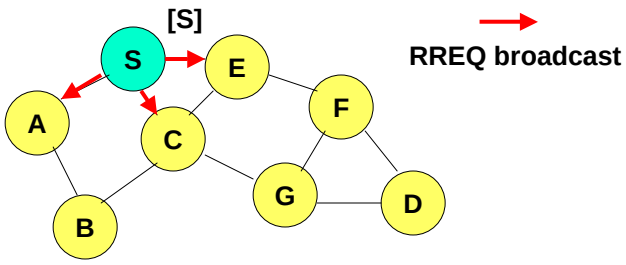
Dynamic Source Routing (DSR)

[Johnson-Maltz-96, Broch et. al. 98-00]

- 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 the network with route request (RREQ) packets (also called query packets).
- Each node appends its own address in the packet header when forwarding RREQ.

Route Discovery in DSR

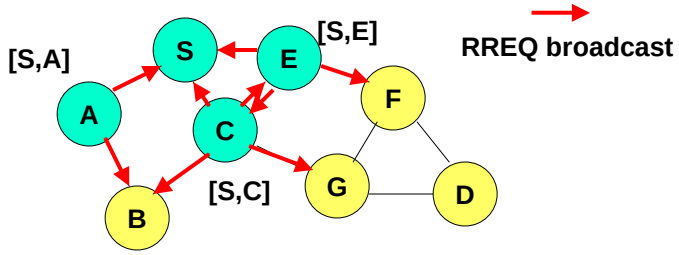


represents a node that has received RREQ for D from S.

[X,..,..] Represents list of addresses appended to RREQ.

A node receiving a RREQ rebroadcasts it exactly once.

Route Discovery in DSR

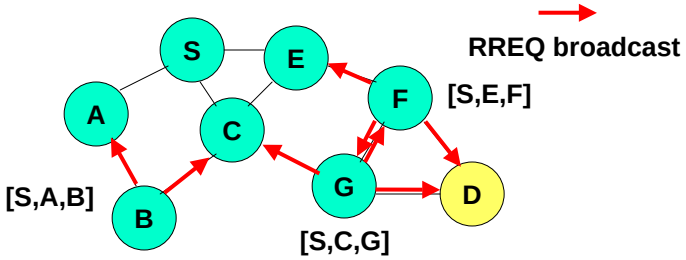


represents a node that has received RREQ for D from S.

[X,..,..] Represents list of addresses appended to RREQ.

A node receiving a RREQ rebroadcasts it exactly once.

Route Discovery in DSR



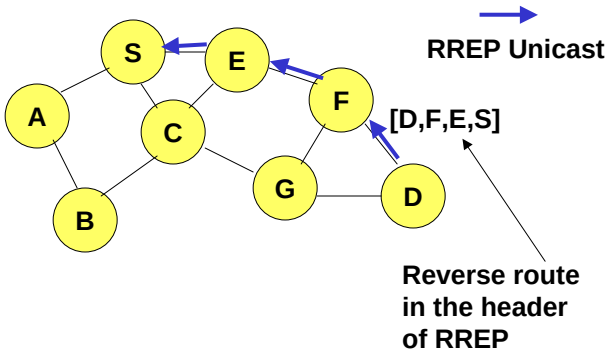
Destination D receives RREQ via G and F.

It does not broadcast it further.

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 reverse route from S to D on which RREQ was received by node D.

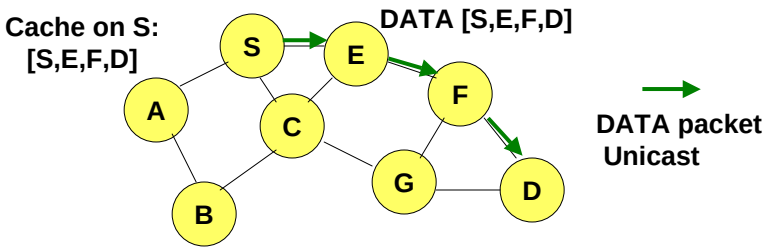
Route Reply in DSR



Route Caching in DSR

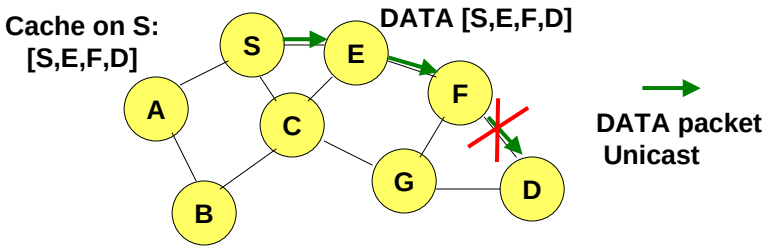
- Node S on receiving RREP, "caches" the route included in the RREP.
- When node S sends a data packet to D, the entire route is included in the packet header
  - Hence the name source routing.
- Intermediate nodes use the source route included in a packet to determine to whom a packet should be forwarded.

Data Delivery in DSR



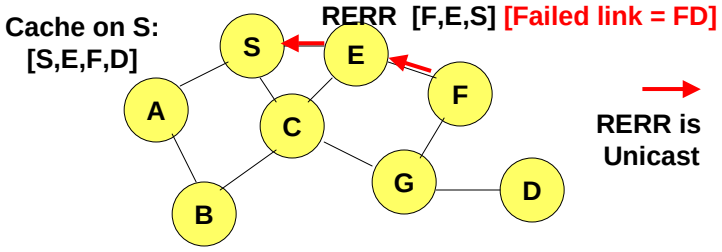
Source route size grows with route length.

Route Error



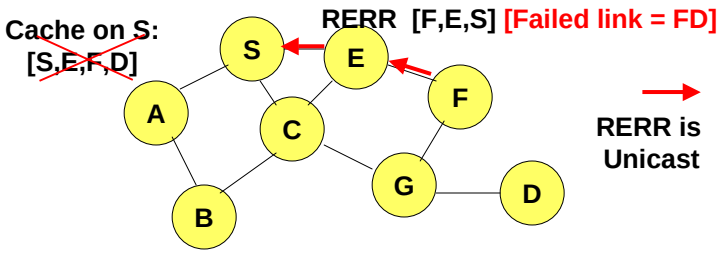
- If the next hop link is broken when a data packet is being forwarded, a Route Error (RERR) is generate and propagated backwards.

Route Error



- If the next hop link is broken when a data packet is being forwarded, a Route Error (RERR) is generate and propagated backwards.
- RERR contains the failed link info.

Route Error

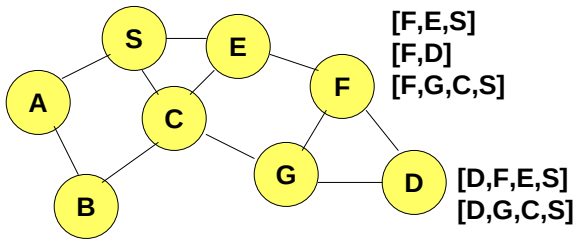


- **When S receives RERR, it erases any cached route with the failed link.**

Aggressive Route Caching

- **Each node caches a new route it learns by *any means***
- When node S finds route [S,E,F,D] to node D, node S also learns route [S,E,F] to node F and so on.
- When node G receives RREQ [S,C] destined for node D, node G learns route [G,C,S] to node S and so on.
- When node F forwards RREP [D,F,E,S], node F learns route [F,D] to node D and so on.
- Basically, when forwarding any packet, the node learns a route to *all* nodes in the source route contained in the packet.

Contents of Caches on Selected Nodes After one RREQ-RREP Cycle



- [P,Q,R] represents cached route at a node P.
- More than one routes may be cached for the same destination.
- Compact data structures may be used to implement route caches (e.g., tree).

Use of Route Caching

- **Salvaging:** 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
- **Reply from Cache:** Node X on receiving a RREQ for some node D can send a Route Reply if node X knows a route to node D.
- Aggressive use of route cache
  - can speed up route discovery.
  - can reduce propagation of route requests.

Route Caching: Beware!

- **Stale caches** can adversely affect performance.
- With passage of time and host mobility, cached routes may become invalid.
- All cached routes containing a failed link are not erased by route error (RERR).
  - Only that route is erased that is attempted to be used
- A sender host may try several stale routes (obtained from local cache, or replied from cache by other nodes), before finding a valid route.

Dynamic Source Routing: Advantages

- **Source routing:** no special mechanism needed to eliminate loops.
- **On demand routing:** 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.
  - Useful when route breaks.

Dynamic Source Routing: Disadvantages

- **Not scalable:** Packet header size grows linearly with route length due to source routing.
- **Network-wide flood:** Flood of route requests may potentially reach all nodes in the network. Too much overhead.
- **Collision:** Care must be taken to avoid collisions between route requests propagated by neighboring nodes
  - insertion of random delays before forwarding RREQ
- **Reply storm problem**: Increased contention if too many route replies come back due to nodes replying using their local cache
  - 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

- **Stale cache problem:** 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.
- Current research: how to invalidate caches effectively.
  - Example: Timer-based. Or propagate the route error widely.

Ad Hoc On-Demand Distance Vector Routing (AODV)

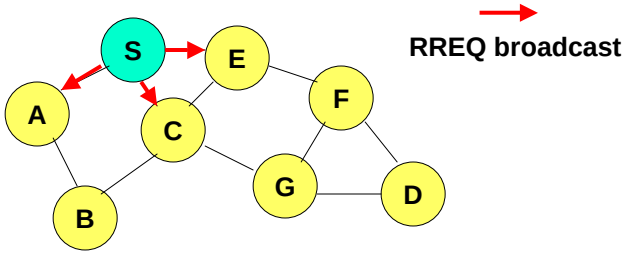
[Perkins-Royer-Das 99,00]

- AODV retains the desirable feature of DSR that routes are maintained only between nodes which need to communicate.
- AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes.
- No caches are used.
  - Only one route per destination in the routing table.
  - Only maintain the freshest route, if multiple possibilities.

AODV

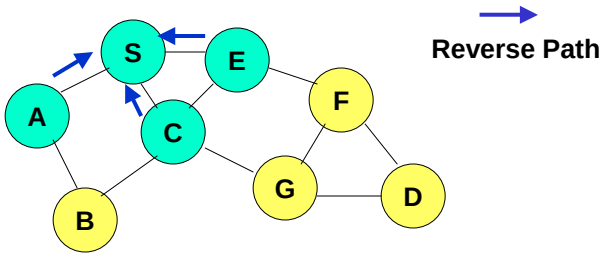
- Route Requests (RREQ) are forwarded in a manner similar to DSR.
- When a node re-broadcasts a RREQ, it sets up a reverse path pointing towards the source.
  - This is so that the RREP can get back to the source.
- When the intended destination receives a RREQ, it replies by sending a RREP.
- RREP travels along the reverse path set up when RREQ is forwarded.

AODV Route Discovery



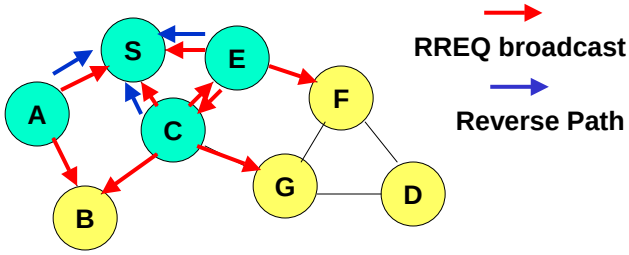
- Source floods route request (RREQ) in the network.
- Reverse paths are formed when a node hears a route request.
- Each node forwards the request only once (pure flooding).

AODV Route Discovery



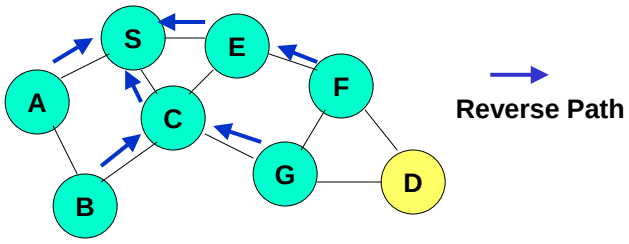
- Source floods route request in the network.
- Reverse paths are formed when a node hears a route request.
- Each node forwards the request only once (pure flooding).

AODV Route Discovery



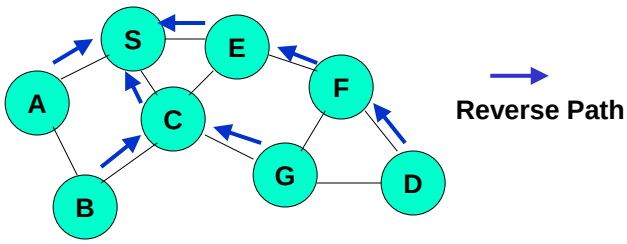
- Uses hop-by-hop routing.
- Reverse paths are formed when a node hears a route request.
- Each node forwards the request only once (pure flooding).

AODV Route Discovery



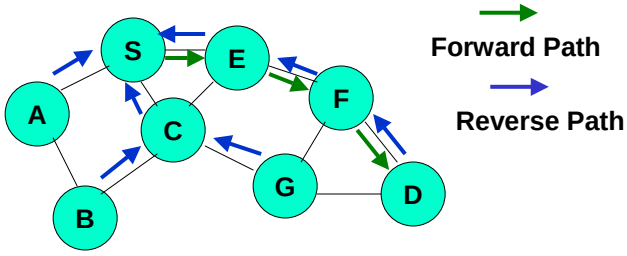
- Uses hop-by-hop routing.
- Reverse paths are formed when a node hears a route request.
- Each node forwards the request only once (pure flooding).

AODV Route Discovery



- Route reply (RREP) is forwarded via the reverse path.

AODV Route Discovery

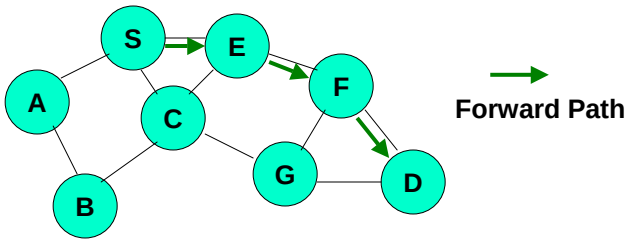


- Route reply is forwarded via the reverse path … thus forming the forward path.
- The forward path is used to route data packets.

Route Expiry on Timeout

- A routing table entry maintaining a reverse path is invalidated after a timeout interval
  - Timeout should be long enough to allow RREP to come back
- A routing table entry maintaining a forward path is also invalidated if unused for certain interval.
  - This means unused routes are purged.
  - Note that the route may still be valid.

Route Expiry



- Unused reverse paths expire based on a timer.

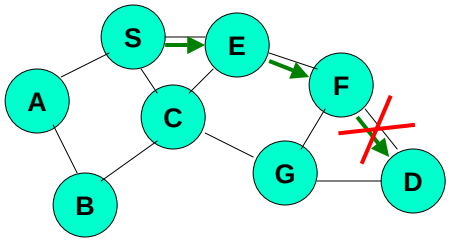
Link Failure Detection

- *Hello* messages: Neighboring nodes periodically exchange hello or keep-alive messages.
- Absence of any hello message for some time is used as an indication of link failure.
- Alternatively, failure to receive several link-layer ACK for a transmitted packet may be used as an indication of link failure.
- Note DSR needs to use one of the above as well.

Route Error

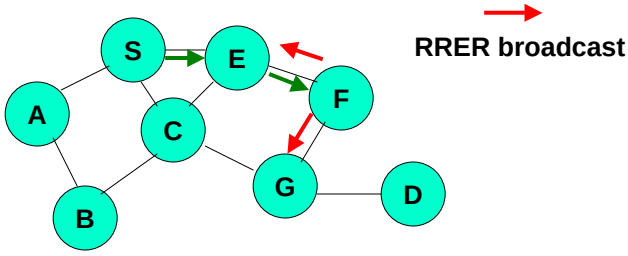
- When a node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a RERR message back to S.
- How to forward the RERR to S? X may not have a route to S.
- RERR is broadcast. Any node having an active route to D, rebroadcasts RERR after invalidating that route.

Route Error Propagation



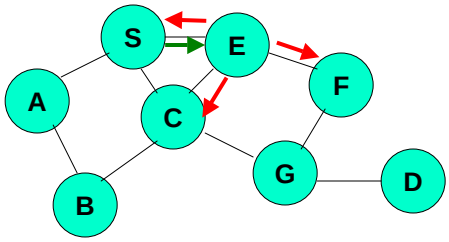
- Link breakage on an active route triggers route error (RERR).

Route Error Propagation



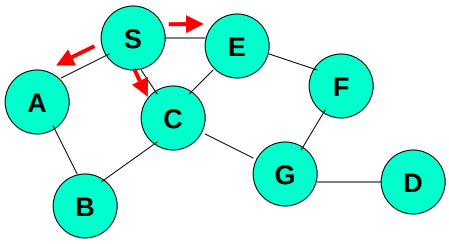
- F invalidates the route to D, and broadcasts RERR.
- E notes that it has a route to D via F, and it continues the process.

Route Error Propagation



- F invalidates the route to D, and broadcasts RERR.
- E notes that it has a route to D via F, and it continues the process.

Route Error Propagation

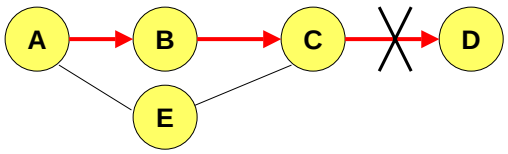


- The entire upstream route is erased.
- S starts fresh route discovery if needed.

Possibility of Routing Loops!

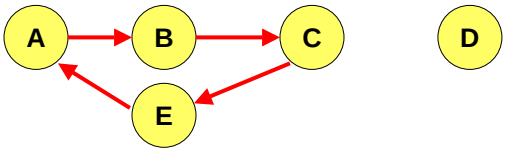
- Useful optimization: An intermediate node with a route to D can reply to route request.
  - Faster operation.
  - Quenches route request flood.
- Wireless reality: Routing messages can get lost.
- It can be shown that above can cause long-term routing loops.

Possibility of Routing Loops!



- Assume that A does not know about failure of link C-D because route error sent by C is lost.
- Now C performs a route discovery for D. Node A receives the route request (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)**

Possibility of Routing Loops!

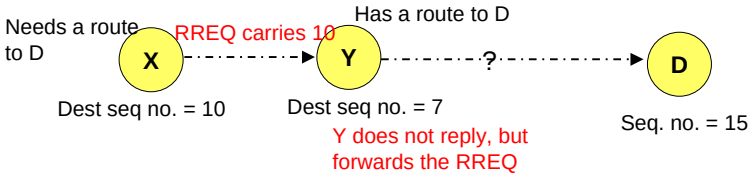


- Assume that A does not know about failure of link C-D because route error sent by C is lost.
- Now C performs a route discovery for D. Node A receives the route request (say, via path C-E-A)
- Node A will reply since A knows a route to D via node B
- Results in a loop (for instance, C-E-A-B-C)

Use of Sequence Numbers in AODV

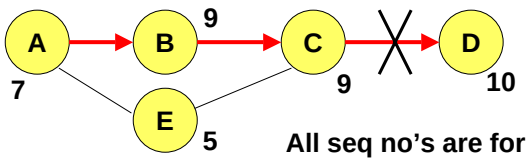
- Each node X maintains a sequence number and increments it at suitable intervals.
- Seq. no. acts like a logical clock.
- Each node Y with a route to X in the routing table, also maintains a destination sequence number for X, which is Y's *latest knowledge* of X's sequence number.
- Destination sequence no. can be used to order routing updates.

Use of Sequence Numbers in AODV



- Loop freedom: The protocol maintains the invariant that the destination sequence number for any destination D never decreases along any valid route.
  - No routing info is accepted from by a node X from any node Y, where Y's destination seq. no. for D is less than X's destination seq. no. for D.
- Freshest route: Given a choice of multiple routes, the protocol always chooses the one with the highest sequence number.

How Using Sequence Numbers can Avoid Loop?



- Link failure increments the destination seq. no. at C (now is 10).
- If C needs a route to D, RREQ carries the current dest. seq. no. (10).
- A does *not* reply as its own dest. seq. no. is less than 10.

Summary: AODV

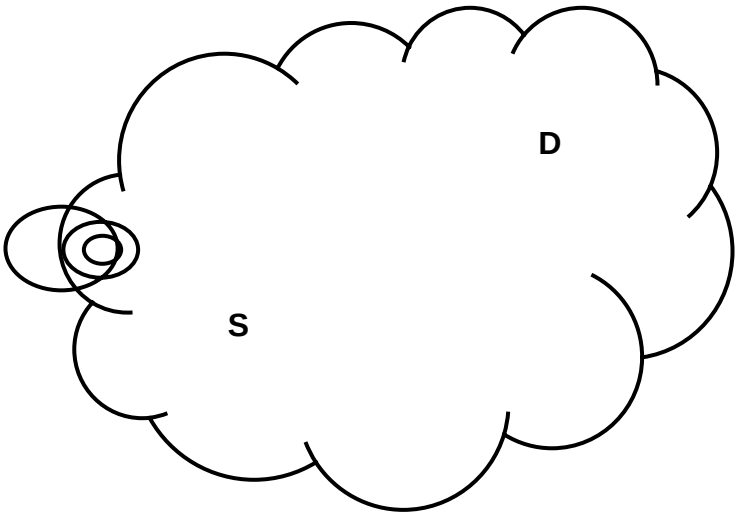
- No source routing. Based on routing tables.
- Use of sequence numbers to prevent loops.
- At most one route per destination maintained at each node
  - Only the freshest one is maintained (via destination seq. no.)
  - Stale route problem is less severe.
  - After link break, all routes using the failed link are erased.
- Unused routes expire even if valid.

Flood Control Optimizations for DSR and AODV

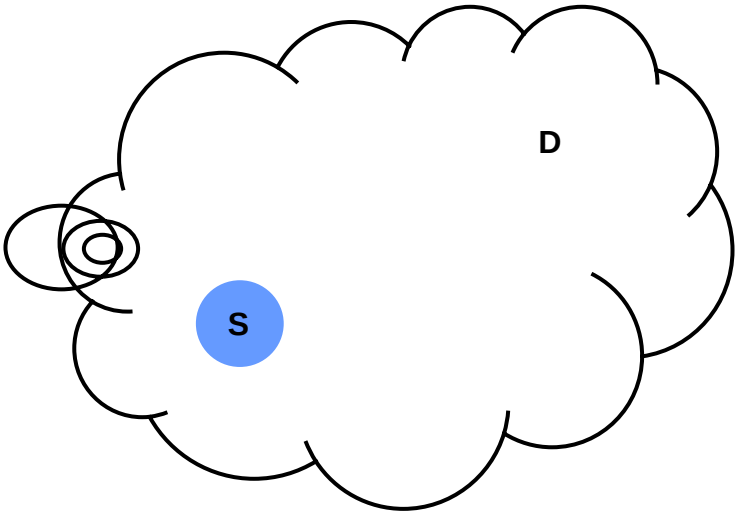
- Note that the whole network is flooded by Route Request (RREQ) packets when route is needed.
  - Large overhead.
  - Intermediate nodes with routes may quench the flood, but not guaranteed and may not happen in all directions.
- Need techniques to control flood.
- Two different sets of techniques
  - Limit the extent of flood to a region where the destination is likely to be found.
  - Eliminate redundant broadcasts.

Flood Control Optimizations

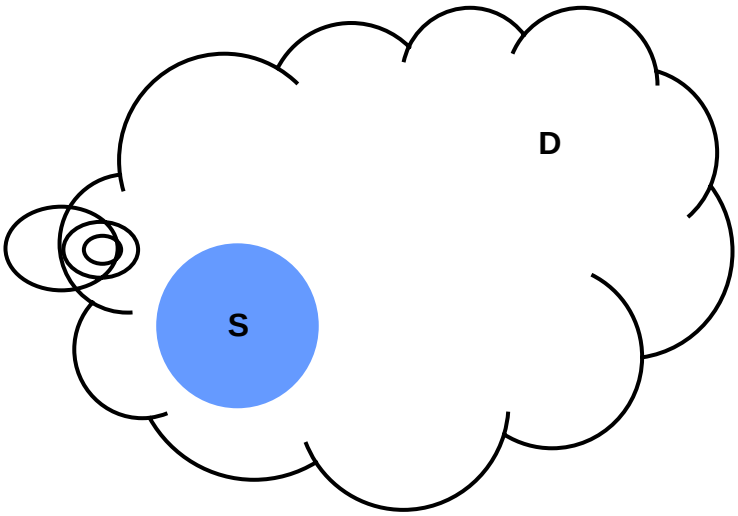
Route Discovery



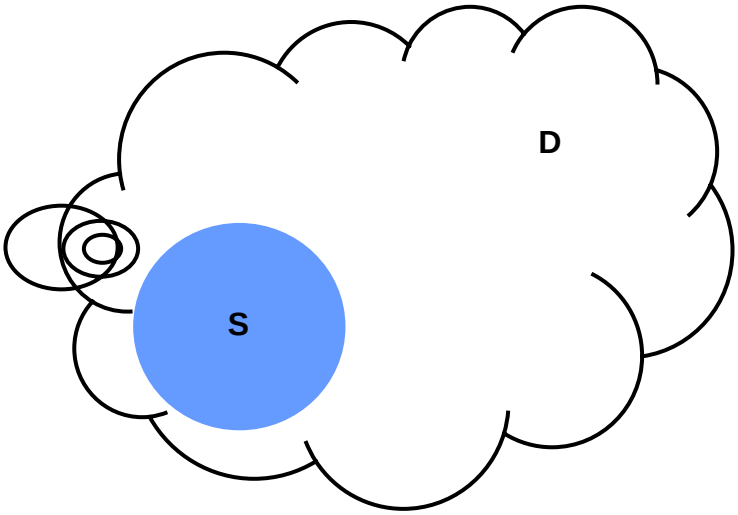
Query Flooding



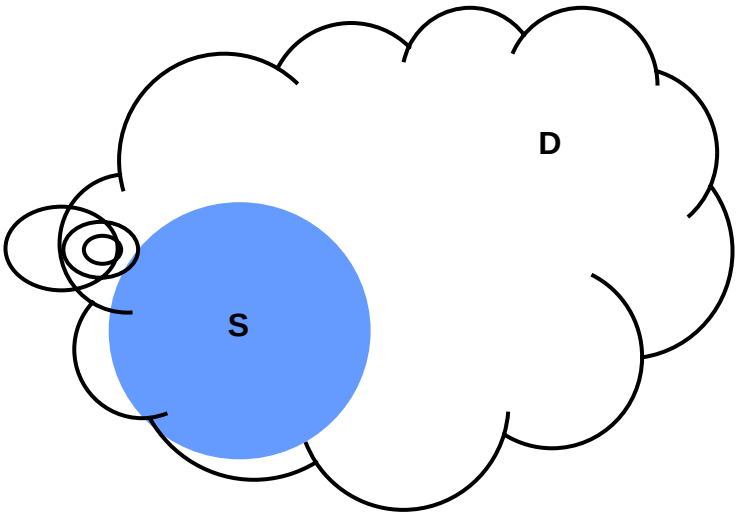
Query Flooding



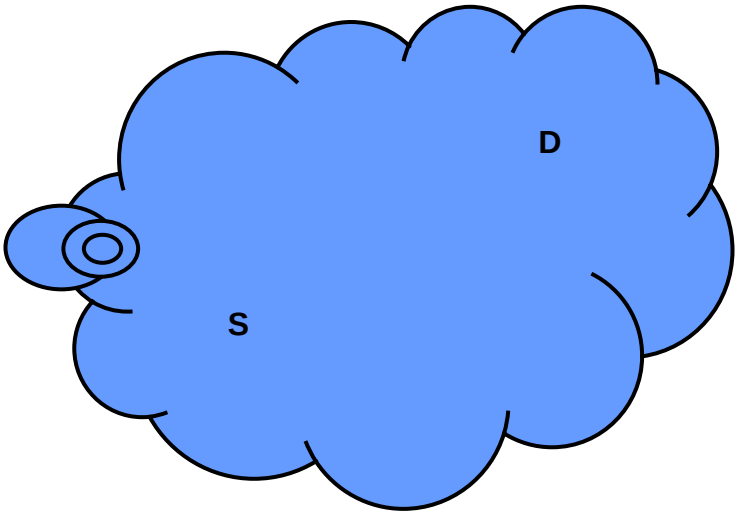
Query Flooding



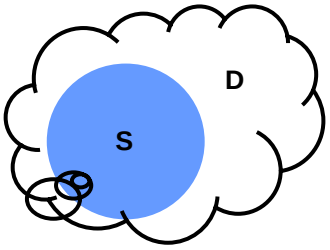
Query Flooding



Whole Network is Flooded

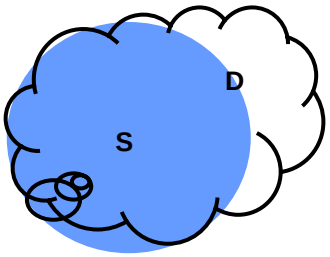


Use of Max Hop Count (TTL) Field



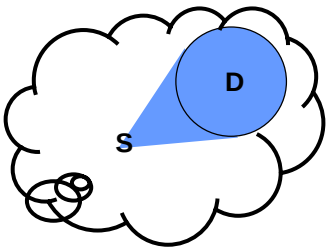
- Limits query propagation within a certain number of hops from the source.
- Expanding ring search.
- Can still flood a substantial part of the network if destination is far away.

Use of Max Hop Count (TTL) Field



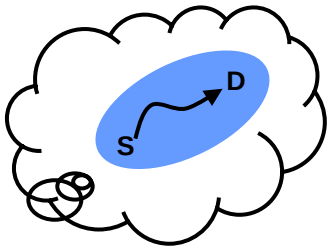
- Limits query propagation within a certain number of hops from the source.
- Expanding ring search.
- Can still flood a substantial part of the network if destination is far away.

Using Location Information



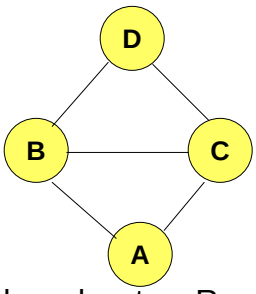
- Uses geographic location info to direct query propagation towards destination (LAR).
  [Ko-Vaidya-98]
- Or flood data packets directly in the cone rooted at source (DREAM).
  [Basagni-Chlamtac-Syrotiuk-98]
- Needs additional hardware (GPS).

Query Localization
[Das-Castenada-99]



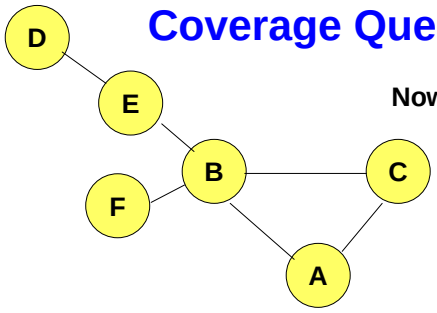
- Exploit "spatial locality."
- Propagate query only in a topological neighborhood of the last valid route between S and D.
- Several heuristics to define neighborhood possible.

Broadcast Storm Problem
[Ni et. al. – 98]



- When node A broadcasts a Route Request, nodes B and C both receive it
- B and C both rebroadcasts (forwards) the request.
- D receives two copies – from B and C. One is sufficient.
- Since B and C can hear each other, can they coordinate so that only one will rebroadcast?

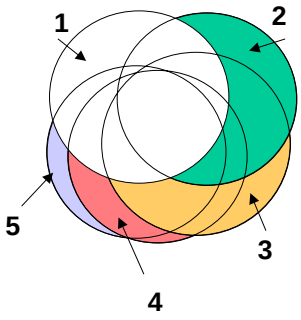
Coverage Question



Now a different topology!

- Suppose, B decides not to rebroadcast after hearing C's rebroadcast.
- RREQ will never reach D.
- B need to be somewhat confident that C's rebroadcast is covering all of B's neighbors before deciding not to rebroadcast.
- How?

Idea: Incremental Coverage is Small



- Suppose, node 1 broadcasts a RREQ.
- Additional nodes randomly located in the coverage area of node 1 will only incrementally add to the total coverage.
- Benefit very small beyond 5 nodes.
- Idea: need not rebroadcast the RREQ if already heard a few broadcasts for the same RREQ in the neighborhood.

Solution for Broadcast Storm

- Counter-Based Scheme: If node X hears more than *k* neighbors broadcasting a given route request, before it can itself rebroadcast it, then node X will not forward the request
- Intuition: *k* neighbors together have probably already forwarded the request to all of X's neighbors. Thus re-broadcasting by X will not have any added benefit.
- Heuristic parameters: Value of k, how long to wait before rebroadcasting?

Proactive Protocols

Link State Routing

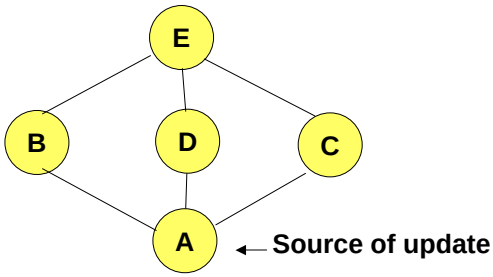
- Each node floods the network with the status of its links
  - Flood can be periodic.
  - Or, when a neighborhood change is detected.
- Each node keeps track of link state information received from other nodes
  - Thus builds its own view of the network connectivity.
- Each node uses its view of network connectivity to construct a routing table for each destination.
  - For example, each node can run a shortest-path algorithm (e.g., Dijkstra's) on its own view of the connectivity graph.
  - Different nodes can use different objective for routing.

Overhead Reduction in Link-State Protocols

- Pure flooding is high overhead. Do not use pure flooding. How?
- **Method 1:** Flood all nodes. But require fewer nodes to do the forwarding. For example, maintain a tree structure rooted at the source of the update.
- **Method 2:** Flood only a **connected dominating set** of nodes and maintain routes only among this dominating set.
  - Dominating set: A subset of nodes such that each node in the network is in this set or a neighbor of some node in the set.

TBRPF: Topology Broadcast with Reverse Path Forwarding

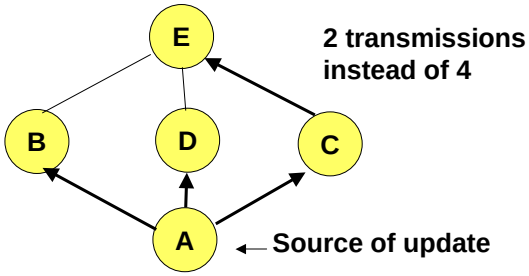
[Bellur-Ogier-Templin-99,00]



- Send link-state updates only via the minimum-hop spanning tree rooted at the source of the update.
- Little cost for maintaining the spanning tree. In a link-state protocol each node has the network connectivity information.

TBRPF: Topology Broadcast with Reverse Path Forwarding

[Bellur-Ogier-Templin-99,00]



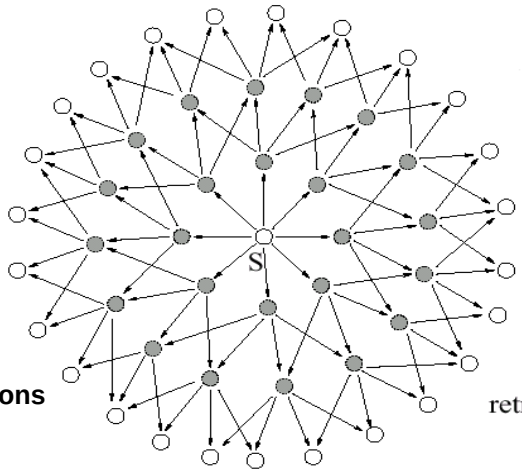
- Send link-state updates only via the minimum-hop spanning tree rooted at the source of the update.
- Little cost for maintaining the spanning tree. In a link-state protocol each node has the network connectivity information.

OLSR: Optimized Link-State Routing

[Jacket, Qayyaum et. al.-99-01]

- Only **multipoint relays** (MPR) participate in routing. Similar to connected dominating set.
- 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 know their 2-hop neighbors, in order to choose the multipoint relays.
  - Select as few multipoint relays as possible.
- Only multipoint relays are used for routing.

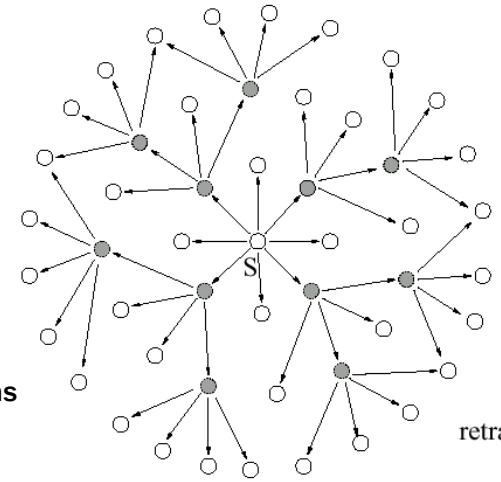
Multipoint Relays



24 retransmissions needed to flood the network

retransmitting node

Multipoint Relays (contd.)



11 retransmissions needed to flood the network

Optimized Link State Routing (OLSR)

- OLSR floods link-state information only through multipoint relays.
- Routes used by OLSR only include multipoint relays as intermediate nodes .
  - Routes are still optimal!
- Each node maintains information about its MPR (multipoint relay) selector set, i.e., set of neighbors that have selected itself as MPR.
- Each node with non-empty MPR selector set periodically floods the network with topology control (TC) messages containing own MPR selector set.
  - This information is used to construct the topology database used for routing calculations.

Distance Vector Routing

- Each node maintains <next hop, #hops> for each destination. This is called distance vector. Same as routing table.
- Each node exchanges its distance vector with its neighbors periodically.
- Upon receiving the distance vector from a neighbor, each node updates its own distance vector.
- Example: Distributed Bellman Ford.

Characteristics of Distance Vector Routing

- Typically cheaper update cost than link-state.
  - In generic link-state routing, each update needs to be sent to ALL nodes.
- Counting to Infinity Problem
  - Takes too long to determine that a destination is unreachable or to an alternative, but much longer route.
- Loops possible because of lack of synchronism in update propagation.
- Both problems can be handled by exchanging additional information or by enforcing synchronization.

Destination-Sequenced Distance-Vector (DSDV)

[Perkins-Bhagwat-94]

- Uses sequence numbers to avoid counting to infinity or looping problems.
- Each node increments and appends its own sequence number when broadcasting its distance vector.
- Each distance vector also includes the destination sequence no.
  - <next hop, #hops, dest. seq. no.> for each destination.
- No update if the dest. seq. no. in the distance vector in the incoming packet is less than the dest. seq. no. in the node's own distance vector.

Hybrid Protocols

Zone Routing Protocol (ZRP)
[Haas-Pearlman-98]

Zone routing protocol combines

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

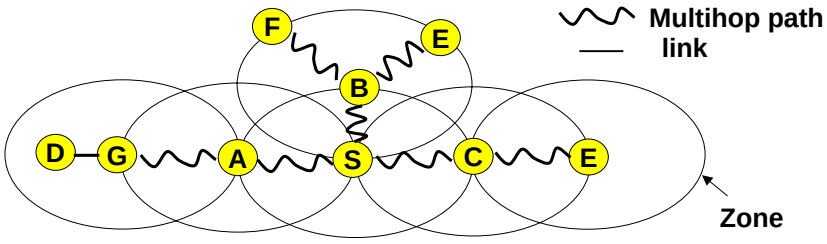
Zones in ZRP

- All nodes within *d* hops 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.
- Bordercasting: The operation of sending a route request query to some or all of a node's peripheral nodes.
  - Full bordercasting
  - Selective bordercasting

Intra and Inter-zone Routing

- Intra-zone routing: Proactively maintain routes to all nodes within the source nodes own zone.
  - Can use link-state or distance-vector protocols.
- Inter-zone routing: Use an on-demand protocol (similar to DSR or AODV) to determine routes to outside zone.
  - Route request query is propagated efficiently from zone to zone via bordercasting.

Route Discovery in ZRP



- S bordercasts route request to A,B, and C.
- They in turn bordercast their zone periphery.
- G recognizes that D is in its zone, and replies to route request.
- G knows route to D via intra-zone routing.

Issues in ZRP

- ZRP tries to balance best of both worlds.
  - Lower route discovery latency than purely on-demand as route request propagates in “quantum” of zone radius.
  - Lower routing packet overhead than proactive, as only routes within zone are proactively maintained.
- Zone radius must be chosen carefully!
- ZRP includes techniques to improve inter-zone route discovery process
  - Preventing loopback: prevent route request coming back to the zone that it already queried.
  - Selective bordercasting: only a subset of peripheral nodes are bordercasted to such that all next stage zones are covered.

Choice of Link Cost Metric

- Usually in MANET protocols, each link is considered unit cost.
  - The shortest path is in number of hops.
  - Reasonable choice as delay in each hop is usually high and each transmission depletes power budget.
- But other interesting cost metrics are also possible
  - Based on some link stability metric
    - Don’t use a node as a next hop that changes neighborhood frequently. It may be moving fast.
  - Based on signal strength.
    - Rely more on links with strong signal strength.
- Also, sometimes load balancing helps.
  - Avoid nodes on congested routes.

Performance Matters

1. 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
  - Typically equal to round-trip time between source and destination.
  - Need to buffer packets while route discovery in progress. Buffer overflow -> packet loss.

2. Routing overhead

- Any packets transmitted for route discovery/maintenance purposes are counted as overheads.
  - Example, RREQ, RREP, RERR packets in on-demand; route update packets in proactive.
- Reactive protocols *may* have lower overhead since routes are determined only if needed
- Proactive protocols *may* result in higher overhead due to continuous route updating

Performance Matters

3. Quality of routes

- Proactive protocols will typically guarantee shortest path.
- On-demand protocols may not always guarantee shortest path.
  - May be shortest path initially. But a new shorter path won't be used until a new route discovery, which won't happen unless the original route breaks.
  - Sub-optimal paths increases data packet latency.

- Lot of performance study in literature via simulations **[Broch et al. 98, Das et. al. 98,00,01]**

Performance Matters

- Actual trade-off depends a lot on traffic and mobility patterns.
- Traffic:
  - Higher traffic diversity (more source-destination pairs) increases overhead in on-demand protocols, as more routes need to be discovered.
- Mobility:
  - Higher mobility will always increase overhead in all protocols.
  - But may not matter much with low traffic diversity for on-demand protocols.
- For very high mobility, no protocol will do well.
  - For example, if the route changes by the time it takes to discover a route …