# LECTURE 9

Ad hoc Networks and Routing

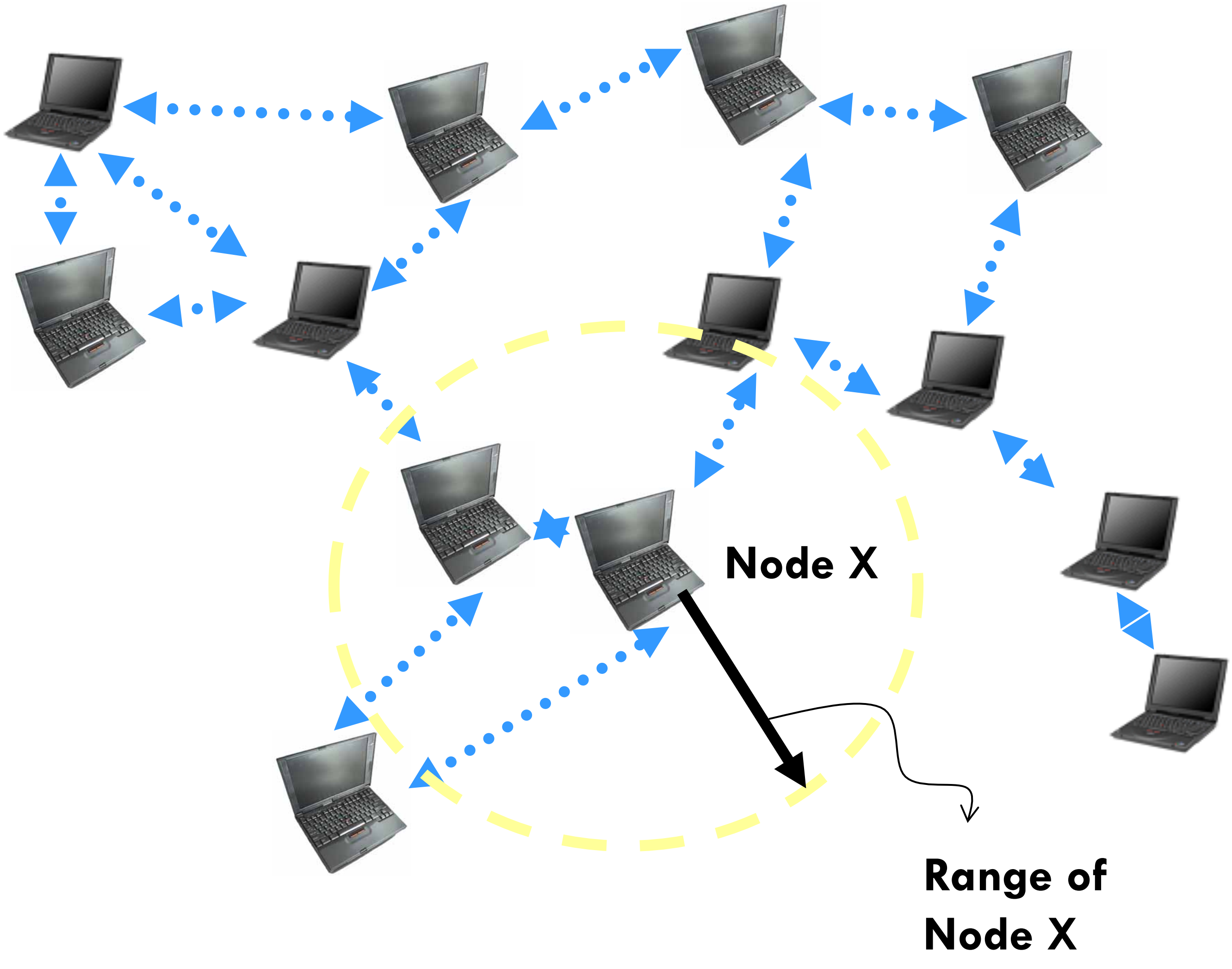
# Ad hoc Networks



- Ad Hoc Networks consist of peer to peer communicating nodes (possibly mobile) – no infrastructure.
- Topology of the network changes dynamically – links appear and disappear dynamically.
- Find application in military deployments, rescue operations, electronic classrooms etc.
- Can be interfaced with the Internet.
- Easy to deploy but difficult to maintain.

# An Example





# Different from Wired Networks



- Links are unstable and break and form often
  - Routing methods need changes
  - Multicasting is more challenging
  - TCP may not work well
- A large number of users trying to access a broadcast channel
- No centralized controller as in cellular or infrastructure networks.

# Routing Preview

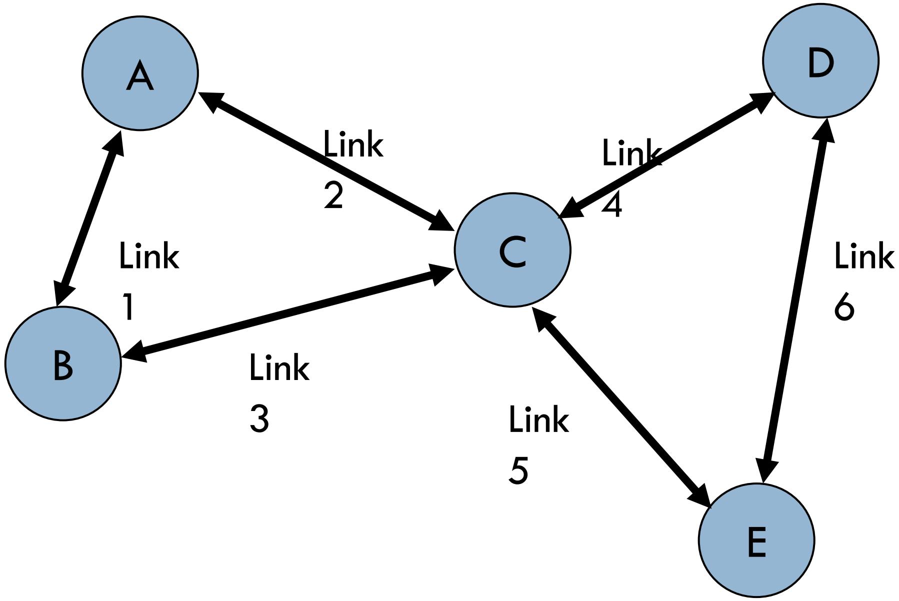


- Distance Vector based
  - Proactive
  - Bellman-Ford based
- On-demand based
  - On a need to find basis
  - Reduces the overhead due to route maintenance
  - However, there is route discovery latency

# An Example



- Consider D – it initially has nothing in its routing table
- When it receives an update from C and E, it notes that these are one hop away.
- Subsequent route updates allow D to form its routing table

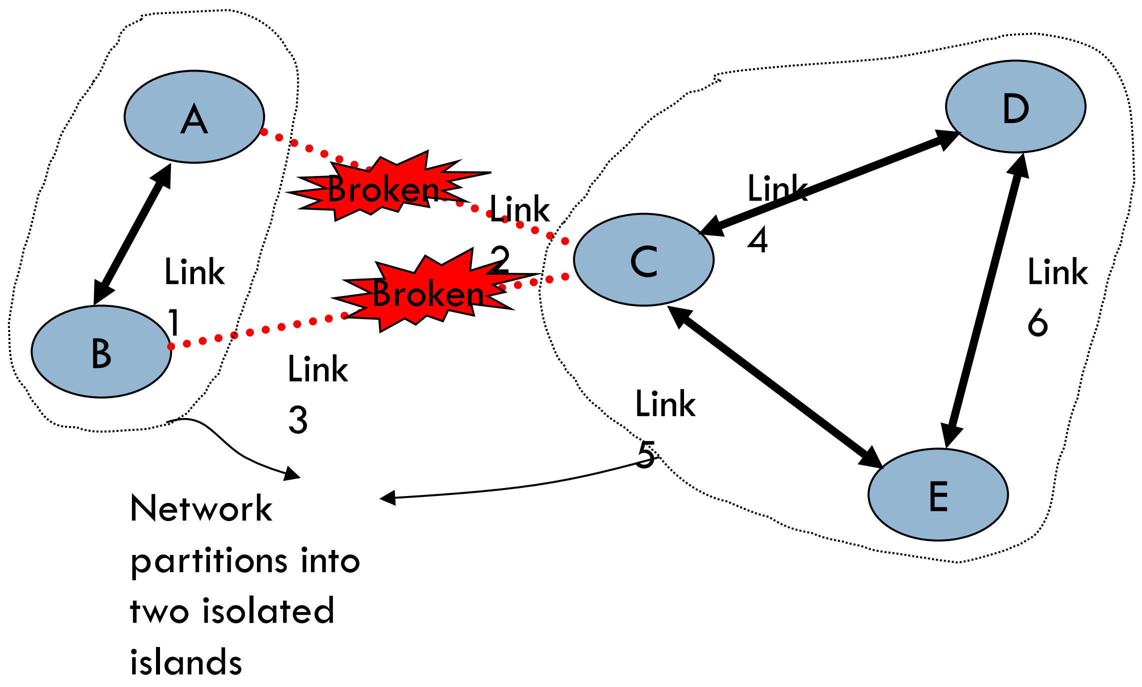



| Destination | Link | Hop count |
|---|---|---|
| A | 4 | 2 |
| B | 4 | 2 |
| C | 4 | 1 |
| D | local | 0 |
| E | 6 | 1 |

# Link failures and loops



- Disadvantage of distance vector routing is formation of loops
- Let Link 2 break and after some time, Link 3 break.



# What happens ?



- After Link 2 is broken, Node A routes packets to C, D and E through Node B.
- Node B detects Link 3 is broken.
  - Sets distance to C, D and E to infinity.
- Assume Node A in the meantime, transmits an update saying that it can reach nodes C, D, and E with appropriate costs (via node B)
- Node B thinks it can reach C, D and E via A
- Node A thinks it can reach C, D and E via B
- Routing Loop
  - Split horizon and Poison reverse – cannot handle all loops.

# DSDV: Destination Sequenced Distance Vector Routing



- In DSDV, each routing table update will contain for each destination node X
  - The address of X
  - The number of hops required to reach X
  - The latest sequence number information received with regards to X
  - This sequence number must have "originated" at node X
- Of all the paths with the same sequence number, the one with the minimum cost is chosen as the route.

# Handling link failures



- When a link is broken, the cost to any destination that is routed via this hop is set to infinity.
- A new broadcast routing packet is created
  - The sequence number is greater than the latest sequence number generated by the destination by “1”.
- When a node receives such a message, if it has a routing entry that reflects a sequence number greater or equal to the one advertised, it triggers a new broadcast
  - With that sequence number
- Indicates the presence of an alternate route to the destination.

# Route Advertisements



- It is possible that a node receives multiple copies with the same sequence number for a destination.
- Due to timing skews, a path with a “bad metric” may be discovered prior to the one with the “best metric”.
- Sending an update for each discovered path may lead to congestion.

# Wait prior to sending



- Thus, each node waits for sometime to collect information before propagating an update.
- Hopefully the best path discovered while waiting
- If a new routing update is received with a lower cost:
  - The new path is used for sending packets to the destination.
  - However, the new path is not advertised until a future time.
- Thus, each node maintains two tables – one for routing, one to be included in advertisements.
- Time for which a node waits until propagating an update is called “damping time”.

# Routing Information Packets

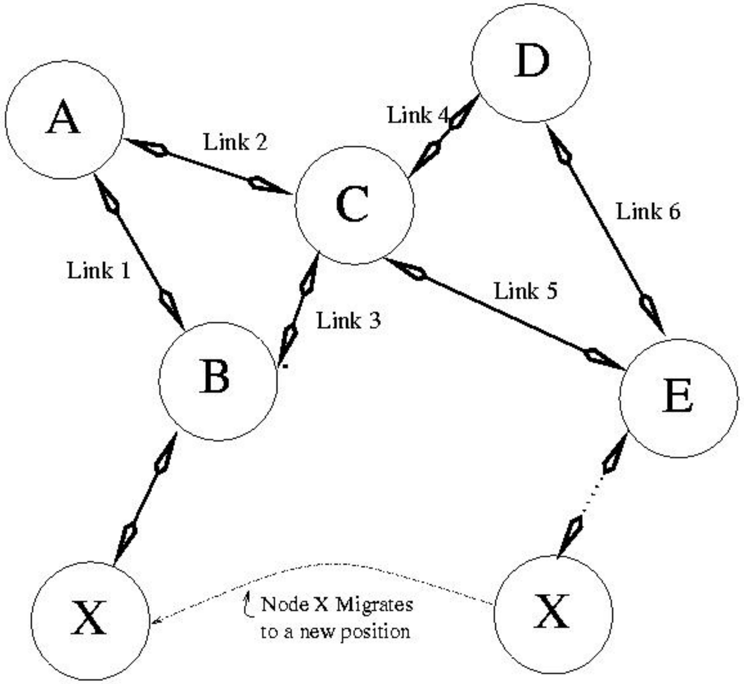


- There are two types
  - Full dump → Carries the entire routing table.
    - This is only transmitted infrequently.
  - Incremental update → Carries only the changes in routing tables
    - Broadcasted more often.

# An Example



- Node X moves from its first position where it is a neighbor of node B, to a new position where it is a neighbor of node E.



# Node A's routing table prior to move



| Destination | Next Hop | Cost | Sequence Number |
|---|---|---|---|
| A | A | 0 | 666 |
| B | B | 1 | 403 |
| C | C | 1 | 102 |
| D | C | 2 | 549 |
| E | C | 2 | 440 |
| X | B | 2 | 331 |

# Evolution of the routing table



- The sequence number entry indicates the last sequence number received from that destination.
- When node X moves, it generates a new routing update, that is sent to E.
- Node E propagates this to C, which in turn, propagates this to other nodes.
- Interim other changes may have occurred
  - Node B detects link failure to node X and triggers other route updates.
- The sequence numbers associated with other destinations could have changed as well.

# Evolved routing table at A



| Destination | Next Hop | Cost | Sequence Number |
|---|---|---|---|
| A | A | 0 | 746 |
| B | B | 1 | 538 |
| C | C | 1 | 212 |
| D | C | 2 | 633 |
| E | C | 2 | 540 |
| X | C | 3 | 424 |

# Purging routing table entries



- To prevent entries from becoming stale, they are purged if no updates are received.
- An additional column is used to indicate when the previous route update was received.
  - An indicator of when the entry needs to be purged.
- If not refreshed, the entry is purged and the cost to the destination is set to infinity.
  - Route updated generated to indicate this.

# Complexity



- Each node's update has to reach every other node in the network.
- Thus the number of messages grows as $O(n^2)$.
- The memory storage at each node is $O(n)$.

# Pros and Cons of DSDV



- Pros
  - Simple to implement
  - Good overhead/storage complexity
- Cons
  - Delays incurred prior to node knowing of a change in topology.
  - Frequent updates may be necessary.
  - Scalability

# On Demand Routing



- Table driven routing – overhead intense
  - Updates periodically
- Reactive or On Demand routing
  - Find route only when needed.
  - Eliminates need for updates
- Disadvantage is that there is a latency incurred in finding a destination.
- Second, route discovered may not be optimal.

# Dynamic Source Routing



- Source specifies entire route in packet header
- No need for intermediate nodes to have up-to-date routing information.
- Guarantees loop free routing
- DSR contains two basic mechanisms
  - Route discovery
  - Route maintenance

# Speeding up route discovery



- A node can enable the promiscuous receive mode
  - Enable underlying hardware to pick up any packet – not just those destined for the node.
  - Filtering packets based on addresses is disabled.
- This enables nodes to eavesdrop on route discovery packets (later) – and thereby cache routes.
- Caching routes speeds up route discovery

# Mechanisms of DSR

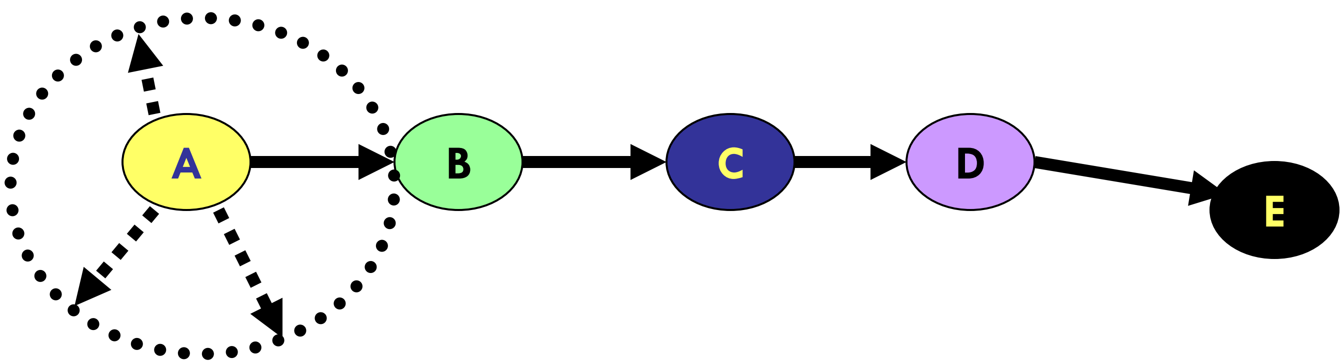


- Route Discovery
  - When a node (say S) wishes to send a packet to a destination (Say D) it searches for a route to D.
- Route maintenance
  - Detect if a source route being used becomes invalid due to link failures (resulting from mobility)
  - Use alternate routes
- Note that there are no routing updates
- When topology static – infrequent route updates

# Route Discovery in DSR



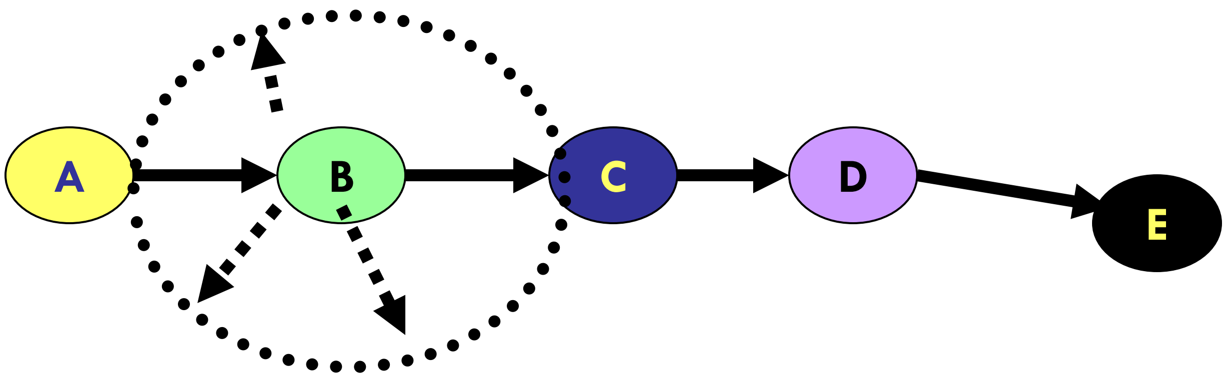
- When a node (say A) wants to route a packet to a destination (E)
  - First, look for route in cache; if available use it
  - If not, initiate a discovery process
- To discover route
  - Transmit ROUTE REQUEST as a “local broadcast” packet.
  - All nodes in range hear this packet.



# Route requests and replies



- A ROUTE REQUEST packet includes initiator ID and request ID.
- A route record which includes intermediate nodes is embedded into packet.
- When the target destination receives this request, it responds with a ROUTE REPLY message
  - Includes a copy of the route record in the ROUTE REQUEST in the response.
  - Response forwarded on the reverse route to the initiator.
- Initiator caches the route in the record upon receiving the ROUTE REPLY and uses that route.



# Role of intermediate nodes



- If a node receives a ROUTE REQUEST and notices that its ID is already in the route record, it discards the request.
  - Controls the flooding of the request.
- In some variants of DSR, it can respond to the source with a route if such a route exists in its cache.

# Support for unidirectional links



- DSR works with a simple modification even if the links in the network are unidirectional.
- In such a case, the destination has to perform a route discovery for the source if no route to the source exists in its cache.
  - When performing the route discovery, it piggybacks the ROUTE REPLY
    - If not there can be an infinite recursion of route discovery instantiations.

# Repeating route discovery

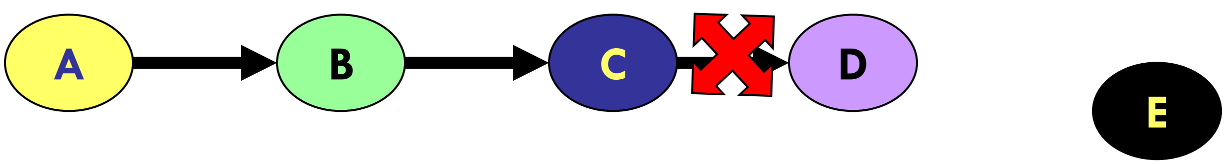


- If no timely response is received (in terms of a ROUTE REPLY), the source node re-attempts to discover a route.
- Care is necessary since the network may be disconnected and the destination may be in a different partition.
  - Exponential back-offs between route discovery attempts
  - Limited number of retries to ensure termination.

# Route Maintenance in DSR



- Relies on Link Layer Reliability
  - A is responsible for packets reaching B, B is responsible for packets reaching C and so on.
  - MAC Layer ACK
- If packet is retransmitted up to a maximum number of times on a link but no confirmation is obtained, a ROUTE ERROR message is generated and sent to source
  - In this example, C sends this message to A.
- Source removes the broken link from cache and looks for alternate route (either cache or new route discovery).



# Caching



- A node forwarding or overhearing may add routing information from the overheard packet into its own cache.
  - Complexities if unidirectional links exist.
- An intermediate node may respond to a route query with its cached route.
- Concatenates the partial route on which the ROUTE REQUEST is received with the cached route to the destination and sends a ROUTE REPLY.
  - This ROUTE REPLY is on behalf of the destination.
  - Care should be taken to ensure that there is no loop
    - None of the nodes on the path on which the ROUTE REQUEST was received should be on the route found in the cache.

# Route reply storms



- If nodes are allowed to respond with cached routes, this could result in reply storms
  - many nodes may try to respond
  - sub-optimal routes are more likely
- Node should wait before sending the reply to see if source begins to transmit on a shorter path than what is cached.

# Averting reply storms

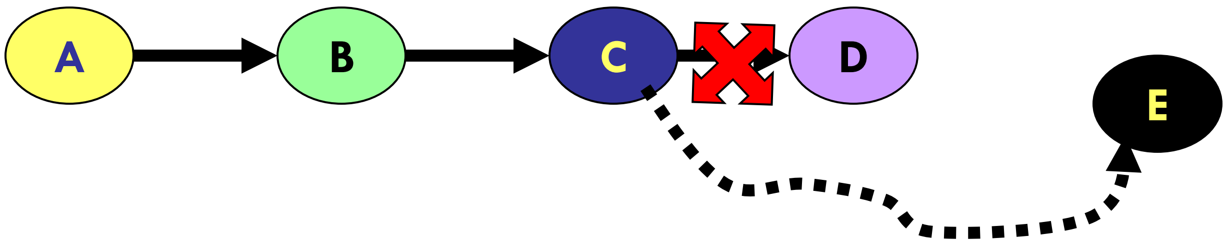


- A node delays its response for a random time
- This delay is $d = H \times (h - 1 + r)$
  - h is the length of the route to be reported.
  - r is a random number between 0 and 1
  - H is a small constant delay (per hop)
- The method randomizes the time at which a node sends its response.
- Note that shorter the route to be reported, the shorter the chosen delay.
- In addition, a ROUTE REQUEST may contain a “hop limit” to limit the number of intermediate nodes sending responses.

# Salvaging packets upon failure



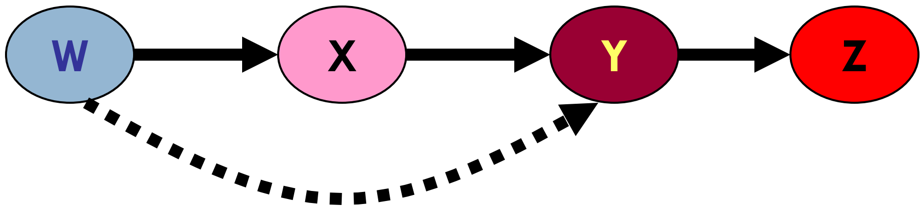
- Upon link failure, the node detecting failure may try to salvage packets by using alternate routes that are possibly cached.
- Packet is typically marked as salvaged
  - Prevents re-salvaging which leads to loops – since cached routes could sometimes become stale.
- Note: Route error still sent back to source.



# Automatic route shortening



- Source routes may be automatically shortened if one or more of the intermediate relays become unnecessary.
  - Later node overhears a packet carrying a source route
  - In the example,  Y hears W's packets.
  - Y then sends a "gratuitous ROUTE REPLY" to the source, indicating the shorter route.



# Better failure handling



- Source piggybacks information in the ROUTE ERROR message on its new ROUTE REQUEST
- Allows other nodes to know about the failure and purge cache entries.
- Nodes overhearing the ROUTE ERROR messages can also refresh their cache entries.

# Pros and Cons of DSR



- Pros
  - DSR provides simple, loop free routing.
  - Scales with the number of connections.
  - Smart techniques such as caching help.
- Cons
  - Scalability
  - Appropriate values for purging etc. hard to determine.
  - Large header in packet since it has to carry information about the entire route.

# AODV: The Ad hoc On-Demand Distance Vector Protocol



- Tries to combine the good properties of DSDV and on-demand routing.
- Link formations or deletions do not result in system-wide broadcasts (like DSDV)
  - Localize the effects of changes
- Packets do not have to carry entire route (like DSR)

# AODV properties in short



- Routes are discovered only on a need-to basis
  - It is on demand in spirit
- Provides loop free routing using destination-based sequence numbers as with DSDV
- Inherently assumes bi-directional links.
- AODV routing tables :
  - Contain destination and next-hop IP addresses and the destination sequence number
  - In addition, for each destination, a node maintains a list of precursor nodes (route from source to current point).
    - Needed for route maintenance as we will see later.

# Unicast Route Establishment



- To find a destination broadcast an RREQ message
- Any node with a current route to the destination (including the destination itself) can respond with an RREP message.
- Route information is maintained by each node in its routing table.
- Information obtained via RREQ and RREP is kept in routing tables.
- Sequence numbers are used to eliminate old routes.
- Routes are also aged out of the system.

# Details of Route Discovery



- Node first checks to see if it has a route
- If not, it creates an RREQ message
  - contains its own ID, a sequence number, destination address, last known sequence number for destination and broadcast ID.
- It broadcasts this RREQ message and sets a time-out.
- When a node gets this RREQ it checks if it has seen the source ID, broadcast ID pair.
  - If yes, discards message
  - If no, record the information and process (next slide) the packet.

# Processing and responding to RREQ

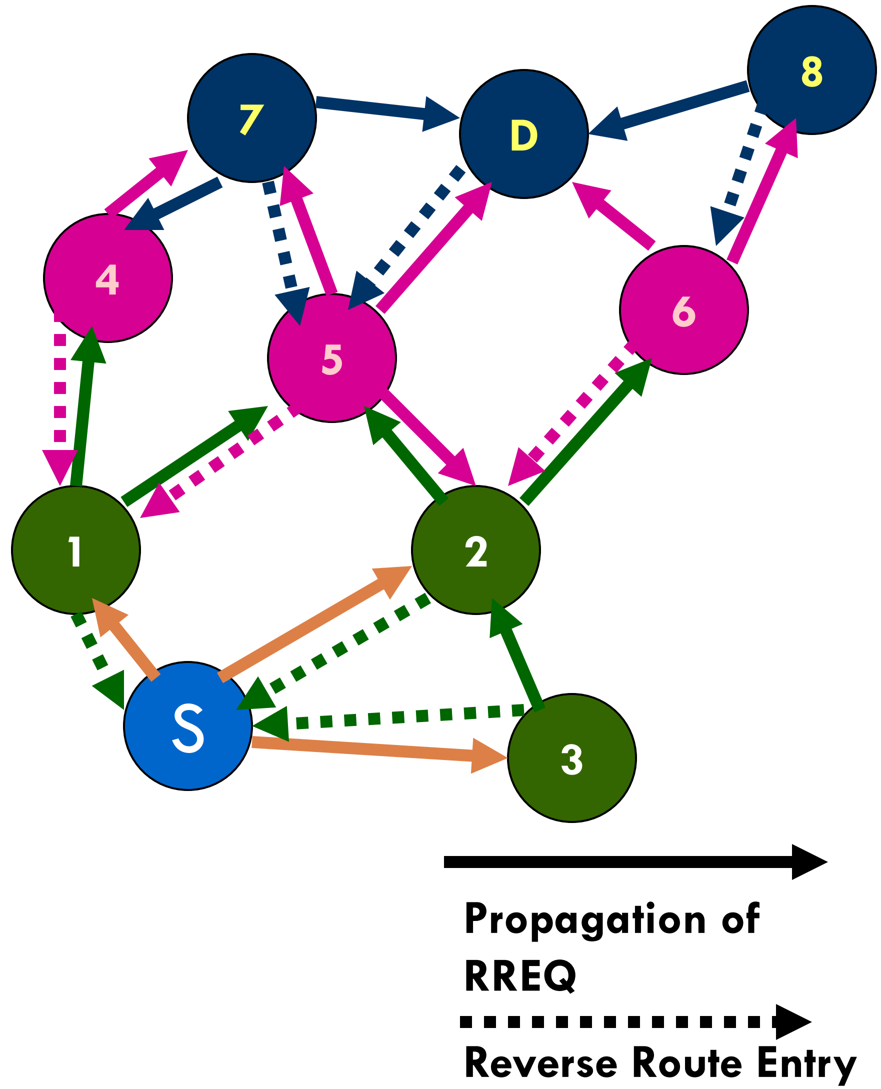


- A node that gets the RREQ sets up a “reverse route” entry for the source in its route table
  - A lifetime is associated with the entry
- It checks if it can respond to the RREQ
  - It must have an unexpired routing entry to the destination to do so.
- If yes, respond with an RREP
- If no, increment hop-count in the RREQ and broadcast the packet to its neighbors.
- If RREQ is lost, i.e., no response received within the time-out, source re-attempts discovery.
  - After a pre-specified number of re-attempts it gives up.

# Example of Route Discovery



- S sends RREQ to 1, 2, 3
- Each establishes a reverse path to S and then forward the RREQ to 4, 5 and 6.
- These nodes establish reverse paths and forward the RREQ and so on.
- Finally, destination D responds along D, 5, 1, S.



# Expanding Ring Search



- Flooding is expensive
- Use small TTL (Time to Live in terms of hop count) first and hope the destination can be found within the TTL.
  - Set time-out and wait.
- If no response is received, increase TTL and try again.
  - RREQ propagates further
- This is called expanding ring search.
- Good if destination is closer than further off.
- If destination is far, can give a performance worse than flooding.

# Responding with an RREP



- If the destination is responding to the RREQ, it includes its "new" sequence number in the RREP message, sets hop-count to `0' and includes a lifetime for which the route is valid.
- If an intermediate node responds to the RREQ, it includes the latest sequence number known for the destination in the RREP message and the hop-count to its own hop-count from the destination.
  - It also includes information on the lifetime for which the route is valid (as per its entry)

# When an RREP is received ..



- When an intermediate node receives an RREP, it sets up a “forward path” entry to the destination in its routing table.
- When a node receives multiple RREPs, it forwards the first one.
- A subsequent RREP is forwarded only if the new RREP contains a greater sequence number (more up to date message) or a smaller hop count (better route).
- Source node begins data transmission as soon as it receives the first RREP message and then, if it discovers a better route, switches to the new route.

# Route maintenance



- A path being used is called an active path.
- Movements that are not along any active path do not trigger any protocol action.
- If the source node moves, it can reinitiate route discovery to re-establish the connection.
- When either the destination or some intermediate node moves, a RERR message is sent to the affected source nodes.
- The RERR message is initiated by the node upstream of the failed link (as with DSR).

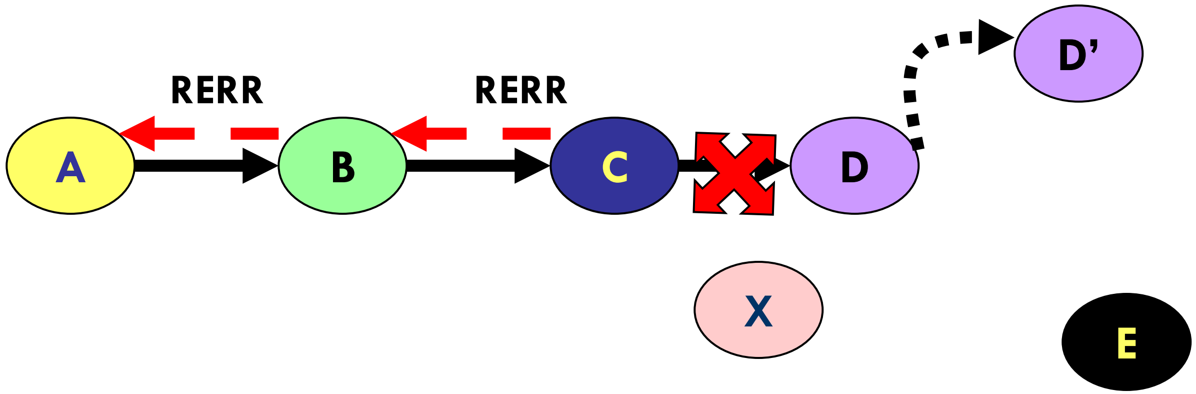
# RERR messages



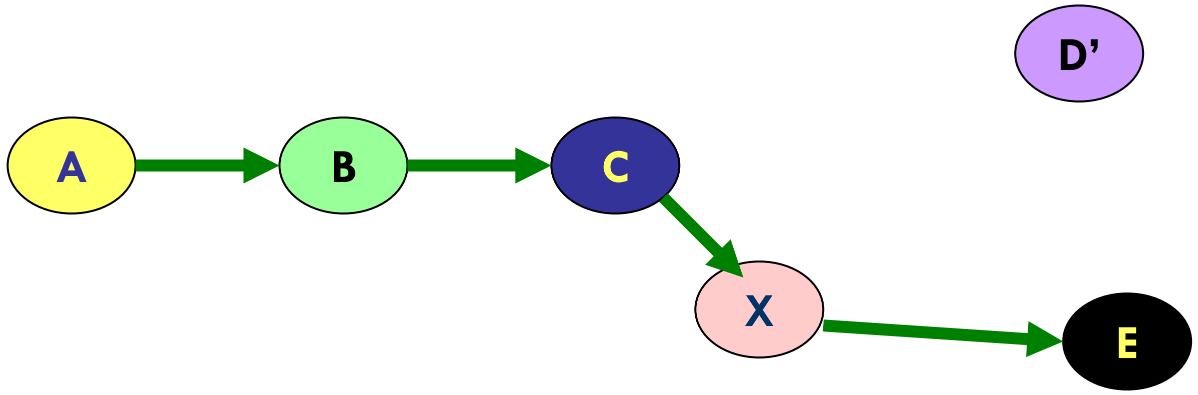
- The RERR message would list the destinations that are now unreachable.
- If the node upstream of the broken link has one or more precursor nodes for destinations, it sends the RERR to those nodes.
- Each such message is now propagated towards the appropriate source.
- Source re-initiates route discovery.
- Note here that routing entries with an infinite metric are not immediately deleted.
  - Destination unreachable is also useful information !

# Example






• Route Failure due to motion of D triggers a RERR message from C to A.




• Subsequently, a new Route Discovery by A finds the new route to E via X.

# One final detail...



- AODV uses HELLO packets to keep track of neighbors
  - Periodically broadcasted
  - Easier to detect link breaks.