Routing in Ad Hoc Wireless Networks

PROF. MICHAEL TSAI
2011/12/15
Ad Hoc Wireless Networks

- No base station or access point to relay the packets
- Relaying is necessary to send information to destinations out of our range
- Initial application: military usage
Why do we need new protocols?

• No centralized control
• No dedicated routers
• Unpredictable network topology changes
• Time-variant wireless channel
  • Link breakage is common in wireless network → Connectivity problem
  • Links are not always bidirectional and/or symmetric
• Power Limitation
Conventional Routing Protocols

- Not designed for highly dynamic and low bandwidth networks

- Loop formation when topology changes

- Flooding causes high control overhead (e.g., Link State)
Count-to-infinity Problem

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
</tbody>
</table>
Count-to-infinity Problem

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Infinity</td>
<td>Null</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
</tbody>
</table>
Count-to-infinity Problem
This continues until the cost reaches infinity (unreachable). During the process, the packets destined for A will bounce back and forth between B and C.
Existing Routing Protocols

**Table-Driven:**
- S and all other nodes maintain full routing information
- Require periodic table update

**Demand-Driven**
- Route is discovered when S wants to talk to D
- A Route only needs to be maintained for as long as S and D are still talking
- EX: Dynamic Source Routing (DSR)

**Hybrid Scheme**
- Network is divided into multiple zones
- Use Table-Driven within the zone
- Demand-Driven across the zones through boundary nodes
## Proactive vs. Reactive Routing

<table>
<thead>
<tr>
<th><strong>Proactive</strong></th>
<th><strong>Reactive</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Table driven</td>
<td>• On demand</td>
</tr>
<tr>
<td>• Rely on periodic update to keep track of the topology change</td>
<td>• Route Discovery by local flood or gossiping</td>
</tr>
<tr>
<td>• No latency in route discovery</td>
<td>• Additional latency during route discovery</td>
</tr>
<tr>
<td>• Need large storage space to keep information of the entire network</td>
<td>• Not appropriate for real-time communication</td>
</tr>
<tr>
<td>• A lot of routing information may never be used</td>
<td>• Route maintenance</td>
</tr>
<tr>
<td></td>
<td>• Feedback from Link Level ACK</td>
</tr>
<tr>
<td></td>
<td>• Issue new route discovery when link breaks</td>
</tr>
</tbody>
</table>
Destination Sequenced Distance Vector (DSDV)  
Proactive Routing Protocols

- Each node advertises a monotonically increasing sequence number
- Each Route entry is tagged with a sequence number generated by destination to prevent loops (*count-to-infinity* problem)
- Sequence number indicates the “freshness” of a route
  - Routes with more recent sequence numbers are preferred for packet forwarding
  - If same sequence number, one having smallest metric is used

Example: DSDV

- For each reachable node in the network the routing entry contains:
  - Destination Address
  - Next Hop
  - Distance (Metric)
  - Sequence Number

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Distance</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>0</td>
<td>S205_A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>1</td>
<td>S334_B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>1</td>
<td>S198_C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>1</td>
<td>S567_D</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>2</td>
<td>S767_E</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>2</td>
<td>S45_F</td>
</tr>
</tbody>
</table>
Count-to-infinity Problem

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
<th>Seq. #</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
<th>Seq. #</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>
Count-to-infinity Problem

Sequence Number:
- Even numbers for link updates from neighbor nodes
- Odd numbers for link updates from the destination itself

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
<th>Seq. #</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Infinity</td>
<td>Null</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
<th>Seq. #</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>
**Count-to-infinity Problem**

Routing update

C’s routing update will not change B’s routing table since the sequence number is smaller (older).
DSDV: Topology changes

- **Assign a metric of $\infty$ to**
  - A broken link
  - Any route through a hop with a broken link

- **$\infty$ routes** are assigned new sequence numbers by any host and immediately broadcast via a triggered update

- If a node has an equal/later sequence number with a finite metric for an **$\infty$ route**, a route update is triggered
Dynamic Source Routing [DSR]
Route Discovery

- **Source node**
  - Broadcasts the Route Request (RREQ) <id, target>

- **Intermediate node**
  - Discards if the id has been seen before, or node is in the route record (header of RREQ)
  - Else append address in the route record and rebroadcast

- **Destination Node**
  - Return Route Reply (RREP)
    - Use previously cached route to source node
    - Call Route Discovery for source node, with route reply piggy backed
    - Use reverse sequence of Route Record, in case of bidirectional links

DSR: Route Maintenance

• **Monitoring the route**
  - Passive Acknowledgement – overhearing the next-hop node sending packet to its next-hop
  - Set a bit in packet to request explicit next hop acknowledgement

• **Route Error**
  - Rely on data link layer to report the broken links;
  - Notify source of the broken link via Route Error (RERR)
  - Source truncates all routes which use nodes mentioned in RERR
  - Initiate new route discovery

F transmits the packet I just sent to her. That means she received my packet correctly.
**DSR: Optimizations for efficiency**

- Use cached entries to create RREP at intermediate node
- Promiscuous mode to add more routes
  - Caching overheard RREQ/RREP
- Use hop based delays to prevent RREP storms
  - A lot of neighbors know the route to target and attempt to send RREP in response to RREQ
  - Delay RREP for a period \( d = H \times (h - 1 + r) \)
    - \( r \): random number between 0 and 1
    - \( H \): small constant delay
    - \( h \): number of hops to source from that node

---

**Diagram:**

- Node A directly connected to node B
- Node B connected to node C
- Node C connected to node D
- Node D has messages from B-C-D
- Node I is 4-hop away from D
- Node I heard A said B-C-D
- Node I heard B said C-D
- Node E, G, H are neighbors of D

**Questions:**

- Where is D?
  - I'm 2-hop away from D
  - I'm 4-hop away from D
Expanding Ring Search

- Route Request Hop Limit
- Use TTL in the packet header to specify the first ring boundary
- RREQ is initially forwarded $n$ times ($n$ hops)
- If destination is not within $n$-hop
  - Increase TTL to a larger value

This is useful if destination is close to the source
Gossiping vs. Flooding

Gossip: Probabilistic Flooding

- **Gossip-Based Routing**
  - Node forward packets with some probability $p_G < 1$

- **How good is it?**
  - 35% less overhead than flooding

- **What determine $P_G$?**

Network Connectivity

Connectivity: Fraction of nodes that is connected to the network

- **Sub-Critical**
  - Low connectivity
  - Mobile nodes are sparsely distributed in the network
  - Performance is limited!!

- **Super-Critical**
  - High connectivity region
  - Most or all the nodes can communicate
Ad-Hoc On Demand Distance Vector Routing (AODV)

- **Protocol overview - Pure on-demand protocol**
  - Node does not maintain knowledge of another node unless it communicates with it
  - Routes discovered on as-needed basis and maintained only as long as necessary
  - Little or no periodic advertisement

AODV – Route Discovery

• **Initiation**
  
  • Source node sends a Route Request (RREQ) when it has no information about destination node in its table
  
  • RREQ contains
    • Source and destination’s address and sequence number
    • Broadcast id
    • Hop count
  
  • Source address and broadcast id uniquely identify RREQ

• **Reverse Path Setup**
  
  • Neighbor increments hop count and broadcasts to neighbors
  
  • Records address of neighbor which first sends the RREQ
AODV – Route Discovery

• **Forward Path Setup**
  
  • Intermediate node satisfies RREQ if
    
    • Destination itself
    • Has route entry in table with destination sequence number ≥ that given in RREQ
  
  • Unicasts RREP to neighbor which sent RREQ
    
    • Source address
    • Destination address and sequence number (updated)
    • Hop count
    • Lifetime
  
  • As RREP travels backwards, each node sets pointer to sending node and updates destination sequence number and timeout entry for source and destination routes
AODV – Route Discovery

• Other nodes
  • RREQ times out : Route Request Expiration Timer
  • Deletes corresponding pointers

• More than one RREP received
  • One with greater destination number
  • Lesser hop count

• Source node starts transmission - updates if a better RREP is received

AODV – Route Maintenance

• Nodes send *hello message* if it has not sent a packet in *hello_interval*
• Failing to receive *allowed_hello_loss* packets consecutively means link is broken
• In case of broken link
  • unsolicited RERR sent to affected source node
  • Source initiates new RREQ
  • Sequence number updated
  • Hop count set = ∞
• Route Caching Timeout after the route is considered invalid
• **Optional** AODV-LL uses link layer ACK instead of *hello messages*
Link Quality Metrics

- The protocol chooses the route with the smallest hop count → Long hops will be included
- Long hops usually have lower SNR → high PER → retransmission!
- Original thought: lower hop count = lower bandwidth usage
- New thought: retransmission means wasted bandwidth
Link Quality Metrics

- Instead of using hop count only, we need to take “link quality” into account!
- What is a good metric for link quality?
  - RSSI (representing SNR)
  - ETX (Expected Transmission Count)
- Then we combine hop count + link quality to choose an optimal route
Example: ETX

Minimize total transmissions per packet (ETX, Expected Transmission Count)

Link throughput ≈ \( \frac{1}{\text{Link ETX}} \)

<table>
<thead>
<tr>
<th>Delivery Ratio</th>
<th>Link ETX</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>50%</td>
<td>2</td>
<td>50%</td>
</tr>
<tr>
<td>33%</td>
<td>3</td>
<td>33%</td>
</tr>
</tbody>
</table>
Measuring delivery ratios

- Each node broadcasts small link probes (134 bytes), once per second
- Nodes remember probes received over past 10 seconds
- Reverse delivery ratios estimated as
  \[ r_{rev} \approx \frac{\text{pkts received}}{\text{pkts sent}} \]
- Forward delivery ratios obtained from neighbors (piggybacked on probes)
Route ETX

Route ETX = Sum of link ETXs

<table>
<thead>
<tr>
<th>Route ETX</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>50%</td>
</tr>
<tr>
<td>2</td>
<td>50%</td>
</tr>
<tr>
<td>3</td>
<td>33%</td>
</tr>
<tr>
<td>5</td>
<td>20%</td>
</tr>
</tbody>
</table>
Example: SNR-based Metrics

- For each link (each of a node’s neighbor), maintain a “expected” SNR value.
  - This is to eliminate the small fading effects in SNR
- Each time receiving a packet, calculate:
  \[
  SSNR_i = \alpha \times SNR + (1 - \alpha) \times SSNR_{i-1}
  \]
  - SNR: SNR value of a newly received packet
  - \(SSNR_{i-1}\): old SSNR value before receiving the packet
  - \(SSNR_i\): new SSNR value
  - \(\alpha\): a sensitivity parameter. \(0 < \alpha \leq 1\).
- Then set a threshold to classify the links into good and bad links
- New route discovery process:
  - First try to discover routes consists of only good links
  - If no route can be found, then relax the condition to include bad links as well.
References for link quality aware routing metrics


