Ad-Hoc On-Demand Distance Vector Routing

Charles E. Perkins

Technology Development Group

Sun Microsystems

Menlo Park, CA

cperkins@eng.sun.com

Pravin Bhagwat

IBM T.J. Watson Research Center

Hawthorne, NY

pravin@watson.ibm.com

Steve Fulmer

University of Alabama

sfulmer@cs.ua.edu

Routing Considerations for Ad-Hoc Networking

Each ad-hoc node is supposed to be a router

Routing protocols solve a lot of problems

- multi-hop paths
- self-starting
- dynamic topology maintenance
- use existing technology

Distance Vector algorithms

- can be made loop free
- use less memory
- localized update operations
- simple to program

More Routing Considerations

Need for demand-driven route establishment

- Aggregation much harder to establish, maintain (WAMIS)
- Need scalability for large node populations
  - Route table size vs. memory and search time
  - Maintenance vs. bandwidth
  - Overhead on data traffic?

RREQ, RREP

RREQ (*route request*) is broadcast

- Reverse path is set up along the way
- RREQ message contains $< bcast\_id, src\_seqno, src\_ip,$ $src\_hops, dest\_seqno, dest\_ip >$

RREP (*route reply*) is unicast back

- From destination if necessary
- From intermediate node if that node has a recent route

Link Breakage

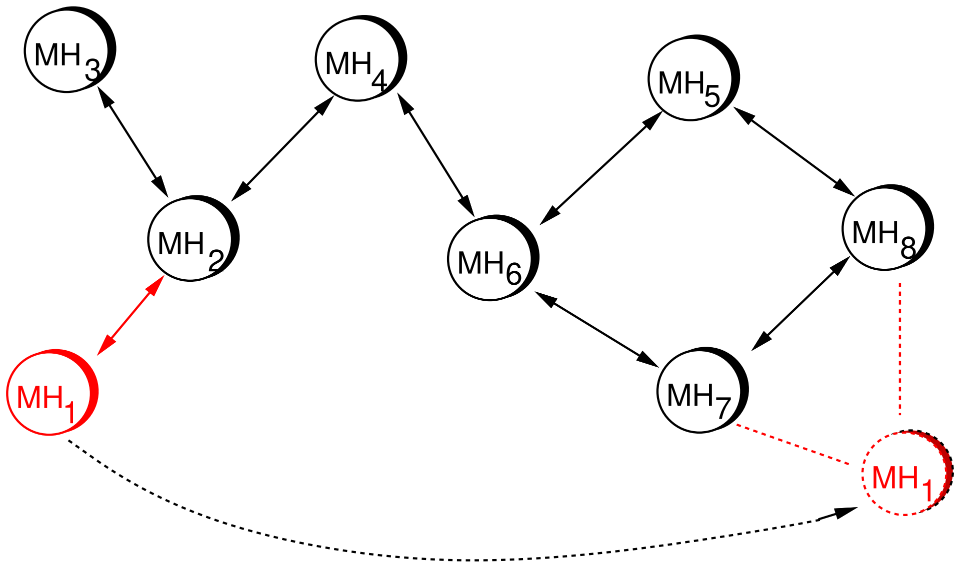
- Nodes remember active routers
- Next hop breaks $\rightarrow$ neighbors using that route are notified
- Notification is a RREP with:
  - metric = ∞
  - dest_seqno = previous + 1

  and is unicast address for each active neighbor

AODV properties

- Quick loop-free convergence
- Distance Vector, using Destination Sequence numbers for route updates (on both forward and reverse paths)
- No *split-horizon/poisoned-reverse*
- Triggered updates and minimal latency for route replies
- Reduced bandwidth utilization
- Limited range broadcast *"hello"* used for neighbor discovery, and to ensure symmetric routes. Needed: determination of whether asymmetric routes are too weak to be useful/usable.
- Enables future aggregation computations

Ad-hoc Networking



Suppose MH_1 moves away from MH_2 towards MH_7, and has active sessions with MH_3 and MH_6. The following actions occur:

- MH_2 notices that its link to MH_1 is broken
- MH_2 checks its routing table, and finds that its link to MH_1 was actively in use by MH_3 and MH_4.
- MH_2 unicasts an ∞-metric route update, with an incremented destination sequence number, to MH_3 and MH_4. MH_3 may subsequently issue a new route request for MH_1.
- MH_4 also notes that its route to MH_1 was actively in use, and forwards the ∞-metric route update to MH_6.

- The ∞-metric route update for MH_1 may also be included in the next *hello* message issued by MH_2
- MH_6 may subsequently issue a new route request for MH_1
- Any subsequent route request for MH_1 which is satisfied by a route reply through MH_2 may cause MH_2 to update its route table

Destination sequencing maintains nice properties of loop-freeness, and eliminates Bellman-Ford "counting to infinity" problem.

Currently, next hop is selected on "number of hops", but should perhaps consider instead "first response" to route request

Simulation Results

Using Maisie.

- Need to insure saturation to compare algorithms
- Apparently most sensitive to *mobility factor* vs. timeouts
- Have results so for up to 100 nodes.

Measurements:

- Average convergence time
- Total bandwidth utilization
- Total errors due to drops
- *goodput*
- parameter dependence on number of nodes, speed
- Comparisons to DSR (*Dynamic Source Routing*)

What does Location Independence mean for ad-hoc networking?

Are *functions* independent of location? hostname?

Where do services reside?

- DNS?
- Obtaining Certificates?
- Foreign Agents for mobile IP?
- Directory Agents for Service Location?

One idea: a *grid* of well-known services, with new candidate servers dynamically appearing in order to preserve a minimum number of hops to the closest service (of, e.g., the abovementioned types)