Mobile Ad Hoc Networks: Routing

Mobile Ad Hoc Networks (MANET)

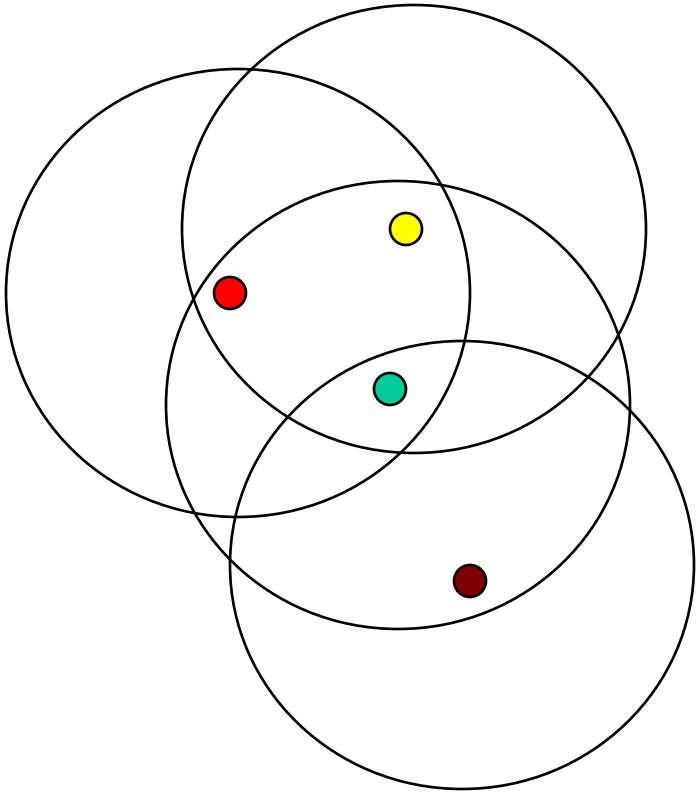
Introduction and Generalities

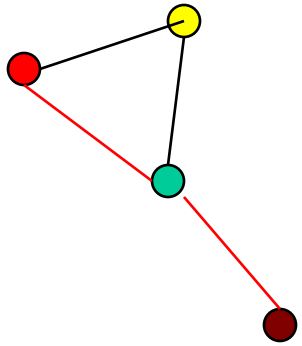
Mobile Ad Hoc Networks

- Formed by wireless hosts which may be mobile
- Without (necessarily) using a pre-existing infrastructure
- Routes between nodes may potentially contain multiple hops

Mobile Ad Hoc Networks

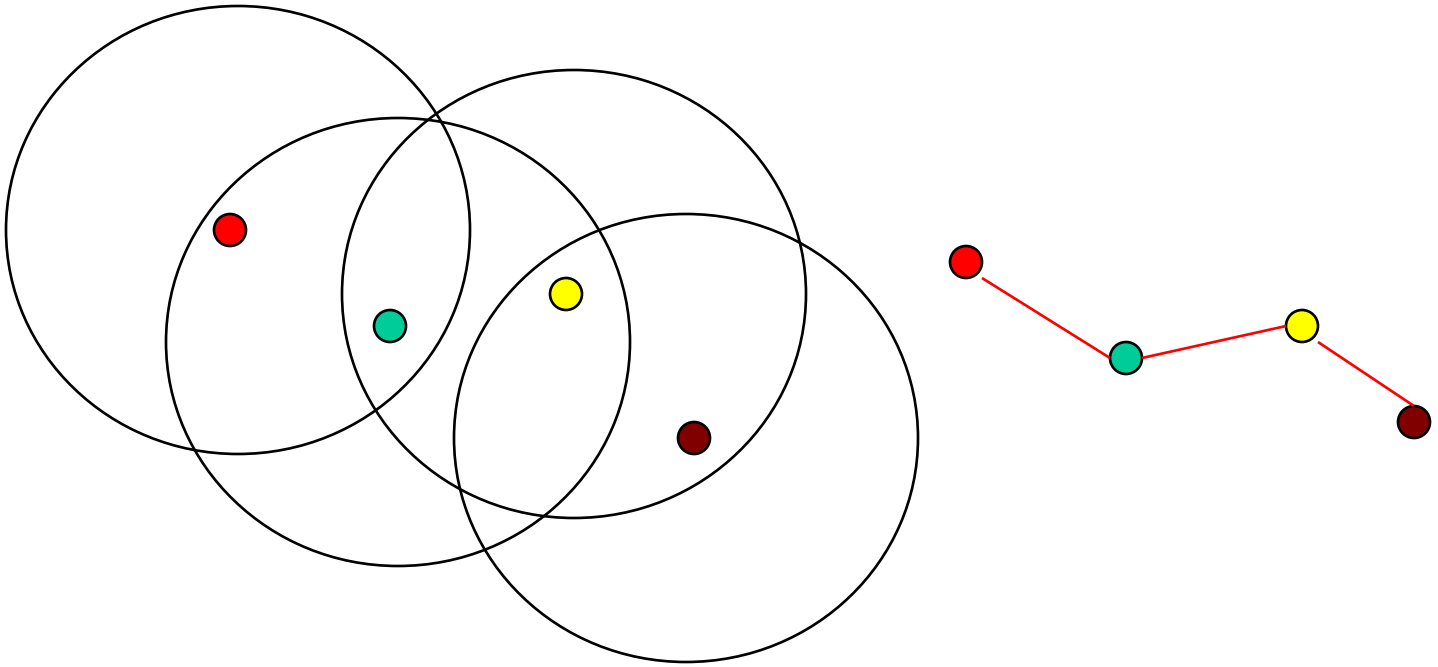
- May need to traverse multiple links to reach a destination





Mobile Ad Hoc Networks (MANET)

- Mobility causes route changes



Why Ad Hoc Networks ?

- Ease of deployment
- Speed of deployment
- Decreased dependence on infrastructure

Many Applications

- Personal area networking
  - cell phone, laptop, ear phone, wrist watch
- Military environments
  - soldiers, tanks, planes
- Civilian environments
  - taxi cab network
  - meeting rooms
  - sports stadiums
  - boats, small aircraft
- Emergency operations
  - search-and-rescue
  - policing and fire fighting

Many Variations

- Fully Symmetric Environment
  - all nodes have identical capabilities and responsibilities
- Asymmetric Capabilities
  - transmission ranges and radios may differ
  - battery life at different nodes may differ
  - processing capacity may be different at different nodes
  - speed of movement
- Asymmetric Responsibilities
  - only some nodes may route packets
  - some nodes may act as leaders of nearby nodes (e.g., cluster head)

Many Variations

- Traffic characteristics may differ in different ad hoc networks
  - bit rate
  - timeliness constraints
  - reliability requirements
  - unicast / multicast / geocast
  - host-based addressing / content-based addressing / capability-based addressing
- May co-exist (and co-operate) with an infrastructure-based network

Many Variations

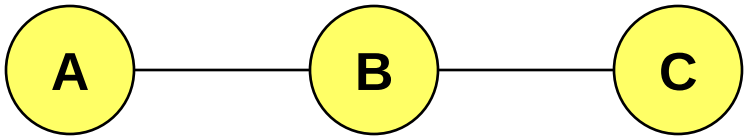
- **Mobility patterns may be different**
  - people sitting at an airport lounge
  - Taxi cabs
  - kids playing
  - military movements
  - personal area network

- **Mobility characteristics**
  - speed
  - predictability
    - direction of movement
    - pattern of movement
  - uniformity (or lack thereof) of mobility characteristics among different nodes

Challenges

- Limited wireless transmission range
- Broadcast nature of the wireless medium
  - Hidden terminal problem
- Packet losses due to transmission errors
- Mobility-induced route changes
- Mobility-induced packet losses
- Battery constraints
- Potentially frequent network partitions
- Need to scale to large networks
- Ease of snooping on wireless transmissions (security hazard)

Hidden Terminal Problem



Nodes A and C cannot hear each other

Transmissions by nodes A and C can collide at node B

Nodes A and C are hidden from each other

Assumption

- Unless stated otherwise, fully symmetric environment is assumed implicitly
  - all nodes have identical capabilities and responsibilities

Unicast Routing
in
Mobile Ad Hoc Networks

Why is Routing in MANET different ?

- Host mobility
  - link failure/repair due to mobility may have different characteristics than those due to other causes
- Rate of link failure/repair may be high when nodes move fast
- New performance criteria may be used
  - route stability despite mobility
  - energy consumption

Unicast Routing Protocols

- Many protocols have been proposed
- Some have been invented specifically for MANET
- Others are adapted from previously proposed protocols for wired networks
- No single protocol works well in all environments
  - some attempts made to develop adaptive protocols

Routing Protocols

- **Proactive protocols** (Maintain updated routes)
  - Determine routes independent of traffic pattern
  - Traditional link-state and distance-vector routing protocols are proactive
  - e.g.,variations of Dist. Vector (Destination Sequence Distance Vector, DSDV) and Link state (Optimized Link State Routing, OLSR)
- **Reactive protocols** (discover routes on-demand)
  - Maintain routes only if needed
  - e.g., Source Routing (Dynamic Source Routing, DSR), Table driven (AODV, ABR, TORA etc.)
- **Hybrid protocols**
  - Combination of reactive and proactive routing
  - e.g., ZRP
- **Location-aware Routing Protocols**
  - Location information helps in optimizing routing
  - e.g., LAR

Trade-Off

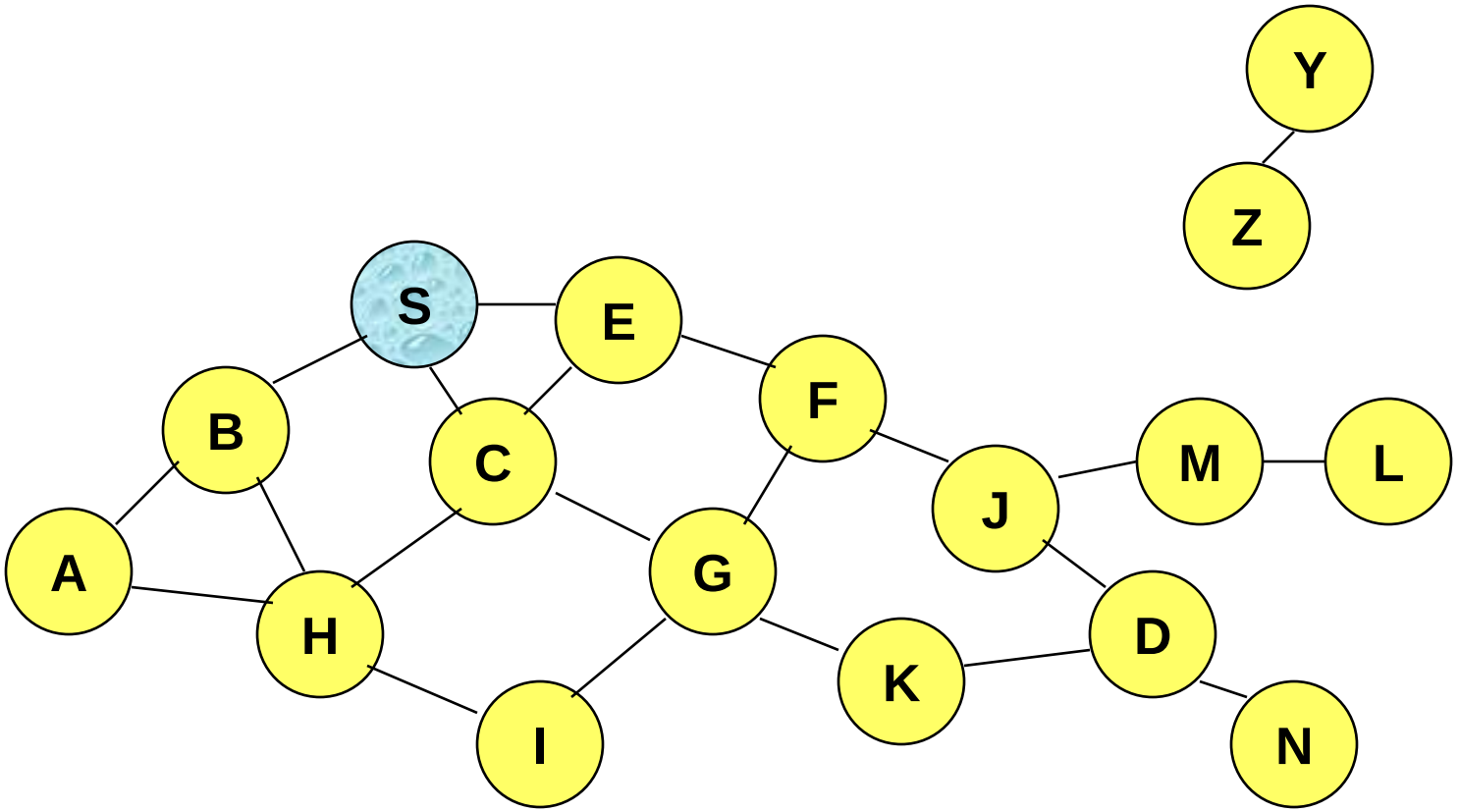
- **Latency of route discovery**
  - Proactive protocols may have lower latency since routes are maintained at all times
  - Reactive protocols may have higher latency because a route from X to Y will be found only when X attempts to send to Y
- **Overhead of route discovery/maintenance**
  - Reactive 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
- **QoS guarantees**
  - In proactive, QoS guarantees related to connection setup, latency, and other real-time requirements can be ensured
  - Quality of a path is not known apriori in reactive.
- **Which approach achieves a better trade-off depends on the traffic and mobility patterns**

Overview of Unicast Routing Protocols

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

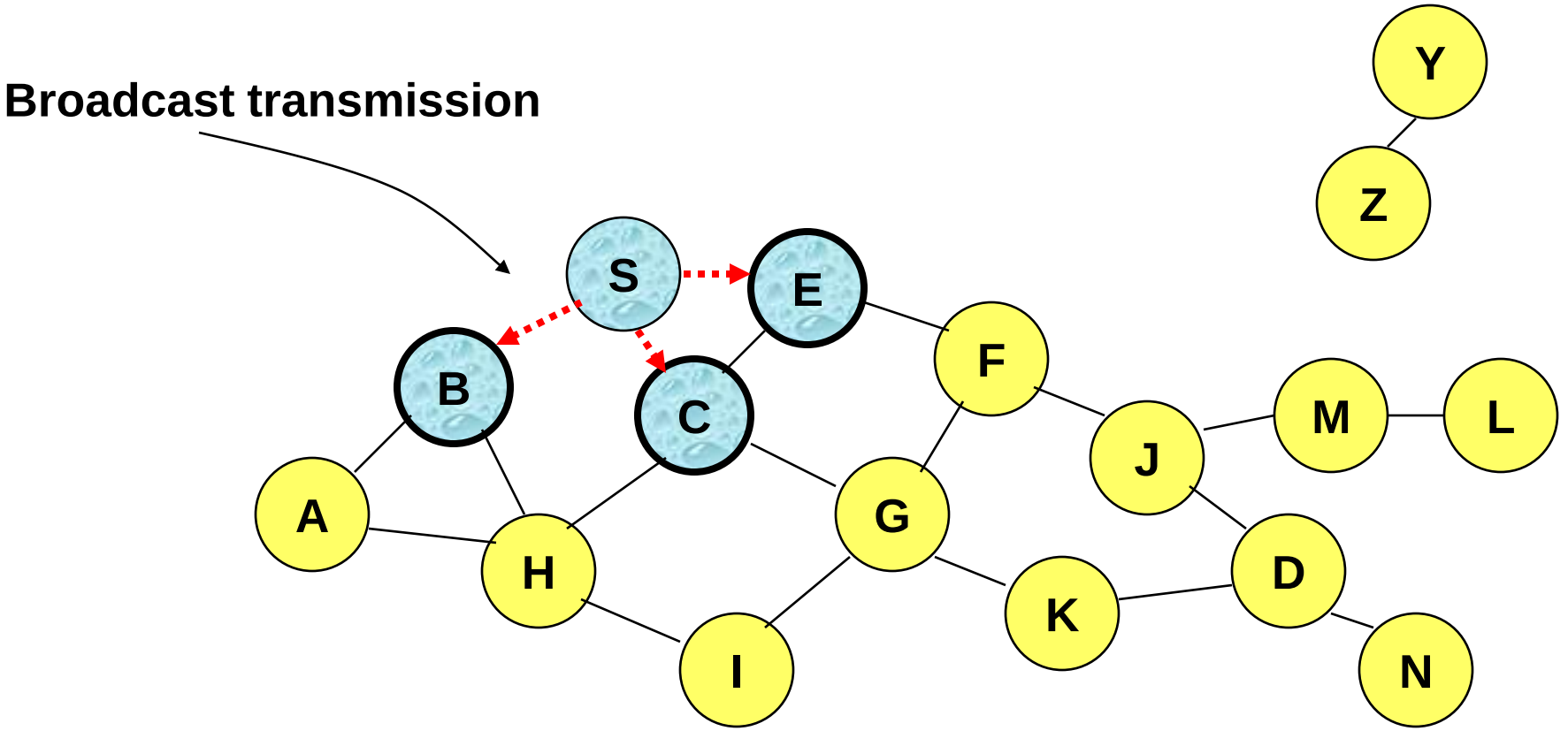
Flooding for Data Delivery



Represents a node that has received packet P

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

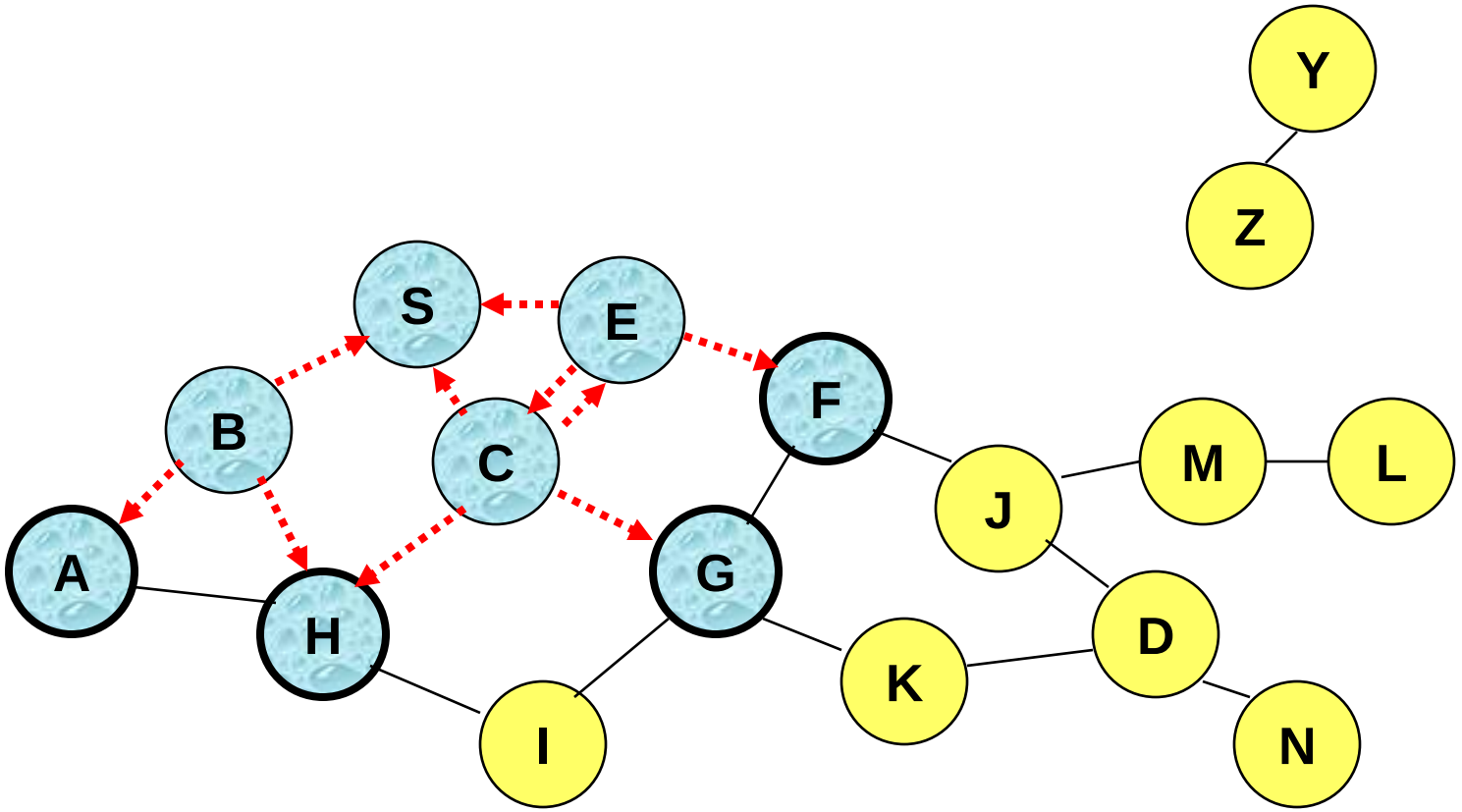
Flooding for Data Delivery



Represents a node that receives packet P for the first time

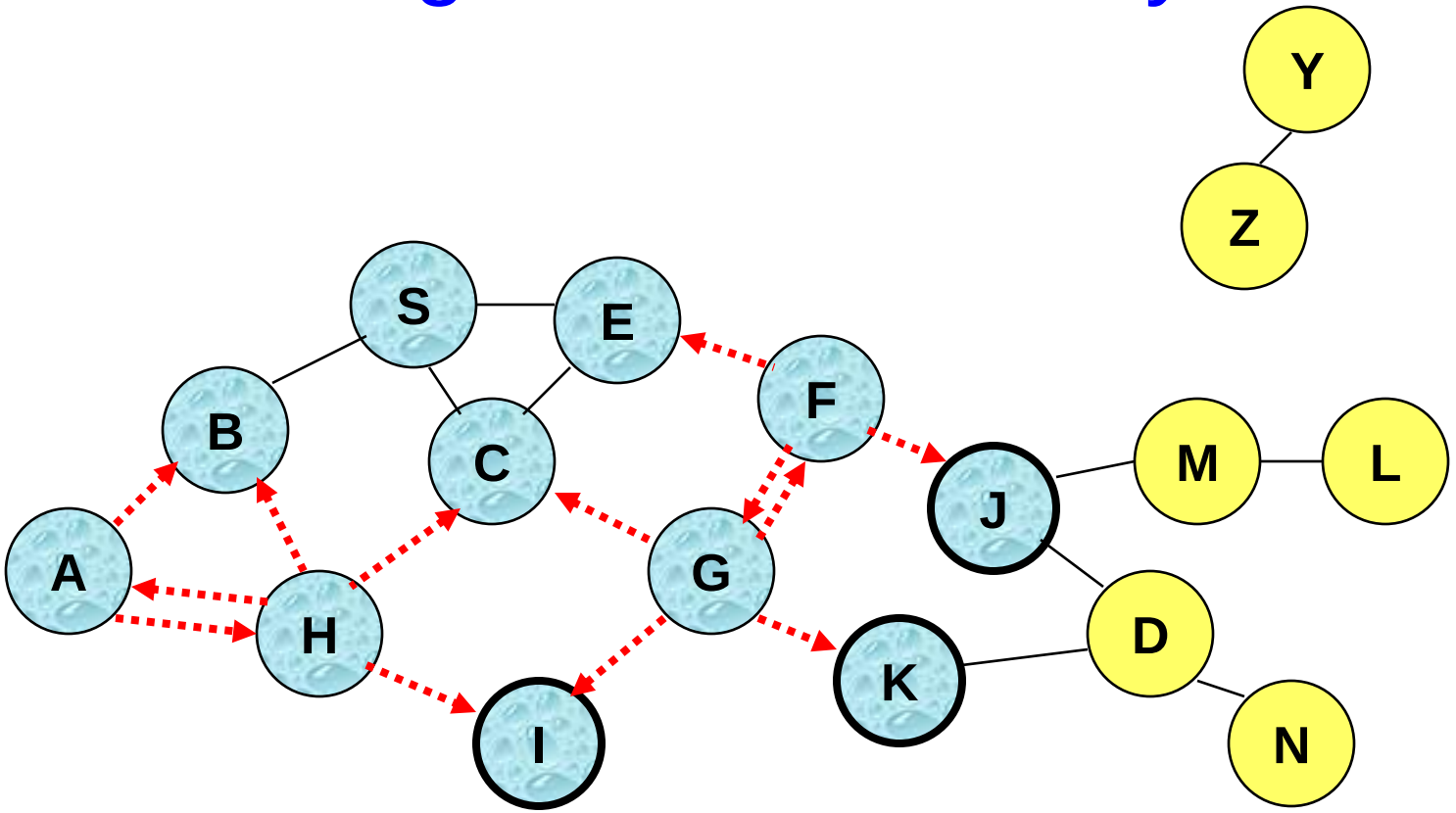
Represents transmission of packet P

Flooding for Data Delivery



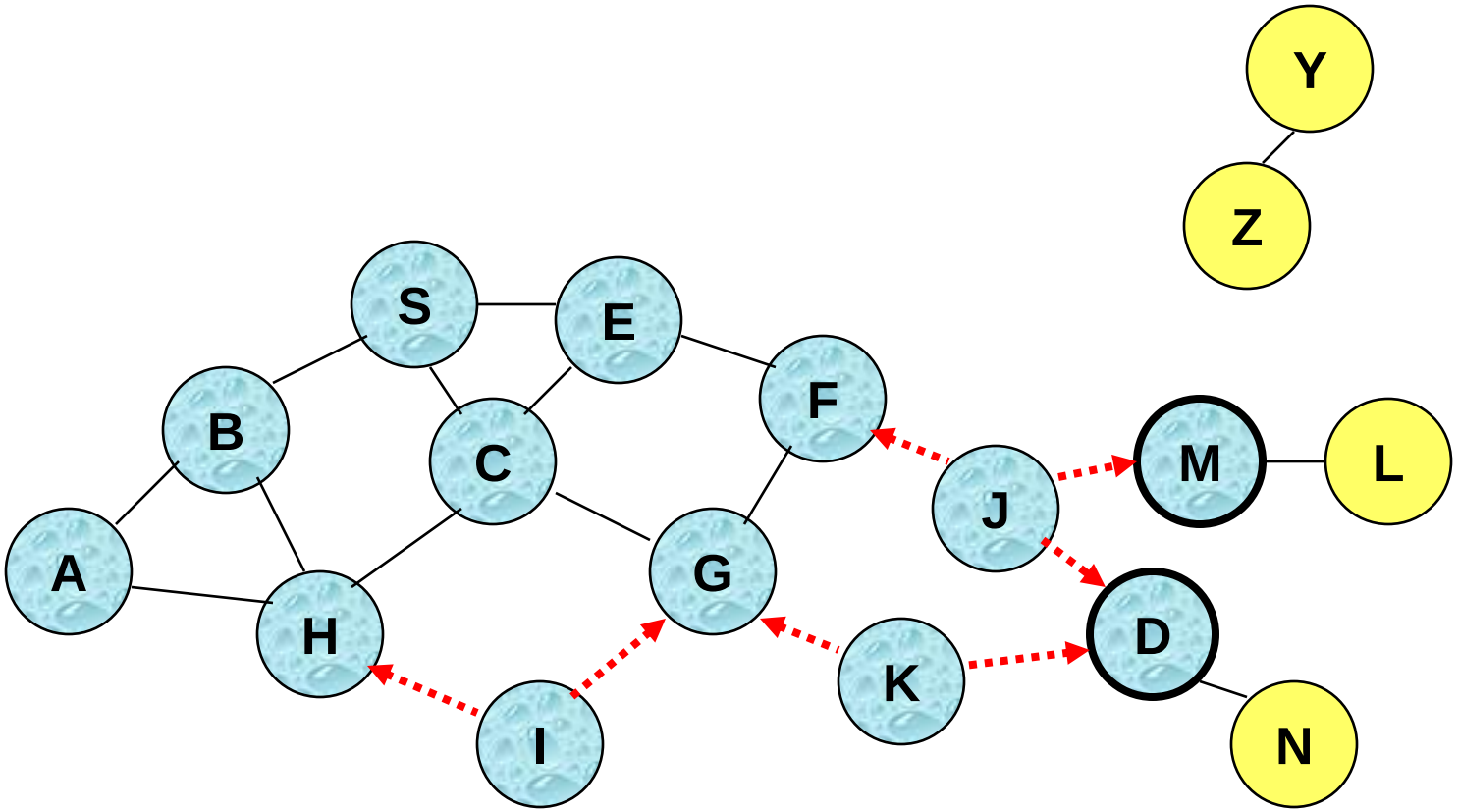
- **Node H receives packet P from two neighbors:**
  potential for collision

Flooding for Data Delivery



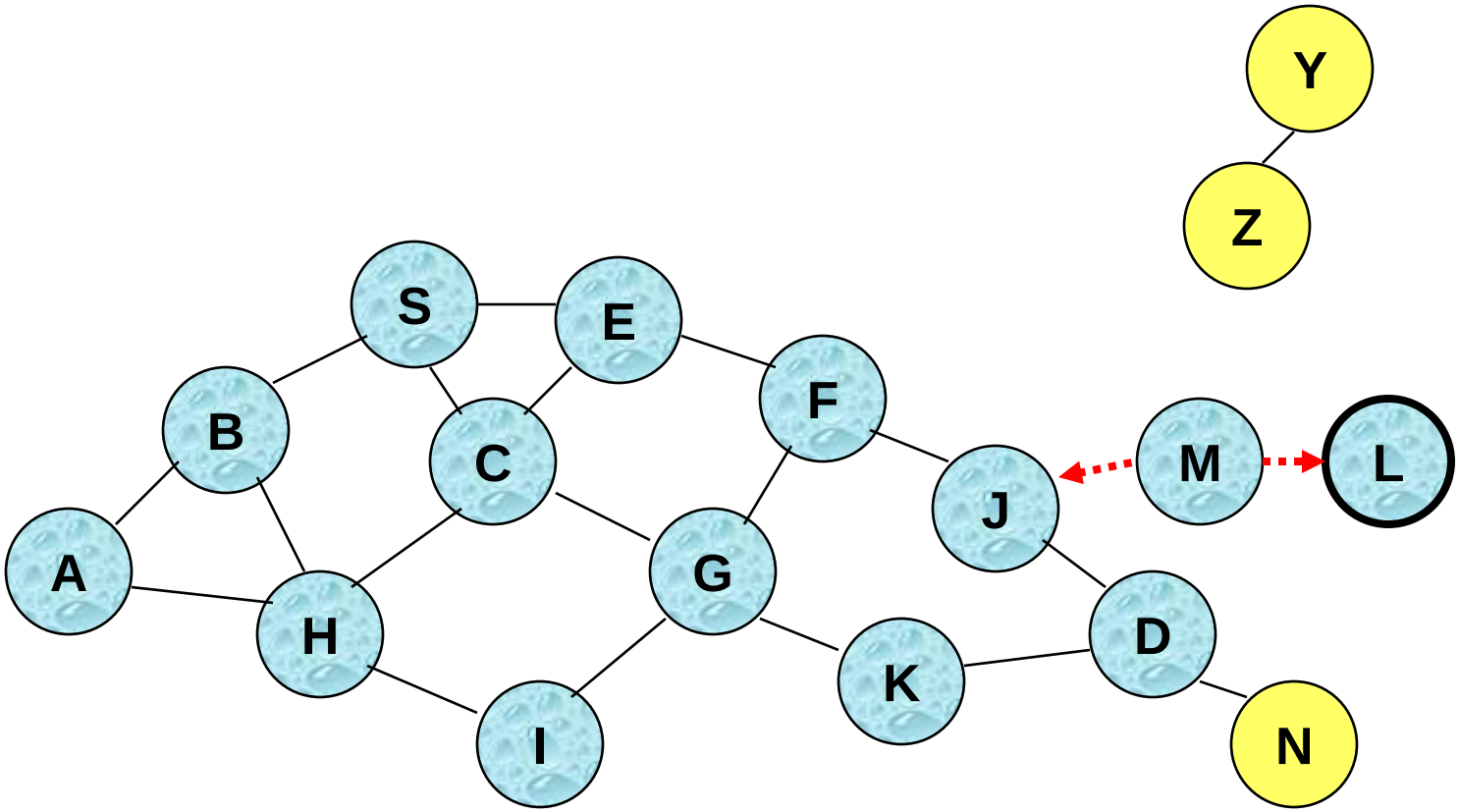
- **Node C receives packet P from G and H, but does not forward it again, because node C has already forwarded packet P once**

Flooding for Data Delivery



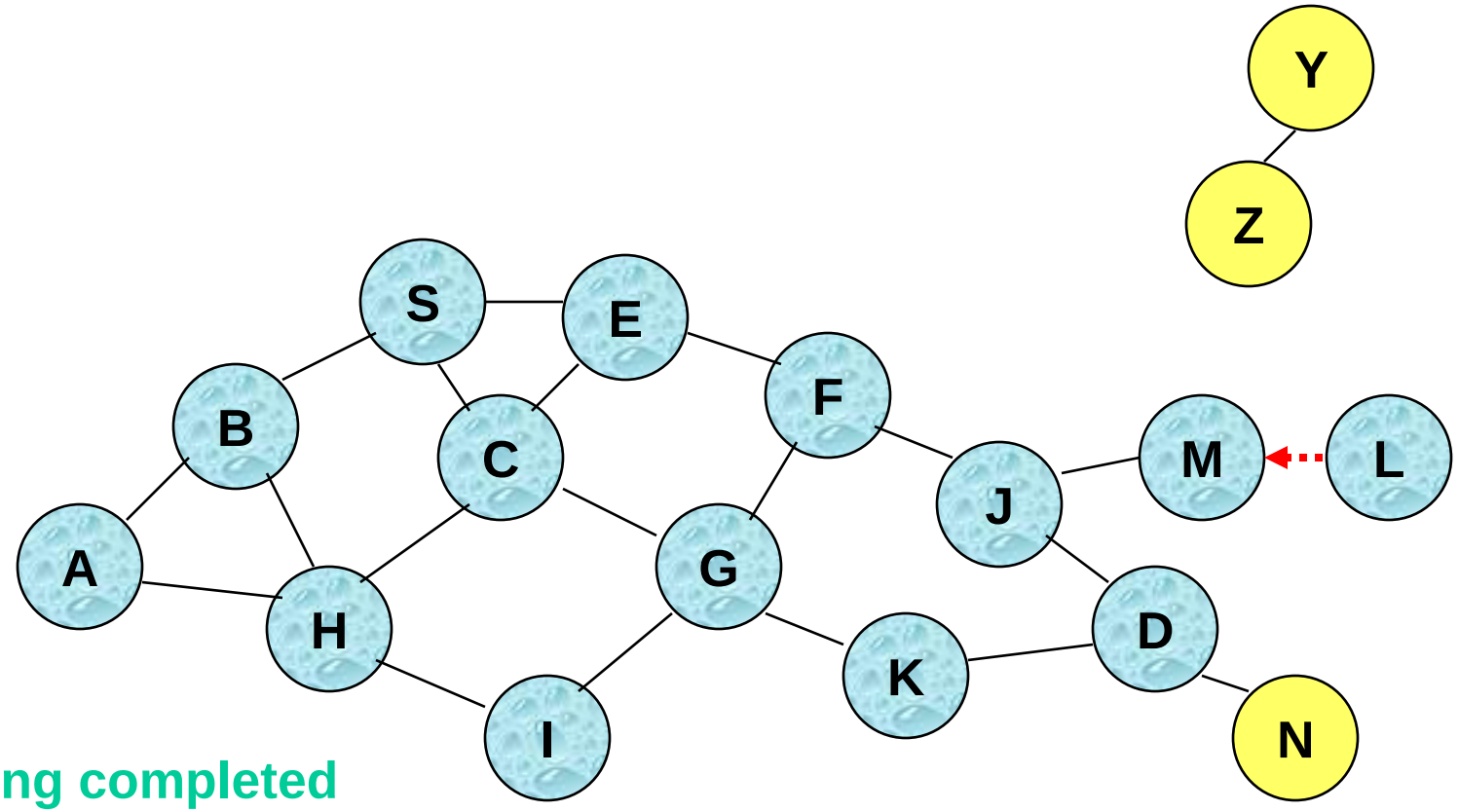
- **Nodes J and K both broadcast packet P to node D**
- **Since nodes J and K are hidden from each other, their transmissions may collide**
  - **=> Packet P may not be delivered to node D at all, despite the use of flooding**

Flooding for Data Delivery



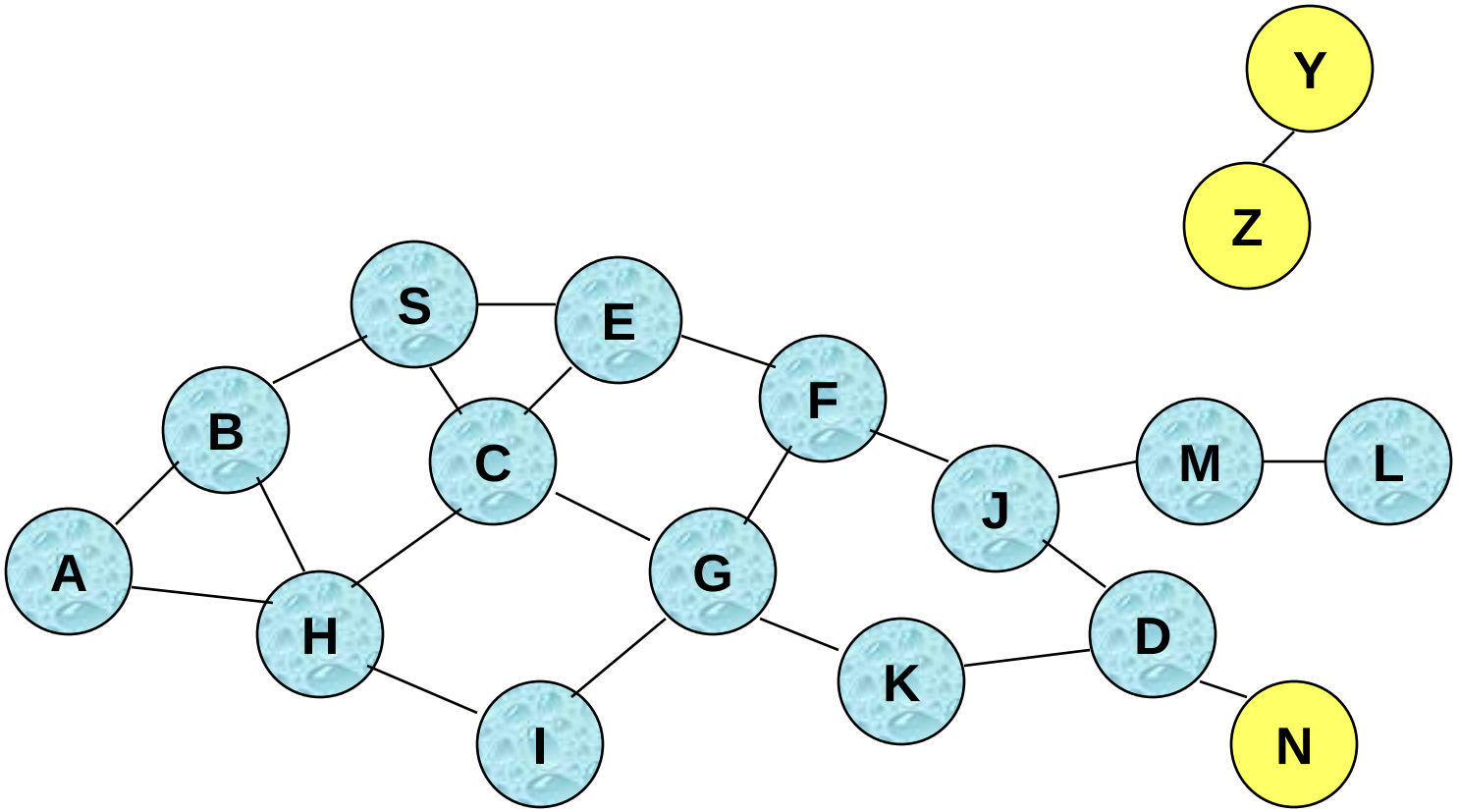
- **Node D does not forward packet P, because node D is the intended destination of packet P**

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, though only the dest has to receive)**

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
- Some improvements: Use of sequence nos to prevent repetitive broadcast, Restricted flooding (floods the network incrementally, increasing the radius of the flood at every successive attempts)

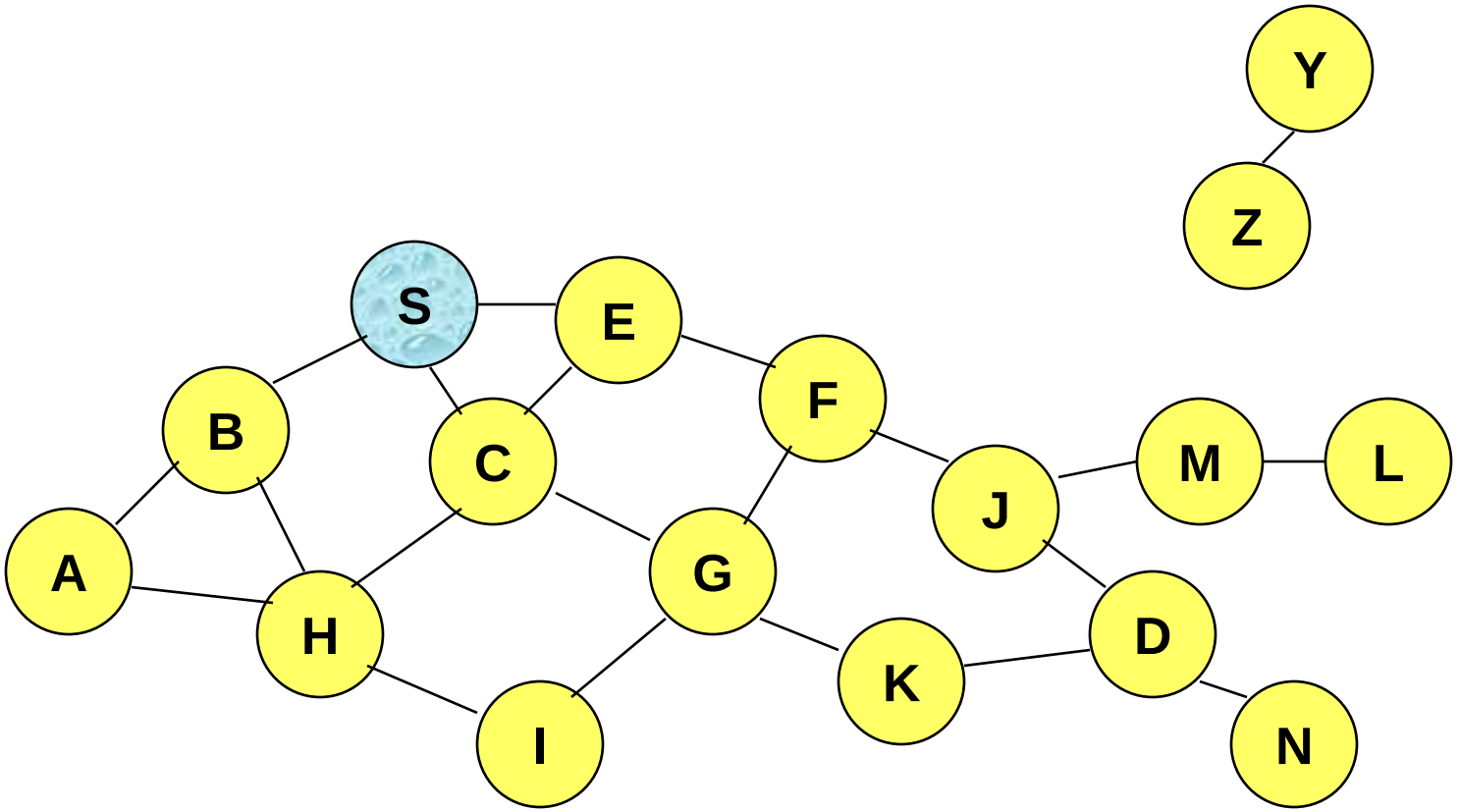
Flooding of Control Packets

- 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

Dynamic Source Routing (DSR) [Johnson96]

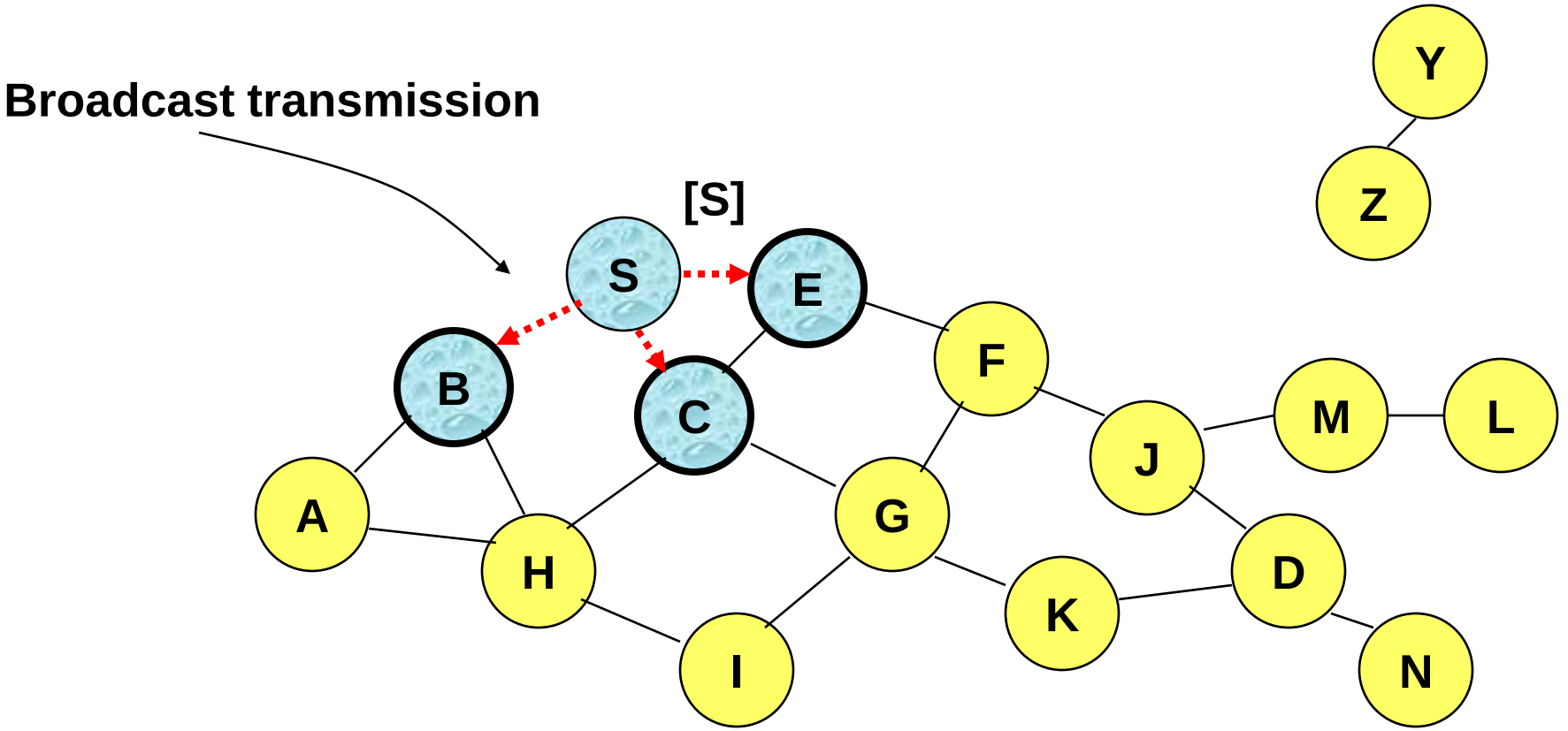
- When node S wants to send a packet to node D, but does not know a route to D, node S initiates a **route discovery**
- Source node S floods **Route Request (RREQ)**
- Each node **appends its own identifier** when forwarding RREQ

Route Discovery in DSR



Represents a node that has received RREQ for D from S

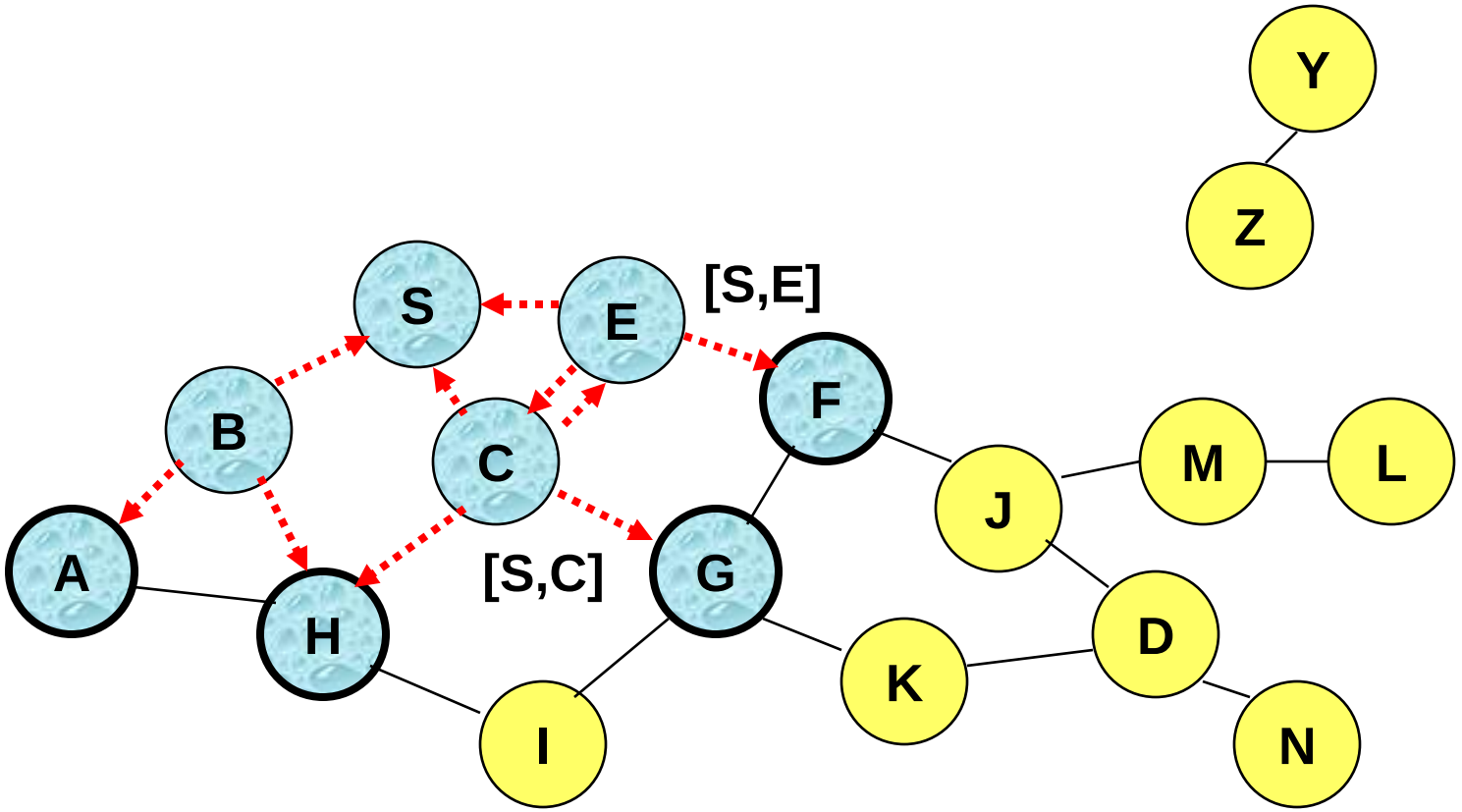
Route Discovery in DSR



Represents transmission of RREQ

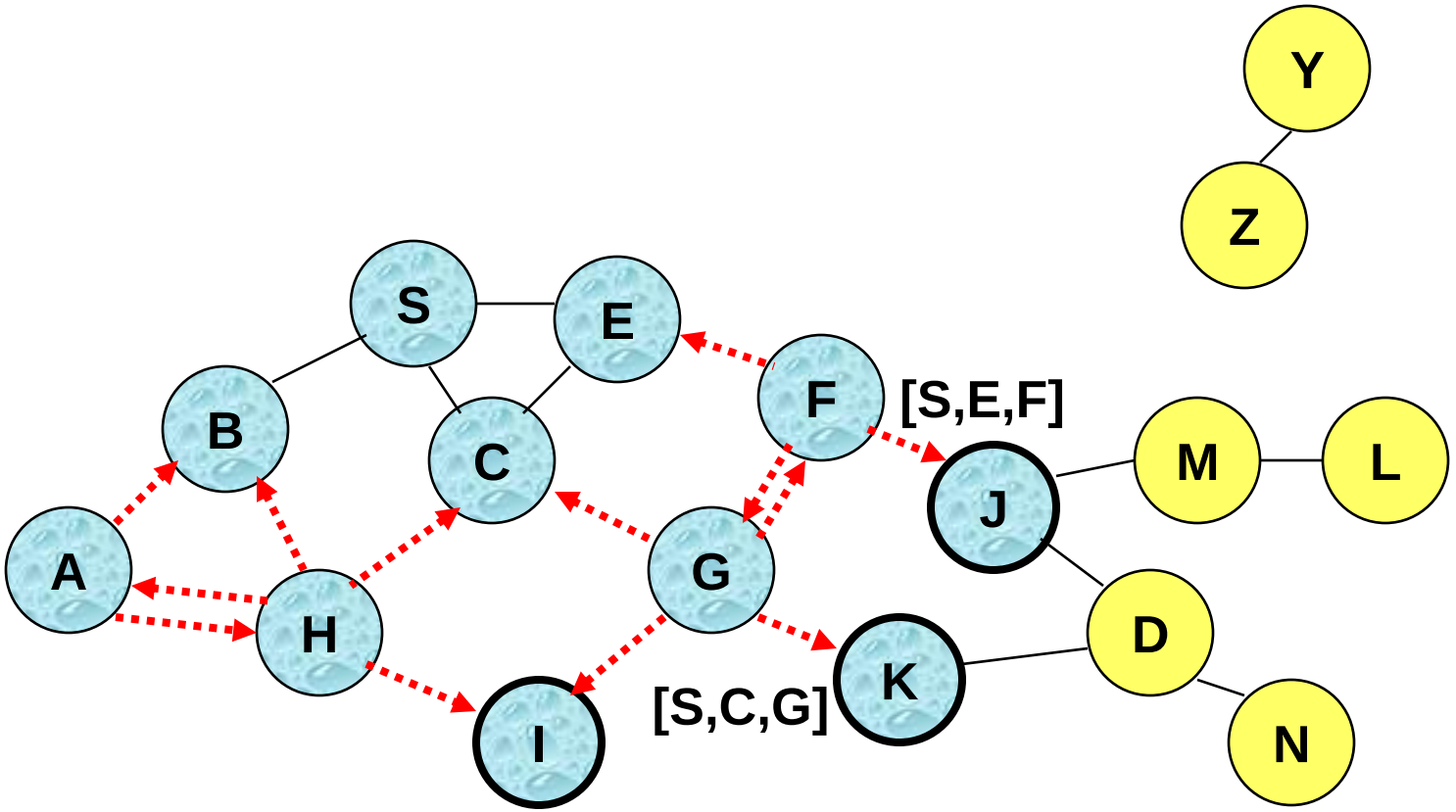
[X,Y] Represents list of identifiers appended to RREQ

Route Discovery in DSR



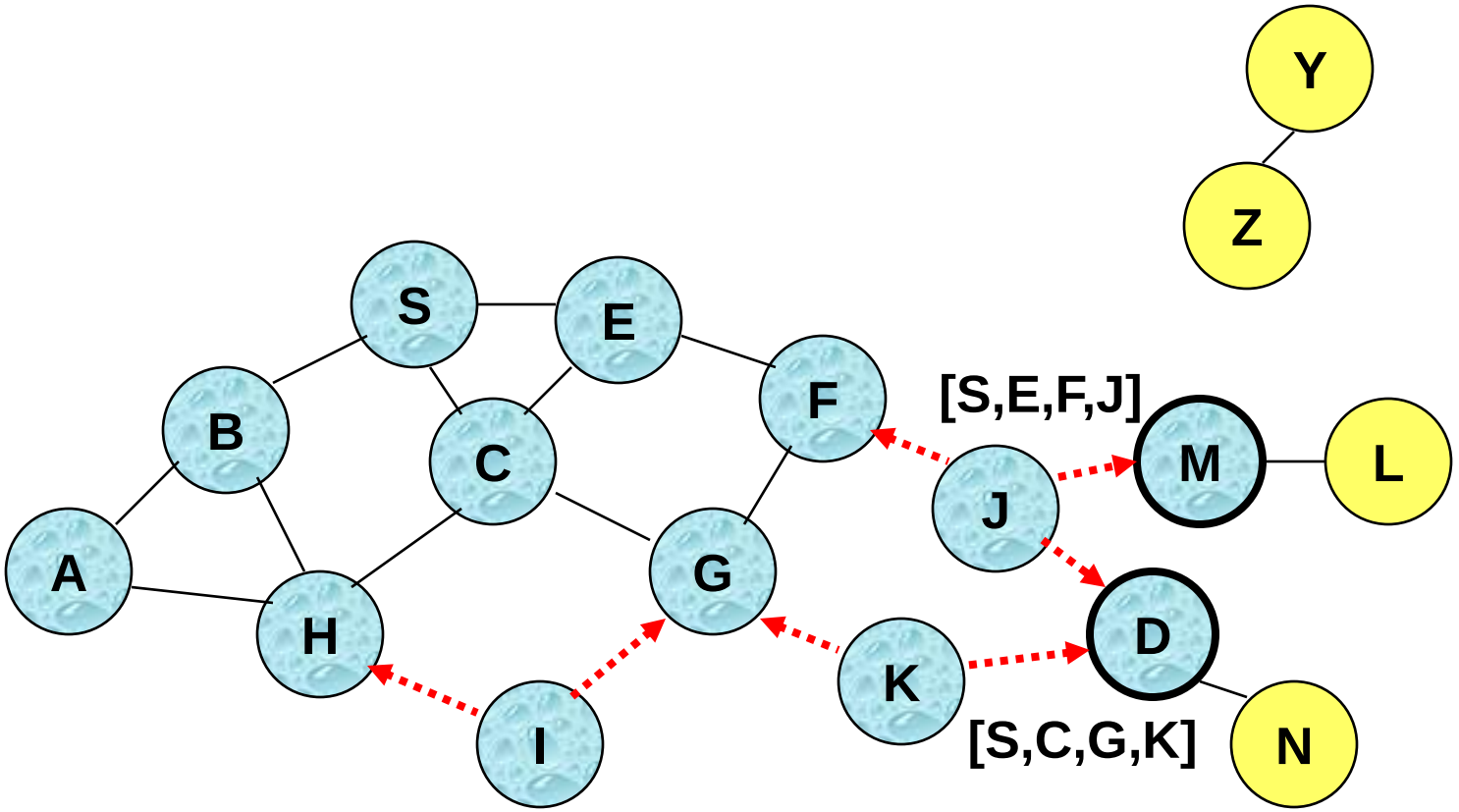
- **Node H receives packet RREQ from two neighbors:**
  potential for collision

Route Discovery in DSR



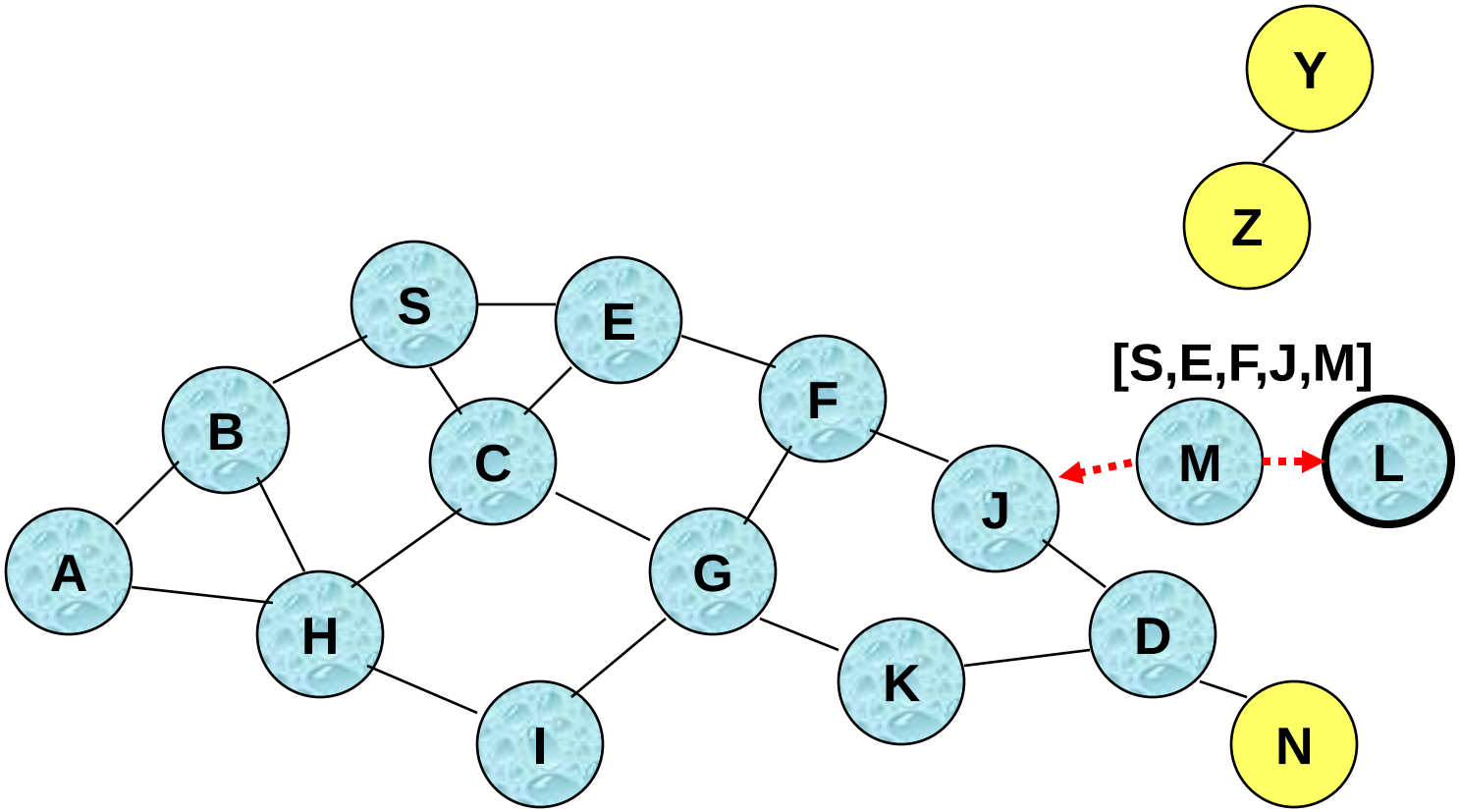
- **Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once**

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

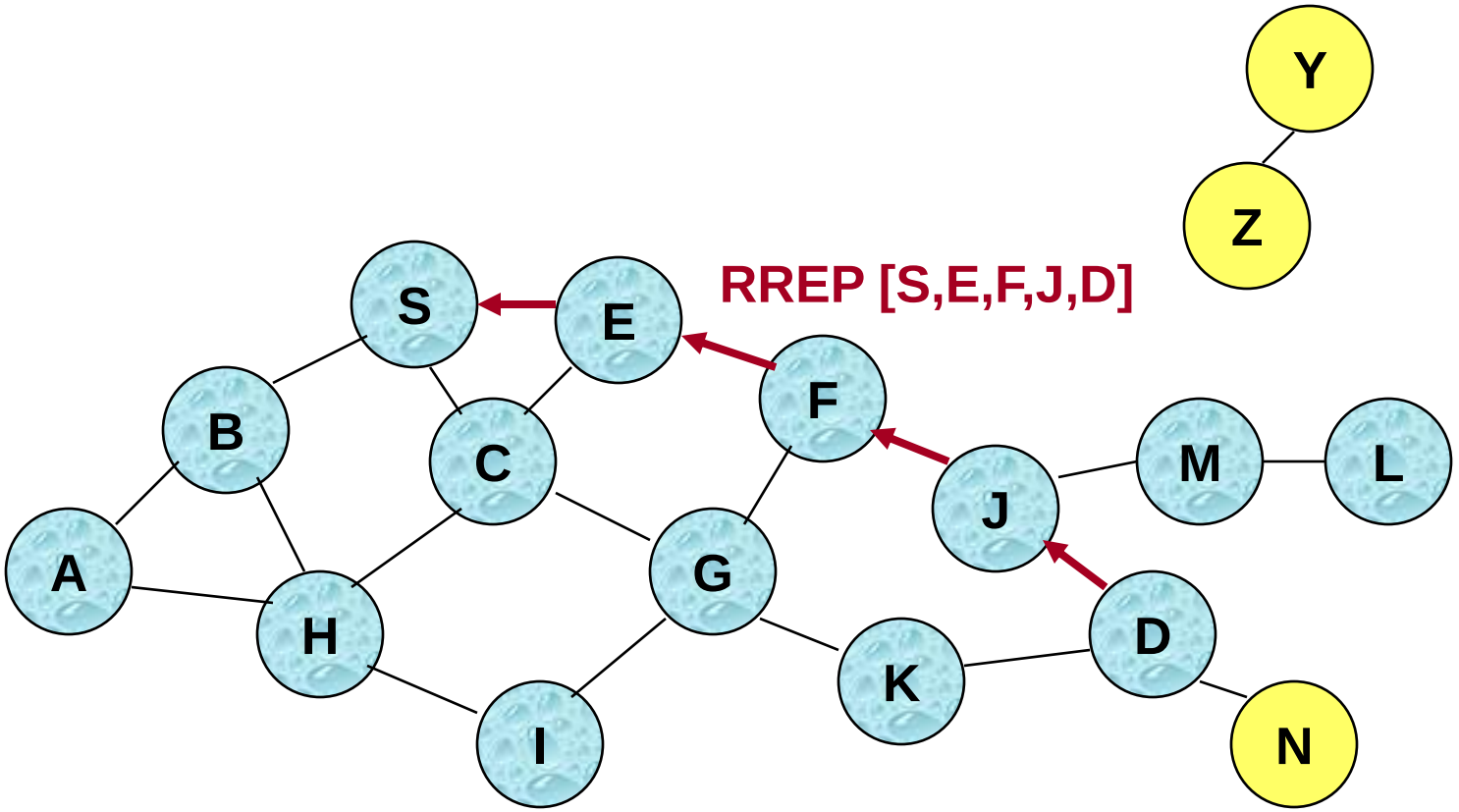


- **Node D does not forward RREQ, because node D is the intended target of the route discovery**

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 (for bi-directional links)
- RREP includes the route from S to D on which RREQ was received by node D

Route Reply in DSR



Represents RREP control message

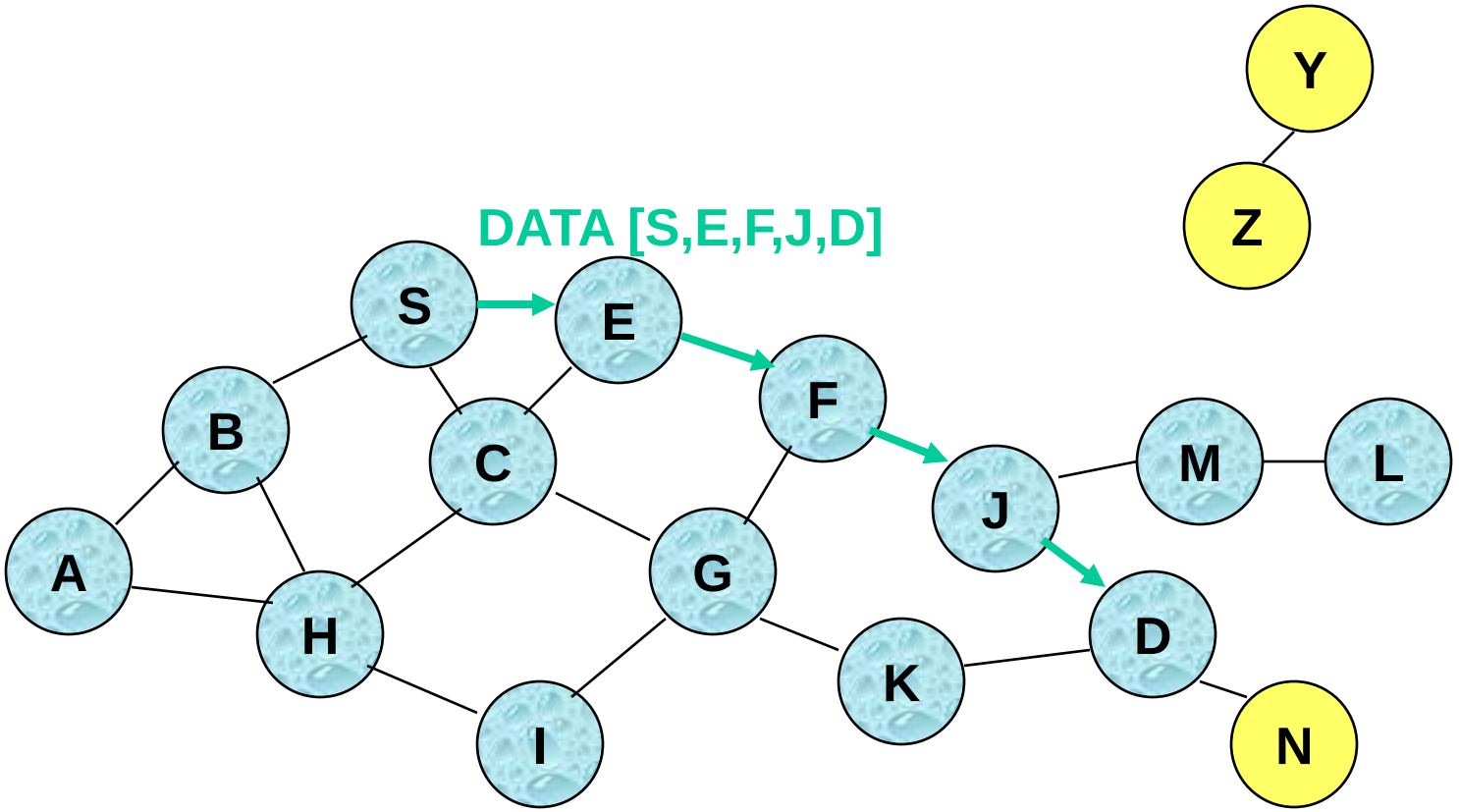
Route Reply in DSR

- 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 received on a link that is known to be bi-directional**
- If unidirectional (asymmetric) links are allowed, then RREP may need a route discovery from node D to node S
  - 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 (RREP) is piggybacked on the Route Request (RREQ) 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 (disadv)

When to Perform a Route Discovery?

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

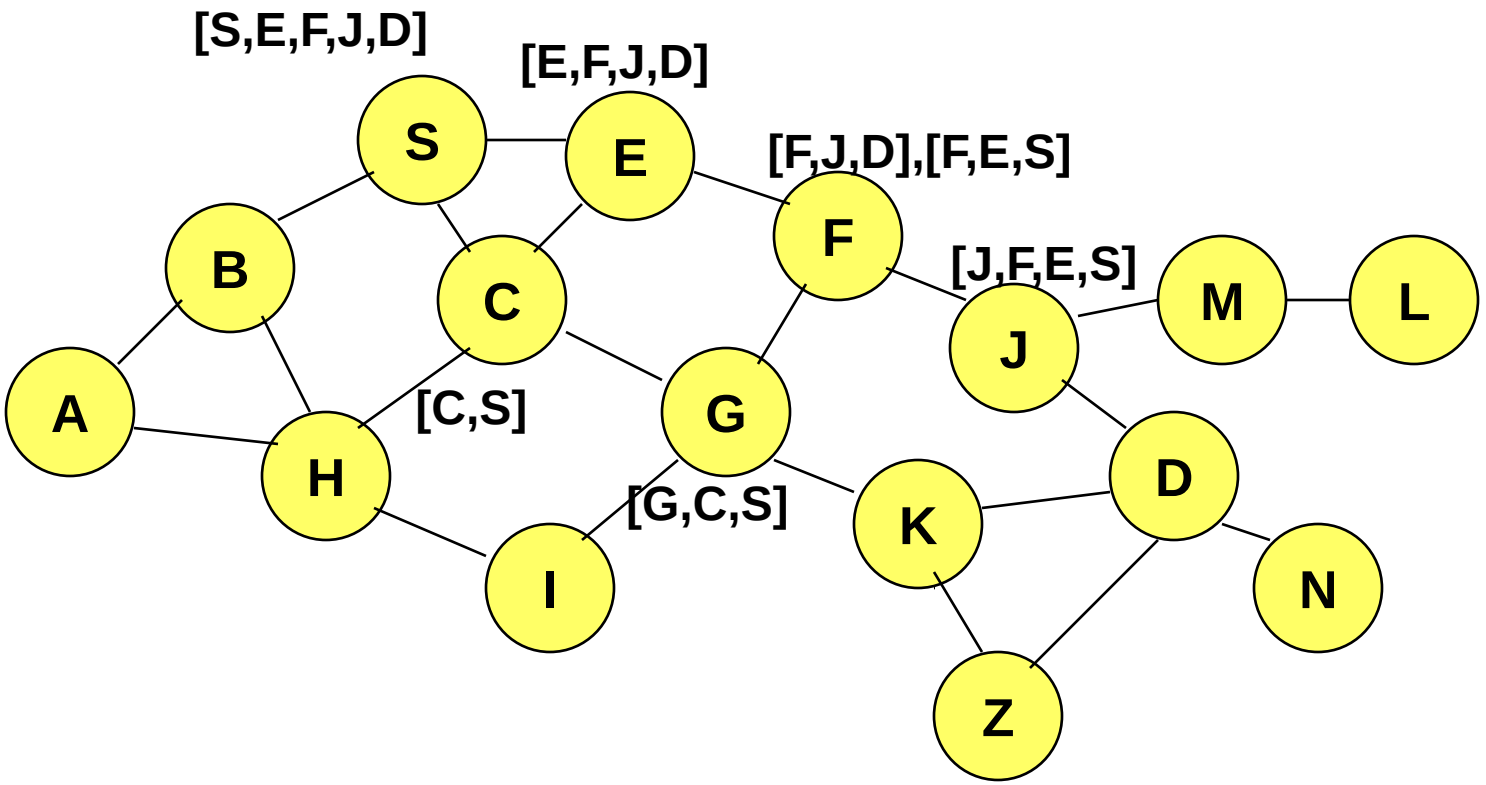
DSR Optimization: **Route Caching**

- 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 **RREQ [S,C,G]** destined for node D, node K learns **route [K,G,C,S]** to node S
- When node F forwards **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 (from a neighbor)**

Use of Route Caching

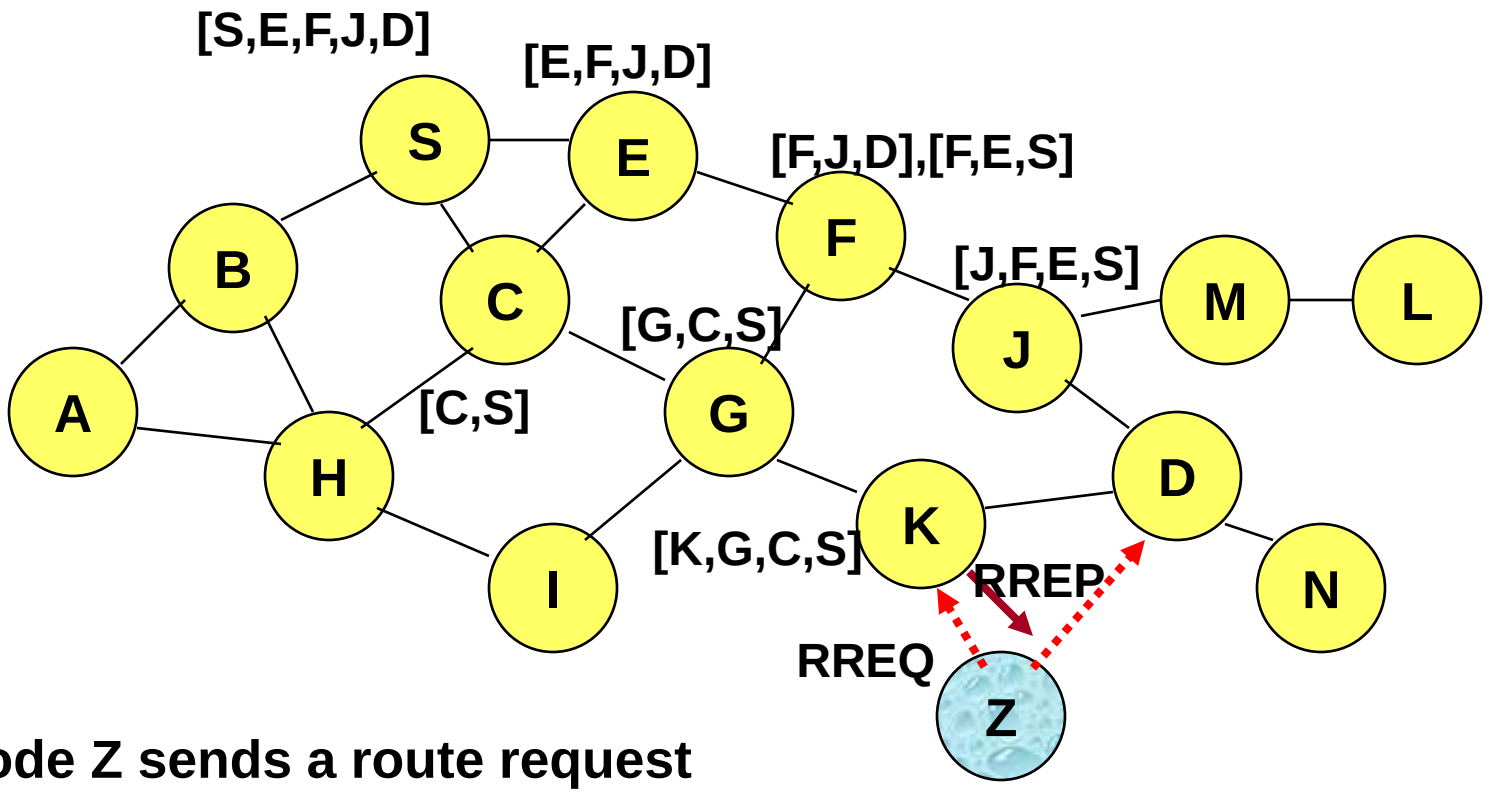
- When node S learns that a route to node D is broken, it uses another route from its local cache, if such a route to D exists in its cache. Otherwise, node S initiates route discovery by sending a RREQ
- Node X on receiving a RREQ for some node D can send a RREP if node X knows a route to node D (stored in its ROUTE CACHE)
- Use of route cache
  - **can speed up route discovery**
  - **can reduce propagation of route requests**

Use of Route Caching



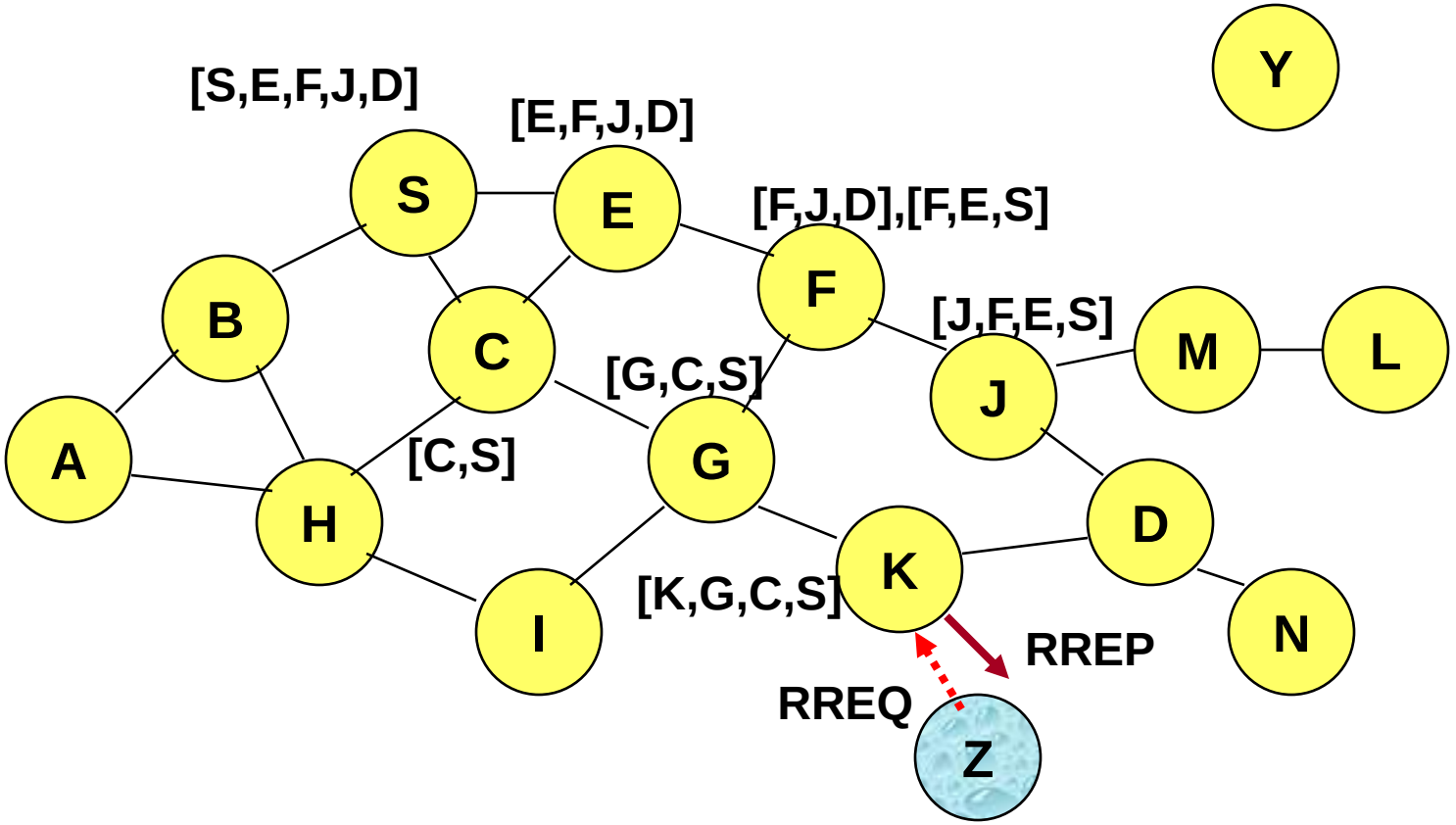
[P,Q,R] **Represents cached route at a node**
(DSR maintains the cached routes in a tree format)

Use of Route Caching:
Can Speed up Route Discovery



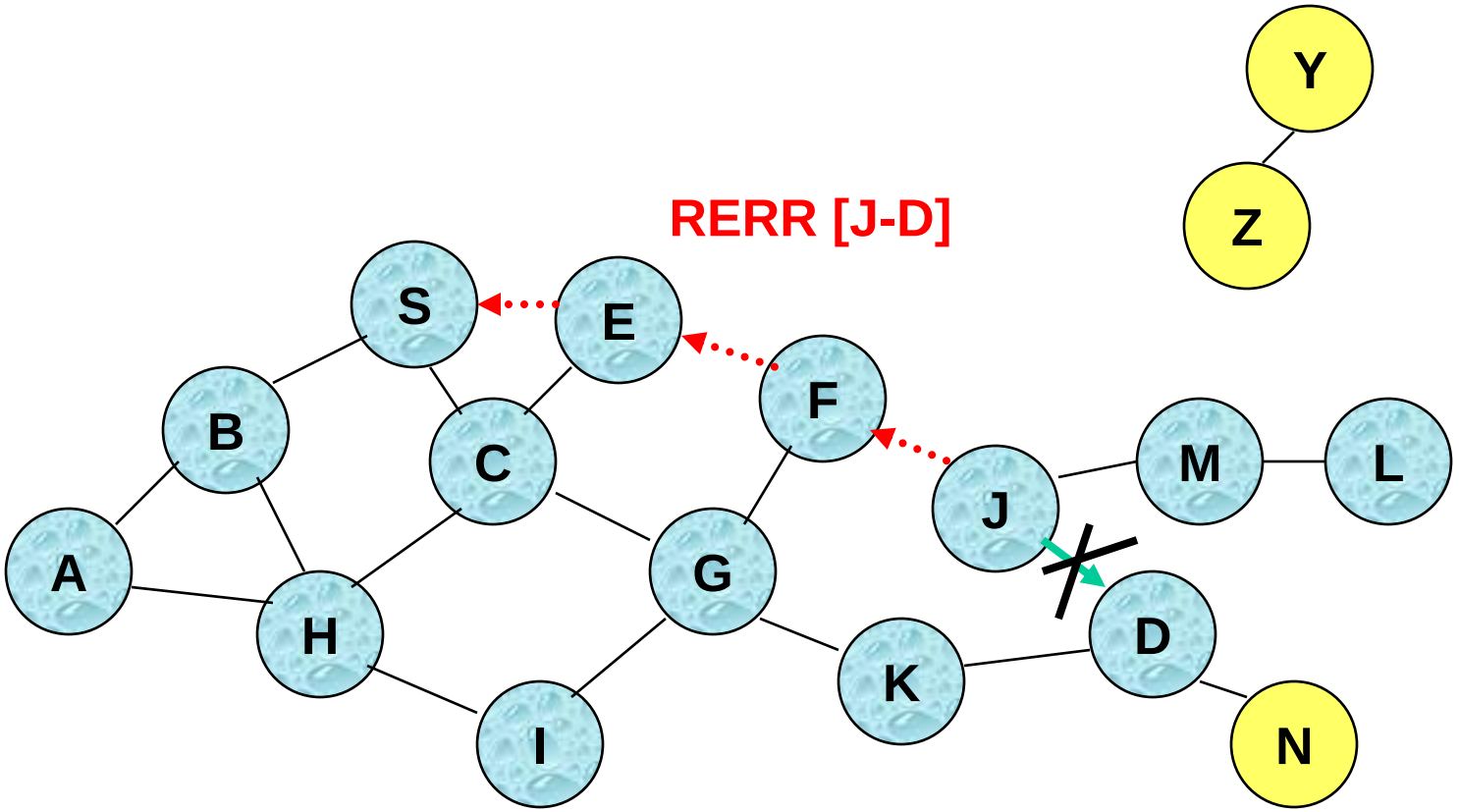
When node Z sends a route request for node C, node K sends back a route reply [Z,K,G,C] to node Z using a locally cached route

Use of Route Caching:
Can Reduce Propagation of Route Requests



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 of 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**
- These have adverse impact on TCP

DSR: 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

DSR: 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 RREQs** propagated by neighboring nodes
  - insertion of random delays before forwarding RREQ
- Increased contention, if too many RREPs come back due to nodes replying using their local caches
  - **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

DSR: Disadvantages (contd.)

- An intermediate node may send RREP 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.
- Some proposals for cache invalidation based on
  - **Static timeouts**
  - **Adaptive timeouts based on link stability**

Flooding of Control Packets

- How to reduce the scope of the **route request** flood?
  - **LAR**
  - **Query localization**
- How to reduce redundant broadcasts ?
  - **The Broadcast Storm Problem**

Location-Aided Routing (LAR)

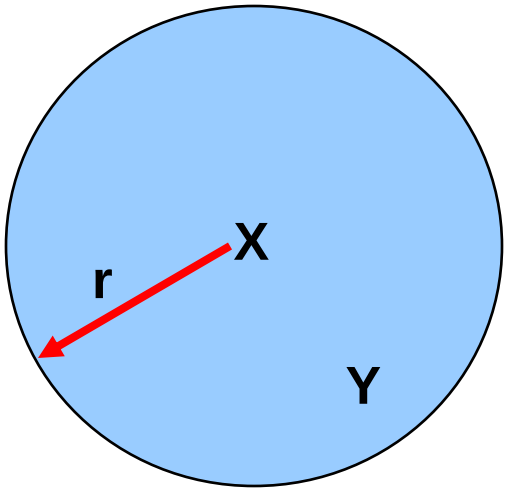
- Exploits location information to limit scope of route request flood
  - Location information may be obtained using GPS (Global Positioning System)
- *Expected Zone* is determined as a region that is expected to hold the current location of the destination
  - Expected region determined based on potentially old location information, and knowledge of the destination's speed
- Route requests limited to a ***Request Zone*** that contains the Expected Zone and location of the sender node

Expected Zone in LAR

X = last known location of node D, at time t_0

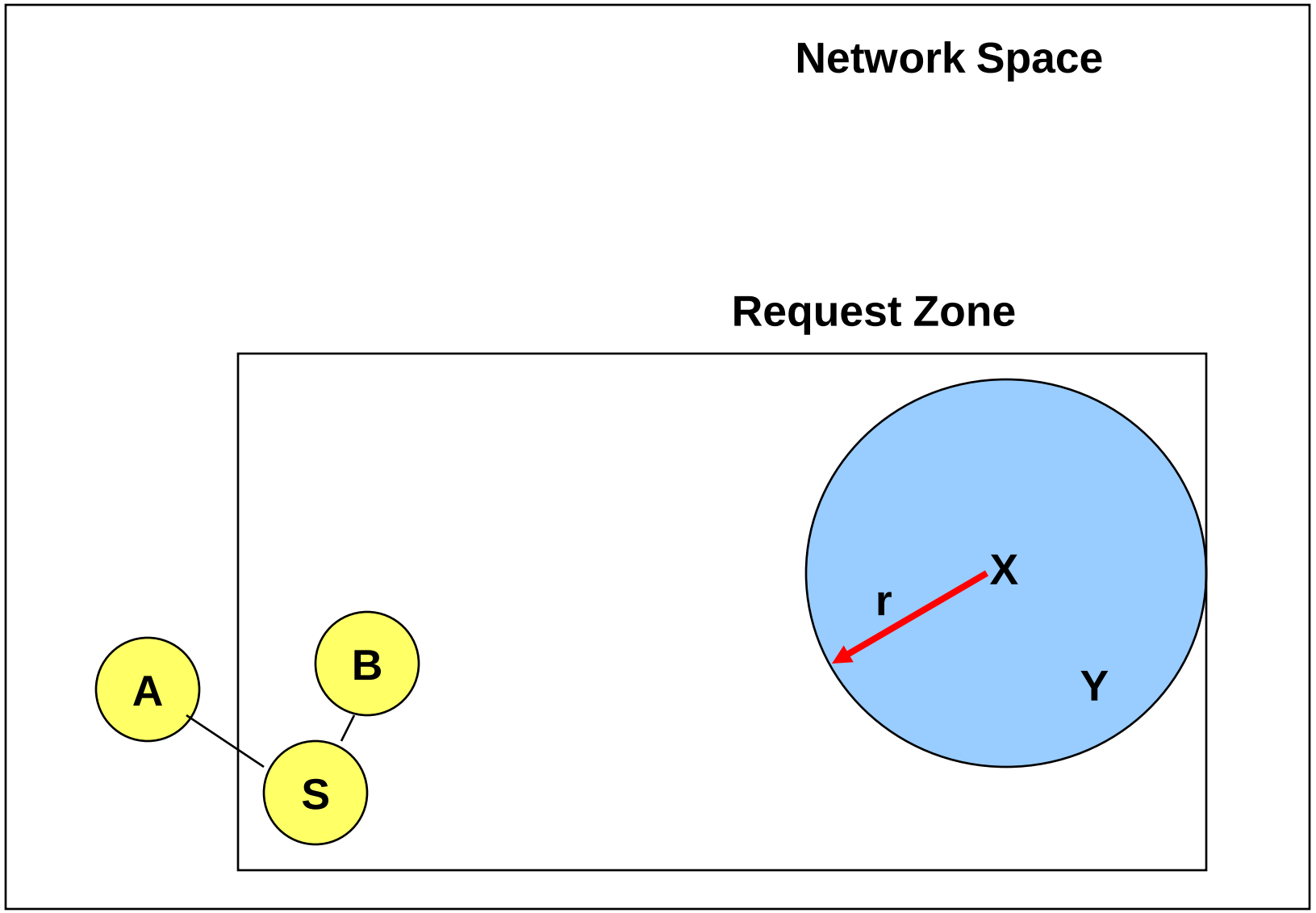
Y = location of node D at current time t_1, unknown to node S

r = (t_1 - t_0) * estimate of D's speed



Expected Zone

Request Zone in LAR



LAR

- Only nodes within the request zone forward route requests
  - Node A does not forward RREQ, but node B does (see the figure in previous slide)
- Request zone explicitly specified in the route request
- Each node must know its physical location to determine whether it is within the request zone

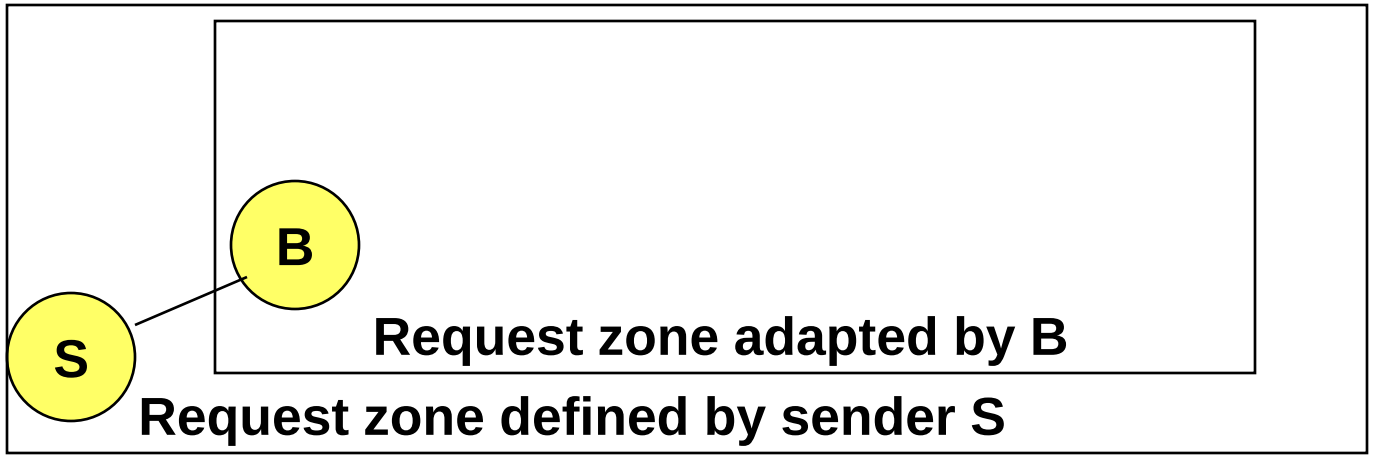
LAR

- Only nodes **within the request zone** forward route requests
- If route discovery using the smaller request zone fails to find a route, the sender initiates another route discovery (after a timeout) using a larger request zone
  - the larger request zone may be the entire network
- Rest of route discovery protocol similar to DSR

LAR Variations: Adaptive Request Zone

- Each node may modify the request zone included in the forwarded request

- Modified request zone may be determined using more recent/accurate information, and may be smaller than the original request zone



LAR Variations: Implicit Request Zone

- In the previous scheme, a RREQ explicitly specified a request zone
- Alternative approach: A node X forwards a RREQ received from Y if node X is deemed to be closer to the expected zone as compared to Y
- The motivation is to attempt to bring the RREQ physically closer to the destination node after each forwarding

Location-Aided Routing

- The basic proposal assumes that, ***initially***, location information for node X becomes known to Y only during a route discovery
- This location information is used for a future route discovery
  - Each route discovery yields more updated information which is used for the next discovery

Variations

- Location information can also be **piggybacked** on any message from Y to X
- Y may also **proactively distribute** its location information
  - Similar to other protocols (e.g., DREAM, GLS)

Location Aided Routing (LAR)

- Advantages
  - reduces the scope of route request flood
  - reduces overhead of route discovery
- Disadvantages
  - Nodes need to know their physical locations
  - Does not take into account possible existence of obstructions for radio transmissions

Query Localization

- Limits RREQ flood without using physical information
- RREQs are propagated only along paths that are *close* to the previously known route
- The *closeness* property is defined without using physical location information

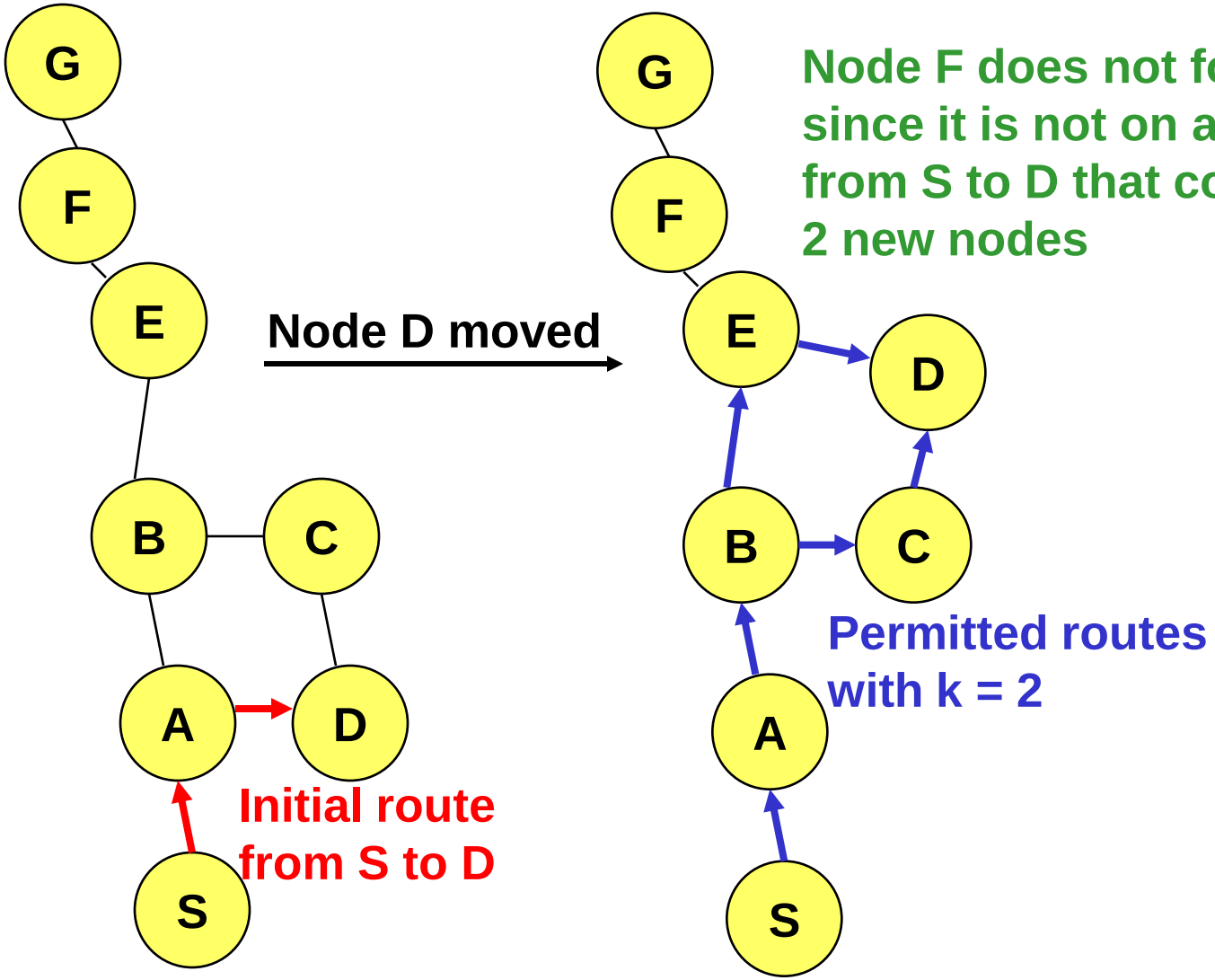
Query Localization

- **Path locality heuristic**: Look for a new path that contains at most ***k*** nodes that were not present in the previously known route

- **Old route is piggybacked on a RREQ**

- RREQ is forwarded only if the accumulated route in the RREQ contains at most ***k*** new nodes that were absent in the old route
  - this limits propagation of the RREQ

Query Localization: Example



Node F does not forward the RREQ since it is not on any route from S to D that contains at most 2 new nodes

Query Localization

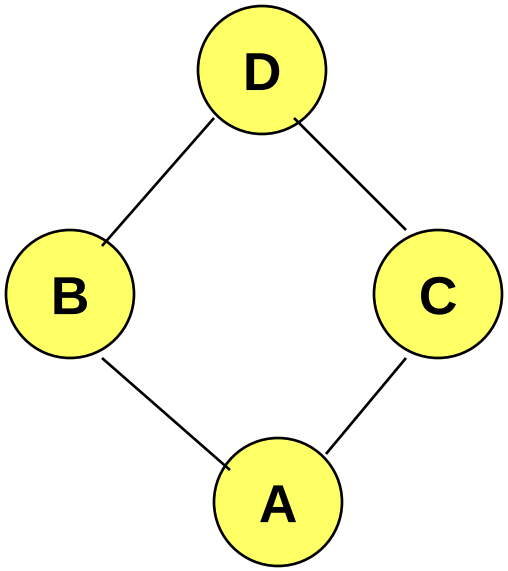
- Advantages:
  - Reduces overhead of route discovery without using physical location information
  - Can perform better in presence of obstructions by searching for new routes in the ***vicinity*** of old routes

- Disadvantage:
  - May yield routes longer than LAR

    (Shortest route may contain more than k new nodes)

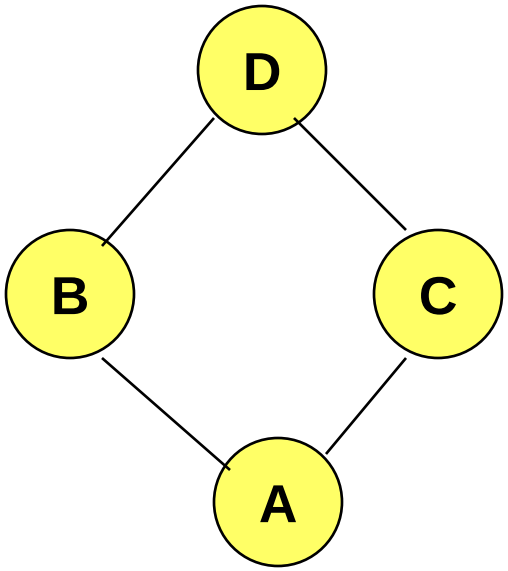
Broadcast Storm Problem

- When node A broadcasts a route query, nodes B and C both receive it
- B and C both forward to their neighbors
- B and C transmit at about the same time since they are reacting to receipt of the same message from A
- This results in a high probability of **collisions**



Broadcast Storm Problem

- **Redundancy:** A given node may receive the same RREQ from too many nodes, when one copy would have sufficed
- Node D may receive from nodes B and C both



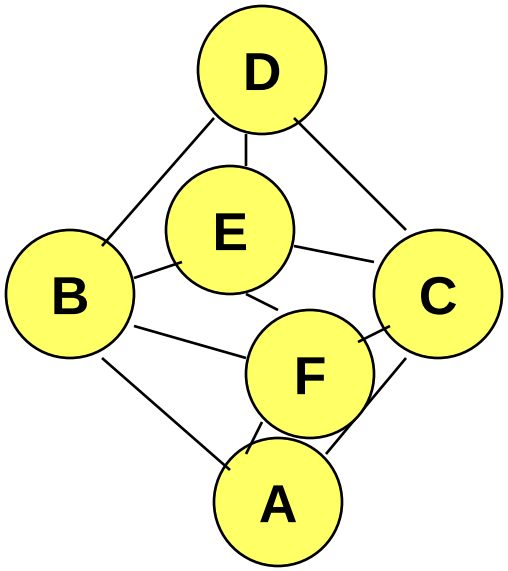
Solutions for Broadcast Storm

- **Probabilistic scheme**: On receiving a route request for the first time, a node will **re-broadcast (forward)** the request with **probability p**

- Also, re-broadcasts by different nodes should be staggered by using a **collision avoidance** technique (wait a random delay when channel is idle)
  - this would reduce the probability that nodes B and C would forward a packet simultaneously in the previous example

Solutions for Broadcast Storms

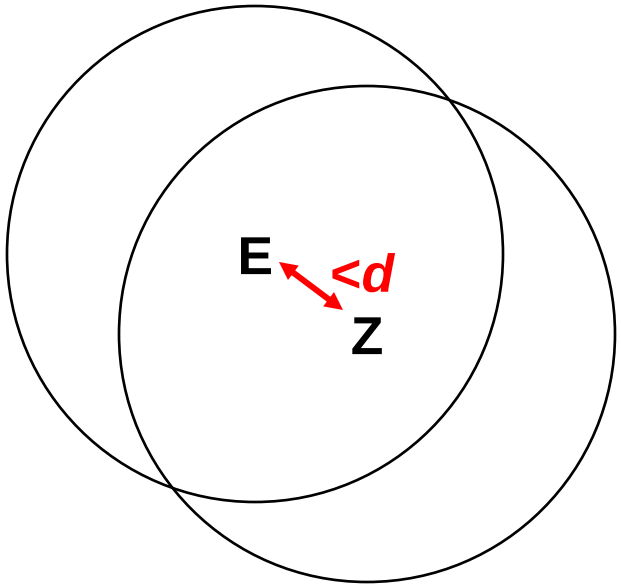
- **Counter-Based Scheme**: If node E hears more than ***k*** neighbors broadcasting a given RREQ, before it can itself forward it, then node E will not forward the request
- Intuition: ***k*** neighbors together have probably already forwarded the request to all of E's neighbors



Solutions for Broadcast Storms

- **Distance-Based Scheme**: If node E hears RREQ broadcasted by some node Z within physical distance *d*, then E will not re-broadcast the request

- Intuition: Z and E are too close, so transmission areas covered by Z and E are not very different
  - if E re-broadcasts the request, not many nodes who have not already heard the request from Z will hear the request



Summary: Broadcast Storm Problem

- Flooding is used in many protocols, such as Dynamic Source Routing (DSR)
- Problems associated with flooding
  - **Collisions**
  - **Contentions**
  - **redundancy**
- Collisions may be reduced by “jittering” (waiting for a random interval before propagating the flood)
- Redundancy may be reduced by selectively re-broadcasting packets from only a subset of the nodes

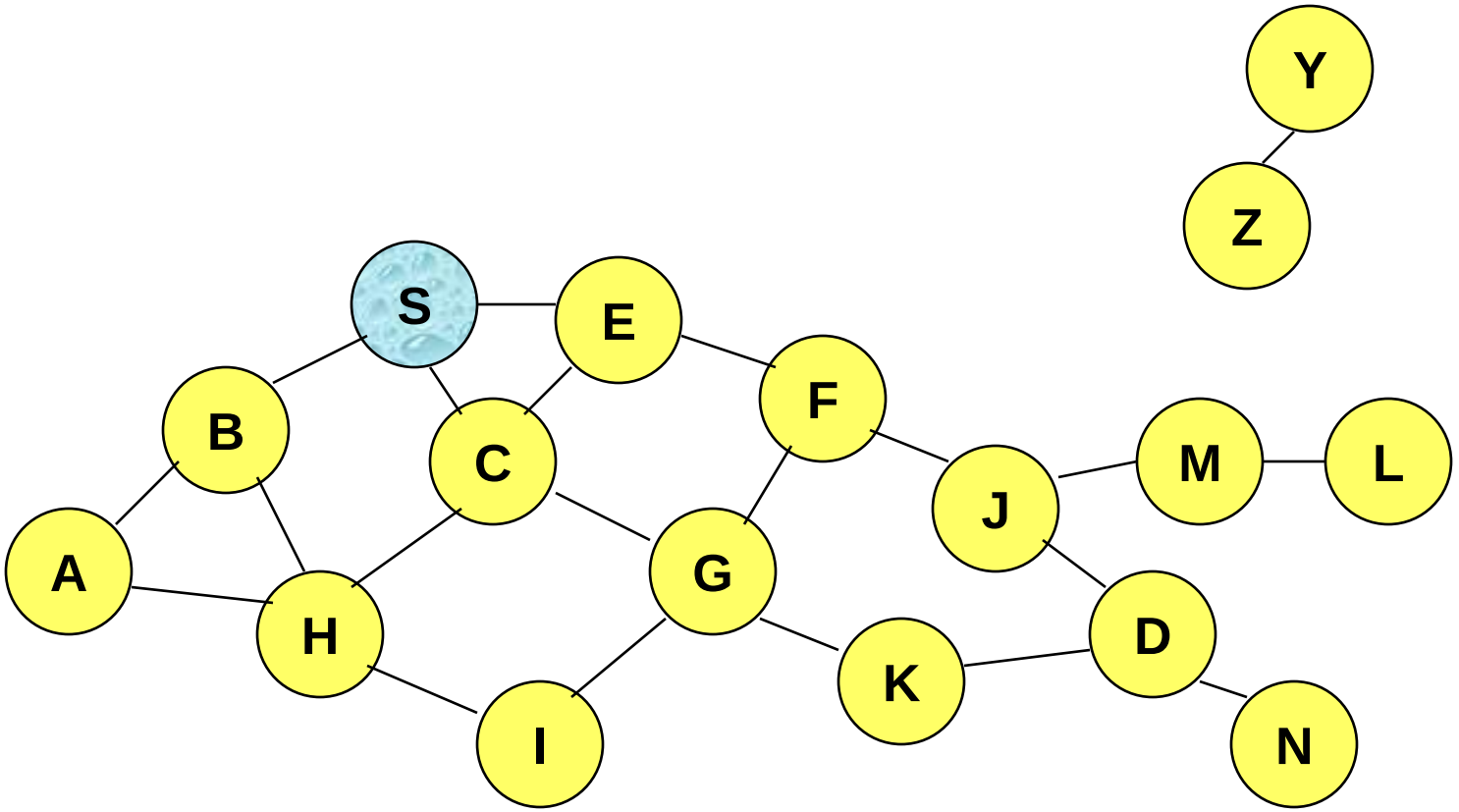
Ad Hoc On-Demand Distance Vector Routing (AODV)

- 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

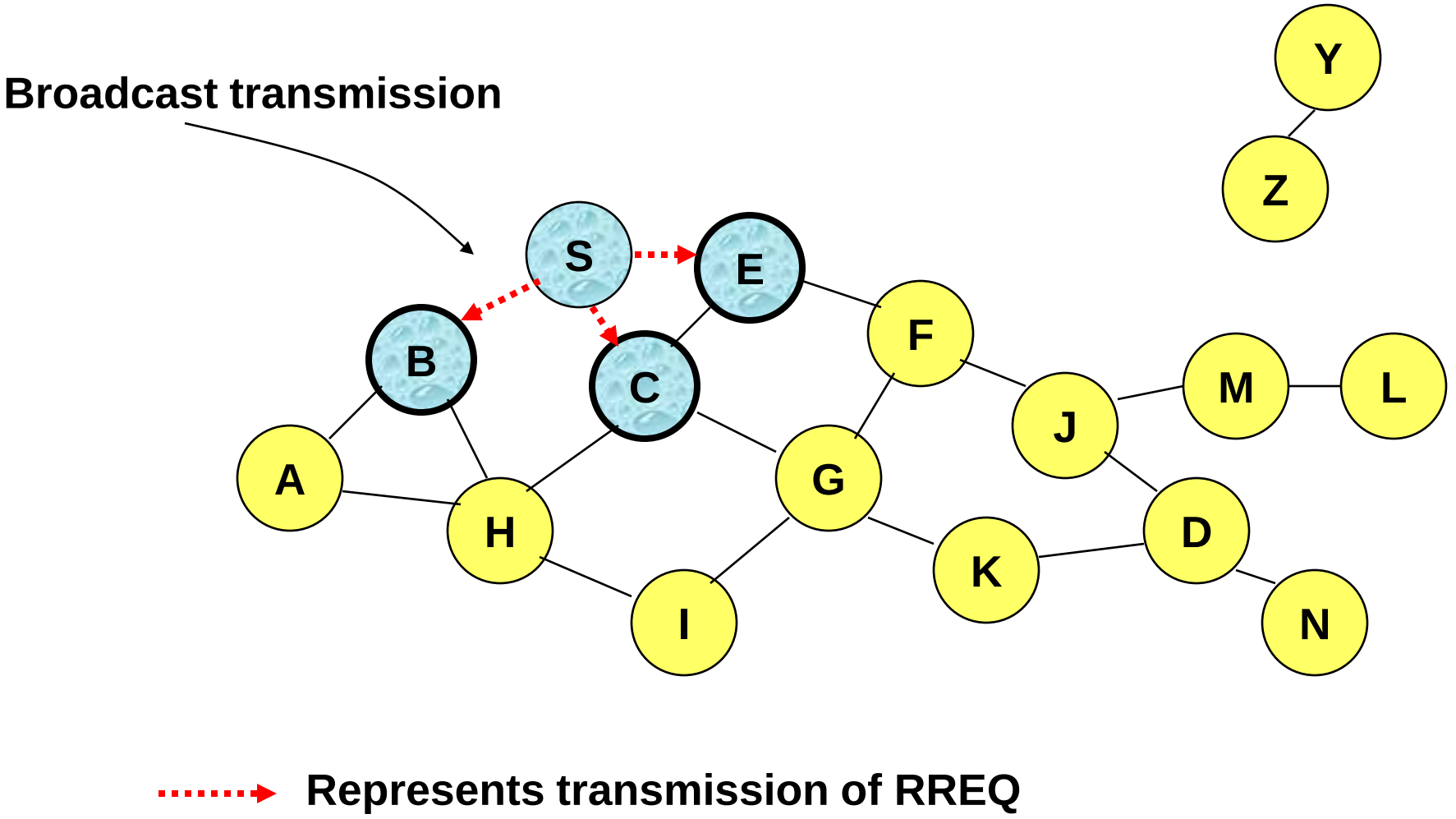
- Route Requests (RREQ) are forwarded in a manner similar to DSR
- When a node re-broadcasts a RREQ, it sets up a reverse path pointing towards the source
  - AODV assumes symmetric (bi-directional) links
- When the intended destination receives a RREQ, it replies by sending a Route Reply (RREP)
- RREP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

Route Requests in AODV

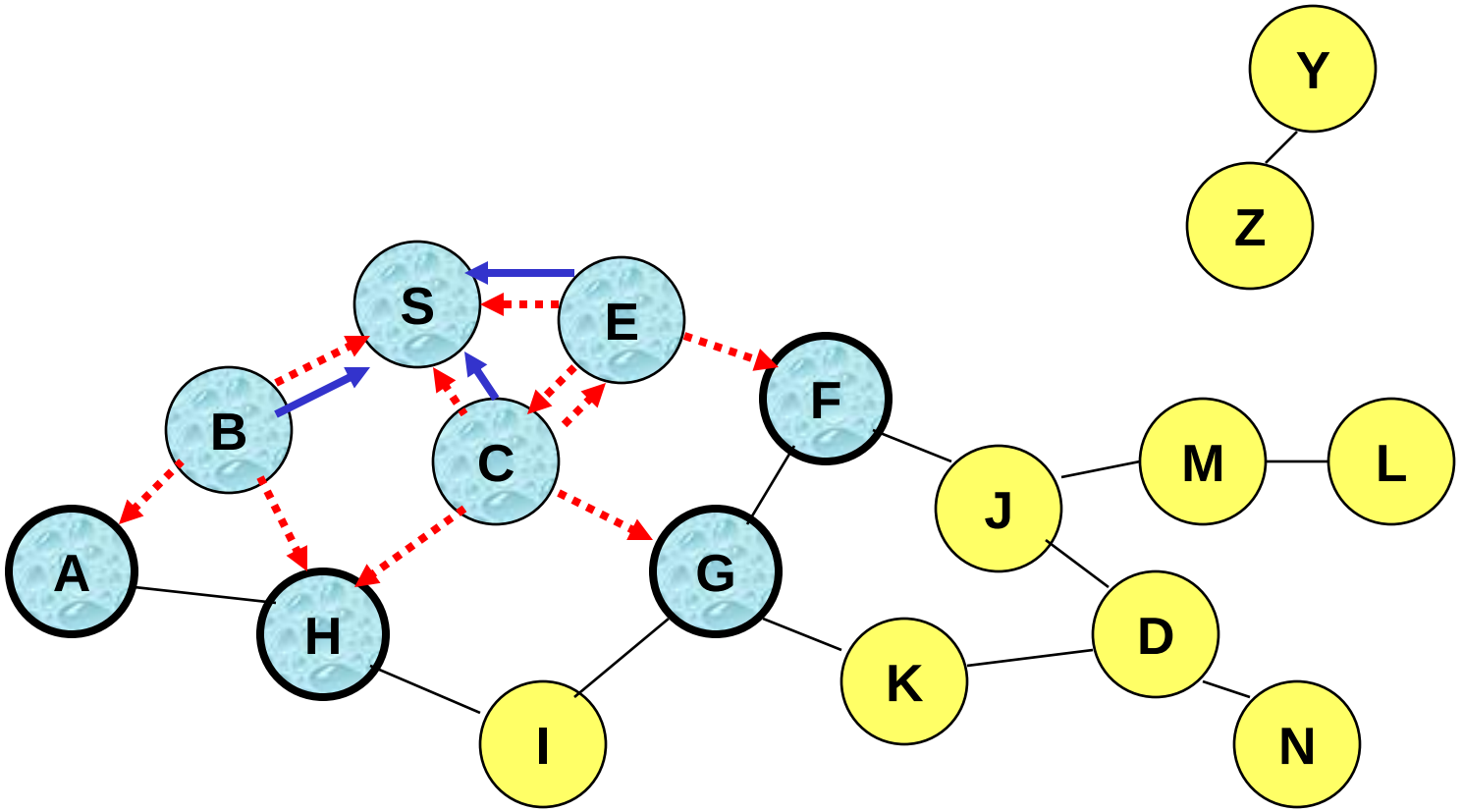


Represents a node that has received RREQ for D from S

Route Requests in AODV

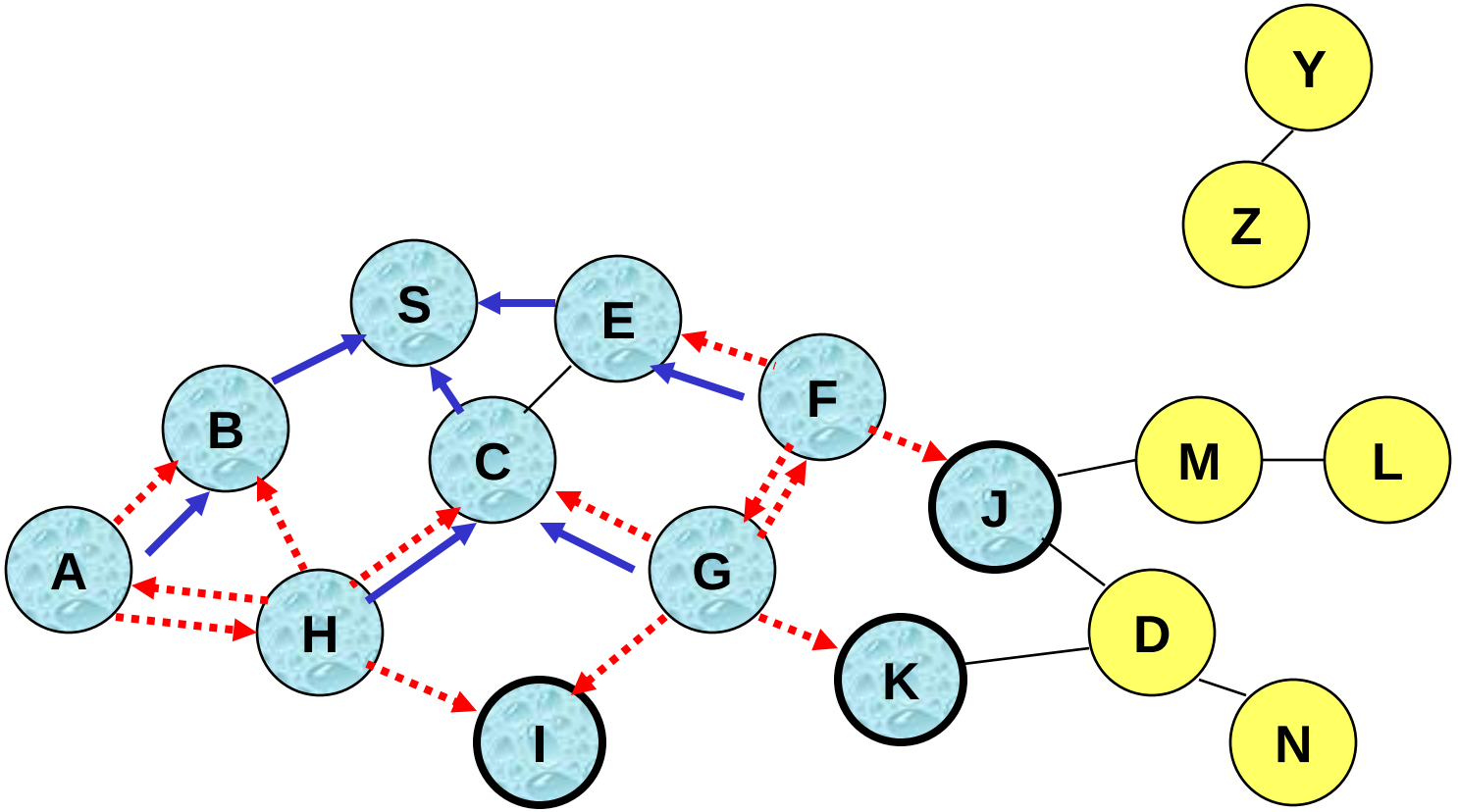


Route Requests in AODV



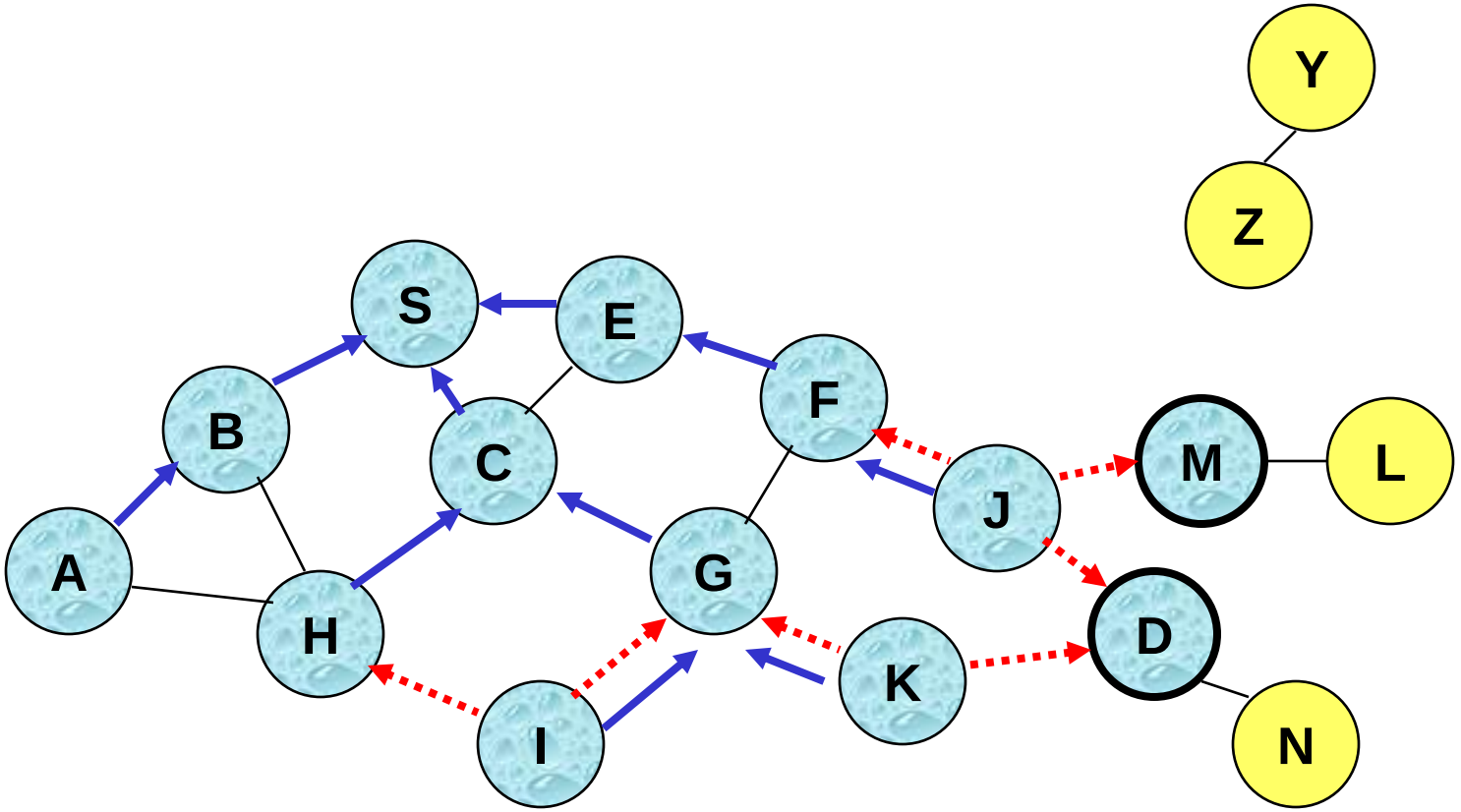
Represents links on Reverse Path

Reverse Path Setup in AODV

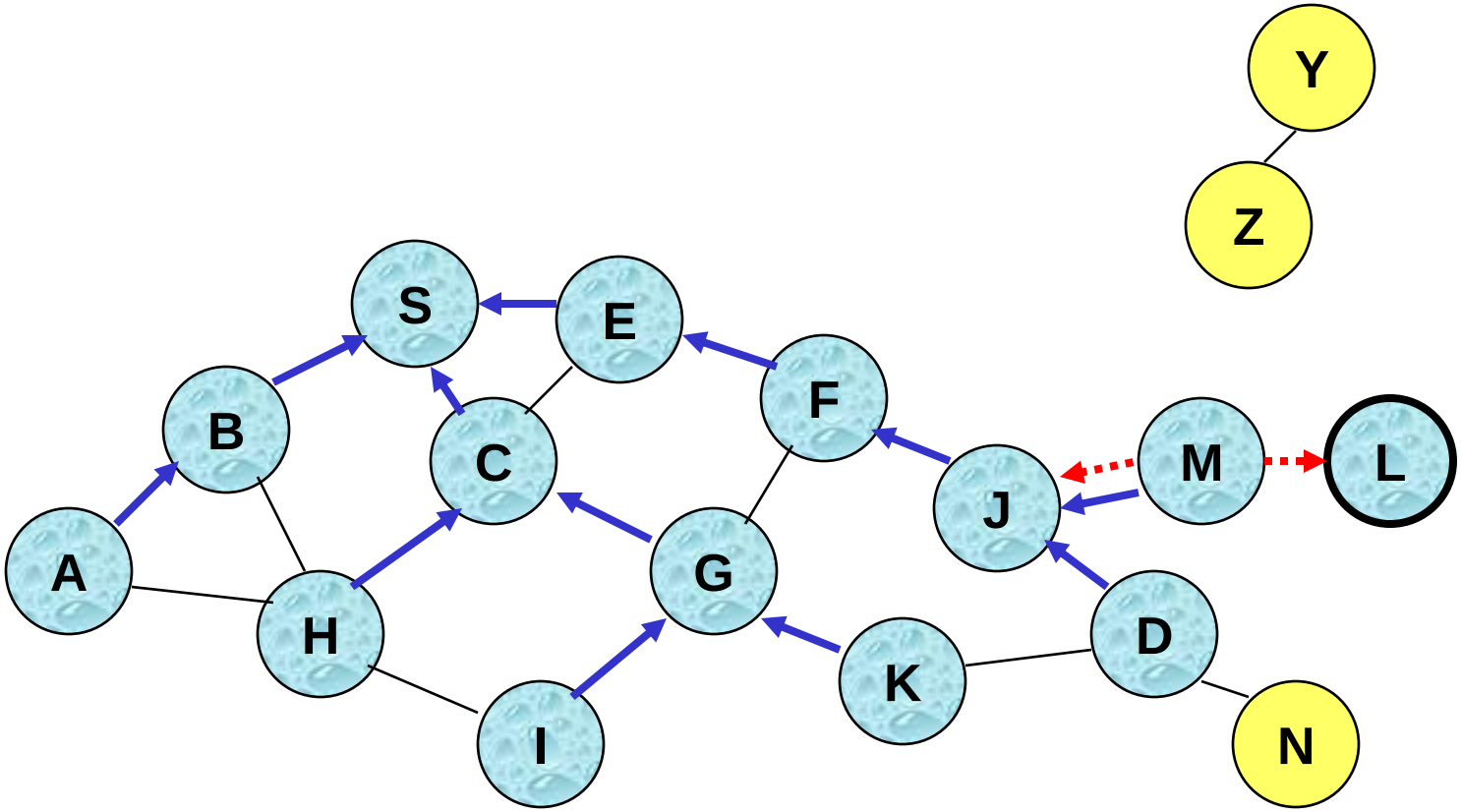


- **Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once**

Reverse Path Setup in AODV

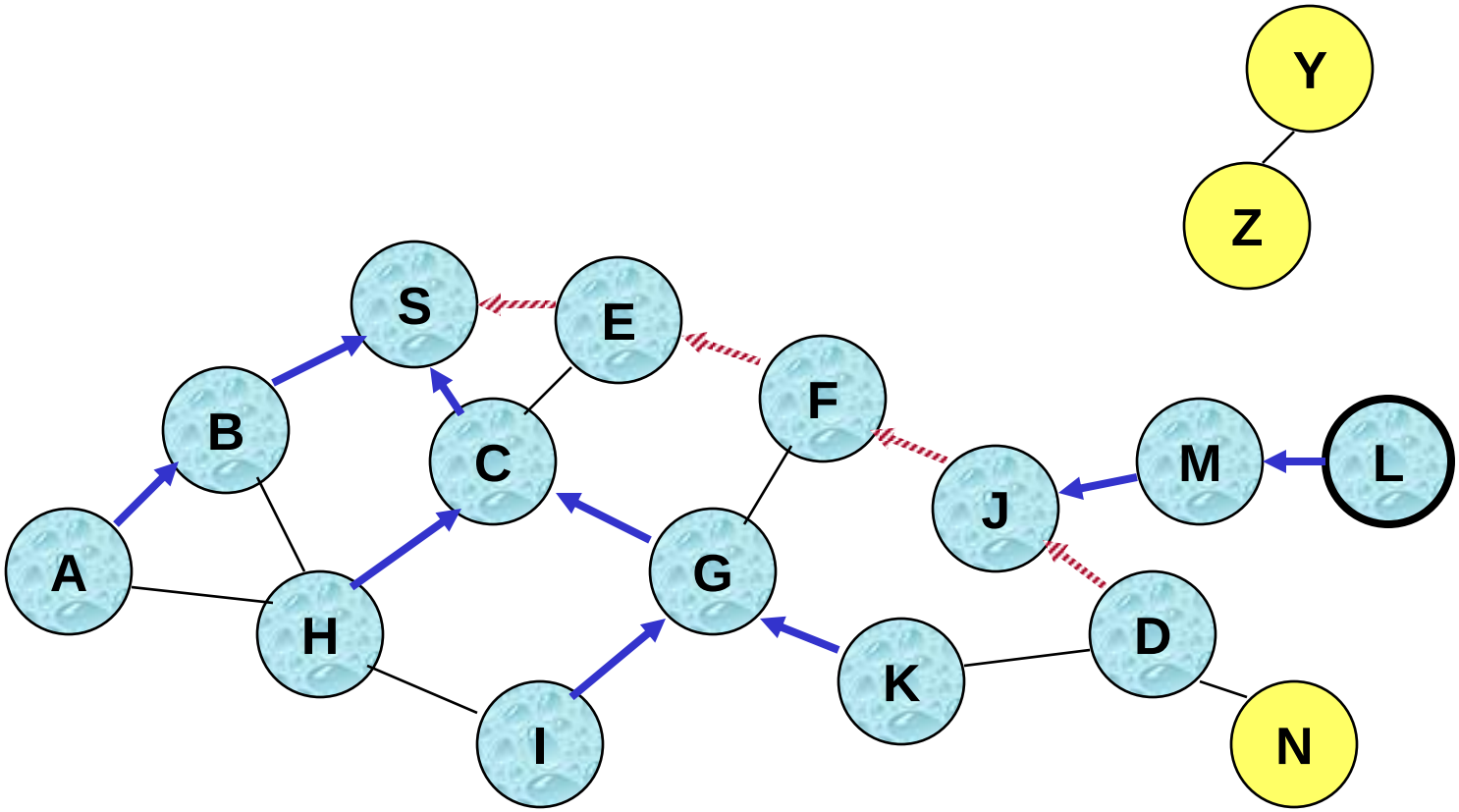


Reverse Path Setup in AODV



- **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

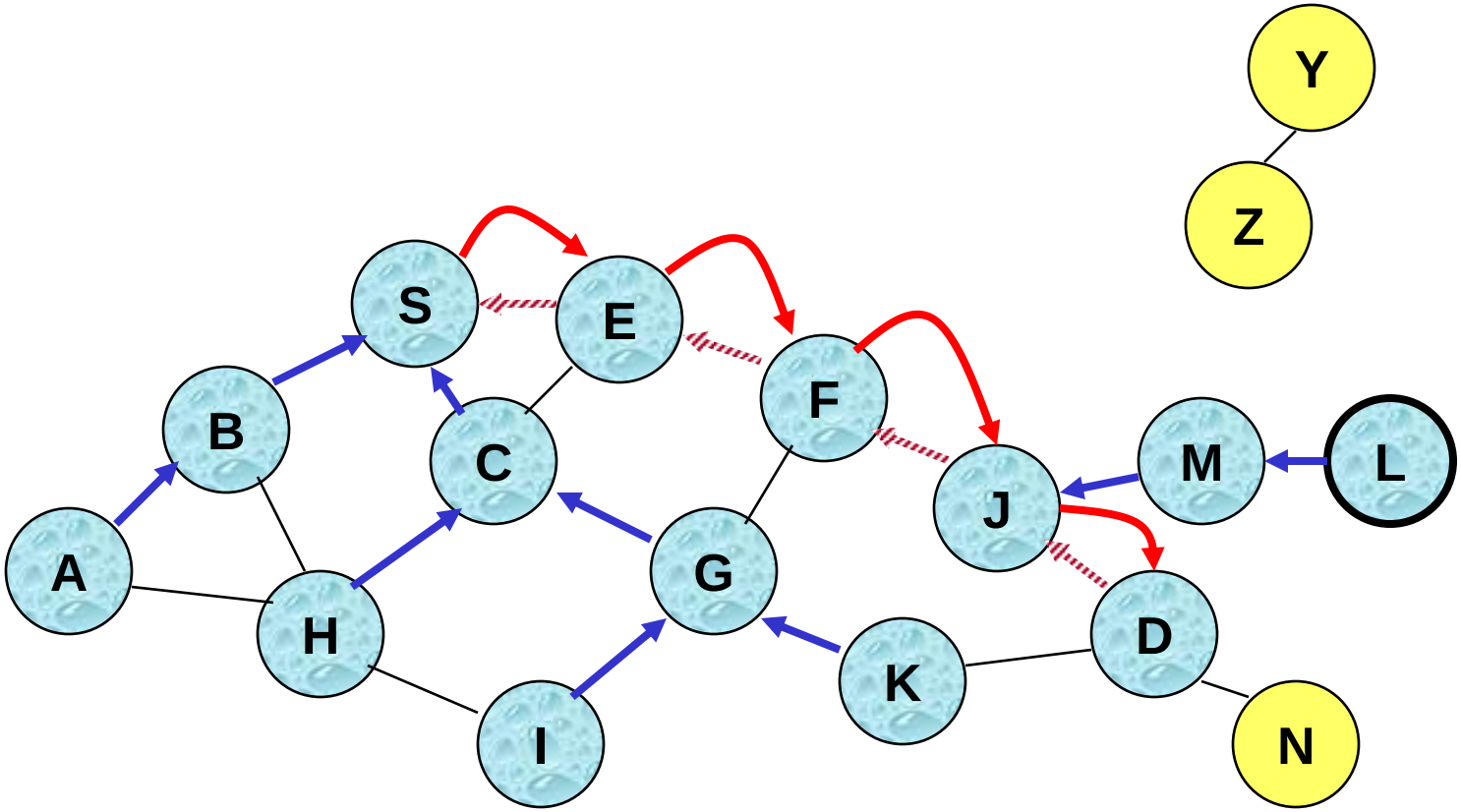
Route Reply in AODV

- An **intermediate node** (not the destination) may also send a 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 RREP when using AODV not as high as DSR
  - **A new RREQ by node S for a destination D is assigned a higher destination sequence number. An intermediate node which knows a route, but with a smaller sequence number, cannot send RREP**

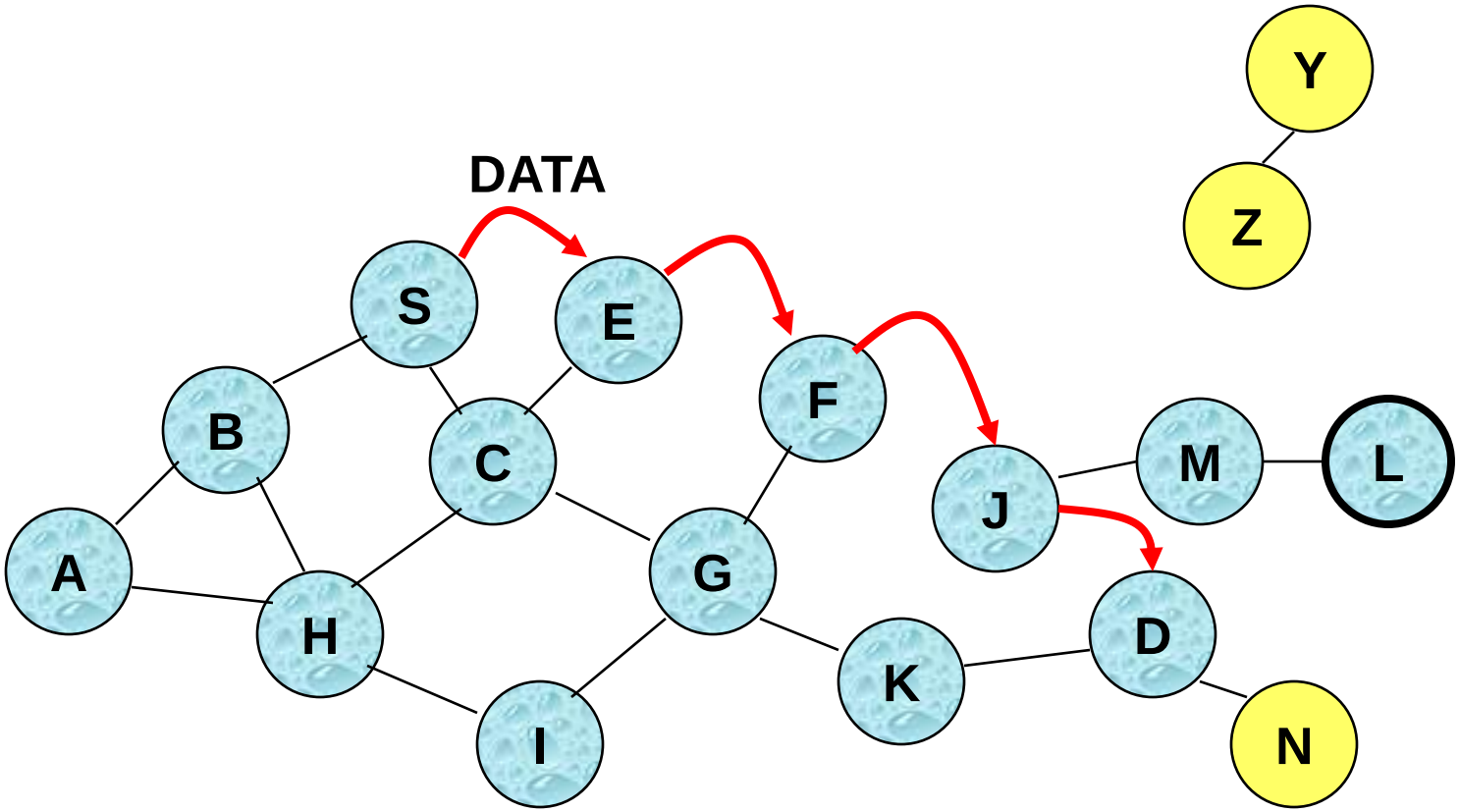
Forward Path Setup in AODV



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 an *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 **Y** of **node X** is considered **active** for a routing table entry if the neighbor **Y** sent a packet within *active_route_timeout* interval which was forwarded using that entry (Also called **active precursors**)
- When the next hop link in a routing table entry breaks (detected thro' periodic Hello message), all **active** neighbors are informed
- Link failures are propagated by means of **Route Error (RERR)** 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** N 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

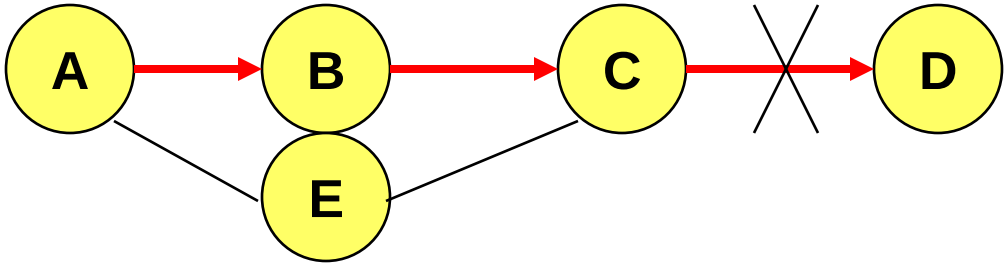
- When node D receives the route request with destination sequence number N, node D will set its sequence number to N, unless it is already larger than N

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

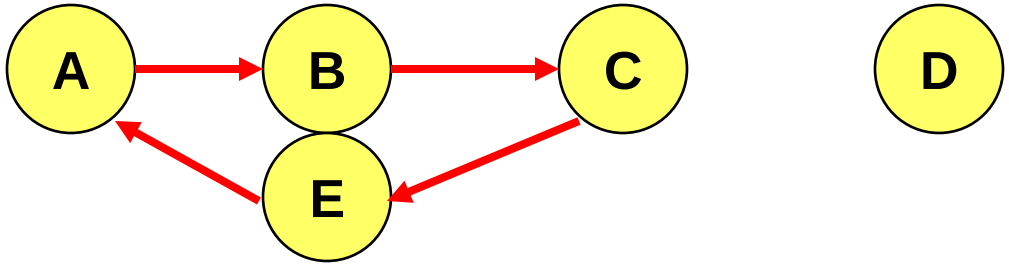
Why Sequence Numbers in AODV

- 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

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

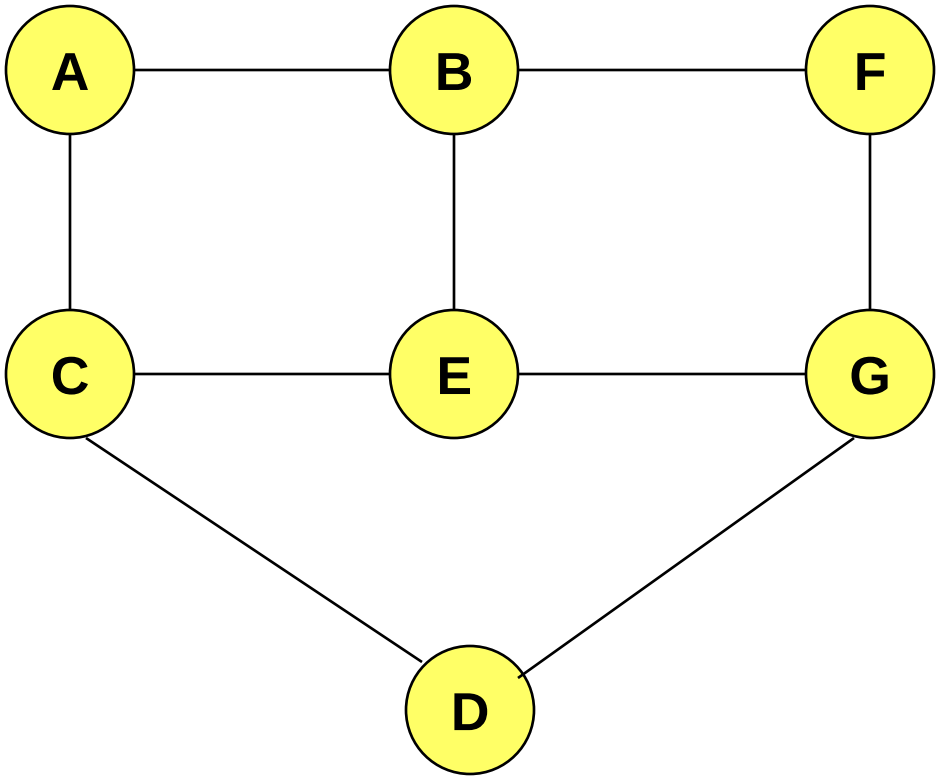
Summary: AODV

- 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
  - DSR may maintain several routes for a single destination
- Unused routes expire even if topology does not change

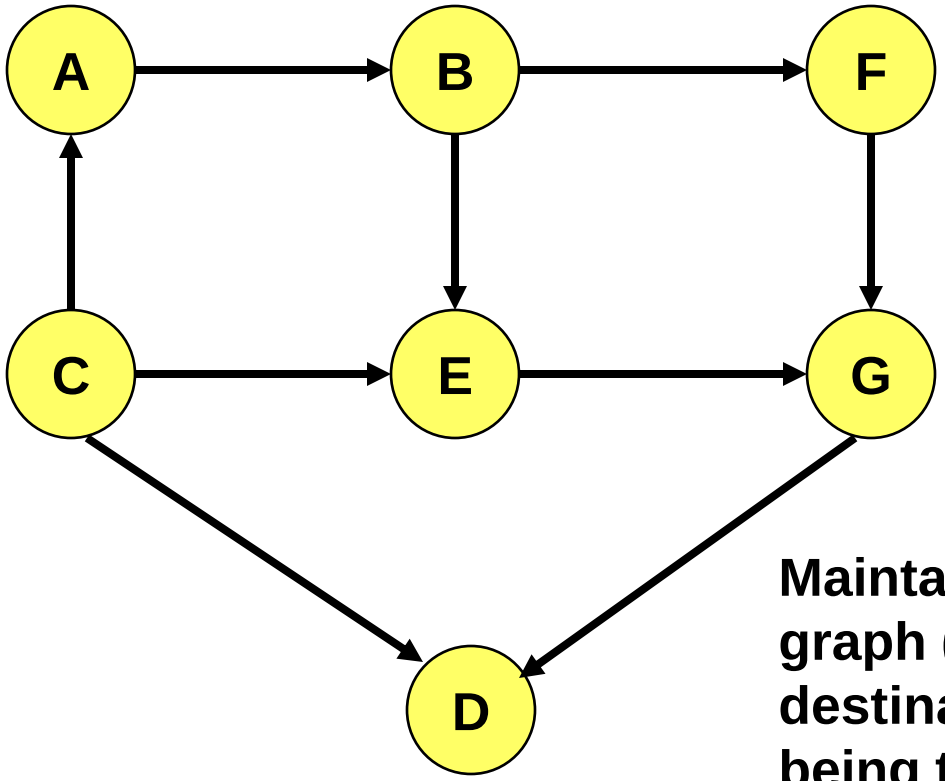
So far ...

- All protocols discussed so far perform some form of flooding
- Now we will consider **protocols which try to reduce/avoid such behavior (i.e. flooding)**

Link Reversal Algorithm [Gafni81]



Link Reversal Algorithm



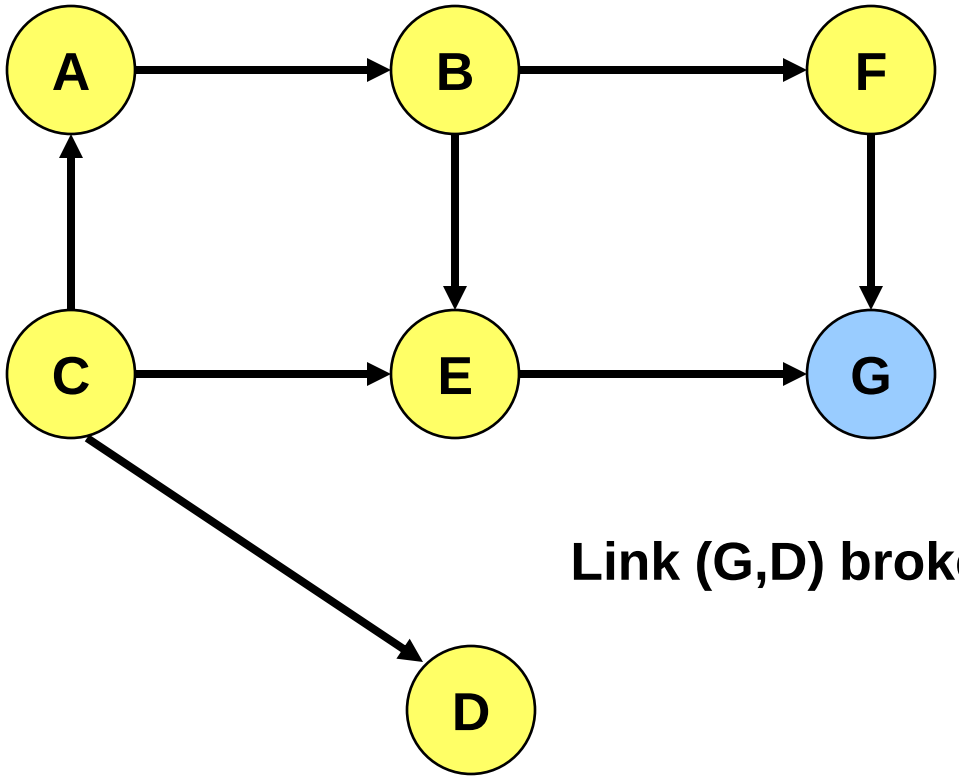
Links are bi-directional

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

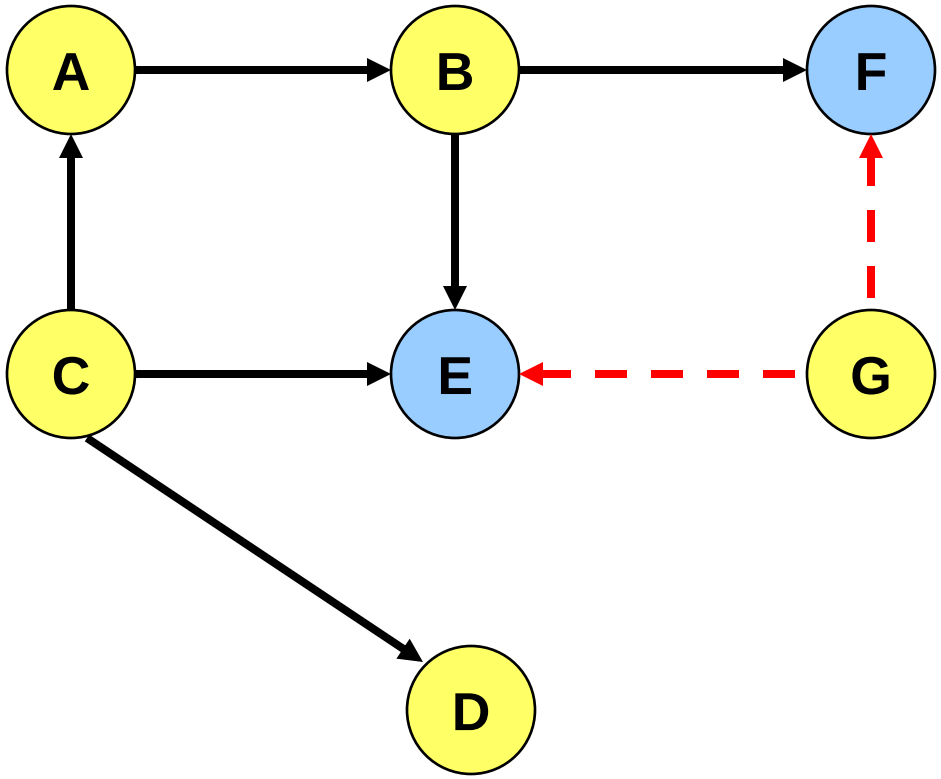


Link (G,D) broke

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

Node G has no outgoing links

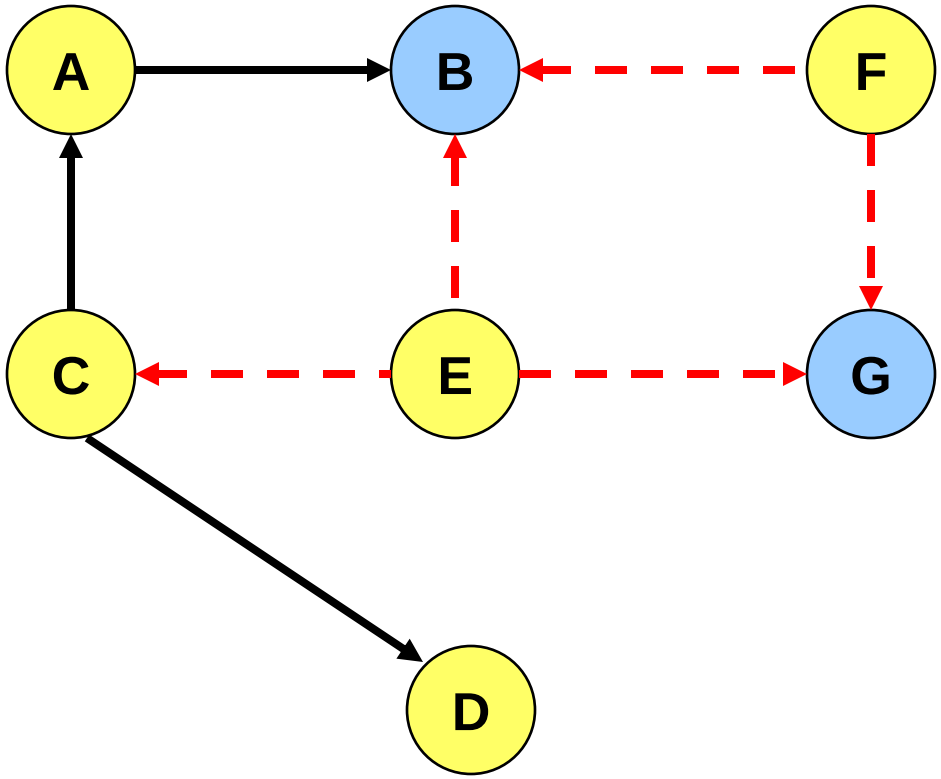
Link Reversal Algorithm



Represents a link that was reversed recently

Now nodes E and F have no outgoing links

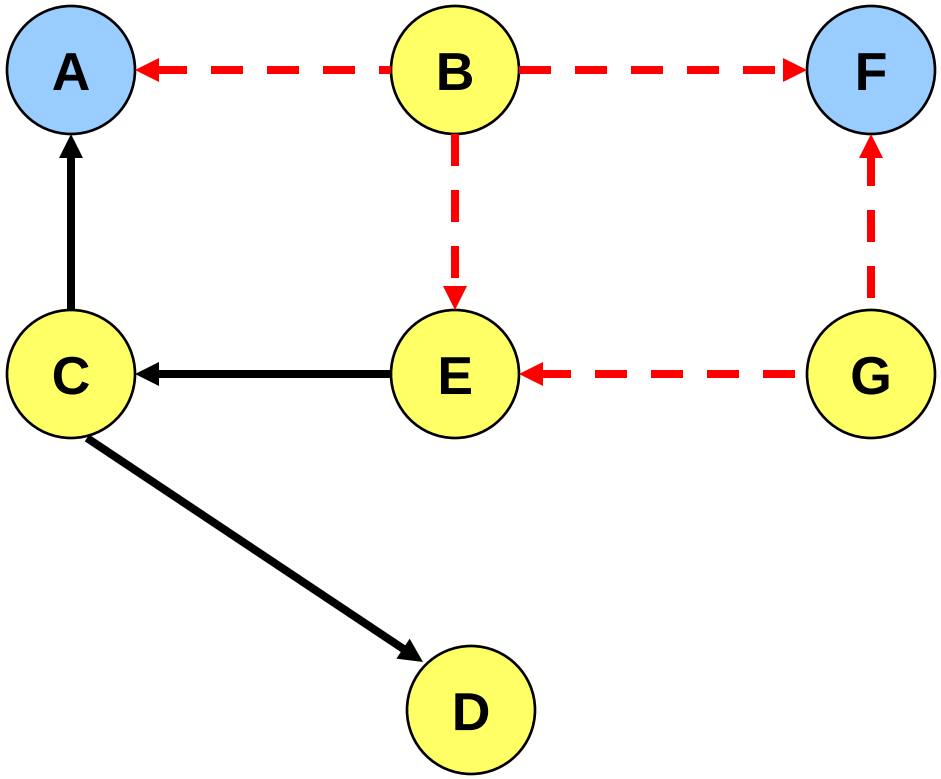
Link Reversal Algorithm



Represents a link that was reversed recently

Now nodes B and G have no outgoing links

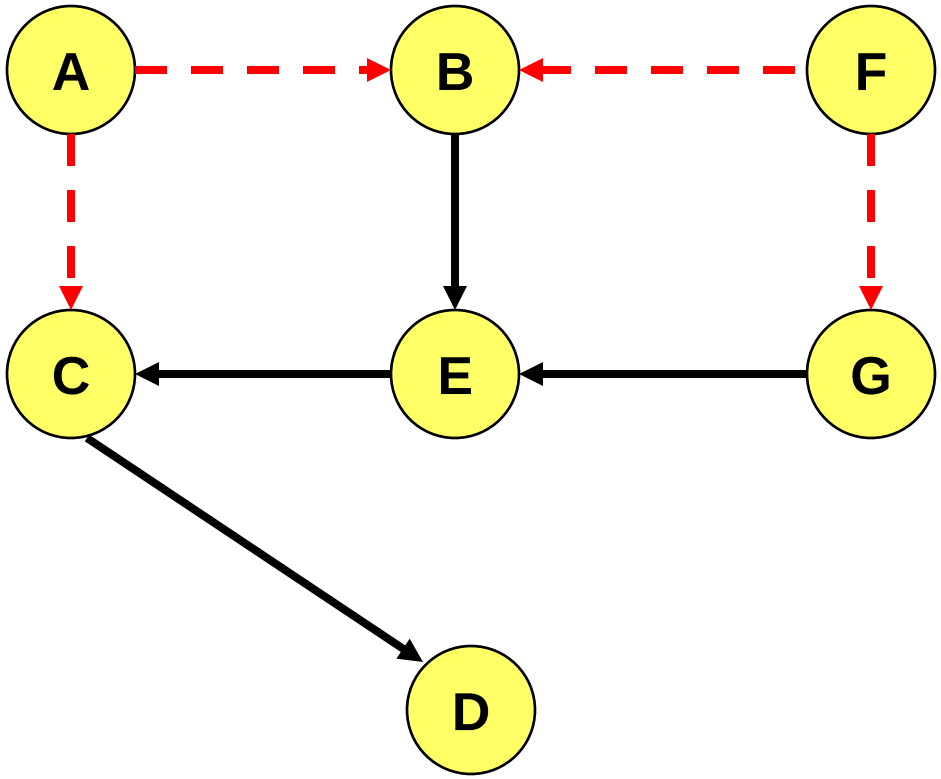
Link Reversal Algorithm



Represents a link that was reversed recently

Now nodes A and F have no outgoing links

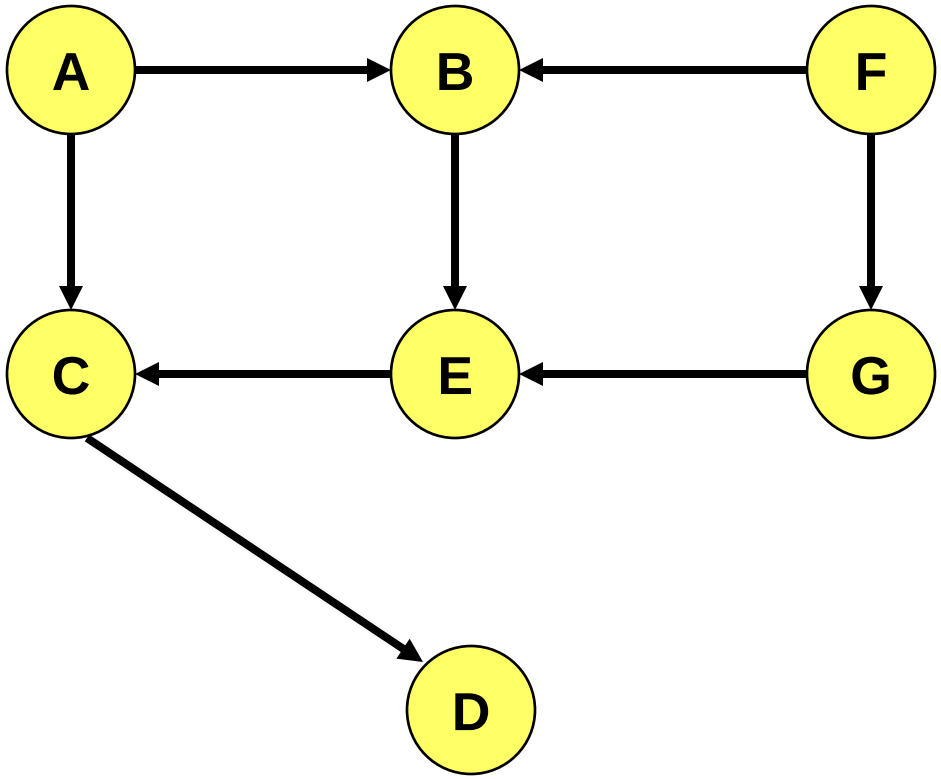
Link Reversal Algorithm



Represents a link that was reversed recently

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

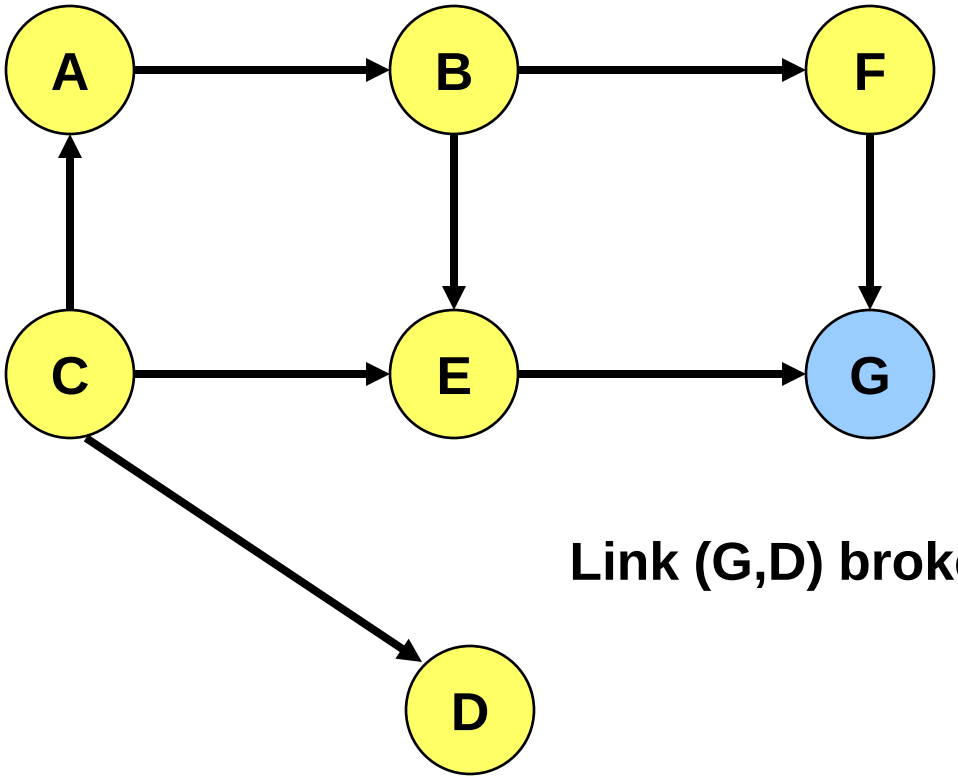
- Attempts to keep link reversals local to where the failure occurred
  - But this is not guaranteed
- **When the first packet is sent to a destination, the destination oriented DAG is constructed**
- The initial construction does result in **flooding of control packets**

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** [Gafni81]: 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*

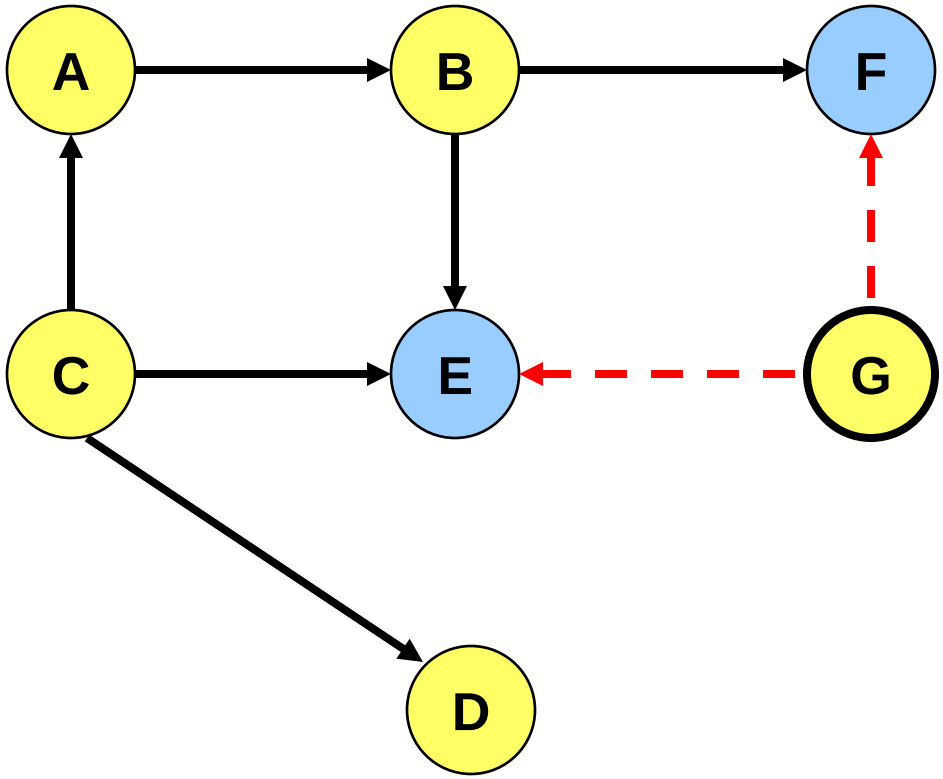
Partial Reversal Method



Link (G,D) broke

Node G has no outgoing links

Partial Reversal Method

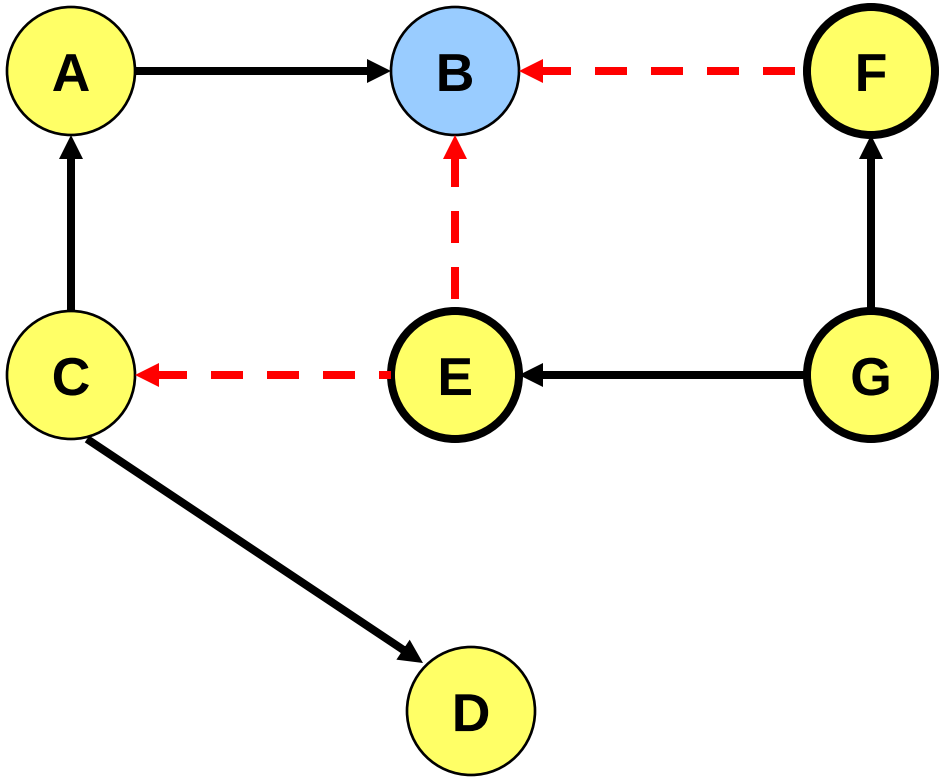


Represents a link that was reversed recently

Represents a node that has reversed links

Now nodes E and F have no outgoing links

Partial Reversal Method

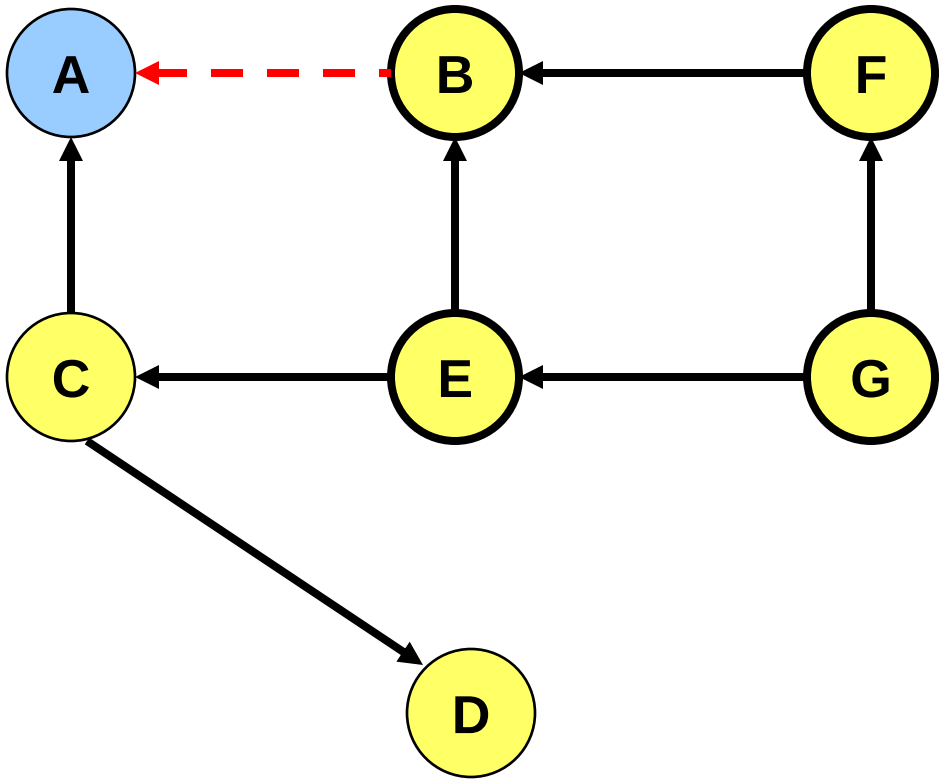


Represents a link that was reversed recently

Nodes E and F *do not* reverse links from node G

Now node B has no outgoing links

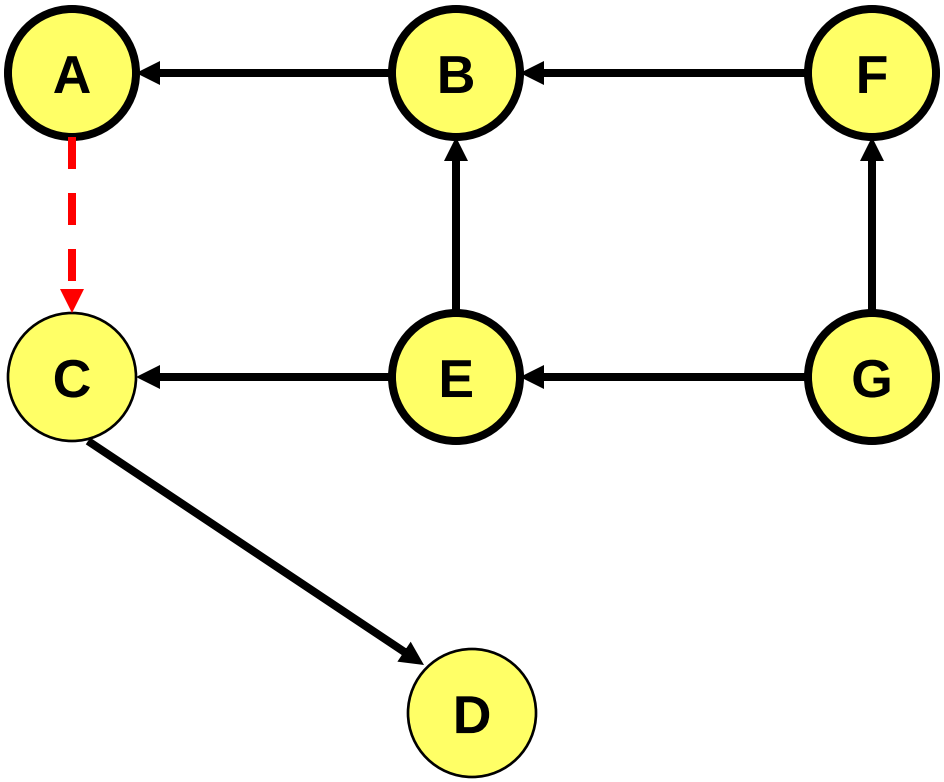
Partial Reversal Method



Represents a link that was reversed recently

Now node A has no outgoing links

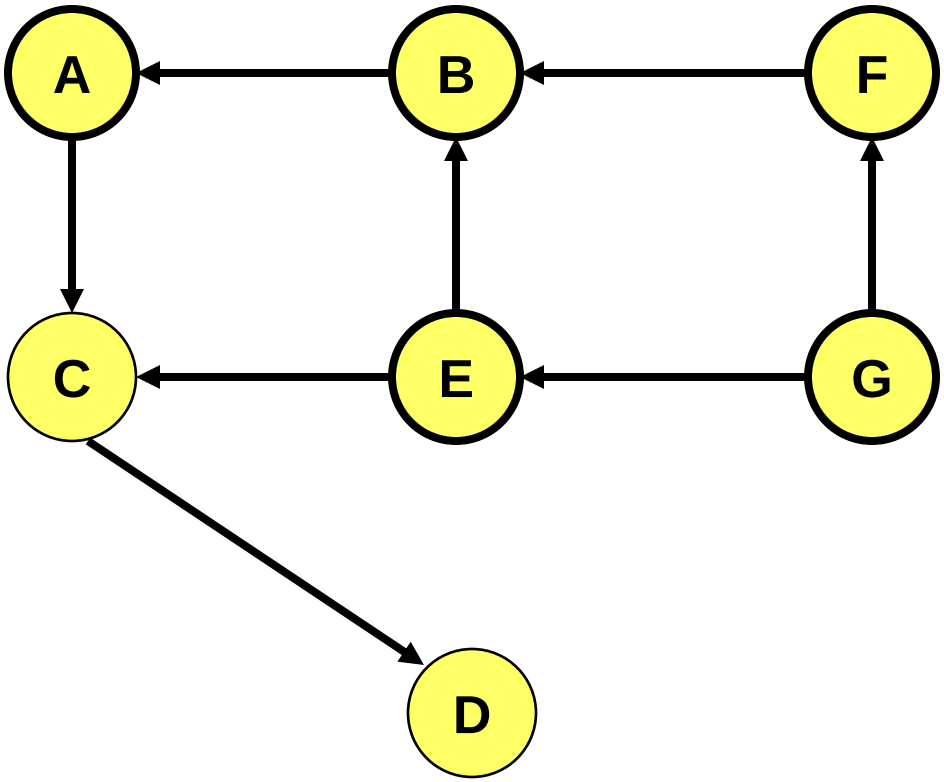
Partial Reversal Method



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

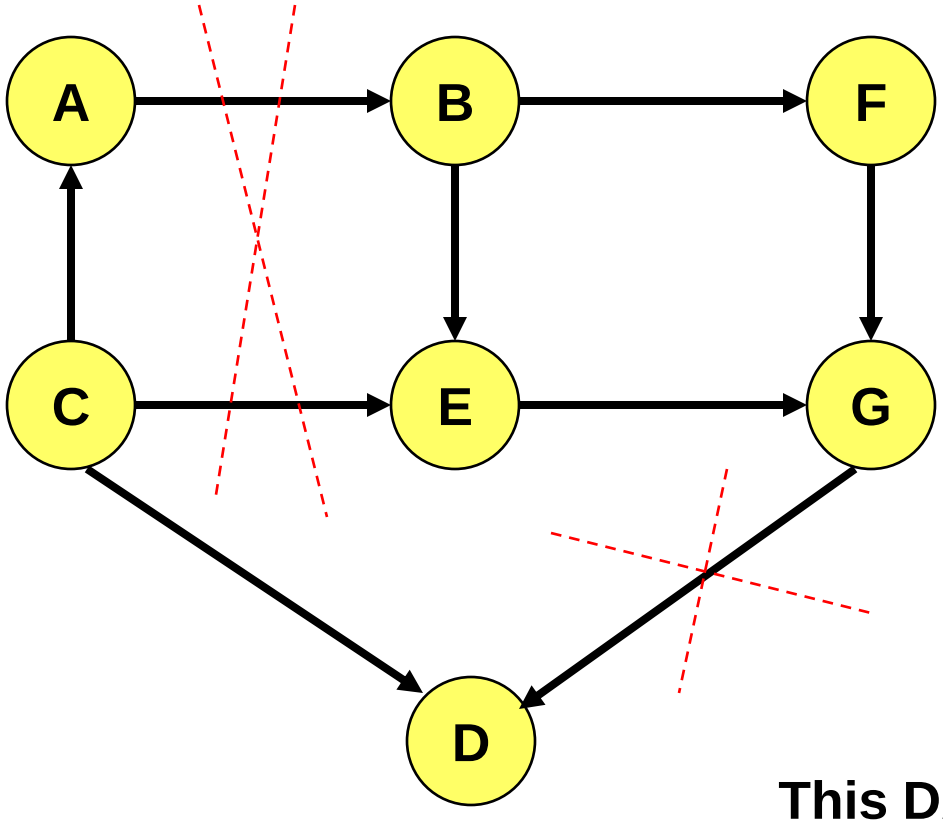
Link Reversal Methods: Advantages

- Link reversal methods attempt to limit updates to routing tables at nodes in the vicinity of a broken link
  - Partial reversal method tends to be better than full reversal method
- Each node may potentially have multiple routes to a destination

Link Reversal Methods: Disadvantage

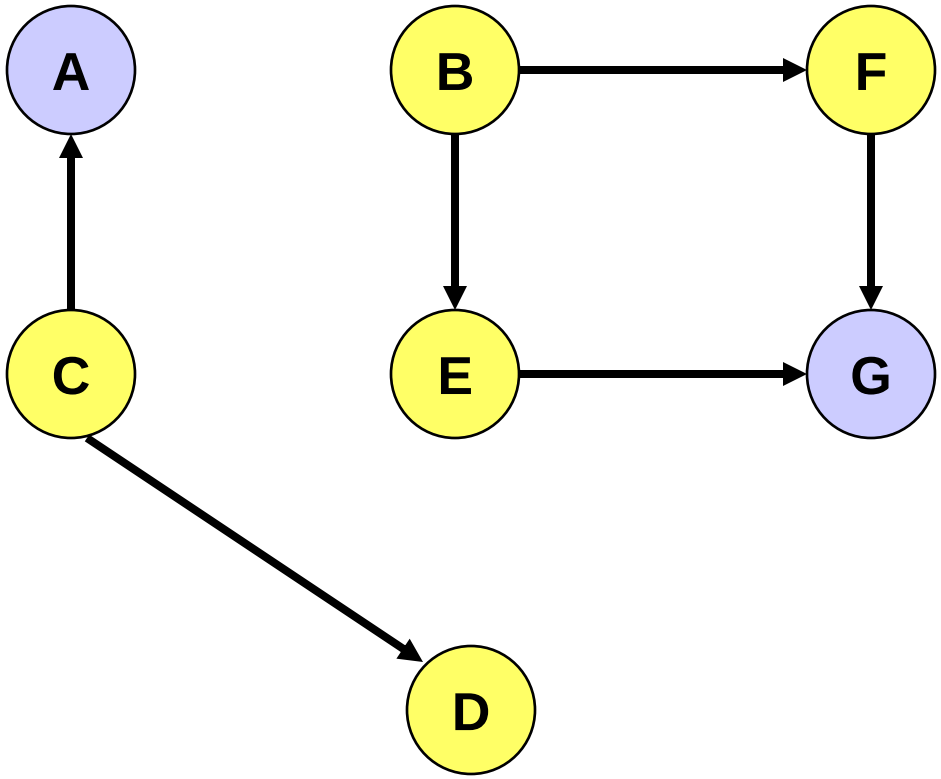
- Need a mechanism to detect link failure
  - hello messages may be used
  - but hello messages can add to contention
- If network is partitioned, link reversals continue indefinitely

Link Reversal in a Partitioned Network



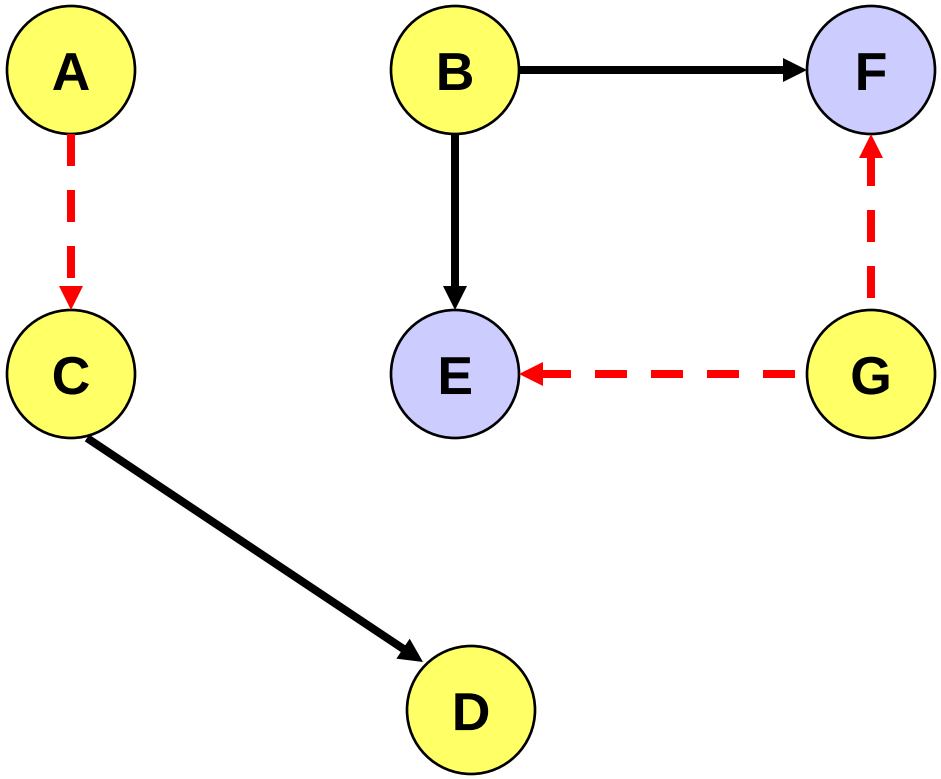
This DAG is for *destination node D*

Full Reversal in a Partitioned Network



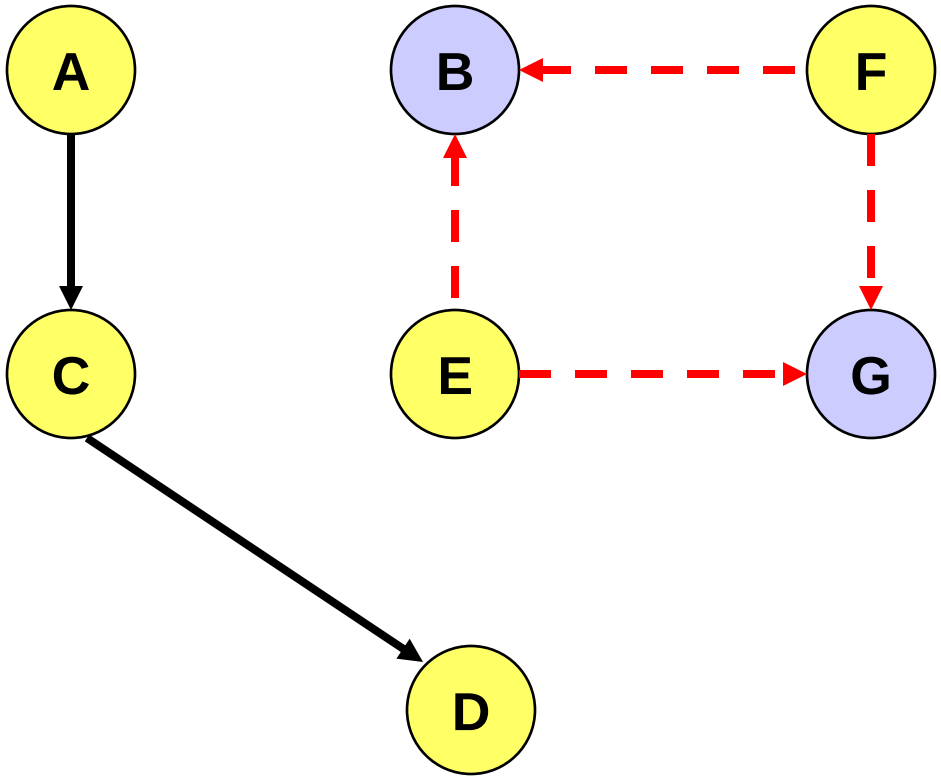
A and G do not have outgoing links

Full Reversal in a Partitioned Network



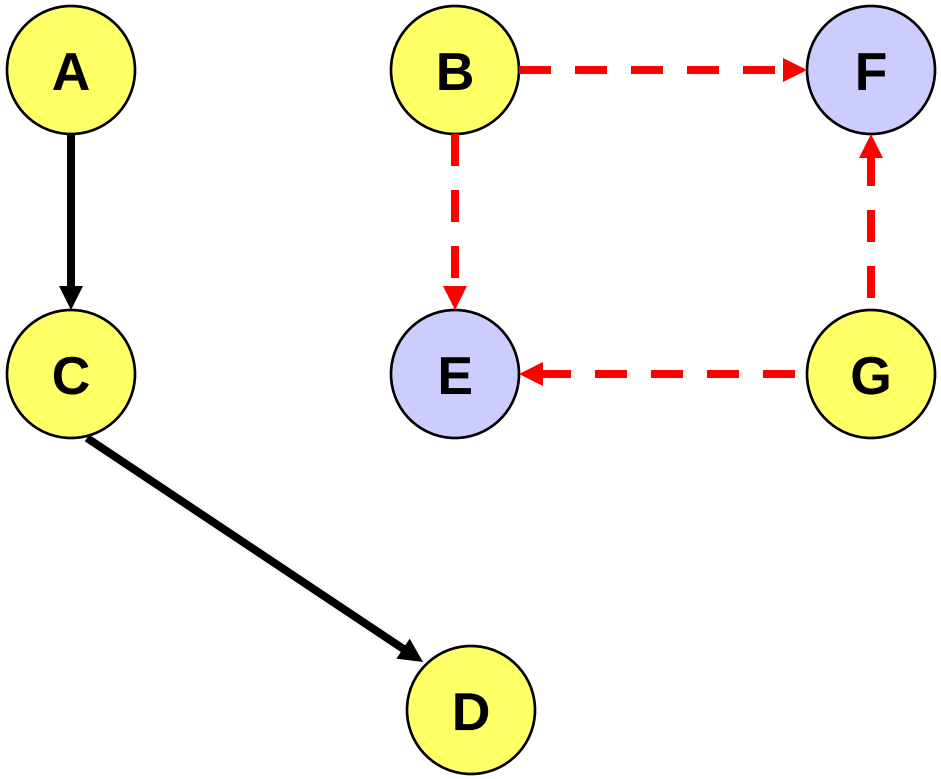
E and F do not have outgoing links

Full Reversal in a Partitioned Network



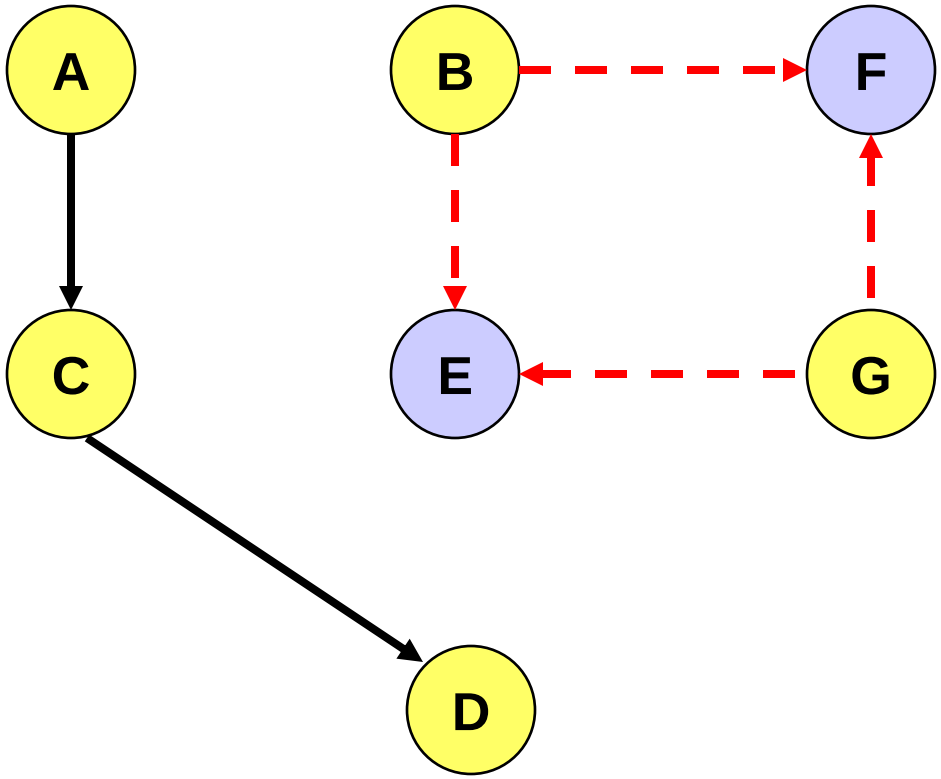
B and G do not have outgoing links

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

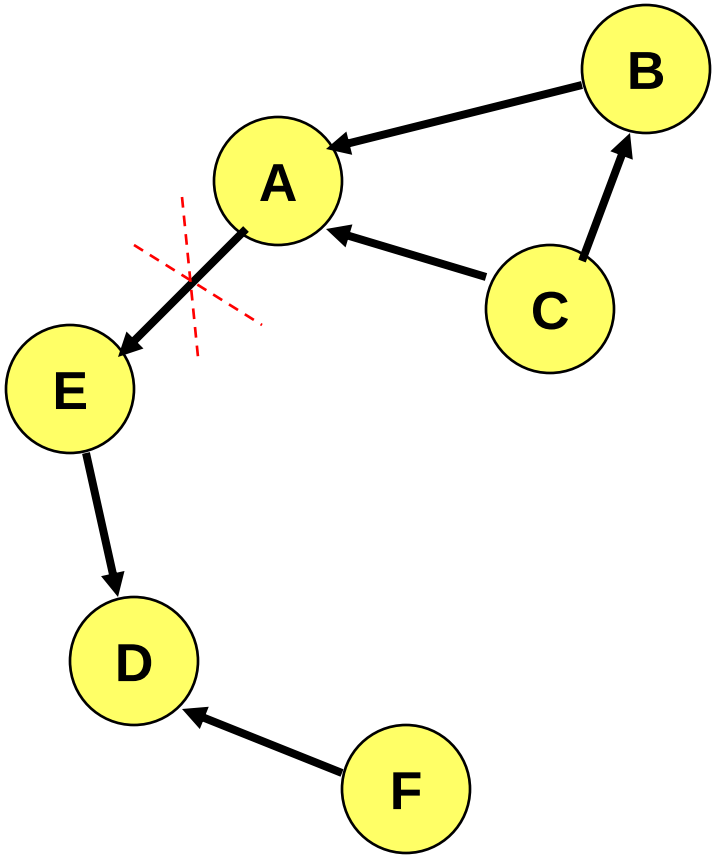
Similar scenario can occur with partial reversal method too

Temporally-Ordered Routing Algorithm (TORA) [Park97Infocom]

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

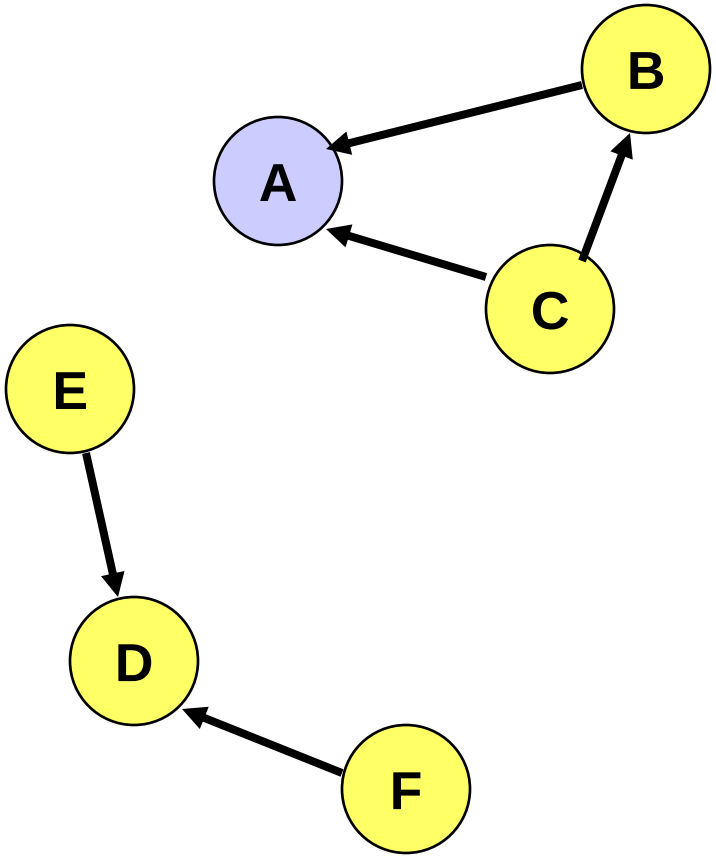
- When a partition is detected, all nodes in the partition are informed, and link reversals in that partition cease

Partition Detection in TORA



DAG for destination D

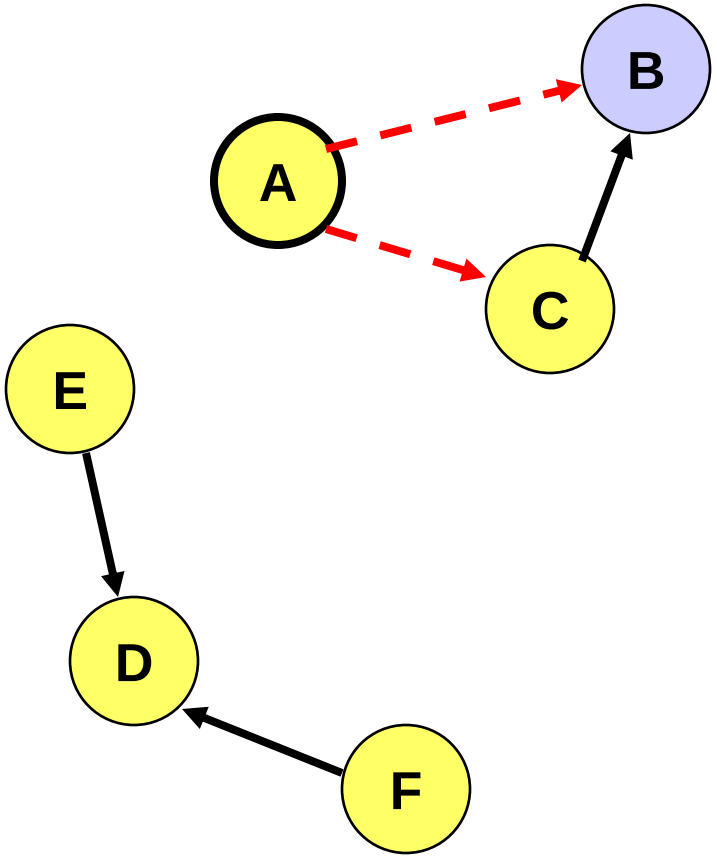
Partition Detection in TORA



TORA uses a modified partial reversal method

Node A has no outgoing links

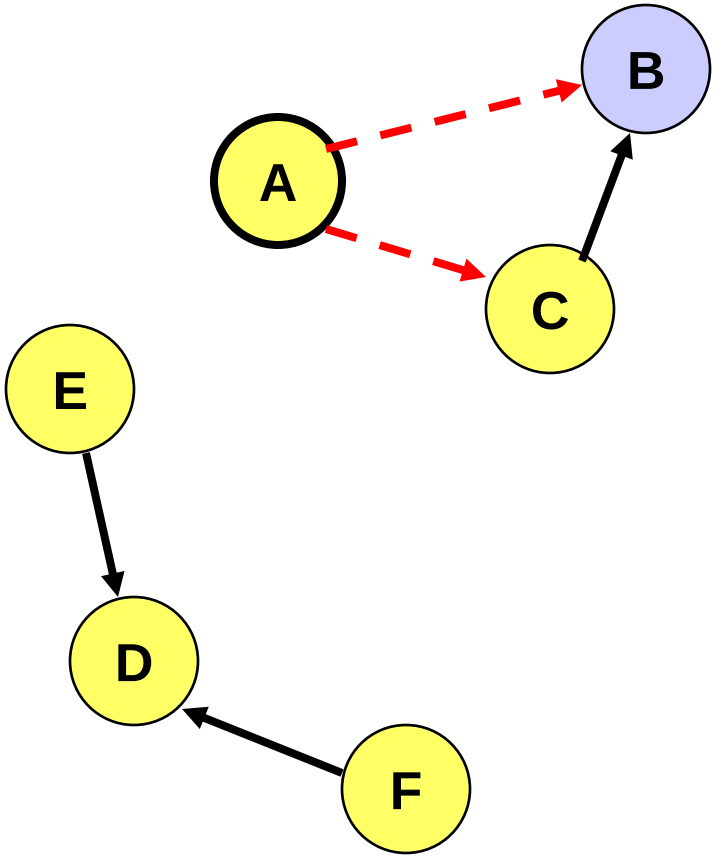
Partition Detection in TORA



TORA uses a modified partial reversal method

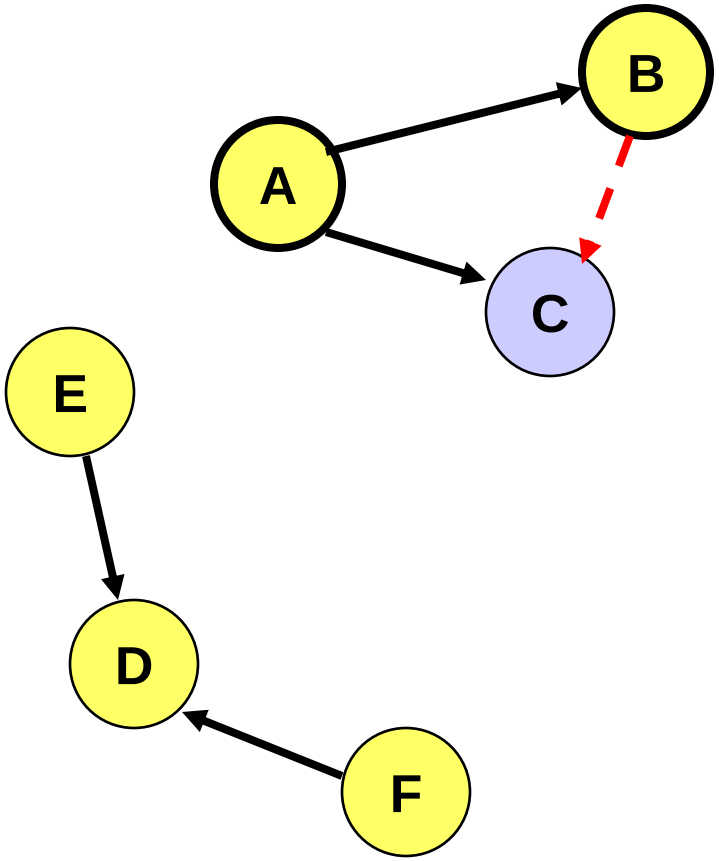
Node B has no outgoing links

Partition Detection in TORA



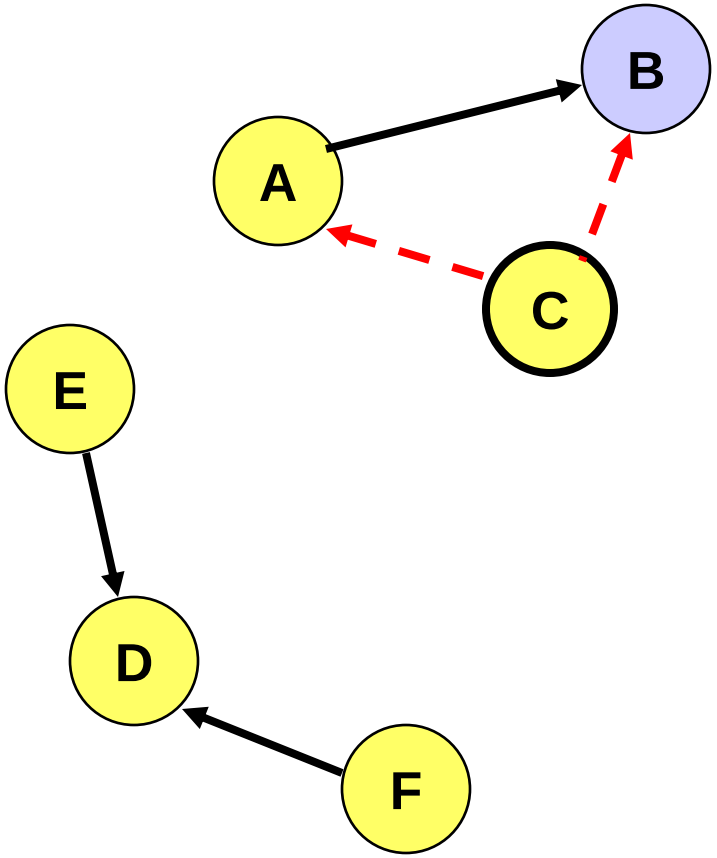
Node B has no outgoing links

Partition Detection in TORA



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

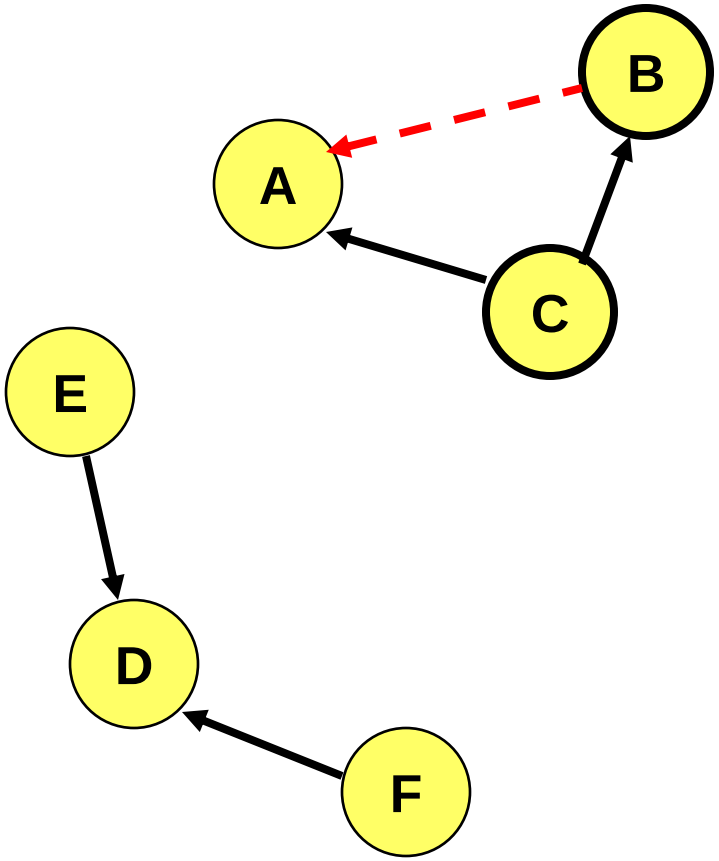
Partition Detection in TORA



Nodes A and B receive the reflection from node C

Node B now has no outgoing link

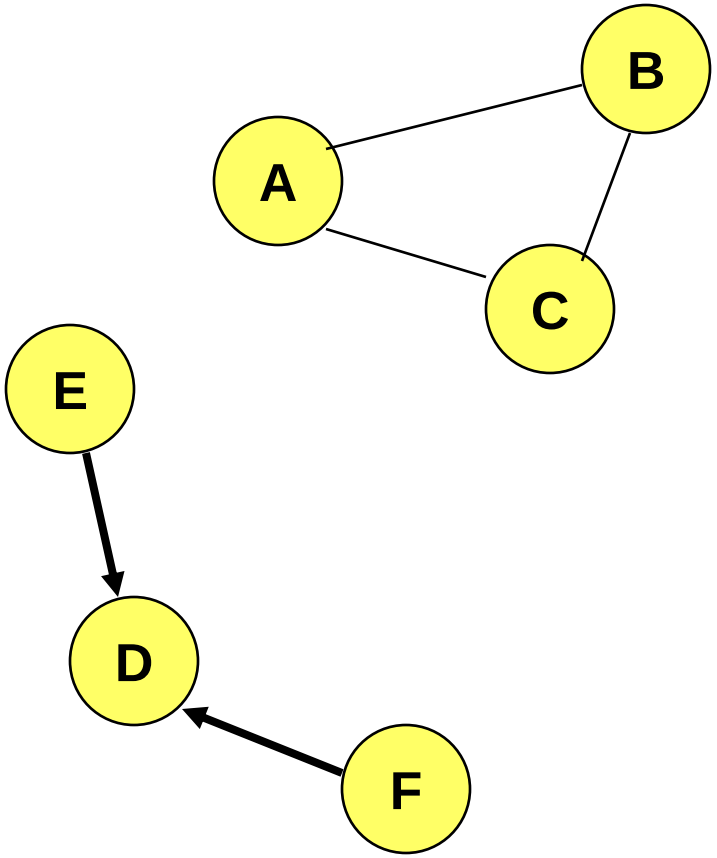
Partition Detection in TORA



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 in [Gafni81] 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 as dictated by [Gafni81]
- 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

TORA Design Decision

- One proposal for modifying TORA optionally allowed a more proactive behavior, such that a DAG would be maintained even if no node is attempting to transmit to the destination
- Moral of the story: The link reversal algorithm in [Gafni81] does not dictate a proactive or reactive response to link failure/repair
- Decision on reactive/proactive behavior should be made based on environment under consideration

So far ...

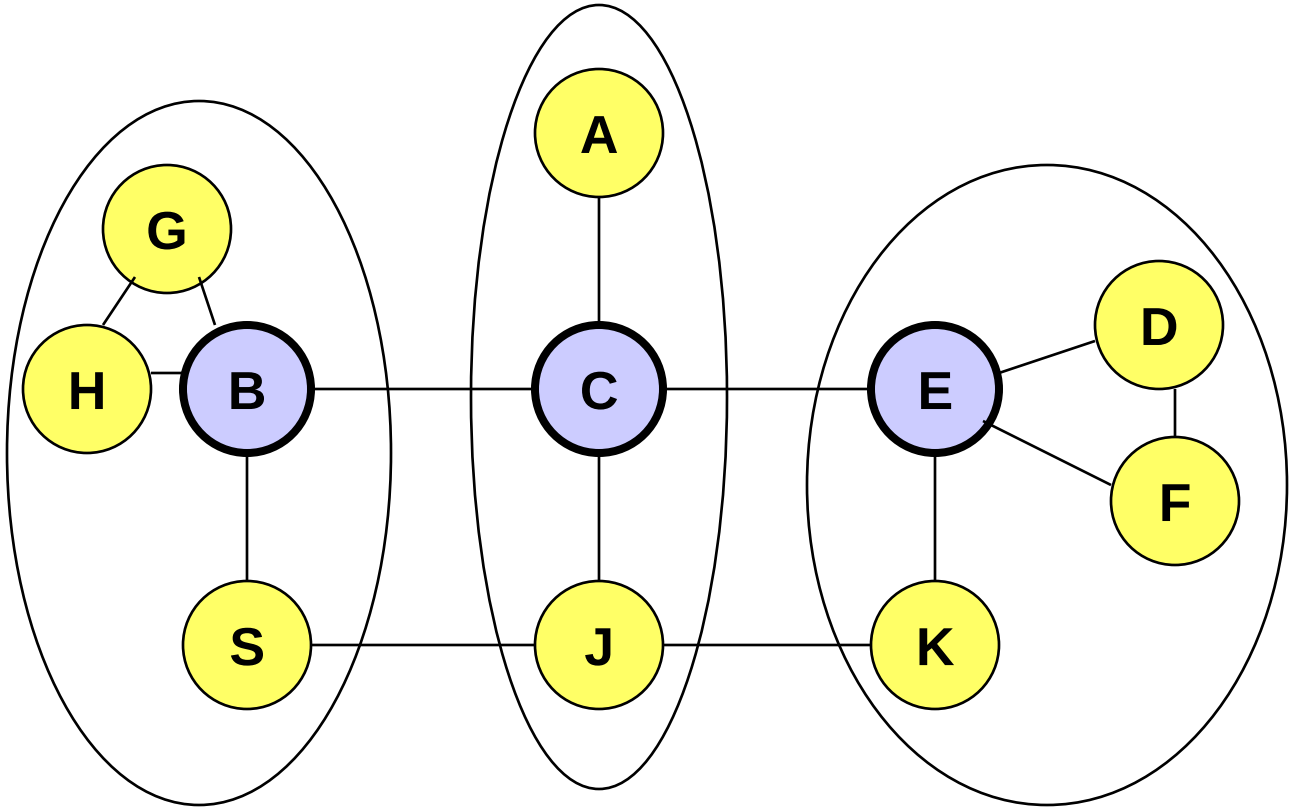
- All nodes had identical responsibilities
- Some schemes propose giving special responsibilities to a subset of nodes
  - Even if all nodes are physically identical
- **Core-based** schemes are examples of such schemes

Asymmetric Responsibilities

Core-Extraction Distributed Ad Hoc Routing (CEDAR) [Sivakumar99]

- A subset of nodes in the network is identified as the *core*
- Each node in the network must be adjacent to at least one node in the core
  - Each node picks one core node as its ***dominator*** **(or leader)**
- Core is determined by periodic message exchanges between each node and its neighbors
  - attempt made to keep the number of nodes in the core small
- Each core node determines paths to nearby core nodes by means of a localized broadcast
  - Each core node guaranteed to have a core node at <=3 hops

CEDAR: Core Nodes



A core node

Node E is the dominator for nodes D, F and K

Link State Propagation in CEDAR

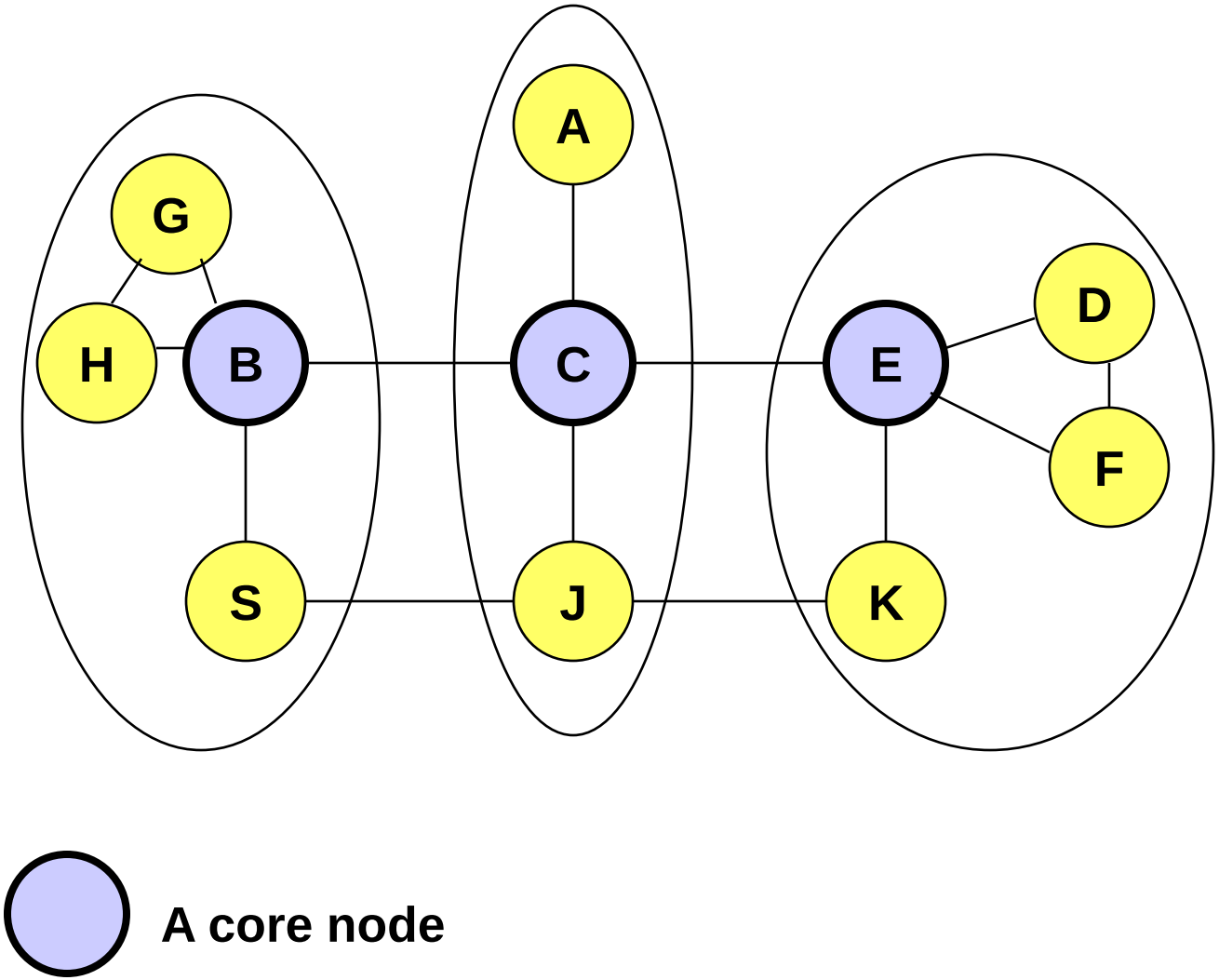
- The distance to which the state of a link is propagated in the network is a function of
  - whether the link is stable -- state of unstable links is not propagated very far
  - whether the link bandwidth is high or low -- only state of links with high bandwidth is propagated far
- Link state propagation occurs among core nodes
  - Link state information includes dominators of link end-points
- Each core node knows the state of local links and stable high bandwidth links far away

Route Discovery in CEDAR

When a node S wants to send packets to destination D

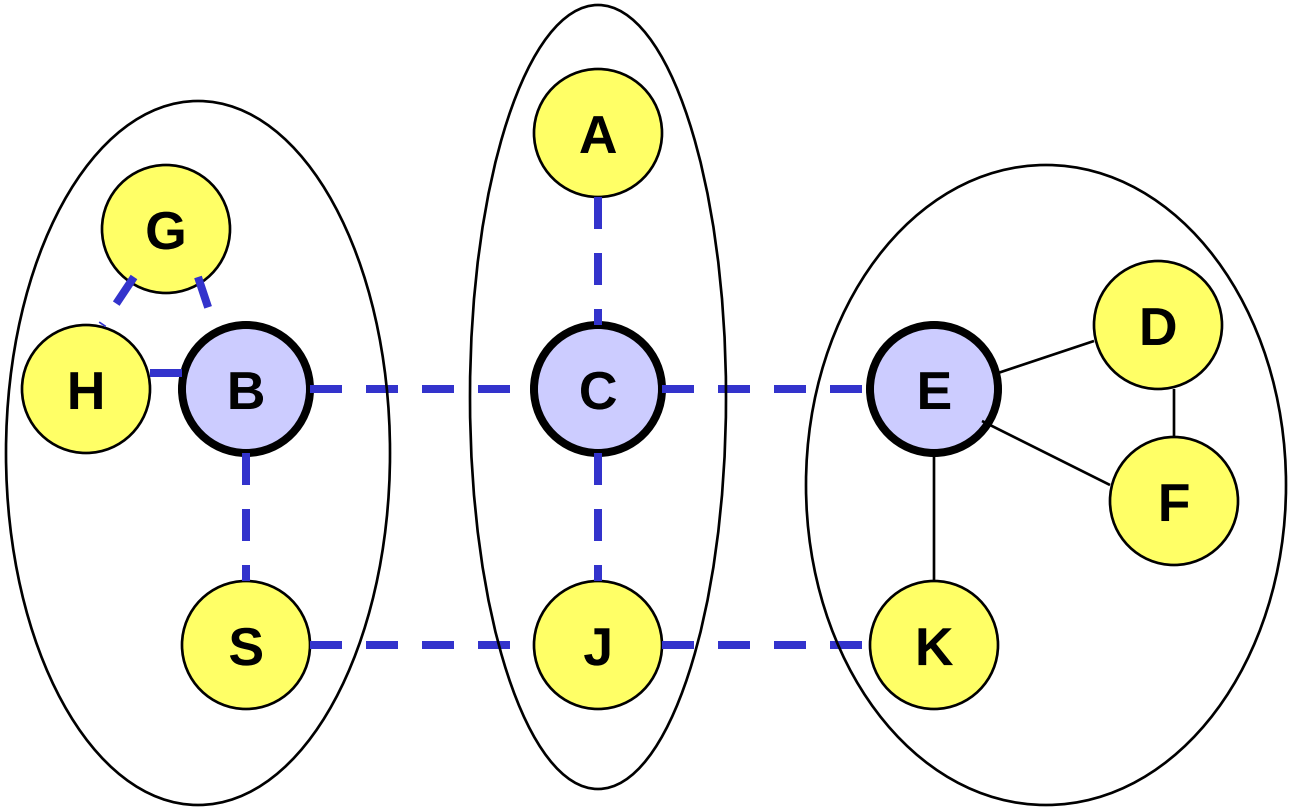
- Node S informs its dominator core node B
- Node B finds a route in the core network to the core node E which is the dominator for destination D
  - This is done by means of a DSR-like route discovery (but somewhat optimized) process among the core nodes
- Core nodes on the above route then build a route from S to D using locally available link state information
- Route from S to D may or may not include core nodes

CEDAR: Core Maintenance



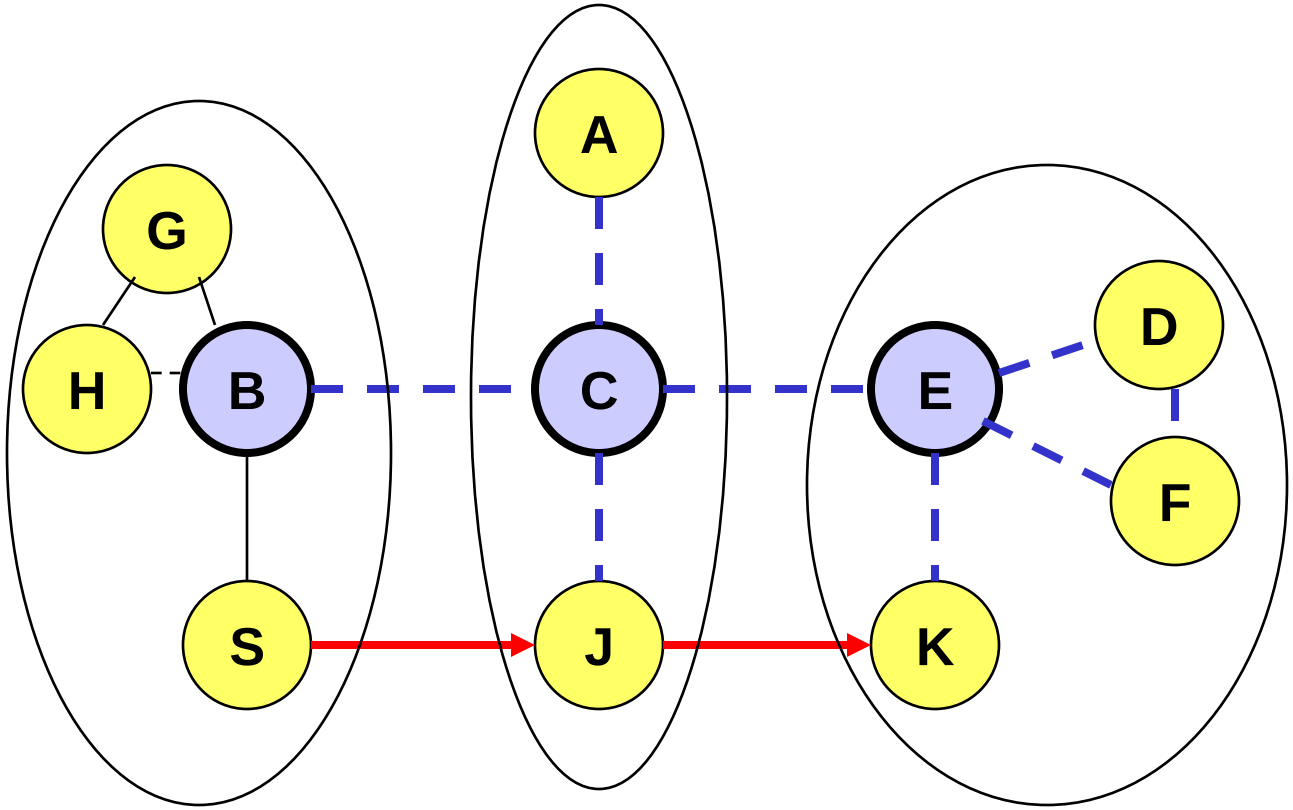
A core node

Link State at Core Nodes



Links that node B is aware of

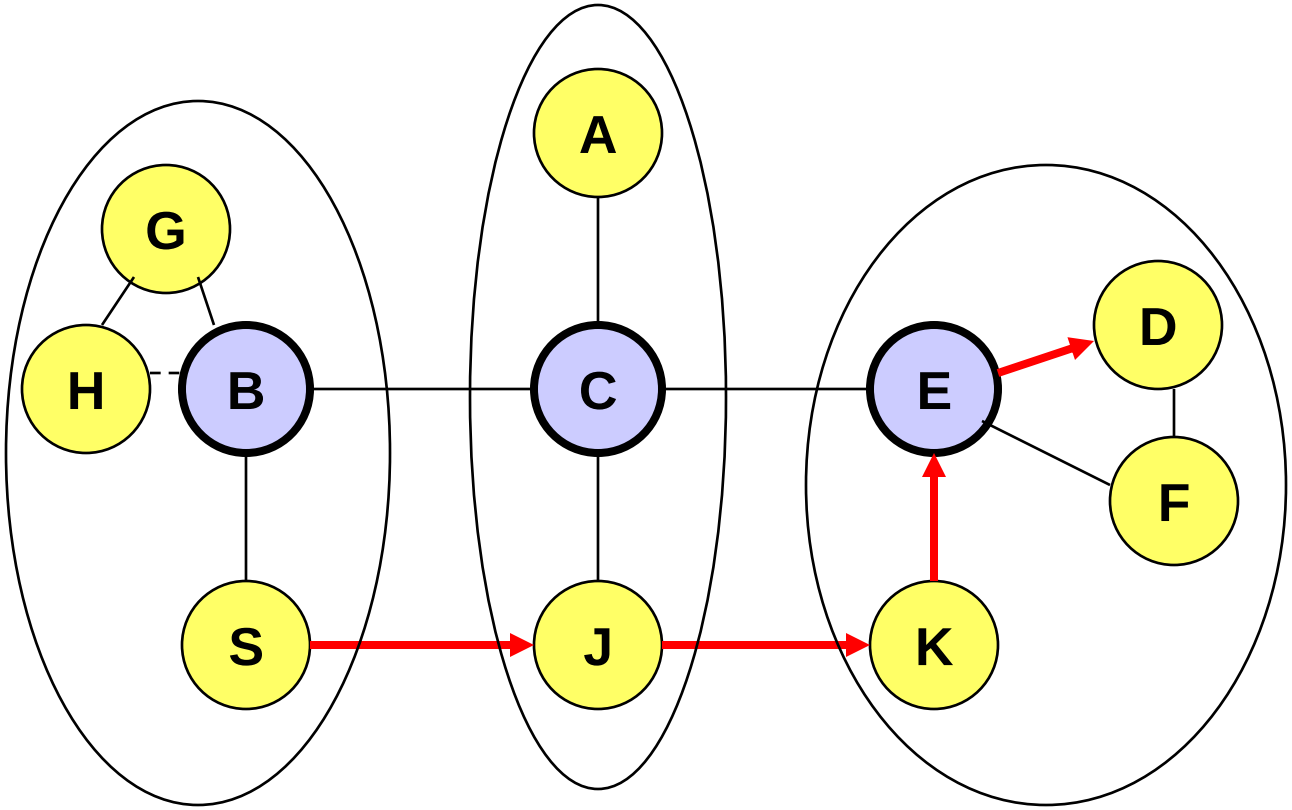
CEDAR Route Discovery



Partial route constructed by B

Links that node C is aware of

CEDAR Route Discovery



Complete route -- last two hops determined by node C

CEDAR

- Advantages
  - Route discovery/maintenance duties limited to a small number of core nodes
  - Link state propagation a function of link stability/quality
- Disadvantages
  - Core nodes have to handle additional traffic, associated with route discovery and maintenance

Asymmetric Responsibilities: Cluster-Based Schemes

- Some cluster-based schemes have also been proposed
- In some cluster-based schemes, a leader is elected for each *cluster* of node
- The leader often has some special responsibilities
- Different schemes may differ in
  - how clusters are determined
  - the way cluster head (leader) is chosen
  - duties assigned to the cluster head

Proactive Protocols

- Most of the schemes discussed so far are reactive
- Proactive schemes based on distance-vector and link-state mechanisms have also been proposed

Link State Routing

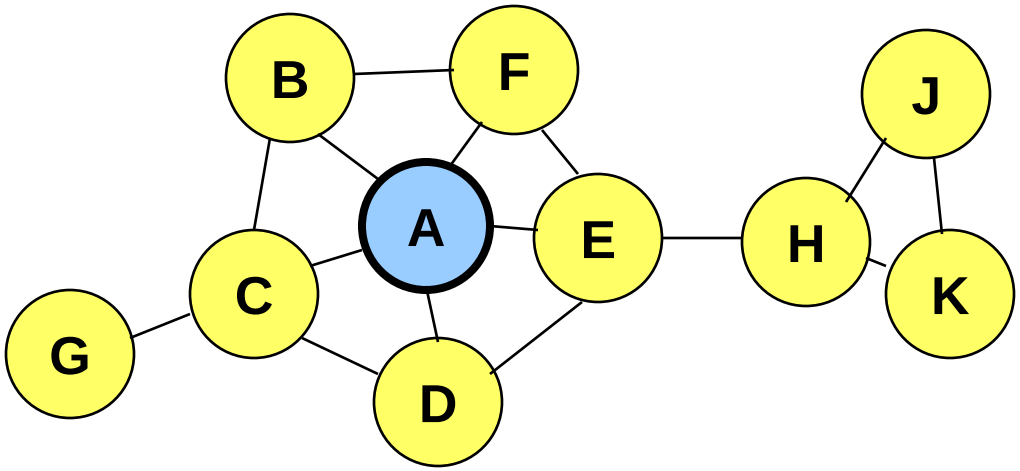
- Each node periodically floods status of its links
- Each node re-broadcasts link state information received from its neighbor
- Each node keeps track of link state information received from other nodes
- Each node uses above information to determine next hop to each destination

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 , other neighbors simply use the info without forwarding it.*
- 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)

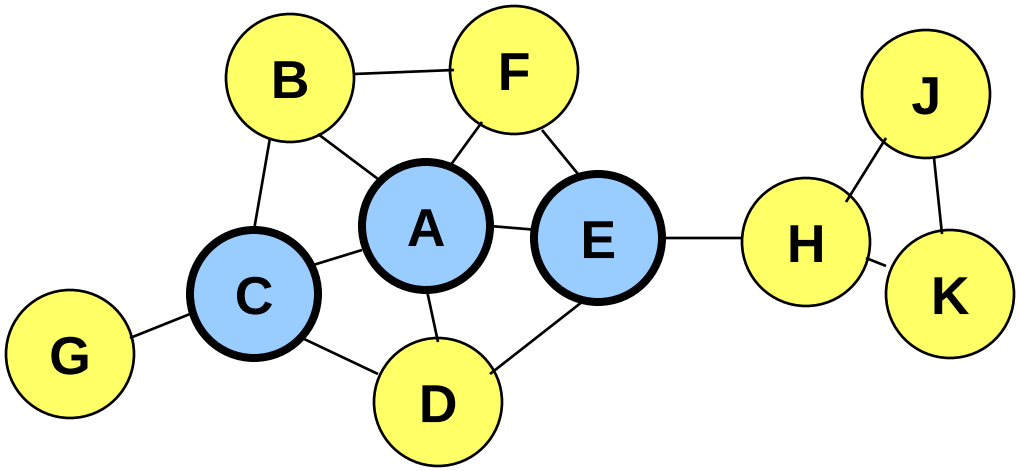
- Nodes C and E are multipoint relays of node A



Node that has broadcast state information from A

Optimized Link State Routing (OLSR)

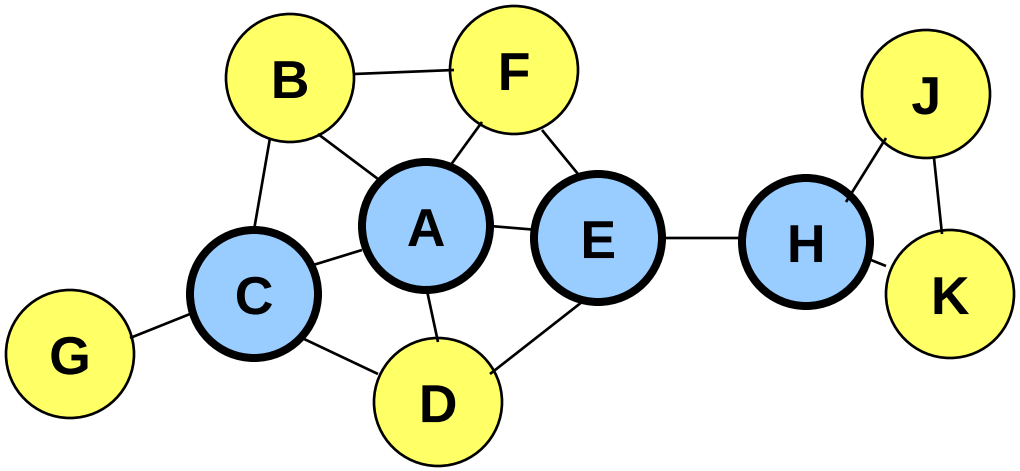
- Nodes **C** and **E** forward **information** received from A, but **B**, **F** and **D** simply use the info without forwarding it



Node that has broadcast state information from A

Optimized Link State Routing (OLSR)

- Nodes E and K (or J) 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 flood itself is restricted to links connecting nodes to respective multipoint relays
- Routes used by OLSR only include multipoint relays as intermediate nodes

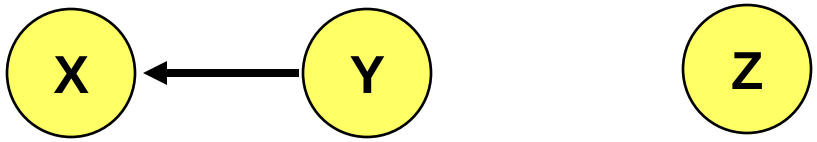
Destination-Sequenced Distance-Vector (DSDV)

- 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)

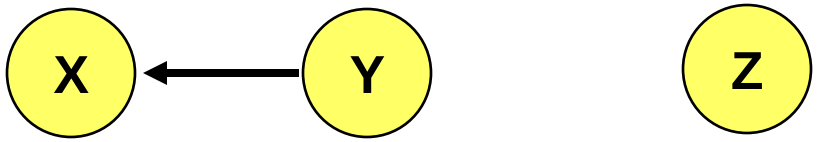
- Assume that node X receives routing information from Y about a route to node Z



- 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)

DSDV

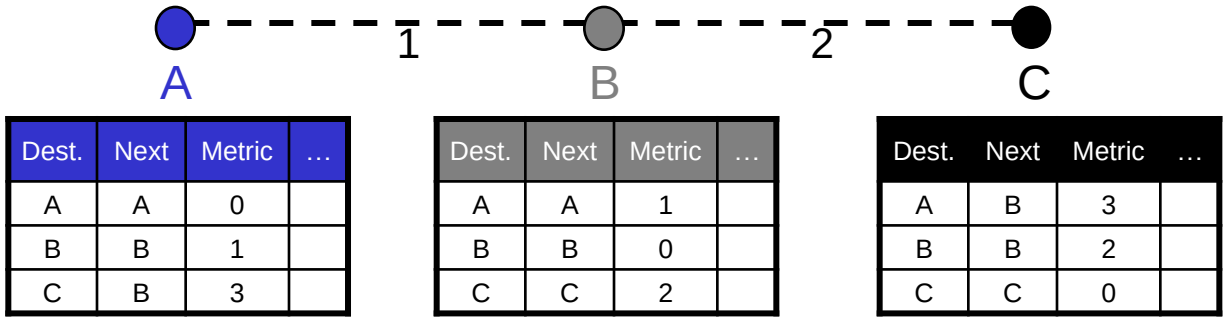
- DSDV (Destination Sequenced Distance Vector)
  - Each node sends and responds to routing control message the same way
  - No hierarchical structure
  - Avoids the resource costs involved in maintaining high-level structure
  - Scalability may become an issue in larger networks

DSDV

- Basic Routing Protocol
  - known also as Distributed Bellman-Ford or RIP
- Every node maintains a routing table
  - all available destinations
  - the next node to reach to destination
  - the number of hops to reach the destination
- Periodically send table to all neighbors to maintain topology
- Bi-directional links are required!

DSDV

Traditional Distance vector tables



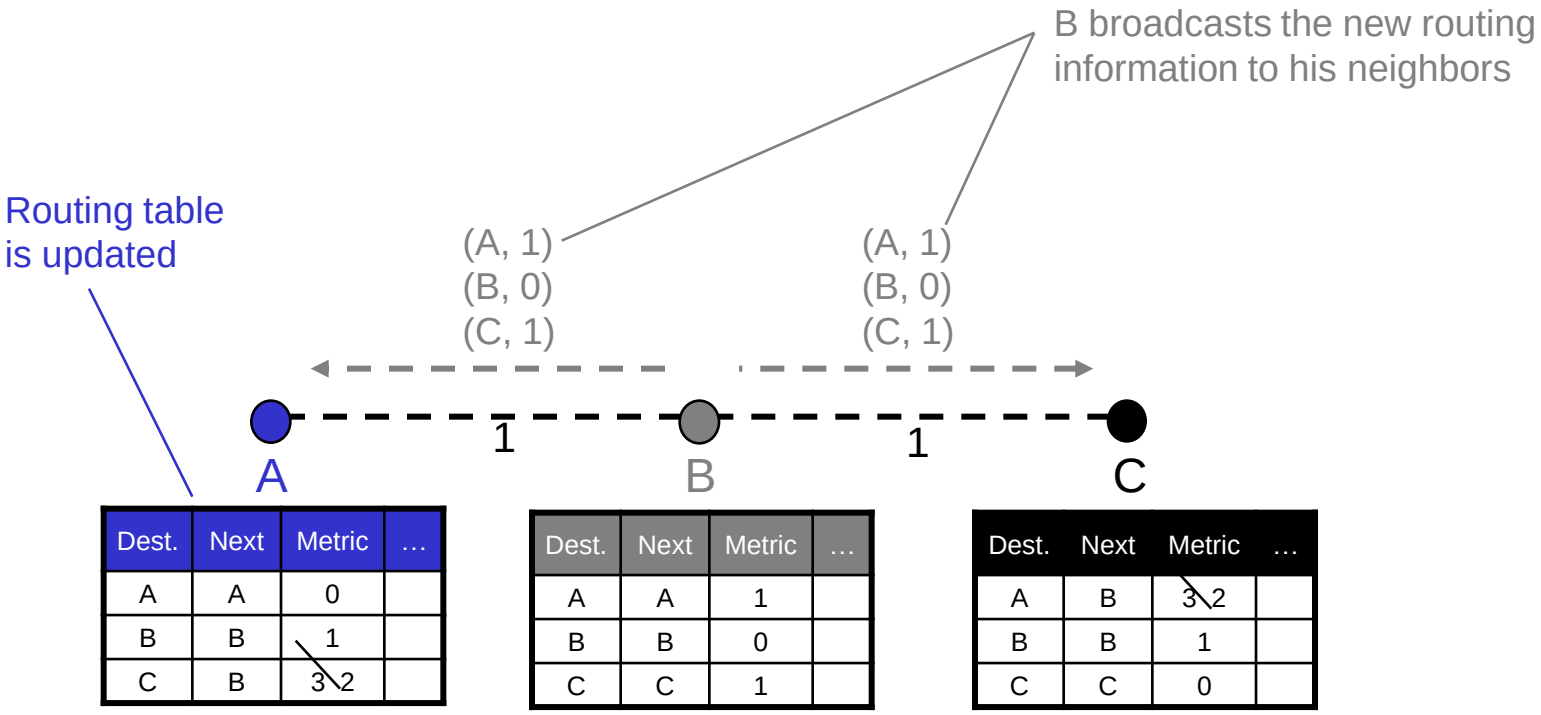
Dest.	Next	Metric	…
A	A	0	
B	B	1	
C	B	3	

Dest.	Next	Metric	…
A	A	1	
B	B	0	
C	C	2	

Dest.	Next	Metric	…
A	B	3	
B	B	2	
C	C	0	

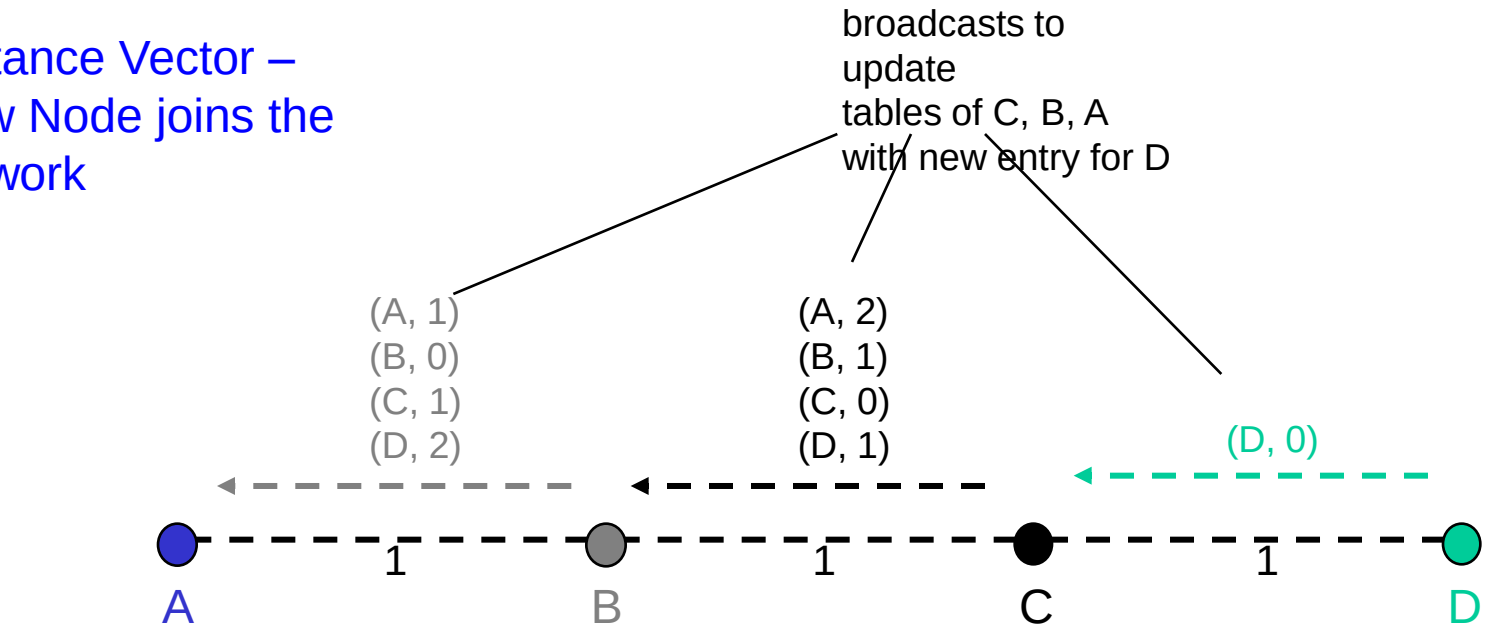
DSDV

Distance Vector Updates



DSDV

Distance Vector – New Node joins the network



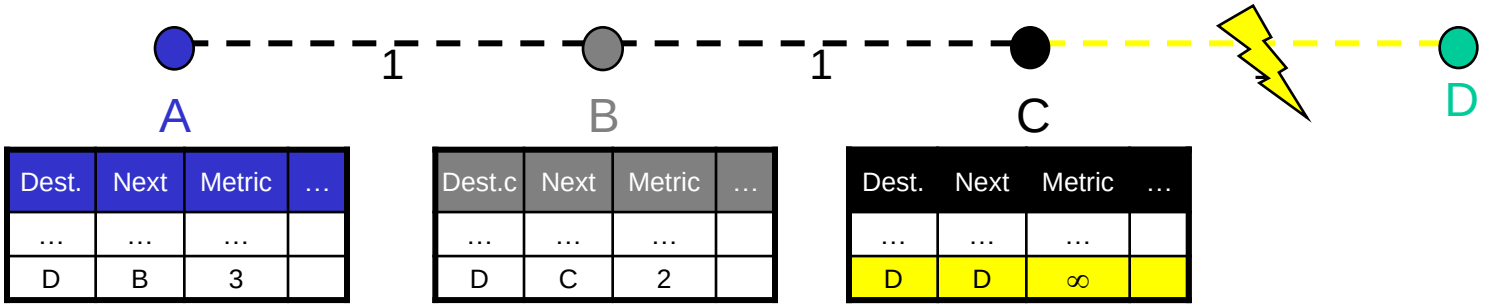
Dest.	Next	Metric	…
A	A	0	
B	B	1	
C	B	2	
D	B	3	

Dest.	Next	Metric	…
A	A	1	
B	B	0	
C	C	1	
D	C	2	

Dest.	Next	Metric	…
A	B	2	
B	B	1	
C	C	0	
D	D	1	

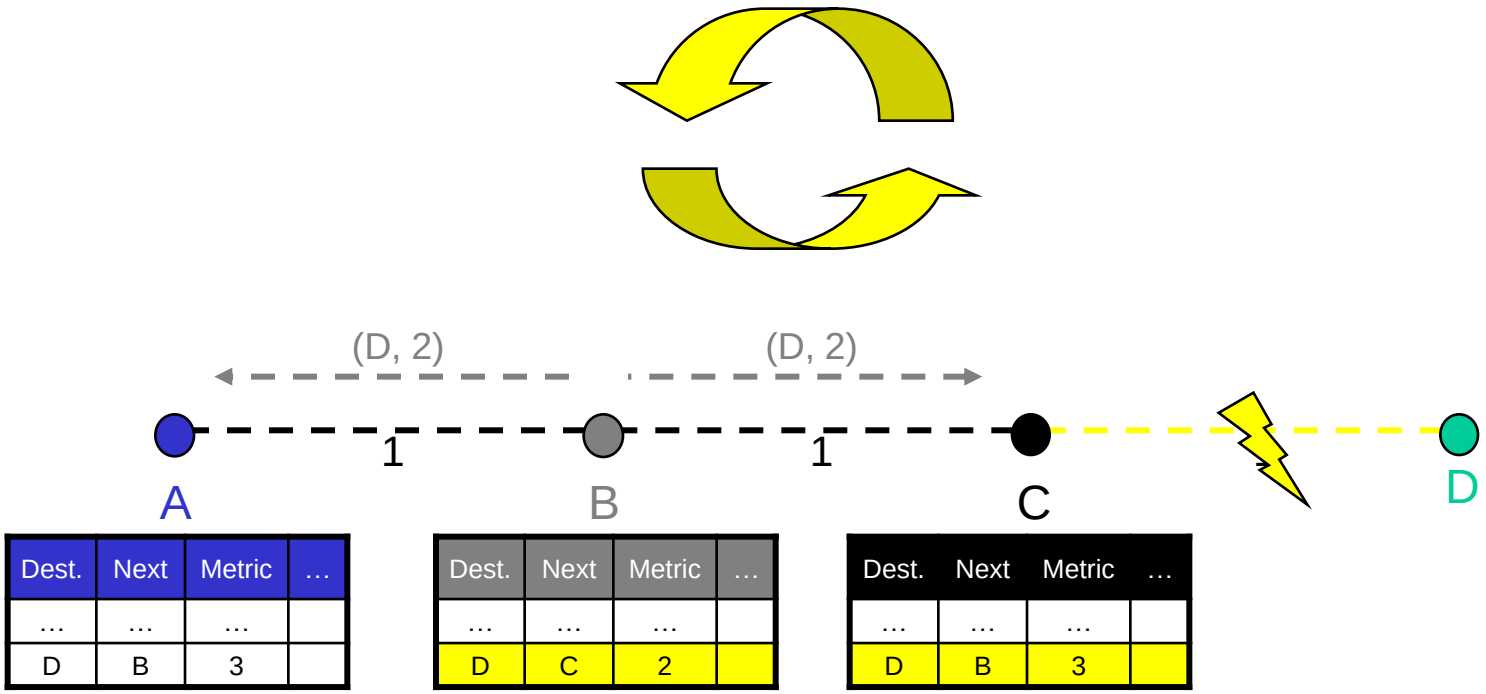
DSDV

Distance Vector – Broken link



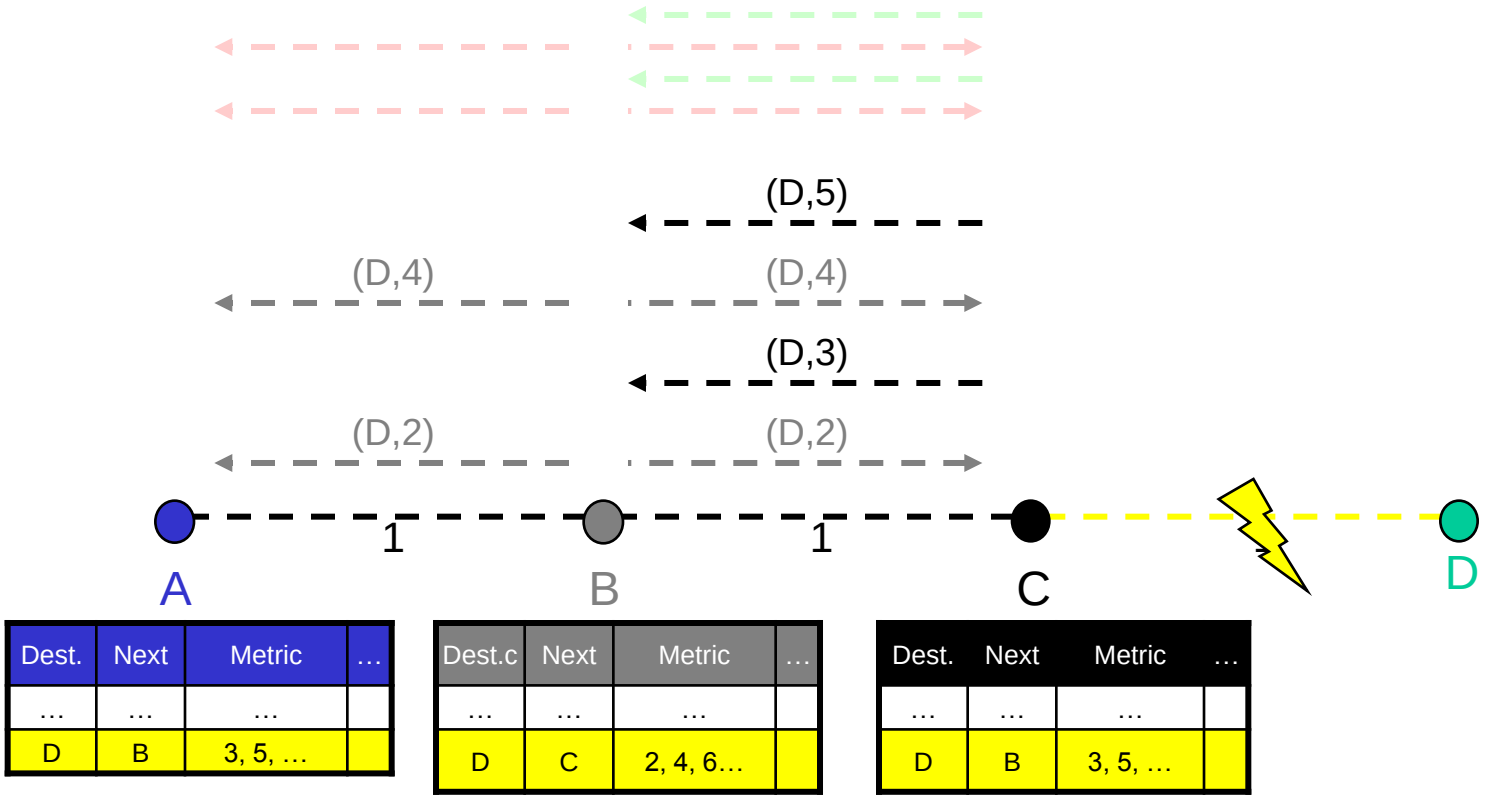
DSDV

Distance Vector - Loops



DSDV

Distance vector - Count to Infinity



DSDV

- Traditional Distance Vector are not suited for ad-hoc networks!
  - Loops
    - Bandwidth reduction in network
    - Unnecessary work for loop nodes
  - Count to Infinity
    - Very slow adaptation to topology changes.
- Solution -> Introduce destination sequence numbers

DSDV

- DSDV keeps the simplicity of traditional Distance Vector Protocols
- DSDV need to guarantee loop freeness
  - New Table Entry for Destination Sequence Number
- DSDV need to allow fast reaction to topology changes
  - Make immediate route advertisement on significant changes in routing table
  - but wait with advertising of unstable routes (damping fluctuations)

DSDV

- Features introduced in DSDV
  - **Sequence number** originated from destination. Ensures loop freeness.
  - **Install Time** when entry was made (used to delete stale entries from table.
  - S**table Data** Pointer to a table holding information on how stable a route is. Used to damp fluctuations in network.

Destination	Next	Metric	Seq. Nr	Install Time	Stable Data
A	A	0	A-550	001000	Ptr_A
B	B	1	B-102	001200	Ptr_B
C	B	3	C-588	001200	Ptr_C
D	B	4	D-312	001200	Ptr_D

DSDV – Route Advertisement

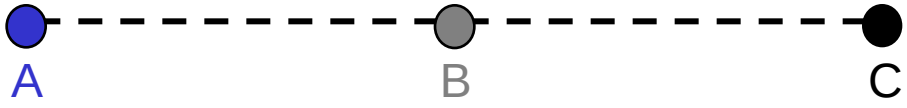
- Advertise to each neighbor own routing information
  - Destination Address
  - Metric = Number of Hops to Destination
  - Destination Sequence Number
  - Other info (e.g. hardware addresses)
- Rules to set sequence number information
  - On each advertisement increase own destination sequence number (use only even numbers)
  - If a node is no more reachable (timeout) increase sequence number of this node by 1 (odd sequence number) and set metric = ∞.

DSDV – Route Selection

- Update information is compared to own routing table
  1. Select route with higher destination sequence number (This ensure to use always newest information from destination)
  2. Select the route with better metric when sequence numbers are equal.

DSDV

DSDV Tables



Dest.	Next	Metric	Seq
A	A	0	A-550
B	B	1	B-100
C	B	2	C-586

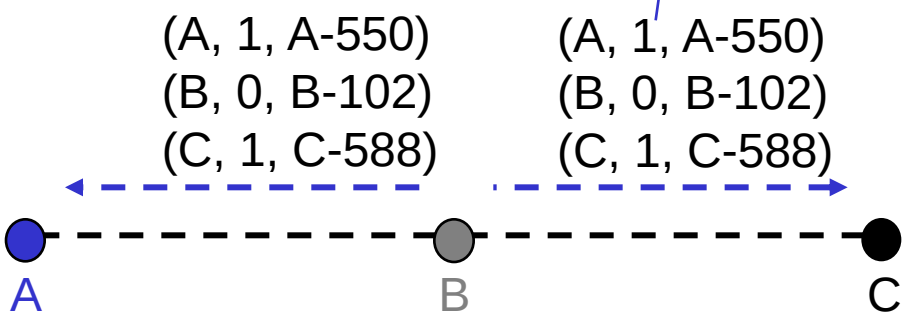
Dest.	Next	Metric	Seq
A	A	1	A-550
B	B	0	B-100
C	C	2	C-588

Dest.	Next	Metric	Seq.
A	B	1	A-550
B	B	2	B-100
C	C	0	C-588

DSDV

DSDV Route Advertisement

B increases Seq.Nr from 100 -> 102
B broadcasts routing information to Neighbors A, C including destination sequence numbers



Dest.	Next	Metric	Seq
A	A	0	A-550
B	B	1	B-102
C	B	2	C-588

Dest.	Next	Metric	Seq
A	A	1	A-550
B	B	0	B-102
C	C	1	C-588

Dest.	Next	Metric	Seq.
A	B	2	A-550
B	B	1	B-102
C	C	0	C-588

DSDV

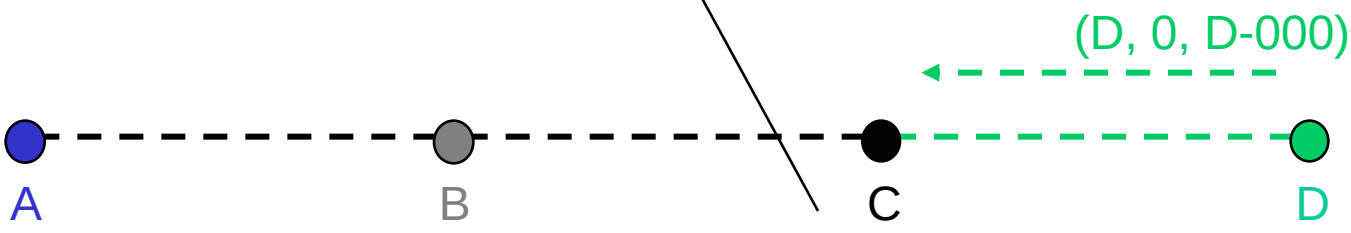
- DSDV Respond to topology changes
  - Immediate advertisements
    - Information on new routes, broken Links, metric change is immediately propagated to neighbors.
  - Full/Incremental Update:
    - Full Update: Send all routing information from own table. (May require multiple packets)
    - Incremental Update: Send only entries that has changed. (Make it fit into one single packet)

DSDV

When new node joins the network

2. Insert entry for D with sequence number D-000. Then immediately broadcast own table.

1. D broadcast for first time Send Sequence number D-000.



Dest.	Next	Metric	Seq.
A	A	0	A-550
B	B	1	B-104
C	B	2	C-590

Dest.	Next	Metric	Seq.
A	A	1	A-550
B	B	0	B-104
C	C	1	C-590

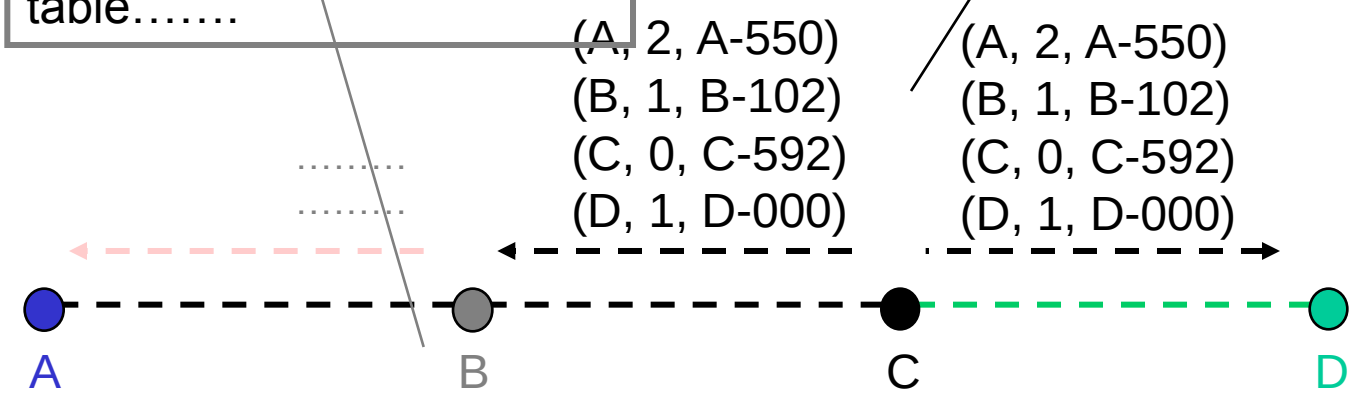
Dest.	Next	Metric	Seq.
A	B	2	A-550
B	B	1	B-104
C	C	0	C-590
D	D	1	D-000

DSDV

(New node (cont.)

3. C increases its sequence number to C-592 then broadcasts its new table.

4. B gets this new information and updates its table…….



Dest.	Next	Metric	Seq.
A	A	0	A-550
B	B	1	B-104
C	B	2	C-590

Dest.	Next	Metric	Seq.
A	A	1	A-550
B	B	0	B-102
C	C	1	C-592
D	C	2	D-000

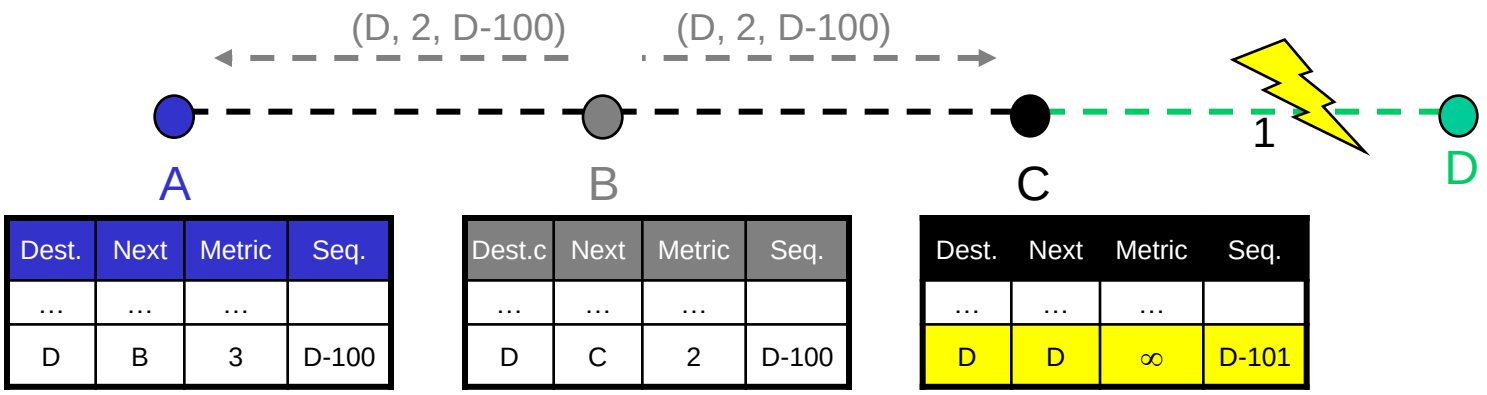
Dest.	Next	Metric	Seq.
A	B	2	A-550
B	B	1	B-102
C	C	0	C-592
D	D	1	D-000

DSDV

No loops, no count to infinity

2. B does its broadcast
-> no affect on C (C knows that B has stale information because C has higher seq. number for destination D)
-> no loop -> no count to infinity

1. Node C detects broken Link:
-> Increase Seq. Nr. by 1
(only case where not the destination sets the sequence number -> odd number)



Dest.	Next	Metric	Seq.
…	…	…	
D	B	3	D-100

Dest.c	Next	Metric	Seq.
…	…	…	
D	C	2	D-100

Dest.	Next	Metric	Seq.
…	…	…	
D	D	∞	D-101

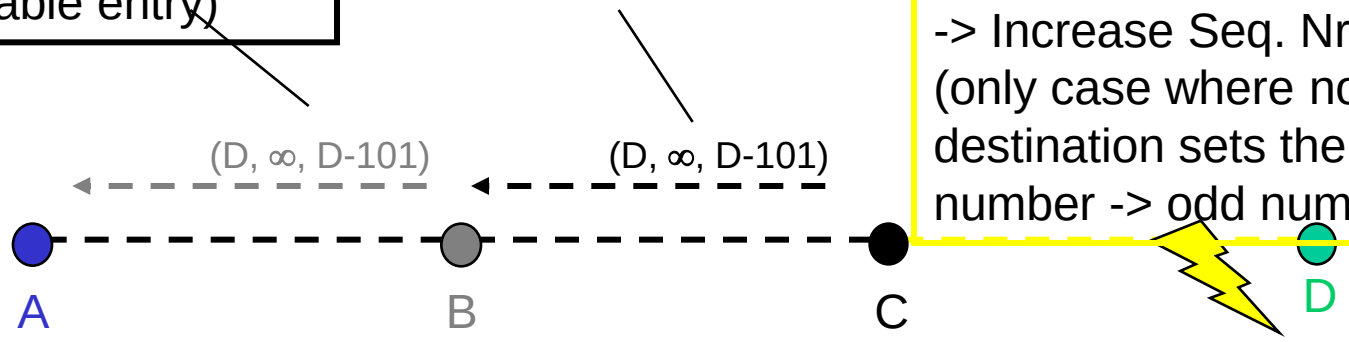
DSDV

Immediate Advertisement

3. Immediate propagation
B to A:
(update information has higher Seq. Nr. -> replace table entry)

2. Immediate propagation
C to B:
(update information has higher Seq. Nr. -> replace table entry)

1. Node C detects broken link:
-> Increase Seq. Nr. by 1
(only case where not the destination sets the sequence number -> odd number)

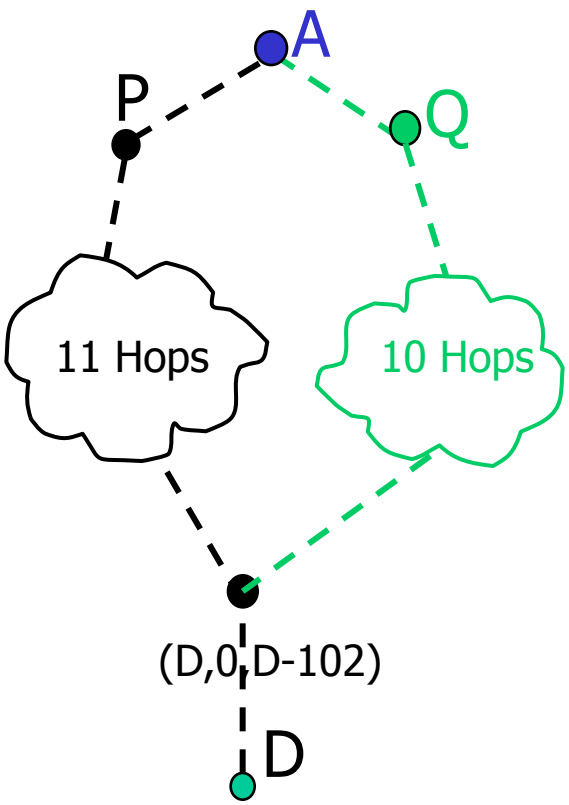


Dest.	Next	Metric	Seq.
…	…	…	…
D	B	3	D-100
D	B	∞	D-101

Dest.c	Next	Metric	Seq.
…	…	…	…
D	C	2	D-100
D	C	∞	D-101

Dest.	Next	Metric	Seq.
…	…	…	
D	D	1	D-100
D	D	∞	D-101

DSDV



Problem of Fluctuations

- Entry for D in A: [D, Q, 14, D-100]
- D makes broadcast with Seq. Nr. D-102
- A receives from P Update (D, 15, D-102)
  -> Entry for D in A: [D, P, 15, D-102]
  A must propagate this route immediately.
- A receives from Q Update (D, 14, D-102)
  -> Entry for D in A: [D, Q, 14, D-102]
  A must propagate this route immediately.

This can happen every time D or any other node does its broadcast and lead to unnecessary route advertisements in the network, so called fluctuations.

DSDV

- Advantages
  - Simple (almost like Distance Vector)
  - Loop free through destination seq. numbers
  - No latency caused by route discovery
- Disadvantages
  - No sleeping nodes
  - Bi-directional links required
  - Overhead: most routing information never used
  - Scalability is a major problem

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

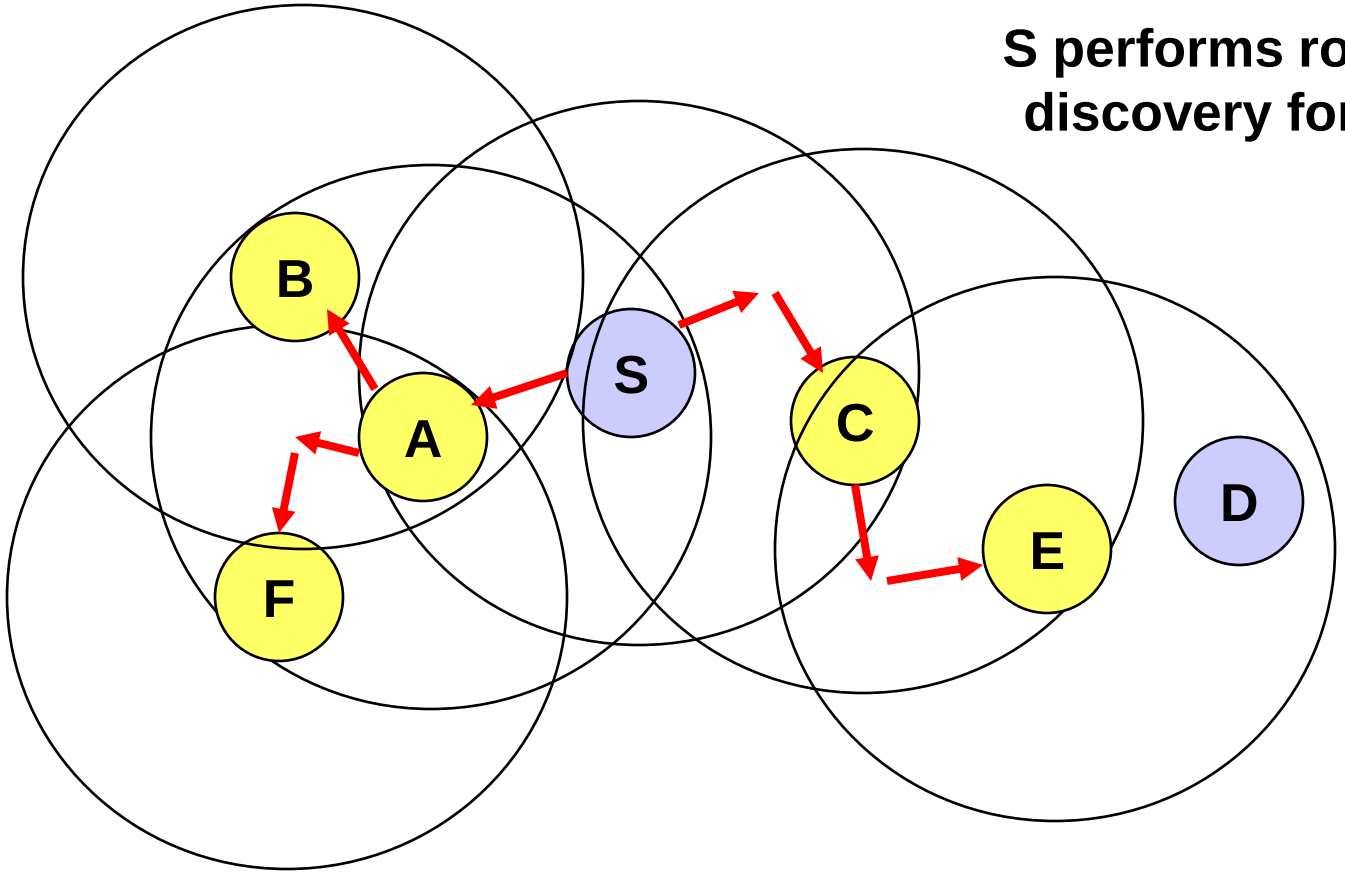
ZRP

- 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

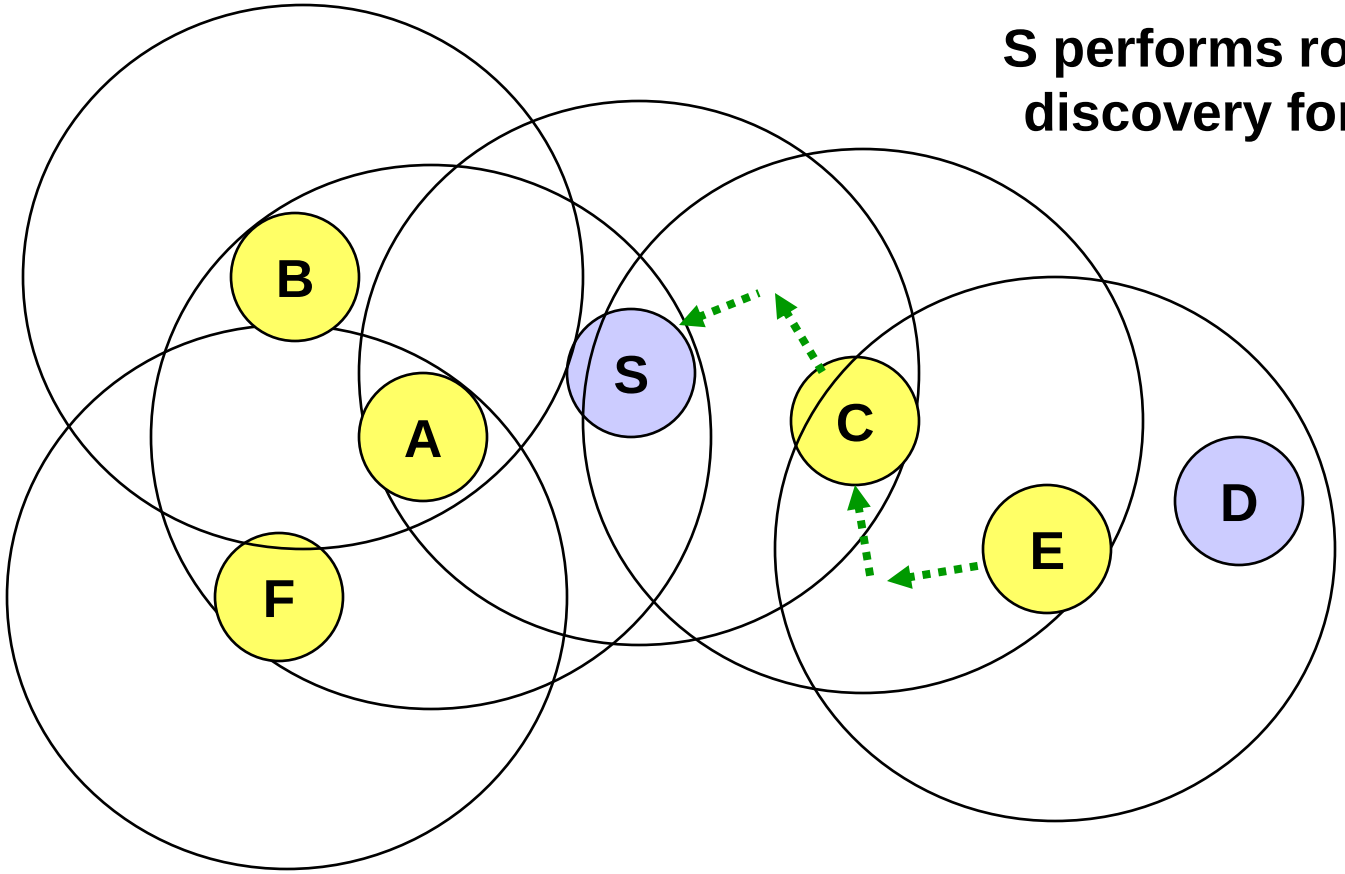
- Intra-zone routing: Pro-actively maintain state information for links within a short distance from any given node
  - Routes to nodes within short distance are thus maintained proactively (using, say, link state or distance vector protocol)
- Inter-zone routing: Use a route discovery protocol for determining 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*

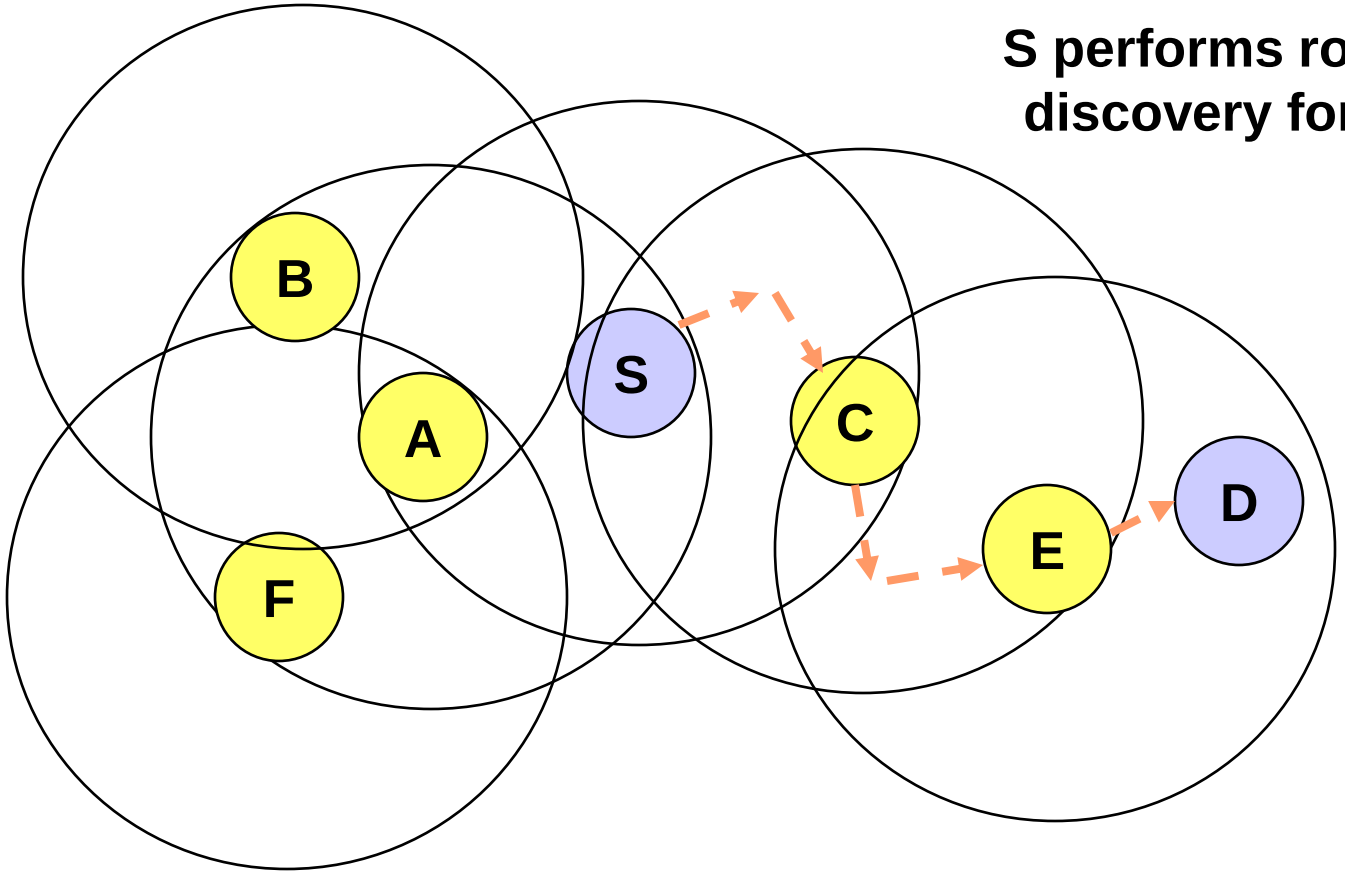


→ **Denotes route request**

ZRP: Example with *d* = *2*



ZRP: Example with *d* = *2*



Denotes route taken by Data

Routing

- Protocols discussed so far find/maintain a route provided it exists
- Some protocols attempt to ensure that a route exists by
  - Power Control
  - Limiting movement of hosts or forcing them to take detours

Power Control

- Protocols discussed so far find a route, on a *given* network topology
- Some researchers propose *controlling* network topology by transmission power control to yield network properties which may be desirable
  - Such approaches can significantly impact performance at several layers of protocol stack
- Method by Watt et al. provides a distributed mechanism for power control which allows for local decisions, but guarantees global connectivity
  - Each node uses a power level that ensures that the node has at least one neighbor in each *cone* with angle $2\pi/3$

Other Routing Protocols

- Plenty of other routing protocols
  - Discussion here is far from exhaustive
- Many of the existing protocols could potentially be adapted for MANET (some have already been adapted as discussed earlier)

Some Variations

Power-Aware Routing

Define optimization criteria as a function of energy consumption. Examples:

- Minimize energy consumed per packet
- Minimize time to network partition due to energy depletion
- Maximize duration before a node fails due to energy depletion

Power-Aware Routing

- Assign a weigh to each link
- Weight of a link may be a function of energy consumed when transmitting a packet on that link, as well as the residual energy level
  - low residual energy level may correspond to a high cost
- Prefer a route with the smallest aggregate weight

Power-Aware Routing

Possible modification to DSR to make it power aware (for simplicity, assume no route caching):

- Route Requests aggregate the weights of all traversed links
- Destination responds with a Route Reply to a Route Request if
  - it is the first RREQ with a given (“current”) sequence number, or
  - its weight is smaller than all other RREQs received with the current sequence number

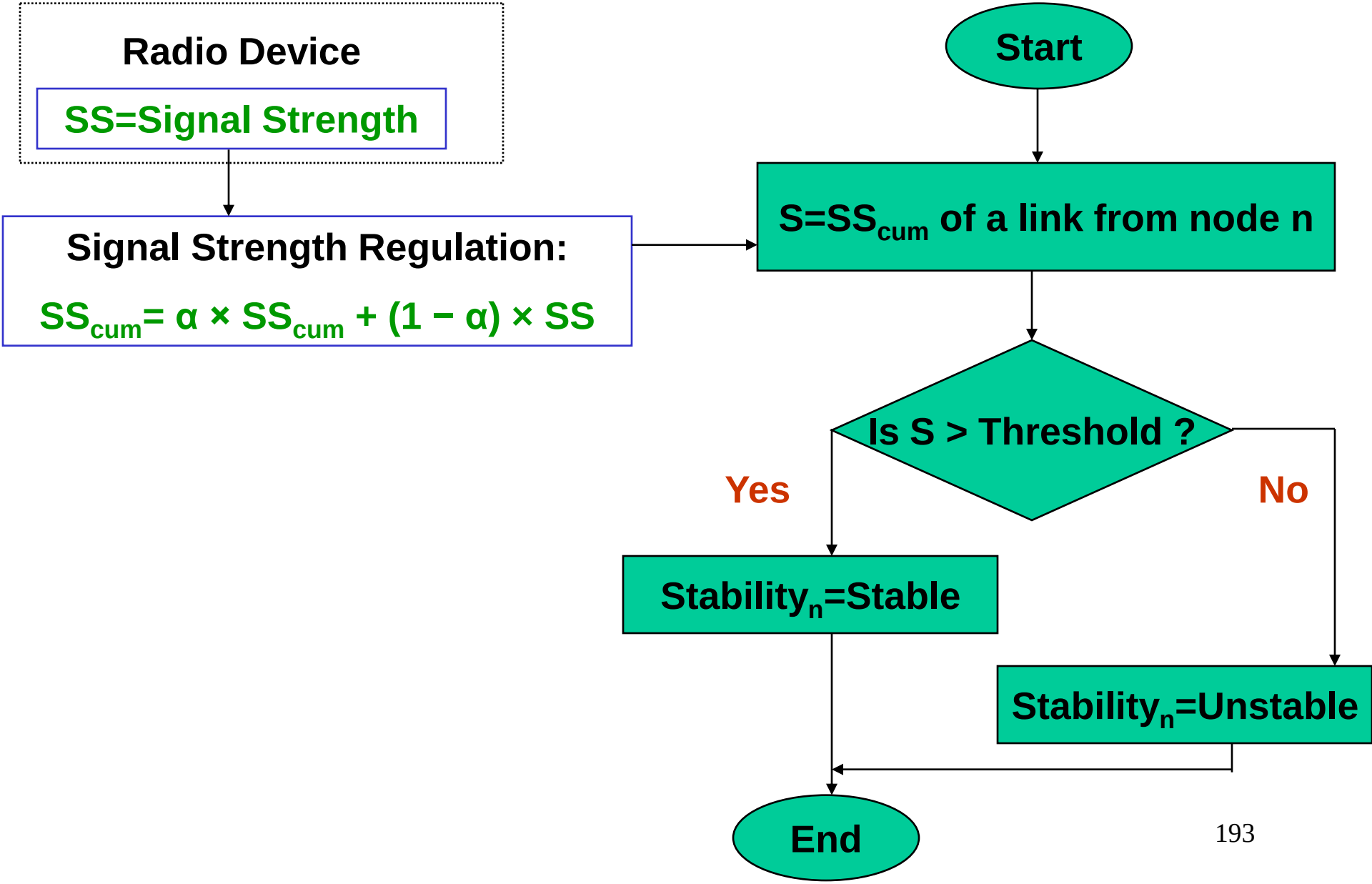
Signal Stability Based Adaptive Routing (SSA)

- Similar to DSR
- A node X re-broadcasts a Route Request (RREQ) received from Y only if the (X,Y) link is deemed to have a ***strong signal stability (link stability is estimated from signal strength information)***
- **Signal stability** is evaluated as a moving average of the signal strength of packets received on the link in recent past
- An alternative approach would be to assign a cost as a function of signal stability

Signal Stability Based Adaptive Routing (SSA)

- SSA tries to find a route on these strong (stable) links. If there exists a route thro' all stable links, then it will be more stable route than an ordinary route
- If it fails finding a route (thro' all stable links) in first trial, it searches a route on all available links (both stable & unstable links)
- **Signal Strength Estimation:**
  - All nodes monitor signals from its neighbor nodes
  - If the strength of the signal received from neighbor nodes is beyond a Threshold, the link from the neighbor node is considered as Strong Stability link
  - Otherwise, the link is treated as Unstable link

Signal Stability Based Adaptive Routing (SSA)



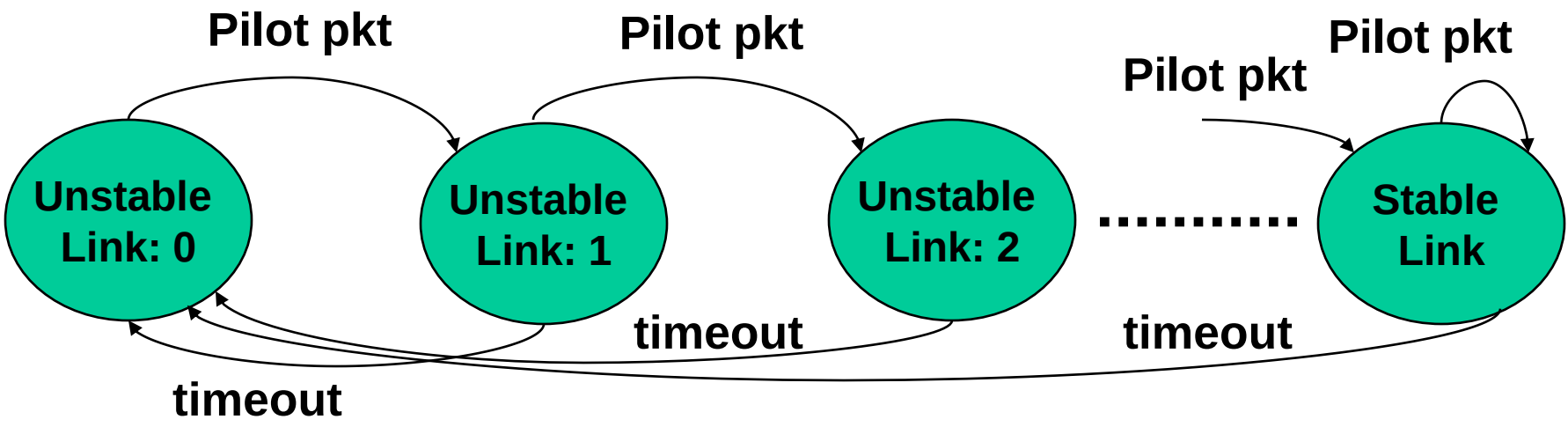
Associativity-Based Routing (ABR)

- Only links that have been stable for some minimum duration are utilized
  - **motivation**: If a link has been stable beyond some minimum threshold, it is likely to be stable for a longer interval. If it has not been stable longer than the threshold, then it may soon break (could be a *transient* link)
- ***Association stability*** determined for each link
  - measures duration for which the link has been stable
- Prefer paths with high aggregate association stability

Associativity-Based Routing (ABR)

- ABR uses **Pilot Signal** to determine link stability
- Every node sends pilot signal periodically
- When a node receives this pilot signal from its neighbor nodes, it records no. of pilot signal received in a counter
- If it receives a no. of pilot signals continuously from a neighbor which is beyond a Threshold, it considers the link as stable
- It searches all the possible routes to find a route that contains more stable (strong) links.

Associativity-Based Routing (ABR)



Route Lifetime: (Route Stability) is determined by the no. of links that compose the route and link stability of each link.

Probability of link (i) failure as $P_{link\_fail(i)}$.

Therefore, prob of Route failure $P_{route\_fail} = (1- \pi (1 – P_{link\_fail(i)}))$

Preemptive Routing

- Add some proactivity to reactive routing protocols such as DSR and AODV
- Route discovery initiated when it appears that an active route will break in the near future
- Initiating route discover *before* existing route breaks reduces discovery latency

QoS Routing

Quality-of-Service

- Several proposals for reserving bandwidth for a flow in MANET
- Also Multipath routing

Multicasting
in
Mobile Ad Hoc Networks

Multicasting

- A multicast group is defined with a unique *group identifier*
- Nodes may join or leave the multicast group anytime
- In traditional networks, the physical network topology does not change often
- In MANET, the physical topology can change often

Multicasting in MANET

- Need to take topology change into account when designing a multicast protocol

- Several new protocols have been proposed for multicasting in MANET

Geocasting
in
Mobile Ad Hoc Networks

Multicasting and Geocasting

- Multicast members may join or leave a multicast group whenever they desire
- Geocast group is defined as the set of nodes that reside in a specified geographical region
- Membership of a node to a geocast group is a function of the node's physical location
  - Unlike multicasting
- Geocasts are useful to deliver location-dependent information

Capacity of Ad Hoc Networks

Capacity of Fixed Ad Hoc Networks

[Gupta00it]

- n nodes in area A transmitting at W bits/sec using a fixed range (distance between a random pair of nodes is O(sqrt(n))
- Bit-distance product that can be transported by the network per second is

$$\Theta (W \text{ sqrt } (A\ n))$$

- Throughput per node

$$\Theta (W \ / \text{ sqrt } (n))$$

Capacity of Mobile Ad Hoc Networks

- Assume random motion
- Any two nodes become neighbors once in a while
- Each node assumed sender for one *session,* and destination for another *session*
- Relay packets through at most one other node
  - Packet go from S to D directly, when S and D are neighbors, or from S to a relay and the the relay to D, when each pair becomes neighbor respectively
- Throughput of each session is O(1)
  - Independent of *n*

Continues from last slide …

- Delay in packet delivery can be large if O(1) throughput is to be achieved
  - Delay incurred waiting for the destination to arrive close to a relay or the sender
- Trade-off between delay and throughput

Measured Capacity

- Confirms intuition
- In fixed networks, capacity is higher if average distance between source-destination pairs is small

Measured Scaling Law

- Measured in static networks
- Throughput declines worse with *n* than theoretically predicted
- Due to limitations of existing MAC protocols
  - Unable to exploit “parallelism” in channel access

Capacity

- How to design MAC and routing protocols to approach theoretical capacity ?
- Open problem