KimonoNet

Enabling Efficient and Reliable Inter-UAV Fluid Data Communication Link

Protocol Specification Document

Document

Version 1.0.11

March 14, 2012

Authors

Bollens, Eric <ebollens@oit.ucla.edu>

Hung, James <james.h@ucla.edu>

Khalapyan, Zorayr <zkhalapyan@oit.ucla.edu>

Norris, Wade <wade.norris@ucla.edu>

Table of Contents

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 purposes and not production use. However, it may be extended post-facto for such use. Distribution of this memo is unlimited.

1.2. Copyright Notice

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 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) and Dynamic Source Routing (DSR) 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) 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.

This protocol makes two additional simplifying assumptions: (1) it does not consider altitude, and (2) it assumes symmetric communication between any two nodes in the graph whereby, if a node can communicate with another node, the reverse is true as well.

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 from another node, the recipient must inspect the beacon's contents. The recipient node must send a beacon in response if and only if the received beacon does not contain an entry for the recipient within its list of neighbors. This response must contain the identification, location and velocity of the node that received the beacon, 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 response but rather wait until the beacon recurrence interval. 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 in response (§2.7.2). If the received beacon contains information about the recipient, the recipient must provide no such response, although it may store any more up-to-date information contained within the received beacon.

2.7.4. Peer Communication

This protocol provides a separate 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.

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 and right-hand rule traversal as described in Greedy Perimeter Stateless Routing.

2.7.6. Quality-of-Service

The routing algorithm will be implemented with three distinct Quality of Service (QoS) grades. These grades address time-sensitivity, while any effort to ensure reliability must come from ULP.

2.7.6.1. Control Data

Beacons are classified as control data. Generally, they are time-sensitive and lossless. It should be noted, however, that in the event of packet loss, time-sensitivity is no longer guaranteed. When packet loss does occur, information contained within the beacon may not be delivered for some period of time, but it will eventually be resent if the data is still pertinent.

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.

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.

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.

2.8. Definitions

Packet Types

PKT-DATA

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

PKT-BEACON

Network packet with Type 1 that is sent on initialization and after a timer expires to allow neighbor nodes to maintain up-to-date neighborhood information.

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.

Structures

PACKET-QUEUE

This queue contains PKT-DATA packets that have arrived at the peer but that are not ultimately destined for this peer, thus queued for transmission once they are popped and a new forwarding address is ascertained based on ROUTING-TABLE-1 and the Greedy Perimeter Stateless Routing algorithm (§3.1).

ROUTING-TABLE-1

This table contains the identifier and location of peers within the network horizon of the node as computed based on their position and velocity from ROUTING-TABLE-2 by the Predictive Velocity Neighbor Maintenance algorithm (§3.2).

ROUTING-TABLE-2

This table contains the identifier, timestamp, location and velocity of all known neighbors and all known neighbors of neighbors.

This table is populated based data within PKT-BEACON, including both information on the transmitting peer, as well as any neighbors learned in a neighborhood reports. Nodes are dropped from this table if their timestamp is older than TMR-ROUTE-EXPIRE-VALUE.

The Predictive Velocity Maintenance algorithm (§3.2) uses these values to build ROUTING-TABLE-1 at a frequency defined by TMR-ROUTE-TABLE-REFRESH.

3. Routing Algorithm

This section describes two different sets of algorithms required for the protocol:

- Greedy Perimeter Selection Routing (§3.1) - A functional implementation of the protocol initially presented by Karp and Kung.
- Predictive Velocity Neighbor Maintenance (§3.2) - A method for maintaining knowledge of immediate neighbors using information from neighbor beacons about two-hop neighbors.

While this protocol makes recommendations concerning the algorithms, other methods may be used that achieve the same effect, taking into account design decisions about computational complexity, communication range and location/velocity accuracy. Generally, a deployment of this protocol should seek uniformity in implementation throughout, but this is not necessary in the event that factors such as heterogeneous hardware make it unviable.

3.1. Greedy Perimeter Selection Routing (GPSR)

3.1.1. Greedy Forwarding

When a packet is popped from PACKET-QUEUE, if HDR-FWD-MODE is FWD-GREEDY, then it is handled as follows:

- The node computes the distance between each of its neighbors and the destination and selects the neighbor closest to the destination as HDR-FWD-DST-ID based on this information.
- If the node itself is closer to HDR-DST-LOC than any of its neighbors, then the packet is switched to FWD-PERIMETER and perimeter forwarding is used instead (§3.1.2).

Greedy forwarding is ideal because it relies only on knowledge of the forwarding nodes immediate neighbors and always makes the most efficient selection based on this data. As such, a packet should be forwarded greedily whenever possible.

However, greedy selection fails when the node itself is the closest node among its neighbors to the intended destination of the packet (local maxima), thus motivating the need for perimeter routing (§3.1.2).

3.1.1.1. Finding a Local Minima among Neighbors

This process determines the node closest to the destination from the nodes in this set by consulting ROUTING-TABLE-1, which contains the set of peers currently within a node's network horizon.

Given $(longitude, latitude) = (\psi, \phi)$ of two points and the radius of Earth r, the haversine formula provides a mechanism for determining the orthodomic distance between two points on a sphere:

$$DIST\big((\psi_1, \phi_1), (\psi_1, \phi_1)\big) = 2r * \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\psi_2 - \psi_1}{2}\right)}\right)$$

When hardware does not support computational complexity related to this formula or when network horizons are limited enough in range, a two-dimensional plane may be used for the calculation. When ranges increase and the hardware allows, the haversine formula can also be extended to support the ellipsoid nature of the Earth as opposed to a uniform radius.

Regardless of the implementation of $DIST(r, d)$, the distance between the final destination $d = (\psi_1, \phi_1)$ and each neighbor r in ROUTING-TABLE-1 R is defined as follows:

$$D = \{x | \forall r \in R, x = DIST(r, d)\}$$

The intended peer p to forward the packet to is thus arrived at as the minimum of this set D:

$$p = \min(D)$$

3.1.1.2. Forwarding the Packet

Before forwarding the packet via FWD-GREEDY, the node n must ascertain if the distance between the peer p and the final destination is less than its own distance from the final destination $d = (\psi_1, \phi_1)$:

$$DIST(p, d) < DIST(n, d)$$

If this is true, then it should modify HDR-FWD-DST-ID to correspond to the identifier for node p and then retransmit the packet. However, if the node itself is the local minimum, then HDR-FWD-DST-MODE should be set to FWD-PERIMETER and the packet should be passed to the forwarding algorithm.

3.1.2. Perimeter Forwarding

Perimeter forwarding is used to circumnavigate a void where greedy forwarding is not possible.

When a packet is in FWD-PERIMETER, it is routed under the following paradigm:

- If the node just transitioned to FWD-PERIMETER, add XHDR fields to packet and set both XHDR-ENTERED-LOC and XHDR-ENTERED-FACE-LOC to node's location.
- If node is closer to HDR-DST-LOC than XHDR-ENTERED-LOC, switch to FWD-GREEDY.
- Based on ROUTING-TABLE-1, compute the bearing to all neighbors and set HDR-FWD-DST-ID to the edge with next angle counter-clockwise from edge between HDR-DST-LOC and XHDR-ENTERED-FACE-LOC.
- If the edge between the node and the chosen HDR-FWD-DST intersects the edge between HDR-DST-LOC and XHDR-ENTERED-LOC, then set XHDR-FACE-ENTERED-LOC-SRC to node's location and XHDR-FACE-ENTERED-LOC-DST to the node where the packet is forwarded.

3.1.2.1. Packet Switched to Perimeter Forwarding

When a packet is switched to perimeter forwarding, its header field must include four properties necessary for perimeter forwarding:

- XHDR-ENTERED-LOC
- XHDR-FACE-ENTERED-LOC
- XHDR-FACE-FIRST-EDGE-SRC
- XHDR-FACE-FIRST-EDGE-DST

Initially, XHDR-ENTERED-LOC, XHDR-FACE-ENTERED-LOC and XHDR-FACE-FIRST-EDGE-SRC are all set to the location of the node where the packet switched into FWD-PERIMETER. XHDR-FACE-FIRST-EDGE-DST is selected based on next hop selection (§3.1.2.2) for FWD-PERIMETER.

Over time, XHDR-FACE-ENTERED-LOC, XHDR-FACE-FIRST-EDGE-SRC and XHDR-FACE-FIRST-EDGE-DST may all change. However, XHDR-ENTERED-LOC will remain constant for the duration of FWD-PERIMETER because it is used to determine when a node may switch the packet back to FWD-GREEDY.

3.1.2.2. Next Hop Selection in Perimeter Forwarding

When a packet is switched to FWD-PERIMETER (§3.1.2.1) or when it is popped from PACKET-QUEUE with FWD-PERIMETER set, the node must then select the next hop destination and set HDR-FWD-DST-ID. This selection may also require updates to XHDR-FACE values.

Considering edges between the node and all of its neighbors, the node selects its neighbor from ROUTING-TABLE-1 with the edge that is next counter-clockwise from the edge between HDR-DST-LOC and XHDR-FACE-ENTERED-LOC.

Given $(longitude, latitude) = (\psi, \phi)$ of two points, the orthodomic bearing may be calculated as:

$$BEARING\big((\psi_1, \phi_1), (\psi_1, \phi_1)\big) = \arctan2(\cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\psi_2 - \psi_1), \sin(\psi_2 - \psi_1)\cos(\phi_2))$$

When the network diameter is relatively small in comparison to the radius of the Earth or when hardware does not allow computation of this complexity, it may be reasonable to simplify this computation to angles on a plane.

Regardless of the computation method, the set of bearings may be computed in relation to the source node $s = (\psi_1, \phi_1)$ as follows for each node r in ROUTING-TABLE-1 R:

$$N = \{x | \forall r \in R, x = BEARING(r, s)\}$$

Having generated N as the set of bearings of all neighbors and s_b as the bearing between HDR-DST-LOC and XHDR-ENTERED-FACE-LOC, then the recipient is selected as the minimum bearing for $n \in N, n_b - s_b > 0$ or else the maximum bearing for $n \in N, n_b - s_b \leq 0$.

The node that is selected based on this algorithm is assigned to HDR-FWD-DST, and XHDR-ENTERED-FACE-LOC values are updated for a selection that changes the face.

3.1.2.3. Detecting Unreachable Host

Perimeter routing determines if a host is unavailable by detecting a loop. The responsibility for this detection rests on the node where the packet entered the face.

Therefore, when a packet is popped from PACKET-QUEUE (§3.1.2.1), the node should check if its identifier corresponds to XHDR-ENTERED-FACE-LOC-SRC. If this is indeed the case, then the packet has undergone a cycle in its planar graph, implying the destination is unreachable, and thus the packet should be dropped. This prevents loops, as FWD-PERIMETER will never traverse the same face twice.

3.1.1.4. Switching Back to Greedy Forwarding

Before determining a next hop (§3.1.2.2), the forwarding node should determine if its distance to HDR-DST-LOC is less than the distance from XHDR-ENTERED-LOC. A node should make this determination by computing the orthodomic distance to HDR-DST-LOC for both itself and XHDR-ENTERED-LOC via $DIST(r, d)$.

If the current node is closer to the destination than when the node entered FWD-PERIMETER, then the packet may return to greedy forwarding. In this case, the node should strip the XHDR fields, mark HDR-FWD-MODE to FWD-GREEDY and employ greedy forwarding (§3.1.1).

3.2. Predictive Velocity Neighbor Maintenance

This extension of GPSR updates ROUTING-TABLE-1 periodically with information contained in ROUTING-TABLE-2 by computing current node positions based on position, velocity and time data:

- Compute the new location of each node in ROUTING-TABLE-2 based on location and velocity.
- Add nodes with a new location less than NET-EFFECTIVE-RANGE to ROUTING-TABLE-1.

This process should run asynchronously from the routing process, although it must block while updating ROUTING-TABLE-1 to avoid a race with the Greedy Perimeter Stateless Routing algorithm (§3.1).

3.2.1. New Location Computation

When performing an update to ROUTING-TABLE-1, the first step is to compute the new location of each node in ROUTING-TABLE-2.

Given the radius of the Earth r, the speed of the node s, the time since the neighbor information was generated Δt, bearing of the node θ, and the location $(longitude, latitude) = (\psi_0, \phi_0)$ of the node as defined in ROUTING-TABLE-2, the new location $(longitude, latitude) = (\psi_1, \phi_1)$ may be calculated:

$$\phi_1 = \operatorname{asin}\left(\sin(\phi_0)\cos\left(\frac{s\Delta t}{r}\right) + \cos(\phi_0)\sin\left(\frac{s\Delta t}{r}\right)\cos(\theta)\right)$$

$$\psi_1 = \psi_0 + \operatorname{arctan2}\left(\cos\left(\frac{s\Delta t}{r}\right) - \sin(\phi_0)\sin(\phi_1)\,,\sin(\theta)\sin\left(\frac{s\Delta t}{r}\right)\cos(\phi_0)\right)$$

This orthodomic position calculation requires velocity in the same units as the radius and time. Alternative methods may be employed for this calculation where necessary, such as using a two dimensional plane if the Earth's radius is proportionally large in comparison to NET-EFFECTIVE-RANGE or if the computation is too complex for the hardware.

It should be recognized, however, that these calculations are propagated between neighboring nodes through beacons. Therefore, while uniformity is not a requisite, a lack thereof may lead to varying ways in which different nodes regard neighbors.

3.2.2. Creation of ROUTING-TABLE-1

Based on the new positions $(longitude, latitude) = (\psi_1, \phi_1)$ for all nodes in ROUTING-TABLE-2, this algorithm should then define ROUTING-TABLE-1 N as comprised of all nodes with a distance from current node n less than or equal to the NET-EFFECTIVE-RANGE d_{max} as such:

$$N = \{n | \forall r \in R, x = DIST(r, n), x \leq d_{max}\}$$

This table may be computed in temporary storage to reduce the amount of time that it must block the forwarding algorithm. However, during the copy into the actual ROUTING-TABLE-1, the forwarding algorithm must not make any decisions based on partial data, so this update must block for the copy.

3.3. Quality of Service

Quality-of-service should be implemented in priority order as follows:

- Control Data
- Communication Data
- Regular Data

At no point in time should the node send data from a lower priority queue when data is waiting to be sent in a higher priority queue.

4. Packet Format

4.1. General Packet Format

4.1.1. Composition

All KimonoNet packets contain three 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 (8)

The common header has a total length of 44 bytes.

4.1.3. Location Data Format

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

- LONGITUDE (8)
- LATITUDE (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 (4)

This data representation has a total length of 8 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 the next hop in packet routing with HDR-FWD-DST-ID.
- Define whether the forwarding mode is FWD-GREEDY or FWD-PERIMETER.
- Provide perimeter routing XHDR parameters if in FWD-PERIMETER.
- Define the size of the variable-length data in octets (maximum data length is 1284 bytes).
- Provide a CRC-32 checksum of common and Type 0 header fields with HDR-HDR-CHK set to 0.

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-QOS (1)
- HDR-HDR-CHK (4)

The total length of the Type 0 header format, including the common header, is 92 bytes.

4.2.3. Type 0 Extended Header

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

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

4.3. Type 1 Payload Format

4.3.1. Composition

Following the common header, the Type 1 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 (40 bytes). To keep packet size below 1500 bytes, HDR-NEIGHBOR-COUNT has a maximum value of 35.

4.3.2. Type 1 Format

The Type 1 header includes the following fields:

- HDR-NEIGHBOR-COUNT (1)
- HDR-PADDING (3)
- *Neighborhood Reports (40)*
- PKT-CHK (4)

The minimum length of a Type 1 packet, including the common header, is 52 bytes.

4.3.3. Type 1 Neighbor Report Format

Following the header, Type 1 packets include HDR-NEIGHBOR-COUNT neighbor reports.

A neighbor report includes the following fields:

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

The total length of a neighbor report is 40 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 1 (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 neighbor report exists for the receiving node, then it must sent its own Type 1 packet in response, including all information gathered from this packet as well.

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 PACKET-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 passes 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 1)

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

This packet is sent under three conditions:

1. Initialization

2. Reception of a PKT-BEACON that does not include the recipient’s ID
3. 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 36 neighbor reports. The expiration condition ensures that PKT-BEACON is sent at minimum every TMR-BEACON-VALUE seconds. Because a PKT-BEACON sent in response to a received PKT-BEACON includes all of the same data, it is not necessary to send a recurring PKT-BEACON if a response PKT-BEACON has already been sent within TMR-BEACON-VALUE interval; therefore, after each PKT-BEACON is sent, regardless of reason, TMR-BEACON is reset.

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 not present in ROUTING-TABLE-2:
    - Add ROUTING-TABLE-2[HDR-SRC-ID] = HDR-SRC-LOC
- Else if HDR-SRC-LOC is newer than ROUTING-TABLE-2[HDR-SRC-ID],
    - Update ROUTING-TABLE-2[HDR-SRC-ID] = HDR-SRC-LOC
- For each NEIGHBOR-REPORT:
    - If NEIGHBOR-ID is not present in ROUTING-TABLE-2:
        - Add ROUTING-TABLE-2[NEIGHBOR-ID] = NEIGHBOR-LOC
    - Else if NEIGHBOR-LOC is newer than ROUTING-TABLE-2[NEIGHBOR-ID]
        - Update ROUTING-TABLE-2[NEIGHBOR-ID] = NEIGHBOR-LOC
- If NEIGHBOR-REPORT did not include the node's own ID:
    - Create PKT-BEACON for node and send via UDP broadcast.
    - Reset TMR-BEACON.

PKT-BEACON is never forwarded.

5.2. Structures

5.2.1. PACKET-QUEUE

PACKET-QUEUE contains PKT-DATA datagrams that had HDR-FWD-DST-ID corresponding to the recipient node but HDR-DST-ID corresponding to some other node. This means that the previous sender intended this node as the next hop for retransmission.

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

The routing algorithm considers always 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.

If HDR-FWD-MODE is FWD-GREEDY, then it will also consider an XHDR component:

- XHDR-ENTERED-LOC
- XHDR-FACE-ENTERED-LOC
- XHDR-FACE-FIRST-EDGE-SRC-LOC
- XHDR-FACE-FIRST-EDGE-DST-LOC

Once the routing algorithm has determined the next hop, HDR-FWD-DST-ID is updated. In addition, HDR-FWD-MODE may also be updated 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

5.4.4. NET-EFFECTIVE-RANGE TO BE DEFINED

The maximum range that the node can transmit.

6. Functions

6.1. Packet Reception

6.1.1. Packet from ULP (FN-ULP)

- Create new PKT-DATA structure
- Set generic HDR-MAGIC, HDR-VERSION and HDR-TYPE (PKT-DATA)
- Set HDR-SRC-[ID|LOC|VEL] based on sender’s ID, location and velocity
- Set HDR-DST-[ID|LOC] based on recipient’s ID and location passed from ULP
- Set HDR-FWD-MODE to FWD-GREEDY
- Store packet from ULP within data segment and set HDR-DATA-LEN in bytes
- Set HDR-DATA-QOS as specified by ULP
- Push to PACKET-QUEUE

6.1.2. Packet from NIC (FN-NIC)

- If HDR-TYPE is PKT-BEACON:
    - If HDR-HDR-CHK fails to validate:
        - Discard PKT-BEACON
    - Else:
        - Pass PKT-BEACON to beacon handler function (FN-BEACON)
- Else if HDR-TYPE is PKT-DATA:
    - 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 PACKET-QUEUE

6.2. Packet Handlers

6.2.1. Beacon Handler (FN-BEACON)

- If HDR-SRC-ID is the same as the node's ID:
    - Discard PKT-BEACON and exit
- If HDR-SRC-LOC is not present in ROUTING-TABLE-2:
    - Add ROUTING-TABLE-2[HDR-SRC-ID] = HDR-SRC-LOC
- Else if HDR-SRC-LOC is newer than ROUTING-TABLE-2[HDR-SRC-ID]:
    - Update ROUTING-TABLE-2[HDR-SRC-ID] = HDR-SRC-LOC
- For each NEIGHBOR-REPORT:
    - If NEIGHBOR-ID is not present in ROUTING-TABLE-2:
        - Add ROUTING-TABLE-2[NEIGHBOR-ID] = NEIGHBOR-LOC
    - Else if NEIGHBOR-LOC is newer than ROUTING-TABLE-2[NEIGHBOR-ID]
        - Update ROUTING-TABLE-2[NEIGHBOR-ID] = NEIGHBOR-LOC
- If NEIGHBOR-REPORT did not include the node's own ID:
    - Create PKT-BEACON for node and send via UDP broadcast
    - Reset TMR-BEACON

6.2.2. Data Handler (FN-DATA)

- Pop PKT-DATA from PACKET-QUEUE
- If HDR-FWD-MODE is FWD-GREEDY:
    - Invoke FN-RT-GREEDY with PKT-DATA
- Else HDR-FWD-MODE is FWD-PERIMETER:
    - Invoke FN-RT-PERIMETER with PKT-DATA

6.3. Routing Algorithms

6.3.1. Greedy Forwarding (FN-RT-GREEDY)

- If HDR-FWD-MODE is FWD-PERIMETER:
    - Set HDR-FWD-MODE to FWD-GREEDY
    - Remove XHDR fields
- If ROUTING-TABLE-1 is empty:
    - Drop PKT-DATA
- Define (d_ψ, d_ϕ) from HDR-DST-LOC
- Define (s_{ID}, s_ψ, s_ϕ) from node ID and location
- Define $id = s_{ID}$
- Define $d = DIST\left((d_\psi, d_\phi), (s_\psi, s_\phi)\right)$
- Define t
- For each node (n_{ID}, n_ψ, n_ϕ) in ROUTING-TABLE-1:
    - Set $t = DIST\left((d_\psi, d_\phi), (n_\psi, n_\phi)\right)$
    - If $t < d$:
        - Set $id = n_{ID}$
        - Set $d = t$
- If $id == s_{ID}$:
    - Invoke FN-RT-PERIMETER with PKT-DATA and return
- Else:
    - Set HDR-FWD-DST-ID to id
    - Transmit PKT-DATA

6.3.2. Perimeter Forwarding (FN-RT-PERIMETER)

- Define (d_ψ, d_ϕ) from HDR-DST-LOC

- Define $(s_{ID}, s_{\psi}, s_{\phi})$ from node ID and location
- If HDR-FWD-MODE is FWD-GREEDY:
    - Add XHDR fields
    - Set XHDR-ENTERED-LOC to (s_{ψ}, s_{ϕ})
    - Set XHDR-FACE-ENTERED-LOC to (s_{ψ}, s_{ϕ})
    - Set XHDR-FACE-FIRST-EDGE-SRC to (s_{ψ}, s_{ϕ})
- Else:
    - Define (e_{ψ}, e_{ϕ}) from XHDR-ENTERED-LOC
    - If $DIST\left((d_{\psi}, d_{\phi}), (s_{\psi}, s_{\phi})\right) < DIST\left((d_{\psi}, d_{\phi}), (e_{\psi}, e_{\phi})\right)$:
        - Invoke FN-RT-GREEDY with PKT-DATA and return
- If ROUTING-TABLE-1 is empty:
    - Drop PKT-DATA
- Define $id = s_{ID}$
- Define (x_{ψ}, x_{ϕ})
- Define (f_{ψ}, f_{ϕ}) from XHDR-ENTERED-FACE-LOC
- Define $b = BEARING\left((d_{\psi}, d_{\phi}), (f_{\psi}, f_{\phi})\right)$
- Define t
- For each node $(n_{ID}, n_{\psi}, n_{\phi})$ in ROUTING-TABLE-1:
    - Set $t = BEARING\left((d_{\psi}, d_{\phi}), (n_{\psi}, n_{\phi})\right)$
    - If [$b > 0$ and $t - b > 0$] or [$b < 0$ and $t - b < 0$]:
        - Set $id = n_{ID}$
        - Set $(x_{\psi}, x_{\phi}) = (n_{\psi}, n_{\phi})$
        - Set $b = t$
- If $id == s_{ID}$:

    - Drop PKT-DATA
- Else if HDR-FWD-MODE is FWD-GREEDY:
    - Set HDR-FWD-MODE to FWD-PERIMETER
    - Set XHDR-FACE-ENTERED-DST to (x_ψ, x_ϕ)
- Else if edge between node and chosen HDR-FWD-DST intersects the edge between XHDR-ENTERED-FACE-SRC and XHDR-ENTERED-FACE-DST
    - Set XHDR-FACE-ENTERED-SRC to (s_ψ, s_ϕ)
    - Set XHDR-FACE-ENTERED-DST to (x_ψ, x_ϕ)
- Set HDR-FWD-DST-ID to id
- Transmit PKT-DATA

6.3.3. Routing Table Update (FN-RT-UPDATE)

- Construct temporary table $R[ID] \rightarrow (\psi, \phi)$
- Define the radius of the Earth r
- Define u_ψ, u_ϕ
- For each node $(n_{ID}, n_\psi, n_\phi, n_\theta, n_s, n_t)$ in ROUTING-TABLE-2 $(ID, longitude, latitude, bearing, speed, timestamp)$:
    - Set $u_\phi = \text{asin}\left(\sin(n_\phi)\cos\left(\frac{n_s n_t}{r}\right) + \cos(n_\phi)\sin\left(\frac{n_s n_t}{r}\right)\cos(n_\theta)\right)$
    - Set $u_\psi = n_\psi + \text{arctan2}\left(\cos\left(\frac{n_s n_t}{r}\right) - \sin(n_\phi)\sin(u_\phi), \sin(n_\theta)\sin\left(\frac{n_s n_t}{r}\right)\cos(n_\phi)\right)$
    - Add $R[n_{ID}] = (u_\psi, u_\phi)$
- Block FN-RT-GREEDY and FN-RT-PERIMETER from invoking
- When no FN-RT-GREEDY or FN-RT-PERIMETER is still occurring:
    - Copy R to ROUTING-TABLE-1