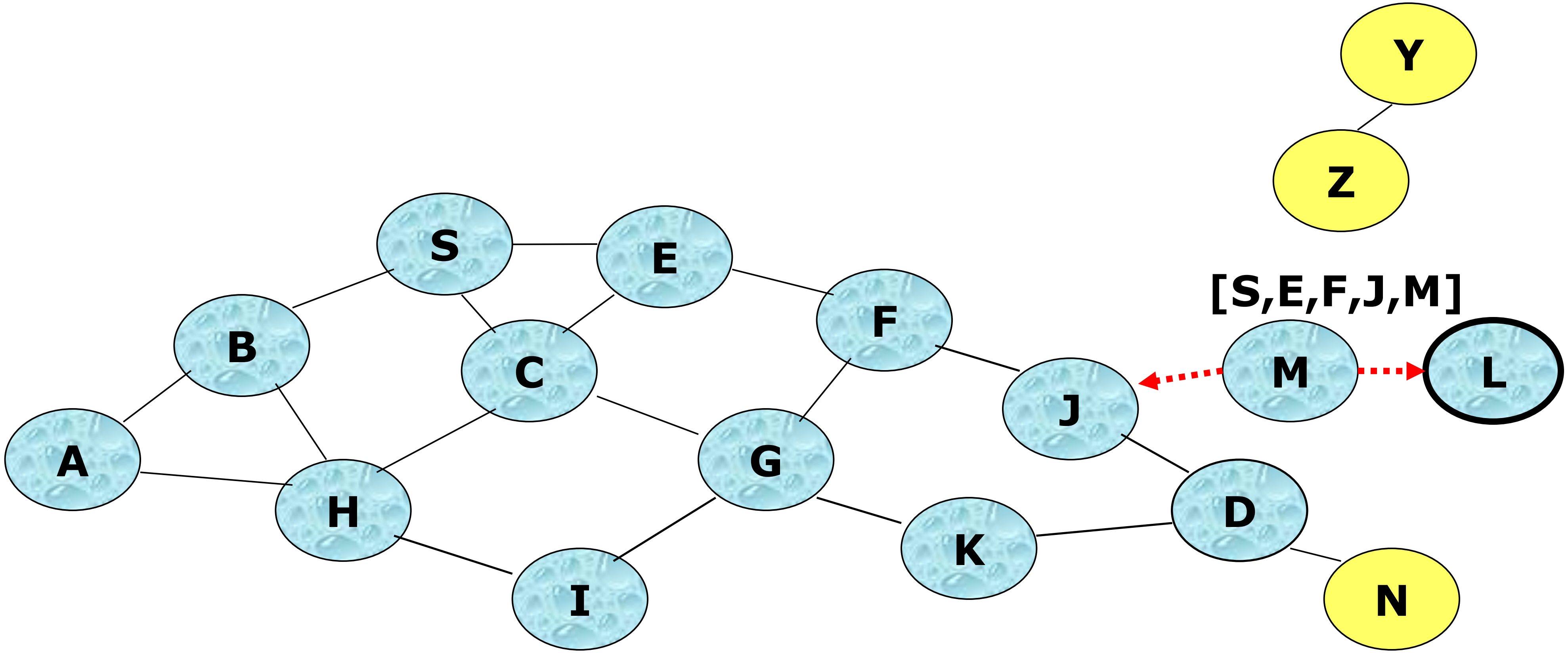
Restricting the flooding (like in AODV):
C receives the RREQ again from G and H, but does not forward it again

DSR Route Discovery



D receives the RREQ from K and J, so the two messages can collide

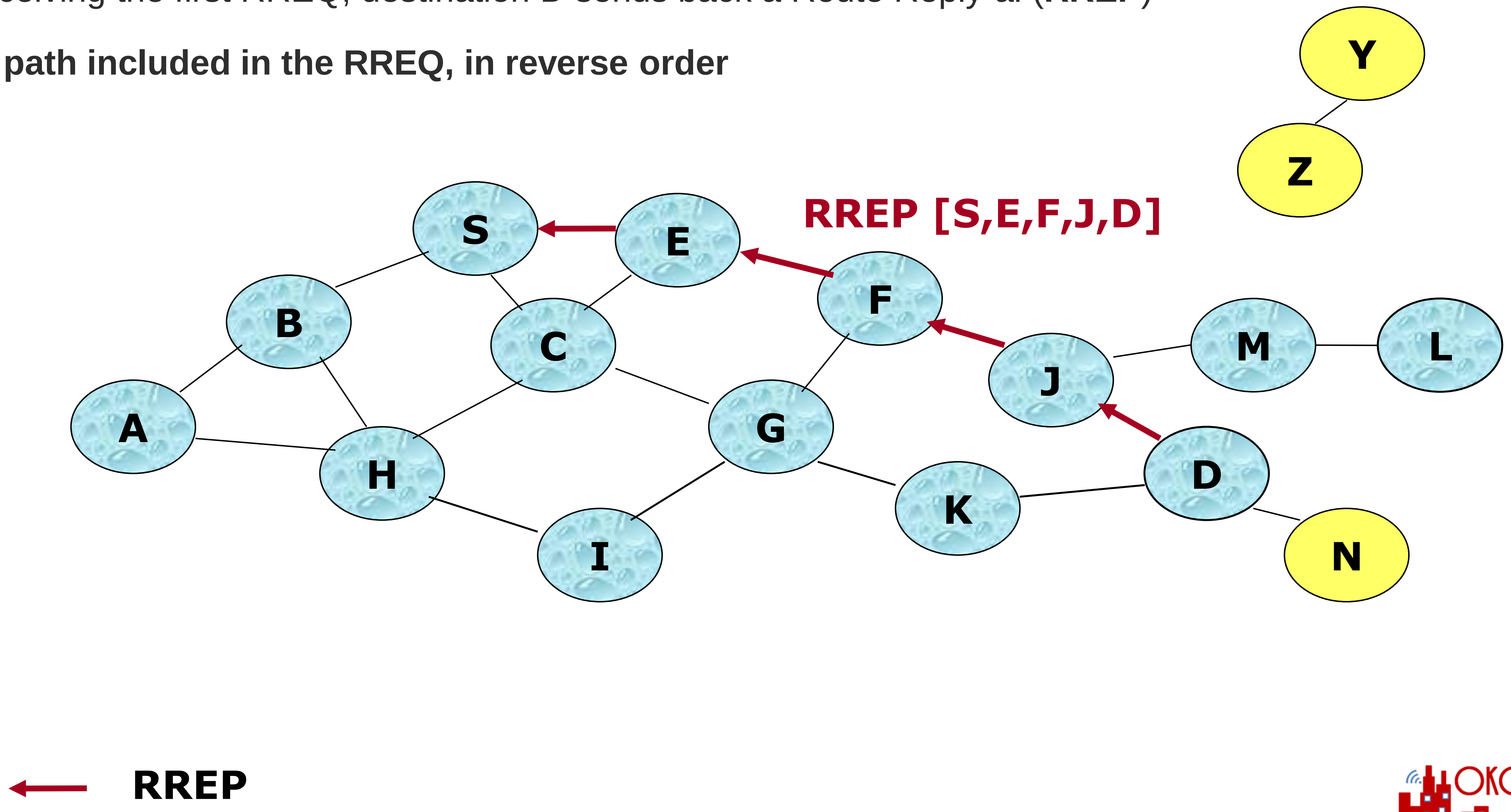
DSR Route Discovery



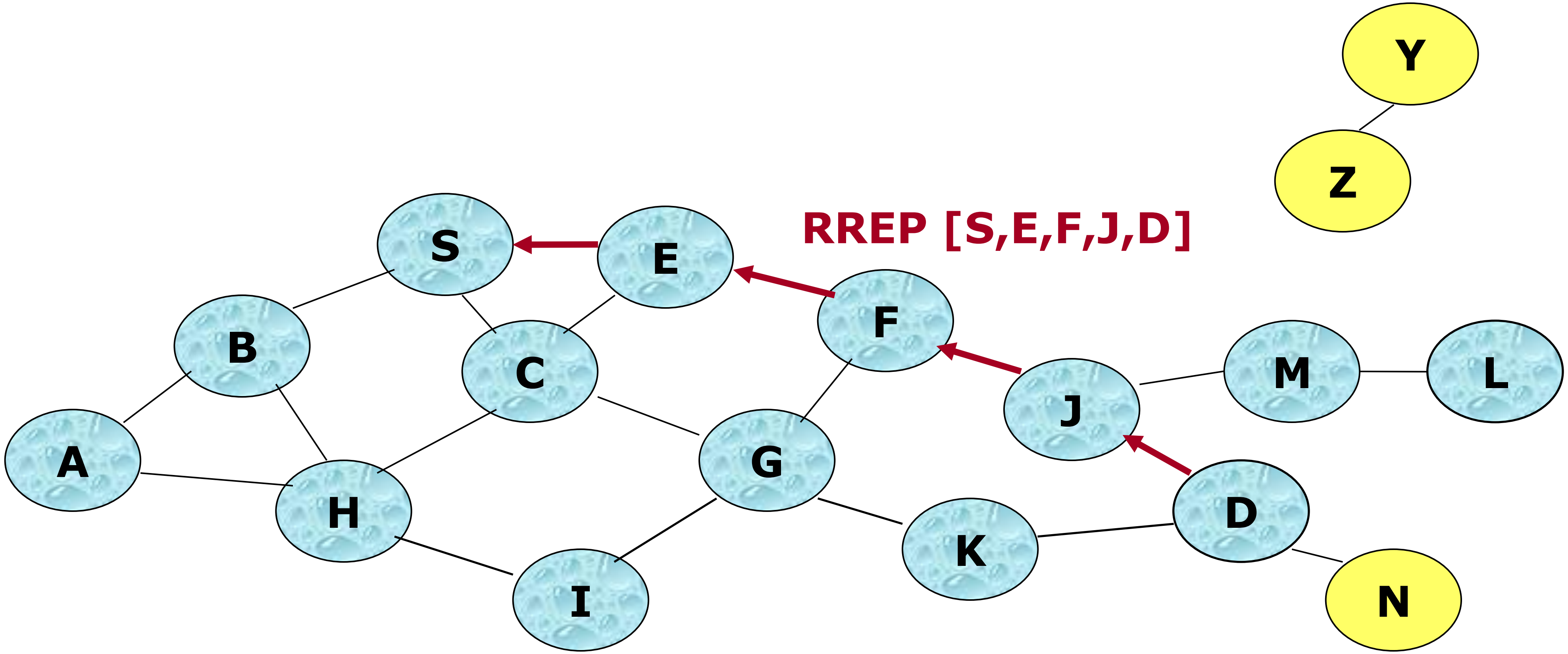
D stops the broadcast of RREQ, as it is the destination

DSR Route Discovery

- After receiving the first RREQ, destination D sends back a Route Reply-al (**RREP**)
- **On the path included in the RREQ, in reverse order**

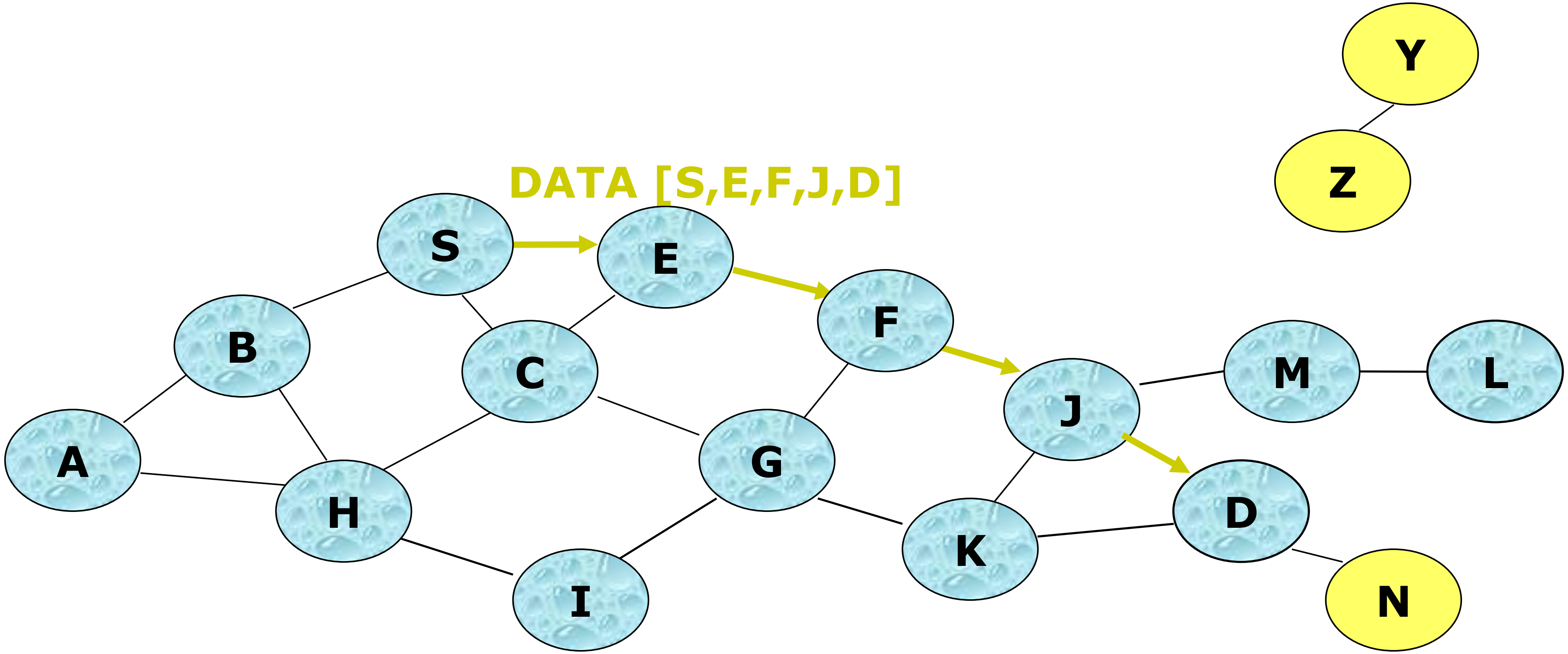


DSR Route Reply



RREP routing message

Data Delivery in DSR



The header increases with the path length

Position-based routing

- Eliminate some drawbacks of the topology-based routing algorithms

- The source makes use of some **localization service**, and finds out the geographical position of the destination.

- If we know the position of the destination, there is no need to build and maintain routes in advance
  - Instead of building the routes, we need a **forwarding strategy**
  - At each node, we select the next hop based on the position of the destination and the position of the neighbors

Localization service

- **Localization service**
  - Helps a node to determine its position
  - In an ad hoc network a localization server is not always available
  - „Chicken-egg” problem: but how do we know the position of the localization server?

- The localization service can be provided by one or several nodes:
  - „some/all to some/all”

- If a source does not know the position of the destination, makes use of such a localization service
  - In case of a cellular (mobile) network, localization is centralized, and at cell level
  - It cannot be applied in ad hoc systems

Localization services

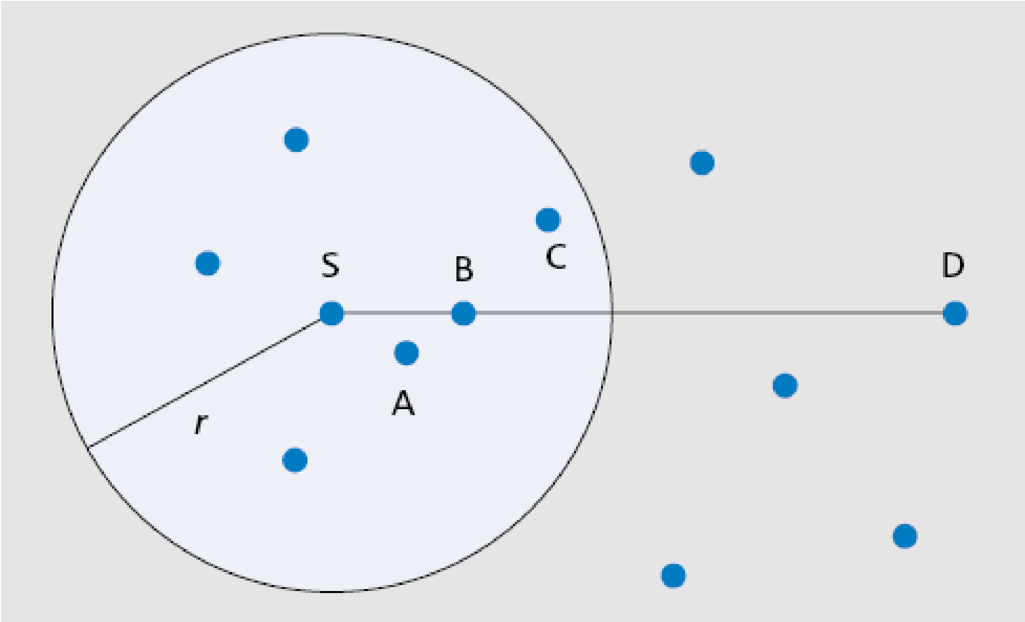
- **Quorum-based** localization
  - *„quorum” = a minimum set of members that are needed by all means to achieve a given goal*
- *„some-to-some”* principle:
  - A reduced set of nodes have a **localization database**
  - These nodes form a **virtual backbone** (inside the backbone topology-based routing!)
  - Each node sends its position information to the closest backbone node
  - Each node asks the position of any destination node from the closest backbone node
  - Updates position information is sent only to a reduced set of backbone nodes (quorum)

Forwarding strategies

- The forwarding decision of an intermediate node:
  - Based on the position information of the destination, embedded in the packet
  - Based on the position of one-hop neighbors
- Positions of the neighbors: known from periodic Hello messages
- Forwarding strategies:
  - Greedy forwarding
    - E.g. MFR, NFP, compass routing
  - Restricted directional flooding
    - E.g., LAR, DREAM
  - Hierarchic solutions

Greedy forwarding

- Which strategy should be used to select the next hop?
- **Most forward within *r* (MFR)**
  - Choose the node that is closest to the destination **D** (Node **C** in the figure)
  - The number of hops is minimized
  - Good strategy if the radio power cannot be changed
- **Nearest with forward progress (NFP)** (node **A**)
  - If the radio power can be adapted
  - Decreases the probability of collisions
- **Compass routing** (node **B**)
  - The smallest angle compared to the SD line
- **Random** choice of a neighbor in the good direction
  - No precise position information needed about neighbors
  - Lower overhead



Greedy forwarding

- Problems:
  - **S** might be closer to the destination **D** than any other node
  - Forwarding might arrive to a local maximum, where there is no way forward
  - **Recovery** mode:
    - If the greedy forwarding stops, we switch to recovery mode
    - If a neighbor can be found again, we switch back to the greedy mode