Politecnico di Milano
Facoltà di Ingegneria dell'Informazione

6 – Ad hoc networks

Reti Mobili Distribuite

Prof. Antonio Capone

Acknowlegments

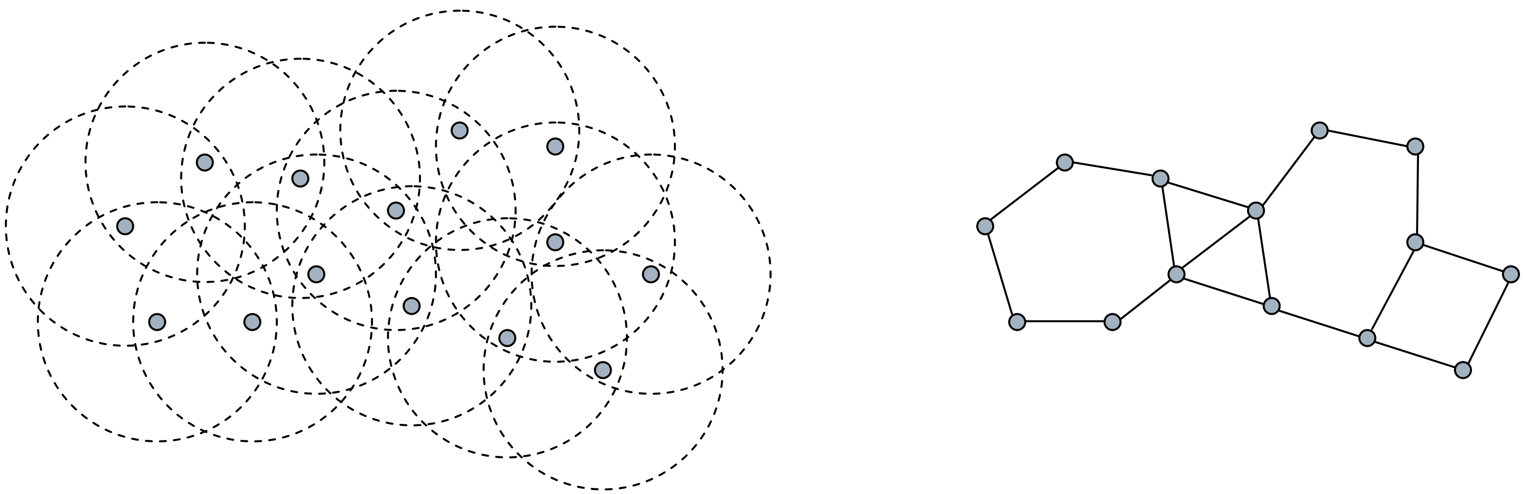
- This class notes are mostly based on the teaching material of:
  - Prof. Eylem Ekici (Ohio State University at Columbus)
  - Prof. Nitin H. Vaidya (University of Illinois at Urbana-Champaign)

Introduction

- Mobile Ad Hoc Networks (MANET):
  - Networks of potentially mobile network nodes
  - Nodes equipped with wireless communication interfaces
  - No pre-established infrastructure
  - Communication between peers involve multiple hops
- Implications
  - Nodes act both as hosts as well as routers
  - Dynamic network topology

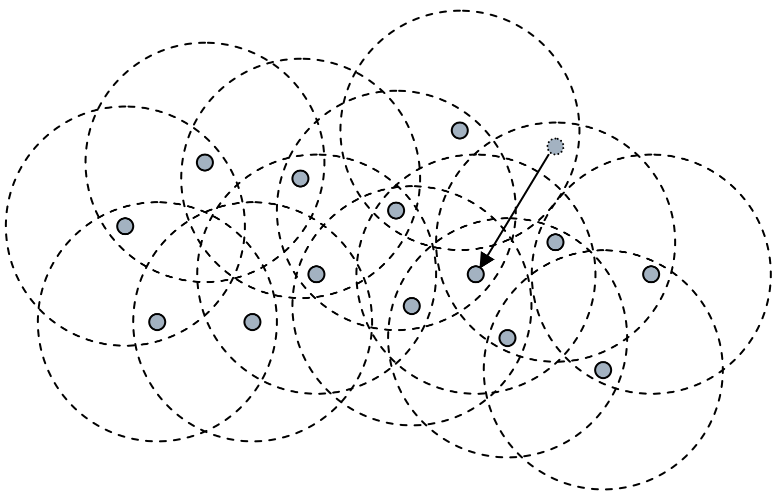
Ad Hoc Network Abstractions

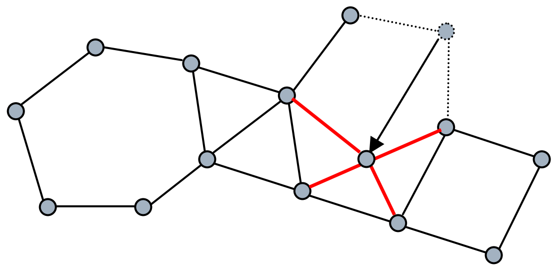
- Every node can communicate directly with a subset of mobile nodes (*neighbors*)
  - Communication "range" of a node varies depending on physical changes
  - Communication range abstracted as circles



Mobile Ad Hoc Networks

- Mobility causes topology changes
  - Topology changes lead to changes in data delivery decisions
  - Introduces real-time adaptation requirements





Mobile Ad Hoc Networks

- Advantages of Mobile Ad Hoc Networks
  - Rapid deployment
    - Particularly important for emergency response and security applications
  - Infrastructure independence
    - No infrastructure needed to kick-start deployment
    - Attractive for disaster recovery (remember Katrina)
  - Flexibility
    - Addition, removal, and relocation of nodes automatically handled
    - Enables new applications where number of participants is dynamic and unpredictable

Example Applications

- Disaster recovery, emergency, security applications
  - Law enforcement
  - Natural and man-made disaster recovery
- Civilian applications
  - Conference room networks
  - Networking in large vessels
  - Personal area networks
  - Vehicular networks
- Military applications
  - Ground-based battlefield networks
  - Hybrid platform networks (land, air, and sea based)

MANET Properties

- Homogeneous MANETs:
  - All nodes carry same properties
    - Communication equipment and “range”
    - Processing capabilities, memory, energy supplies
  - All nodes have identical functionalities
    - All nodes are hosts and routers
    - Leads to flat organization of the network
- Heterogeneous MANETs:
  - Nodes have different hardware
    - Communication equipment and “range”
    - Variation of node resources
    - Leads to inherent hierarchical organization
  - Nodes with diverse functions
    - Host vs. router, cluster member vs. cluster head

Node Mobility Properties

- Node mobility descriptors
  - Speed, Direction, Movement patterns
- Movement of groups of nodes
  - Highly uncorrelated movements
    - Exhibition halls, festival grounds
  - Highly correlated movements
    - Commuters on trains, truck convoys
  - Coordinated movements
    - Movement of military units
  - Hybrid mobility
    - Movement of personal area networks

Data Traffic Properties

- Data traffic is generally application-dependent
  - Bandwidth requirements
  - Timeliness constraints
  - Reliability constraints
  - Security constraints
- Effects on delivery methods
  - Point-to-point vs. point-to-multi-point
  - Pure MANET vs. access to infrastructure
- Addressing requirements
  - Host-based, content-based, other...

Problems to Address

- Physical layer
  - Range, symmetry, power control...
- MAC layer
  - Hidden terminal problem, asymmetrical links, error control, energy efficiency, fairness
- Network layer
  - Point-to-point, point-to-multi-point, flat, hierarchical, proactive, reactive, hybrid, mobility-tailored
- Transport layer
  - Packet loss discrimination, intermediate buffering

Ultimate Goal

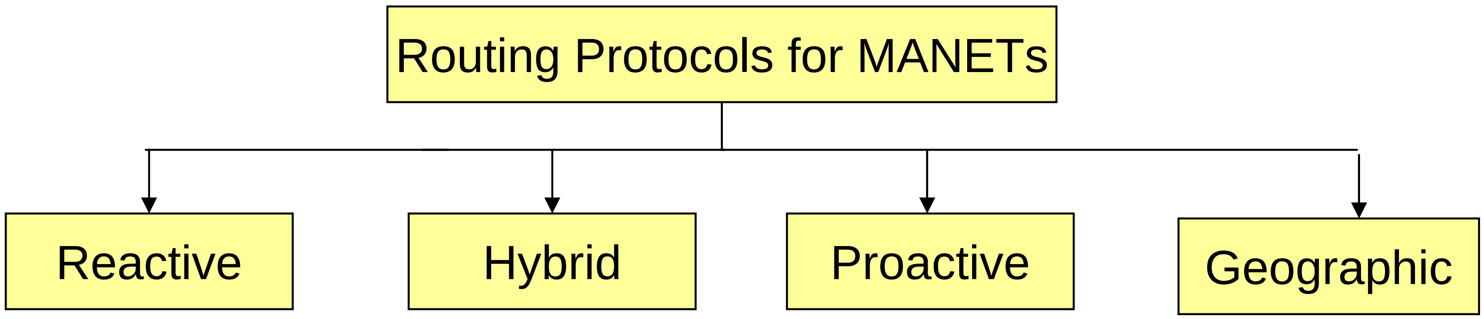
- Develop solutions that can
  - Be used in all ad hoc networks
  - Satisfy various application-level constraints
  - Adapt to changing topological properties
  - Integrate various types of nodes into MANET
  - Interact with fixed infrastructures
- This goal has not been reached so far

Routing in
Mobile Ad Hoc Networks

Introduction

- Routing in ad hoc networks should account for host mobility, which leads to dynamic topologies
- Routing protocols designed for static (or slowly changing) networks
  - May not keep up with the rate of change
  - Waste limited resources
  - May not cater to specific performance criteria such as energy consumption
- As usual, no single protocol is optimal for all ad hoc network types and conditions

Protocol Classification



- Reactive Protocols
  - Determine the paths on-demand
- Proactive Protocols
  - Maintain paths regardless of traffic conditions
- Hybrid Protocols
  - Generally maintain local paths proactively, and create large scale paths reactively
- Geographic Protocols
  - Based on geographical location of nodes

Protocol Classification

- Reactive Protocols
  - Generally involve large delays between the request and first packet delivery
  - Incur low overhead in low traffic scenarios
- Proactive Protocols
  - Packets are immediately delivered as paths are already established
  - Results in high path maintenance overhead since the paths are kept regardless of traffic patterns
- Hybrid Protocols
  - Operate midway of delay and overhead performance

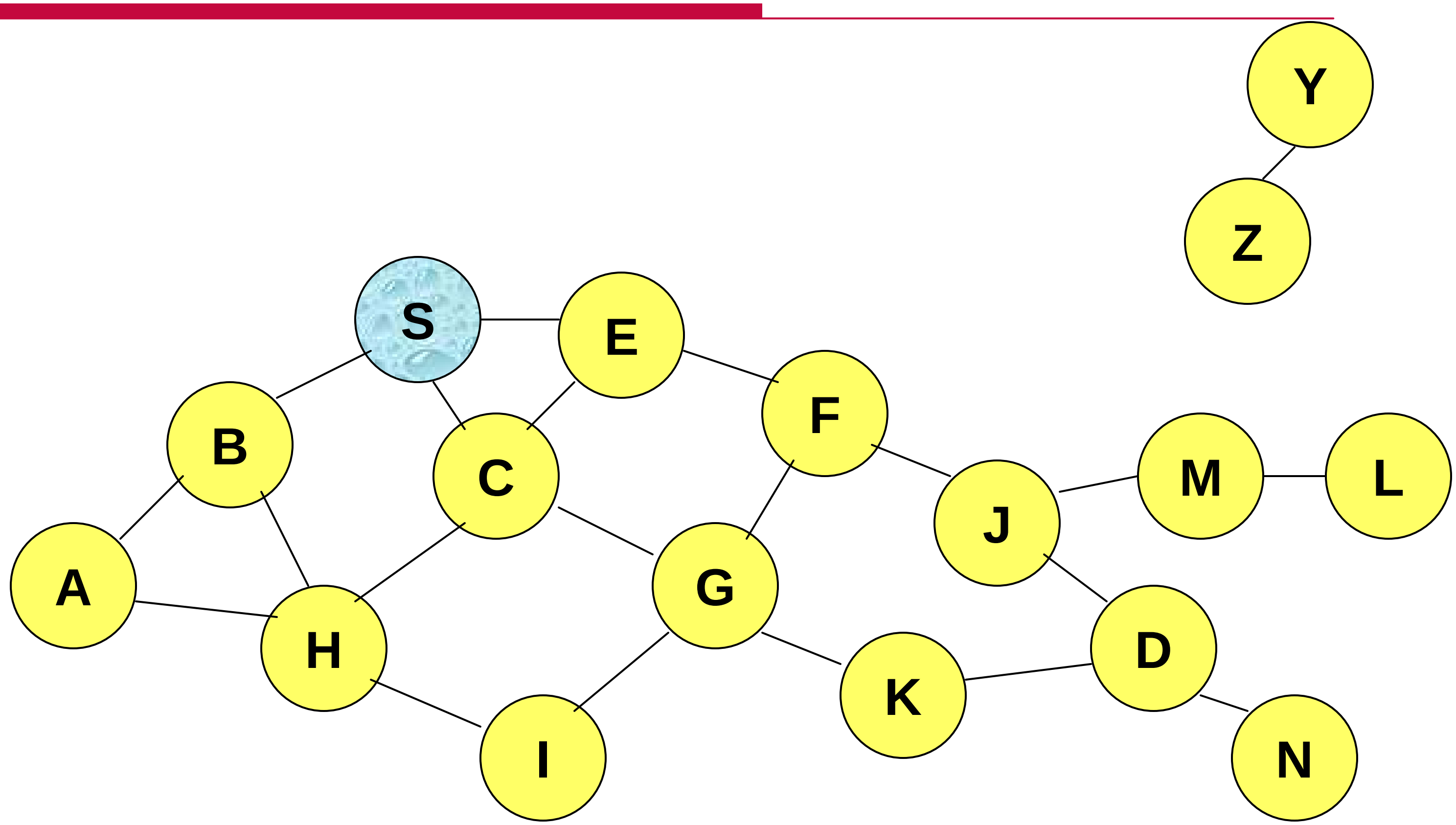
Trade-Off

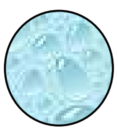
- Latency of route discovery
  - Proactive protocols may have lower latency since routes are maintained at all times
  - Reactive protocols may have higher latency because a route from X to Y will be found only when X attempts to send to Y
- Overhead of route discovery/maintenance
  - Reactive protocols may have lower overhead since routes are determined only if needed
  - Proactive protocols can (but not necessarily) result in higher overhead due to continuous route updating
- Which approach achieves a better trade-off depends on the traffic and mobility patterns

Flooding for Data Delivery

- Sender S broadcasts data packet P to all its neighbors
- Each node receiving P forwards P to its neighbors
- Sequence numbers used 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 Data Delivery

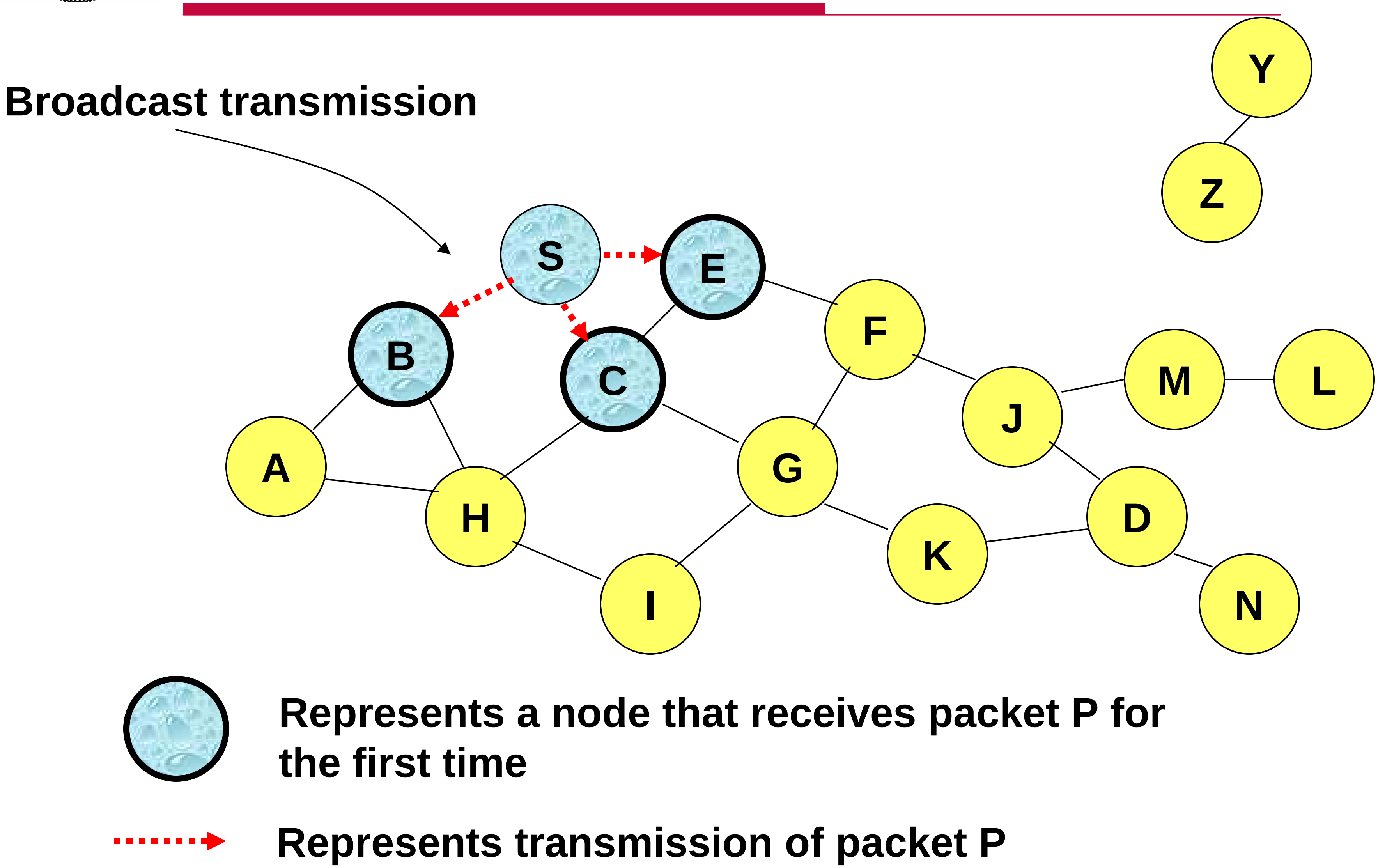




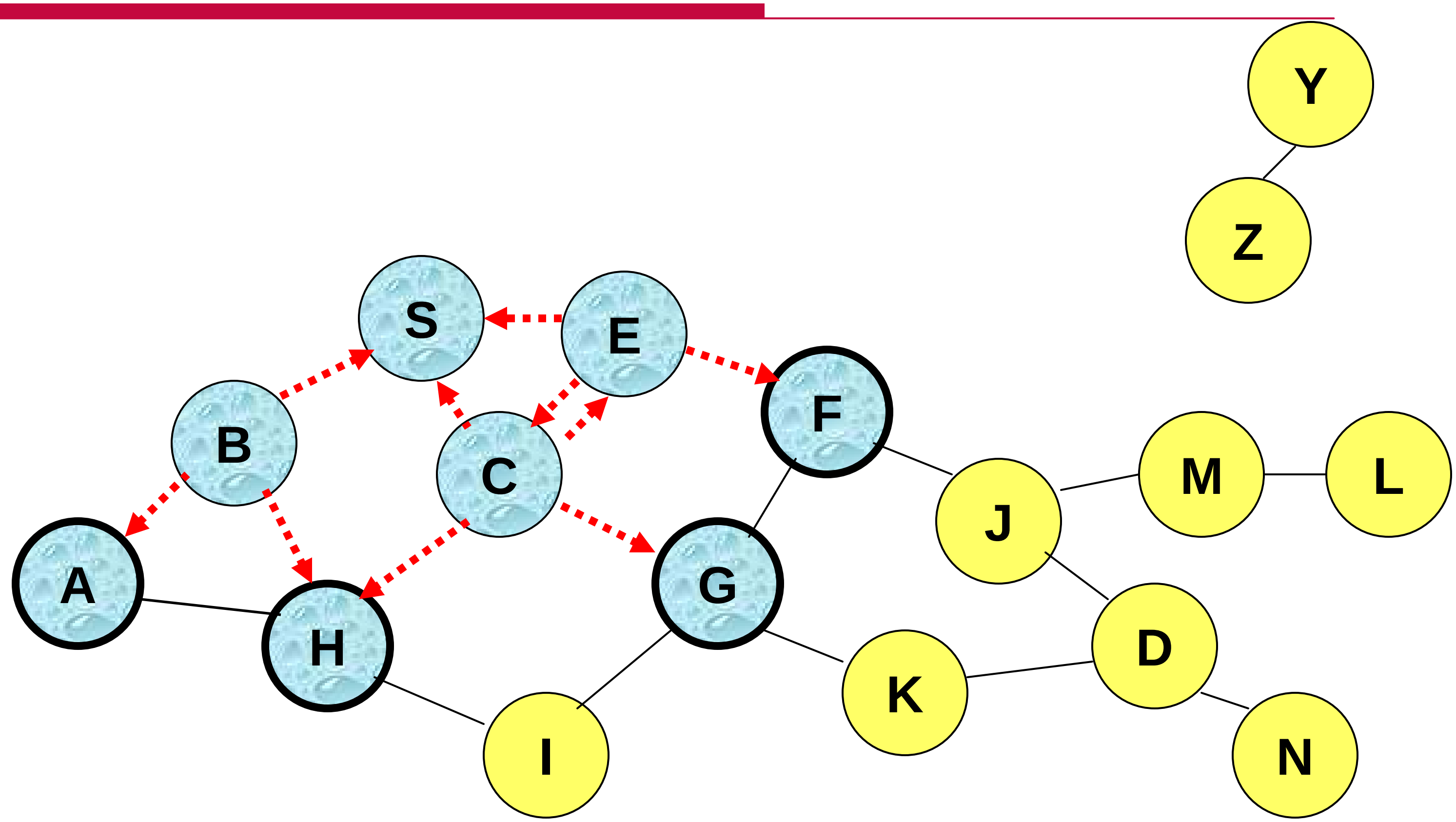
Represents a node that has received packet P

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

Flooding for Data Delivery

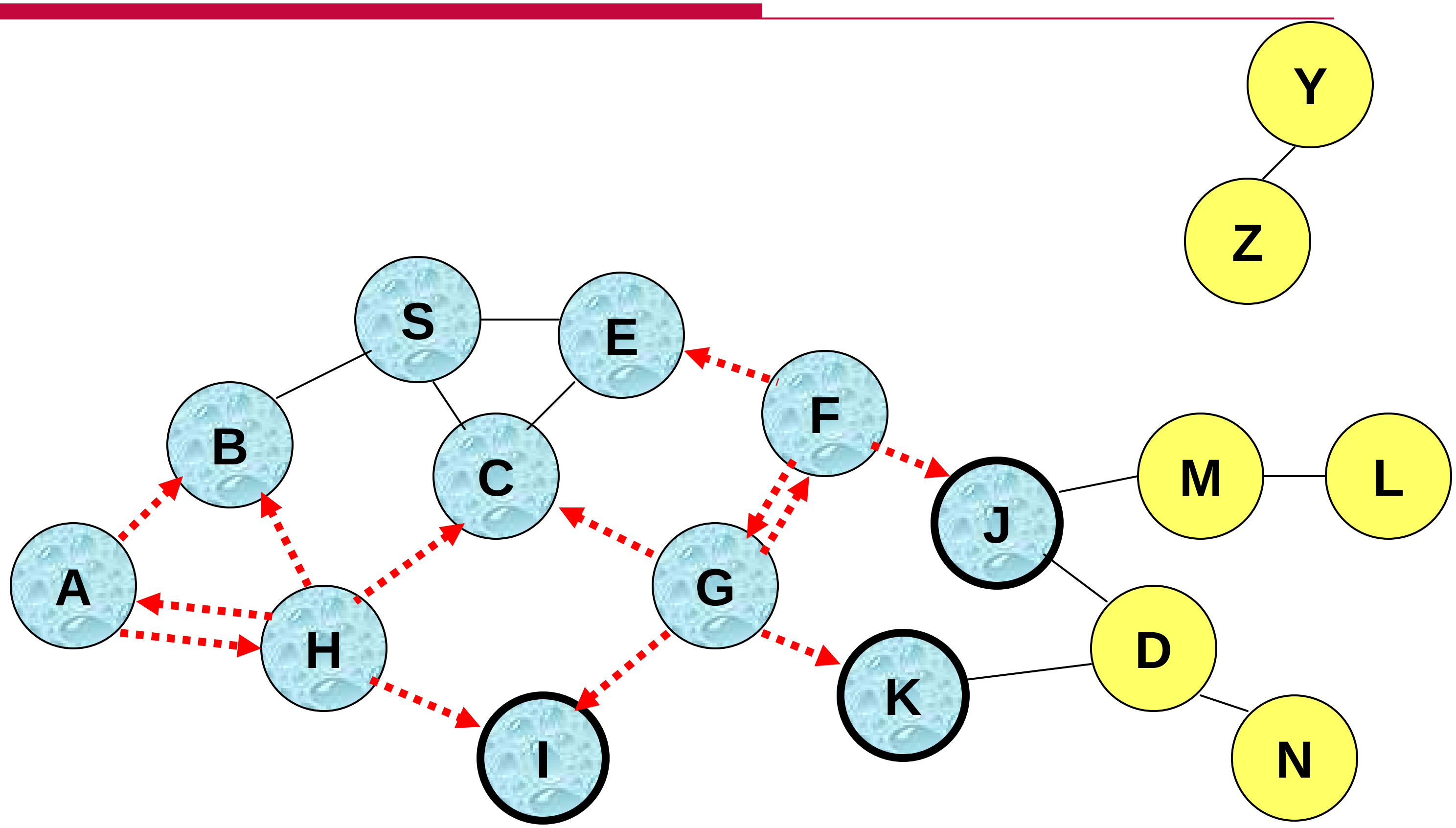


Flooding for Data Delivery



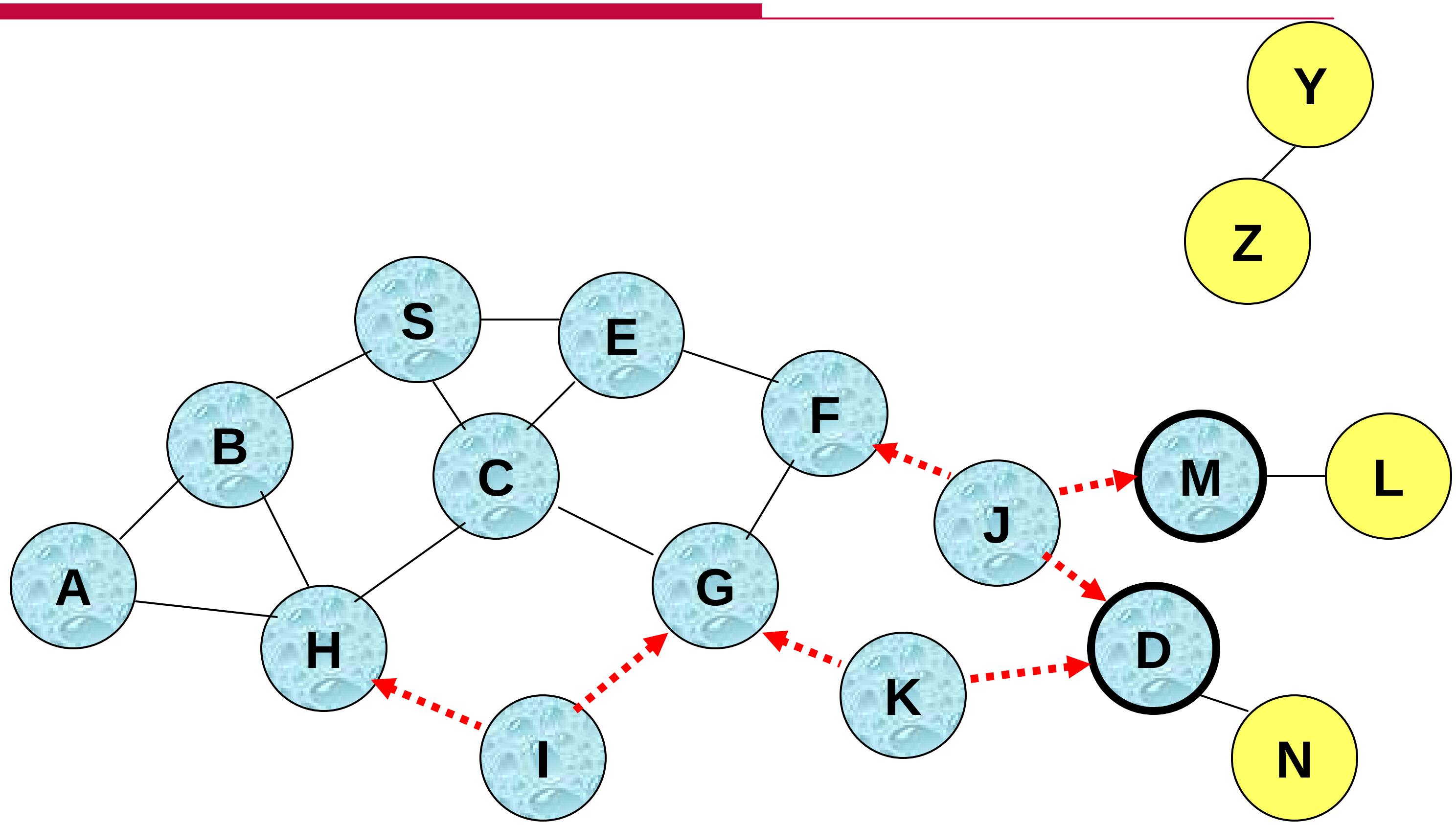
- **Node H receives packet P from two neighbors: potential for collision**

Flooding for Data Delivery



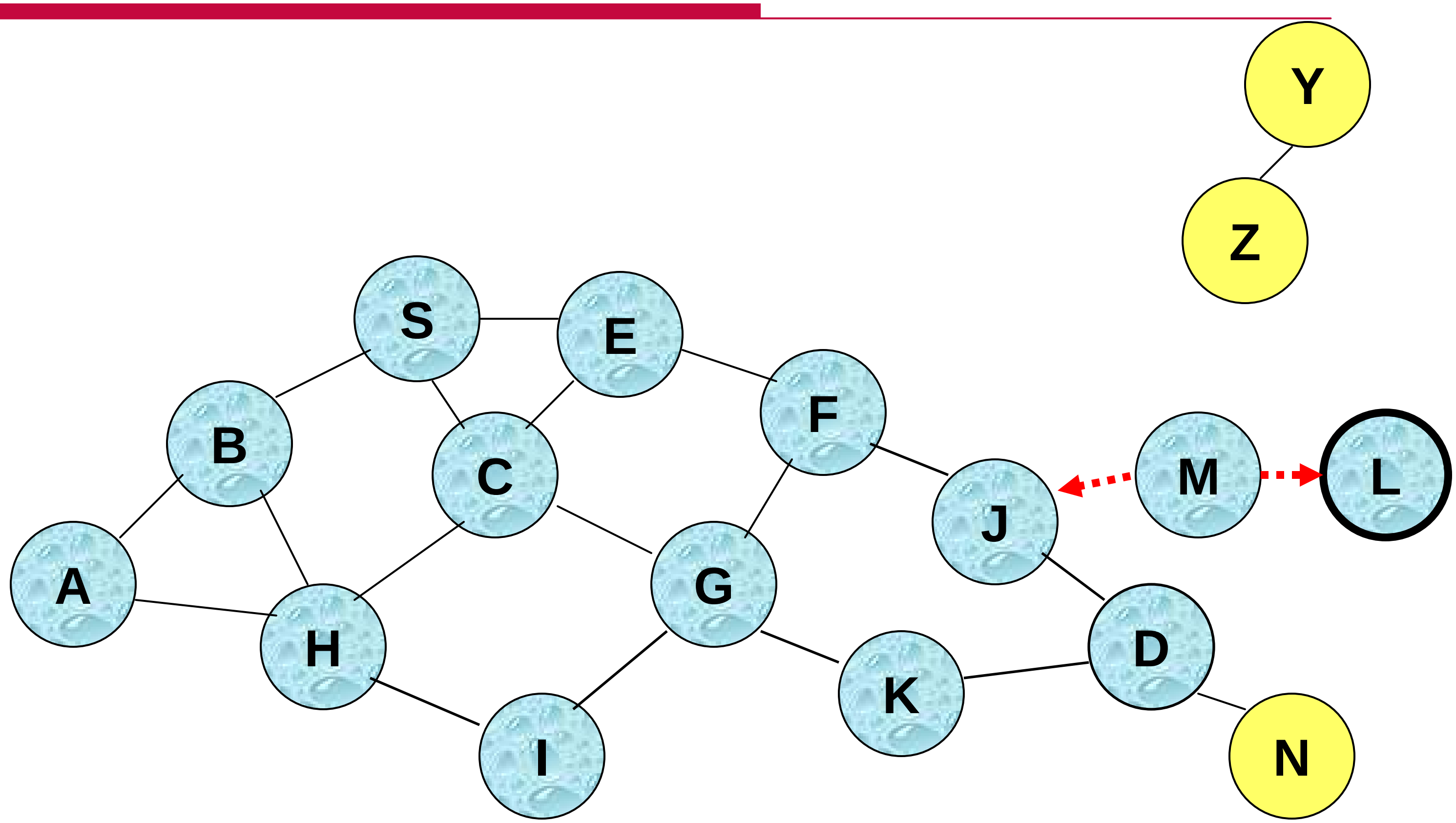
- Node C receives packet P from G and H, but does not forward it again, because node C has already forwarded packet P once

Flooding for Data Delivery



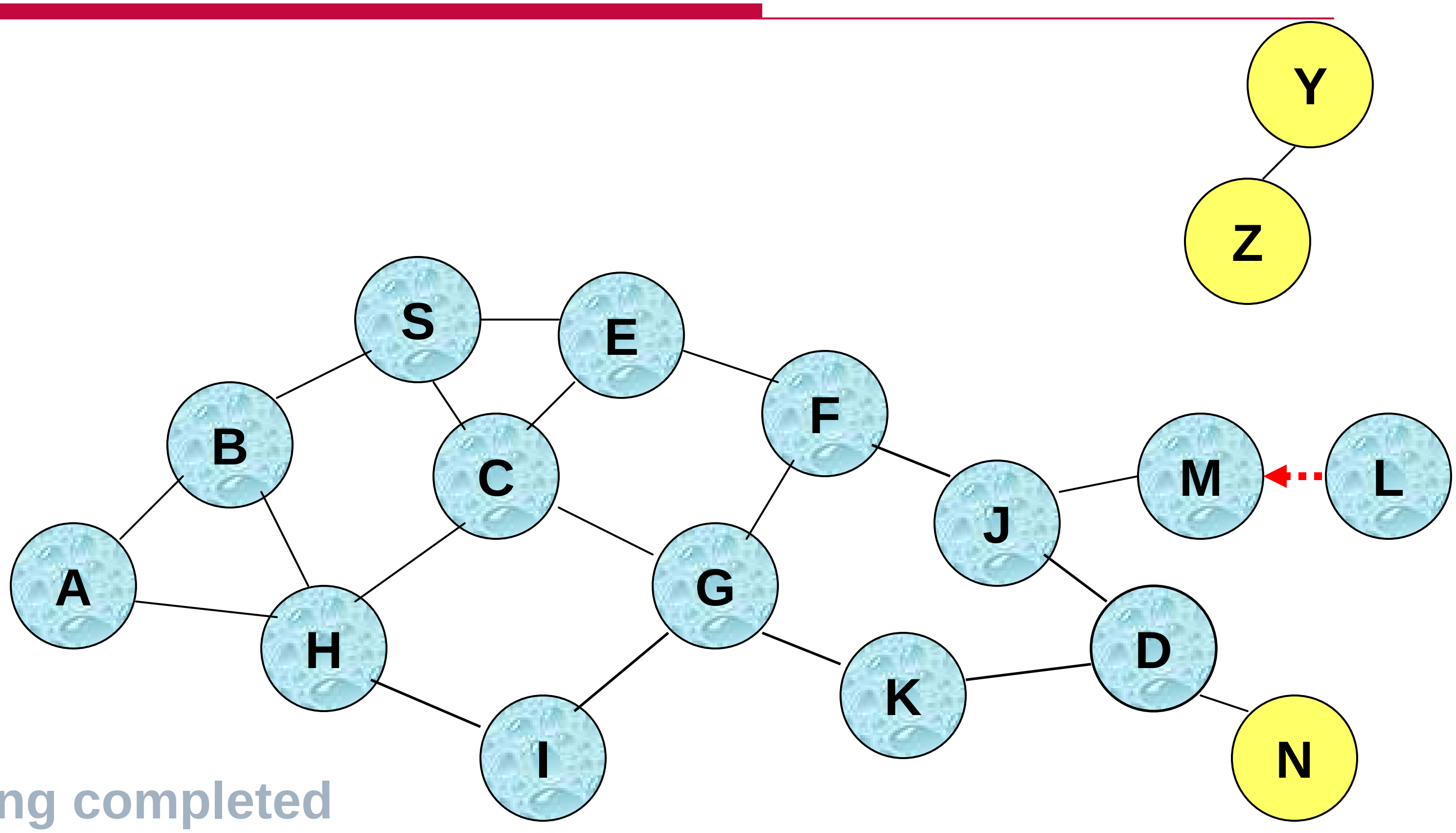
- Nodes J and K both broadcast packet P to node D
- Since nodes J and K are hidden from each other, their transmissions may collide => Packet P may not be delivered to node D at all, despite the use of flooding

Flooding for Data Delivery



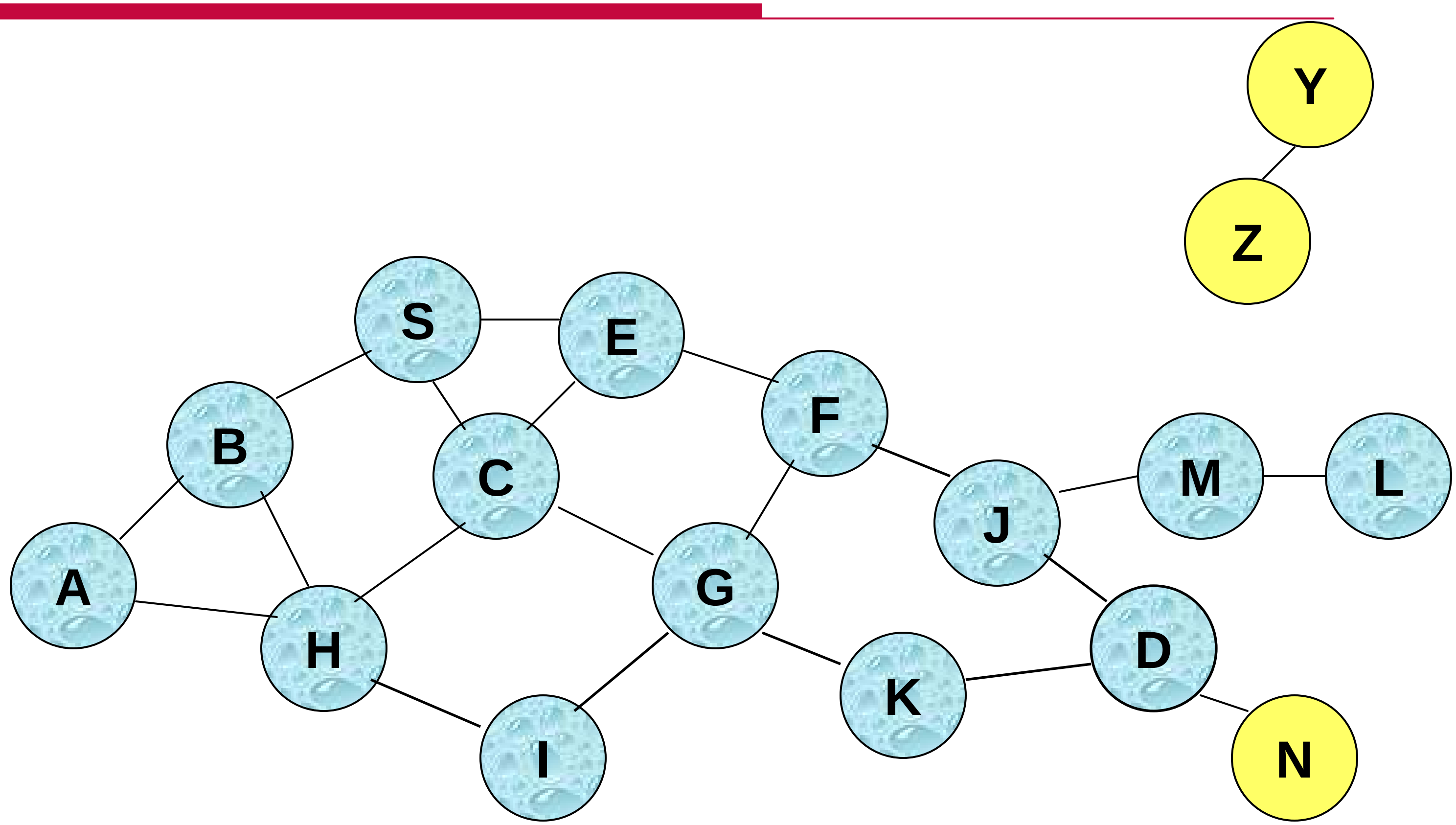
- Node D does not forward packet P, because node D is the intended destination of packet P

Flooding for Data Delivery



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

Flooding for Data Delivery



- Flooding may deliver packets to too many nodes (in the **worst case**, all nodes reachable from sender may receive the packet)

Flooding for Data Delivery: Advantages

- Simplicity
- May be more efficient than other protocols when rate of information transmission is low enough that the overhead of explicit route discovery/maintenance incurred by other protocols is relatively higher
  - this scenario may occur, for instance, when nodes transmit small data packets relatively infrequently, and many topology changes occur between consecutive packet transmissions
- Potentially higher reliability of data delivery
  - Because packets may be delivered to the destination on multiple paths

Flooding for Data Delivery: Disadvantages

- Potentially, very high overhead
  - Data packets may be delivered to too many nodes who do not need to receive them
- Potentially lower reliability of data delivery
  - Flooding uses broadcasting -- hard to implement reliable broadcast delivery without significantly increasing overhead
    - Broadcasting in IEEE 802.11 MAC is unreliable
  - In our example, nodes J and K may transmit to node D simultaneously, resulting in loss of the packet
    - in this case, destination would not receive the packet at all

Flooding of Control Packets

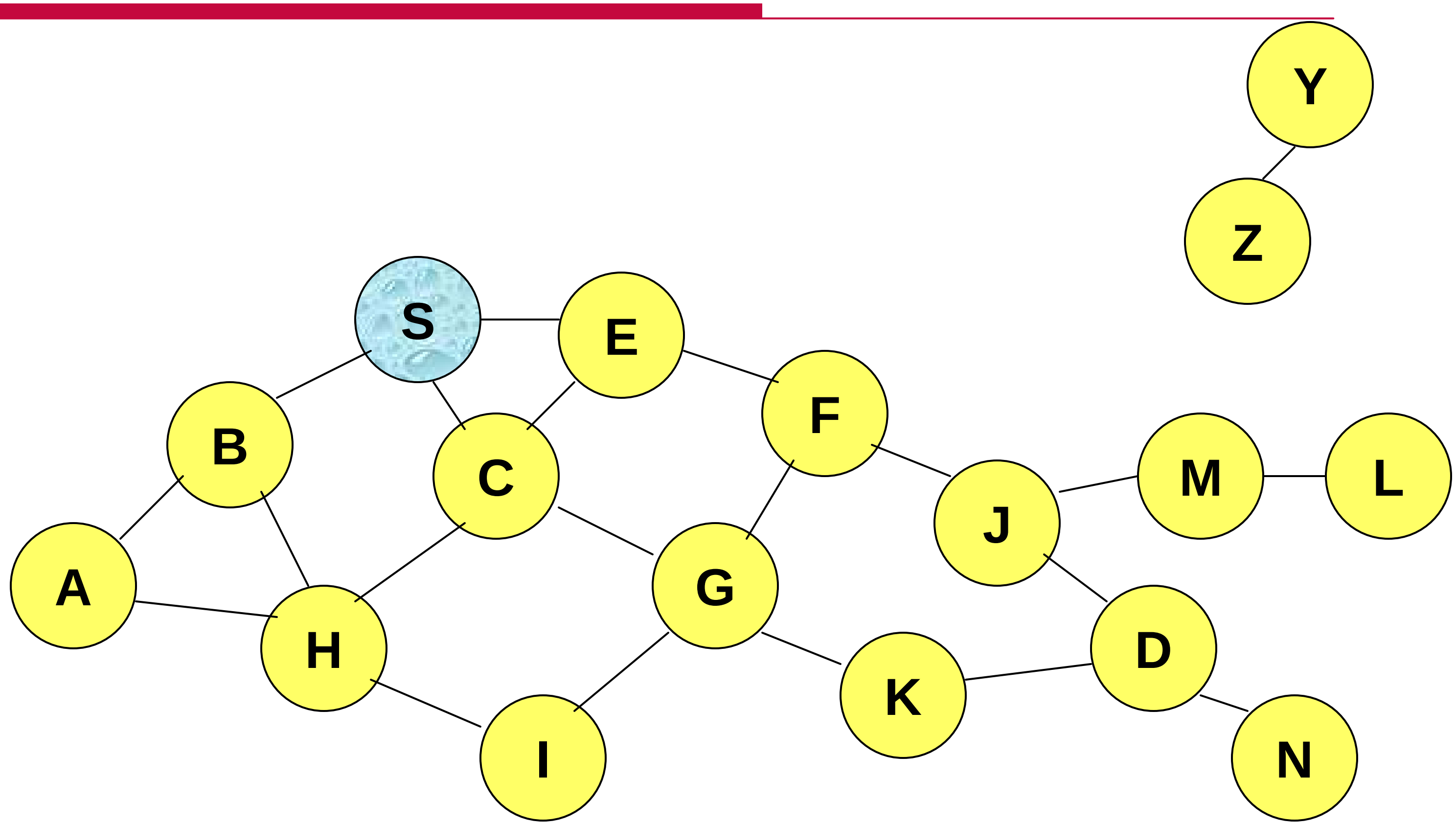
- Many protocols perform (potentially *limited*) flooding of control packets, instead of data packets
- The control packets are used to discover routes
- Discovered routes are subsequently used to send data packet(s)
- Overhead of control packet flooding is amortized over data packets transmitted between consecutive control packet floods

Reactive Protocols

Dynamic Source Routing (DSR)

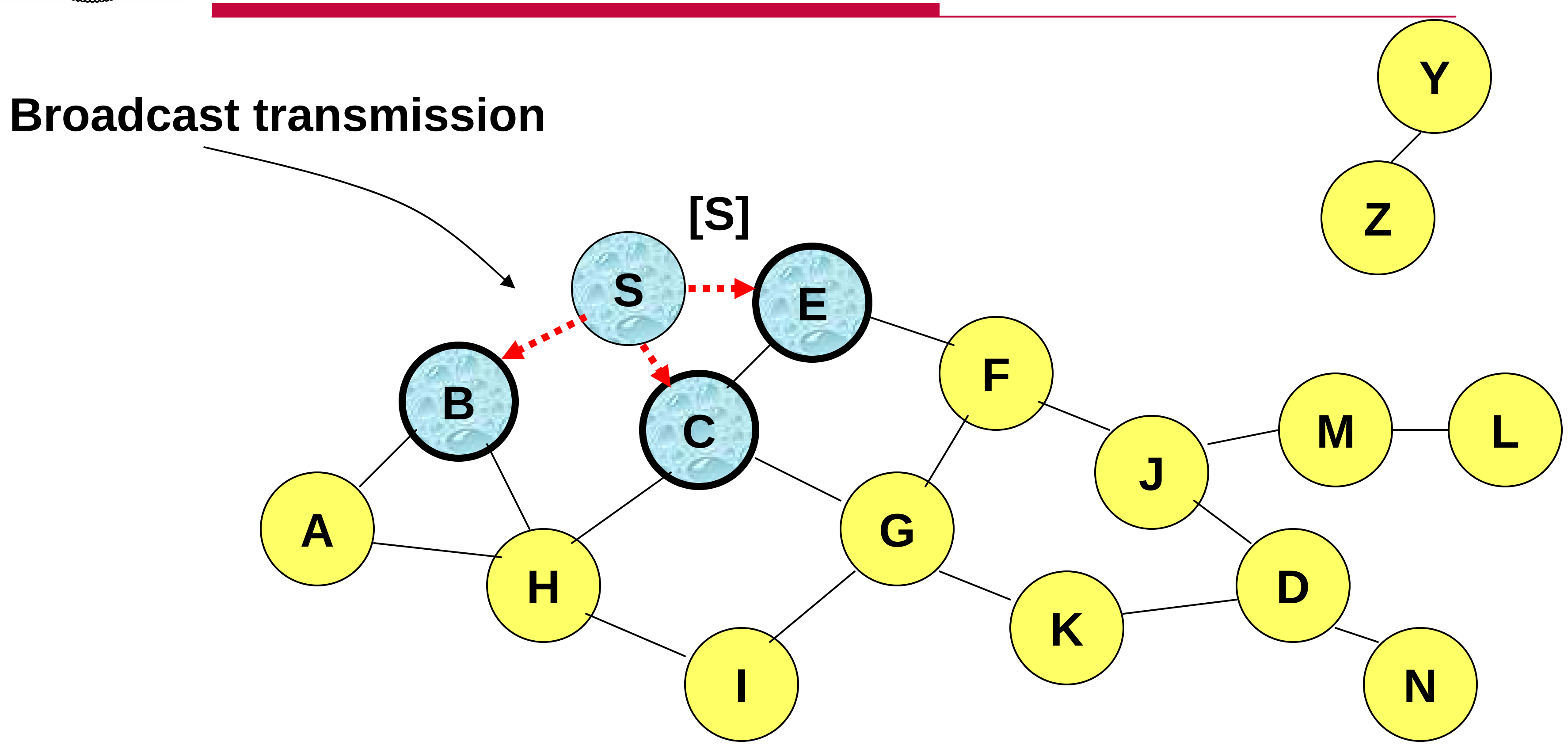
- When node S wants to send a packet to node D, but does not know a route to D, node S initiates a route discovery
- Source node S floods Route Request (RREQ)
- Each node appends own identifier when forwarding RREQ

Route Discovery in DSR



Represents a node that has received RREQ for D from S

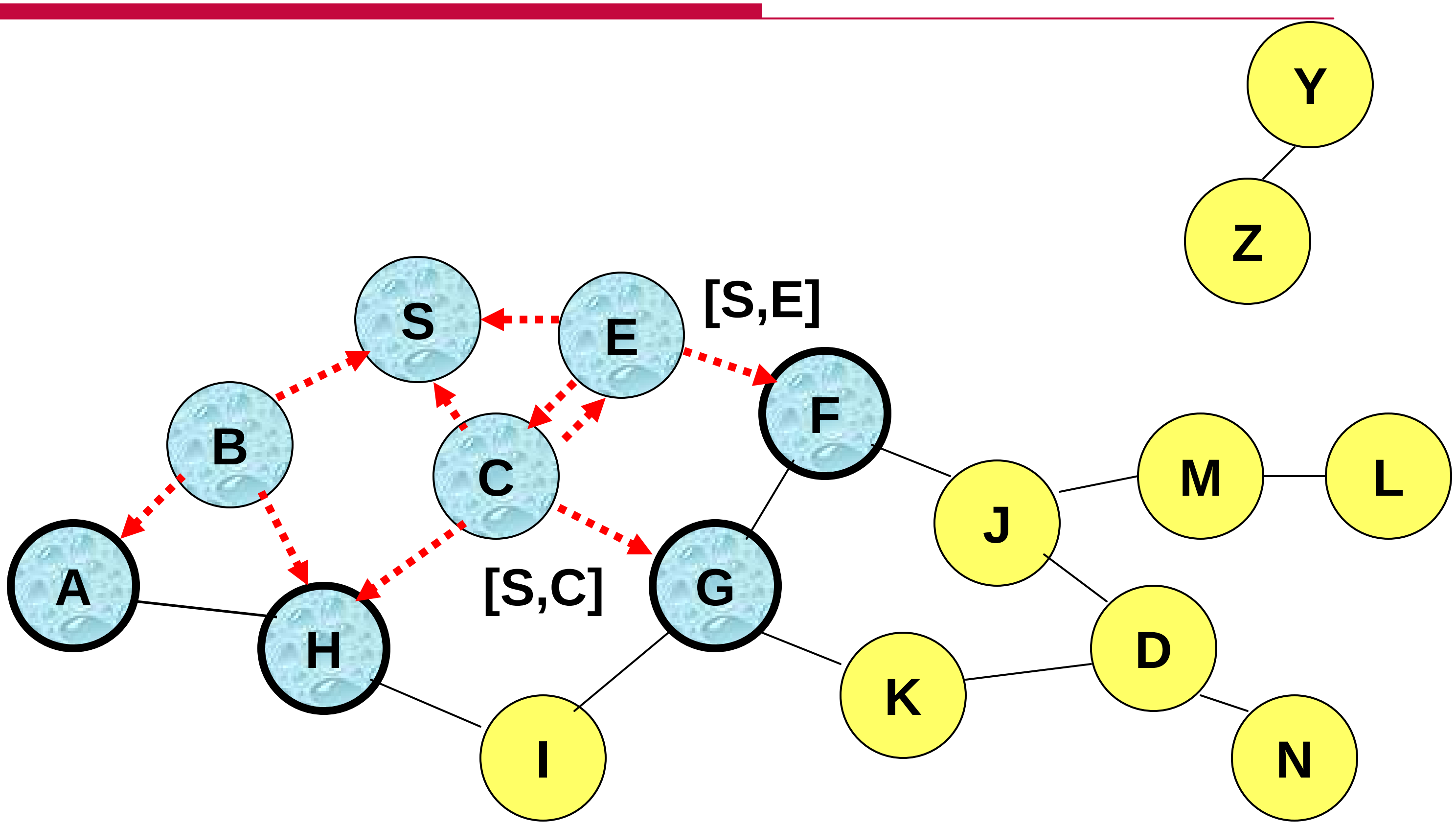
Route Discovery in DSR



Represents transmission of RREQ

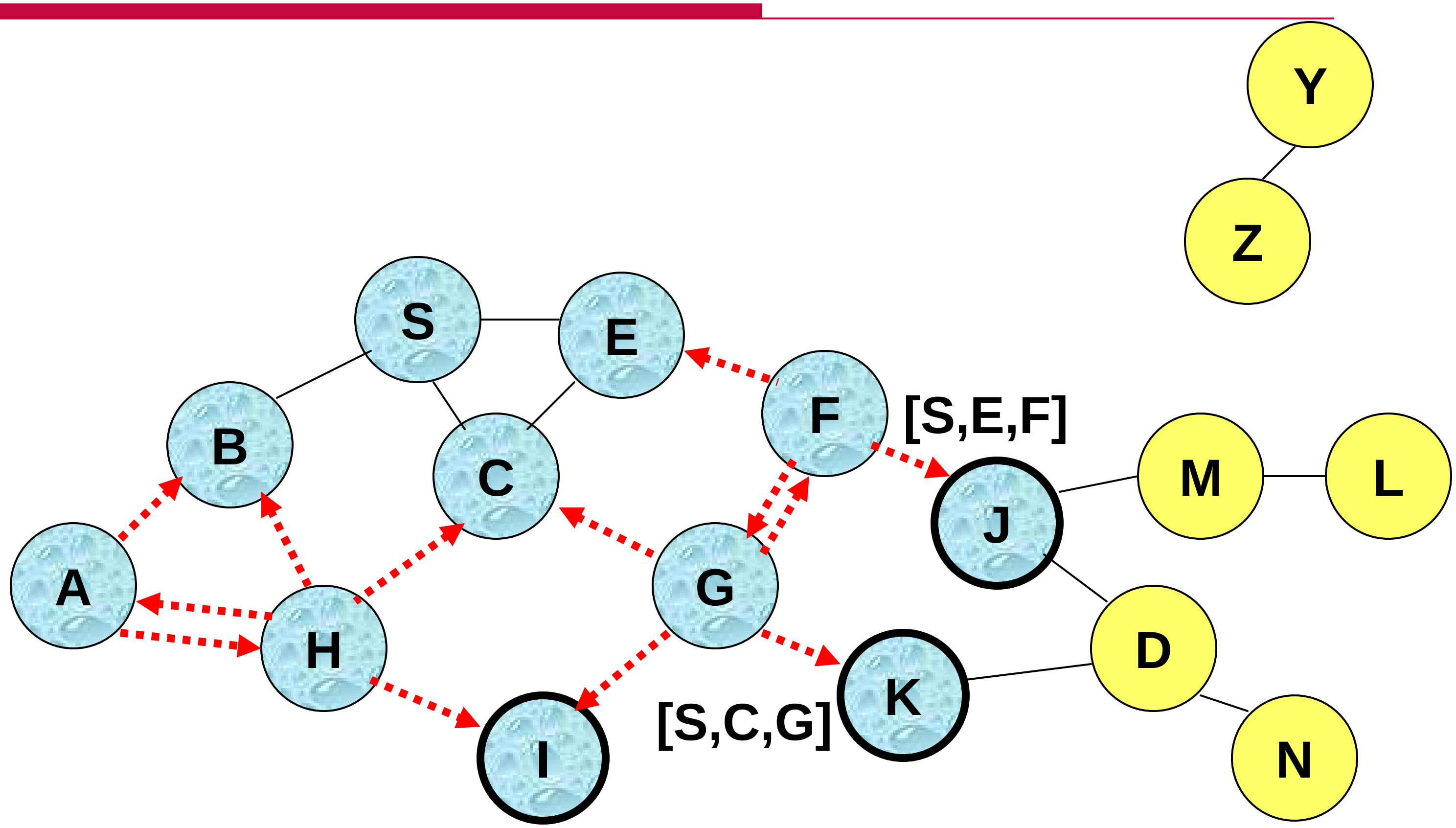
[X,Y] Represents list of identifiers appended to RREQ

Route Discovery in DSR



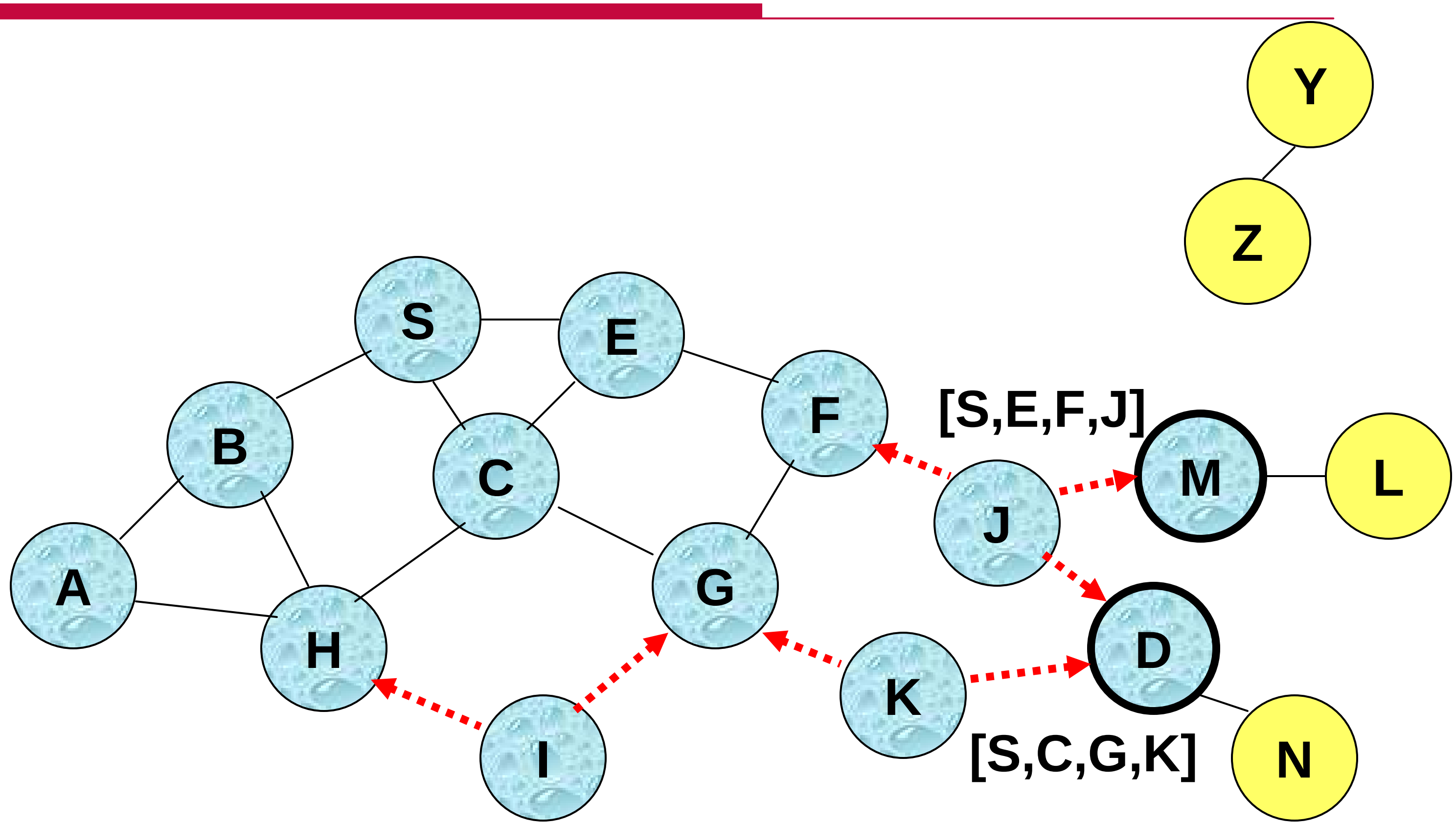
- Node H receives packet RREQ from two neighbors:
  potential for collision

Route Discovery in DSR



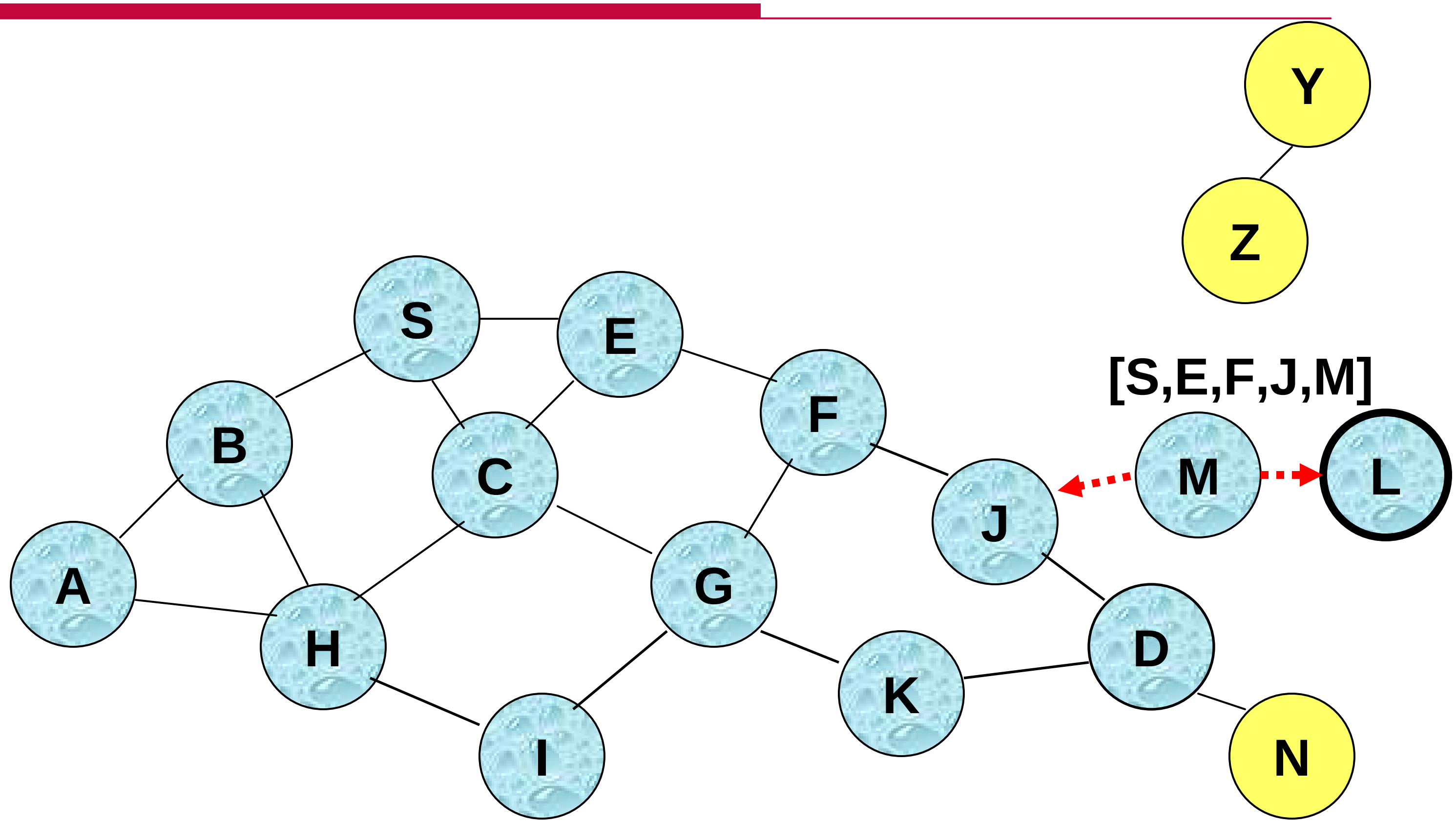
• Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

Route Discovery in DSR



- Nodes J and K both broadcast RREQ to node D
- Since nodes J and K are hidden from each other, their transmissions may collide

Route Discovery in DSR

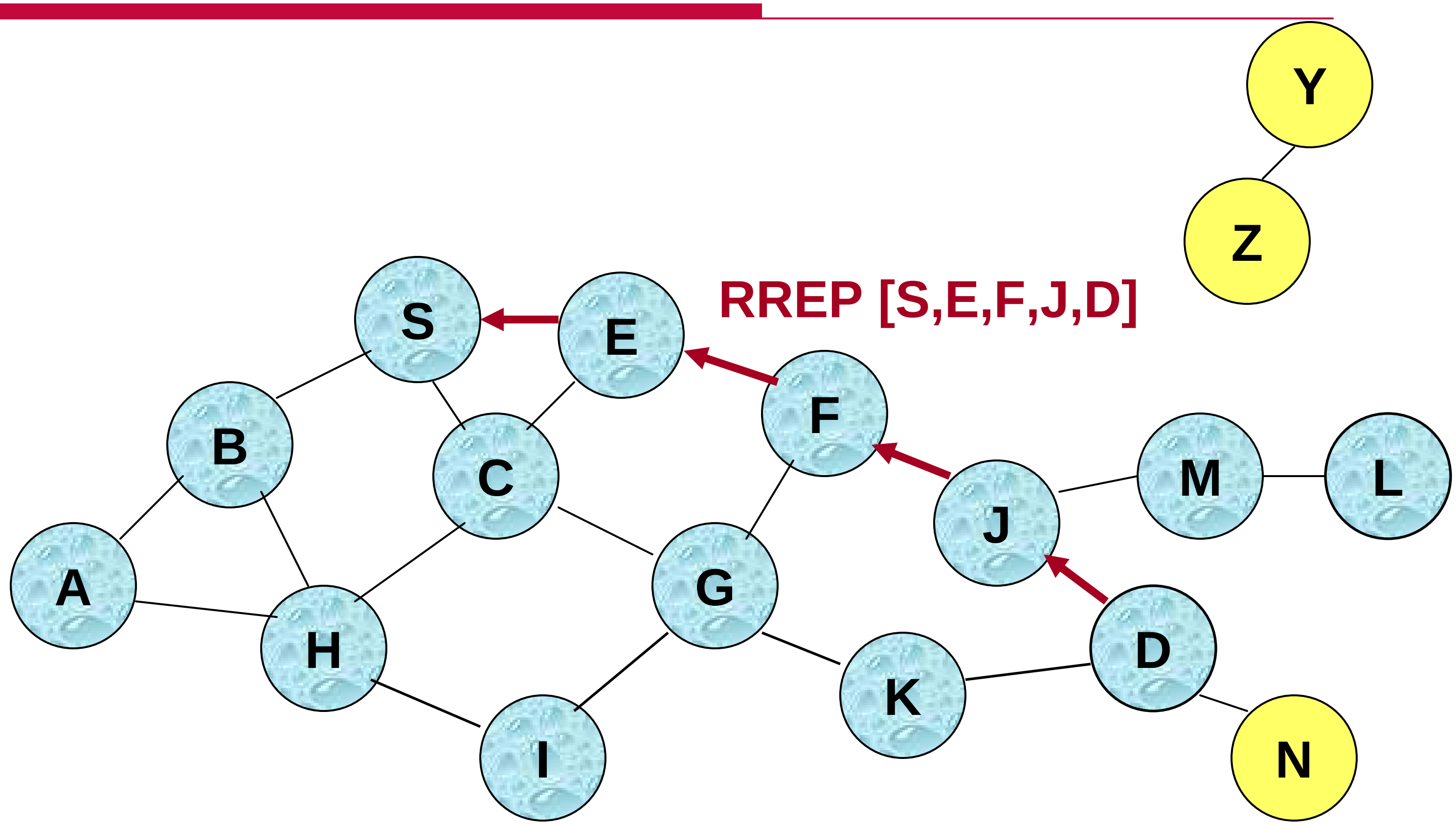


- Node D **does not forward** RREQ, because node D is the **intended target** of the route discovery

Route Discovery in DSR

- Destination D on receiving the first RREQ, sends a Route Reply (RREP)
- RREP is sent on a route obtained by reversing the route appended to received RREQ
- RREP includes the route from S to D on which RREQ was received by node D

Route Reply in DSR



Represents RREP control message

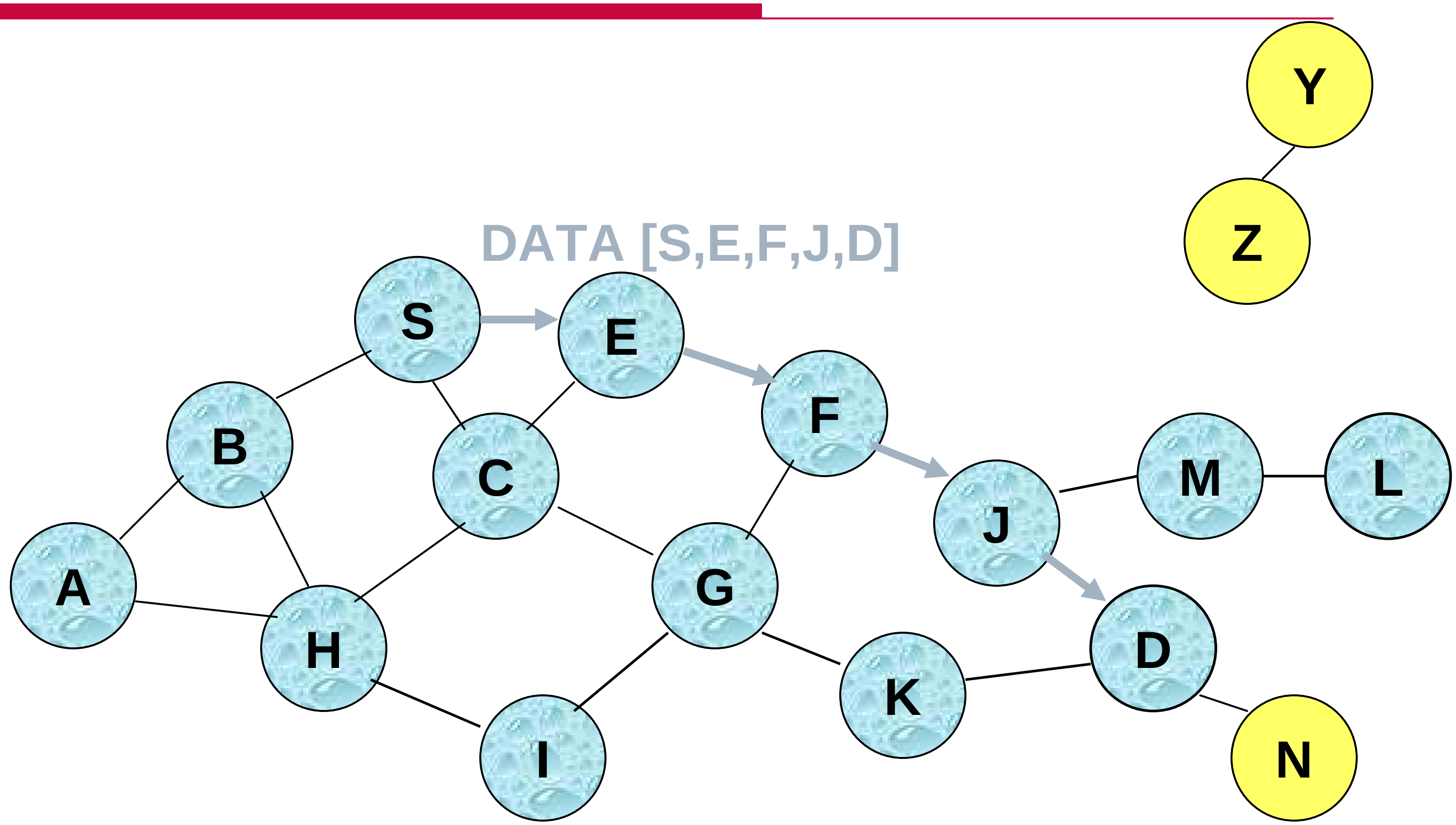
Route Reply in DSR

- Route Reply can be sent by reversing the route in Route Request (RREQ) only if links are guaranteed to be bi-directional
  - To ensure this, RREQ should be forwarded only if it received on a link that is known to be bi-directional
- If unidirectional (asymmetric) links are allowed, then RREP may need a route discovery for S from node D
  - Unless node D already knows a route to node S
  - If a route discovery is initiated by D for a route to S, then the Route Reply is piggybacked on the Route Request from D.
- If IEEE 802.11 MAC is used to send data, then links have to be bi-directional (since Ack is used)

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

When to Perform a Route Discovery

- When node S wants to send data to node D, but does not know a valid route node D

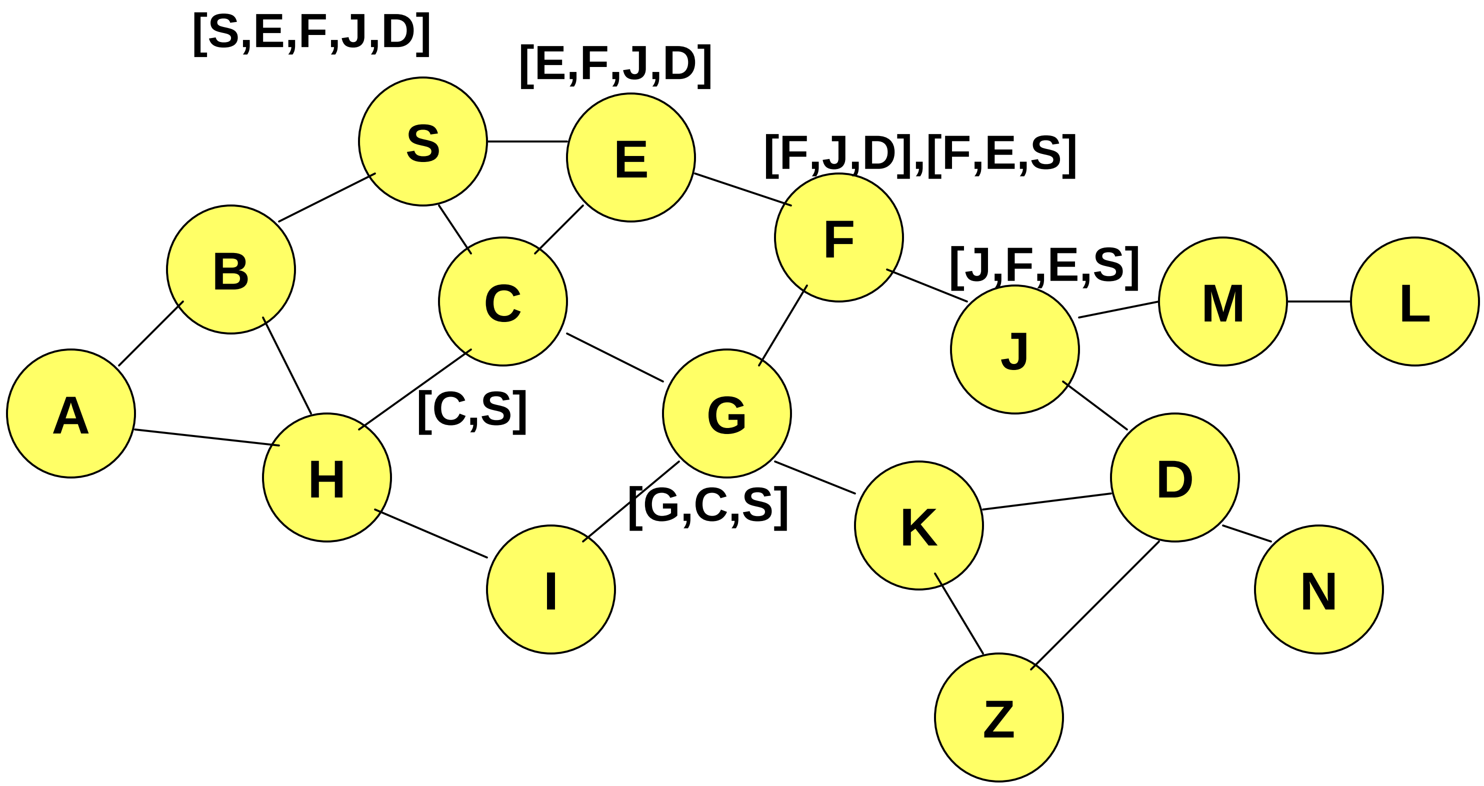
DSR Optimization: Route Caching

- Each node caches a new route it learns by *any means*
- When node S finds route [S,E,F,J,D] to node D, node S also learns route [S,E,F] to node F
- When node K receives Route Request [S,C,G] destined for node, node K learns route [K,G,C,S] to node S
- When node F forwards Route Reply RREP [S,E,F,J,D], node F learns route [F,J,D] to node D
- When node E forwards Data [S,E,F,J,D] it learns route [E,F,J,D] to node D
- A node may also learn a route when it overhears Data packets

Use of Route Caching

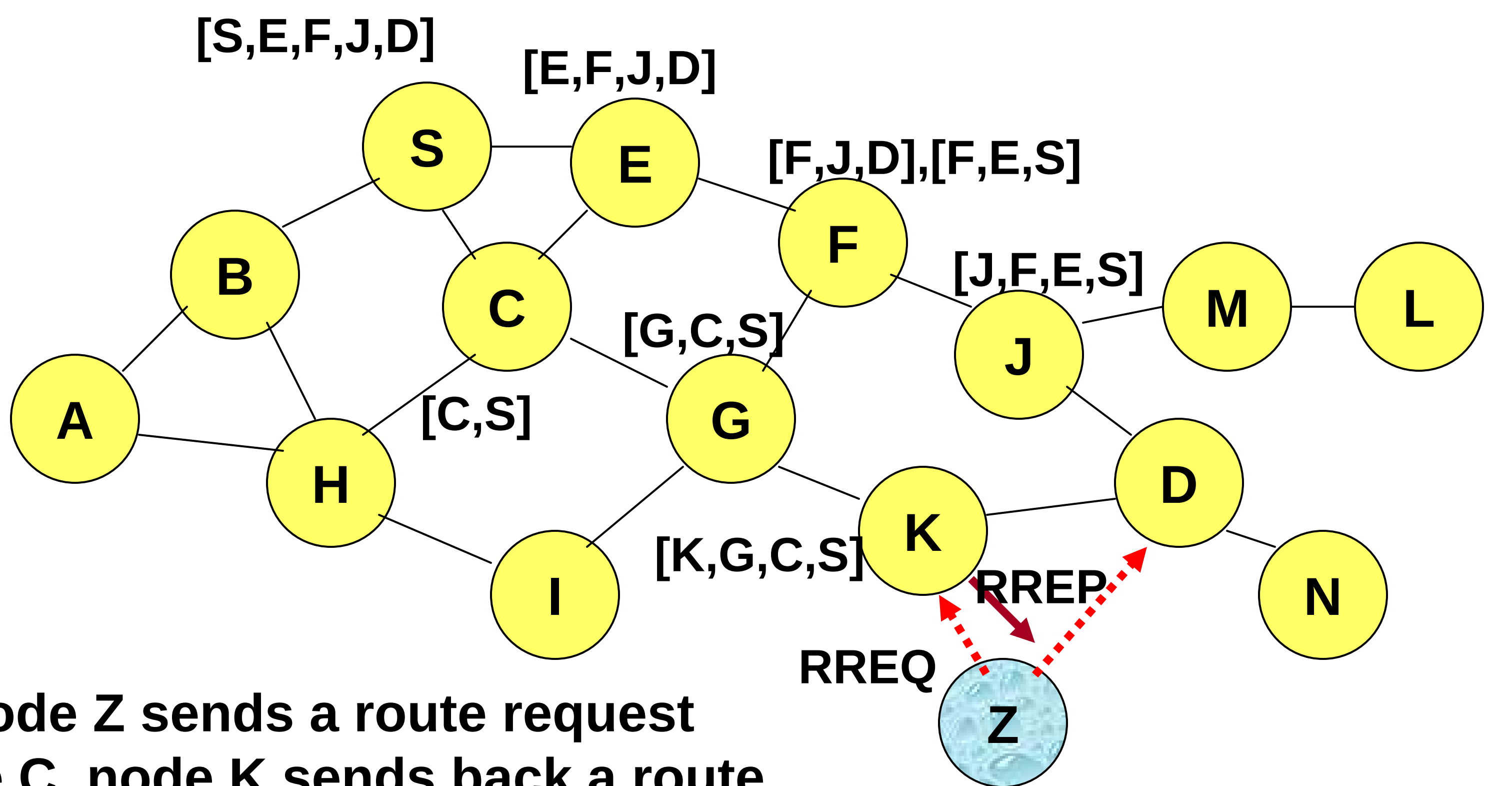
- When node S learns that a route to node D is broken, it uses another route from its local cache, if such a route to D exists in its cache. Otherwise, node S initiates route discovery by sending a route request
- Node X on receiving a Route Request for some node D can send a Route Reply if node X knows a route to node D
- Use of route cache
  - can speed up route discovery
  - can reduce propagation of route requests

Use of Route Caching



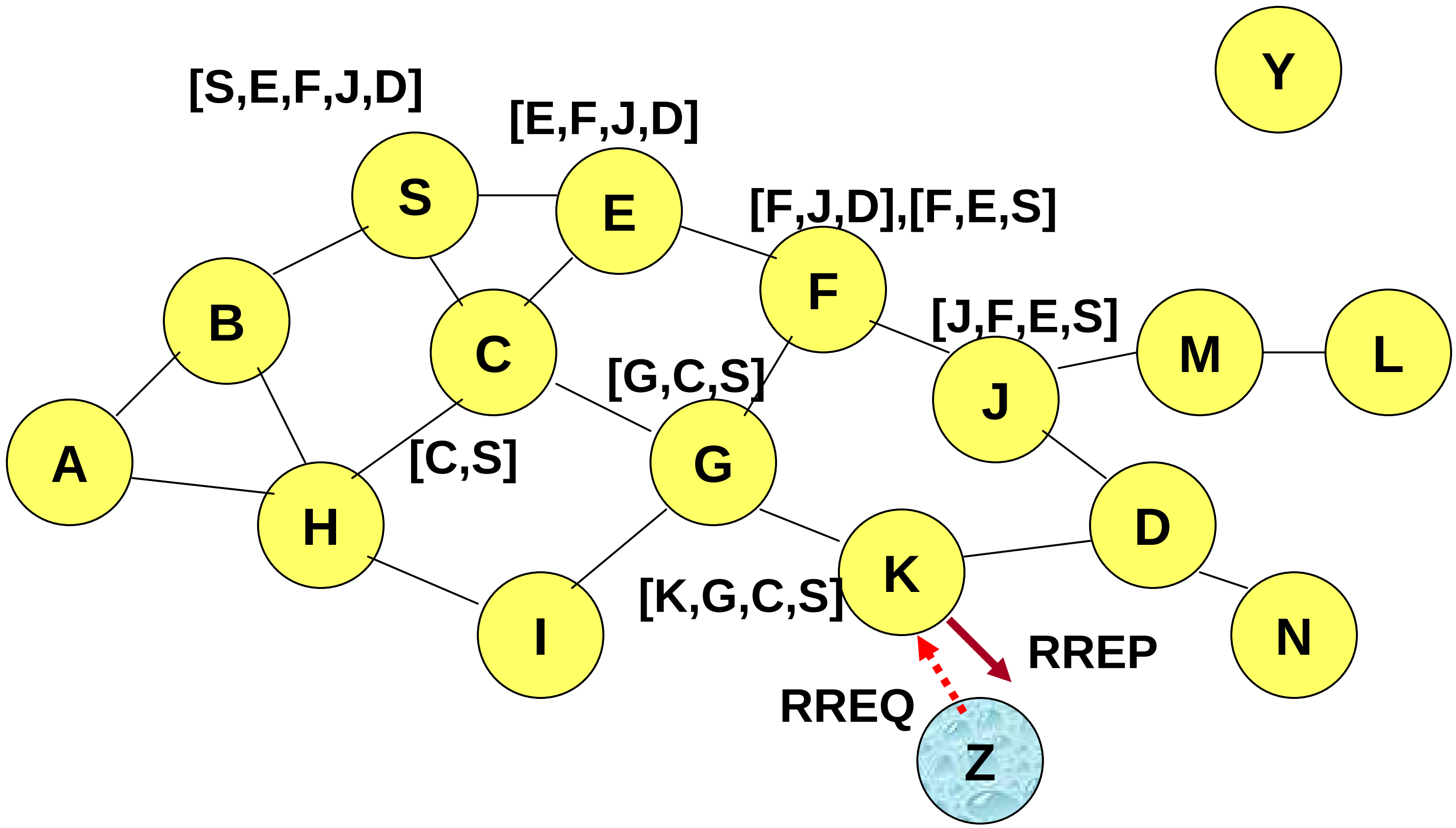
[P,Q,R] Represents cached route at a node
(DSR maintains the cached routes in a tree format)

Route Caching: Can Speed up Route Discovery



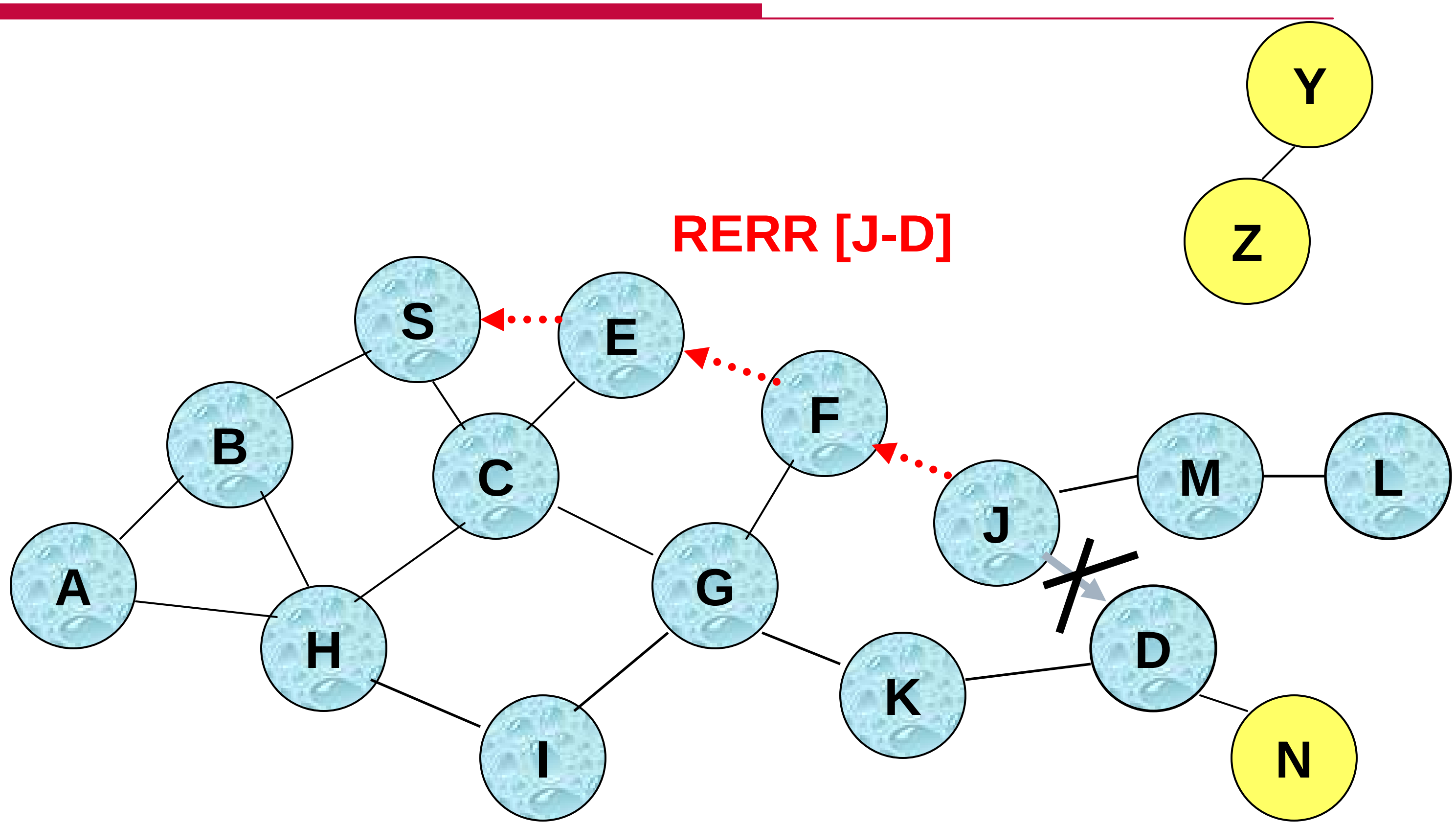
When node Z sends a route request for node C, node K sends back a route reply [Z,K,G,C] to node Z using a locally cached route

Route Caching: Can Reduce Propagation of Route Requests



Assume that there is no link between D and Z.
Route Reply (RREP) from node K limits flooding of RREQ.
In general, the reduction may be less dramatic.

Route Error (RERR)



J sends a route error to S along route J-F-E-S when its attempt to forward the data packet S (with route SEFJD) on J-D fails

Nodes hearing RERR update their route cache to remove link J-D

Route Caching: Beware!

- Stale caches can adversely affect performance
- With passage of time and host mobility, cached routes may become invalid
- A sender host may try several stale routes (obtained from local cache, or replied from cache by other nodes), before finding a good route

Dynamic Source Routing: Advantages

- Routes maintained only between nodes who need to communicate
  - reduces overhead of route maintenance
- Route caching can further reduce route discovery overhead
- A single route discovery may yield many routes to the destination, due to intermediate nodes replying from local caches

Dynamic Source Routing: Disadvantages

- Packet header size grows with route length due to source routing
- Flood of route requests may potentially reach all nodes in the network
- Care must be taken to avoid collisions between route requests propagated by neighboring nodes
  - insertion of random delays before forwarding RREQ
- Increased contention if too many route replies come back due to nodes replying using their local cache
  - Route Reply *Storm* problem
  - Reply storm may be eased by preventing a node from sending RREP if it hears another RREP with a shorter route

Dynamic Source Routing: Disadvantages

- An intermediate node may send Route Reply using a stale cached route, thus polluting other caches
- This problem can be eased if some mechanism to purge (potentially) invalid cached routes is incorporated.

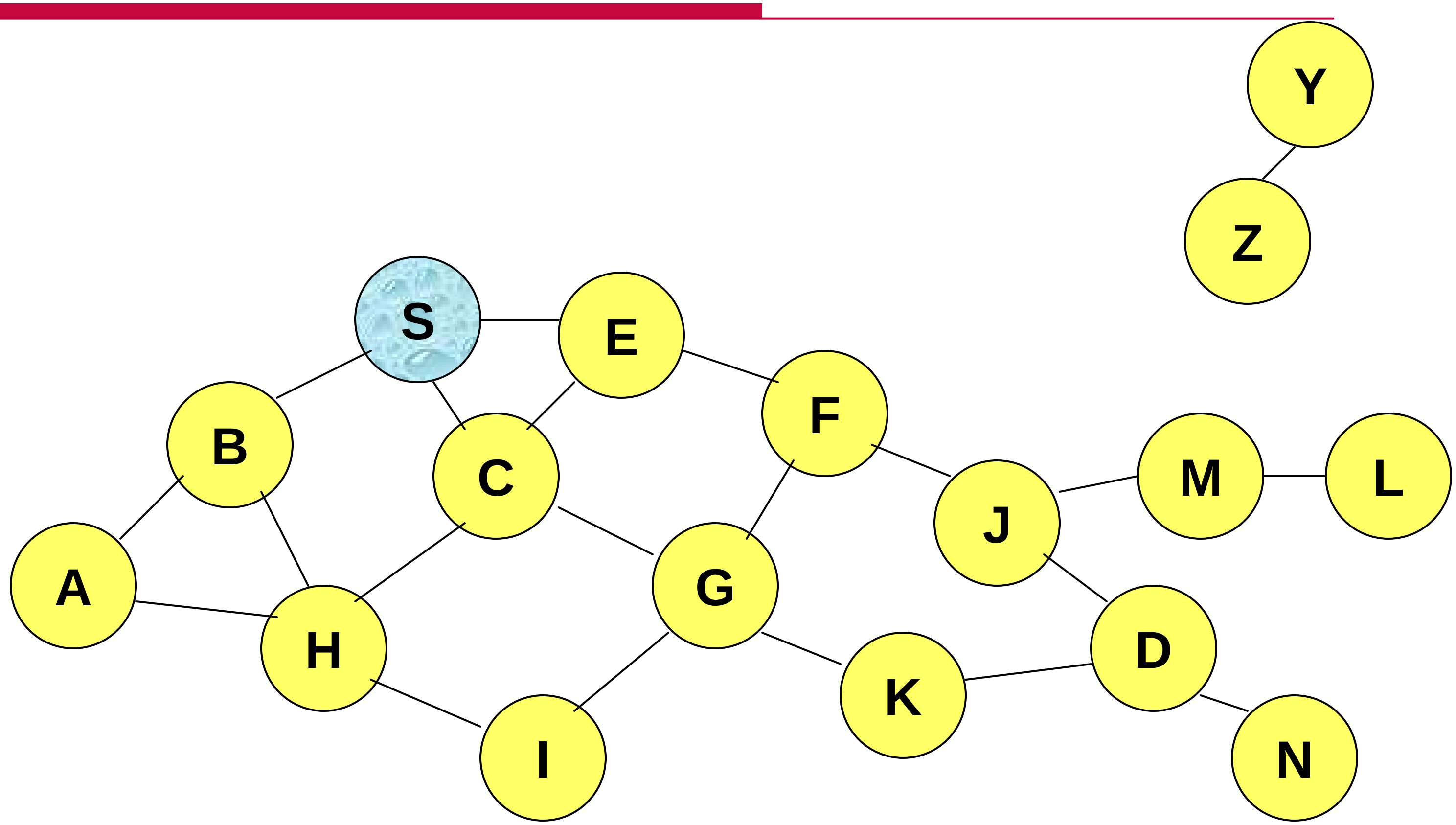
Ad Hoc On-Demand Distance Vector Routing (AODV)

- DSR includes source routes in packet headers
- Resulting large headers can sometimes degrade performance
  - particularly when data contents of a packet are small
- 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

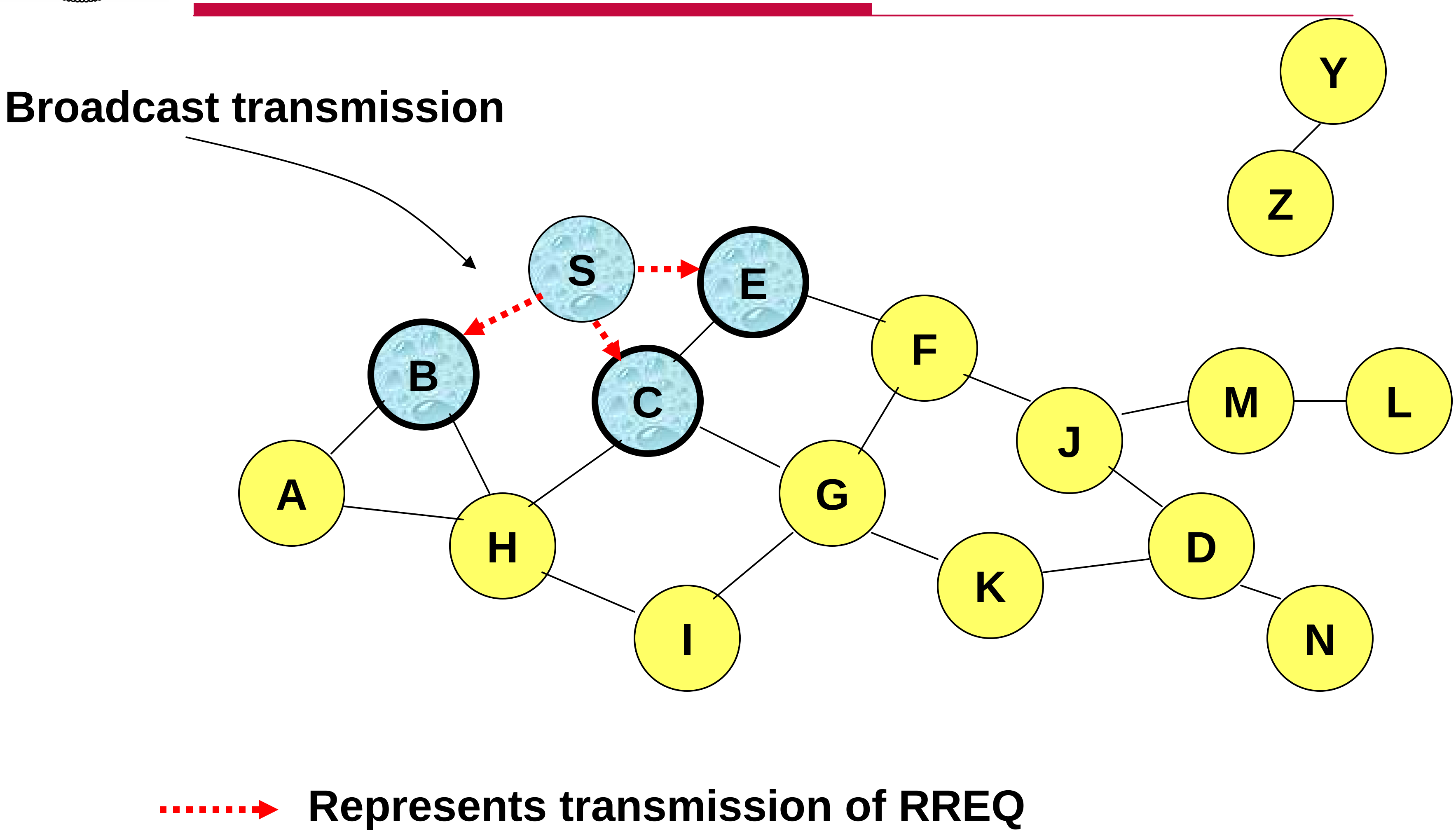
- Route Requests (RREQ) are forwarded in a manner similar to DSR
- When a node re-broadcasts a Route Request, it sets up a reverse path pointing towards the source
  - AODV assumes symmetric (bi-directional) links
- When the intended destination receives a Route Request, it replies by sending a Route Reply
- Route Reply travels along the reverse path set-up when Route Request is forwarded

Route Requests in AODV

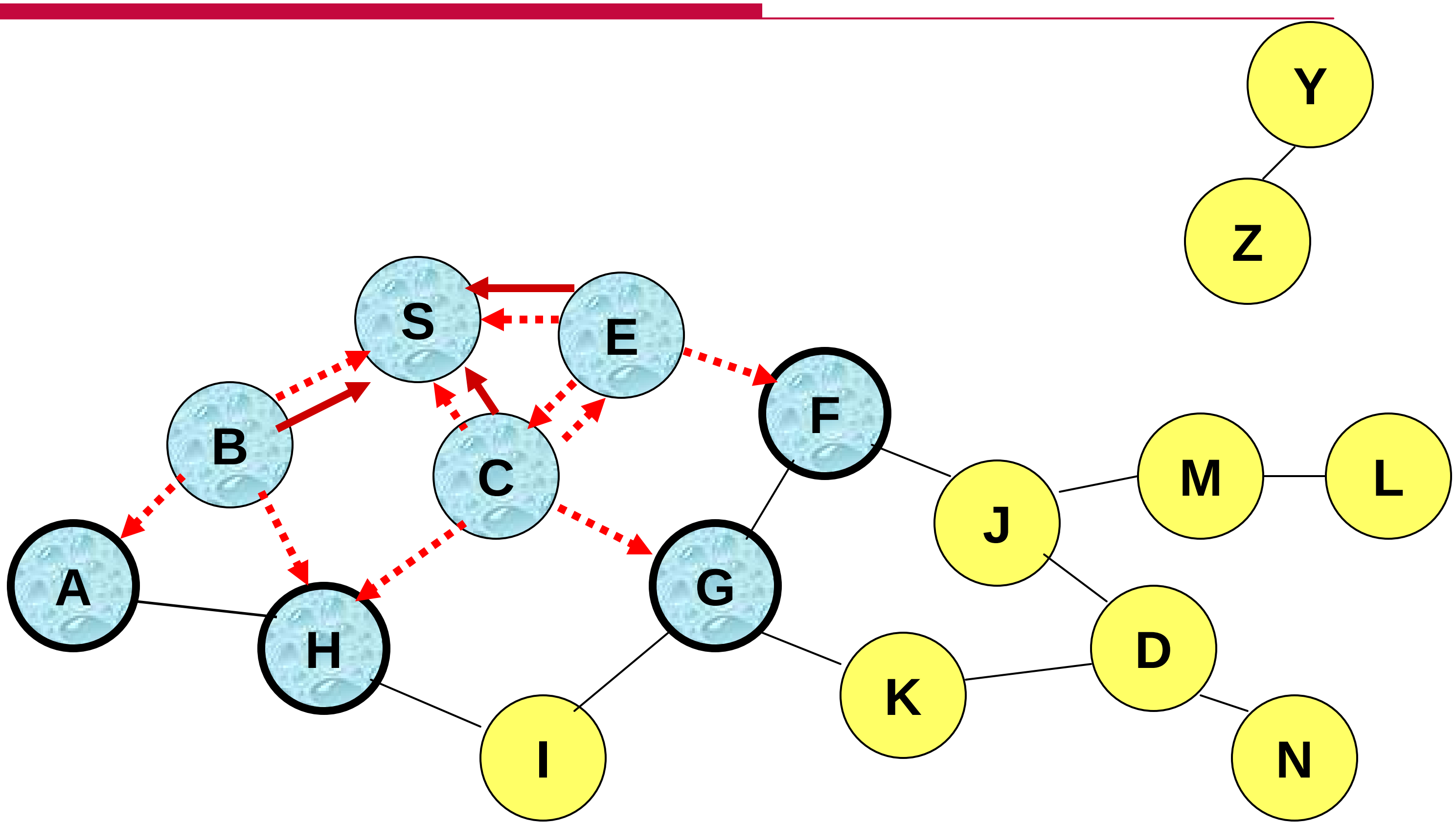


Represents a node that has received RREQ for D from S

Route Requests in AODV

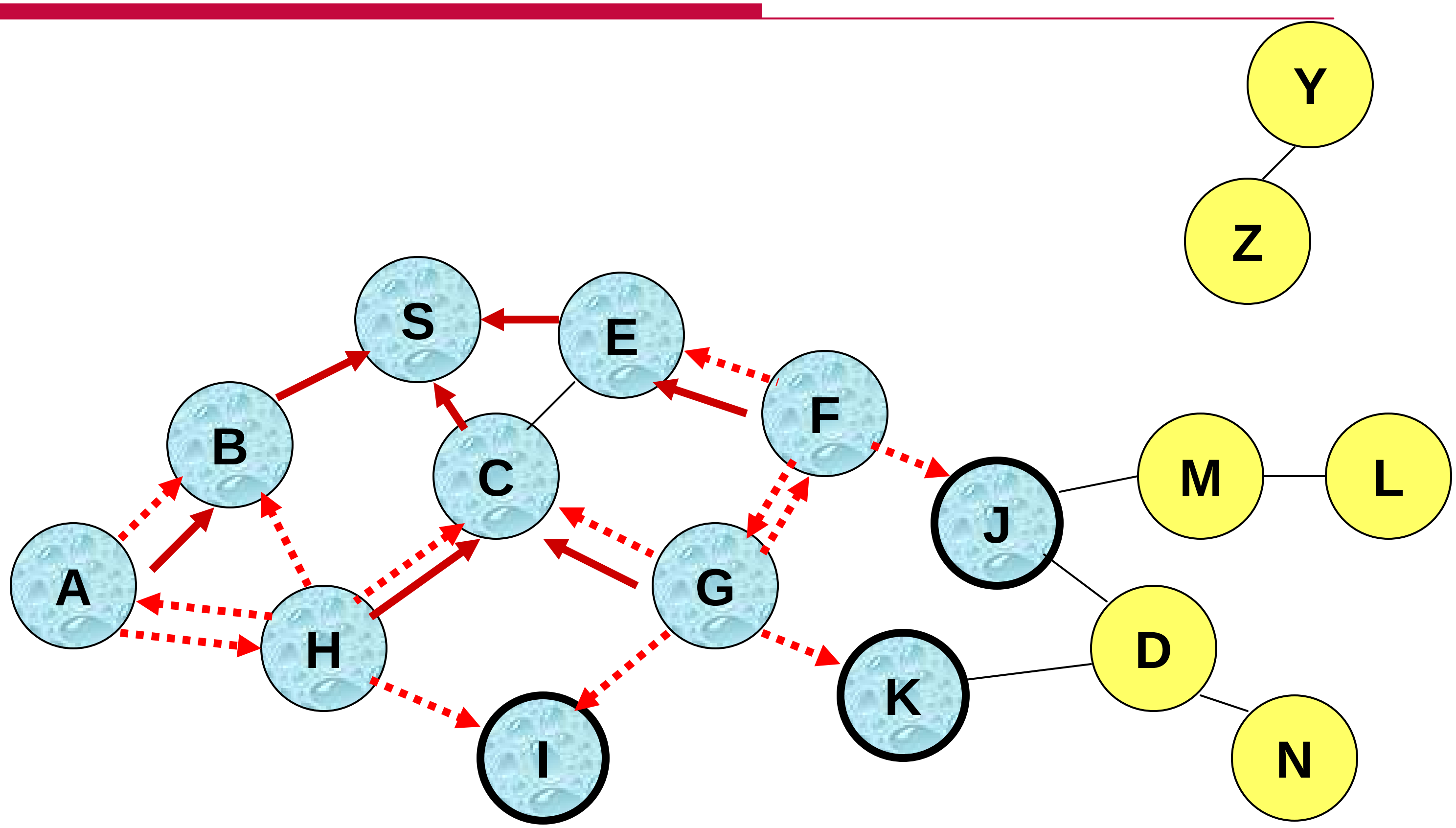


Route Requests in AODV



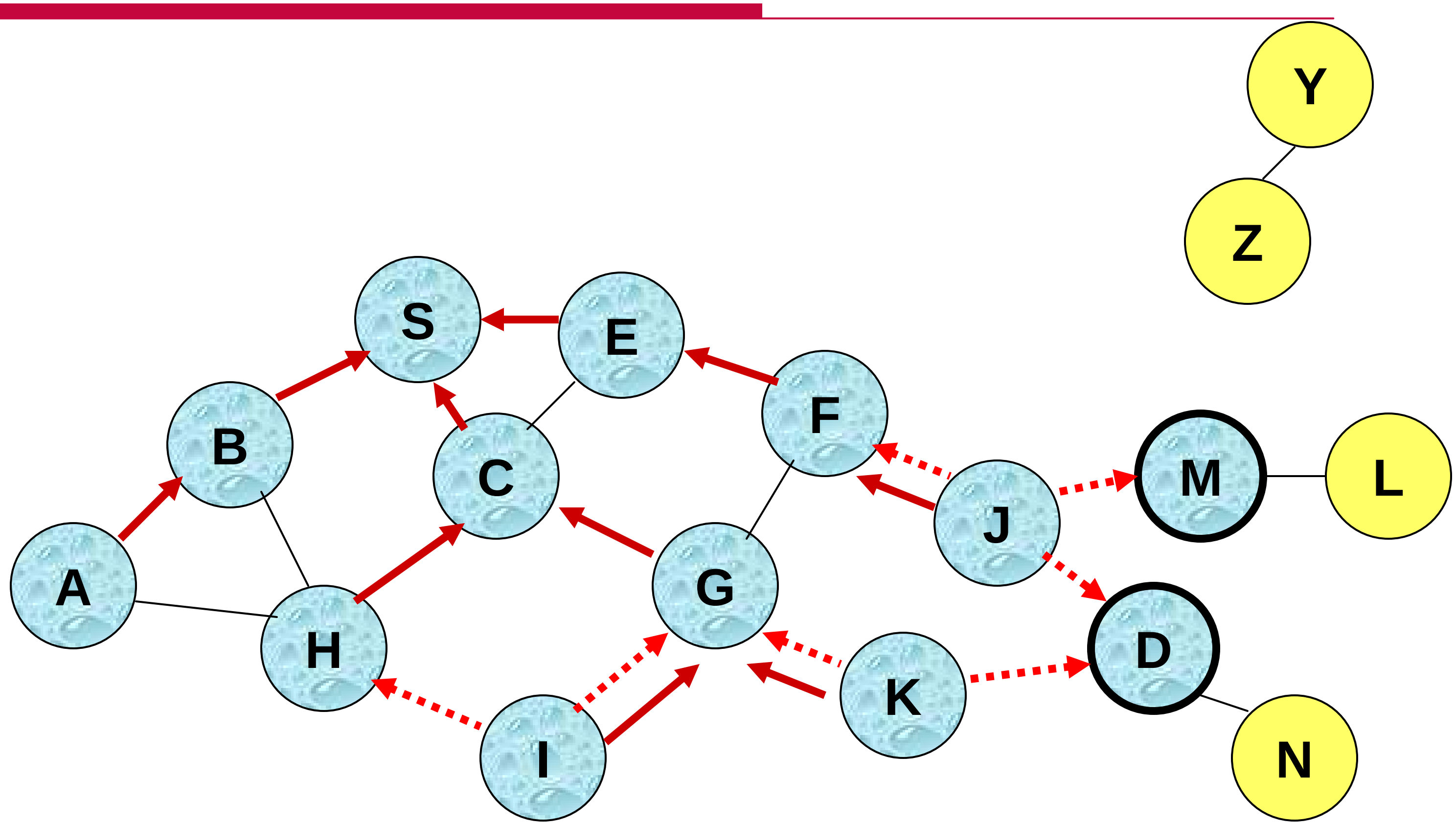
Represents links on Reverse Path

Reverse Path Setup in AODV

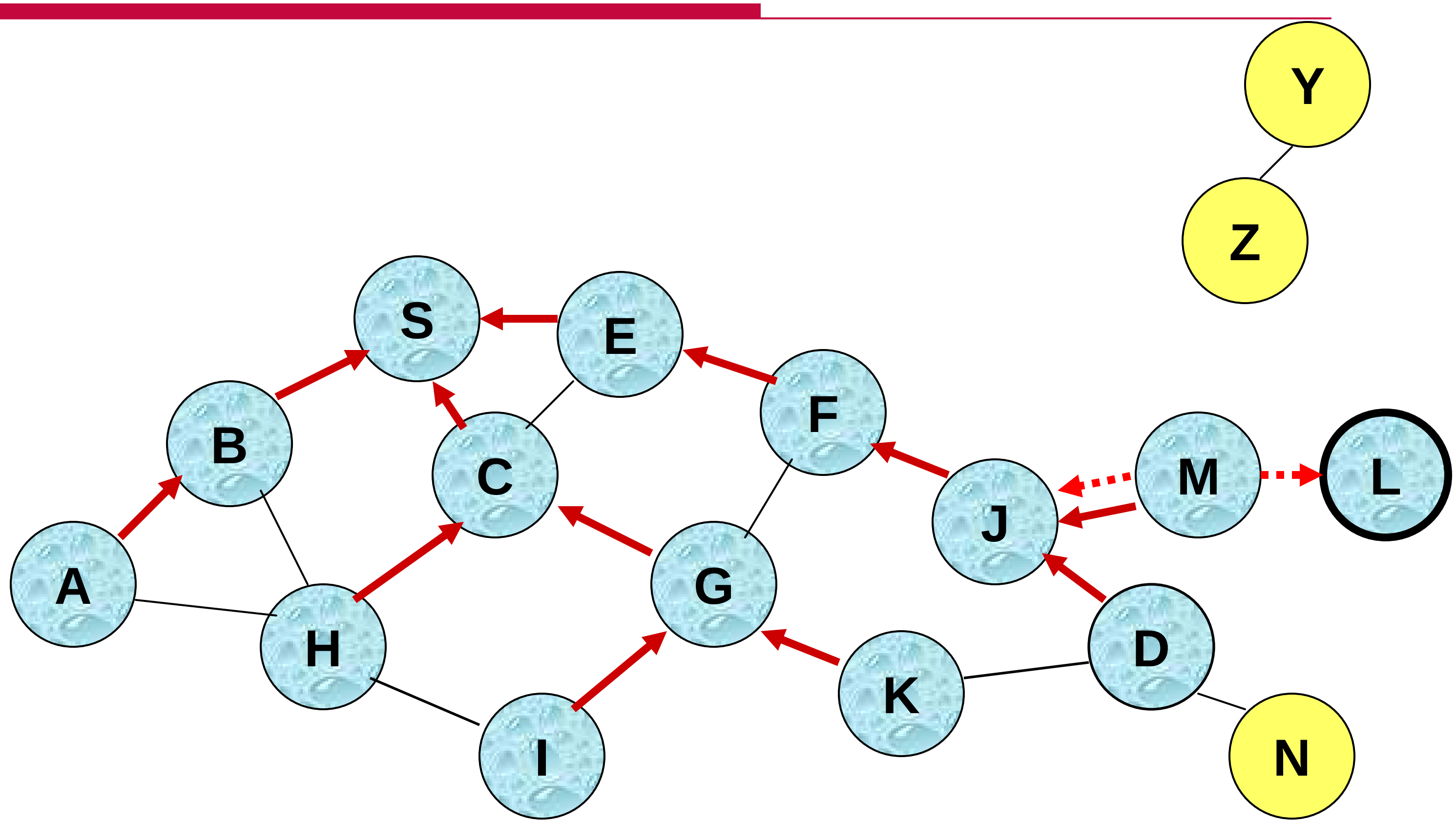


- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

Reverse Path Setup in AODV

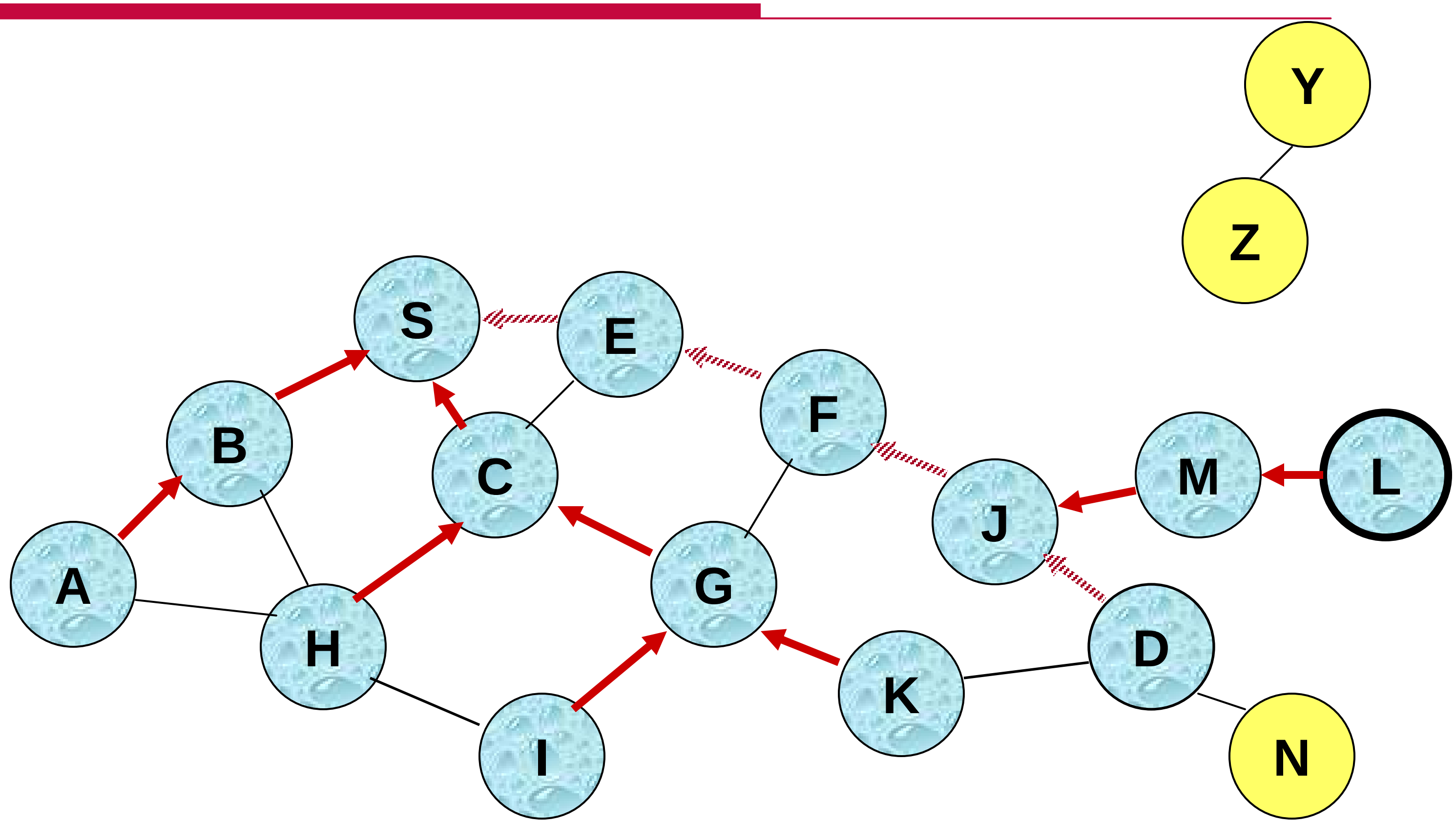


Reverse Path Setup in AODV



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

Route Reply in AODV

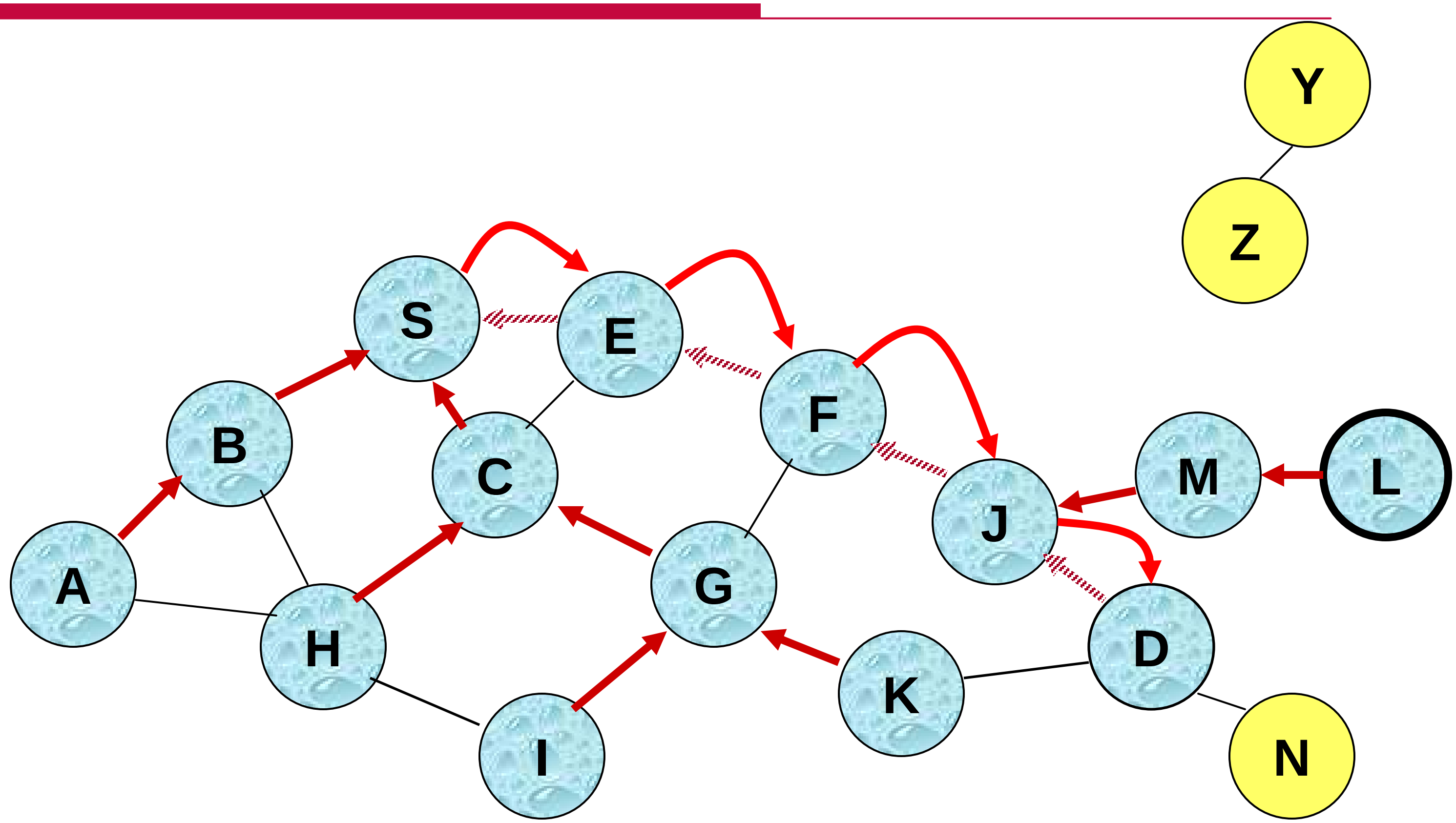


Represents links on path taken by RREP

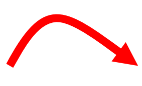
Route Reply in AODV

- 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
  - 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

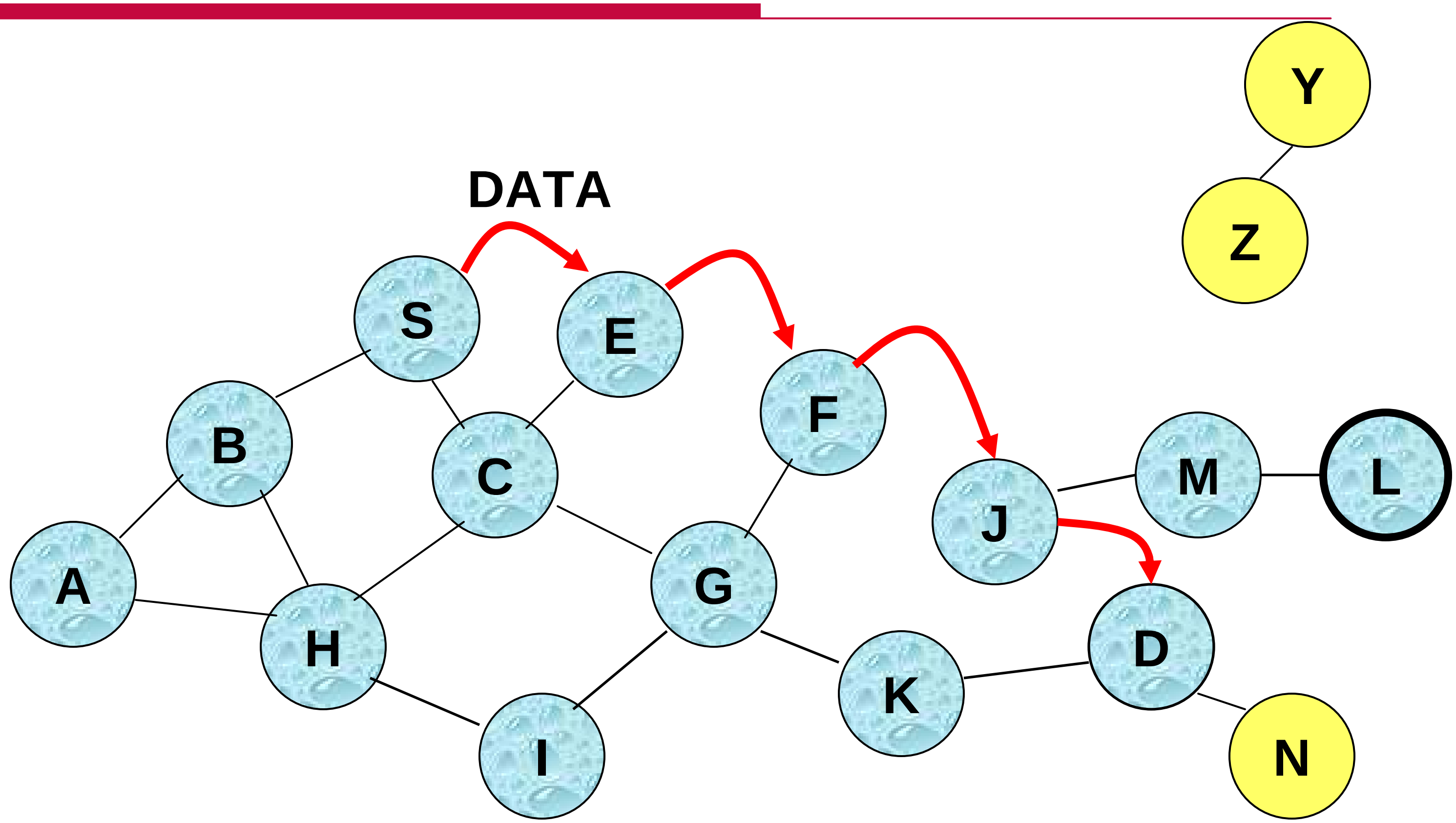
Forward Path Setup in AODV



Forward links are setup when RREP travels along the reverse path

 Represents a link on the forward path

Data Delivery in AODV



Routing table entries used to forward data packet.
Route is *not* included in packet header.

Timeouts

- A routing table entry maintaining a reverse path is purged after a timeout interval
  - timeout should be long enough to allow RREP to come back
- A routing table entry maintaining a forward path is purged if *not used* for a *active_route_timeout* interval
  - if no data is being sent using a particular routing table entry, that entry will be deleted from the routing table (even if the route may actually still be valid)

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 which was forwarded using that entry
- When the next hop link in a routing table entry breaks, all active neighbors are informed
- Link failures are propagated by means of Route Error messages, which also update destination sequence numbers

Route Error

- When node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a RERR message
- Node X increments the destination sequence number for D cached at node X
- The incremented sequence number *N* is included in the RERR
- When node S receives the RERR, it initiates a new route discovery for D using destination sequence number at least as large as *N*

Destination Sequence Number

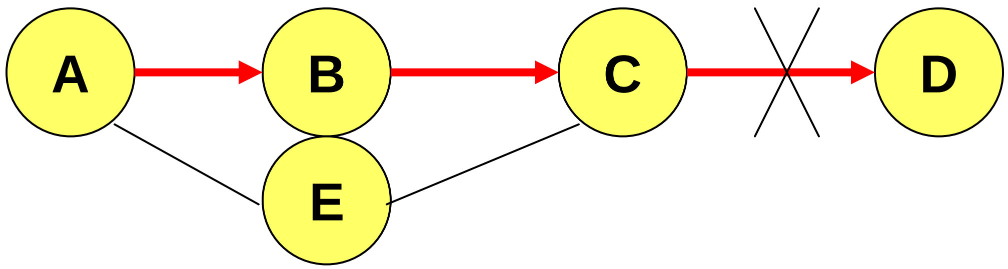
- Continuing from the previous slide **…**
- When node D receives the route request with destination sequence number N, node D will set its sequence number to N, unless it is already larger than N

Link Failure Detection

- *Hello* messages: Neighboring nodes periodically exchange hello message
- Absence of hello message is used as an indication of link failure
- Alternatively, failure to receive several MAC-level acknowledgement may be used as an indication of link failure

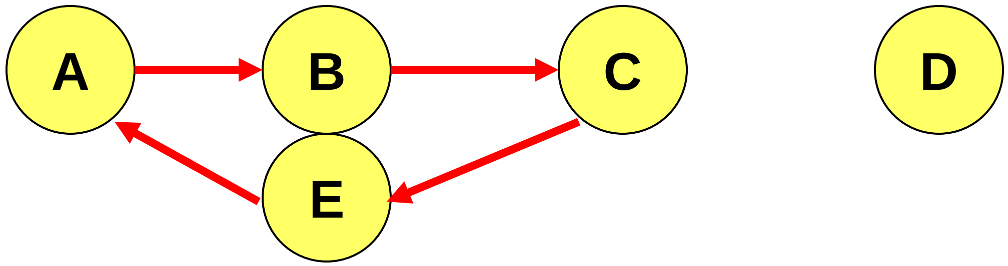
Why Sequence Numbers in AODV

- To avoid using old/broken routes
  - To determine which route is newer
- To prevent formation of loops



  - Assume that A does not know about failure of link C-D because RERR sent by C is lost
  - Now C performs a route discovery for D. Node A receives the RREQ (say, via path C-E-A)
  - Node A will reply since A knows a route to D via node B
  - Results in a loop (for instance, C-E-A-B-C)

Why Sequence Numbers in AODV



- Loop C-E-A-B-C

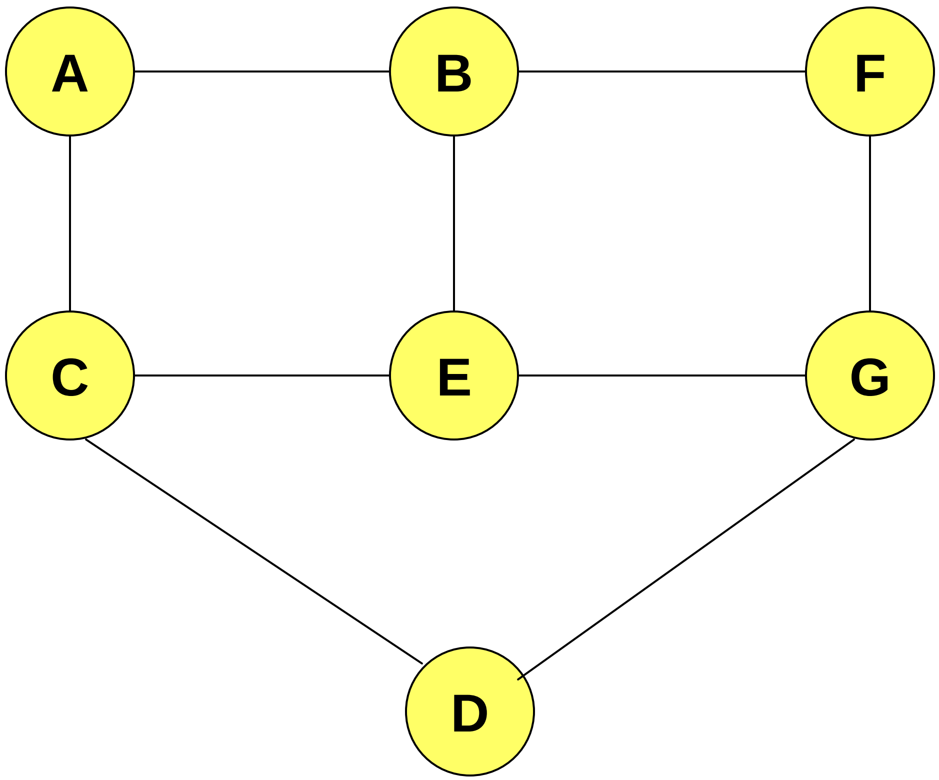
Optimization: Expanding Ring Search

- Route Requests are initially sent with small Time-to-Live (TTL) field, to limit their propagation
  - DSR also includes a similar optimization
- If no Route Reply is received, then larger TTL tried

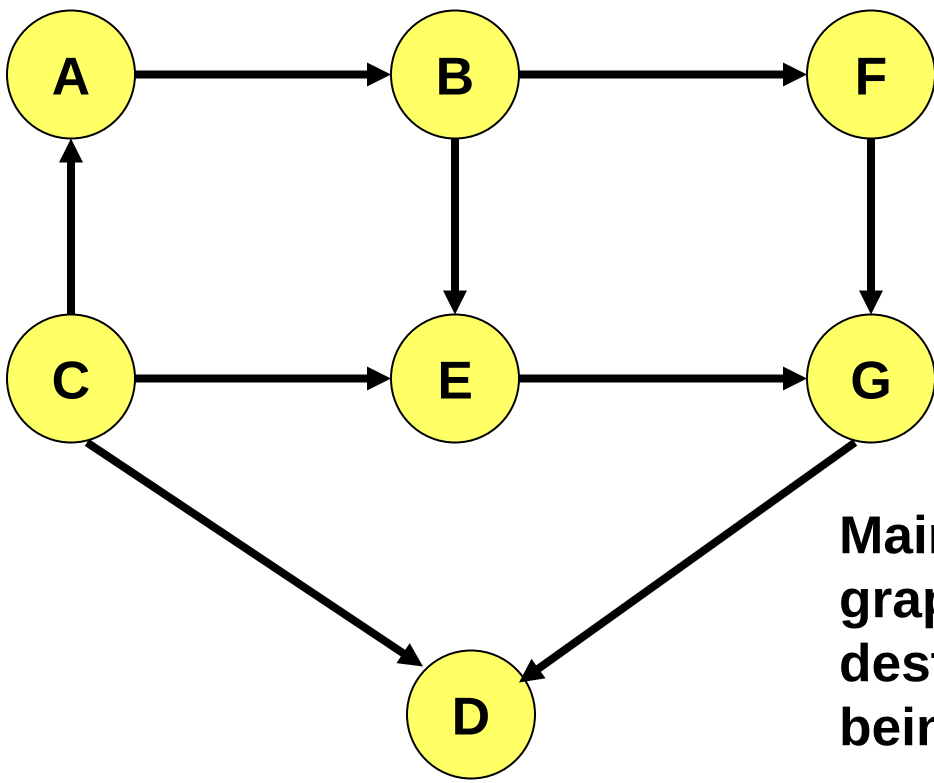
Summary: AODV

- Routes need not be included in packet headers
- Nodes maintain routing tables containing entries only for routes that are in active use
- At most one next-hop per destination maintained at each node
  - Multi-path extensions can be designed
  - DSR may maintain several routes for a single destination
- Unused routes expire even if topology does not change

Link Reversal Algorithm



Link Reversal Algorithm



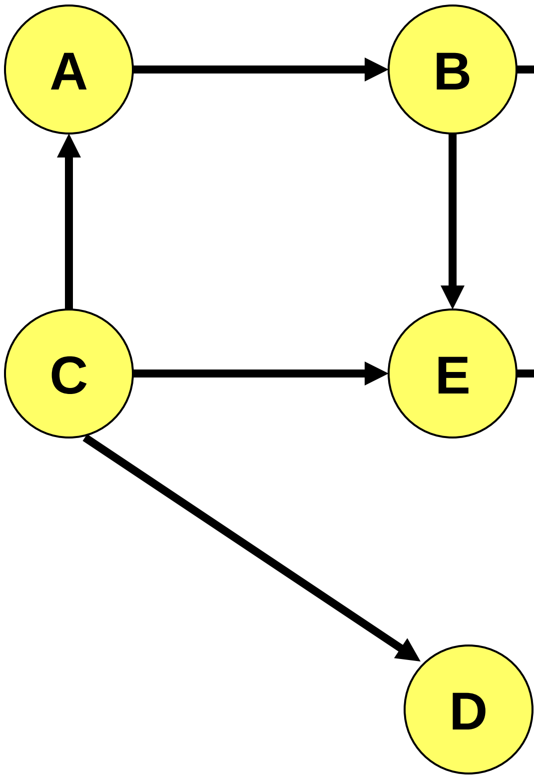
Links are bi-directional

But algorithm imposes logical directions on them

Maintain a directed acyclic graph (DAG) for each destination, with the destination being the *only sink*

This DAG is for *destination node D*

Link Reversal Algorithm

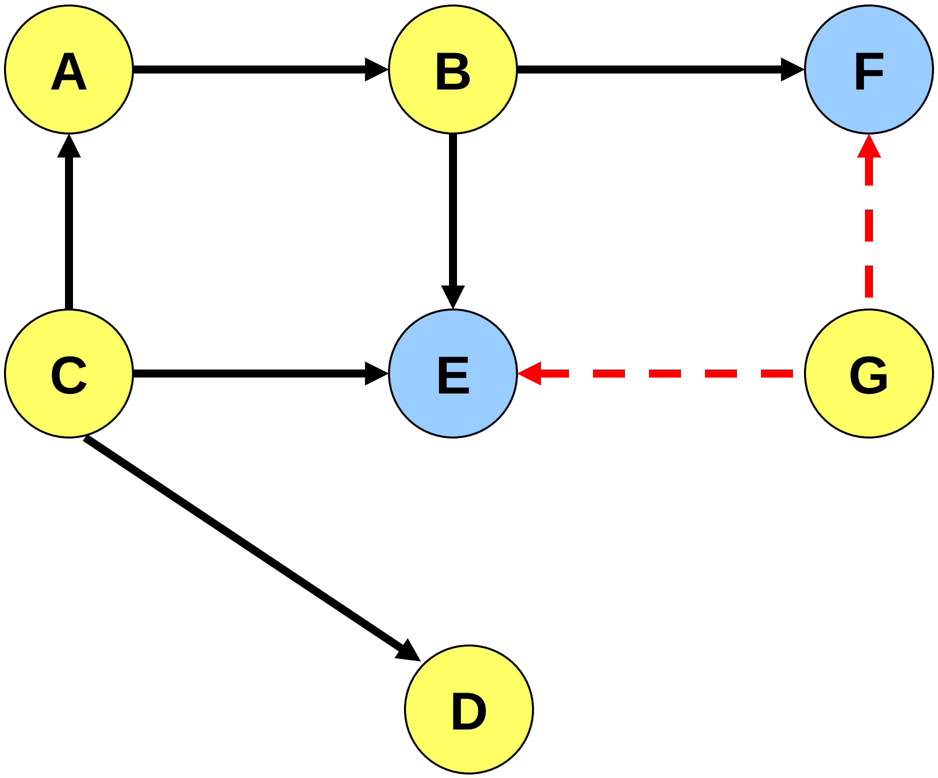


Link (G,D) broke

Any node, other than the destination, that has no outgoing links reverses all its incoming links.

Node G has no outgoing links

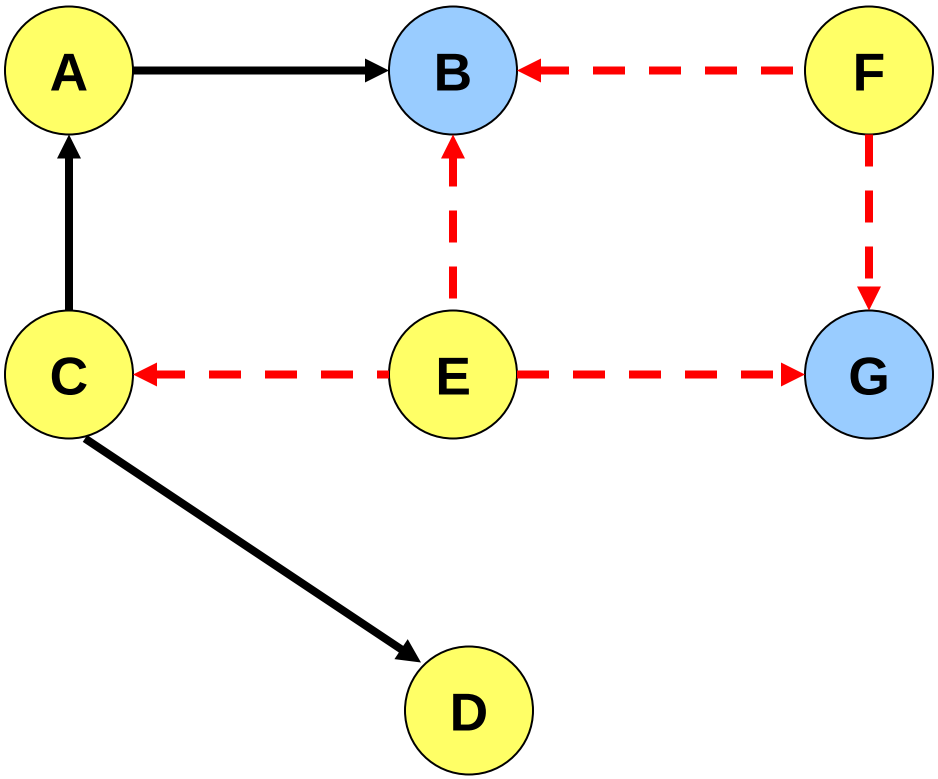
Link Reversal Algorithm



Represents a link that was reversed recently

Now nodes E and F have no outgoing links

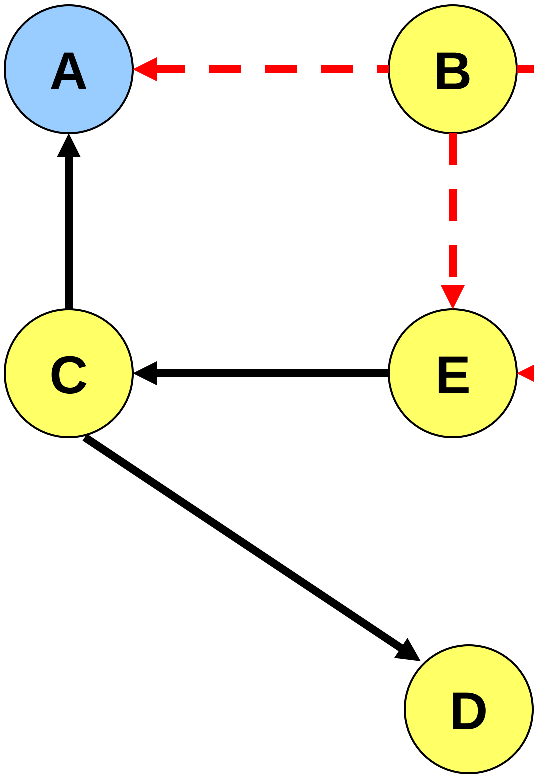
Link Reversal Algorithm



Represents a link that was reversed recently

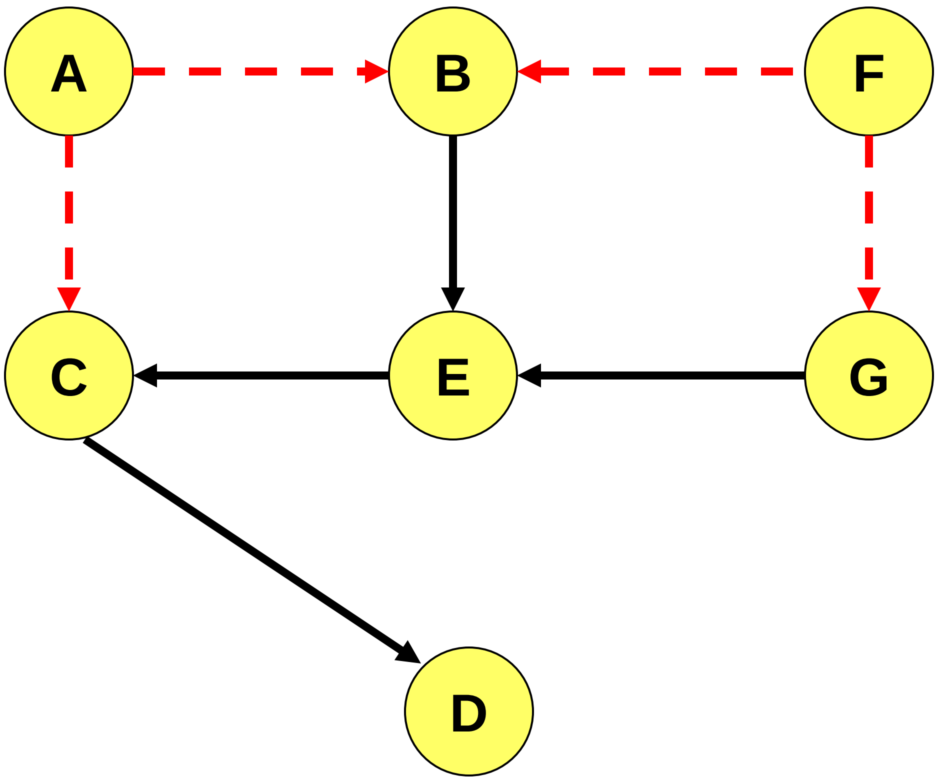
Now nodes B and G have no outgoing links

Link Reversal Algorithm



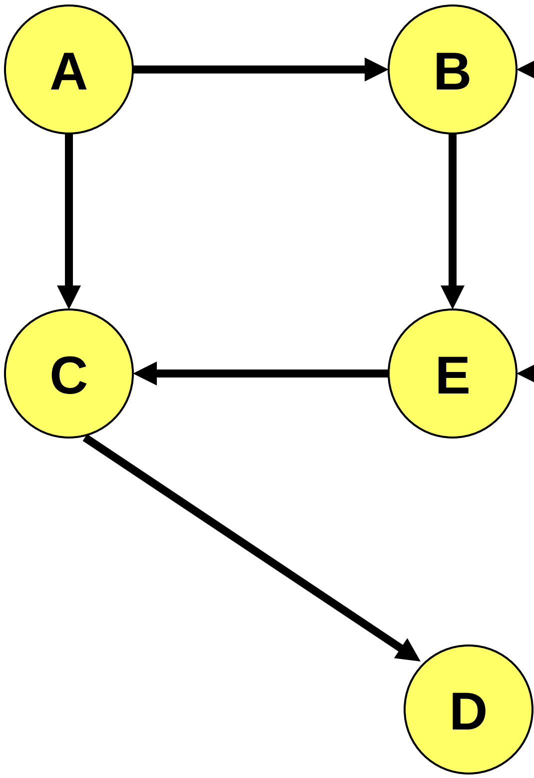
Now nodes A and F have no outgoing links

Link Reversal Algorithm



Now all nodes (other than destination D) have an outgoing link

Link Reversal Algorithm



DAG has been restored with only the destination as a sink

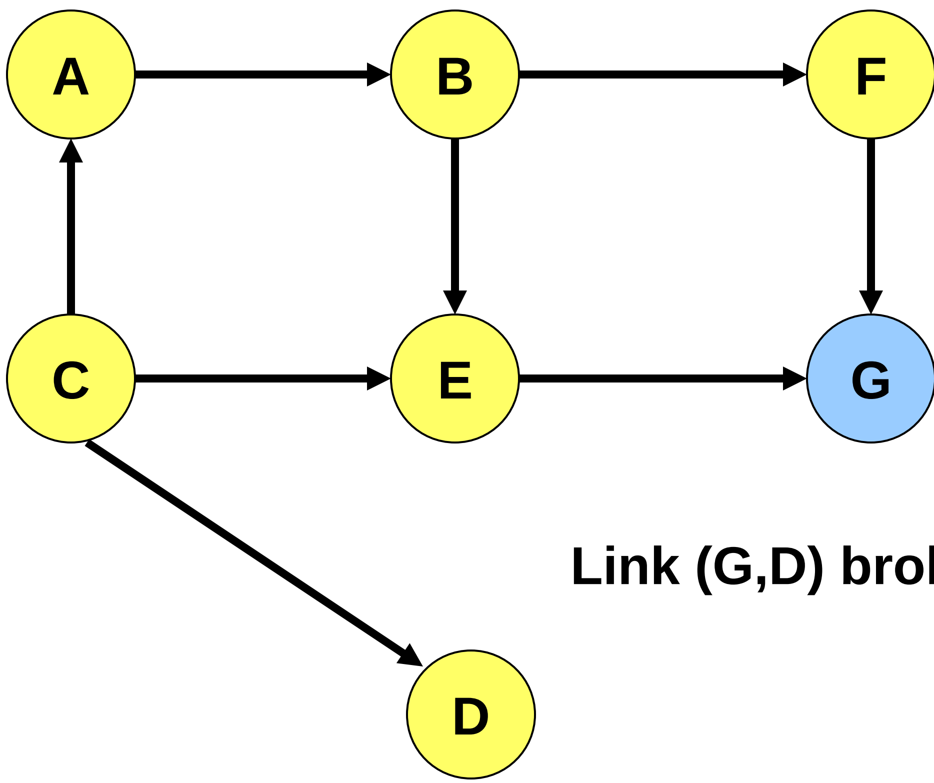
Link Reversal Algorithm

- Attempts to keep link reversals local to where the failure occurred
  - But this is not guaranteed
- When the first packet is sent to a destination, the destination oriented DAG is constructed
- The initial construction does result in flooding of control packets

Link Reversal Algorithm

- The previous algorithm is called a full reversal method since when a node reverses links, it reverses *all* its incoming links
- Partial reversal method: A node reverses incoming links from only those neighbors who have not themselves reversed links "previously"
  - If all neighbors have reversed links, then the node reverses all its incoming links
  - "Previously" at node X means *since the last link reversal done by node X*

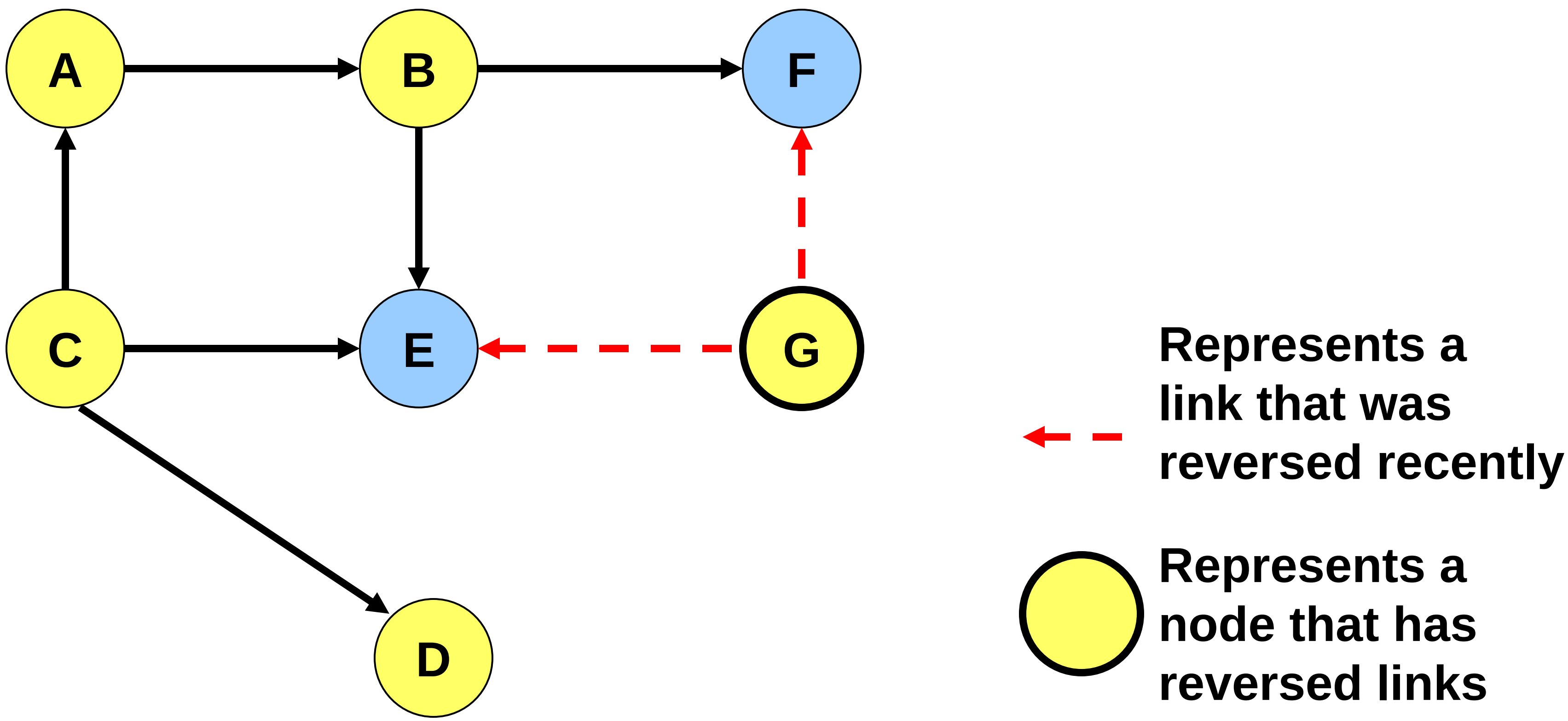
Partial Reversal Method



Link (G,D) broke

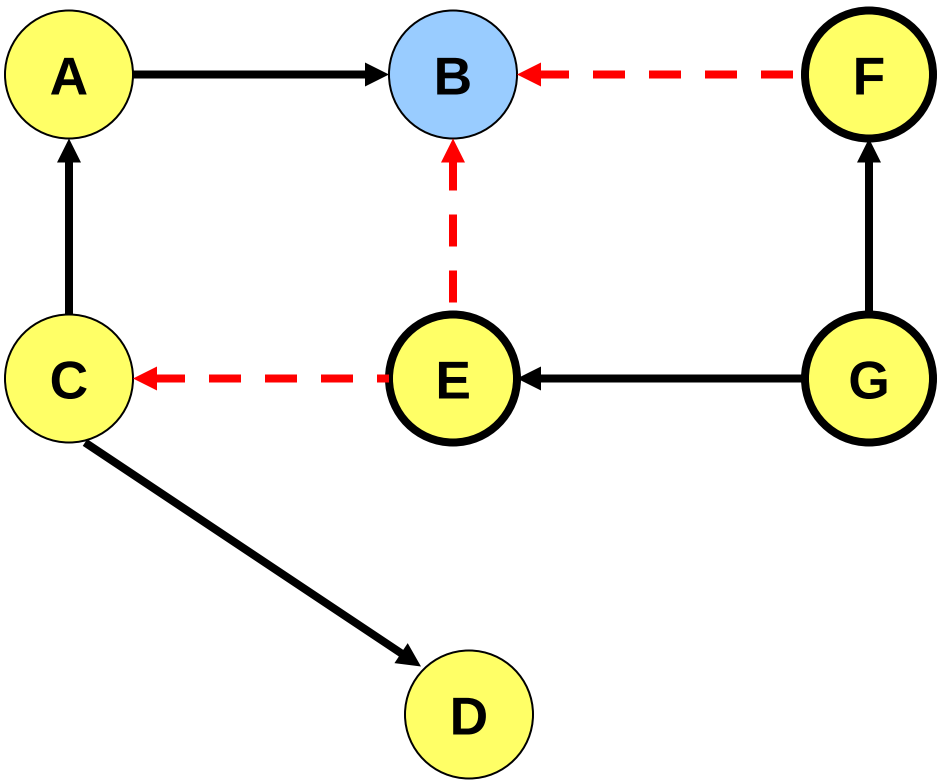
Node G has no outgoing links

Partial Reversal Method



Now nodes E and F have no outgoing links

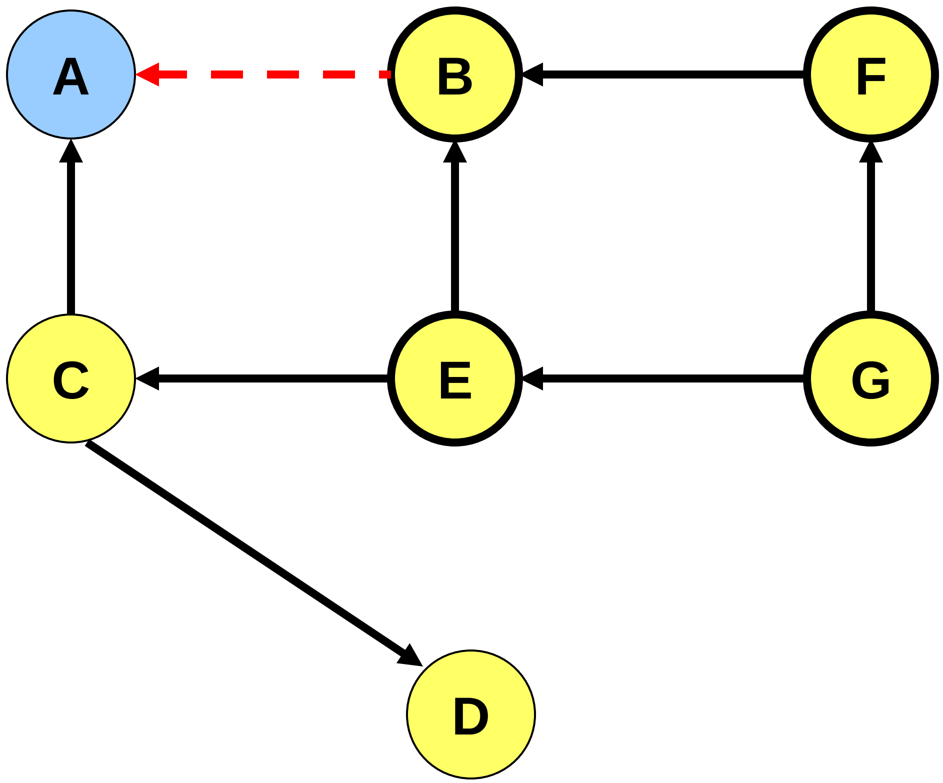
Partial Reversal Method



Represents a link that was reversed recently

Nodes E and F *do not* reverse links from node G
Now node B has no outgoing links

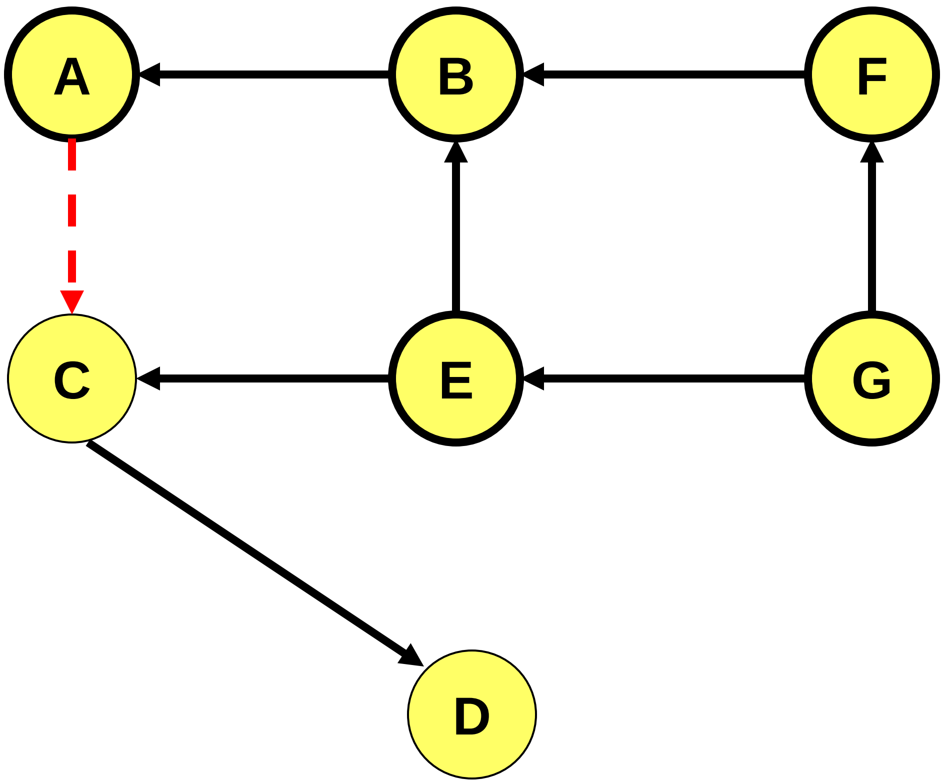
Partial Reversal Method



Represents a link that was reversed recently

Now node A has no outgoing links

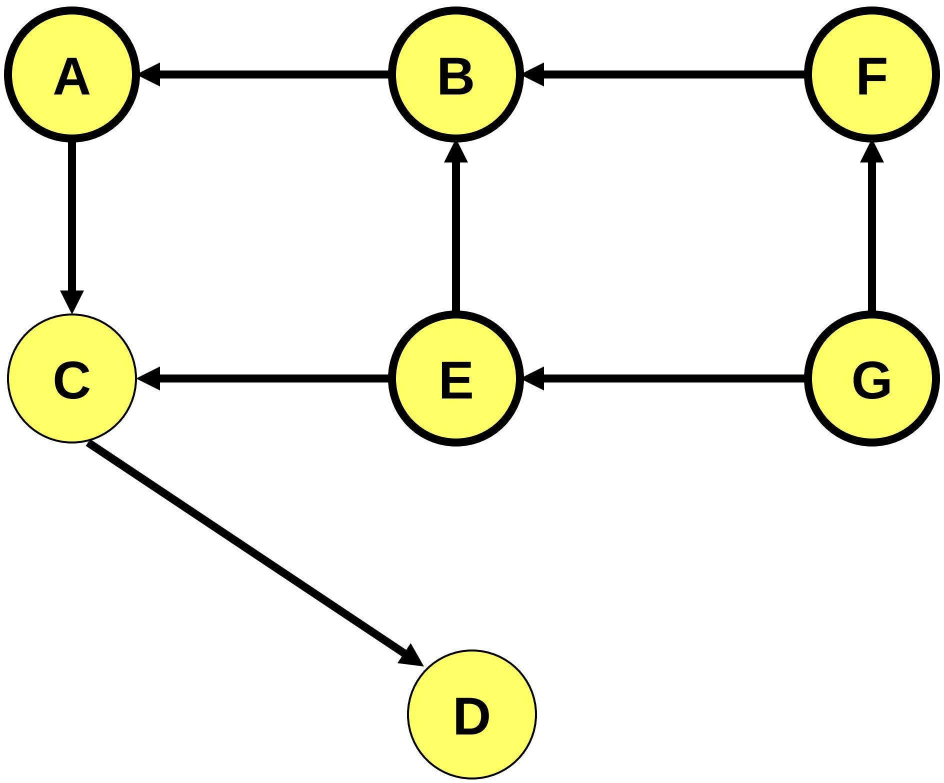
Partial Reversal Method



Represents a link that was reversed recently

Now all nodes (except destination D) have outgoing links

Partial Reversal Method



DAG has been restored with only the destination as a sink

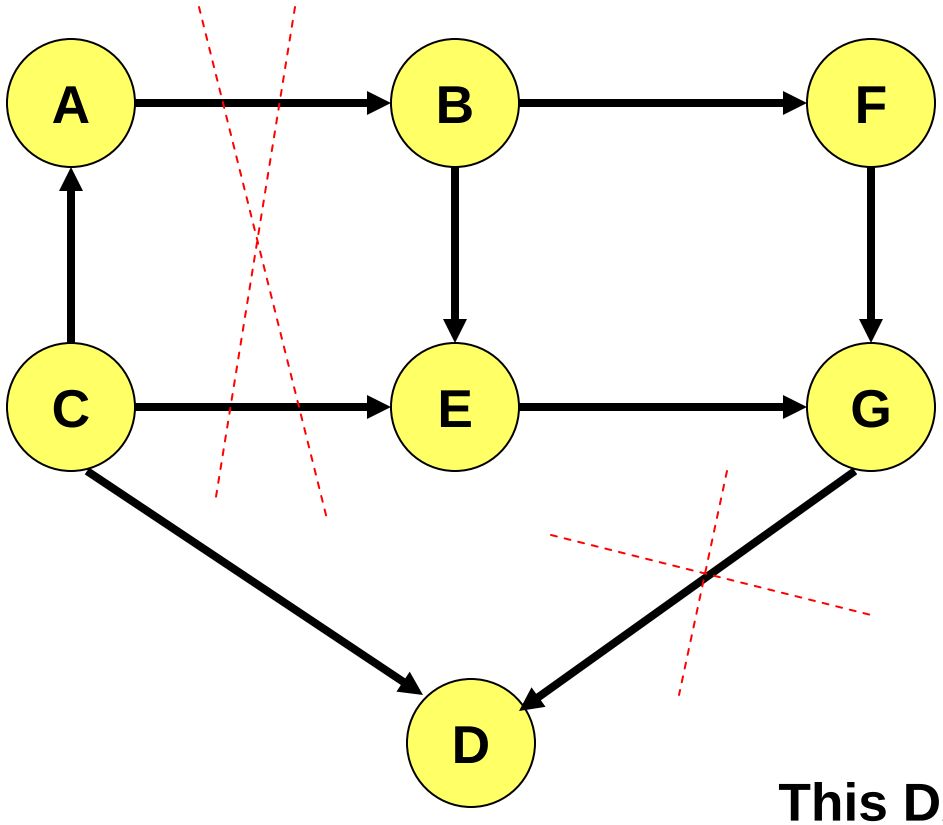
Link Reversal Methods: Advantages

- Link reversal methods attempt to limit updates to routing tables at nodes in the vicinity of a broken link
- Each node may potentially have multiple routes to a destination

Link Reversal Methods: Disadvantage

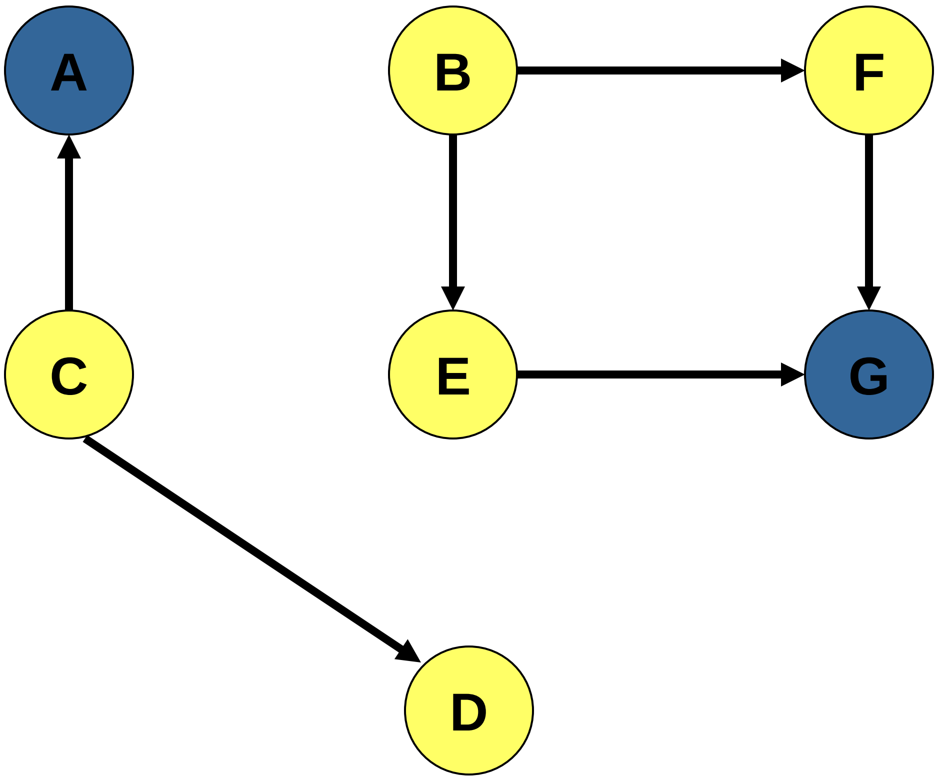
- Need a mechanism to detect link failure
  - hello messages may be used
  - but hello messages can add to contention
- If network is partitioned, link reversals continue indefinitely

Link Reversal in a Partitioned Network



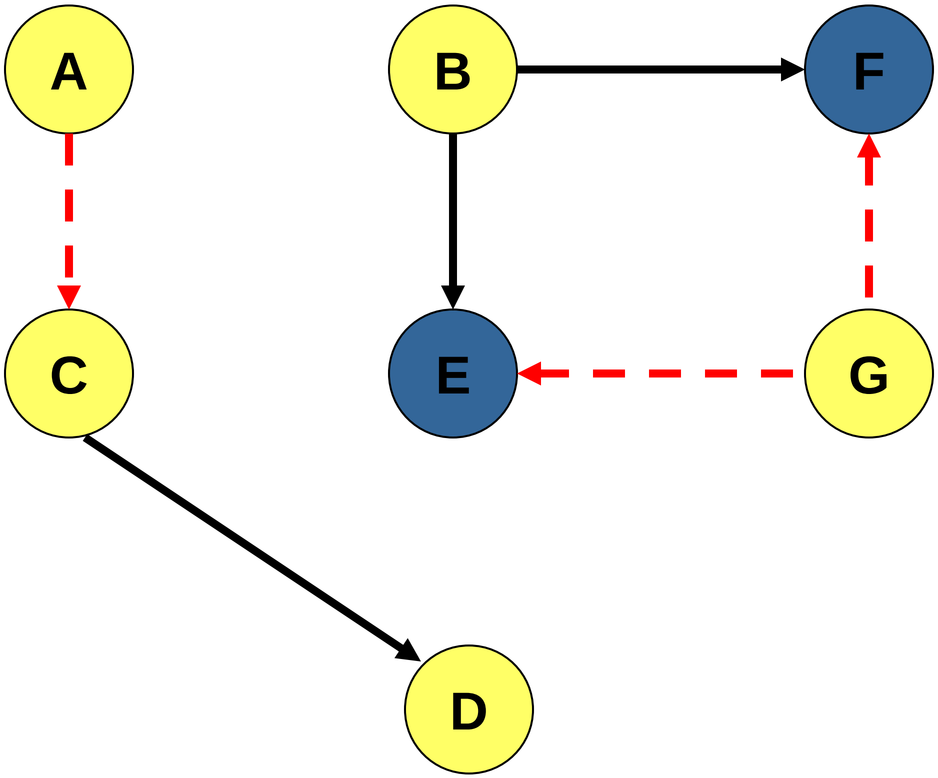
This DAG is for ***destination node D***

Full Reversal in a Partitioned Network



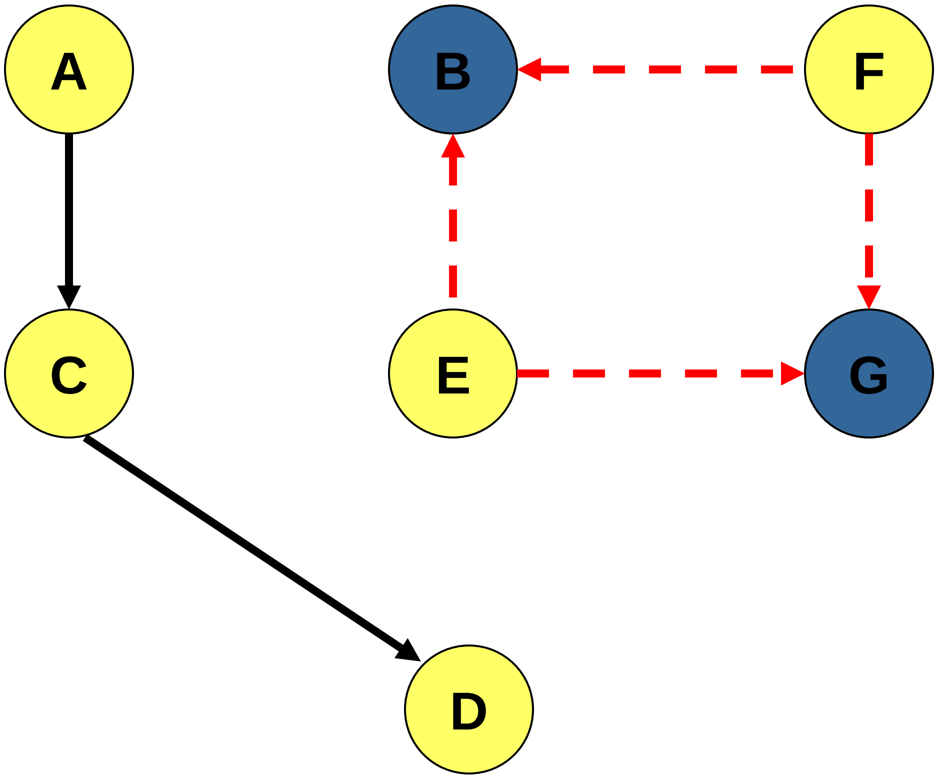
A and G do not have outgoing links

Full Reversal in a Partitioned Network



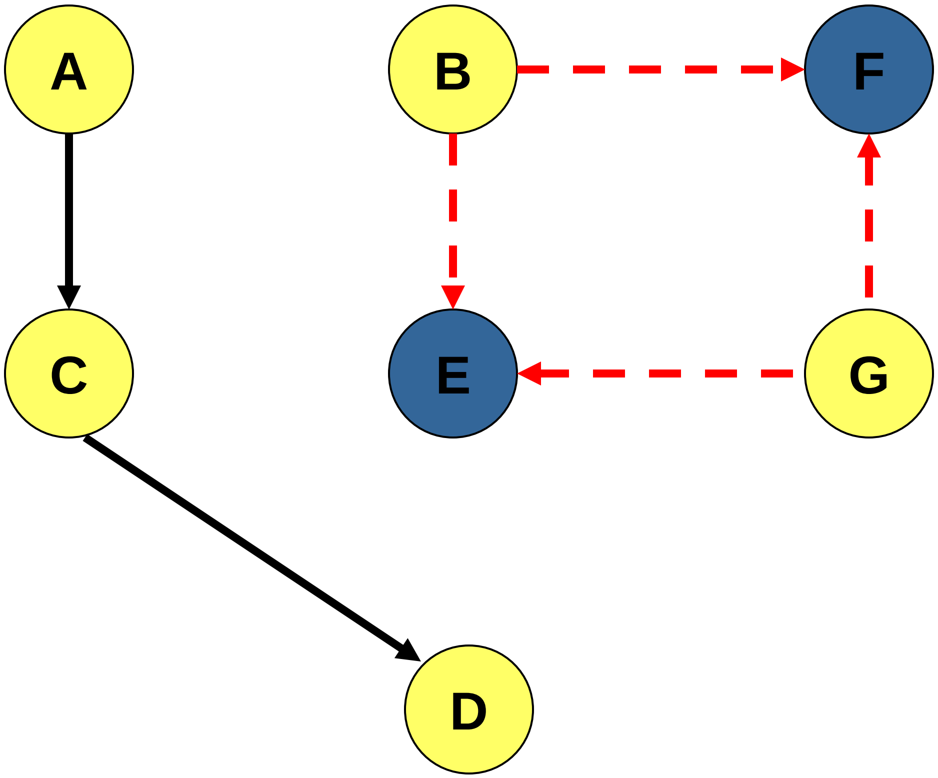
E and F do not have outgoing links

Full Reversal in a Partitioned Network



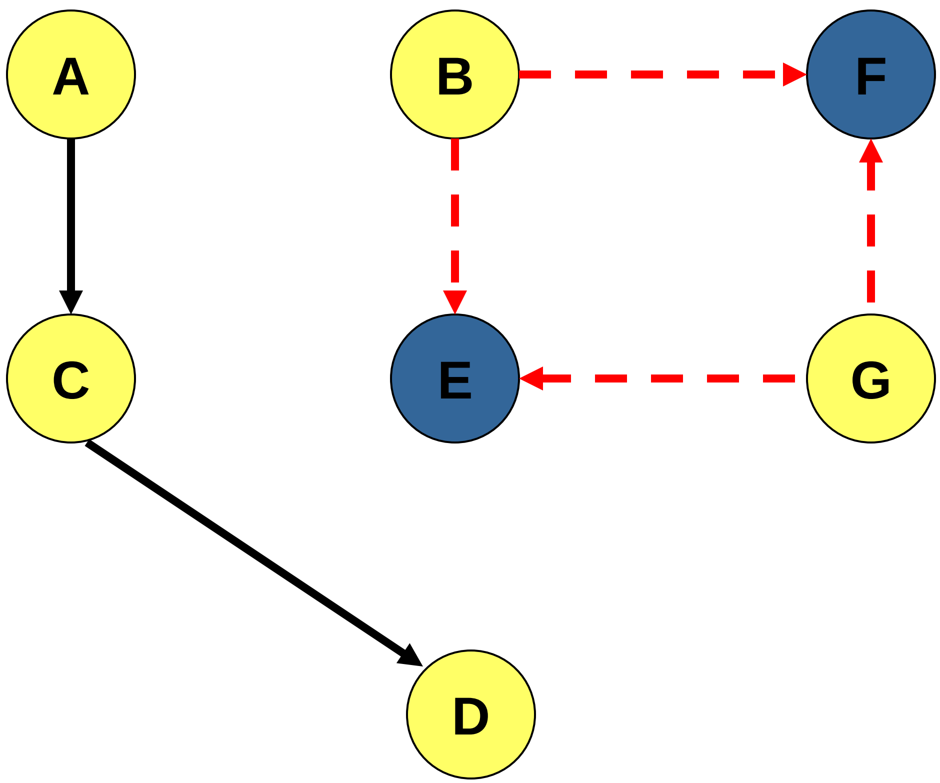
B and G do not have outgoing links

Full Reversal in a Partitioned Network



E and F do not have outgoing links

Full Reversal in a Partitioned Network



In the partition disconnected from destination D, link reversals continue, until the partitions merge

Need a mechanism to minimize this wasteful activity

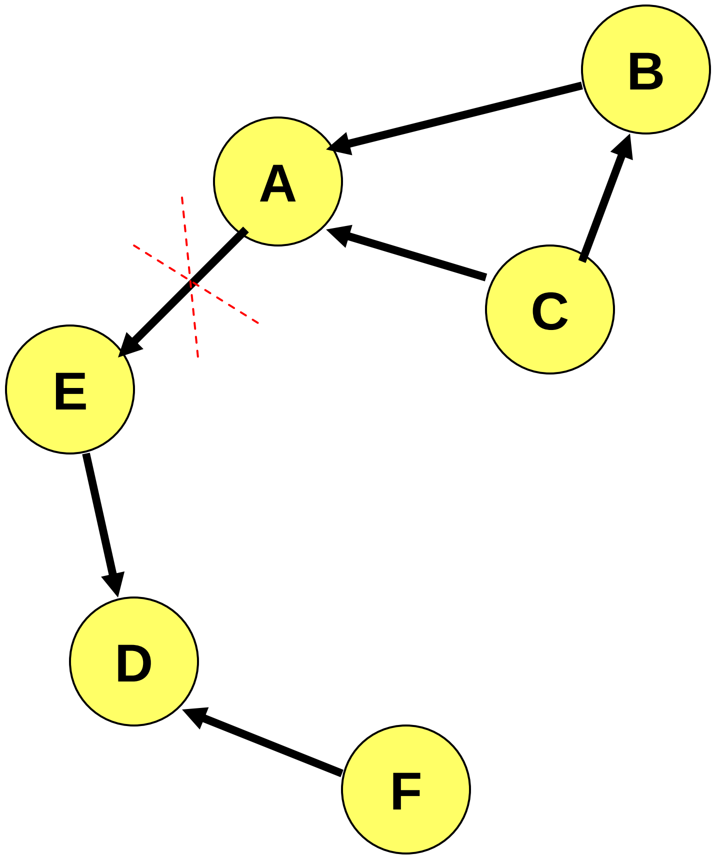
Similar scenario can occur with partial reversal method too

Temporally-Ordered Routing Algorithm (TORA)

- TORA modifies the partial link reversal method to be able to detect partitions

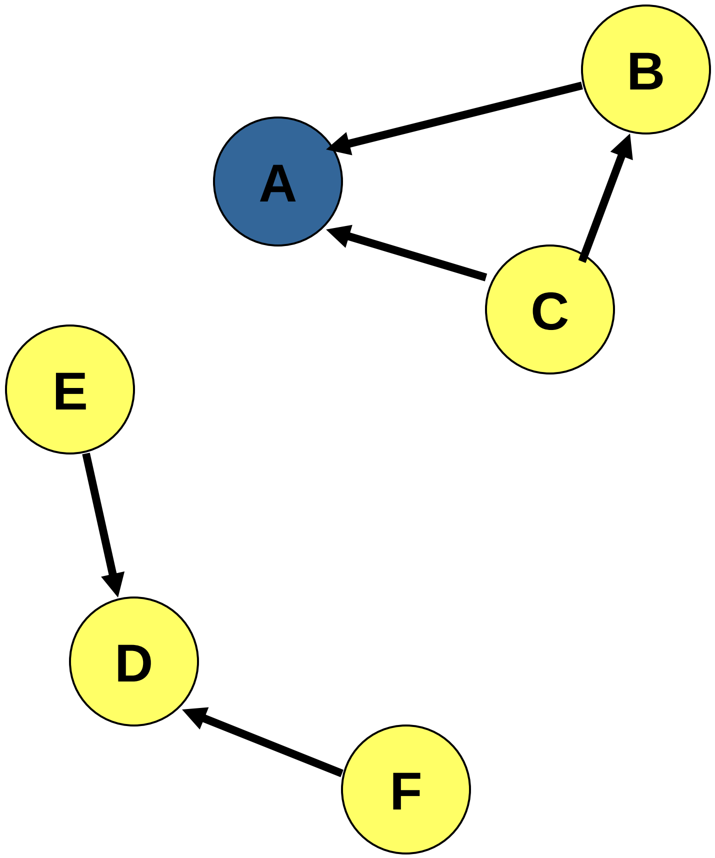
- When a partition is detected, all nodes in the partition are informed, and link reversals in that partition cease

Partition Detection in TORA



DAG for destination D

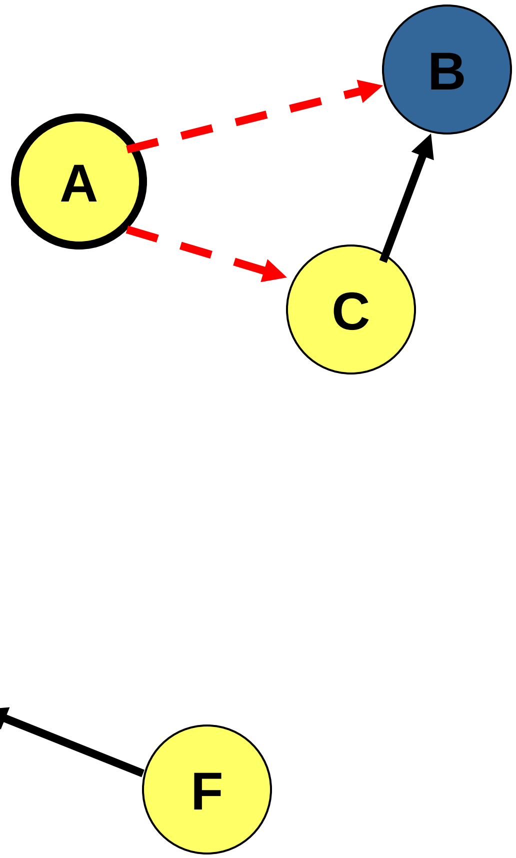
Partition Detection in TORA



TORA uses a modified partial reversal method

Node A has no outgoing links

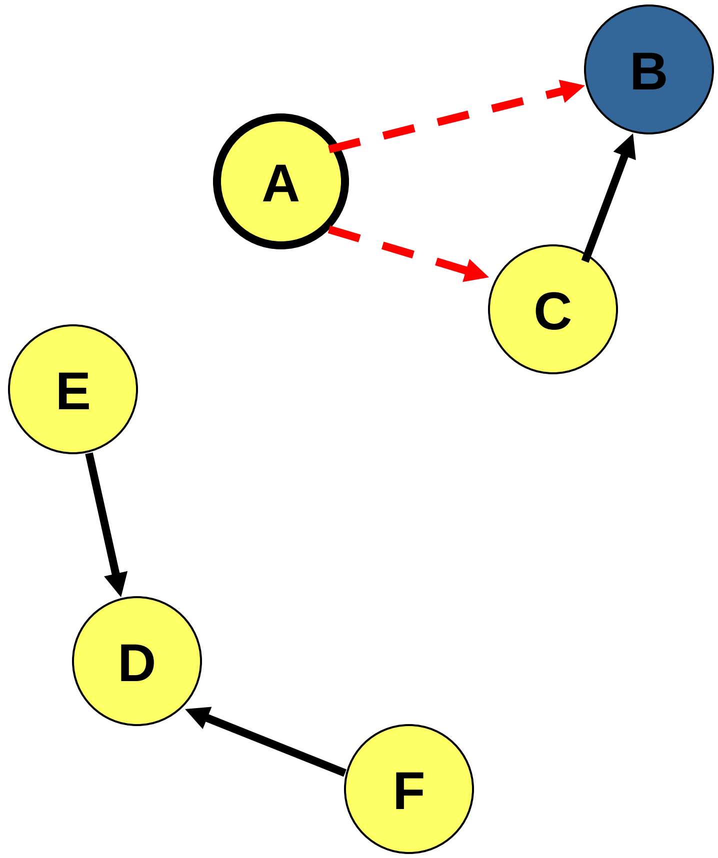
Partition Detection in TORA



TORA uses a modified partial reversal method

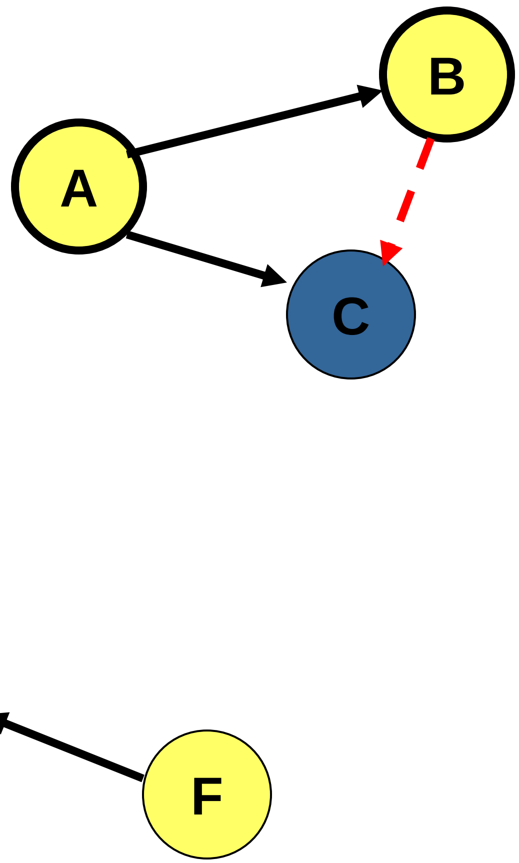
Node B has no outgoing links

Partition Detection in TORA



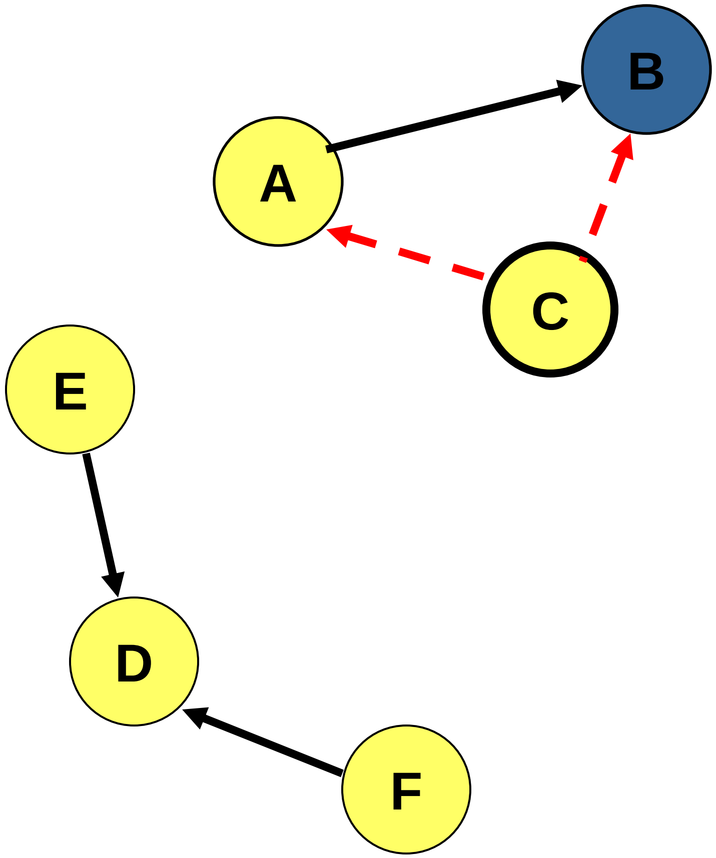
Node B has no outgoing links

Partition Detection in TORA



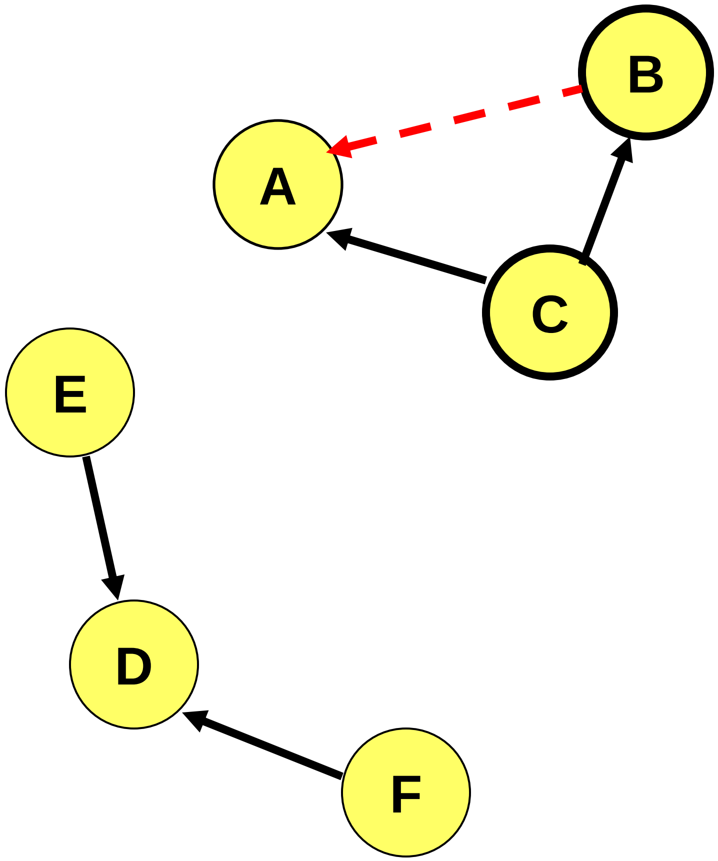
Node C has no outgoing links -- all its neighbor have reversed links previously.

Partition Detection in TORA



Nodes A and B receive the reflection from node C
Node B now has no outgoing link

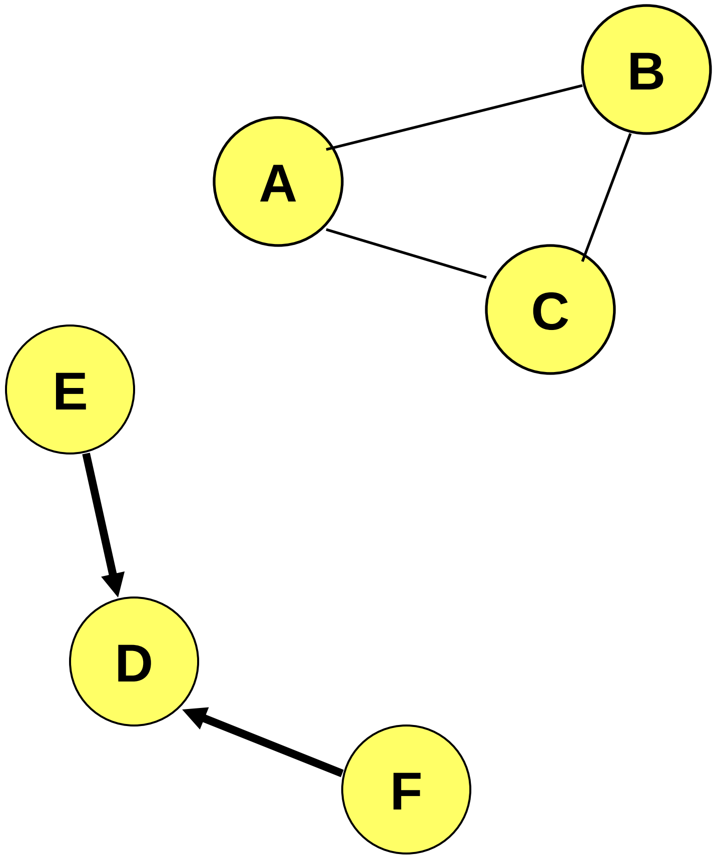
Partition Detection in TORA



Node B **propagates** the **reflection** to node A

Node A has received the reflection from all its neighbors.
Node A determines that it is partitioned from destination D.

Partition Detection in TORA



On detecting a partition, node A sends a clear (CLR) message that purges all directed links in that partition

TORA

- Improves on the partial link reversal method by detecting partitions and stopping non-productive link reversals
- Paths may not be shortest
- The DAG provides many hosts the ability to send packets to a given destination
  - Beneficial when many hosts want to communicate with a single destination

TORA Design Decision

- TORA performs link reversals
- However, when a link breaks, it looses its direction
- When a link is repaired, it may not be assigned a direction, unless some node has performed a route discovery after the link broke
  - if no one wants to send packets to D anymore, eventually, the DAG for destination D may disappear
- TORA makes effort to maintain the DAG for D only if someone needs route to D
  - Reactive behavior

Proactive Protocols

Link State Routing

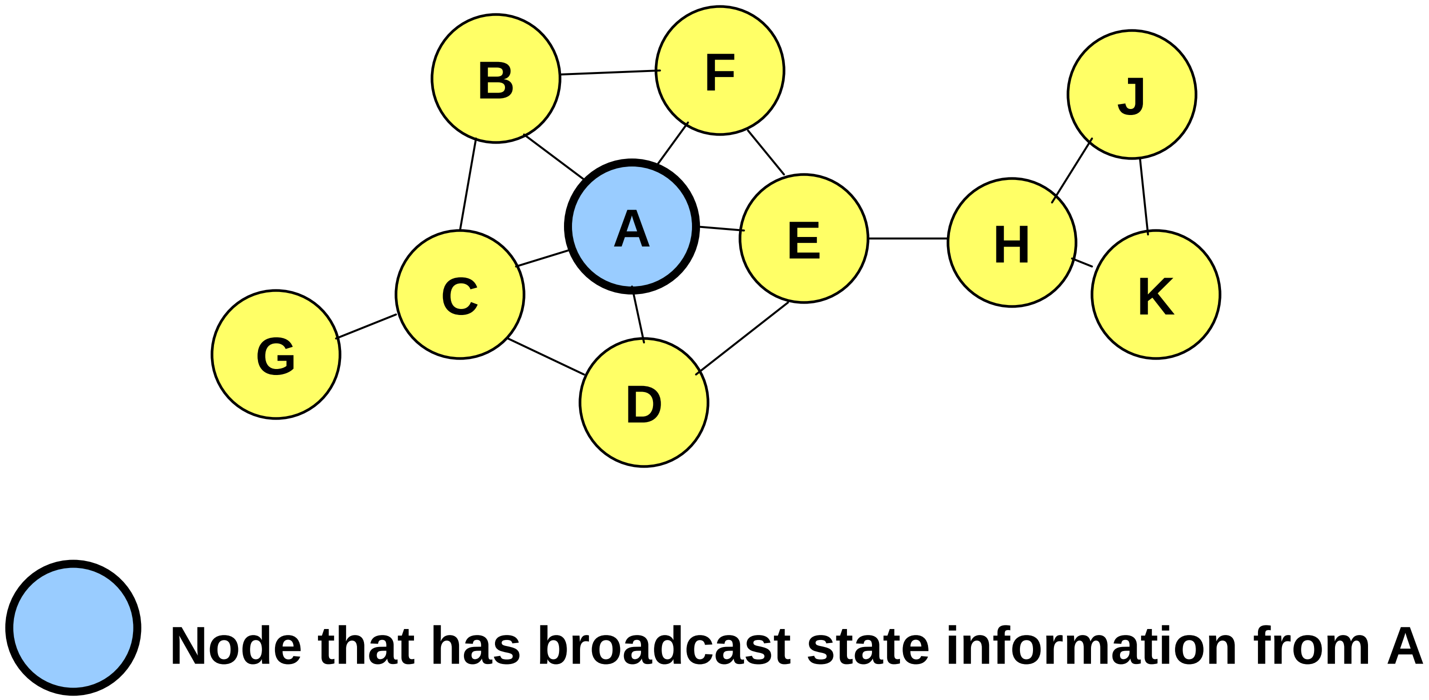
- Each node periodically floods status of its links
- Each node re-broadcasts link state information received from its neighbor
- Each node keeps track of link state information received from other nodes
- Each node uses above information to determine next hop to each destination

Optimized Link State Routing (OLSR)

- The overhead of flooding link state information is reduced by requiring fewer nodes to forward the information
- A broadcast from node X is only forwarded by its *multipoint relays*
- Multipoint relays of node X are its neighbors such that each two-hop neighbor of X is a one-hop neighbor of at least one multipoint relay of X
  - Each node transmits its neighbor list in periodic beacons, so that all nodes can know their 2-hop neighbors, in order to choose the multipoint relays

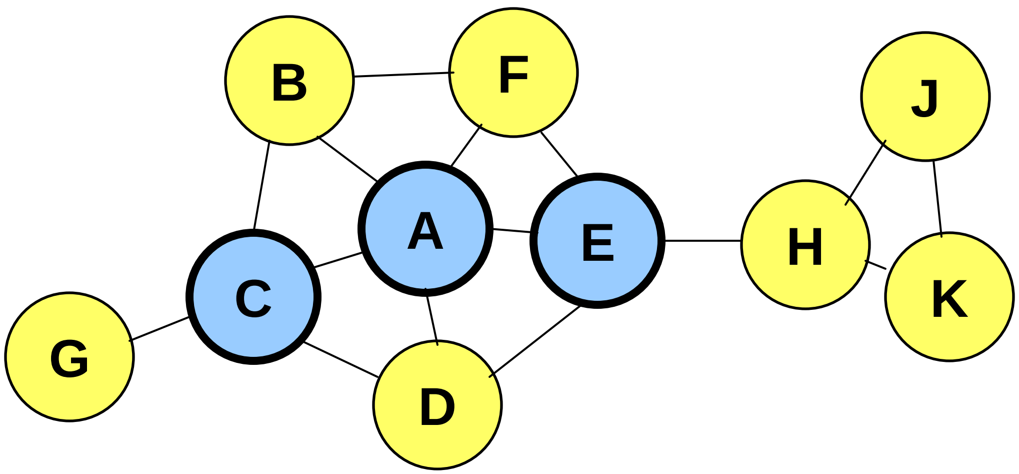
Optimized Link State Routing (OLSR)

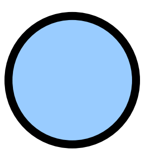
- Nodes C and E are multipoint relays of node A



Optimized Link State Routing (OLSR)

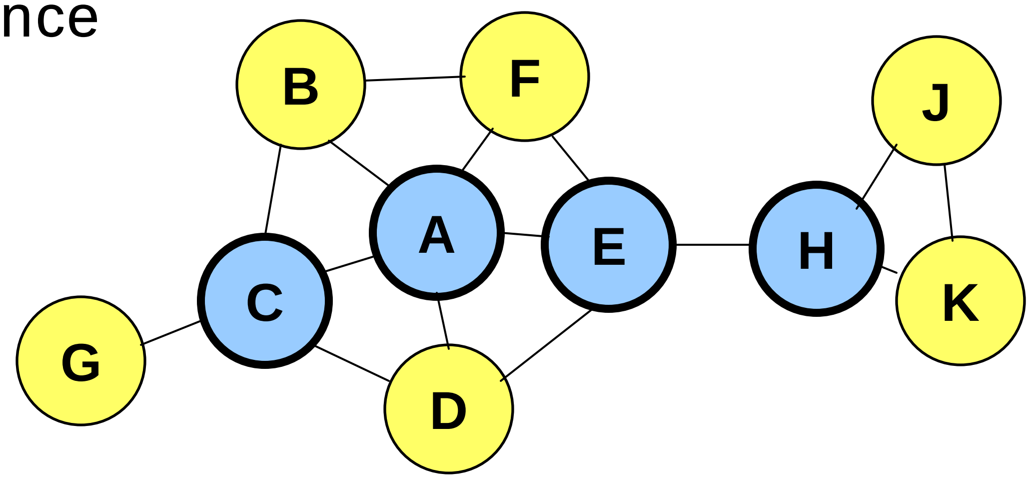
- Nodes C and E forward information received from A

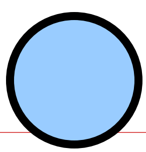




Optimized Link State Routing (OLSR)

- Nodes E and K are multipoint relays for node H
- Node K forwards information received from H
  - E has already forwarded the same information once





Node that has broadcast state information from A

OLSR

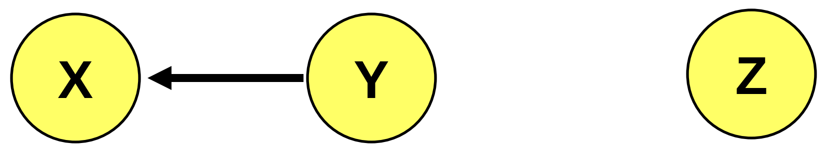
- OLSR floods information through the multipoint relays
- The flooded information itself is for links connecting nodes to respective multipoint relays
- Routes used by OLSR only include multipoint relays as intermediate nodes

Destination-Sequenced Distance-Vector (DSDV)

- Each node maintains a routing table which stores
  - next hop towards each destination
  - a cost metric for the path to each destination
  - a destination sequence number that is created by the destination itself
  - Sequence numbers used to avoid formation of loops
- Each node periodically forwards the routing table to its neighbors
  - Each node increments and appends its sequence number when sending its local routing table
  - This sequence number will be attached to route entries created for this node

Destination-Sequenced Distance-Vector (DSDV)

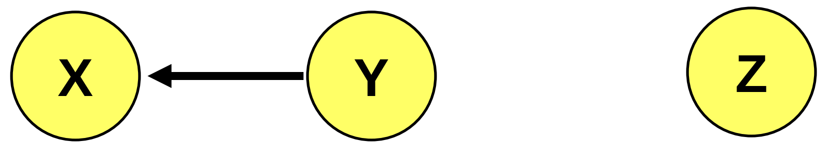
- Assume that node X receives routing information from Y about a route to node Z



- Let S(X) and S(Y) denote the destination sequence number for node Z as stored at node X, and as sent by node Y with its routing table to node X, respectively

Destination-Sequenced Distance-Vector (DSDV)

- Node X takes the following steps:



  - If S(X) > S(Y), then X ignores the routing information received from Y
  - If S(X) = S(Y), and cost of going through Y is smaller than the route known to X, then X sets Y as the next hop to Z
  - If S(X) < S(Y), then X sets Y as the next hop to Z, and S(X) is updated to equal S(Y)

Hybrid Protocols

Zone Routing Protocol (ZRP)

Zone routing protocol combines

- Proactive protocol: which pro-actively updates network state and maintains route regardless of whether any data traffic exists or not
- Reactive protocol: which only determines route to a destination if there is some data to be sent to the destination

ZRP

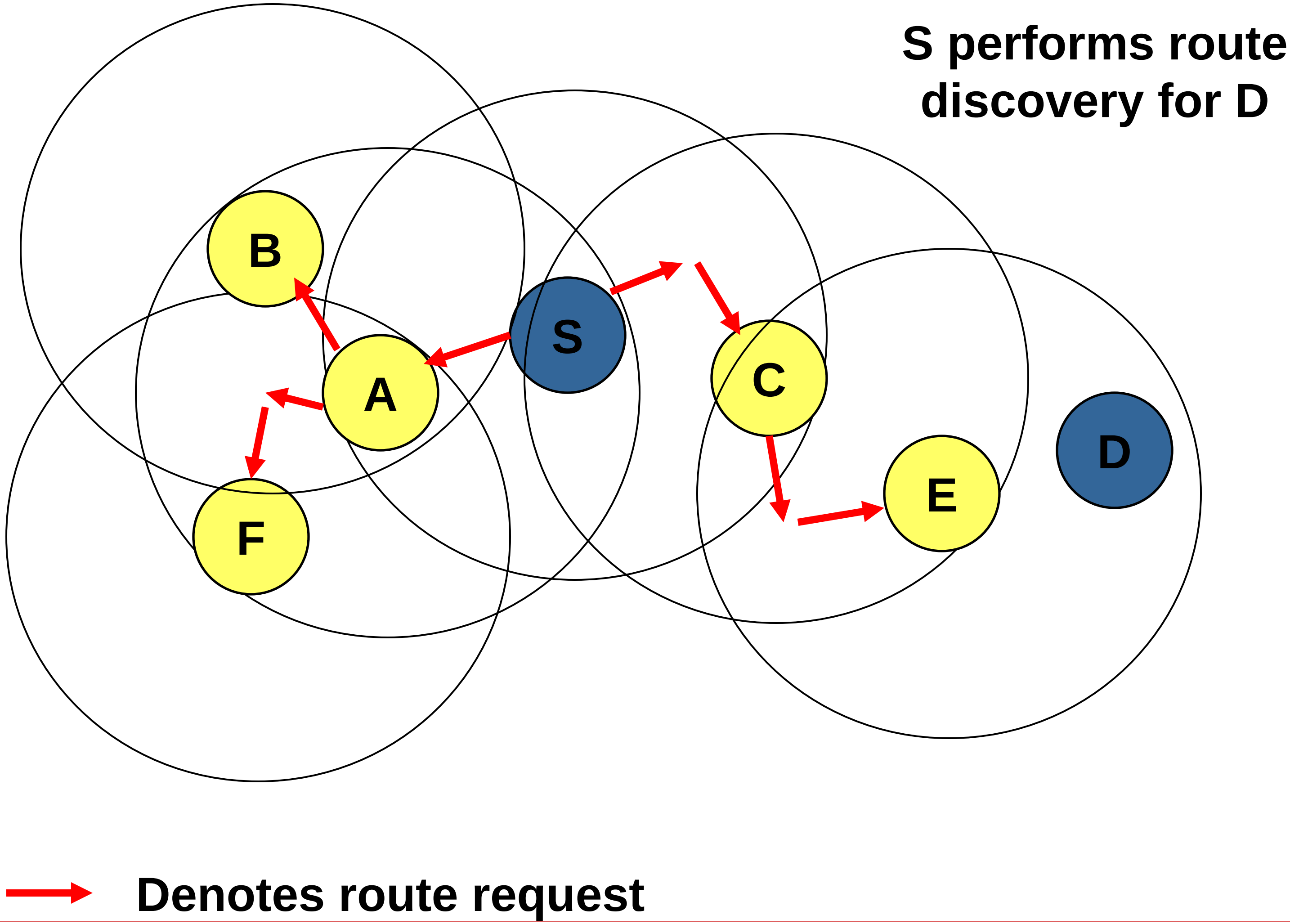
- All nodes within hop distance at most *d* from a node X are said to be in the routing zone of node X

- All nodes at hop distance exactly *d* are said to be peripheral nodes of node X's routing zone

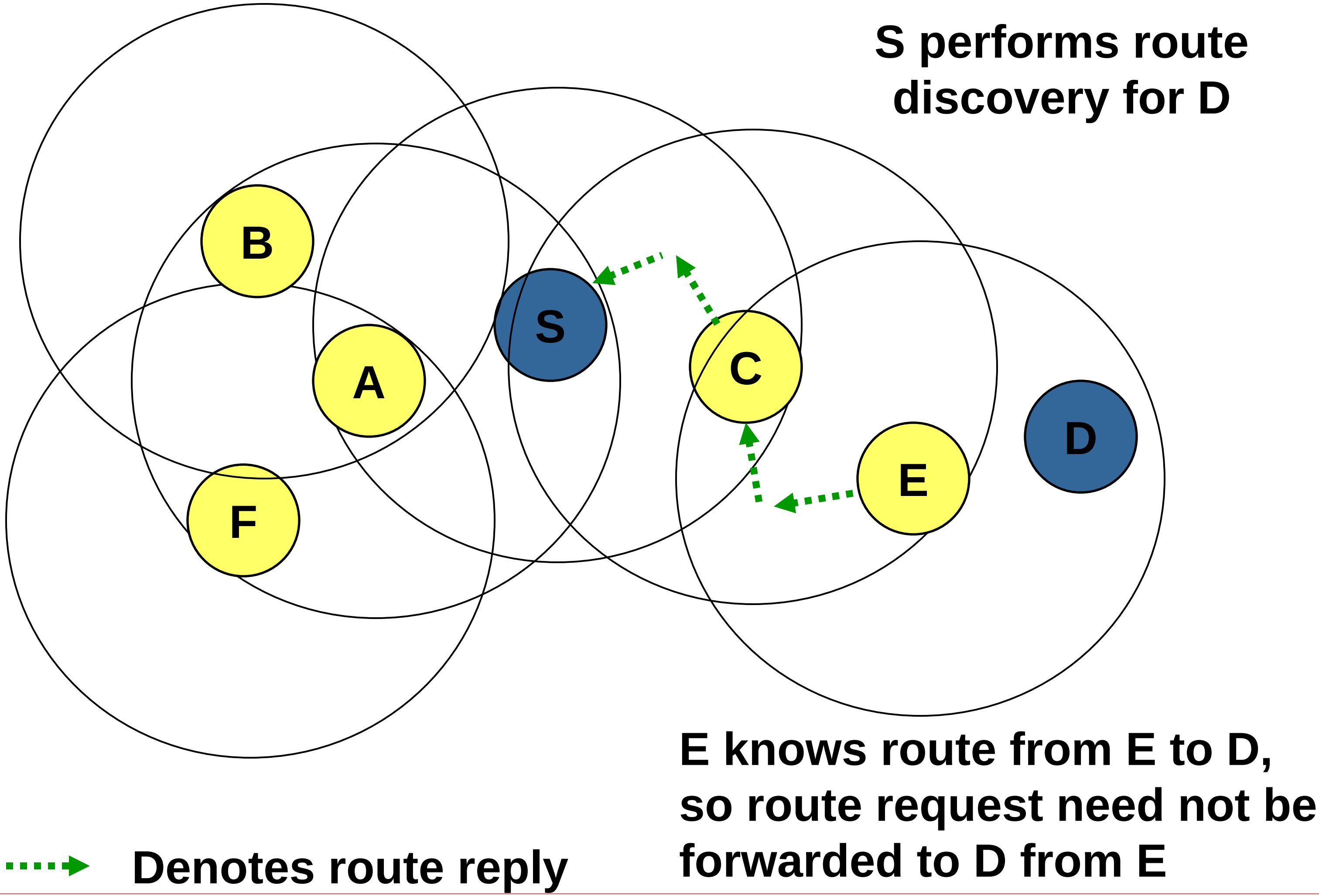
ZRP

- Intra-zone routing: Pro-actively maintain state information for links within a short distance from any given node
  - Routes to nodes within short distance are thus maintained proactively (using, say, link state or distance vector protocol)
- Inter-zone routing: Use a route discovery protocol for determining routes to far away nodes. Route discovery is similar to DSR with the exception that route requests are propagated via peripheral nodes.

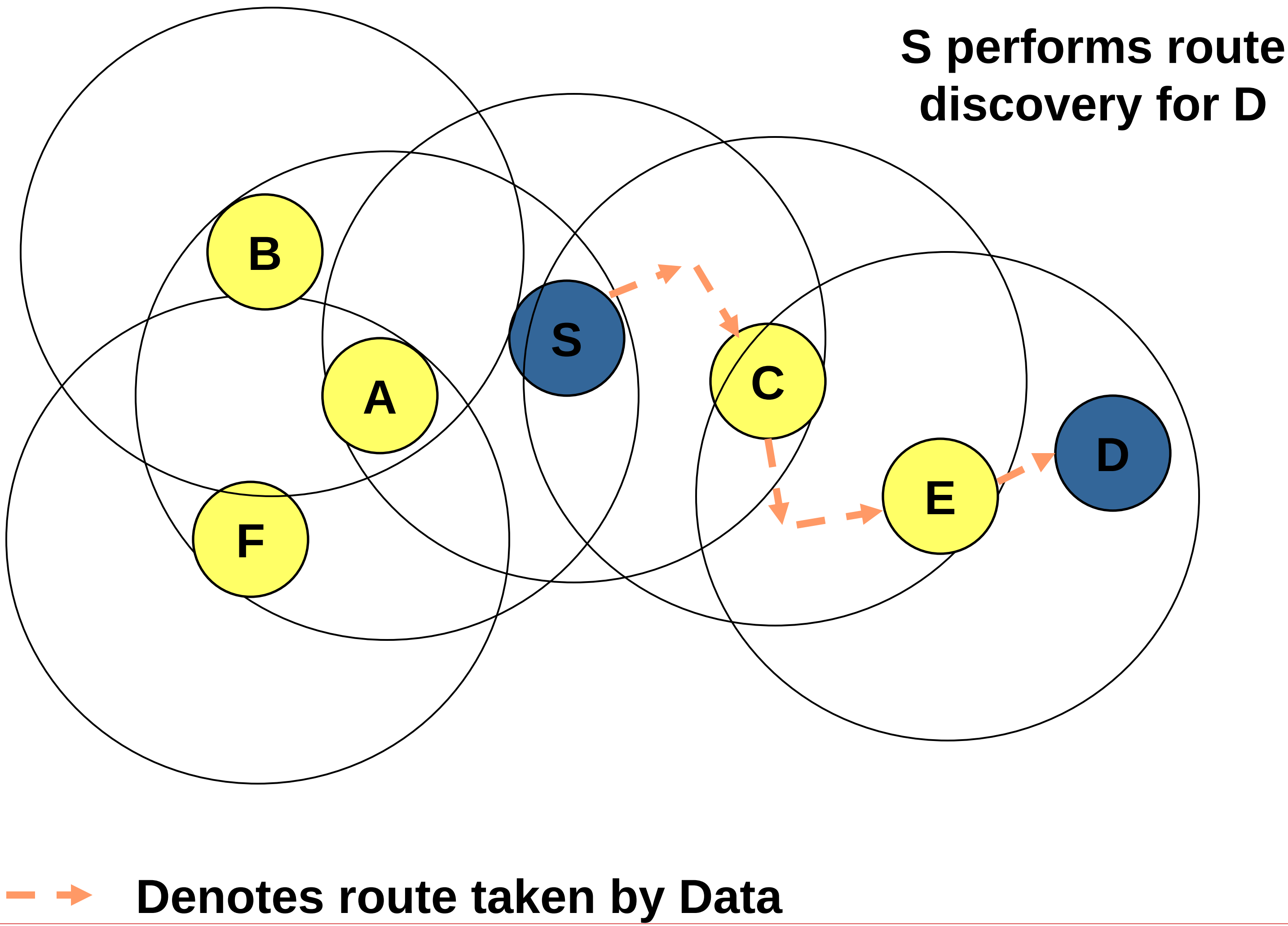
ZRP: Example with Zone Radius = *d* = 2



ZRP: Example with *d = 2*



ZRP: Example with *d = 2*

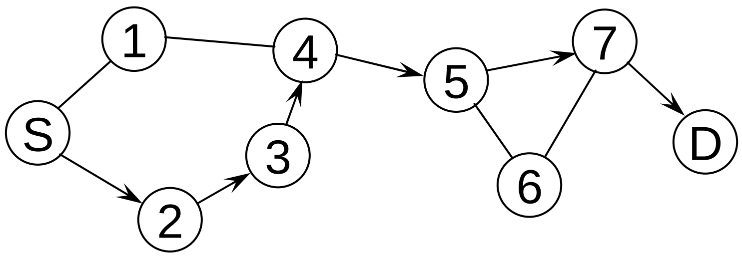


Geographic routing

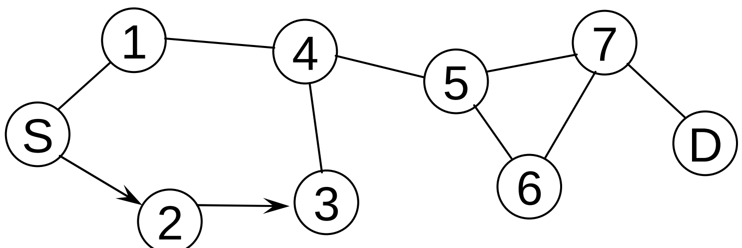
Geographic Distance Routing (GEDIR)

- Rather than maintaining routing tables and discovering paths, one can also use the geographic location of nodes
  - Requires that each node knows it own location (e.g., using GPS)
  - Requires knowledge of all neighbor locations
- It is based on sending the packet to the neighbor that is closest to the destination
  - Works only if nodes are located densely
  - Obstacles and low node density may lead to routing failures

GEDIR – Example



Regular Operation
(not necessarily minimum hop)

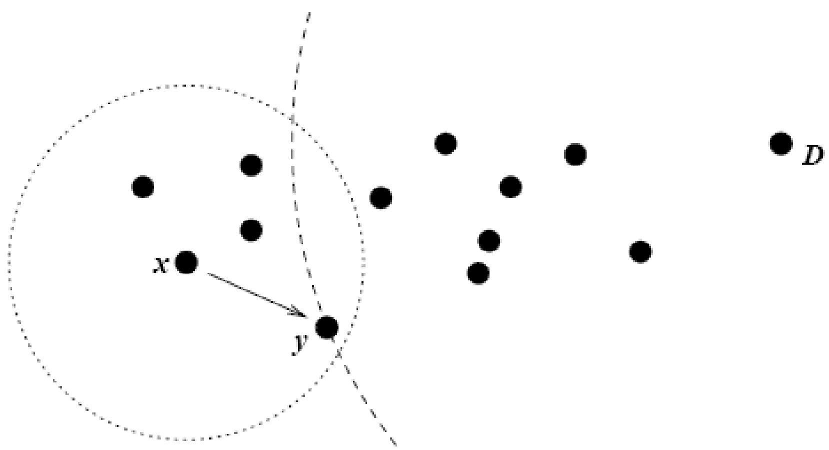


Routing fails because 3 has no neighbors closer to D than itself

- To overcome the problem of not finding closer neighbors, expanded local search algorithms are also proposed
  - When stuck, broadcast a path discovery request with small TTL, use discovered path for forwarding data

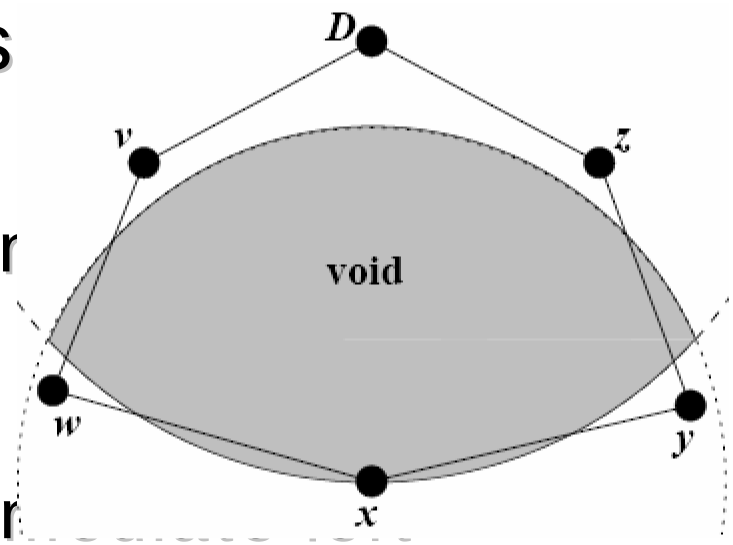
Greedy Perimeter Stateless Routing (GPSR)

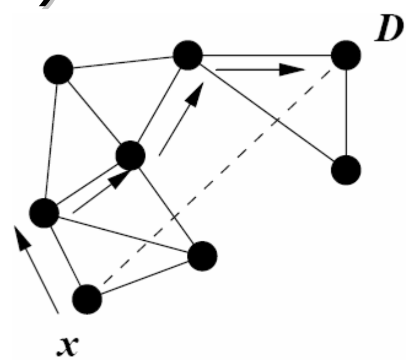
- Another geographic routing algorithm
- Like GEDIR, it is also based on greedy forwarding
  - Maintain a list of neighbors with their locations
  - Send the packet to the node nearest to the destination (Most Forward within Radiu – MFR)
  - Avoid routing loops

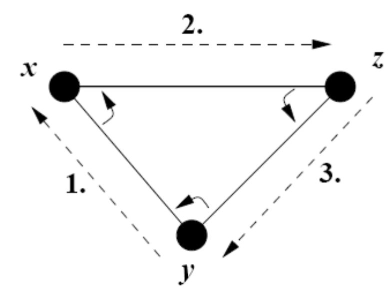


GPSR

- Avoiding routing gaps
  - Use perimeter routing
  - Mark the line connectir the intermediate node with destination
  - Take the hop to its imr (counter-clockwise)
  - Right hand rule!

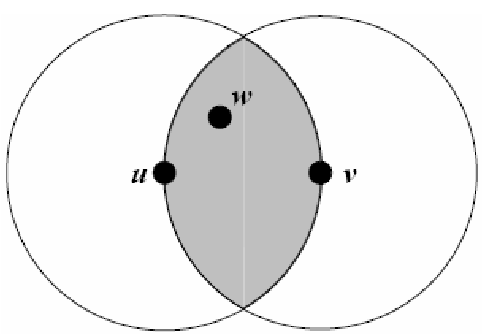




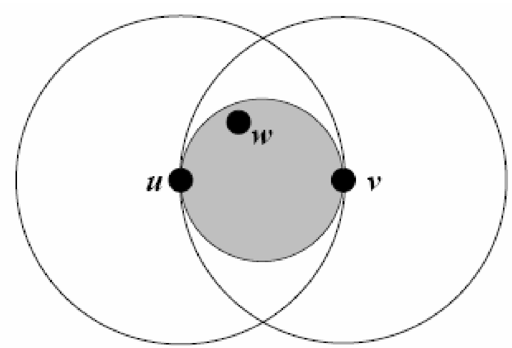


GPSR

- Perimeter routing requires that graphs are planar
  - No edge in the graph crosses another edge
- Planarization algorithms



Relative Neighbor Graph



Gabriel Graph

In both cases, eliminate link uv