**KimonoNet  
Enabling Efficient and Reliable Inter-UAV   
Fluid Data Communication Link**

Protocol Specification Document

*Document*  
Version 1.0.01  
February 26, 2012

*Authors*  
Bollens, Eric <ebollens@oit.ucla.edu>   
Hung, James <james400mhz@yahoo.com>  
Khalapyan, Zorayr <zkhalapyan@oit.ucla.edu> Norris, Wade <wade.norris@ucla.edu>

**Table of Contents**

1. Introduction 5

1.1. Status of This Memo 5

1.2. Copyright Notice 5

1.3. Abstract 5

2. Overview 5

2.1. Background Information 5

2.2. Proposed Solution 6

2.3. Use Cases 6

2.4. Terminology 7

2.5. User Characteristics 7

2.5.1. Autonomous Nodes 8

2.5.2. End Points 8

2.6. Constraints 8

2.6.1. High Churn 8

2.6.2. Mixed Horizons 9

2.7. Functions 9

2.7.1. Initialization Beacon 9

2.7.2. Beacon Acknowledgement 9

2.7.3. Recurring Beacon 10

2.7.4. Peer Communication 10

2.7.5. Location and Velocity Routing 10

2.7.6. Quality-of-Service TODO: Define QoS More Thoroughly 11

2.8. Definitions 12

Packet Types 12

Forwarding Modes 12

3. Routing Algorithm 13

3.1. Greedy Perimeter Selection Routing (GPSR) 13

3.1.1. Greedy Forwarding 13

3.1.2. Perimeter Forwarding 13

3.2. Predictive Velocity in Neighbor Maintenance 13

3.3. Quality of Service 13

4. Packet Format 14

4.1. General Packet Format 14

4.1.1. Composition 14

4.1.2. Common Header Format 14

4.1.3. Location Data Format 14

4.1.4. Vector Data Format 15

4.2. Type 0 Packet Format 15

4.2.1. Composition 15

4.2.2. Type 0 Header Format 15

4.2.3. Type 0 Extended Header 16

4.3. Type 2 & 3 Payload Format 16

4.3.1. Composition 16

4.3.2. Type 2 & 3 Header Format 16

4.3.3. Type 2 & 3 Neighbor Report Format 16

5. Composition 17

5.1. Packets 17

5.1.1. PKT-DATA (Type 0) 17

5.1.2. PKT-BEACON (Type 2) 18

5.1.3. PKT-BEACON-ACK (Type 3) 19

5.2. Structures 19

5.2.1. ROUTING-QUEUE 19

5.2.2. ROUTING-TABLE-1 20

5.2.3. ROUTING-TABLE-2 20

5.3. Timers 20

5.3.1. TMR-BEACON 20

5.4. Values 21

5.4.1. TMR-BEACON-VALUE 21

5.4.2. ROUTE-TABLE-REFRESH-VALUE 21

5.4.3. ROUTE-TABLE-EXPIRE-VALUE 21

6. Functions 21

7. State Machine 21

# 1. Introduction

## 1.1. Status of This Memo

This document specifies a prototype for a network protocol that may enable efficient and reliable network communication in fluid and sparse ad hoc networks. The specification outline forthwith is intended for academic use and not for production use. However, it may be extended post-facto for such use. Distribution of this memo is unlimited.

## 1.2. Copyright Notice

Copyright © Eric Bollens, James Hung, Zorayr Khalapyan and Wade Norris (2012). All Rights Reserved.

## 1.3. Abstract

This memorandum addresses a peer-to-peer network research topic related to routing and transport control issues in sparse networks of highly mobile ad hoc peers. Traditional routing algorithms are not suited for this topology given the high mobility and high churn of its constituents, which cause routes to shift quickly.

This sort of scenario may exist, for example, when deploying unmanned aerial vehicles in an area of operations without pervasive Internet connection. Under such a scenario, limited routes exist to a destination, and these may lie well beyond the network horizon of an individual node, motivating the need to establish routing across the peer-to-peer network.

The protocol described herein provides an implementation of Greedy Perimeter Selection Routing[[1]](#endnote-1) that adds two-hop neighbor awareness to minimize control packets and improve reliability in fluid and sparse graphs.

# 2. Overview

## 2.1. Background Information

The military has significantly increased the computing power it places in the field in recent years; however, the increase in network coverage has not followed suit. One option to increasing coverage relies on the development of an ad hoc peer-to-peer network whereby network-enabled devices in the field may form routes through a remote operational zone back to a secure base station.

Rather than providing the route via stationary relays, which make easy targets, unmanned aerial vehicles and other mobile equipment already equipped with basic network capabilities may provide such a network in hostile environments.

The described network may form a very sparse graph with limited routes to any endpoint. Further, given changing positions of nodes in the network and possible calamities that might befall its nodes, the network shall suffer from high churn. Consequently, an implementation must adapt quickly to topological changes and provide reasonably reliable service even when individual route exist only tenuously.

Beyond military application, this sort of research topic may provide insight into broader ad hoc networking issues. To address the dynamics of ad hoc networks, a number of protocols including Ad hoc On-Demand Distance Vector Routing (AODV)[[2]](#endnote-2) and Dynamic Source Routing (DSR)[[3]](#endnote-3) have taken the approach of on-demand routing, ascertaining a path at send-time rather than predetermining routes. These suffer from disadvantages including inconsistent, unreliable data routing and extra control overhead. Other ad hoc protocols such as the Optimized Link State Routing Protocol (OSLR)[[4]](#endnote-4) employ table-driven approaches reminiscent to traditional distance-vector protocols, but they struggle to adequately handle the rapidly changing nature of such a network without inordinate communication.

## 2.2. Proposed Solution

Given the unsuitability of existing ad hoc protocols in addressing the problems of a fluid and sparse network graph, as exists in the motivating example (§2.1), this protocol proposes a different approach. It leverages Greedy Perimeter Stateless Routing (GPSR), first proposed by Karp and Kung (2000), for neighbor selection, but extends it with neighborhood composition prediction based on two-hop knowledge. This greater topological knowledge allows the protocol to reduce control traffic while allowing a node to make more efficient and reliable choices about next-hop routing.

## 2.3. Use Cases

The motivating use case for this routing protocol is military operations with a high frequency of change in the network topology. In military operations, UAVs move rapidly and may also disappear from the network due to other failures. Consequently, the algorithm must not depend on the continuing existence of any given node in the network, but it may predict topology changes based on two-hop knowledge.

Further due to the sparse nature of this node graph, the algorithm must operate efficiently given limited routing options. The majority of nodes in the network will likely be out of range of any individual node, and the routing protocol must leverage knowledge of other nodes’ locations and velocities to make transmissions when in range.

This routing protocol may also extend beyond military operations. It has applicability to any network with high churn where the location and velocity of nodes is generally known. For example, this ad hoc communication protocol could very easily be extended to support orbiting satellites or maritime expeditions. However, this algorithm will not likely expand to other scenarios where position is not readily available or velocity regularly changes unexpectedly, and in such situations, other approaches may be better suited.

## 2.4. Terminology

Position Knowledge – Nodes have knowledge about the geographical location of their neighbors in terms of longitude and latitude, along with accuracy and the time of sampling. This directly allows them to perform GPSR-based routing, as well as make predictions about changing topology in conjunction with velocity knowledge.

Network Layer (NET) – A protocol that enables broadcast/multicast addressing or supports promiscuous mode to deliver all KimonoNet packets sent within communication range.

Network Interface Card (NIC) – A wireless interface that provides ad hoc networking capabilities.

Quality of Service (QoS) – Classification of a packet as time-sensitive, loss-intolerant, both or neither.

Two-Hop Knowledge – Nodes gain knowledge about both their direct neighbors and the neighbors of their neighbors through the beacon packet. In conjunction with velocity knowledge, this allows nodes to understand the changing state of their neighbors with less routing control packets.

Upper Layer Protocol (ULP) – A protocol that passes data to the protocol described herein.

Velocity Knowledge – Nodes have knowledge not only about the position of their neighbors, but also their velocity. In conjunction with two-hop knowledge, this allows nodes to understand the changing state of their neighbors with less routing control packets.

## 2.5. User Characteristics

Two primary constituents exist in this ad hoc scenario:

1. Autonomous nodes that send packets from ULP and route packets received from NET.
2. End points that receive data packets from NET and deliver packets to ULP.

In the motivating scenario, unmanned aerial vehicles meet the former classification whereas command posts and other external uplinks satisfy the latter.

This protocol assumes that the initial communication between these two user groups is sent from an autonomous node to an end point of known location. This avoids the need to introduce a flooding search algorithm in addition to the routing and transport mechanism that this protocol shall present. However, in real world scenarios, a search algorithm is anticipated to be necessary.

### 2.5.1. Autonomous Nodes

The primary constituents of the network, autonomous nodes have (1) awareness of their position and velocity, (2) a NIC that supports ad hoc communication, (3) a network layer that supports broadcast, multicast or promiscuous packet delivery, and (4) a running instance of the KimonoNet client. Further, these nodes are regarded as autonomous because they make independent decisions about position and velocity without considering its implications on routing.

Autonomous nodes collect data or accomplish an objective and then seek further instructions. In order to transmit this data or receive further instructions, these nodes introduce data packets into the ad hoc network. These packets are addressed to the known location of an endpoint (§2.5.2), and then node forwards this packet to a peer based on the routing algorithm (§3).

Further, beyond introducing packets into the network, autonomous nodes must also receive data packets passed to them by other nodes and then forward them on based on the routing algorithm.

### 2.5.2. End Points

Under this scenario, a command post or other external uplink to the Internet serves as the destination endpoint (sink) of a packet originating within the ad hoc network. Because this protocol does not consider a mechanism to determine endpoint location, it instead requires that the originator knows the location of the end point and that the position of the endpoint does not change over time.

As with autonomous nodes, end points must also have (1) awareness of their position and velocity, (2) a NIC that supports ad hoc communication, (3) a network layer that supports broadcast, multicast or promiscuous packet delivery, and (4) a running instance of the KimonoNet client that minimally supports transport functionality.

Except for in the case of the reliable QoS classification (surveillance data), the end point may use location and velocity information to reply to an autonomous node by ascertaining its position. Because an autonomous node has variable position, all responses from an end point should have QoS classification of time-sensitive but loss-tolerant.

## 2.6. Constraints

### 2.6.1. High Churn

Due to the operating environment, the routing protocol must handle high churn effectively.

Each autonomous node has a velocity that affects its position over time; consequently, the neighborhood for a given node may change swiftly and completely as nodes enter and recede from a network horizon. The routing algorithm handles this by considering 2-hop neighbors and extrapolating velocity over time to determine which nodes will be leaving range and which will be entering so as to help minimize required control traffic.

Further, given the motivating scenario, military operations also include the real possibility of nodes becoming unavailable due to failure or destruction. The algorithm presented must handle this independent influence on churn adequately, compensating for the fact that a route may become unavailable quickly and without warning. However, this protocol acknowledges that the unexpected disappearance of a node may cause the loss of a packet, but it takes strides to minimize the likelihood of this and provides a reliable QoS classification that may be used at the cost of no timeliness guarantee on delivery.

### 2.6.2. Mixed Horizons

In a sparse network, only a small set of nodes is likely within in range of any individual node; further, adjacent neighborhoods will share only a few nodes, if any. The routing algorithm must thus take this into account, as even adjacent neighborhoods may not be able to communicate and a route must instead be established through other nodes to traverse the gap.

This constraint is the reason that link-state management is non-optimal: by the time global route information has propagated, routes may have changed substantially due to mobility. This protocol thus operates under the assumption that nodes cannot self-route data through the entire network, and instead that nodes make decisions based solely on their local neighborhood.

## 2.7. Functions

### 2.7.1. Initialization Beacon

Upon initialization, a peer must transmit a beacon packet that allows for discovery by neighboring nodes. This beacon employs NET to provide a description of the node's identity, location and velocity vector to all other nodes within its network horizon.

During initialization, the node has no awareness of its neighbor; therefore, the beacon shall contain only information about the peer and not any neighbors. This differs from beacon acknowledgements (§2.7.2) and recurring beacons (§2.7.3), which can also include information about neighboring nodes.

### 2.7.2. Beacon Acknowledgement

Upon receiving a beacon or beacon acknowledgement from another node, the recipient must inspect the beacon’s contents. The recipient node must send a beacon acknowledgement if and only if the received beacon or beacon acknowledgement does not contain an entry for the recipient within its list of neighbors. This beacon acknowledgement must contain the identification, location and velocity of the node that received the beacon or beacon acknowledgement, as well as any neighbors within its network horizon.

If the received beacon does contain an entry for the recipient, the recipient must not send a beacon acknowledgement under any circumstance. This prevents acknowledgement loops. It may, however, store any more up-to-date information contained within the received beacon or beacon acknowledgement.

### 2.7.3. Recurring Beacon

At regular intervals, a node shall transmit another beacon packet. If the node has knowledge of any neighbors within its network horizon, the beacon should also include the identification, position and velocity vector of these peers. It shall not include any nodes that are not within its own network horizon.

When a node receives a beacon that does not include information about itself, it must transmit a beacon acknowledgement (§2.7.2). If the received beacon contains information about the recipient, the recipient must provide no acknowledgement, although it may store any more up-to-date information contained within the received beacon.

### 2.7.4. Peer Communication

This protocol provides a third packet format for conveying data. Each data packet includes a forwarding address, and only the node with this identifier shall receive and act on this data. If the node is also the intended recipient, then this data packet should be provided to ULP. If the node is not the intended recipient, then the node must instead implement the routing algorithm to define a new forwarding address and then retransmit the packet. This will allow for a data packet to travel across the network to its ultimate destination.

**TODO: IT NEEDS TO BE DECIDED HOW QOS WILL BE IMPLEMETED! THIS FOLLOWING STATEMENT MAY BE COMPLETELY INACCURATE…** In the event that reliable QoS is required, a beacon acknowledgement including the data identifier may prove necessary. Therefore, each data packet shall also include a sequence number.

### 2.7.5. Location and Velocity Routing

In the event that a node is not the intended destination for a data packet it is forwarded, then it must use the routing algorithm to interpret knowledge of its local neighborhood and determine the next node to which the packet should be addressed. It must then retransmit the packet as such.

Based on Greedy Perimeter Stateless Routing (GPSR), the selection of this next relay node occurs based on the current forwarding mode of the packet.

#### 2.7.5.1. Greedy Forwarding Mode

If the packet is being forwarded under the greedy mode, and the node can select a neighbor closer to the final destination, then this neighbor is selected. If the packet is being forwarded under greedy mode, but the node is the local maxima to the packet’s intended destination in its neighborhood, then the forwarding mode is switched to perimeter mode.

The protocol seeks to use greedy routing as often as possible, but a sparse graph may require detours to reach the intended destination. In order to simultaneously minimize control packets and ensure greedy selection whenever possible, the routing algorithm shall use not only location information for its neighbors, but also a predictive algorithm to determine if a neighbor of a neighbor, as described in the neighbor's beacon, has come into range since the beacon was sent.

#### 2.7.5.2. Perimeter Mode

A data packet is switched to perimeter forwarding mode when it encounters a node that is a local maxima to the packet’s intended destination based on the nodes in its neighborhood. It remains in perimeter mode until it crosses the vector between the node where it entered perimeter mode and its final destination.

While in perimeter mode, the next neighbor is selected based on Relative Neighborhood Graph (RNG) planarization[[5]](#endnote-5) and right-hand rule traversal as described in Greedy Perimeter Stateless Routing.

### 2.7.6. Quality-of-Service TODO: Define QoS More Thoroughly

The routing algorithm will be implemented with four distinct Quality of Service (QoS) grades.

#### 2.7.6.1. Control Data

Beacons and beacon acknowledgements are classified as control data, which must be transmitted and interpreted immediately upon reception and are loss-intolerant. However, it should be noted that time sensitivity is only guaranteed in the scenario when packet loss does not occur. When packet loss does occur, information contained in a beacon or beacon acknowledgement may not be delivered for some period of time, but it will eventually be delivered if the data is still pertinent.

Time-sensitive data packets may not be classified as control data. For time-sensitive data packets, the communication data classification (§2.7.6.2) should be used, and, for loss-intolerant data packets, the surveillance data classification (§2.7.6.3) should be used. This protocol does not support multi-hop data transmission that is both time-sensitive and reliable.

#### 2.7.6.2. Communication Data

While not critical for the proper functioning of the ad hoc network, these data packets are regarded as time sensitive and thus, with the exception of control data, communication data should be transmitted as soon as possible.

When timeliness is required, data packets shall not also be classified as reliable and must thus be loss tolerant. Any effort to ensure reliability of communication data must come from ULP.

#### 2.7.6.3. Surveillance Data TODO

While not time sensitive, surveillance data should be regarded as lossless. TODO: Determine how we’re going to provide reliable delivery – hop-by-hop chain of responsibility ensures that a data packet is never sent that isn’t received by some other node, but if that node is lost while it has the data in buffer, then the data will be lost too.

#### 2.7.6.4. Standard Data

This data does not have any specific quality of service requirements. It is routed with deference to communication data, while with the same priority as surveillance data without the requirement of reliability that service data imposes.

## 2.8. Definitions

### Packet Types

#### PKT-DATA

Network packet with Type 0 that contains data to be routed.

#### PKT-BEACON

Network packet with Type 2 that is sent on initialization and after a timer expires to allow neighbor nodes to maintain up-to-date neighborhood information.

#### PKT-BEACON-ACK

Network packet with Type 3 that is sent upon receipt of any PKT-BEACON or PKT-BEACON-ACK that does not include the node in its neighbor report.

### Forwarding Modes

#### FWD-GREEDY

Select via greedy mechanism the neighbor closest to destination for forwarding.

#### FWD-PERIMETER

Use planar traversal to bypass when node cannot do greedy selection because it represents a local minima in greedy selection.

# 3. Routing Algorithm

## 3.1. Greedy Perimeter Selection Routing (GPSR)

### 3.1.1. Greedy Forwarding

When a packet is received in greedy mode:

* Node forwards received packet to neighbor closest to destination node.
* If node is closest of its known neighbors, then switch to perimeter forwarding.

**TODO: Please see Karp/Kung paper on the topic for now**

### 3.1.2. Perimeter Forwarding

When a packet is received in or switched to perimeter mode...**TODO**

**TODO: Please see Karp/Kung paper on the topic for now**

## 3.2. Predictive Velocity in Neighbor Maintenance

This extension of GPSR updates ROUTING-TABLE-1 that contains direct hop neighbors with information contained in ROUTING-TABLE-2 by factoring in velocity and time.

**TODO: Need to define this method**

## 3.3. Quality of Service

* Reliable QoS grade is simplex (end point cannot reply).
* Except for in the case of the reliable QoS classification (surveillance data), the end point may use location and velocity information to reply to an autonomous node by ascertaining its position. Because an autonomous node has variable position, all responses from an end point should have QoS classification of time-sensitive but loss-tolerant.

# 4. Packet Format

## 4.1. General Packet Format

### 4.1.1. Composition

All KimonoNet packets contain two components:

* Common Header
* Type-specific Header
* Type-specific Data

The common header type field HDR-TYPE defines the format of the type-specific header and data. The common header is always fixed width itself, and the type-specific header is fixed width for each given type, although the type-specific header varies between the types.

The full length of the header (common and type-specific combined) must be 4-byte word aligned. If it is not, then any extra bytes to achieve this alignment should be composed of 0x00.

### 4.1.2. Common Header Format

The common header includes the following fields:

* HDR-MAGIC (2)
* HDR-VERSION (1)
* HDR-TYPE (1)
* HDR-SRC-ID (8)
* HDR-SRC-LOC (24)
* HDR-SRC-VEC (12)

The common header has a total length of 48 bytes.

### 4.1.3. Location Data Format

Location data is represented as an object containing the following fields:

* LATITUDE (8)
* LONGITUDE (8)
* ACCURACY (4)
* TIME (4)

This data representation has a total length of 24 bytes.

### 4.1.4. Vector Data Format

Vector data is represented as an object containing the following fields:

* SPEED (4)
* BEARING (8)

This data representation has a total length of 12 bytes.

## 4.2. Type 0 Packet Format

### 4.2.1. Composition

The Type 0 packet format begins with the common header.

Following the common header, Type 0 defines additional header fields for several purposes:

* Define the destination for packet routing via HDR-DST-ID and HDR-DST-LOC.
* Define whether the forwarding mode is FWD-GREEDY or FWD-DETOUR.
* Define the size of the variable-length data in octets (maximum data length is 2048 bytes).
* Provide a CRC-16 checksum of common and Type 0 header fields with HDR-HDR-CHK set to 0.

The total length of the Type 0 header format, including the common header, is 96 bytes.

### 4.2.2. Type 0 Header Format

The Type 0 header includes the following fields:

* HDR-DST-ID (8)
* HDR-DST-LOC (24)
* HDR-FWD-DST-ID (8)
* HDR-FWD-MODE (1)
* HDR-DATA-LEN (2)
* HDR-HDR-CHK (2)
* HDR-QOS (1)
* HDR-PADDING (2)

### 4.2.3. Type 0 Extended Header

If HDR-FWD-MODE provides additional fields needed for perimeter forwarding as defined by the GPSR paper:

* XHDR-ENTERED-LOC (24)
* XHDR-FACE-ENTERED-LOC (24)
* XHDR-FACE-FIRST-EDGE-SRC (24)
* XHDR-FACE-FIRST-EDGE-DST (24)

This extended header adds an overhead of 96 bytes. This could be heavily optimized in a production implementation, but is kept generic for this prototypical implementation.

These bytes are not included in the HDR-HDR-CHK.

## 4.3. Type 2 & 3 Payload Format

### 4.3.1. Composition

Following the common header, the Type 2 & 3 payload defines two additional header fields.

If HDR-NEIGHBOR-COUNT is greater than zero, then it has a non-zero data field with length as the multiple of HDR-NEIGHBOR-COUNT and the neighbor report packet length (44 bytes). To keep packet size below 1500 bytes, HDR-NEIGHBOR-COUNT has a maximum value of 34.

### 4.3.2. Type 2 & 3 Header Format

The Type 2 &3 header includes the following fields:

* HDR-NEIGHBOR-COUNT (1)
* HDR-PKT-CHK (2)

The total length of the Type 2 & 3 header format, including the common header, is 52 bytes.

### 4.3.3. Type 2 & 3 Neighbor Report Format

Following the header, Type 2 & 3 packets include HDR-NEIGHBOR-COUNT neighbor reports.

A neighbor report includes the following fields:

* NEIGHBOR-ID (8)
* NEIGHBOR-LOC (24)
* NEIGHBOR-VEC (12)

The total length of a neighbor report is 44 bytes.

# 5. Composition

## 5.1. Packets

All packets are sent via UDP broadcast, and all nodes listen for UDP broadcast.

Among packets received over UDP broadcast, a node acts based on HDR-TYPE:

* **Type 0 (Data)**: A node processes data packets if HDR-FWD-DST-ID is node ID. When HDR-DST-ID is also the node ID, then the data packet is passed to ULP. Otherwise, the routing algorithm determines the next HDR-FWD-DST and rebroadcasts the UDP packet again.
* **Type 2 (Beacon)**: A node processes all beacon packets. If the HDR-SRC-LOC timestamp is newer than the latest record in ROUTING-TABLE for the sender, then it updates the table. It also updates ROUTING-TABLE for any neighbor report with a newer NEIGHBOR-LOC timestamp. If no row for HDR-SRC-ID exists, then the node transmits a Type 3 (Beacon Acknowledgement) packet in response, including the new node in its neighbor report.
* **Type 3 (Beacon Acknowledgement)**: A node processes all beacon acknowledgement packets. It treats them identically to Type 2 packets with one exception: if it receives a Type 3 packet that contains its own node ID in a neighbor report, it does not send its own beacon acknowledgement. This prevents loops in neighbor discovery.

Further details on these packets follow forthwith.

### 5.1.1. PKT-DATA (Type 0)

This packet contains data that the network routes through internal nodes until it reaches a destination.

When PKT-DATA arrives via UDP broadcast:

* If HDR-SRC-ID is the same as the node's ID:
  + Discard PKT-DATA.
* Else if HDR-FWD-DST-ID is not the same as the node's ID:
  + Discard PKT-DATA.
* Else if HDR-DST-ID is the same as node's ID:
  + Extract and pass to ULP.
* Else:
  + Pass to ROUTING-QUEUE.

This conceptually means that the node drops any packet that it itself originated or that it receives without being marked as the intended node to forward. Meanwhile, it extracts and asses any data packet to ULP if it is the final destination, or else it places the packet in the routing queue

### 5.1.2. PKT-BEACON (Type 2)

This packet is used to deliver information about a node to all nodes within reception range. It also conveys information about any direct neighbors it has that it knows about.

This packet is sent under two conditions:

1. Initialization
2. Expiration of TMR-BEACON

Whenever PKT-BEACON is sent, TMR-BEACON is reset to TMR-BEACON-VALUE.

On initialization, PKT-BEACON will include no neighbor reports; in future iterations, the packet may contain up to 38 neighbor reports.

The expiration condition, meanwhile, ensures that either PKT-BEACON or PKT-BEACON-ACK is sent every TMR-BEACON-VALUE seconds. Because PKT-BEACON-ACK includes the same information as PKT-BEACON, it is not necessary to send PKT-BEACON if PKT-BEACON-ACK has just been sent.

When PKT-BEACON arrives via UDP broadcast:

* If HDR-SRC-ID is the same as the node's ID:
  + Discard PKT-BEACON and exit.
* If HDR-SRC-LOC is newer than ROUTING-TABLE[HDR-SRC-ID],
  + Update ROUTING-TABLE[HDR-SRC-ID] = HDR-SRC-LOC
* For each NEIGHBOR-REPORT:
  + If NEIGHBOR-LOC is newer than ROUTING-TABLE[NEIGHBOR-ID]
    - Update ROUTING-TABLE[NEIGHBOR-ID] = NEIGHBOR-LOC
* Create PKT-BEACON-ACK for node and send via UDP broadcast.
* Reset TMR-BEACON.

PKT-BEACON is never forwarded.

### 5.1.3. PKT-BEACON-ACK (Type 3)

This packet is sent in response to reception of any PKT-BEACON or PKT-BEACON-ACK that does not include the node receiving PKT-BEACON-ACK in its neighbor report.

Whenever PKT-BEACON-ACK is sent, TMR-BEACON is reset to TMR-BEACON-VALUE. This ensures that only the minimum required number of PKT-BEACON packets are sent.

When PKT-BEACON arrives via UDP broadcast:

* If HDR-SRC-ID is the same as the node's ID:
  + Discard PKT-BEACON and exit.
* If HDR-SRC-LOC is newer than ROUTING-TABLE[HDR-SRC-ID],
  + Update ROUTING-TABLE[HDR-SRC-ID] = HDR-SRC-LOC
* For each NEIGHBOR-REPORT:
  + If NEIGHBOR-LOC is newer than ROUTING-TABLE[NEIGHBOR-ID]
    - Update ROUTING-TABLE[NEIGHBOR-ID] = NEIGHBOR-LOC
* If NEIGHBOR-REPORT did not include the node's ID:
  + Create PKT-BEACON-ACK for node and send via UDP broadcast.
  + Reset TMR-BEACON.

PKT-BEACON-ACK is never forwarded.

## 5.2. Structures

### 5.2.1. ROUTING-QUEUE

ROUTING-QUEUE contains PKT-DATA datagrams added because they had HDR-FWD-DST-ID corresponding to the node. This means that the previous relay intended this node as the next hop.

When a datagram is popped from ROUTING-QUEUE, the routing algorithm is invoked to determine the new forwarding destination address for PKT-DATA.

The routing algorithm considers three components:

* HDR-FWD-MODE: Whether the packet is currently in greedy or perimeter mode.
* HDR-DST-LOC: The intended final destination of the packet.
* ROUTING-TABLE-1: The table of known neighbors

Once this address has been determined, HDR-FWD-DST-ID is updated. In addition, HDR-FWD-MODE may also be updated, depending on if the routing algorithm has to switch forwarding mode.

### 5.2.2. ROUTING-TABLE-1

This table contains all current neighbors through which DATA-PKT may be routed.

The values in this table are refreshed at an interval of ROUTING-TABLE-REFRESH-VALUE based on the position and velocity data contained within ROUTING-TABLE-2. This allows for nodes to be considered neighbors that have been learned about only from a neighbor report by a direct neighbor, if their location and velocity data indicates as such.

### 5.2.3. ROUTING-TABLE-2

This table contains all known neighbors and all known neighbors of neighbors.

During a refresh of ROUTING-TABLE-1, the location, velocity and timestamps contained in this table are used to determine all direct neighbors. This allows neighbors of neighbors to be considered as neighbors if their location and velocity information indicates that they have moved into range.

All routes more than ROUTE-TABLE-EXPIRE-VALUE should be removed from this table before a refresh of ROUTING-TABLE-1 to ensure that routes do not become too stale.

## 5.3. Timers

### 5.3.1. TMR-BEACON

This timer tracks how long since the last PKT-BEACON or PKT-BEACON-ACK was sent.

If this timer expires, the node is responsible for sending a new PKT-BEACON with a priority equivalent to Control Data.

This timer is thus reset whenever PKT-BEACON or PKT-BEACON-ACK is sent.

The value that this timer is always set to is defined as TMR-BEACON-VALUE.

## 5.4. Values

### 5.4.1. TMR-BEACON-VALUE TO BE DEFINED

Default value is **TO BE DEFINED**

### 5.4.2. ROUTE-TABLE-REFRESH-VALUE TO DEFINE THIS BETTER

Default value is TMR-BEACON / 2

### 5.4.3. ROUTE-TABLE-EXPIRE-VALUE TO DEFINE THIS BETTER

Default value is TMR-BEACON \* 4

# 6. Functions TODO

# 7. State Machine TODO

1. GPSR Specification Citation [↑](#endnote-ref-1)
2. AODV Specification Citation [↑](#endnote-ref-2)
3. DSR Specification Citation [↑](#endnote-ref-3)
4. OLSR Specification Citation [↑](#endnote-ref-4)
5. RNG Citation [↑](#endnote-ref-5)