Routing in MANETs

CS-6777 Mobile Ad Hoc Networking

Memorial University of Newfoundland

Why Routing?

- MANETs are multi-hop and dynamic
- How do we send packets to a destination in such networks?
  - Flooding – too expensive
  - Unicasting – gotta be smart!
- Routing
  - To find, use and manage multi-hop paths for forwarding data on behalf of specific end systems to particular destination systems

Relevant Factors

- Number of peer nodes
- Type and degree of link dynamics
- Expected user traffic patterns
- Network density
- Lower layer technologies
- Mobility

What do MANET Nodes and Networks Look Like?

- Single interface device is acceptable
  - Rather different from wired networks
- Other differences from wired networks
  - Neighbor motion, environment change
  - Sorts of interference (incl. self)
  - Asymmetric links often exist
  - Lower-layer protocol (retransmission, reliability, etc.)

What makes routing interesting?

- Lower node capacity
  - Computation, communication, power
  - Light-weight protocols – especially in sense of energy consumption since batteries don't follow the Moore's Law!
- Limited broadcast
- Higher loss rate
- Link unreliable or congested?
- More frequent topology changes
  - Mobility, time-varying link conditions, node power saving strategies
- Less secured physical media

Forms of MANETs

- Autonomous
  - Stand-alone multi-hop mobile wireless network
- Infrastructure extension
  - Mesh networks
  - Some nodes connected to infrastructure directly and relay messages for other nodes further away
- Wireless grid
  - Infrastructure formed by wireless connections, and mobile devices may communicate to infrastructure directly or via multi-hop

How Hot is Hot?

- Hottest in MANET research
  - About 20 RFC's
    - MANET routing protocol performance (RFC 2501)
    - AODV (RFC 3561)
    - OLSR (RFC 3626)
    - TBRPF (RFC 3684)
    - DSR (RFC 4728)
    - Jitter in MANETs (RFC 5148)
    - General MANET msg format (RFC 5444)
    - Representing multi-value time (RFC 5497)
    - IANA allocation for MANET protocols (RFC 5498)
    - …
  - Many Internet Drafts
  - Hundreds of protocols proposed!

Goal of Routing

- Finding route(s)
- Minimal overhead
  - Communication
  - Processing
- Maintenance
- Loop prevention
- Others

Taxonomy of MANET Routing

- Routing info
  - Topology-based routing
    - Distance vector
    - Link state
  - Position-based routing
- Info exchange
  - Proactive
  - Reactive
  - Hybrid
- Architecture
  - Flat
  - Hierarchical

Proactive Approaches

- Each node maintains a route to every other node in the network at all times
  - AKA "table driven"
- Creation and maintenance via periodic or event-triggered updates
  - Link addition/deletion or directionality change
- Pros and cons
  - Responsive
  - Significant communication overhead
  - $O(n)$ information stored in each node

DSDV

Destination-Sequenced Distance Vector

Perkins and Bhagwat SigComm'94

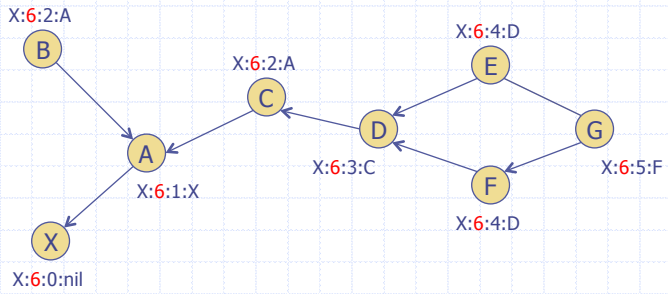
- Distance vector
  - Count-to-infinity problem
- Destination sequence number
  - A counter assigned by each destination
  - Increases every period
  - Used as **primary** distance update criterion for "guaranteed freshness" of route to that destination node
- Full and incremental updates

DSDV Explained

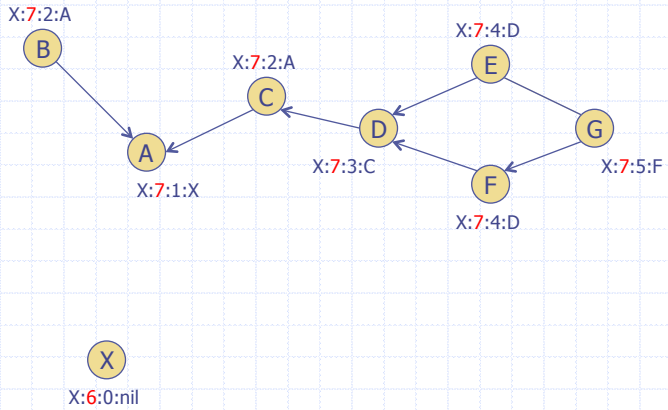
- Routing table entry

  `<dest_addr, dest_seq, dest_dist, next_hop>`
- Originators generate *even* sequence numbers
  - When a link to a destination is lost, the node detecting such a loss uses an odd sequence number (i.e. +=1) as a signal
- Route update at node *v*
  - When node *v* receives route update from neighbor *u* regarding destination *d*
  - If *d*'s sequence number as of *u* is even and greater than as of *v*
    - *v* updates its routing table entry regarding *d* to *u*'s sequence number of d and *u*'s distance to *d* added by 1
  - if same sequence numbers of *d*, update according to *u* if closer – regular distance vector algorithm
- Loop free – an odd sequence number indicates that the corresponding destination node is unreachable

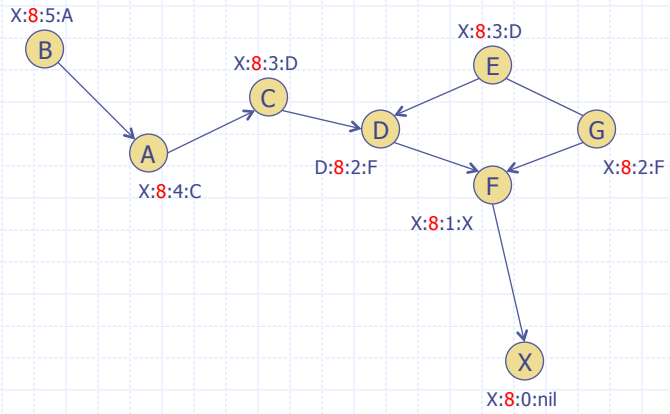
DSDV Example



DSDV Example



DSDV Example



OLSR

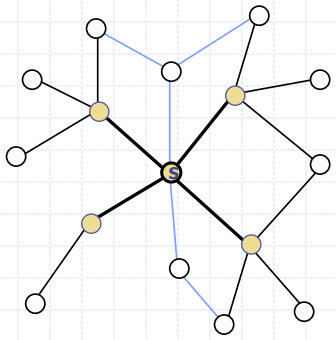
Optimized Link State Routing

Clausen, Jacquet, Laouiti, Muhlethaler, Qayyum and Viennot - RFC 3626

- Link state routing
- Using multi-point relays (MPRs) to reduce overhead of network floods and size of update packets
  - The MPRs of each node is selected such that when it broadcasts a message, retransmitting the message by its MPRs will cover the 2-hop neighborhood
  - Only MPRs broadcast LS info
  - Only links from a node to its MPRs are advertised in LS dissemination

OLSR Explained

- MPR of node *s*
  - Subset of $N_1(s)$ that dominates $N_2(s)$
- Node *s* determines its MPRs in greedy process
  - Minimization of the number of MPR for given *s* is NP-complete



FSR

Fisheye State Routing

Pei, Gerla and Chen 2000

- Link-state
- LS updated with different frequency depending on distance
  - For node *v*, the link state of (*a*,*b*) is broadcast to *v*'s neighbors at a frequency monotonically decreasing of the distance from *v*
  - A way to reduce control overhead
  - Nodes have more accurate information of close links while having summarized information of distant links

Reactive Approaches

- Routes are only discovered when they are actually needed
  - AKA "on demand"
  - Often involves flooding route requests throughout the network
  - Reduction in overhead needed
- Pros and cons
  - Saves network bandwidth in route maintenance
  - Route acquisition latency

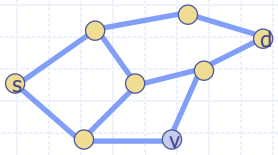
AODV Ad-hoc On-demand Distance Vector

Perkins and Royer – RFC 3561

- Distance Vector
  - Sets routing info at each node en route
  - Per-node sequence number (as in DSDV)
- Route request (RREQ)
- Route reply (RREP)
- Route error report (RERR)

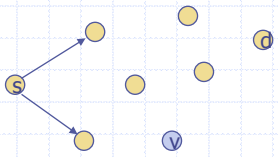
AODV Route Request (RREQ)

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
- When a node receives a RREQ for the first time, it
  - creates a reverse route to *s*
  - checks to see if it has a valid (unexpired) route to *d*
    - If not, rebroadcasts the RREQ
    - If so and the entry's sequence number of *d* is no less than dest_seq, it "replies" the request



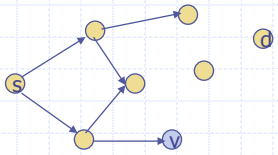
AODV Route Request (RREQ)

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
- When a node receives a RREQ for the first time, it
  - creates a reverse route to *s*
  - checks to see if it has a valid (unexpired) route to *d*
    - If not, rebroadcasts the RREQ
    - If so and the entry's sequence number of *d* is no less than dest_seq, it "replies" the request



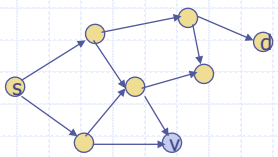
AODV Route Request (RREQ)

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
- When a node receives a RREQ for the first time, it
  - creates a reverse route to *s*
  - checks to see if it has a valid (unexpired) route to *d*
    - If not, rebroadcasts the RREQ
    - If so and the entry's sequence number of *d* is no less than dest_seq, it "replies" the request



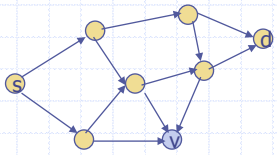
AODV Route Request (RREQ)

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
- When a node receives a RREQ for the first time, it
  - creates a reverse route to *s*
  - checks to see if it has a valid (unexpired) route to *d*
    - If not, rebroadcasts the RREQ
    - If so and the entry's sequence number of *d* is no less than dest_seq, it "replies" the request



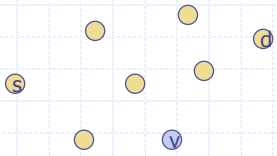
AODV Route Request (RREQ)

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
- When a node receives a RREQ for the first time, it
  - creates a reverse route to *s*
  - checks to see if it has a valid (unexpired) route to *d*
    - If not, rebroadcasts the RREQ
    - If so and the entry's sequence number of *d* is no less than dest_seq, it "replies" the request



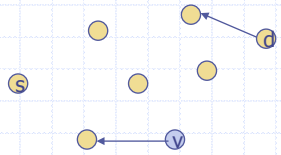
AODV Route Reply (RREP)

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the reverse path
- A RREP includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
    - hop_cnt is now from *d*
    - dest_seq is as of *d* / *v*
- When a node receives a RREP, it
  - creates a forward route to *d*
    - Updates the forward route if forward route exists
  - increments hop_cnt by 1
  - sends it to the next node along the reverse path



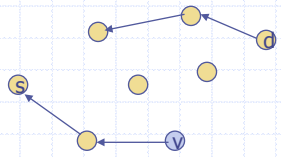
AODV Route Reply (RREP)

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the reverse path
- A RREP includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
    - hop_cnt is now from *d*
    - dest_seq is as of *d* / *v*
- When a node receives a RREP, it
  - creates a forward route to *d*
    - Updates the forward route if forward route exists
  - increments hop_cnt by 1
  - sends it to the next node along the reverse path



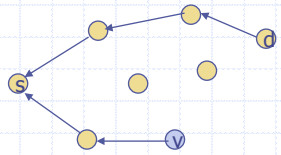
AODV Route Reply (RREP)

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the reverse path
- A RREP includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
    - hop_cnt is now from *d*
    - dest_seq is as of *d* / *v*
- When a node receives a RREP, it
  - creates a forward route to *d*
    - Updates the forward route if forward route exists
  - increments hop_cnt by 1
  - sends it to the next node along the reverse path



AODV Route Reply (RREP)

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the reverse path
- A RREP includes
  - <dest_addr, dest_seq, src_addr, src_seq, hop_cnt, rreq_id>
    - hop_cnt is now from *d*
    - dest_seq is as of *d* / *v*
- When a node receives a RREP, it
  - creates a forward route to *d*
    - Updates the forward route if forward route exists
  - increments hop_cnt by 1
  - sends it to the next node along the reverse path



AODV Route Error (RERR)

- When a node *u* determines that a neighbor is unreachable and it is on an active route to a destination, it creates a RERR containing a list of such destinations
- Then it sends RERR to upstream neighbors that uses *u* as the next hop to these destinations
- RERR propagates to all the relevant sources
- Route repair
  - Node *u* may choose to originate a RREQ to search for *d*
  - If not successful, sends RERR back up

Other Improvements of AODV

- Expanding ring search
  - Use TTL to control the scope of RREQ flooding
  - Starts with small TTL and increases if routes to destination *d* cannot be found
- Immunity to uni-directional links (UDLs)
  - Why are UDL detrimental?
  - RREP-ACK added
- Gratuitous route to source *s*

DSR

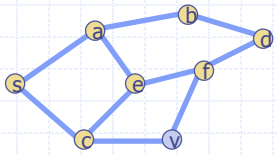
Dynamic Source Routing

Johnson and Maltz – RFC 4728

- Source routing
  - Source *s* has and appends to each data packet the complete route to destination *d*
  - RREQ and RREP packets accumulate full route during propagation
- *s* can maintain multiple routes to *d* as backup

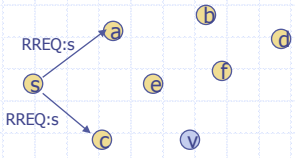
DSR RREQ

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes full path that it has traversed
- When an intermediate node receives a RREQ for the first time, it appends its address and rebroadcasts RREQ
- Until RREQ reaches *d* or some node *v* that has a valid route to *d*



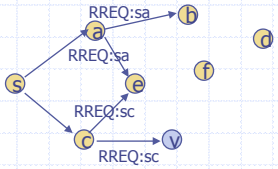
DSR RREQ

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes full path that it has traversed
- When an intermediate node receives a RREQ for the first time, it appends its address and rebroadcasts RREQ
- Until RREQ reaches *d* or some node *v* that has a valid route to *d*



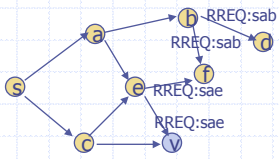
DSR RREQ

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes full path that it has traversed
- When an intermediate node receives a RREQ for the first time, it appends its address and rebroadcasts RREQ
- Until RREQ reaches *d* or some node *v* that has a valid route to *d*



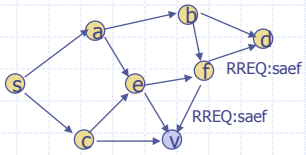
DSR RREQ

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes full path that it has traversed
- When an intermediate node receives a RREQ for the first time, it appends its address and rebroadcasts RREQ
- Until RREQ reaches *d* or some node *v* that has a valid route to *d*



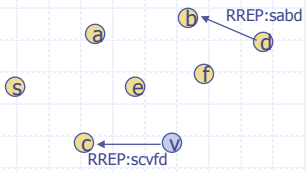
DSR RREQ

- A source node *s* floods a RREQ when it does not know how to route to a destination *d*
- A RREQ includes full path that it has traversed
- When an intermediate node receives a RREQ for the first time, it appends its address and rebroadcasts RREQ
- Until RREQ reaches *d* or some node *v* that has a valid route to *d*



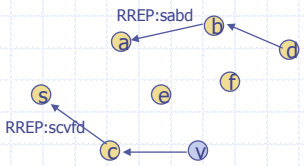
DSR RREP

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the accumulated path
- A RREP includes full path that RREQ has traversed
- Nodes on the path forward all RREPs, until RREP reaches *s*
  - *s* can have multiple routes to *d*



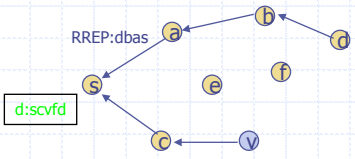
DSR RREP

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the accumulated path
- A RREP includes inversed full path that RREQ has traversed
- Nodes on the path forward all RREPs, until RREP reaches *s*
  - *s* can have multiple routes to *d*



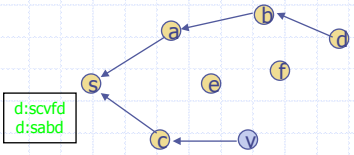
DSR RREP

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the accumulated path
- A RREP includes inversed full path that RREQ has traversed
- Nodes on the path forward all RREPs, until RREP reaches *s*
  - *s* can have multiple routes to *d*



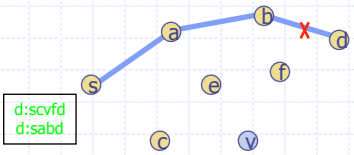
DSR RREP

- When a RREQ reaches *d* or a node *v* that knows a valid route to *d*, a RREP is unicast to *s* along the accumulated path
- A RREP includes inversed full path that RREQ has traversed
- Nodes on the path forward all RREPs, until RREP reaches *s*
  - *s* can have multiple routes to *d*



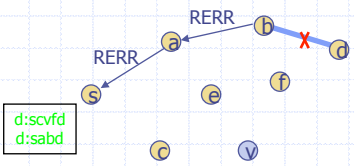
DSR RERR

- When a node determines that the next hop of a packet is not reachable, it unicasts a RERR to *s*
- Then *s* shall choose another route, if available, or shall issue a RREQ



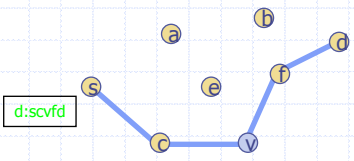
DSR RERR

- When a node determines that the next hop of a packet is not reachable, it unicasts a RERR to *s*
- Then s shall choose another route, if available, or shall issue a RREQ



DSR RERR

- When a node determines that the next hop of a packet is not reachable, it unicasts a RERR to *s*
- Then s shall choose another route, if available, or shall issue a RREQ



Hybrid Approaches

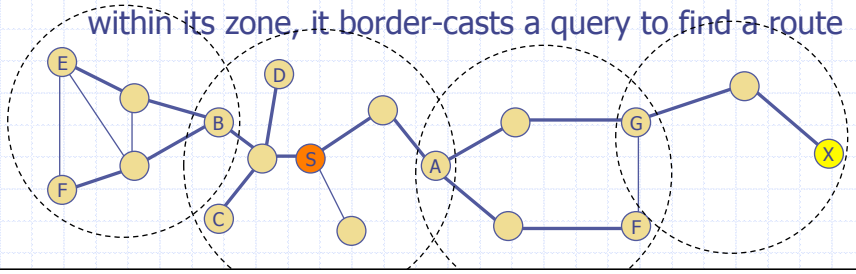
- Combining reactivity and proactivity
  - To take advantages of both
  - Various dimensions of combination

ZRP

Zone Routing Protocol

Pearlman and Haas 1999

- Each node has a zone of fixed radius
  - Within zone: link state routing
  - Inter zone: on demand
- Border-cast
  - When a source determines that destination is not within its zone, it border-casts a query to find a route



Position-Based Approaches

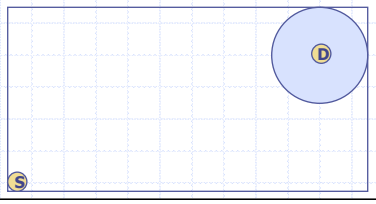
- Using position info to aid routing
  - Absolution locations, e.g. from GPS
  - Relative locations, e.g. using smart antennas
- Either reactive or proactive
- Examples
  - Direction-restricted flooding
  - Position-based unicasting

LAR

Location Aided Routing

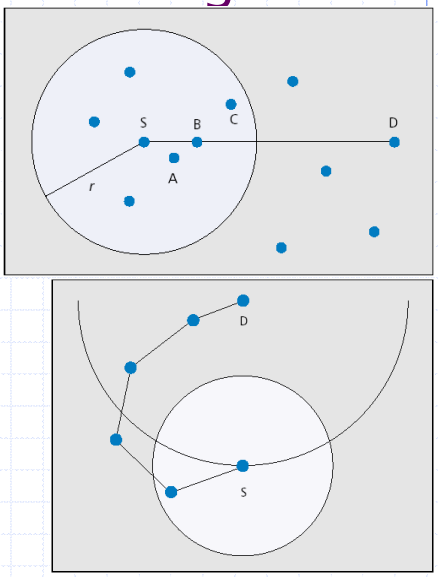
Ko and Vaidya 1998

- On demand routing with restricted route search (flooding) area
- Expected zone – area that destination *d* is most likely to be discovered
  - Circle calculated based on previously known location and velocity of *d*
- Request zone – area that RREQ is forwarded
  - Smallest rectangle containing both source *s* and destination *d*'s expected zone



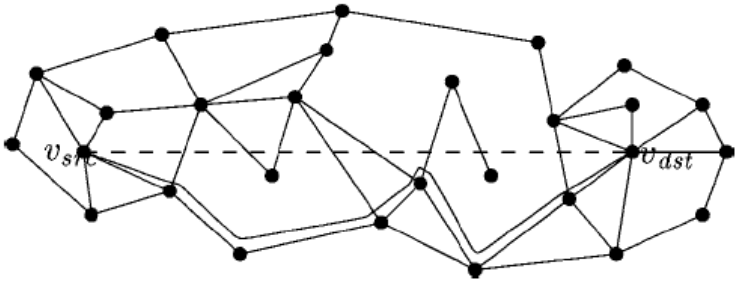
Position-Based Unicasting

- Positions of destination and neighbors known
- Hop-by-hop forwarding based on destination position
- Choose a neighbor that is closer to destination
  - Problem?



Face Routing

- Face-2 [Bose, Morin, Stojmenovic & Urrutia 99]
- GPSR [Karp & Tung 00]
  - Greedy Perimeter Stateless Routing
- Both done on planar graphs, e.g. RNG & GG



Hierarchical Approaches

- Nodes organized as substructures
  - One-level vs. multi-level
- Reduced overhead
  - Control traffic in network
  - Routing information storage space
- Important approach to scaling
- Organization may have extra costs

LANMAR

Landmark Ad Hoc Routing

Pei, Gerla and Hong 2000

- Nodes divided into subnets
  - Subnet members move as group
    - Have common interests
    - Have a leader called landmark
- Hierarchical routing
  - Within subnet: FSR among all nodes of the subnet
  - Inter subnets: DV among landmarks
- When destination within subnet, use FSR; otherwise, use DV to route packet to a node in destination's subnet

Multi-Path Approaches

- Multiple paths maintained at sources or intermediate nodes
  - To enhance reliability
- More used in reactive routing protocols
  - Why?
- Shared link/node vs. link/node-disjoint

Energy-Conserving Approaches

- Factor in energy concerns when evaluating routes
  - Distance is not the only thing that matters in MANETs
- Various metrics
  - Minimum total energy route
  - Minimum max energy route
  - Maximum remaining energy route

Security-Aware Approaches

- Route hacks
  - Lie about routes
  - Modify routing control packets, e.g. RREQ, RREP
- Cures
  - Authenticate all intermediate nodes
  - Share key between source and destinations
  - More …

Your inputs

- Any other issues related to routing in MANETs?
- Your solutions to them?