**Readme**

This is an overview of this document.

To complete the message structure sections, I don’t fully understand how the AODVv2 information maps to the RFC5444 message. Section 10 and Section 16.2 and 16.3 don’t make it completely clear.

* Is the “Sequence Number List” the Address Block TLV, or is “SeqNum” the TLV, or are there separate TLVs for “OrigAddr SeqNum” and “TargAddr SeqNum”? Is there an “UnreachableAddr SeqNum” TLV too?
* Similarly for the “Metric List”

Most of the yellow highlights show small elements that need a clarification.

I have not mentioned limiting AODVv2 messages as mentioned in Section 13. A test for this could be included in each section before any further steps are taken to create/regenerate a message.

There are some other discussion points highlighted in yellow.

Validity time is mentioned in the draft in a couple of places but not explained.

The phrase “routable and unicast” is used a lot. To me this suggests that a network address should not be used, that it should always be a host address. I think what it means is that it should not be from the multicast IP address range.

**RREQ Message Structure:**

msg\_hop\_limit Part of RFC5444 Message Header

msg\_hop\_count Part of RFC5444 Message Header

MetricType (optional) RFC5444 Message TLV

AddressList and PrefixList RFC5444 Address Block

OrigAddr

TargAddr

OrigAddr PrefixLength Addresses combined into head, tail, mid, prefix length

TargAddr PrefixLength

SeqNum and Metric RFC5444 Address Block TLV Block

Sequence Number List RFC5444 Address Block TLV (Section 10)

OrigAddr SeqNum RFC5444 Address Block TLV (Appendix B1)

TargAddr SeqNum (optional) RFC5444 Address Block TLV (Appendix B1)

MetricList RFC5444 Address Block TLV (Section 10)

OrigAddr Metric RFC5444 Address Block TLV (Appendix B1)

TargAddr Metric (optional) RFC5444 Address Block TLV (Appendix B1)

Is the Sequence Number List the TLV? And its length can vary depending on if both OrigAddr SeqNum and TargAddr SeqNum are present?

Or are OrigAddrSeqNum and TargAddrSeqNum both TLVs which may or may not be included?

**RREQ Generation:**

Upon receiving an IP packet, when there is no Active or Idle route to its destination, an RREQ should be generated. However, before sending, we should check if we have recently already sent an RREQ for this destination and may be waiting for a response, or if we have reached our limit of AODVv2 messages and should wait to avoid transmitting too many AODVv2 messages to the network.

It makes sense that OrigAddr is the source IP Address of the IP Packet triggering the RREQ, and TargAddr is the destination IP Address of the IP Packet triggering the RREQ. If using host addresses we will send RREQs unnecessarily and end up with more routes than necessary. Subnet addresses could be used in order that we don’t end up with many host routes when one subnet route would be easier to maintain. Perhaps OrigAddr, since we should know what the subnet configuration is, could be replaced with the subnet address. For the destination, we will not know what subnet it is on so we should use the destination IP address for TargAddr rather than a subnet address.

In constructing the RREQ, AODVv2 uses Address List, Prefix List, Sequence Number List and Metric List. RFC5444 will take the Address List and Prefix List, combining into an Address Block for best compression. The Sequence Number List and Metric List are stored in new Address Block TLVs which will be placed in the Address Block TLV Block after the Address Block in the RFC5444 message.

The procedure for RREQ generation is as follows:

1. Set msg\_hop\_limit to MAX\_HOPCOUNT
2. Set msg\_hop\_count to zero, if including it. When would you not?
3. Include the MetricType TLV if requesting a route for a non-default metric type and set the correct value for Metric Type in the TLV
4. For the Address List, set OrigAddr and TargAddr
5. For the Prefix Length List
   1. If using a subnet address for the OrigAddr, the Prefix Length List should contain the OrigAddr’s prefix. TargAddr’s prefix should be omitted/set to null?
   2. If the OrigAddr is a host IP address, no Prefix Length List is needed
6. For the Sequence Number List
   1. Increment the SeqNum as described in Section 6.4 (Sequence Numbers)\*
   2. Set OrigSeqNum in the Sequence Number List to the new sequence number
   3. If a Broken or Expired route exists matching TargAddr using longest prefix matching, set TargSeqNum in the Sequence Number List to the sequence number stored on this route, else set it to null/omit it?
7. For the Metric List
   1. Set the OrigAddrMetric in the MetricList.
   2. TargAddrMetric is unknown and is omitted/set to null?

\*This may mean to increment the router’s SeqNum, the interface’s SeqNum, or the OrigAddr’s SeqNum, although 6.4 is not entirely clear on what IP address has an associated sequence number.

The RREQ is multicast to the LL-MANET-Routers IP and MAC address [RFC 5498, 4291?]. If multicast or broadcast is not possible, unicast is allowed as described in Section 5.

Should RREQ be stored at originator?

**RREQ Reception:**

Upon receiving an RREQ, the following steps are performed:

1. Check if the sender is on the blacklist of AODVv2 routers without bidirectional links
   1. If on the blacklist, check the Blacklist Remove Time
      1. If Current Time < Remove Time, ignore this message for further processing
      2. If not, remove the Blacklist entry and continue processing
   2. If not, continue processing
2. Check if the message is a duplicate or redundant by comparing to entries in the RREQ table as described in Section 8.6 (Suppressing Redundant RREQ Messages)
   1. If duplicate or redundant, ignore this message for further processing
   2. If not, save the information in the RREQ table to identify future duplicates, and continue
3. Check if the message contains the required data elements: msg\_hop\_limit, OrigAddr, TargAddr, OrigAddrMetric, OrigSeqNum, and that OrigAddr and TargAddr are valid addresses (routable and unicast)
   1. If not, ignore this message for further processing
   2. If so, continue processing
4. If the MetricType TLV is present, check that the MetricType is known.
   1. If not, ignore this message for processing
   2. If so, continue processing
5. Check that the OrigAddrMetric is not greater than MAX\_METRIC[MetricType] – Cost(Link)
   1. If so, ignore this message for processing
   2. If not, continue processing
6. Process the route to OrigAddr as specified in Section 7.1 (Evaluating Incoming Routing Information)
7. Check if the TargAddr belongs to one of the Router Clients
   1. If so, the router generates a RREP as specified in Section 8.4 (RREP Generation)
   2. If not, continue to RREQ regeneration

**RREQ Regeneration:**

Unless the router is prepared to advertise the new route, it halts processing. By sending a RREQ, a router advertises that it will forward packets to the OrigAddr contained in the RREQ according to the information enclosed. The router MAY choose not to regenerate the RREQ, though this could decrease connectivity in the network or result in non-optimal paths.

The circumstances under which a router MAY choose not to regenerate a RREQ are not specified in this document, though some examples may include the router being heavily loaded and wishes not to advertise routing for more traffic, or being low on energy and having to reduce energy expended for sending AODVv2 messages or packet forwarding.

The procedure for RREQ regeneration is as follows:

1. Check the msg\_hop\_limit
   1. If it is zero, do not regenerate
   2. If not, decrement the value by one
2. Check if msg\_hop\_count is present and greater than or equal to MAX\_HOPCOUNT
   1. If so, do not regenerate
   2. If not, increment it by one
3. Change OrigAddrMetric to match the route table entry for OrigAddr, which should match the advertised value in the received RREQ plus the cost of the link to the router which forwarded the RREQ

The RREQ SHOULD be multicast to the LL-MANET-Routers IP and MAC address [RFC 5498, 4291?]. If the received RREQ was unicast, the regenerated RREQ can be unicast to the next hop address of the route to TargAddr, if known; otherwise it should be multicast as per default behaviour.

**RREP Message Structure:**

msg\_hop\_limit Part of RFC5444 Message Header

msg\_hop\_count Part of RFC5444 Message Header

AckReq RFC5444 Message TLV

MetricType (optional) RFC5444 Message TLV

AddressList and PrefixList RFC5444 Address Block

OrigAddr

TargAddr

OrigAddr PrefixLength Addresses combined into head, tail, mid, prefix length

TargAddr PrefixLength

SeqNum and Metric RFC5444 Address Block TLV Block

Sequence Number List RFC5444 Address Block TLV (Section 10)

OrigAddr SeqNum RFC5444 Address Block TLV (Appendix B1)

TargAddr SeqNum RFC5444 Address Block TLV (Appendix B1)

MetricList RFC5444 Address Block TLV (Section 10)

OrigAddr Metric RFC5444 Address Block TLV (Appendix B1)

TargAddr Metric RFC5444 Address Block TLV (Appendix B1)

Is the Sequence Number List the TLV? And its length can vary depending on if both OrigAddr SeqNum and TargAddr SeqNum are present?

Or are OrigAddrSeqNum and TargAddrSeqNum both TLVs which may or may not be included?

**RREP Generation:**

In constructing the RREP, AODVv2 uses Address List, Prefix List, Sequence Number List and Metric List. RFC5444 will take the Address List and Prefix List, combining into an Address Block for best compression. The Sequence Number List and Metric List are stored in new Address Block TLVs which will be placed in the Address Block TLV Block after the Address Block in the RFC5444 message.

The TargAddr in the received RREQ will likely be a host address, whereas the AODVv2 router probably has a route to the entire subnet that TargAddr is part of. It could advertise the subnet and prefix length in the RREP to avoid other routers needing one route for each host device.

The procedure for RREP generation is as follows:

1. Set msg\_hop\_limit to the msg\_hop\_count from the received RREQ message
2. Set msg\_hop\_count, if including it, to zero. When would you not include it?
3. Include the AckReq TLV if an Acknowledgement is requested (as described in Section 14.5)
4. If MetricType is not DEFAULT\_METRIC\_TYPE, include the MetricType TLV and set the type accordingly
5. For the Address List, set OrigAddr and TargAddr
6. For the Prefix Length List
   1. If using a subnet address for the TargAddr, the Prefix Length List should contain the TargAddr’s prefix. OrigAddr’s prefix should be omitted/set to null? – what if OrigAddr was a subnet address with a prefix – should we include prefix info in the reply or is it unnecessary since the intermediate nodes already received the RREQ?
   2. If the TargAddr is a host IP address, no Prefix Length List is needed
7. For the Sequence Number List
   1. Do not set the OrigAddr SeqNum (omit/null?)
   2. Set the TargAddr SeqNum by incrementing the Sequence Number as described in Section 6.4 (Sequence Numbers)\*
8. For the Metric List
   1. Do not set the OrigAddrMetric (omit/null?)
   2. Set the TargAddrMetric to the metric stored with the route

The RREP is sent by unicast to the IP address of the next hop of the RREP\_Gen’s route to OrigAddr.

\*This may mean to increment the router’s SeqNum, the interface’s SeqNum, or the OrigAddr’s SeqNum, although 6.4 is not entirely clear on what IP address has an associated sequence number.

**RREP Reception:**

Upon receiving an RREP, the following steps are performed:

1. Check if the sender is on the blacklist of AODVv2 routers without bidirectional links
   1. If on the blacklist, check the Blacklist Remove Time
      1. If Current Time < Remove Time, ignore this message for further processing
      2. If not, remove the Blacklist entry and continue processing
   2. If not, continue processing
2. If the RREP was multicast, check if the message is a duplicate or redundant by comparing to entries in the RREP table as described in Section 14.4 (Multicast RREP Response to RREQ)
   1. If duplicate or redundant, ignore this message for further processing
   2. If not, save the information in the RREP table to identify future duplicates, and continue
3. Check if the message contains the required data elements: msg\_hop\_limit, OrigAddr, TargAddr, TargAddrMetric, TargSeqNum, and that OrigAddr and TargAddr are valid addresses (routable and unicast)
   1. If not, ignore this message for further processing
   2. If so, continue processing
4. If the MetricType TLV is present, check that the MetricType is known.
   1. If not, ignore this message for processing
   2. If so, continue processing
5. Check that the TargAddrMetric is not greater than MAX\_METRIC[MetricType] – Cost(Link)
   1. If so, ignore this message for processing
   2. If not, continue processing
6. Process the route to TargAddr as specified in Section 7.1 (Evaluating Incoming Routing Information)
7. Check if the AckReq TLV is present
   1. If so, send a RREP\_ACK as seen in Appendix B4.
8. Check if the OrigAddr belongs to one of the Router Clients
   1. If so, the RREP satisfies a previously sent RREQ. Processing is complete and data can now be forwarded along the route
   2. If not, continue to RREP regeneration

**RREP Regeneration:**

Similar to rules for RREQ regeneration, unless the router is prepared to advertise the new route, it halts processing. By forwarding a RREP, a router advertises that it will forward packets to the TargAddr contained in the RREP according to the information enclosed. The router MAY choose not to regenerate the RREP, for the same reasons as mentioned under RREQ regeneration, though this could decrease connectivity in the network or result in non-optimal paths.

The procedure for RREP regeneration is as follows:

1. Check the msg\_hop\_limit
   1. If it is zero, do not regenerate
   2. If not, decrement the value by one
2. Check if msg\_hop\_count is present and greater than or equal to MAX\_HOPCOUNT
   1. If so, do not regenerate
   2. If not, increment it by one
3. Change TargAddrMetric to match the route table entry for TargAddr, which should match the advertised value in the received RREP plus the cost of the link to the router which forwarded the RREP
4. Include the AckReq TLV if it was included in the original RREP or if this device is configured to request acknowledgement of RREP messages?

The RREP SHOULD be unicast to the next hop on the route to OrigAddr. If no forwarding route exists to OrigAddr, a RERR should be transmitted to TargAddr.

**RERR Message Structure:**

msg\_hop\_limit Part of RFC5444 Message Header

msg\_hop\_count Part of RFC5444 Message Header

PktSource RFC5444 Message TLV

MetricType (optional) RFC5444 Message TLV

Unreachable AddressList and PrefixList RFC5444 Address Block

(one or many)

UnreachableAddr Addresses combined into head, tail, mid, prefix length

UnreachableAddr PrefixLength

SeqNum RFC5444 Address Block TLV Block

Sequence Number List RFC5444 Address Block TLV (Section 10)

(one or many)

UnreachableAddr SeqNum RFC5444 Address Block TLV (Appendix B1)

Is the Sequence Number List the TLV? And its length can vary?

Or is AddrSeqNum a TLV which may or may not be included one or more times?

Is metric type important in a RERR, wouldn’t all routes to Address be down regardless of metric type?

**RERR Generation:**

There are two events which trigger generation of a RERR message. The first is the arrival of a packet for which there is no route to the destination address. This can be a packet forwarded by the routing process, or a RREP when there is no route to OrigAddr. The second is when a link breaks and one or more routes will be marked as Broken.

In constructing the RERR, AODVv2 uses Address List, Prefix List, Sequence Number List and Metric List. RFC5444 will take the Address List and Prefix List, combining into an Address Block for best compression. The Sequence Number List and Metric List are stored in new Address Block TLVs which will be placed in the Address Block TLV Block after the Address Block in the RFC5444 message.

The procedure for RERR generation is as follows:

1. Set msg\_hop\_limit to MAX\_HOPCOUNT
2. Set msg\_hop\_count to zero, if including it. SHOULD be included When would you not?
3. If the RERR was triggered by an Undeliverable Packet, the PktSource TLV MUST be included, containing the source IP address of the Undeliverable Packet
4. Include the MetricType TLV if reporting a broken route for a non-default metric type and set the correct value for Metric Type in the TLV
5. For the Unreachable Address List
   1. If the RERR was triggered by an Undeliverable Packet, insert the destination IP address of the packet which triggered the RERR, or if the packet was a RREP, insert the OrigAddr
   2. If the RERR was triggered by a broken link, include the addresses of all previously Active routes which are now broken. If the configuration option ENABLE\_IDLE\_IN\_RERR is enabled, include any previously Idle routes which have now broken
6. For the Prefix Length List
   1. Insert the prefixes associated with the addresses in the Unreachable Address List (would this be known for the undeliverable RREP case?)
7. For the Sequence Number List
   1. If known, the sequence numbers associated with the routes to the addresses in the Unreachable Address List SHOULD be included in the Sequence Number List (would there ever be some known, some unknown?)

If the RERR is sent in response to an Undeliverable Packet:

* It SHOULD be sent to the multicast IP and MAC address for LL-MANET-Routers
* It MAY be sent unicast to the next hop towards the source IP address of the packet which triggered the RERR

If the RERR is sent in response to a broken link:

* It MUST be sent to the multicast IP and MAC address for LL-MANET-Routers

Note that an ICMP Undeliverable Packet notification does not require a RERR message to be generated.

**RERR Reception:**

Upon receiving an RRER, the following steps are performed:

1. Check if the sender is on the blacklist of AODVv2 routers without bidirectional links
   1. If on the blacklist, check the Blacklist Remove Time
      1. If Current Time < Remove Time, ignore this message for further processing
      2. If not, remove the Blacklist entry and continue processing
   2. If not, continue processing
2. Check if the message contains the required data elements: msg\_hop\_limit and at least one Unreachable Address
   1. If not, ignore this message for further processing
   2. If so, continue processing
3. If the MetricType TLV is present, check that the MetricType is known.
   1. If not, ignore this message for processing
   2. If so, continue processing
4. For each Unreachable Address,
   1. Check that the address is valid (routable and unicast)
   2. Check that there is a route with the same MetricType matching the address using longest prefix matching
   3. Check that the route’s next hop is the sender of the RERR
   4. Check that the route’s next hop interface is the interface on which the RERR was received
   5. Check that the Unreachable Address SeqNum is either unknown, or is greater than the route’s SeqNum
   6. If any of the above are false, the Unreachable Address does not need to be advertised in a regenerated RERR
   7. If all of the above are true:
      1. If the route’s prefix length is the same as the Unreachable Address’s prefix length, set the route state to Broken.
      2. If the prefix length is shorter than the original route, the route MUST be expunged from the routing table, since it is a sub-route of the larger route which is reported to be broken.
      3. If the prefix length is different, create a new route with the Unreachable Address and its prefix, and set the state to Broken.
5. If there are no Unreachable Addresses which need to be advertised in a regenerated RERR, take no further action
6. Regenerate the RERR

**RERR Regeneration:**

The procedure for RERR generation is as follows:

1. Check the msg\_hop\_limit
   1. If it is zero, do not regenerate
   2. If not, decrement the value by one
2. Check if msg\_hop\_count is present and greater than or equal to MAX\_HOPCOUNT
   1. If so, do not regenerate
   2. If not, increment it by one
3. If the PktSource TLV was included in the original RERR, copy it into the regenerated RERR
4. If the MetricType TLV was included in the original RERR copy it into the regenerated RERR
5. For the Unreachable Address List
   1. Include all Unreachable Addresses which have been determined to need regeneration
6. For the Unreachable Address Prefix Length List
   1. Insert the prefixes associated with the addresses in the Unreachable Address List
7. For the Unreachable Address Sequence Number List
   1. Include corresponding sequence number information in the Unreachable Address Sequence Number List

If precursor lists are maintained, the outgoing RERR SHOULD be sent to the active precursors of the broken route as specified in Section 14.3. (one RERR with all the broken routes, to all the precursors of all the broken routes? – will probably give some precursors info they don’t care about)

If the original RERR was received by multicast, the regenerated RERR SHOULD be sent to the multicast IP and MAC address for LL-MANET-Routers.

If the original RERR contained the PktSource TLV, and a route exists to the source address, the regenerated RERR MUST be sent unicast to the next hop of the route towards PktSource.

Otherwise the regenerated RERR MUST be sent to the multicast IP and MAC address for LL-MANET-Routers.

**Evaluating Incoming Routing Information**

After determining that the incoming information is correctly formatted and contains values in the correct ranges, the AODVv2 router will use the information to update local routing information if possible. This section explains how to decide if the incoming information should be used to update the route table and how to update.

The incoming message may be a RREQ or a RREP. If it is a RREQ, it contains information about a route to OrigAddr. An RREP contains information about a route to TargAddr. In the following text, RteMsg is used to denote the received message, and AdvRte is used to denote the route information contained. AdvRte has the following properties:

* AdvRte.Address = OrigAddr (in a RREQ) or TargAddr (in a RREP)
* AdvRte.PrefixLength = OrigAddrPrefixLength (in a RREQ) or TargAddrPrefixLength (in a RREP) if present, else the maximum prefix length for the address family
* AdvRte.SeqNum = OrigAddrSeqNum (in a RREQ) or TargAddrSeqNum (in a RREP)
* AdvRte.MetricType = MetricType in the incoming RteMsg, if present, else DEFAULT\_METRIC\_TYPE
* AdvRte.Metric = Metric in the incoming RteMsg
* AdvRte.Cost = AdvRte.Metric + Cost of link to advertising router, calculated for same MetricType
* AdvRte.VALIDITY\_TIME? = validity time in the incoming RteMsg

Route denotes the current routing table entry and HandlingRtr is the router receiving the RteMsg.

To process the incoming information:

1. If a route entry already exists matching the AdvRte’s Address and Metric Type and PrefixLength?, determine whether AdvRte contains an update for the Route:
   1. Check if the information is stale (AdvRte.SeqNum < Route.SeqNum)
      * If stale, the incoming information is not allowed to update the current route, since that might result in routing loops. In this case the HandlingRtr MUST NOT update the current route table entry using the routing information for AdvRte.Address
      * If the information has a newer or identical sequence number, continue further checks to determine if it should be used to update the current route table entry
   2. Check if the information is safe against loops (LoopFree (AdvRte, Route) == TRUE)
      * See Section 6.5 for information on the LoopFree function. If LoopFree returns false, it cannot be guaranteed that the incoming information will avoid a routing loop, and it MUST NOT be used to update the current route table entry
   3. Check if the information shows a more costly route (AdvRte.Cost >= Route.Metric)
      * If the advertised route’s cost is the same or greater than the current route, and the current route’s state is Active or Idle, the incoming information does not offer any improvement, and using this information to update the current route table entry is not recommended
      * If the advertised route’s cost is the same or greater than the current route, but the current route’s state is Broken or Expired, this information can be used to update the current route table entry and put it back in the Idle state
      * If the advertised route’s cost is lower than the current route, this information offers improvement and SHOULD be used to update the current route table entry
   4. Update the entry as follows:
      * Prefix Length = AdvRte.PrefixLength
      * Sequence Number = AdvRte.SeqNum
      * Next Hop Address = Router which forwarded the RteMsg (IP.SourceAddress from the RteMsg)
      * Next Hop Interface = Interface on which the RteMsg was received
      * Metric = AdvRte.Cost
      * State = Idle if updating Broken or Expired route, don’t change if updating an Active/Idle route.
      * Last Used = Current Time
      * Expiration Time = AdvRte.VALIDITY\_TIME if present, or Current Time + ACTIVE\_INTERVAL + MAX\_IDLETIME

(AdvRte.SeqNum > Route.SeqNum) OR

((AdvRte.SeqNum == Route.SeqNum) AND

[(AdvRte.Cost < Route.Metric) OR

((Route.State == Broken) && LoopFree (AdvRte, Route))])

This is different to the text in the same section – i.e. this says if it’s newer, use it no matter what. Also loopfree is only checked if sequence numbers are the same, the cost is higher and the route is broken. Loopfree should always be checked I think?

1. If the routing table does not contain an entry matching AdvRte’s Address and MetricType and PrefixLength?, create one as follows:
   * + Address = AdvRte.Address
     + Prefix Length = AdvRte.PrefixLength
     + Sequence Number = AdvRte.SeqNum
     + Next Hop Address = Router which forwarded the RteMsg (IP.SourceAddress from the RteMsg)
     + Next Hop Interface = Interface on which the RteMsg was received
     + MetricType = AdvRte.MetricType
     + Metric = AdvRte.Cost
     + State = Active
     + Last Used = Current Time
     + Expiration Time = MAXTIME? / RteMsg.VALIDITY\_TIME if present, or Current Time + ACTIVE\_INTERVAL + MAX\_IDLETIME
     + Precursors List (optional) = empty

**Redundant RREQ Table (Section 6.6)**

When RREQ messages are flooded in a MANET, an AODVv2 router may receive similar RREQ messages from more than one of its neighbours. To avoid processing and transmission of redundant RREQ messages, while still enabling proper handling of earlier RREQ messages that may have somehow been delayed in the network, it is necessary for each AODVv2 router to keep a list of information about RREQ messages which it has recently received.

The stored information about each RREQ is:

* + - OrigAddr
    - TargAddr
    - OrigNodeSeqNum
    - TargNodeSeqNum – is this necessary?
    - MetricType
    - Metric
    - Timestamp (Current Time at the time the entry is updated)

When a RREQ is received, it is checked against the RREQ Table to see if it contains redundant information. If so it does not need to be processed, and the router can avoid regenerating multicast RREQ messages which would not only cause unnecessary signalling traffic and interference but also unnecessary processing at other routers.

The process for comparison is as follows:

1. Look for a “comparable” entry in the RREQ Table with the same OrigAddr and TargAddr and the same MetricType
2. If there is none, create an entry to store information about the received RREQ, and continue to process the RREQ since it contains new information
3. If there is an entry, and it has a lower OrigAddrSeqNum than the received RREQ, replace it with the new RREQ and continue to process the RREQ since it contains new information
4. If there is an entry and it has a higher OrigAddrSeqNum than the received RREQ, do not replace the entry and do not process the RREQ. The currently stored RREQ contains newer information
5. If there is an entry and it has the same OrigAddrSeqNum and a higher Metric than the received RREQ, replace it with the new RREQ information and continue to process the RREQ since it contains better information, i.e. it has come from a shorter route than the original
6. If there is an entry and it has the same OrigAddrSeqNum and a Metric less than or equal to the received RREQ, do not replace the entry and do not process the RREQ