Packets

This chapter outlines the encoding/format of various packets.

- MAC/PHY Packet Structure
- Beacon Packet Structure

MAC/PHY Packet Structure

All messages sent over the air take the form of a packet. Packets encapsulate the actual user data payload with information required for the network to operate. They are similar to their 802.15.4 counterparts, but extended to allow larger payloads.

This page specifically describes the lowest level PHY and MAC packet headers, which encapsulate the actual BlazeNet protocol data.

Headers

These go at the start of the packet, before the user payload. Going from outermost to innermost header:

Synchronization

While this is not technically part of the data, it's used by the underlying radio subsystem to recognize the frame. The sync header is the very first part of the packet to be transmitted, and is always the same. It's first a 32-bit preamble sequence (generated and specified by the radio PHY) followed by a fixed synchronization word – this is specified by the network, with a default value of `0xF6`.

PHY

A single byte PHY header is added to the start of the packet. This consists of a the entire 8 bits as a length counter for the rest of the packet; a packet length of up to 255 is permitted.

Byte 0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Length Counter							

MAC

Next, a variable length header is used to indicate the source and destination address of the packet, as well as some additional flags and information about encryption, if used. At a minimum, the header will be 6 bytes in length, and contains the following information:

0	**1**	**2**	**3**	**4**	**5**

Flags	Sequence number (tag)	Source Address	Destination Address

Source and destination address are short addresses, which are assigned to the nodes when they associate to the network. These addresses may change between associations, but are guaranteed to be constant for a node as long as it's associated to the network. (Higher level protocols should implement a form of address resolution to convert more convenient, permanent logical addresses to these network addresses, similar to IPv4 ARP.)

The sequence number is a monotonically increasing counter. It is used to identify the packet for acknowledgement, and may be used as an input to security schemes (but may NOT be used as a key.) Devices can use this to drop duplicate packets, though they must pay attention to correctly implement counter roll-over.

Flags

7	6	5	4	3	2	1	0
Reserved (must be set to 0)	Fragment	Endpoint			Ack request	Data pending	Security Enabled

When "ack request" is set, the recipient of the message should generate an acknowledgement response, and send it to the source of this message, once it's been processed. Data pending is set if the source device has additional buffered packets for the recipient. Security enabled indicates the MAC header is followed by a security header, containing information on how to authenticate and decrypt the packet. Packets can be larger than an underlying PHY maximum packet size.

Lastly, the endpoint value specifies how the payload of the packet is to be interpreted. The following values are assigned:

- 0b000: Network control (used for network management, such as association requests)
- 0b001: Acknowledgement response
- 0b010: User data

All other endpoint values are reserved, and packets containing such values will be discarded.

Security

If the MAC header indicates the packet has security, this variable length (5 byte minimum) security header will be added immediately after the MAC header, but before the payload. The first part of the header indicates the type of security scheme used, as well as a unique nonce for the message. (TODO: define nonce overflow handling better)

0	1	2	3	4

Security Type	Frame Counter

The frame counter is a unique monotonic counter, used as a nonce for message authentication, encryption and anti-replay protection.

Security type is an enumeration that defines if there are any additional security header data, as well as the crypto scheme in use to protect the message. The following values are defined; messages containing any other security type will be discarded by the protocol layer:

- 0x00: No encryption or security, only anti-replay via frame counter
- 0x01: AES-CCM-128
  - Authentication + encryption (hardware accelerated)
  - Packet is followed with a 16-byte MAC trailer
- 0x02: AES-CTR-128
  - Encryption only (hardware accelerated)
  - No additional header data (existing frame counter is used)
- 0x03: ChaCha20-Poly1305
  - Authentication + encryption (hardware accelerated)
  - Packet followed with a 16-byte tag trailer (for authentication)

Key Source

For all encryption types that perform encryption, the security header is followed up with a key header. This identifies the key index, and an optional key source value.

Byte 0								Byte 1-4
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Has Source?	Key Index							Key Source

This structure will always have the first byte. If the key index is in the "default" realm (that is, the keys negotiated during association via key exchange) the "has source" flag will be cleared, and the key index corresponds directly to such a "well known" key.

Otherwise, the 4 byte key source field follows the key index to qualify the key index. (Currently, the only implemented values are a 16-bit node id, with the high 16 bits set to all 0's. Additionally, a value of all 1's indicates that a network shared key will be used.)

Fragmentation

If the “fragmented” bit in the MAC header flag field is set, the packet is transmitted in multiple pieces over the air. This is useful to emulate larger minimum packet sizes (such as a 1280 byte MTU for IPv6) for upper protocol layers without any extra work by simply passing large packets to

the network stack.

All such fragments have an additional byte header:

7	6	5	4	3	2	1	0
Reserved (must be 0)				Fragment Index			

The fragment index specifies the offset into the actual packet data, that is, the offset of the first byte of this packet’s payload in the logical packet’s payload buffer. The last packet in a fragmented transmission is indicated by a payload length smaller than the PHY maximum.

Footers

Additional information needed to validate packets goes after the payload data.

Checksum

Contains a 16-bit checksum over all headers (starting with the PHY header) and payload data.

This checksum is to be a 16-bit CRC, using the CCITT polynomial with a seed of `0xFFFF`. CRCs are calculated LSB first, and output most significant byte (and bit in the byte) first.

The radio layer shall discard all received packets where the checksum footer's value does not match the computed packet checksum. Packet headers shall not be interpreted until the checksum has been validated, as they may have been corrupted otherwise.

Beacon Packet Structure

This page describes the format of beacon frames.

Fixed Part

Every beacon frame starts out with these fields:

Optional Part

More optional data can be specified as part of the beacon. These optional fields are each specified as tag/length/value tuples. Both the tag and length are a byte; the length does not include these two bytes of a header, but only the payload. Zero length values are allowed.

Client devices should ignore any tags that it doesn’t understand how to decode.

Buffered Traffic Map

- Type: `0x01`
- Length: 2 bytes min

If one or more periodic devices have buffered traffic pending in the coordinator, it will insert a buffered traffic map (BTM) optional tag into the beacon. The BTM has as its first (and only mandatory) value a 16-bit value N that contains the periodic device id of the first (lowest) device with buffered traffic.

If there are any more devices with pending traffic, the message will be larger than this 16-bit quantity; the rest of the message is interpreted as a bitmap, where bit 0 of byte 0 corresponds to device N+1. If a bit is set, that device has pending messages.