Routing in Wireless D2D Networks

Prof. Ioannis Stavrakakis

D2D Networks

- Nodes are computing and communication devices, e.g laptops, PDAs, mobile phones, even sensors
- Nodes organize and maintain the network by themselves
- A node is both a host and a router
- IP is used at the network layer
- First and foremost issue: **routing**

Problems with Routing

1. **Self-organizing networks**
   - Need for distributed algorithms
2. **Topology changes dynamically**
   - Mobile nodes (joining in or leaving)
   - Unannounced loss of network connectivity due to the time-varying channel nature
3. **Link failure / repair**
   - Network partitions
   - Loop formation during temporary node failures

Problems with Routing

4. **Asymmetric links:** links may be unidirectional
5. **Limited bandwidth**
   - Protocols using flooding create high traffic and control overhead
6. **Different performance criteria**
   - Number of hops (delay)
   - Available Bandwidth
   - Route stability despite mobility
   - Energy consumption

Remarks/Assumptions

1. Find/maintain a route provided it exists
   - Another problem is to ensure that a route exists, e.g., through power control [Ramanathan'00,Wattwnhofer'00]
2. Assume that nodes are willing to cooperate

6

Routing Protocols for D2D

- **Topology based routing**
  - Proactive approach, e.g., DSDV, OLSR
  - Reactive approach, e.g., DSR, AODV
  - Hybrid approach, e.g., Cluster, ZRP

- **Position based routing**
  - Location Services: e.g., DREAM, Quorum-based, GLS, Home zone etc.
  - Forwarding Strategy: e.g., Greedy, GPSR, RDF, Hierarchical

7

The Proactive Approach

- Attempts to build and maintain consistent, up-to-date routing information from each node to every other node in the network, BEFORE a route is needed.
  - Respond to changes by propagating updates throughout the network
  - Good for connection-less traffic where you may send traffic to any node at any time
- Based on distance vector or link-state mechanisms

8

The Reactive Approach

- Routes are only created when desired by the source node
- Two-stage procedure
  - **Route discovery protocol:** route discovery ends either when a route is discovered or all possible communication paths have been examined
  - Once a route is established, it's maintained by some sort of **route maintenance protocol**

9

Trade-off

	Proactive Approach	Reactive Approach
Latency	Low	High
Overhead	High	Low

- Reactive more suitable for a mobile environment with limited bandwidth
- Proactive preferred when time constraints are important

10

Proactive Protocols

DV for fixed networks, DSDV for wireless networks

11

Distance Vector Routing

- Route discovery algorithm used by RIP (**DVR does not address route maintanance**)
- Based on Bellman-Ford or Ford-Fulkerson algorithms but DVR does not require an a-priori knowledge of the network

12

Distance Vector Routing (1)

- Let's consider that **cost coincides with distance, i.e., number of hops** (it could be expressed also in terms of bandwidth, latency,…)
- **Each node** has a **routing table** where distance to **all** **r**eachable **destinations** is recorded
  1) Each table entry contains:
     a) Destination node ID (i.e., IP address)
     b) Next hop
     c) Distance (number of hops) to the destination
  2) Each node keeps track and informs its neighbors of its distance to every destination (main idea of distance vector-based protocols)
- A router updates its distance to a destination based on its neighbors distance to the destination

13

Distance Vector Routing (2)

1. Initially, each node sets the cost of the link to itself to 0 and to any other node to infinity
2. A node broadcasts its routing information (list of destinations and distance)
3. Upon receiving information from neighbors, a node computes the minimum cost-path toward destination and makes changes to its routing table if needed
4. Either periodically or every time a node updates its table, it broadcasts routing information to its neighbors (then the procedure repeats from step 2)

- The algorithm **converges by iteration**

14

Distance Vector Routing (2)

1. Initially, each node sets the cost of the link to itself to 0 and to any other node to infinity
2. A node broadcasts its routing information (list of destinations and distance)
3. Upon receiving information from neighbors, a node **computes the minimum cost-path toward destination** and makes changes to its routing table if needed
4. Either periodically or every time a node updates its table, it broadcasts routing information to its neighbors (then the procedure repeats from step 2)

- The algorithm **converges by iteration**

15

More Details on Step 3

- At **node i and time step n+1**, the minimum cost path between nodes **i** and **j** is computed as

$$D^{n+1}(i,j) = \min\left\{D^n(i,j), \min_{k \in N}\left\{d(i,k) + D^n(k,j)\right\}\right\}$$

where:

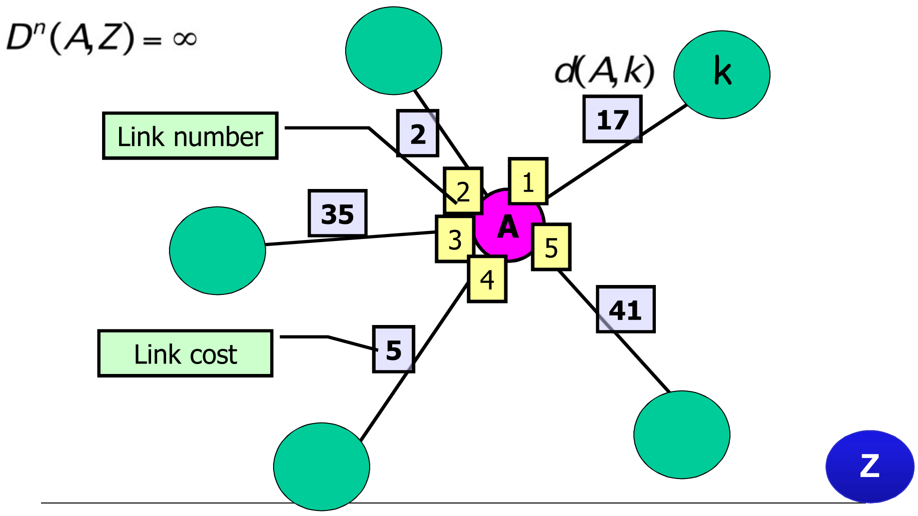
$D^n(i,j)$ is the path cost from *i* to *j* at time step *n*

$d(i,k)$ is the cost of the link between *i* and its neighbor *k*

N is the set of *i*'s neighbours

16

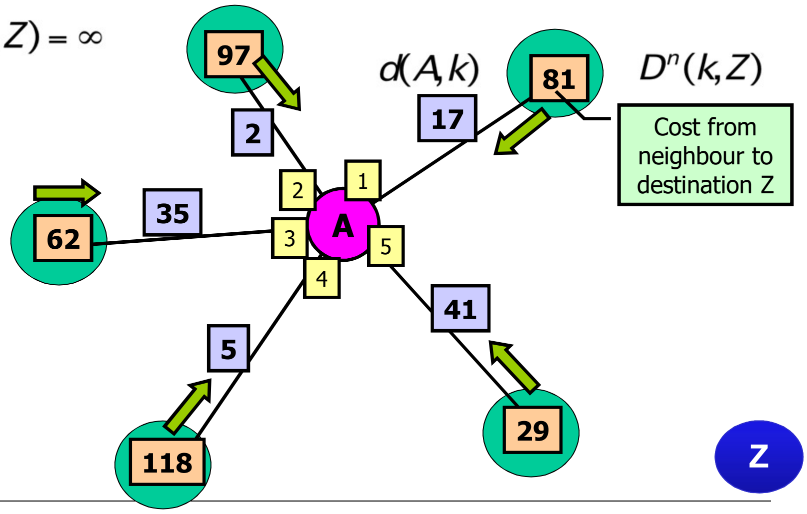
Distance Vector Routing for Address "Z"



17

Distance Vector Routing for Address "Z"

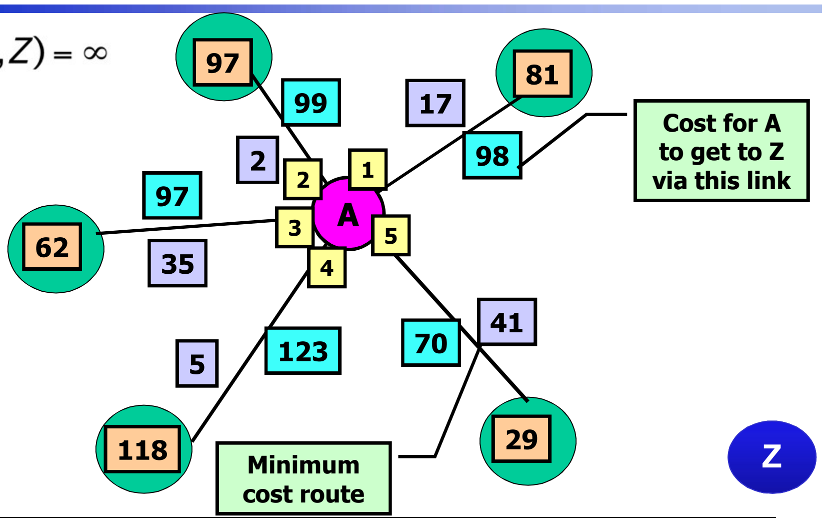
$D^n(A,Z) = \infty$



18

Distance Vector Routing for Address "Z"

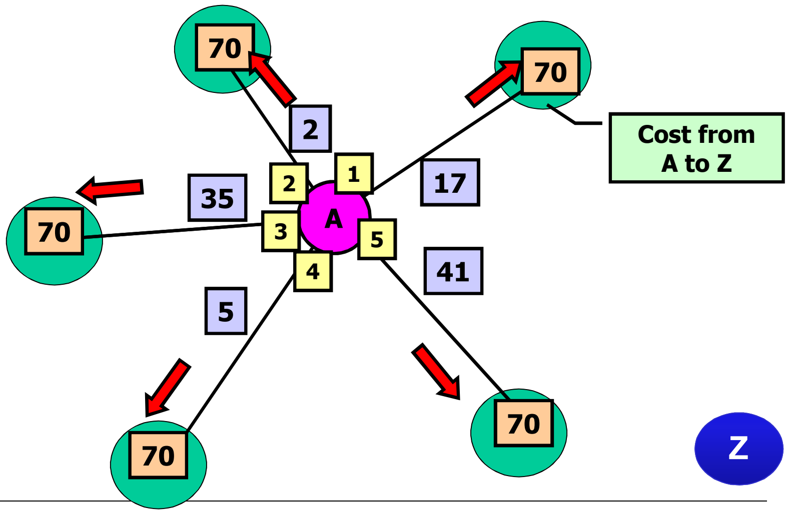
$D^n(A,Z) = \infty$



$$D^{n+1}(A,Z) = \min_{k \in N}\{d(A,k) + D^n(k,Z)\}$$

19

Distance Vector Routing for Address "Z"



20

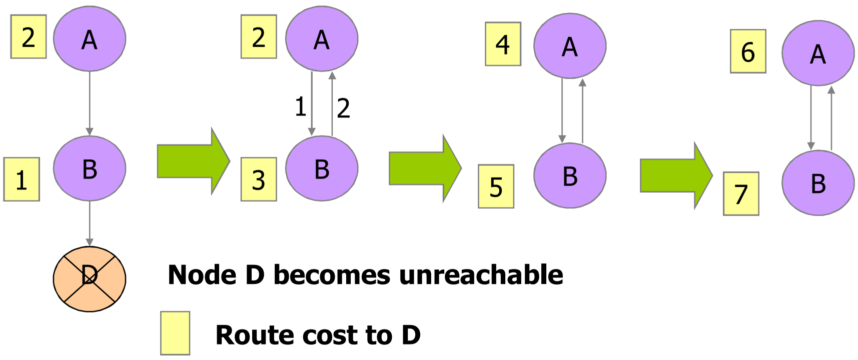
Problems with Distance Vector

1. **Does not scale well** with the number of nodes in the network (overhead $O(n^2)$)
2. **Slow convergence** to the lowest cost route
3. **Slow recovery time** if there are link failures, hence routing problems during disruption times
   - **Count-to-infinity**: Router loops may take place when degradation of a link cost occurs
     - Solved only when the cost of the alternative route including a loop reaches the cost of the degraded link

21

Count-to-Infinity: An Example

Cost of the link A-B and B-D equal to 1 => cost of the path between A and D equal to 2



It occurs only if A sends its update before B

22

DSDV (1): Destination Sequenced Distance Vector Routing

- Proposed by Perkins ['94], based on Bellman-Ford routing mechanism
- Each node maintains a routing table that records all possible destinations
- Each table entry contains
  1. Destination node ID (i.e., IP address)
  2. Next hop
  3. Number of hops to the destination
  4. A sequence number (SN), used to distinguish "old" vs. "new" routes and avoid loops

23

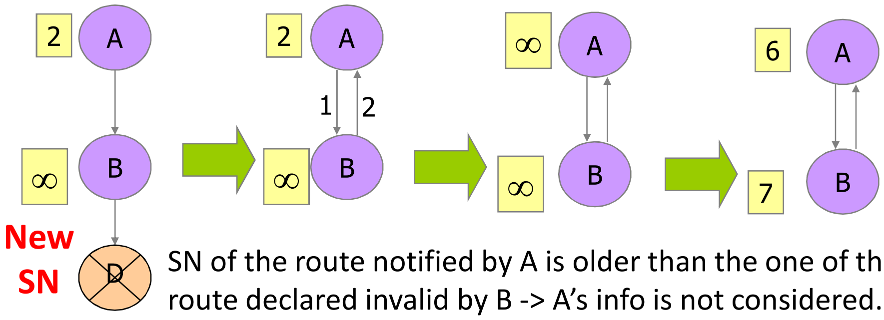
DSDV (2)

- Upon reception of a route update:
  - If the node doesn't have such information, it adds an entry to its table
  - Otherwise, it checks the SN and updates the table only in case of fresher information, i.e., the route with most recent SN is used
  - If two routes have the same SN, the route with the smaller metric (== shorter route) is used
- When a link fails, an ∞ metric is used and the route SN is increased

24

Count-to-Infinity: An Example

Cost of the link A-B and B-D equal to 1 => cost of the path between A and D equal to 2



SN of the route notified by A is older than the one of the route declared invalid by B -> A's info is not considered. B sends an update to A, and A too declares the route to D (through B) as invalid

25

Detecting Link Failure

- Detection:
  - If data is flowing over a link, failure of link layer to deliver a packet can be used to assume link failure
  - If a node does not hear for a long time (how long?) from its neighbor

Proactive Protocol

OLSR

OLSR

Optimized Link State Routing [Jacquet'00]

- OLSR is based on the link state routing approach
- Traditional approach in Link State Routing:
  - Every node generates control packets to advertise its links status (i.e., its one-hop neighbors and the links cost)
  - Such information is propagated over the whole network (flooding)

30

OLSR

Optimized Link State Routing [Jacquet'00]

- However, in OLSR:
  - Sources build routes proactively by using only **MultiPoint Relay** nodes (MPRs)
  - Only MPRs need to **generate and forward** link state updates
- OLSR thus leads to efficient flooding of control messages in the network: superfluous broadcast packet retransmission as well as the size of the link state packets are reduced

31

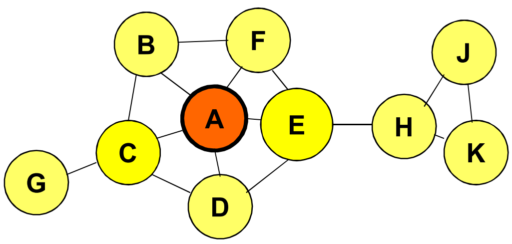
MPRs and Their Selection

- Every node in the network selects its own MPR(s)
- A node selects its MPRs among its 1-hop neighbors so that it can reach all nodes that are within 2-hop away, through symmetric links
- Nodes exchange 1-hop neighbor lists to know their 2-hop neighbors and link type, and choose the MPRs
- **No. of MPRs per node should be minimized**

32

MPRs: Example (1)

- Hp: **all links are bi-directional**

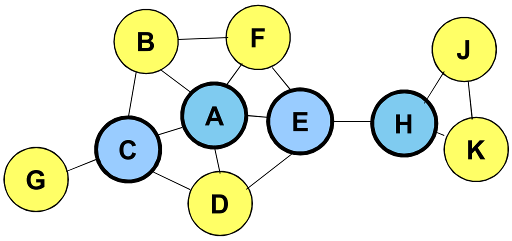


Nodes C and E are multipoint relays of node A

33

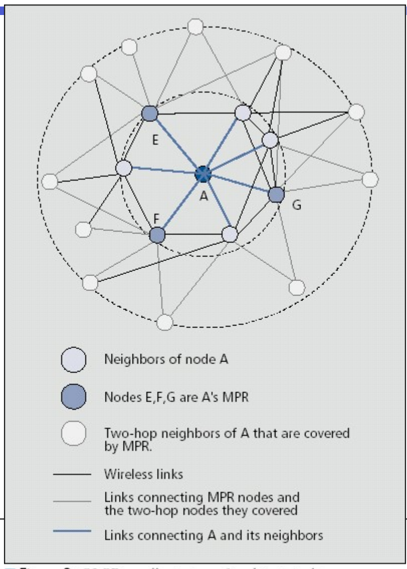
MPRs: Example (2)

- C selects A as MPR, while E selects A and H
- Node E is a multipoint relay also for node H
  - If the link H-J were unidirectional, then K would also be selected as MPR by H
  - Indeed, OLSR never uses unidirectional links



34

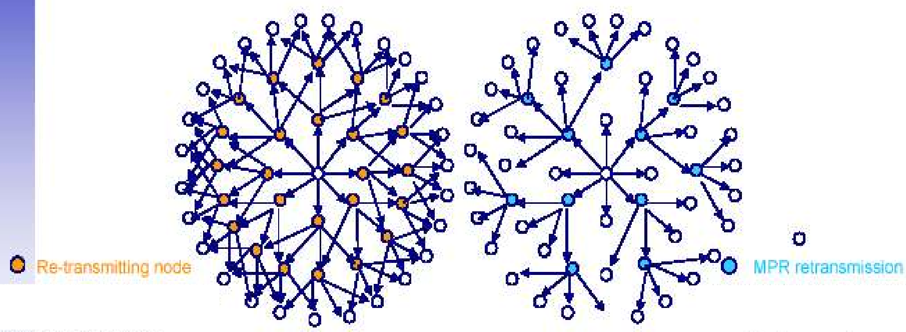
MPRs: Example (3)



■ Figure 2. *OLSR: an illustration of multipoint relays.*

35

Controlled Flooding of Link State Updates



25 retransmissions to deliver a message up to 3 hops

11 retransmissions to deliver a message up to 3 hops

Controlled Flooding of Link State Updates

- OLSR is particularly suited for dense networks
- In sparse networks, every neighbor becomes a multipoint relay, then OLSR reduces to pure link state protocol

37

How It Works: Hellos

- The generic node X broadcasts Hello messages every Hello interval to its 1-hop neighbors:
  - Hello message contains list of known 1-hop neighbors and
  - the link status (symmetric, asymmetric, or MPR) of its 1-hop neighbors

How It Works: Neighbor Table

- Node X builds a **Neighbor Table** that includes all its 1-hop neighbors and, for each of them, all 2-hop neighbors that can be reached through it. The link types are also recorded.
  - X selects its MPR nodes among its 1-hop neighbors such that it can reach all nodes that are within 2-hops away through symmetric links
  - Once X has selected a neighbor, say Y, as MPR, X tags its link with Y as MPR and, through the next Hello, X will notify Y about it
  - **Y maintains the list of nodes that selected Y as an MPR**

How It Works: Topology Control

- MPR nodes generate and broadcast Topology Control (TC) messages every TC interval to advertize link states, specifically
  - A TC message contains list of 1-hop neighbors who have selected this node as an MPR
  - Only MPR nodes can forward TC messages (but note that all network nodes connected through symmetric links will receive them) -> more efficient flooding
  - Nodes receiving TC messages use them to build their **Topology Table** (and for routing table calculation afterwards)

40

Some Remarks

- Control messages do not require a reliable transmission since they are periodically sent
- Each control message includes a sequence number so that old messages with outdated information can be detected and discarded

41

Some Observations (1)

- Routes are **all composed of MPRs** only (except source and destination): MPRs form a network backbone that is in charge of routing all traffic
- Asymmetric links are never used
- MPR nodes selection impact:
  - How much more traffic must MPR nodes handle? Higher load leads to higher energy consumption
- Node mobility impact
  - Consequences? Particularly for MPR nodes

Some Observations (2)

- Hello interval impact on overhead vs. protocol reactivity:
  - Recall: Hellos are sent by all nodes to their 1-hop neighbors (but they are not rebroadcast)
- TC interval impact on overhead vs. protocol reactivity:
  - Recall: TC messages are sent only by MPR nodes to advertize link state but they reach all network nodes

Reactive Protocols

DSR
AODV

DSR
Dynamic Source Routing [Johnson'96]

- **Entirely "on demand":** No periodic messages or advertisements at any layer
- **Dynamic:** it maintains a "soft state," i.e., all state is discovered when needed and can be easily re-discovered if needed after a failure without impacting the protocol functioning
- **Source Routing**: the source specifies in the header of each data packet the entire route, not only the next hop (**no need for routing tables**)

Assumptions

- The network is fairly small (up to 200 nodes, i.e., network diameter of 10-15 nodes)
- Each node maintains a cache containing all source routes of which it is aware
  - Routes in the cache are continuously updated as they are learned
  - Several routes can be cached to the same destination
- Either bi- or uni-directional links are present

DSR: Two Phases

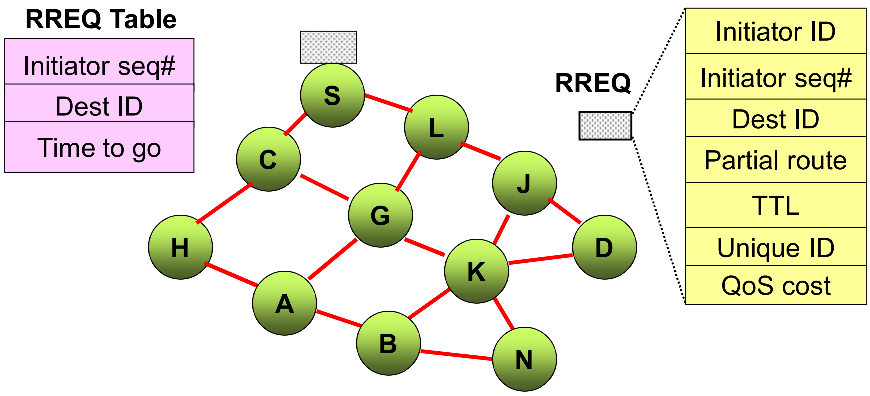
The protocol consists of two phases:

- **Route discovery:**
  - Started by **S** when **S** needs to send data to **D** and doesn't have any route to **D** in cache
- **Route Maintenance:**
  - **While using a route to D**, **S** can detect if the route is not longer valid and, in case, send an error message
  - Upon route failure, **S** may use another route (if it knows it) or start a new route discovery

DSR

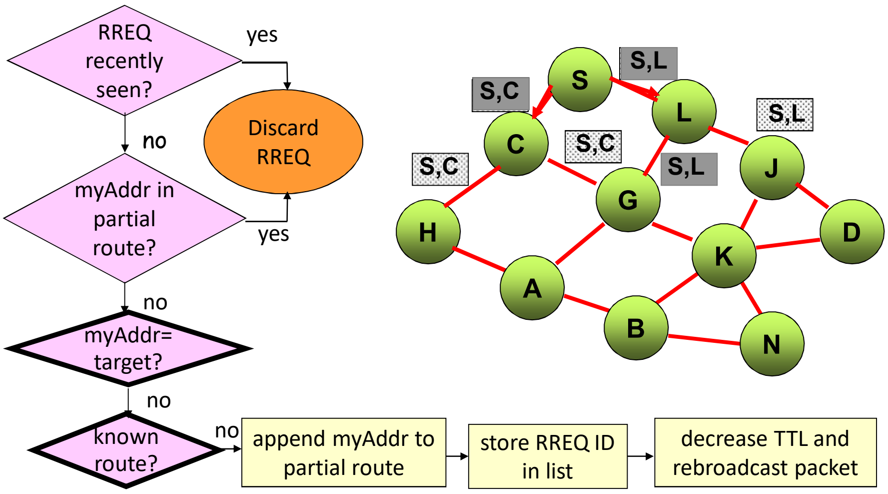
- DSR uses four control messages:
  - Route REQuest packets (RREQ)
  - Route REPly packets (RREP)
  - Route ERRor packets (RERR)
  - ACKnowledgments (ACKs)
- RREP can be piggybacked to RREQ packets and ACKs to IP data packets

Route Discovery



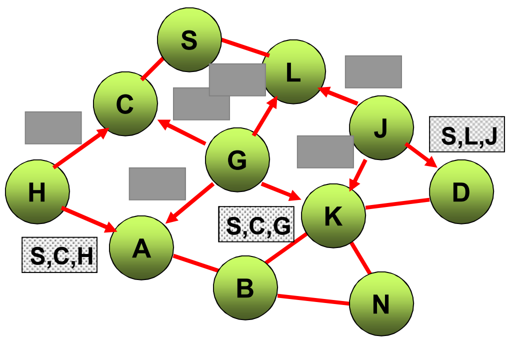
If no reply by the “time to go”, **S** sends a new RREQ till a max no. of attempts have been made

Route Discovery



54

Route Discovery



Route discovery succeeded !

55

Route Reply (1)

- Once the message gets to the destination or reaches a node with an unexpired cached entry, a **RREP** is sent
  - If it is an intermediate node, it appends the cached route to the route record
  - If it is the destination, upon receiving the RREQ, it places the final route record into the RREP
- The route cost is returned by **D** (or intermediate node) in the RREP
  - Cost depends on the target QoS metric (energy consumption, latency, bandwidth, etc.)

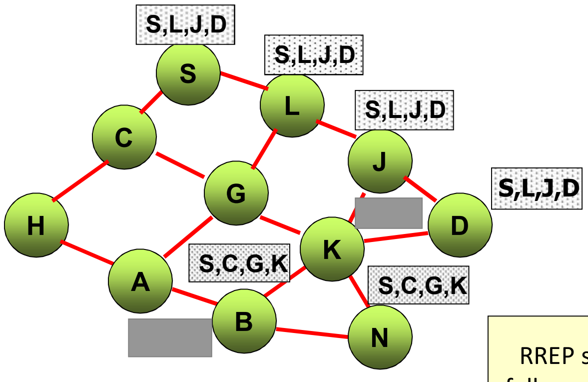
56

Route Reply (2)

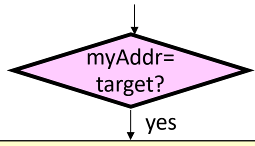
- The replying node must have a route back to the source
  - RREP can be sent on the route obtained by *reversing* the route appended to received RREQ
  - If it's a unidirectional link, the node can use any route it knows to the source
  - If the node does not have any route to the source, it performs a route discovery. The route reply is piggybacked on the new RREQ

57

Route Reply



RREP



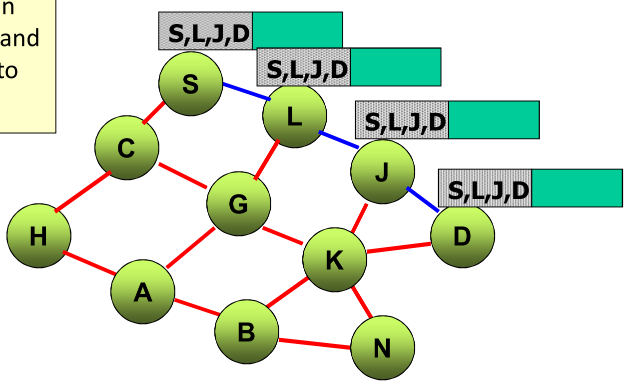
Place route record and cost into RREP (cost depends on target QoS metric)

RREP sent by *reversing* the route followed by RREQ. If unidirectional, use any cached route to S. If none, send an RREQ and piggyback RREP

58

Data Delivery

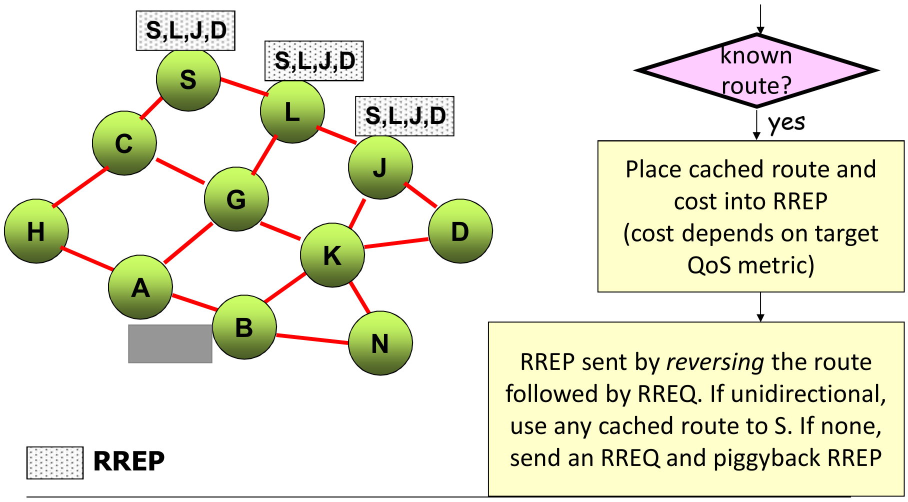
Write route in packet header and send packet to next hop



DATA

59

Route Caching



60

Route Caching

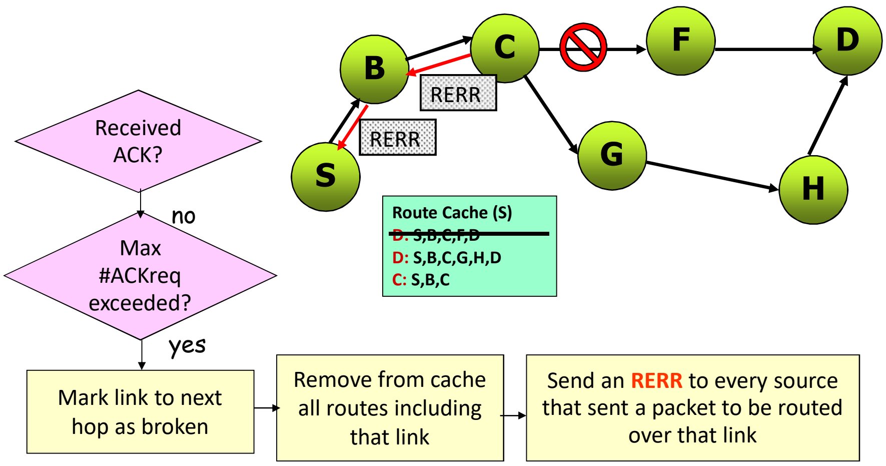
- **S** caches the route to **D** contained in the RREP
  - For each **D**, more than one route can be cached, thus **S** can perform traffic routing over several possible routes
- Advantages of route caching
  - Speeds up routing
  - Reduces propagation of route requests

61

Route Maintenance

- Along a source route, each node transmitting a packet is responsible for correct transmission confirmation
- An ACK is needed for confirmation
  - MAC or link-layer ACKs can be used
  - Overhearing next node forwarding the packet (**passive ACK**: A hears B sending to C)
  - Use of DSR-ACKs: the node transmitting the packet can require an ACK (which may follow a different route back)

Route Maintenance



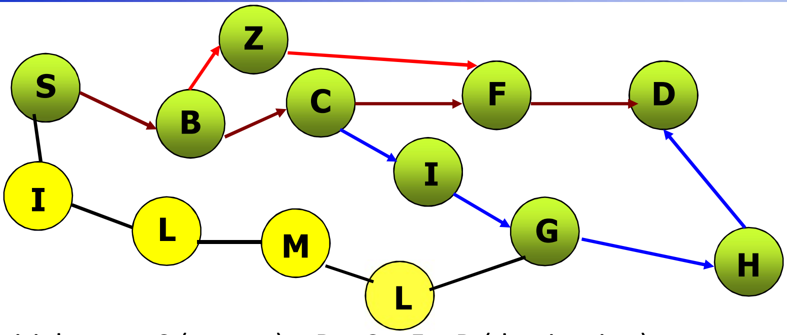
Route Maintenance

- Nodes overhearing RERR remove from their cache all routes including the broken link
- **S** uses another route from its cache, if it has any; otherwise it starts a new route discovery
- The transport layer (e.g., TCP) should take care of data retransmissions

Optimizations

- **Promiscuous mode:** listening to arbitrary routes in use (A hears B sending to C)
  - Replace with shorter routes / Store new routes
  - But promiscuous mode takes energy
- **Packet salvaging**
  - Upon link failure
  - If intermediate node has another route to **D**, it replaces the original source
  - However, no double salvage is allowed !!! (flag in the packet header indicates if the packet has been saved already once)

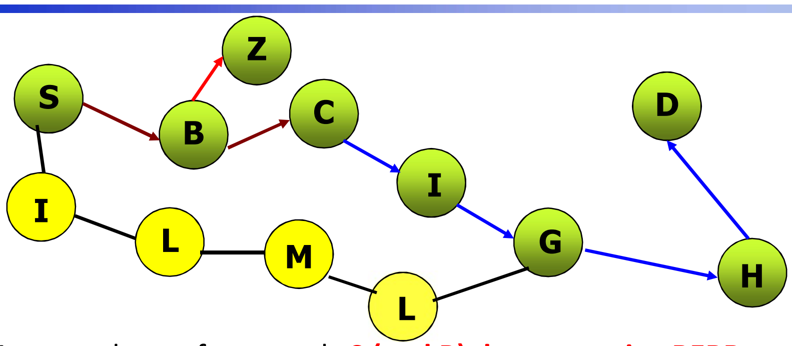
Packet Salvaging: An Example (1)



- Initial route: S (source) – B – C – F – D (destination)
- However, S and C have an alternative route to **D** in their cache (S-B-Z-F-D and C-I-G-H-D, respectively)
- At a certain time, F moves out of the network

67

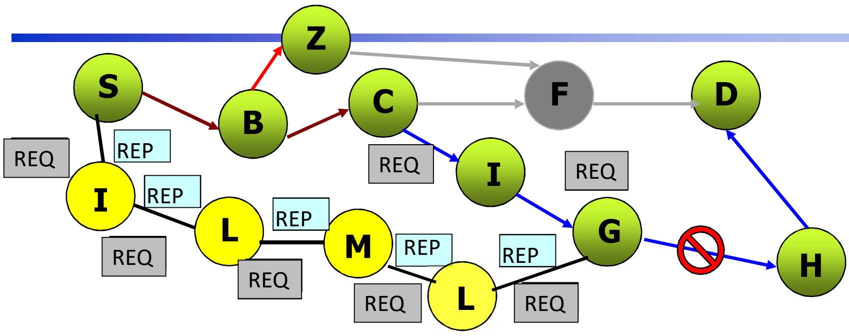
Packet Salvaging: An Example (2)



- Assume that unfortunately **S (and B) do not receive RERR**, from either C or Z, then S keeps sending its packets toward C
- C can salvage the packets **by replacing the original route** (**S**-B-C-F-**D**) in the packet with the new route (**C**-I-G-H-**D**)

68

Packet Salvaging: An Example (3)



- Now, assume the G-H link fails, then IF G could save the packet again, a loop could have been formed. Indeed
  (i) G does not know that the packet was originated by S (it only sees C as originator),
  (ii) S could respond to a RREQ by G with route S-B-Z-F-D (If C does not send the RRERR or its RRERR does not reach S before G's RREQ)

DSR: Advantages

- Enrirely reactive
- "Soft state" and support of unidirectional links
- Source routing
  - No need for routing decisions by intermediate nodes
  - Nodes can learn new routes from relayed packets
  - Guarantee that routes are loop-free
- Caching
  - Reduced route discovery overhead
  - One route discovery may yield many routes to D, due to intermediate nodes replying from local caches

DSR: Disadvantages

- RREQ flooding may be huge
  - Possible collisions between RREQs propagated by neighboring nodes
  - Contention between RREPs come back due to use of local caches (RREP *Storm* problem)
- Packet delays/jitters due to on-demand routing
- Headers may become too long with respect to data payloads -> suitable for small networks
- Cached routes may become invalid; stale caches can adversely affect performance

72