Workgroup: Network Working Group
Internet-Draft:
draft-schulte-amp-mesh-protocol-00
Published: 28 April 2021
Intended Status: Experimental
Expires: 30 October 2021
Authors: A. S. Schulte
Technische Universitaet Berlin

AMP Mesh Protocol

Abstract

This memo describes a decentralized multi-domain mesh networking protocol for low power embedded systems. Its decentralized architecture allows for large scale dynamic topologies across multiple wireless domains. The protocol is optimized for low power wireless devices by using zero-maintenance addressing algorithms. A decentralized ad-hoc reactive routing algorithm enables fast route convergence with low communication overhead.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on 30 October 2021.

Copyright Notice

Table of Contents

1. Introduction

1.1. Motivation

Our modern society is heavily influenced by ubiquitous embedded computing devices. Wearables, smart home, or manufacturing devices are quite useful on their own, but only live up to their full potential, when they start to communicate with other devices. Communication between embedded devices is what makes automated homes, buildings, or factories smart. Autonomous communication enables the devices to form a larger system-of-systems. Aggregation of information from the entire communication domain enables the system to be aware of its environment and react to it.

Embedded devices often communicate wireless, using technologies such as WLAN, Bluetooth, or IEEE 802.15.4 based protocols. These communication stacks are designed and optimized for specific use cases and environments. In large systems, such as smart factories, many fundamentally different device classes might be used. This can range from handbeld mobile devices to large manufacurting equipment. These have different communication requirements and therefore use different technologies. This document proposes a dedicated networking protocol for wireless embedded devices to enable efficient on-site inter-domain communication.

1.2. Scope

The AMP Mesh Protocol (AMP) is designed to facilitate the formation of a dynamic and self optimizing ad-hoc network across wireless domains. It is therefore a layer 3 networking protocol. AMP transports connection-less datagrams between individual nodes. The protocol focuses on two main features: addressing and routing.

The protocol adheres to the ISO/OSI layers and does not depend on, or use, any TCP/IP technology. Features like fragmentation, congestion control, or Quality-of-Service guarantees are not part of the protocol and left to other layers. The same is true for name resolution and node or service discovery.

1.3. Interfaces

To ensure broad compatibility, AMP demands only basic features of the data link layer. It must provide bidirectional connectivity

between neighboring nodes. There needs to be an automatically maintained link state for each connection. Each frame on that link must be able to transport 1024 bytes. Optionally, the data-link layer may provide a leader-election mechanism to be used in address distribution.

AMP provides multiple services to upper layers. Each node is assigned a unique address in the network upon boot. The protocol provides the information if a remote node with a certain address is reachable through the network, as well as it's distance as hop count. Datagrams can be sent with an optional acknowledgment.

1.4. AMP Terminology

Node: A computing device, participating in the network with an assigned address. A single physical system may incorporate multiple virtual nodes.

Domain: From AMP's perspective, a domain is any set of nodes, connected via the same data link layer. This may be for example a WLAN network, a Bluetooth Mesh, or a wired bus.

Gateway: A gateway consists of two nodes from different domains. They share a common layer 2 technology to transparently transfer messages between domains. A gateway may be a single physical device with multiple interfaces.

Layer: Layers are to be interpreted as in the ISO/OSI convention.

Address: An address is a unique identifier assigned to a node in the network. AMP defines its own address format.

Address Pool: An address pool describes a range of addresses. A pool is defined by a start address and the address count, i.e. the pool size.

Parent: In the scope of address assignment (ZAL/AQ and ZAL/DE), a parent is the node assigning address pools to a child.

Child: In the scope of address assignment (ZAL/AQ and ZAL/DE), a child is a node requesting address pools from potential parents.

Bin: In scope of distribution equalisation (ZAL/DE), a bin are the nodes within signal transmission ranges of each other, i.e. the immediate neighbors of a node.

1.5. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

2. Protocol Operation

2.1. Operating Environment

To enable the protocol to be used in a broad variety of environments, it does not require any specific topology of the used data link layers. The data link layer must provide a bidirectional link between neighboring nodes.

AMP operates on an undirected, connected, and finite graph. The preferred topology is a sparse mesh. Within a given scope, there only ever exists one network. There is no concept of subnets, partitions, or merges.

All nodes in the network are created equal. The network is fully decentralized and does not rely on the services of single nodes. There is no central coordinator. All algorithms are fully distributed.

The network is self-configuring and self-healing. Nodes may join or leave the network at runtime. Links between nodes may be formed or dropped at will. Nodes are expected to be nomadic: They may change their position, address, and connectivity in the network, but not very frequently. The routing algorithm reacts to topology changes. Addresses may be assigned and revoked at any time.

Adhering to the best-effort principle, all nodes should execute the required networking operations to their best of ability. The network relies on the cooperation of all nodes. It uses the least amount of messages, i.e. is most efficient, when all nodes implement the full standard.

2.2. Relation to other Protocols

AMP is an overlay network, specifically designed to be used atop a wide variety of communication technologies. It is not exclusively bound to use classic link-layer technologies. AMP messages may for example be transported via Ethernet, IP, or ZigBee. In the scope of this document, technologies used for data transmission are refered to as "layer 2" or "data link layer". Gateways translate between these incompatible domains. Gateways are nodes that happen to have more than one communication interface.

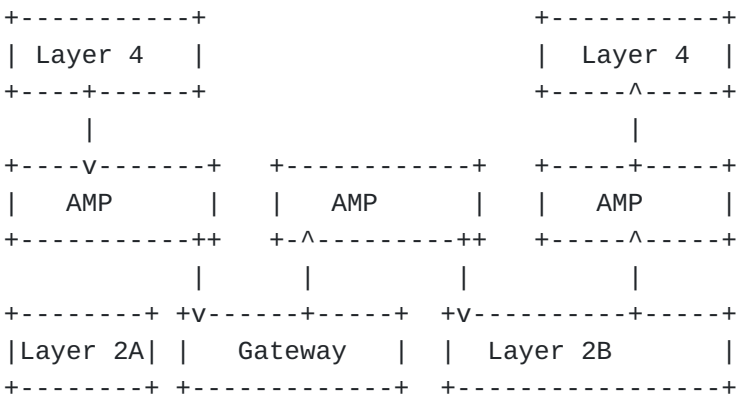

Figure 1: Layers and Gateways

Figure 1 shows a message-flow between two nodes in separate domains. The data is transferred between domains via the gateway medium.

2.3. Addressing

One of AMP's core features is addressing. This does include the address format a well as the assignment and management mechanisms. These mechanisms are fully distributed throughout the network. To minimize the number of transferred messages, the "Zero-Maintenance Address Allocation" (ZAL) algorithms by Hu and Li are applied [4]. The paper describes three algorithms to manage address space in a network: Address Allocation (ZAL/AL), Address Acquisition (ZAL/AQ), and Distribution Equalization (ZAL/DE).

Following the ZAL approach, address assignment is done in a decentralized manner, using as few messages as possible. Upon boot, a node has no address. To participate in the network a node requests the assignment of an address pool from its neighbors (ZAL/AQ). Available address pools are selected and assigned (ZAL/AL). This process continues recursively, so each node re-distributes its assigned address space to child nodes. Address pools are leased to a child as long as the link, via which the assignment was performed, is active. If the number of available addresses in a section of the network falls under the specified limit, the distribution equalization (ZAL/DE) algorithm is triggered. If none of the neighboring nodes has assignable address space, the child assigns a random temporary address with the reserved prefix to itself (see Section 3).

Address assignment starts from an an initial node, which holds the full address pool of the domain. AMP defines the recursive address assignment algorithm. It does explicitly not define how the initial node is selected. The initial node and address space should be configured manually by the network's administrator. Alternatively, the election of the initial node may be done by layer 2. Many

protocols such as ZigBee or Bluetooth Mesh have an intrinsic leader which may be used as initial node for the respective domain. The actual address pool should be defined by the administrator. It can optionally be assigned automatically by a deterministic algorithm or an auxiliary protocol. The initial address pool size per domain should be at least 2^32.

2.3.1. Address Format and Notation

AMP uses 64-bit addresses. This deliberately oversized address space allows for efficient and robust distributed address assignment. Large pool sizes result in a lower risk of address depletion. The probability of duplicated temporary addresses is also reduced greatly.

The text representation of AMP addresses follows the same specification as IPv6 addresses as defined in RFC 4291 [2]. The addresses byte values are written in hexadecimal notation and split in 4 blocks of 16 bits, separated by a colon. Addresses with leading zeros should be shortened as described in RFC 4291 [2].

All addresses are uniquely assigned in a network and used for unicast. There are no classes or scopes. With the exception of the following set, all addresses can be assigned.

0000:0000:0000:0000
: Unspecified address: Reserved for Address Acquisition.

  Is shortened to "::".

FFFF:FFFF:FFFF:FFFF
: Invalid Address. Message must be dropped immediately.

FFxx:xxxx:xxxx:xxxx
: Prefix for temporary addresses.

2.3.2. Domain Separation

Although an AMP network is always interpreted as a single coherent network, the address assignment algorithm is only ever executed within the boundaries of a domain. Address management messages must not be sent via gateways.

This domain separation adds to the robustness of address assignment and revocation: When the node that assigned an address block goes offline, the address pool and all derived child-pools become invalid. When an intermediate node in the addressing tree goes offline, the branch becomes stale and new addresses need to be assigned to the affected nodes. If the initial node, the root of the addressing tree, goes offline, all addresses within the respective

domain become stale. A new initial node needs to be selected and new pools need to be assigned throughout the entire domain. Keeping the address assignment within domains reduces the number of necessary reassignments to a confined part of the larger network.

2.3.3. Address Acquisition on Boot (ZAL/AQ)

Upon boot, a node has neither any knowledge over the network, nor an address. To participate in the network, it needs to acquire an address.

The joining node sends HELLO messages via all its available interfaces. Sender and receiver address must be unspecified (::) to indicate a ZAL/AQ request. All potential parent nodes should reply with a set of assignable address pools in a POOL_ADVERTISEMENT message. If a potential parent does not reply within a timeout period, this node should be ignored. The child must choose the advertisement with the highest count of total addresses. If two advertisements offer the same number of addresses, the child may choose between them at will.

To accept a set of advertised addresses, the child replies with a POOL_ACCEPTED message to the selected parent. This should be acknowledged by the parent with a POOL_ASSIGNED message. Only when the assignment is explicitly completed, the child must use the advertised address pools. It must not start doing so any sooner. When the POOL_ASSIGNED message is received, the child must assign the lowest of the received addresses to itself. No other address shall be used for communication. When the assignment is complete, the child should send HELLO messages to the remaining neighbors. The sender and receiver address must be populated with the repective addresses. This indicates to the other neighbors, their advertisement was not accepted. After the child has an assigned address, it should recursively advertise and assign address pools itself.

If the parent is no longer able to assign the previously advertised address pools, it replies with an empty POOL_ADVERTISEMENT message. In this case the pools must not be used by the child. The child may chose another parent or restart the ZAL/AQ algorithm with new HELLO messages.

The POOL_ADVERTISEMENT and POOL_ASSIGNED messages must contain the address of the parent as sender address. This means there is an implicit neighbor discovery mechanism built in to the ZAL/AQ mechanism. The child knows all of its neighbors, since they announced their address in the advertisement. The chosen parent knows that the child did choose the lowest of its assigned

addresses. Any remaining nodes should receive a HELLO message with the chosen address.

As discussed above, AMP does not define how the initial node and the root address pool in a domain is chosen. When bootstrapping a new domain, the ZAL/AQ algorithm is first executed between the initial node and its neighbors. The algorithm then cascades recursively throughout the entire domain. Nodes with no immediate connection to the initial node send out HELLO messages and reply with empty POOL_ADVERTISEMENT messages. When none of the neighbors replies with a populated advertisement, the child should continue sending HELLO messages periodically. The interval may optionally be increased with a back-off algorithm. This idle state is maintained until one or more neighbors acquired addresses and answer with populated advertisements. Optionally, a node can chose random address from the reserved address space to itself while waiting.

When a node receives a HELLO message with a populated sender address and an undefined receiver address from a newly established link, the ZAL/AQ algorithm must not be triggered. This message means that the neighboring node already has an address and simply announces itself. The receiving peer should answer with a fully populated HELLO message to complete the handshake.

2.3.4. Address Allocation (ZAL/AL)

The ZAL/AQ algorithm discussed above defines the communication pattern between a child and its potential parent. The address allocation algorithm (ZAL/AL) is triggered during this process in the potential parent nodes. It defines how the assignable addresses are safely managed.

The algorithm is designed to prevent address duplication. The pools are reserved before they are advertised by the parent. They must only be used by the child when the assignment is completed. If the confirmation message is lost, the reserved pools might be lost. If a link goes down, the child must no longer use the addresses assigned over this link. A parent can therefore safely reassign the pools, without the risk of address duplication.

A node holds a list of address pools, it was assigned. Each entry consists of the pool itself and its current state: {AVAILABLE | RESERVED | ASSIGNED}. Derived from this list, the total count of available addresses is known. When an assignment request arrives, half of the available space should reserved for the request. In case of an uneven count, the number must be rounded down. Reservation of addresses should start at the numerically highest available address. Counting down from this highest value, pools are reserved until the desired count is reached. If necessary, a pool in the list is split,

to fit the exact count. The state of these pools is changed from AVAILABLE to RESERVED. The reserved pools are then send in an POOL_ADVERTISEMENT message to the requesting node. If there are no addresses available, the advertisement message should be send with empty payload. Address assignment requests should be processed in the order they arrive.

If the child answered with a POOL_ACCEPTED message, the state of the address pools is changed from RESERVED to ASSIGNED. The parent completes the process by sending a POOL_ASSIGNED message. This message must only be sent after the internal state is changed. If the child answered with a HELLO message, the advertisement was rejected and the address pools can be assigned to other nodes. The state of the address pools is changed from RESERVED to AVAILABLE.

2.3.5. Address Space Rebalancing (ZAL/DE)

Even with large address pools, the available space can be depleted quickly in bad conditions. The distribution equalisation (ZAL/DE) algorithm is designed to redistribute address space throughout the domain, if a bin has less available address space than the defined target capacity.

The probability of address space depletion is relatively low, especially with large address pool sizes. With the recommended initial pool size of 2^32, the probability is negligible, especially for smaller domains. Even if the address space is depleted in a region of the domain, joining nodes use an address from the reserved pool for temporary addresses. The probability for address duplications is also negligible. The implementation of the ZAL/DE algorithm is therefore optional. The decision to omit this feature should be based on a careful evaluation of the probabilities of depletion for a given use case.

The algorithm itself as well as the equations to calculate bin target capacities and probabilities are defined in the paper by Hu and Li [4] and are not reproduced here. Only the AMP specific messages and details are described here.

The initialization of ZAL/DE is based on the target bin capacity Se. The value of Se depends on the size of the initial address pool. To eliminate the need to distribute Se throughout the domain, it is set to a fixed value, based on the recommended initial pool size of 2^32. In AMP domains, Se always set to the value 12.

As all addressing algorithms, ZAL/DE is only executed within the boundaries of a given domain. Associated messages must not be delivered via gateways.

When a bin falls under its target capacity and the ZAL/DE mechanism is triggered, the BIN_CAPACITY_REQUEST and BIN_CAPACITY_REPLY messages are used to determine the distribution of address space. To redistribute address space, the POOL_ADVERTISEMENT, POOL_ACCEPTED, and POOL_ASSIGNED messages are used as described in the ZAL/AQ algorithm. Both source and destination address must be specified at all times.

When pools are distributed horizontally through the domain, nodes must keep track from where they received address pools and to where they were assigned. If the link via which a pool was received goes offline, the addresses are no longer valid and need to be revoked. This is done by sending POOL_REVOKED messages to all children, which were assigned addresses from the now invalid pool. The detailed revocation process is described below.

2.3.6. Address Revocation

There are three ways for addresses to be revoked: Graceful, with either a GOODBYE or a POOL_REVOKED message, or ungraceful, by a link going offline. Revocation means that the revoked address pools are no longer valid and therefore not usable or assignable. Revocations must be forwarded to nodes, that were assigned the revoked addresses. Revocations therefore cascade through the network. Revocations can happen anytime, even during the ZAL/AQ process.

Graceful revocation happens with GOODBYE or POOL_REVOKED messages. If a node goes offline in a controlled manner, it must send a GOODBYE message to all its neighbors, informing them of the imminent shutdown. The receiving nodes must revoke the associated address pools and should reply with a GOODBYE_ACK message. The node which is powering down should wait for these acknowledgments. If an neighbor did not acknowledge, the GOODBYE message should be resend after a timeout period.

Receiving a POOL_REVOKED message means, an upstream node has gone offline, which invalidated the associated address pools. Child nodes which were assigned the now invalid addresses must immediately be notified with a POOL_REVOKED message.

Ungraceful shutdown or connection loss can happen at any time. Since the layer 2 infrastructure is required to maintain and provide a link state, nodes immediately know when a neighbor is no longer available. In this case, all address pools which were assigned over the now unavailable link, are invalid. This is true for pools retrieved via the ZAL/AQ algorithm at startup and redistributed pools via ZAL/DE. POOL_REVOKED messages must be send to the affected neighbors.

2.4. Routing

AMP uses a decentralized reactive ad-hoc routing algorithm, similar to AODV [3]. The distance vector routing algorithm uses the hop count as metric. A route is represented by its destination, the associated interface, the hop count and a timeout. By using a timeout, unused routes are removed after a while. Also stale routes, where the nodes are no longer active, are eventually removed. The timeout counter is started along with the creation of the route and should be reset, every time a route is used. Nodes start with zero knowledge over the network and continuously build up a routing table on demand. Routing tables are considered volatile and must not be reused when a node restarts, rejoins, or moves within a network.

The routing algorithm is designed for low communication overhead, fast convergence and passive route maintenance and optimization.

Upon boot, a node initiates the ZAL/AQ algorithm and subsequently knows its immediate neighbors. These are added to the routing table with a hop count of 1. Since the connectivity to the neighbours is monitored via the layer 2 link state, there must not be a timeout for these routes.

2.4.1. Route Detection

When a node has no route for a desired destination, it should initiate a route discovery. A ROUTE_DISCOVERY message should be send via all interfaces. The destination node should respond with a ROUTE_REPLY. Route discoveries should be flooded by all nodes, but not to the interface it was received from. A node may receive multiple replies on different interfaces. Only the route with the least hops should be kept. If multiple interfaces have the same hop count to a destination, only one should be kept. If no reply to the discovery message was received, the node should not send any data messages to the irresponsive destination address. Optionally, the hop limit of the discovery message can be set to a low count and increased in subsequent runs, to limit the range of the request and therefore the number of generated messages.

When a forwardable message is received, the routing table should always be validated against it. This allows for passive route discovery and maintenance. If source address of the message is not known yet, a route should be created. If a route is already known, but the hop count of the received message is lower than the known one, the route should be updated. If a route is used or updated, the timeout should be reset.

When a ROUTE_DISCOVERY message is received, the node should add the source address to its routing table. If the source is already

present in the routing table, the node should only reply, if the hop count of the discovery message is equal or less than in the existing route. This reduces communication overhead for nonoptimal routes.

2.4.2. Route Updates

A nodes routing table should be updated with every incoming message, regardless of the type and recipient. If there is no entry for the source-address of the message, a new route with the address, receiving interface, hop count, and a timeout should be created. If there is an entry in the table, the hop count should be validated. If it is lower than the stored value, it should be updated. If the receiving interface is different to the old route, it should also be changed. When a route is used or updated, the timeout should be reset.

2.4.3. Route Removal

Routes may be deleted on three different occasions: timeout, address revocation, or a link going offline. When a route times out, it should be removed from the table, since it is no longer in active use. When address pools are revoked via a GOODBYE or POOL_REVOKED message, routes to these addresses must be removed. The subsequent assignments are no longer valid and must therefore no longer be forwarded to. When a link goes offline, all routes associated with that interface must be removed. Also routes to addresses which are actively or passively revoked by the link going down must be removed.

2.4.4. Forwarding

Of the four message categories (see Section 3), only data and route-discovery messages are forwardable. Addressing and control messages must never be forwarded.

When a message is received which is not addressed to the receiving node, the message should be forwarded. The messages hop count must be incremented. If a route is found in the local routing table, the timeout of the used route should be reset. The message is then transmitted via the link, associated with the route's interface. If there is no known route for the desired destination, the message should be flooded to all interfaces, except the one, the message was received from. The routing table should be updated before forwarding a message.

There are several measures in place to prevent loops, duplicates, and redundant messages: Route discoveries with a higher hop count than the locally stored route should be dropped. If the shortest route to a destination is associated with the same interface a message is received from, the message should be dropped to avoid

loops. A node must not initiate a route discovery for a destination of a forwardable message it received. If incrementing the hop count would exceed the hop limit, the message must be dropped.

Nodes with tight processing or memory budget may omit the routing algorithm entirely. Instead of selecting the best interface, messages may be flooded via all interfaces. Although this behaviour is legal, it is not recommended, since it produces higher message load on the network.

2.5. Datagrams

The main purpose of this protocol is the delivery of payload data between nodes. Data is transported in connectionless DATAGRAM messages. Datagrams are standalone messages, comparable to UDP. AMP does not add any further context to the message, other than the header fields needed for the network operation. A datagram can transport up to 1003 bytes of payload.

Many applications might rely on a delivery guarantee. To eliminate the need for every application to implement such a mechanism, AMP offers the ACKNOWLEDGED_DATAGRAM. This message adds an identification code to a datagram. When a node receives a ACKNOWLEDGED_DATAGRAM, it should reply with a DATAGRAM_ACK message to the sender. This acknowledges the successful transfer of the datagram, by including the received identification code. The transaction is uniquely identified by the combination of the source address, the destination address, and the identification code.

The 16-bit identification code allows for up to 65536 in-flight messages per node pair. The sender must keep track of the used identification codes and ensure their uniqueness.

AMP does provide the messages for acknowledgment, but no redelivery or timeout algorithms. The implementation of these mechanisms is left to upper layers or the application itself.

2.6. Gateways

Gateways are a core component of AMP, enabling inter-domain communication. Gateways consist of two nodes in different domains, connected by a common interface.

In the scope of addressing, a gateway link is special and needs to be treated with care. Gateway nodes must be aware of the fact they are a gateway. A gateway link is not part of a domain, addressing messages must therefore not be transmitted via this link. Gateway links must only be used after a node has acquired a valid address. In the scope of routing and forwarding, gateway links are treated as every other link in the network graph.

Gateway links may be formed and dropped at any time. To initiate a gateway, a node sends a HELLO message with its source address and unspecified destination address. If the second node replies with a fully populated HELLO message, the neighbor discovery is complete and the gateway is operational. A gateway is terminated, when the link goes offline or one of the nodes sends a GOODBYE message.

Gateways may consist of two separate nodes in two domains, connected by a common physical interface. Alternatively a single physical node with interfaces to multiple domains may behave as soft gateway. Data can be transferred internally, instead of a common interface. In any other regard, soft gateways must behave as if they were physically separate nodes.

There may be multiple gateways between two domains. This is recommended, since gateways can be a potential bottle-neck and single-point-of failure. More gateways between domains result in a more robust network.

3. Message Specification

Sets of transmittes bits in the AMP Mesh Protocol are called messages. There are four message classes for different features of the protocol. These message classes, the message header and common features are specified here.

The network byte ordering is big-endian. Messages must not be longer than 1024 bytes, including the header. If necessary, zeroes should be added as padding.

3.1. Message Header

All messages have a common header. It contains the minimal data set needed for the operation of the network. The three header fields required by all messages are the message type, the source address, and destination address. Some message types may add additional header fields and payload.

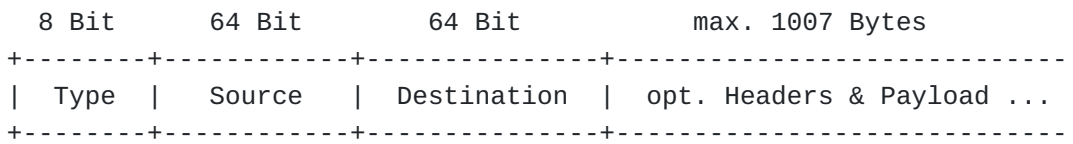

Figure 2: Message Header

Type:

Unsigned 8-bit integer

Specifies the message type

Source:
Unsigned 64-bit integer

Source address of the message

Destination:
Unsigned 64-bit integer

Destination address of the message

Payload:
Up to 1007 bytes

Optional auxiliary headers and payload

The first 8 bit define the messages type. The notation for message types is hexadecimal. The first nibble indicates the message class, the second nibble defines the type. This structure allows for future additions. The class distinction in the first nibble using hexadecimal letters also increases readability for humans.

After the message type, source and destination address are given as 64-bit unsigned integers. If not explicitly stated otherwise in the message specification, both header fields must always be populated with valid addresses.

Not all messages add a payload. Some messages, such as HELLO, do not need more data than type and addresses. Detailed specifications for all messages are given below.

A distinction is to be made between forwardable and non-forwardable messages. Addressing and control messages must never be forwarded. They are only ever exchanged between neighbors. Address and control messages with other addresses than the two peer's or the unspecified (::) address must be dropped immediately. Data and routing messages may be forwarded. The header must always be fully populated. If it contains an invalid or unspecified address, the message must be dropped immediately.

Forwardable messages add hop counter and hop limit fields. The hop counter must start with zero and must be incremented on every hop. If incrementing the hop count would exceed the hop limit, the message must be dropped.

3.2. Addressing Messages

Addressing messages must only be exchanged between neighboring nodes. Addressing messages must not be forwarded. Source and destination address must only contain the unspecified or the peers' addresses.

Addressing messages must never be exchanged over a gateway link. Addressing messages are only valid within the same domain.

Addressing messages are identified by a hexadecimal "A" in the high nibble of the message type.

3.2.1. POOL_ADVERTISEMENT

This message is used in ZAL/AQ and ZAL/DE algorithms. It advertises assignable address pools to a neighboring node.

The advertisement adds a list of assignable address pools to the payload of the message. First, the count of message pools is given, then the pools are listed. Each pool is defined by the starting address and the size of the pool. Restricted by the message size, up to 62 address pools can be added. If there are no assignable pools available, the message should be send with an empty payload.

If used during the ZAL/AQ algorithm, the destination address must be unspecified. In all other cases, address fields must be populated.

The POOL_ADVERTISEMENT adds the following fields to the message:

Message type 0xA1

Count:
Unsigned 8-bit integer

Gives the number of message pools

1 to 62 message pools, each consisting of:

Address pool start address:
Unsigned 64-bit integer

Address pool size:
Unsigned 64-bit integer

3.2.2. POOL_ACCEPTED

This message is used in ZAL/AQ and ZAL/DE algorithms. It indicates that a node accepted the address pool advertisement of a parent. No additional header fields or payload are added to the message.

If used during the ZAL/AQ algorithm, the source address must be unspecified. In all other cases, address fields must be populated.

Message type 0xA2

3.2.3. POOL_ASSIGNED

This message is used in ZAL/AQ and ZAL/DE algorithms. It indicates that address pools are assigned to a child node.

Adds a list of the assigned address pools to the payload of the message. This must be the same set of pools as in the original advertisement. First, the count of message pools is given, then the pools are listed. Each pool is defined by the starting address and the size of the pool. Restricted by the message size, up to 62 address pools can be added.

If used during the ZAL/AQ algorithm, the destination address must be unspecified. In all other cases, address fields must be populated.

Message type 0xA3

Count:
Unsigned 8-bit integer

Gives the number of message pools

1 to 62 message pools, each consisting of:

Address pool start address:
Unsigned 64-bit integer

Address pool size:
Unsigned 64-bit integer

3.2.4. POOL_REVOKED

Indicates that a set of address pools is no longer valid and routable. Their usage for routing and communication must immediately be stopped.

Adds a list of the revoked address pools to the payload of the message. First, the count of message pools is given, then the pools are listed. Each pool is defined by the starting address and the size of the pool. Restricted by the message size, up to 62 address pools can be added.

Message type 0xA5

Count:
Unsigned 8-bit integer

Gives the number of message pools

1 to 62 message pools, each consisting of:

Address pool start address:
Unsigned 64-bit integer

Address pool size:
Unsigned 64-bit integer

3.2.5. BIN_CAPACITY_REQUEST

This message is used in the ZAL/DE algorithm. It is used to determine the available address space in the local bin.

This message type does not add any payload.

Message type 0xA5

3.2.6. BIN_CAPACITY_REPLY

This message is used in the ZAL/DE algorithm. It is a reply to the BIN_CAPACITY_REQUEST. The message adds the amount of available address space as payload.

Message type 0xA6

Bin capacity:
Unsigned 64-bit integer

3.3. Control Messages

Control messages must not be forwarded. There is no payload in control messages.

Control messages are identified by a hexadecimal "C" in the high nibble of the message type.

3.3.1. HELLO

This message is used by to announce the existence and address of a node to its neighbors. If the source address is empty, the ZAL/AQ algorithm is initiated.

This message type does not add any payload.

Message type 0xC1

3.3.2. GOODBYE

This message is used to indicate the graceful shutdown of a node or the termination of a link.

This message type does not add any payload.

Message type 0xC2

3.3.3. GOODBYE_ACK

This message is used as acknowledgement for a GOODBYE message, so the retiring node knows that its GOODBYE message was received and processed.

This message type does not add any payload.

Message type 0xC3

3.4. Data Messages

Data is transported through the network as datagrams. Data messages are forwardable. They add hop counter and hop limit header fields.

Data messages are identified by a hexadecimal "D" in the high nibble of the message type.

3.4.1. DATAGRAM

This message is used to transport data through the network.

Message type 0xD1

Hop count:
Unsigned 8-bit integer

Hop limit:
Unsigned 8-bit integer

Payload length:
Unsigned 16-bit integer

Gives the length of the payload in bytes

Payload:
Up to 1003 bytes of payload

3.4.2. ACKNOWLEDGED_DATAGRAM

This message adds an identification code to a datagram. This enables the receiver of the message to acknowledge the successful transfer of the message to the sender. The message can be uniquely identified by the combination of source address, destination address, and identification code.

Message type 0xD2

Hop count:
Unsigned 8-bit integer

Hop limit:
Unsigned 8-bit integer

Identification Code:
Unsigned 16-bit integer

Used to uniquely identify the message

Payload length:
Unsigned 16-bit integer

Gives the length of the payload in bytes

Payload:
Up to 1001 bytes of payload

3.4.3. DATAGRAM_ACK

This message is used to acknowledge, that a ACKNOWLEDGED_DATAGRAM was successfully received. Although this message is in the class of data messages, it does not add any payload other than the header fields listed below.

Message type 0xD3

Hop count:
Unsigned 8-bit integer

Hop limit:
Unsigned 8-bit integer

Identification Code:
Unsigned 16-bit integer

Used to uniquely identify the message

3.5. Routing Messages

Routing messages are used to discover routes between two nodes in the network. Routing messages are forwardable, but do not transport any payload other than hop counter and hop limit header fields.

Routing messages are identified by a hexadecimal "F" (for "find") in the high nibble of the message type.

3.5.1. ROUTE_DISCOVERY

This message is used to initiate a route discovery.

Message type 0xF1

Hop count:
Unsigned 8-bit integer

Hop limit:
Unsigned 8-bit integer

3.5.2. ROUTE_REPLY

This message is the reply to a ROUTE_DISCOVERY. The hop limit must be set equal the hop counter of the received ROUTE_DISCOVERY message.

Message type 0xF2

Hop count:
Unsigned 8-bit integer

Hop limit:
Unsigned 8-bit integer

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

To keep wide compatibility with low power devices, the protocol does not have any built-in security features. The protocol is therefore vulnerable to malicious nodes. Both the addressing and the routing algorithm can be interfered with by modified or malicious messages. AMP is to be considered inherently insecure.

5.1. Out-of-Scope Attacks

Data integrity is left to the underlying layers. There are no encryption or authentication features. If needed, they must be added by higher layers. Without added security by other layers in the communication stack, AMP is susceptible for eavesdropping, replay, message insertion, deletion, modification, and man-in-the-middle attacks.

5.2. Denial of Service Attacks

Both direct and distributed denial of service attacks are possible. A node can force its direct neighbours to invest memory and processing resources by sending large datagrams with malicious header fields. This can be invalid addresses or a hop count/hop limit which require the message to be dropped. To prevent this attack, the attacked node can simply drop the connection to the malicious neighbor. This is fully compliant with the best-effort principle. The routing algorithm will adapt to the change in topology.

A distributed denial of service attack can be executed by forging the source address of a ROUTE_REQUEST. When a malicious node sends route requests to multiple nodes in the network, they all send responses to the node with the forged source address. This attack is somewhat dampened by the routing algorithm. ROUTE_REQUEST messages with a non-ideal route are dropped. A successful DDoS attack therefore requires inferred knowledge about the networks topology.

The hop limit field can be forged by malicious nodes. If it is set to a higher value than intended by the sender, this can result in network congestion. This is especially true for ROUTE_DISCOVERY messages, which are selectively flooded. This attack is confined to the boundaries of a domain.

5.3. Attacks on the Addressing Algorithm

The addressing algorithms can be interfered with, by provoking duplicate addresses. A malicious node can advertise address pools, which it was not officially assigned. Alternatively, the same pool can be assigned to multiple nodes.

Address revocations can also be malicious. Therefore messages must only be processed, if they are received over the link the addresses were originally assigned over. This contains the impact of a misbehaving node to a single branch of the addressing tree.

5.4. Attacks on the Routing Algorithm

Manipulated hop count header fields can interfere with the routing algorithm. Off-path attackers can direct selected message flow towards them by decrementing the hop count, which enables MITM attacks.

Maliciously incremented hop counts can lead to route diversions. Traffic can be diverted to other parts of the network, which can result in higher overall network load and lead to congestion. This attack requires at least some knowledge over the networks topology.

6. References

6.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[2] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[3] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <https://www.rfc-editor.org/info/rfc3561>.

6.2. Informative References

[4] Hu, Z. H. and B. L. Li, "ZAL: Zero-Maintenance Address Allocation in Mobile Wireless Ad Hoc Networks", March 2005, <https://doi.org/10.1109/ICDCS.2005.87>.

Author's Address

Aljoscha Schulte
Technische Universitaet Berlin

Email: a.schulte@tu-berlin.de