Ad-Hoc On-Demand Distance Vector Routing

Charles E. Perkins

Advanced Network Development

Sun Microsystems

Menlo Park, CA

cperkins@eng.sun.com

Elizabeth Royer

Dept of Electrical & Computer Engineering

University of California

Santa Barbara, University of California

eroyer@alpha.ece.uscb.edu

What an ad-hoc routing protocol needs

- Multi-hop paths
- Self-starting
- Dynamic topology maintenance
- Loop-free
- Low consumption of memory, bandwidth
- Scalable to large node populations
- Localized effect of link breakage
- Minimal overhead for data transmission
- Rapid convergence
- and ... Multicast

AODV: Ad-Hoc On-Demand Distance Vector Routing

- Quick loop-free convergence
- Route creation *on demand*, localizing the effect of topology changes, and minimizing control traffic.
- Distance Vector, using Destination Sequence numbers for route updates (on both forward and reverse paths)
- Triggered updates and minimal latency for route replies
- Reduced bandwidth utilization
- two-dim'l metric: <seq#, hop-count>
- Enables future aggregation computations

AODV Unicast Route Discovery

RREQ (*route request*) is broadcast

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

RREP (*route reply*) is unicast back

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

Route Request (RREQ) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |           Reserved            |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Broadcast ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Destination Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Source IP Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Source Sequence Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source sequence number helps set up *short-lived* reverse route

Destination sequence # is the *last known* for the requested destination (or, zero)

Hop Count incremented by every intermediate node

Route Reply (RREP) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |L|              Reserved             | Hop Count |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Lifetime                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Lifetime controls how long the remains valid at an intermediate node after it is no longer *active*

Hop Count incremented by every intermediate node

If broadcast with TTL=1, serves as a *hello* message

Link Breakage

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

  and is sent to each active neighbor

AODV Multicast Route Discovery

Message types:

- RREQ, with new flags 'J' (*Join*) and 'R' (*Repair*)
- RREP
- MINV

Multicast routes have a destination sequence number and multiple next hops

- Multicast Group Leader extension for RREQ, RREP

Multicast Invalidate Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |          Reserved             |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Destination Sequence Number                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Sequence Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A prospective multicast group member only needs one link to the multicast distribution tree, but it may receive multiple RREP messages.

The MINV message prunes all the extra branches from the multicast tree. Intermediate nodes that are not part of the multicast group, that receive MINV on their only outgoing link, prune themselves from the tree.

Tree Maintenance

- Multicast group leader maintains group sequence number
- Pruning (if node leaves tree)
  - leaf nodes send MINV
  - intermediate nodes remain

Multicast group leader broadcasts GROUP_HELLO periodically, containing the multicast group address, sequence number, and group leader's address.

Link Breakage

Nodes remember multicast tree branches

Node further from multicast group leader initiates link repair

Only nodes which are closer to group leader can send RREP

Node initiating repair selects new branch, sends MINV

No response after RREQ_RETRIES means tree is partitioned

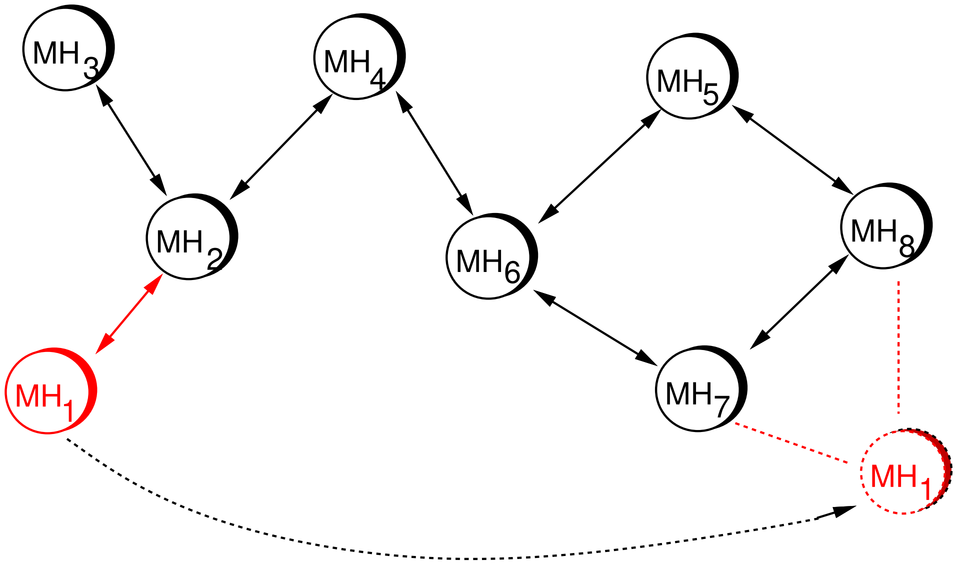
Initiating node becomes new group leader, issues Group Hello

Merging Disconnected Trees

If a node hears Group Hello from two group leaders, it can repair the multicast tree

- Sends RREQ with 'R' (*repair*) flag set to group leader of its partition
- Only group leader can respond to RREQ with R flag set
- Group leader sends RREQ with 'J' (*Join*) flag set to other group leader
- Other group leader sends RREP to node
- Group Leader with smaller IP address becomes new group leader
- New Group Leader broadcasts Group Hello with new group leader information

Ad-hoc Networking Example



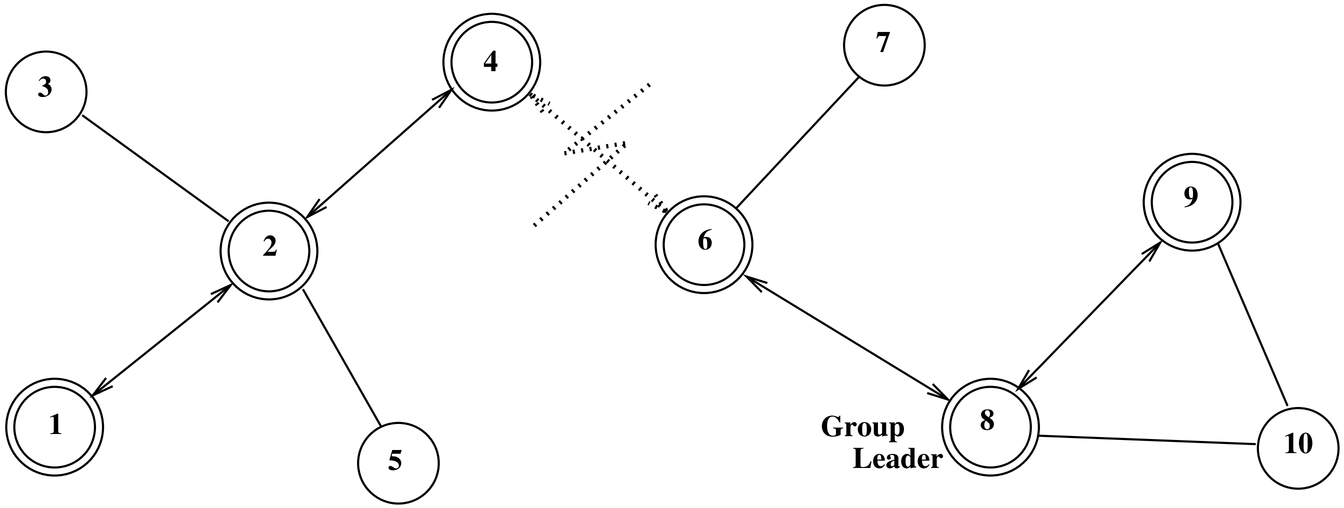
Suppose MH_1 moves away from MH_2 towards MH_7, and has active sessions with MH_3 and MH_6. The following actions occur:

- 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.

Repairing Multicast Tree Breaks



Node 4 detects link breakage, initiates group repair

Only nodes in subtree containing group leader can issue RREP

After RREQ_RETRIES, node 4 broadcasts GROUP_HELLO message as a new leader