Mobile Ad Hoc Networking Working Group   Charles E. Perkins
INTERNET DRAFT   Nokia Research Center
14 July 2000   Elizabeth M. Royer
                                   University of California, Santa Barbara
                                                       Samir R. Das
                                               University of Cincinnati

               Ad hoc On-Demand Distance Vector (AODV) Routing
                        draft-ietf-manet-aodv-06.txt

Status of This Memo

   This document is a submission by the Mobile Ad Hoc Networking Working
   Group of the Internet Engineering Task Force (IETF). Comments should
   be submitted to the manet@itd.nrl.navy.mil mailing list.

   Distribution of this memo is unlimited.

   This document is an Internet-Draft and is in full conformance with
   all provisions of Section 10 of RFC2026. Internet-Drafts are working
   documents of the Internet Engineering Task Force (IETF), its areas,
   and its working groups. Note that other groups may also distribute
   working documents as Internet-Drafts.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at
   any time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   The list of current Internet-Drafts can be accessed at:
        http://www.ietf.org/ietf/1id-abstracts.txt
   The list of Internet-Draft Shadow Directories can be accessed at:
        http://www.ietf.org/shadow.html.

Abstract

   The Ad Hoc On-Demand Distance Vector (AODV) routing protocol is
   intended for use by mobile nodes in an ad hoc network. It offers
   quick adaptation to dynamic link conditions, low processing and
   memory overhead, low network utilization, and determines unicast
   between sources and destinations. It uses destination sequence
   numbers to ensure loop freedom at all times (even in the face of
   anomalous delivery of routing control messages), solving problems
   (such as ``counting to infinity'') associated with classical distance
   vector protocols.

                              Contents

1. Introduction

   The Ad Hoc On-Demand Distance Vector (AODV) algorithm enables
   dynamic, self-starting, multihop routing between participating mobile
   nodes wishing to establish and maintain an ad hoc network. AODV
   allows mobile nodes to obtain routes quickly for new destinations,
   and does not require nodes to maintain routes to destinations that
   are not in active communication. AODV allows mobile nodes to respond
   quickly to link breakages and changes in network topology. The
   operation of AODV is loop-free, and by avoiding the Bellman-Ford
   ``counting to infinity'' problem offers quick convergence when the
   ad hoc network topology changes (typically, when a node moves in the
   network). When links break, AODV causes the affected set of nodes to
   be notified so that they are able to invalidate the routes using the
   broken link.

   One distinguishing feature of AODV is its use of a destination
   sequence number for each route entry. The destination sequence
   number is created by the destination for any route information it
   sends to requesting nodes. Using destination sequence numbers
   ensures loop freedom and is simple to program. Given the choice
   between two routes to a destination, a requesting node always selects
   the one with the greatest sequence number.

2. Overview

   Route Requests (RREQs), Route Replies (RREPs), and Route Errors
   (RERRs) are the message types defined by AODV. These message types
   are handled by UDP, and normal IP header processing applies. So, for
   instance, the requesting node is expected to use its IP address as
   the source IP address for the messages. The range of dissemination
   of broadcast RREQs can be indicated by the TTL in the IP header.
   Fragmentation is typically not required.

   As long as the endpoints of a communication connection have valid
   routes to each other, AODV does not play any role. When a route to
   a new destination is needed, the node uses a broadcast RREQ to find
   a route to the destination. A route can be determined when the RREQ
   reaches either the destination itself, or an intermediate node with
   a 'fresh enough' route to the destination. A 'fresh enough' route
   is an unexpired route entry for the destination whose associated

sequence number is at least as great as that contained in the RREQ.
The route is made available by unicasting a RREP back to the source
of the RREQ. Since each node receiving the request caches a route
back to the source of the request, the RREP can be unicast back from
the destination to the source, or from any intermediate node that
is able to satisfy the request back to the source. A RREQ can be
conditioned by requirements on the path to the destination, namely
bandwidth or delay bounds.

Nodes monitor the link status of next hops in active routes. When a
link break in an active route is detected, a RERR message is used to
notify other nodes that the loss of that link has occurred. The RERR
message indicates which destinations are now unreachable due to the
loss of the link.

AODV is a routing protocol, and it deals with route table management.
Route table information must be kept even for ephemeral routes, such
as are created to temporarily store reverse paths towards nodes
originating RREQs. AODV uses the following fields with each route
table entry:

  - Destination IP Address

  - Destination Sequence Number
  - Hop Count (number of hops needed to reach destination)

  - Last Hop Count (described in subsection 8.2.1)
  - Next Hop

  - List of Precursors (described in Section 8.1)
  - Lifetime (expiration or deletion time of the route)

  - Routing Flags

3. AODV Terminology

   This protocol specification uses conventional meanings [1] for
   capitalized words such as MUST, SHOULD, etc., to indicate requirement
   levels for various protocol features. This section defines other
   terminology used with AODV that is not already defined in [3].

      active route

         A routing table entry with a finite metric in the Hop Count
         field. A routing table may contain entries that are not active
         (invalid routes or entries). They have an inifnite metric
         in the Hop Count field. Only active entries can be used to
         forward data packets. Invalid entries are eventually deleted.

   forwarding node

      A node which agrees to forward packets destined for another
      destination node, by retransmitting them to a next hop which is
      closer to the unicast destination along a path which has been
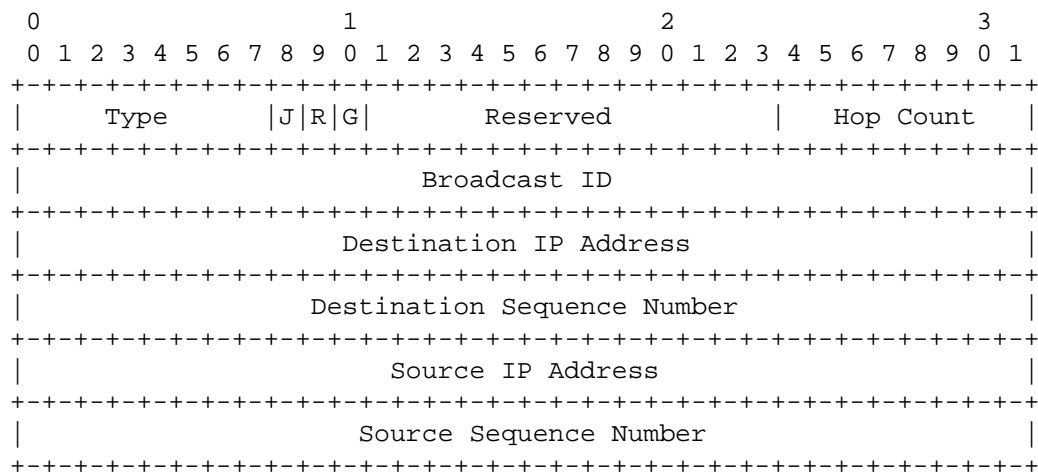   set up using routing control messages.

   forward route

      A route set up to send data packets from a source to a
      destination.

   reverse route

      A route set up to forward a reply (RREP) packet back to the
      source from the destination or from an intermediate node having
      a route to the destination.

4. 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      |J|R|G|       Reserved        |   Hop Count   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Broadcast ID                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Destination IP Address                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Destination Sequence Number                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Source IP Address                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Source Sequence Number                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

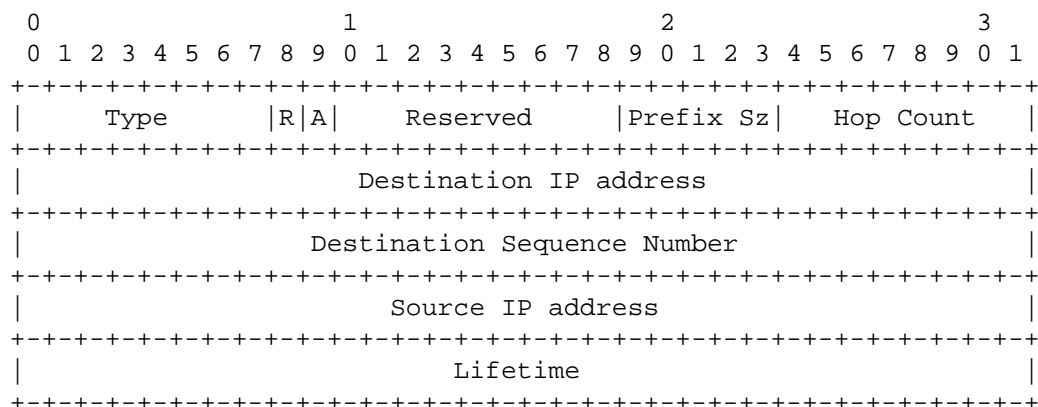   The format of the Route Request message is illustrated above, and
   contains the following fields:

      Type   1

      J   Join flag; reserved for multicast.

      R   Repair flag; reserved for multicast.

      G   Gratuitous RREP flag; indicates whether a
                    gratuitous RREP should be unicast to the node

specified in the Destination IP Address field (see
sections 8.2, 8.5)

   Reserved   Sent as 0; ignored on reception.

   Hop Count   The number of hops from the Source IP Address to
                  the node handling the request.

   Broadcast ID   A sequence number uniquely identifying the
                  particular RREQ when taken in conjunction with the
                  source node's IP address.

   Destination IP Address
                  The IP address of destination for which a route is
                  desired.

   Destination Sequence Number
                  The last sequence number received in the past by
                  the source for any route towards the destination.

   Source IP Address
                  The IP address of the node which originated the
                  Route Request.

   Source Sequence Number
                  The current sequence number to be used for route
                  entries pointing to (and generated by) the source
                  of the route request.

5. 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      |R|A|   Reserved    |Prefix Sz|   Hop Count     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Lifetime                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
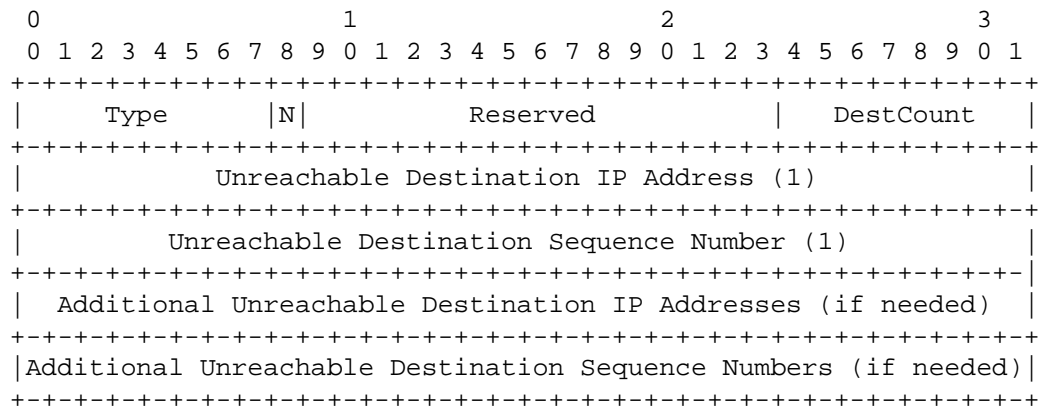
The format of the Route Reply message is illustrated above, and
contains the following fields:

Type   2

R   Repair flag; used for multicast.

A   Acknowledgment required; see sections 7 and 8.6.

Reserved   Sent as 0; ignored on reception.

Prefix Size   If nonzero, the 5-bit Prefix Size specifies that the
                indicated next hop may be used for any nodes with
                the same routing prefix (as defined by the Prefix
                Size) as the requested destination.

Hop Count   The number of hops from the Source IP Address to
                the Destination IP Address. For multicast route
                requests this indicates the number of hops to the
                multicast tree member sending the RREP.

Destination IP Address
                The IP address of the destination for which a route
                is supplied.

Destination Sequence Number
                The destination sequence number associated to the
                route.

Source IP Address
                The IP address of the source node which issued the
                RREQ for which the route is supplied.

Lifetime   The time for which nodes receiving the RREP consider
                the route to be valid.

  Note that the Prefix Size allows a Subnet Leader to supply a route
  for every host in the subnet defined by the routing prefix, which
  is determined by the IP address of the Subnet Leader and the Prefix
  Size. In order to make use of this feature, the Subnet Leader has to
  guarantee reachability to all the hosts sharing the indicated subnet
  prefix. The Subnet Leader is also responsible for maintaining the
  Destination Sequence Number for the whole subnet.

6. Route Error (RERR) 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      |N|          Reserved           |   DestCount   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Unreachable Destination IP Address (1)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Unreachable Destination Sequence Number (1)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|  Additional Unreachable Destination IP Addresses (if needed) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Additional Unreachable Destination Sequence Numbers (if needed)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Route Error message is illustrated above, and
contains the following fields:

    Type   3

    N   No delete flag; set when a node has performed a local
                repair of a link, and upstream nodes should not delete
                the route.

    Reserved   Sent as 0; ignored on reception.

    DestCount   The number of unreachable destinations included in the
                message; MUST be at least 1.

    Unreachable Destination IP Address
                The IP address of the destination which has become
                unreachable due to a link break.

    Unreachable Destination Sequence Number
                The last known sequence number, incremented by one,
                of the destination listed in the previous Unreachable
                Destination IP Address field.

  The RERR message is sent whenever a link break causes one or more
  destinations to become unreachable. The unreachable destination
  addresses included are those of all lost destinations which are now
  unreachable due to the loss of that link.

7. Route Reply Acknowledgement (RREP-ACK) Message Format

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

    Type   4

    Reserved   Sent as 0; ignored on reception.

   The RREP-ACK message is used to acknowledge receipt of an RREP
   message. It is used in cases where the link over which the RREP
   message is sent may be unreliable.

8. AODV Operation

   This section describes the scenarios under which nodes generate
   RREQs, RREPs and RERRs for unicast communication, and how the message
   data are handled.

8.1. Maintaining Route Utilization Records

   For each valid route maintained by a node (containing a finite Hop
   Count metric) as a routing table entry, the node also maintains a
   list of precursors that may be forwarding packets on this route.
   These precursors will receive notifications from the node in the
   event of detection of the loss of the next hop link. The list of
   precursors in a routing table entry contains those neighboring nodes
   to which a route reply was generated or forwarded.

   Each time a route is used to forward a data packet, its Lifetime
   field is updated to be current time plus ACTIVE_ROUTE_TIMEOUT.

8.2. Generating Route Requests

   A node broadcasts a RREQ when it determines that it needs a route
   to a destination and does not have one available. This can happen
   if the destination is previously unknown to the node, or if a
   previously valid route to the destination expires or is broken
   (i.e., an infinite metric is associated with the route). The
   Destination Sequence Number field in the RREQ message is the last
   known destination sequence number for this destination and is copied
   from the Destination Sequence Number field in the routing table. If

no sequence number is known, a sequence number of zero is used. The
Source Sequence Number in the RREQ message is the node's own sequence
number. The Broadcast ID field is incremented by one from the last
broadcast ID used by the current node for the same destination. The
Hop Count field is set to zero.

A source node often expects to have bidirectional communications with
a destination node. In such cases, it is not sufficient for the
source node to have a route to the destination node; the destination
must also have a route back to the source node. In order to cause
this to happen as efficiently as possible, any generation of an RREP
by an intermediate node (as in section 8.4) for delivery to the
source node, should be accompanied by some action which notifies the
destination about a route back to the source node. The source node
selects this mode of operation in the intermediate nodes by setting
the 'G' flag. See section 8.5 for details about actions taken by the
intermediate node in response to a RREQ with the 'G' flag set.

After broadcasting a RREQ, a node waits for a RREP. If the RREP is
not received within NET_TRAVERSAL_TIME milliseconds, the node MAY
rebroadcast the RREQ, up to a maximum of RREQ_RETRIES times. Each
rebroadcast MUST increment the Broadcast ID field.

Data packets waiting for a route (i.e., waiting for a RREP after RREQ
has been sent) SHOULD be buffered. The buffering SHOULD be FIFO. If
a RREQ has been rebroadcast RREQ_RETRIES times without receiving any
RREP, all data packets destined for the corresponding destination
SHOULD be dropped from the buffer and a Destination Unreachable
message delivered to the application.

8.2.1. Controlling Route Request broadcasts

To prevent unnecessary network-wide broadcasts of RREQs, the
source node SHOULD use an expanding ring search technique as an
optimization. In an expanding ring search, the source node initially
uses a TTL = TTL_START in the RREQ packet IP header and sets the
timeout for receiving a RREP to 2 * TTL * NODE_TRAVERSAL_TIME
milliseconds. Upon timeout, the source rebroadcasts the RREQ with
the TTL incremented by TTL_INCREMENT. This continues until the
TTL set in the RREQ reaches TTL_THRESHOLD, beyond which a TTL =
NET_DIAMETER is used for each rebroadcast. Each time, the timeout
for receiving a RREP is calculated as before. Each rebroadcast
increments the Broadcast ID field in the RREQ packet. The RREQ
can be rebroadcast with TTL = NET_DIAMETER up to a maximum of
RREQ_RETRIES times.

When a RREP is received, the Hop Count used in the RREP packet is
remembered as Last Hop Count in the routing table. When a new route

to the same destination is required at a later time (e.g., upon route
loss), the TTL in the RREQ IP header is initially set to this Last
Hop Count plus TTL_INCREMENT. Thereafter, following each timeout the
TTL is incremented by TTL_INCREMENT until TTL = TTL_THRESHOLD is
reached. Beyond this TTL = NET_DIAMETER is used as before.

As a further optimization, timeouts MAY be determined dynamically via
measurements, instead of using a statically configured value related
to NODE_TRAVERSAL_TIME. To accomplish this, the RREQ may carry the
timestamp via an extension field as defined in Section 11 to be
carried back by the RREP packet (again via an extension field). The
difference between the current time and this timestamp will determine
the route discovery latency. The timeout may be set to be a small
factor times the average of the last few route discovery latencies
for the concerned destination. These latencies may be recorded as
additional fields in the routing table.

If the optimizations described in this section are used, an expired
routing table entry SHOULD NOT be expunged before DELETE_PERIOD.
Otherwise, the soft state corresponding to the route (e.g., Last Hop
Count) will be lost. In such cases, a longer routing table entry
expunge time may be specified. Any routing table entry waiting for a
RREP should not be expunged before RREP_WAIT_TIME.

8.3. Forwarding Route Requests

When a node receives a broadcast RREQ, it first checks to determine
whether it has received a RREQ with the same Source IP Address
and Broadcast ID within at least the last BROADCAST_RECORD_TIME
milliseconds. If such a RREQ has been received, the node silently
discards the newly received RREQ. The rest of this subsection
describes actions taken for RREQs that are not discarded.

8.3.1. Processing Route Requests

When a node receives a RREQ, the node checks to determine whether it
has an active route to the destination. If the node does not have
an active route, it rebroadcasts the RREQ from its interface(s) but
using its own IP address in the IP header of the outgoing RREQ. The
Destination Sequence Number in the RREQ is updated to the maximum
of the existing Destination Sequence Number in the RREQ and the
destination sequence number in the routing table (if an entry exists)
of the current node. The TTL or hop limit field in the outgoing IP
header is decreased by one. The Hop Count field in the broadcast
RREQ message is incremented by one, to account for the new hop
through the intermediate node.

If the node, on the other hand, does has an active route for the
destination, it compares the destination sequence number for that
route with the Destination Sequence Number field of the incoming
RREQ. If the existing destination sequence number is smaller than
the Destination Sequence Number field of the RREQ, the node again
rebroadcasts the RREQ just as if it did not have an active route to
the destination.

The node generates a RREP (as discussed further in section 8.4) if
either:

    (i)   it has an active route to the destination, and the
                node's existing destination sequence number is greater
                than or equal to the Destination Sequence Number of the
                RREQ, or

    (ii)   it is itself the destination.

The node always creates or updates a reverse route to the Source IP
Address in its routing table. If a route to the Source IP Address
already exists, it is updated only if either

    (i)   the Source Sequence Number in the RREQ is higher than
                the destination sequence number of the Source IP Address
                in the route table, or

    (ii)   the sequence numbers are equal, but the hop count as
                specified by the RREQ is now smaller than the existing
                hop count in the routing table.

When a reverse route is created or updated, the following actions are
carried out:

 1. the Source Sequence Number from the RREQ is copied to the
    corresponding destination sequence number;

 2. the next hop in the routing table becomes the node broadcasting
    the RREQ (it is obtained from the source IP address in the IP
    header and is often not equal to the Source IP Address field in
    the RREQ message);

 3. the hop count is copied from the Hop Count in the RREQ message;

 4. the lifetime of the route is the higher of its current lifetime
    (for an active route) and current time plus REV_ROUTE_LIFE.

Even if the route is not updated because the existing route has a
higher destination sequence number, but if it is scheduled to expire

before REV_ROUTE_LIFE, its lifetime is still updated to be current
time plus REV_ROUTE_LIFE.

This reverse route would be needed in case the node receives an
eventual RREP back to the node which originated the RREQ (identified
by the Source IP Address).

8.4. Generating Route Replies

If a node receives a route request for a destination, and either
has a fresh enough route to satisfy the request or is itself the
destination, the node generates a RREP message and unicasts it back
to the node indicated by the Source IP Address field of the received
RREQ. The node generating the RREP message copies the Source and
Destination IP Addresses in RREQ message into the corresponding
fields in the RREP message which is to be sent back toward the
source of the RREQ. Additional operations are slightly different,
depending on whether the node is itself the requested destination, or
instead if it is an intermediate node with an admissible route to the
destination.

As the RREP is forwarded to the source, the Hop Count field is
incremented by one at each hop. Thus, when the RREP reaches the
source, the Hop Count represents the distance, in hops, of the
destination from the source.

8.4.1. Route Reply Generation by the Destination

If the generating node is the destination itself, it uses a
destination sequence number at least equal to a sequence number
generated after the last detected change in its neighbor set and at
least equal to the destination sequence number in the RREQ. If the
destination node has not detected any change in its set of neighbors
since it last incremented its destination sequence number, it MAY use
the same destination sequence number. The destination node places
the value zero in the Hop Count field of the RREP.

The destination node copies the value MY_ROUTE_TIMEOUT into
the Lifetime field of the RREP. Each node MAY make a separate
determination about its value MY_ROUTE_TIMEOUT.

8.4.2. Route Reply Generation by an Intermediate Node

If node generating the RREP is not the destination node, but
instead is an intermediate hop along the path from the source to the

destination, it copies the last known destination sequence number in
the Destination Sequence Number field in the RREP message.

The intermediate node places its distance in hops from the
destination (indicated by the hop count in the routing table) in the
Hop Count field in the RREP.

When the intermediate node updates its route table for the source
of the RREQ, it puts the last hop node (from which it received the
RREQ, as indicated by the source IP address field in the IP header)
into the precursor list for the forward path route entry -- i.e., the
entry for the Destination IP Address. Furthermore, the intermediate
node puts the next hop towards the destination in the precursor list
for the reverse route entry -- i.e., the entry for the Source IP
Address field of the RREQ message data.

The intermediate node calculates the Lifetime field of the RREP by
subtracting the current time from the expiration time in its route
table entry.

8.5. Generating Gratuitous RREPs

When a node receives a RREQ and responds with a RREP, it does not
forward the RREQ any further. If all incarnations of a single
RREQ are replied to by intermediate nodes, the destination does
not receive any copies of the RREQ. Hence, it does not learn of a
route to the source node. This can be problematic if the source is
attempting to establish a TCP session. In order that the destination
learn of routes to the source node, the source node SHOULD set the
gratuitous RREP ('G') flag in the RREQ if the session is going to be
run over TCP, or if the destination should receive the gratuitous
RREP for any other reason. Intermediate nodes receiving a RREQ
with the 'G' flag set and responding with a RREP SHOULD unicast a
gratuitous RREP to the destination node.

The RREP that is sent to the source of the RREQ is the same as
before. The gratuitous RREP that is to be sent to the desired
destination contains the following values in the RREP message fields:

   Hop Count The Hop Count as received in the RREQ

   Destination IP Address
             The IP address of the node that generated the RREQ

   Destination Sequence Number
             The Source Sequence Number from the RREQ

        Source IP Address
                    The IP address of the destination node

        Lifetime   The remaining lifetime of the route towards the
                    destination node, as known by the intermediate node.

    The gratuitous RREP is then sent to the next hop along the path to
    the destination node.

8.6. Forwarding Route Replies

    When a node receives a RREP message, it first compares the
    Destination Sequence Number in the message with its own copy of
    destination sequence number for the Destination IP Address. The
    forward route for this destination is created or updated only if
    (i) the Destination Sequence Number in the RREP is greater than the
    node's copy of the destination sequence number, or (ii) the sequence
    numbers are the same, but the route is no longer active or the Hop
    Count in RREP is smaller than the hop count in route table entry. If
    a new route is created or the old route is updated, the next hop is
    the node from which the RREP is received, which is indicated by the
    source IP address field in the IP header; the hop count is the Hop
    Count in the RREP message plus one; the expiry time is the current
    time plus the Lifetime in the RREP message; the destination sequence
    number is the Destination Sequence Number in the RREP message.

    The current node can now begin using this route to send data packets
    to the destination.

    If the current node is not the source node as indicated by the Source
    IP Address in the RREP message AND a forward route has been created
    or updated as described before, the node consults its route table
    entry for the source node to determine the next hop for the RREP
    packet, and then forwards the RREP towards the source with its Hop
    Count incremented by one.

    When any node generates or forwards a RREP, the precursor list for
    the corresponding destination node is updated by adding to it the
    next hop node to which the RREP is forwarded. Also, at each node the
    (reverse) route used to forward a RREP has its lifetime changed to
    current time plus ACTIVE_ROUTE_TIMEOUT.

    If a node forwards an RREP over a link that is likely to have errors,
    the node MAY set the 'A' flag to require that the recipient of the
    RREP is to acknowledge receipt of the RREP by sending a RREP-ACK
    message back.

8.7. Hello Messages

   A node MAY offer connectivity information by broadcasting local
   Hello messages as follows. Every HELLO_INTERVAL milliseconds, the
   node checks whether it has sent a broadcast (e.g., a RREQ or an
   appropriate layer 2 message) within the last HELLO_INTERVAL. If it
   has not, it MAY generate a broadcast RREP with TTL = 1, called a
   Hello message, with the message fields set as follows:

      Destination IP Address
               The node's IP address.

      Destination Sequence Number
               The node's latest sequence number.

      Hop Count   0

      Lifetime   ALLOWED_HELLO_LOSS * HELLO_INTERVAL

   A node MAY determine connectivity by listening for packets from
   its set of neighbors. If it receives no packets for more than
   ALLOWED_HELLO_LOSS * HELLO_INTERVAL milliseconds, the node SHOULD
   assume that the link to this neighbor is currently broken. When this
   happens, the node SHOULD proceed as in Section 8.9.

8.8. Maintaining Local Connectivity

   Each forwarding node SHOULD keep track of its active next hops (i.e.,
   which next hops have been used to forward packets towards some
   destination within the last ACTIVE_ROUTE_TIMEOUT milliseconds). This
   is done by updating the Lifetime field of a routing table entry used
   to forward data packets to current time plus ACTIVE_ROUTE_TIMEOUT
   milliseconds. For purposes of efficiency, each node may try to learn
   which of these active next hops are really in the neighborhood at the
   current time using one or more of the available link or network layer
   mechanisms, as described below.

    - Any suitable link layer notification, such as those provided by
      IEEE 802.11, can be used to determine connectivity, each time
      a packet is transmitted to an active next hop. For example,
      absence of a link layer ACK or failure to get a CTS after sending
      RTS, even after the maximum number of retransmission attempts,
      will indicate loss of the link to this active next hop.

    - Passive acknowledgment can be used when the next hop is expected
      to forward the packet, by listening to the channel for a
      transmission attempt made by the next hop. If transmission is
      not detected within NEXT_HOP_WAIT milliseconds or the next hop is

not a forwarding node (and thus is never supposed to transmit the
packet) one of the following methods should be used to determine
connectivity.

   *   Receiving an ICMP ACK message from the next hop. The ICMP
       ACK message SHOULD be sent to a forwarding node by a next hop
       which is also the destination as in the in the IP header of
       the packet. This should be done only when this destination
       has not sent any packets to the concerned forwarding node
       within the last HELLO_INTERVAL milliseconds.

   *   A RREQ unicast to the next hop, asking for a route to the
       next hop.

   *   An ICMP Echo Request message unicast to the next hop.

If a link to the next hop cannot be detected by any of these methods,
the forwarding node SHOULD assume that the link is broken, and take
corrective action by following the methods specified in Section 8.9.

8.9. Route Error Messages

A node initiates a RERR message in three situations:

   (i)   if it detects a link break for the next hop of an active
              route in its routing table, or

   (ii)   if it gets a data packet destined to a node for which it
              does not have an active route, or

   (iii)   if it receives a RERR from a neighbor for one or more
              active routes.

For cases (i) and (ii), the destination sequence numbers in the
routing table for the unreachable destination(s) are incremented by
one. Then RERR is broadcast with the unreachable destination(s) and
their incremented destination sequence number(s) included in the
packet. For case (i), the unreachable destinations are the broken
next hop, and any additional destinations which are now unreachable
due to the loss of this next hop link. For case (ii), there is only
one unreachable destination, which is the destination of the data
packet that cannot be delivered. The DestCount field of the RERR
packet indicates the number of unreachable destinations included in
the packet.

For cases (i) and (ii), for each unreachable destination the node
copies the value in the Hop Count route table field into the Last

Hop Count field, and marks the Hop Count for this destination as
infinity, and thus invalidates the route.

For case (iii) when a node receives a RERR message, for each
unreachable destination included in the packet, the node determines
whether the source node (as indicated by the source IP address in the
IP header) forwarding the RERR packet is its own next hop used to
reach this destination. If so, the node takes the following actions:

   (a)   updates the corresponding destination sequence number
              with the Destination Sequence Number in the packet, and

   (b)   marks the Hop Count for this destination as infinity,
              and thus invalidates the route.

   (c)   checks the precursor list for this destination. If one
              or more of these precursor lists are non-empty, the node
              creates a RERR message, including as unreachable each
              destination with a non-empty precursor list. It also
              includes their destination sequence numbers, and then
              broadcasts this RERR message.

When a node receives a RERR message, it always updates its
destination sequence number(s) for the unreachable destination(s)
included in the packet using the corresponding sequence numbers
included in the message. When a node broadcasts a RERR message, it
always deletes the precursor list of each unreachable destination
included in the message.

When a node invalidates a route to a neighboring node, it must also
delete that neighbor from any precursor lists for routes to other
nodes. This prevents precursor lists from containing stale entries
of neighbors with which the node is no longer able to communicate.
The node should inspect the precursor list of each destination entry
in its routing table, and delete the lost neighbor from any list in
which it appears.

8.9.1. Local Repair

When a link break in an active route occurs, the node upstream of
that break MAY choose to repair the link locally if the destination
is no farther than MAX_REPAIR_TTL hops away. To repair the link
break itself, it increments the sequence number for the destination
and then broadcasts a RREQ for that destination. The TTL of the RREQ
should initially be set to the following value:
        max (MIN_REPAIR_TTL, 0.5 TTL to source) + LOCAL_ADD_TTL.
Thus, local repair attempts should never be visible to the source
node, and will always have minimum TTL equal to MIN_REPAIR_TTL

+ LOCAL_ADD_TTL. The node initiating the repair then waits the
discovery period to receive RREPs in response to the RREQ. If, at
the end of the discovery period, it has not received a RREP for that
destination, it proceeds as described in Section 8.9 by creating a
RERR message for that destination.

On the other hand, if the nodes does receive one or more RREPs during
the discovery period, the node proceeds as described in Section 8.6,
creating a route table entry for that destination. It then compares
the hop count of the new route with the value in the last hop count
route table entry for that destination. If the hop count of the
newly determined route to the destination is greater than the hop
count of the previously known route, as recorded in the last hop
count field, the node MAY create a RERR message for the destination
and send this message to the source node. The node sets the 'N' flag
of the RERR, and then broadcasts this message if it has one or more
precursor nodes for this route table entry.

A node which receives a RERR message with the 'N' flag set MUST
NOT delete the route to that destination. The only action taken
should be the retransmission of the message, if the RERR arrived
from the next hop along that route, and if there are one or more
precursor nodes for that route to the destination. When the source
node receives a RERR message with the 'N' flag set, if this message
came from its next hop along its route to the destination then the
source node MAY choose to reinitiate route discovery, as described in
Section 8.2.

Local repair of link breaks in active routes sometimes results in
increased path lengths to those destinations. Repairing the link
locally is likely to increase the number of data packets which are
able to be delivered to the destinations, since data packets will not
be dropped as the RERR travels to the source node. Sending a RERR
to the source node after locally repairing the link break allows the
source to find a fresh route to the destination which is more optimal
based on current node positions. However, it does not require the
source node to rebuild the route, as the source may be done, or
nearly done, with the data session.

When a link breaks along an active route, there are often multiple
destinations which become unreachable. The node which is upstream
of the broken link tries an immediate local repair for only the one
destination towards which the packet was traveling. Other routes
using the same link MUST be marked as broken, but the node handling
the local repair MAY flag each such newly broken route as locally
repairable; this local repair flag in the route table MUST be reset
when the route times out (i.e., after the route has been not been
active for ACTIVE_ROUTE_TIMEOUT). Before the timeout occurs, these
other routes will be repaired as needed when packets arrive for the

other destinations. Alternatively, depending upon local congestion,
the node MAY begin the process of establishing local repairs for the
other routes, without waiting for new packets to arrive.

8.10. Route Expiry and Deletion

   If the Lifetime of an active routing entry expires, the following
   actions are taken.

    1. The entry is invalidated by copying the Hop Count to the Last Hop
       Count field and then making the Hop Count infinity.

    2. The destination sequence number of this routing entry is
       incremented by one.

    3. The Lifetime field is updated to current time plus DELETE_PERIOD.
       Before this time, the entry MUST NOT be deleted.

   Note that the Lifetime field plays dual role -- for an active route
   it is the expiry time, and for an invalid route it is the deletion
   time.

   These actions are also taken whenever a route entry is invalidated
   for any reason, for example, for link breakage or receiving a RERR.

   If a data packet is received for an invalid route, the Lifetime
   field is always updated to current time plus DELETE_PERIOD. The
   determination of DELETE_PERIOD is discussed in Section 12

8.11. Actions After Reboot

   A node participating in the ad hoc network must take certain
   actions after reboot as it will have lost its prior sequence
   number and as well as its last known sequence numbers for various
   other destinations. However, there may be neighboring nodes which
   are using this node as an active next hop. This can potentially
   create routing loops. To prevent this possibility, each node on
   reboot waits for DELETE_PERIOD. In this time, it does not respond
   to any routing packets. However, if it receives a data packet,
   it broadcasts a RERR as described in subsection 8.9 and resets
   the waiting timer (Lifetime) to expire after current time plus
   DELETE_PERIOD.

   It can be shown that by the time the rebooted node comes out of
   the waiting phase and becomes an active router again, none of its
   neighbors will be using it as an active next hop any more. Its own
   sequence number gets updated once it receives a RREQ from any other

   node, as the RREQ always carries the maximum destination sequence
   number seen en route.

9. AODV and Aggregated Networks

   AODV has been designed for use by mobile nodes with IP addresses
   that are not necessarily related to each other, to create an ad hoc
   network. However, in some cases a collection of mobile nodes MAY
   operate in a fixed relationship to each other and share a common
   subnet prefix, moving together within an area where an ad hoc network
   has formed. Call such a collection of nodes a ``subnet''. In this
   case, it is possible for a single node within the subnet to advertise
   reachability for all other nodes on the subnet, by responding with
   a RREP message to any RREQ message requesting a route to any node
   with the subnet routing prefix. Call the single node the ``subnet
   router''. In order for a subnet router to operate the AODV protocol
   for the whole subnet, it has to maintain a destination sequence
   number for the entire subnet. In any such RREP message sent by the
   subnet router, the Prefix Size field of the RREP message MUST be
   set to the length of the subnet prefix. Other nodes sharing the
   subnet prefix SHOULD NOT issue RREP messages, and SHOULD forward RREQ
   messages to the subnet leader.

10. Using AODV with Other Networks

   In some configurations, an ad hoc network may be able to provide
   connectivity between external routing domains that do not use AODV.
   If the points of contact to the other networks can act as subnet
   routers (see Section 9) for any relevant networks within the external
   routing domains, then the ad hoc network can maintain connectivity to
   the external routing domains. Indeed, the external routing networks
   can use the ad hoc network defined by AODV as a transit network.

   In order to provide this feature, a point of contact to an external
   network (call it an Infrastructure Router) has to act as the subnet
   router for every subnet of interest within the external network for
   which the Infrastructure Router can provide reachability. This
   includes the need for maintaining a destination sequence number for
   that external subnet.

   If multiple Infrastructure Routers offer reachability to the same
   external subnet, those Infrastructure Routers have to cooperate (by
   means outside the scope of this specification) to provide consistent
   AODV semantics for ad hoc access to those subnets.

11. Extensions

   RREQ and RREP messages have extensions defined in the following
   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      |    Length     |    type-specific data ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   where:

     Type   1

     Length   The length of the type-specific data, not including the
               Type and Length fields of the extension.

   Extensions with types between 128 and 255 may NOT be skipped. The
   rules for extensions will be spelled out more fully, and conform with
   the rules for handling IPv6 options.

11.1. Hello Interval Extension 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      |    Length     |        Hello Interval ...     |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  | ... Hello Interval, continued |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

     Type   2

     Length   4

     Hello Interval
               The number of milliseconds between successive
               transmissions of a Hello message.

   The Hello Interval extension MAY be appended to a RREP message with
   TTL == 1, to be used by a neighboring receiver in determine how long
   to wait for subsequent such RREP messages (i.e., Hello messages; see
   section 8.7).

12. Configuration Parameters

   This section gives default values for some important values
   associated with AODV protocol operations. A particular mobile
   node may wish to change certain of the parameters, in particular
   the NET_DIAMETER, NODE_TRAVERSAL_TIME, MY_ROUTE_TIMEOUT,
   ALLOWED_HELLO_LOSS, RREQ_RETRIES, and possibly the HELLO_INTERVAL. In
   the latter case, the node should advertise the HELLO_INTERVAL in its
   Hello messages, by appending a Hello Interval Extension to the RREP
   message. Choice of these parameters may affect the performance of
   the protocol.

        Parameter Name   Value
        ----------------------   -----
        ACTIVE_ROUTE_TIMEOUT   3,000 Milliseconds
        ALLOWED_HELLO_LOSS   2
        BROADCAST_RECORD_TIME   2 * NET_TRAVERSAL_TIME
        DELETE_PERIOD   see note below
        HELLO_INTERVAL   1,000 Milliseconds
        LOCAL_ADD_TTL   2
        MAX_REPAIR_TTL   0.3 * NET_DIAMETER
        MY_ROUTE_TIMEOUT   2 * ACTIVE_ROUTE_TIMEOUT
        NET_DIAMETER   35
        NEXT_HOP_WAIT   NODE_TRAVERSAL_TIME + 10
        NODE_TRAVERSAL_TIME   40
        REV_ROUTE_LIFE   NET_TRAVERSAL_TIME
        NET_TRAVERSAL_TIME   3 * NODE_TRAVERSAL_TIME * NET_DIAMETER / 2
        RREQ_RETRIES   2
        TTL_START   1
        TTL_INCREMENT   2
        TTL_THRESHOLD   7

   DELETE_PERIOD should be an upper bound on the time for which
   an upstream node A can have a neighbor B to be an active next
   hop for destination D, while B has invalidated the route to D.
   Beyond this time B can delete the route to D. The determination
   of the upper bound somewhat depends on the characteristics of
   the underlying link layer. For example, if the link layer
   feedback is used to detect loss of link DELETE_PERIOD must be
   at least ACTIVE_ROUTE_TIMEOUT. If there is no feedback and hello
   messages must be used, DELETE_PERIOD must be at least maximum of
   ACTIVE_ROUTE_TIMEOUT and ALLOWED_HELLO_LOSS * HELLO_INTERVAL. If
   hello messages are received from a neighbor but data packets to that
   neighbor are lost, (due to temporary link asymmetry, e.g.) we have
   to make more concrete assumptions about the underlying link layer.
   We assume that such asymmetry cannot persist beyond a certain certain
   time, say, a multiple K of ALLOWED_HELLO_LOSS * HELLO_INTERVAL.

In other words, it cannot not be the case that a node receives K
subsequent hello messages from a neighbor, while that same neighbor
fails to receive any data packet from the node in this period. This
is a reasonable assumption as this AODV specification works only with
symmetric links. Covering all possibilities,

          DELETE_PERIOD = K * max (ACTIVE_ROUTE_TIMEOUT,
     ALLOWED_HELLO_LOSS * HELLO_INTERVAL) (K = 5 is recommended).

NET_DIAMETER measures the maximum possible number of hops between
two nodes in the network. NODE_TRAVERSAL_TIME is a conservative
estimate of the average one hop traversal time for packets and should
include queueing delays, interrupt processing times and transfer
times. ACTIVE_ROUTE_TIMEOUT SHOULD be set to a longer value (at
least 10,000 milliseconds) if link-layer indications are used to
detect link breakages such as in IEEE 802.11 [2] standard. TTL_START
should be set to at least 2 if Hello messages are used for local
connectivity information. Performance of the AODV protocol is
sensitive to the chosen values of these constants, which often depend
on the characteristics of the underlying link layer protocol, radio
technologies etc.

13. Security Considerations

Currently, AODV does not specify any special security measures.
Route protocols, however, are prime targets for impersonation
attacks, and must be protected by use of authentication techniques
involving generation of unforgeable and cryptographically strong
message digests or digital signatures. It is expected that, in
environments where security is an issue, that IPSec authentication
headers will be deployed along with the necessary key management to
distribute keys to the members of the ad hoc network using AODV.

14. Acknowledgements

We acknowledge with gratitude the work done at University of
Pennsylvania within Carl Gunter's group, as well as at Stanford and
CMU, to determine some conditions (especially involving reboots and
lost RERRs) under which previous versions of AODV could suffer from
routing loops. Contributors to those efforts include Karthikeyan
Bhargavan, Joshua Broch, Dave Maltz, Madanlal Musuvathi, and
Davor Obradovic. The idea of a DELETE_PERIOD, for which expired
routes (and, in particular, the sequence numbers) to a particular
destination must be maintained, was also suggested by them.

We also acknowledge the comments and improvements suggested by SJ Lee
(especially regarding local repair) and Mahesh Marina.

References

[1] S. Bradner. Key words for use in RFCs to Indicate Requirement
    Levels. Request for Comments (Best Current Practice) 2119,
    Internet Engineering Task Force, March 1997.

[2] IEEE 802.11 Committee, AlphaGraphics #35, 10201 N.35th Avenue,
    Phoenix AZ 85051. Wireless LAN Medium Access Control MAC and
    Physical Layer PHY Specifications, June 1997. IEEE Standard
    802.11-97.

[3] Charles E. Perkins. Terminology for Ad-Hoc Networking (work in
    progress). draft-ietf-manet-terms-00.txt, November 1997.

A. Draft Modifications

   The following are major changes between this version (06) of the AODV
   draft and the previous version (05):

   - Removed sections on Broadcast, Multicast, QoS, and
      Autoconfiguration into separate documents

   - Added Gratuitous RREP feature

   - Added optional ability for RREP sender to require acknowledgement
      (NOT YET!)

   - Added local repair (NOT YET!)

Author's Addresses

   Questions about this memo can be directed to:

      Charles E. Perkins
      Communications Systems Laboratory
      Nokia Research Center
      313 Fairchild Drive
      Mountain View, CA 94303
      USA
      +1 650 625 2986
      +1 650 691 2170 (fax)
      charliep@iprg.nokia.com

      Elizabeth M. Royer
      Dept. of Electrical and Computer Engineering
      University of California, Santa Barbara
      Santa Barbara, CA 93106
      +1 805 893 7788
      +1 805 893 3262 (fax)
      eroyer@alpha.ece.ucsb.edu

      Samir R. Das
      Department of Electrical and Computer Engineering
      & Computer Science
      University of Cincinnati
      Cincinnati, OH 45221-0030
      +1 513 556 2594
      +1 513 556 7326 (fax)
      sdas@ececs.uc.edu