BANO protocol notes

texane@gmail.com

1 Overview

The BANO protocol has been designed to implement a wireless network where a resourceful device known as the *base* interacts with smaller devices known as the *nodes*. The protocol especially targets domotics applications, but is not limited to them.

In a typical architecture, the base centralizes node information. It reports node information to the user, and schedules appropriate actions based on the user configuration.

To ease understanding, this document focuses on the common case where base and node roles are clearly distinct. However, there is no strong limitation that prevents a node to act as a base, and a base to act as a node. Also, the protocol does not limit the base count.

The BANO protocol relies on 2 messages used between bases and nodes for accessing $(key, value)$ pairs:

- the *SET* message is used to set a value given a specific key,
- the *GET* message is used to get a value given a specific key.

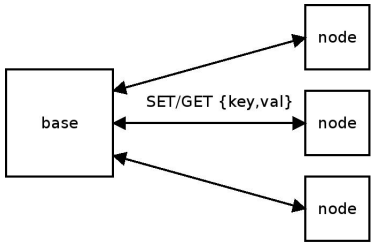

Figure 1: key value exchange

These 2 messages are used to implement the following operations:

- *synchronous SET* with optional acknowledgement: a base sets a node value using a *SET* message. If required, an acknowledgement can be sent back as a flagged message,
- *synchronous GET*: a base gets a node value by sending a *GET* message, and waits for the corresponding reply,
- *asynchronous SET*: a base gets node key value pairs by listening for incoming *SET* messages.

2 Transport layer

BANO is designed for wireless networks. The hardware (RF chipsets) and logic (low level protocol) used to transport messages are refer to as the *transport layer*. The following points were considered for choosing a transport layer:

- packet based, with at least 16 bytes data payloads,
- packet ordering is not required,
- packet acknowledgement is not required,
- packet routing is not required,
- if provided by hardware, addressing must be at least 4 bytes. Otherwise, addressing is implement in software and the payload size must be increased by 4 bytes,
- if implemented by hardware, the CRC must be at least 2 bytes. Otherwise, the CRC can be implemented in software.

After investigations, the following 2 chipsets from Nordic Semiconductor were selected:

- NRF905 (covers sub GHz bands),
- NRF24L01P (covers 2.4GHz band).

3 Messages

3.1 Message format

A message consists of a 128 bits sequence (16 bytes). Any field larger than 1 byte uses the little endian encoding. A header is always present and its format is common to all the messages. The payload contents vary according to the operation.

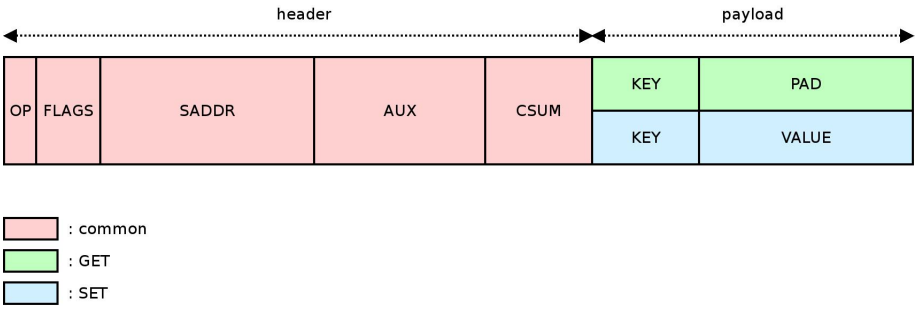

Figure 2: message format

- *OP*, 2 bits
  One of the following message operation:
  - BANO_OP_GET (0x0): get a value given a key,
  - BANO_OP_SET (0x1): set a value given a key.
- *FLAGS*, 6 bits
  A combination of the following flags:
  - BANO_FLAG_REPLY (1 << 0): this message replies or ack a previous operation,
  - BANO_FLAG_ERR (1 << 1): error during operation execution.
- *SADDR*, 32 bits
  The source address (sender node or base).
- *AUX*, 24 bits
  Auxiliary data. Usage of this field is left to the application.
- *CSUM*, 16 bits
  2 bytes checksum covering all the message bytes. During computation, the CSUM field is considered zero. The use of a checksum is optional.
- *KEY*, 16 bits
  The message target key.
- *VALUE*, 32 bits
  The message target value. The protocol does not specify a format for values. For instance, it can be used to transmit integer or real values.
- *PAD*, 32 bits
  Padding. Contents are undefined.

NOTE: Currently, the destination address is handled by the transport layer, and is not part of the message format. On the contrary, the source address is not handled by the transport layer and is included in the message.

NOTE: if a checksum is already computed by the transport layer hardware, the CSUM is not used.

3.2 Message integrity

TODO

3.3 Message acknowledgement

TODO

3.4 Message forwarding

TODO repeater

4 Addressing

Bases and nodes are identified using 32 bits *addresses*. Bases use a known static address. Node addresses are randomly generated at programming time, along with a random 32 bits *seed*. If required, the following mechanism is proposed to detect address collision.

At any time, the base can detect by checking the uniqueness of nodes $(addr, seed)$ pairs. To retrieve $(addr, seed)$ pairs, the base sends a message to all the nodes that it has discovered so far:

Nodes reply with the following message:

Address collision is detected if one or more identical $(addr, seed)$ pairs are received. The base resolves the conflict and sends new addresses to the nodes:

5 Security

Implementing security adds complexity not required in all cases and goes against some other BANO protocol design choices (stateless, common case, short message size...). However, it is undeniable that there are node messages that requiring some or all of the following security features:

- hiding message contents,
- preventing replay attack,
- preventing brute force,
- preventing tampering attacks,
- detecting node flood.

Each point is addressed in the following sections.

5.1 Hiding message contents

Symetric block ciphers can be used to hide message contents. The cipher block size should comply with the BANO message size, ie. 16 bytes. Thus, a 128 bits cipher is chosen (AES 128). Also, encryption is enabled for all the messages of a given node. If this is a limitation, a node key can allow the node to switch switch the encryption mode. In this case, and because of statelessness, the encryption mode toggling should be sent in 2 encrypted and clear versions.

Ciphers rely on a secret key shared between the base and the node. There is no support for sharing the key built in the protocol. In a typical situation, the node is programmed with a random key that is made known to the base using the configuration interface.

5.2 Preventing replay attack

The usual way to prevent replay attack is for the node message to convey a seed. This seed must not be known by the attacker at the time the message is generated.

5.3 Preventing brute force

TODO

5.4 Preventing tampering attacks

TODO man in the middle ...

5.5 Detecting node flood

TODO

6 Design considerations

6.1 Focus on common case

The protocol is designed with common case in mind. Any feature that is not mandatory should not impact its design. For instance, security is a feature that would impact message format if made mandatory. Another example is the lack of message routing support.

6.2 Node simplicity

Nodes resource requirements should be as low as possible, enabling low cost 8 bits microcontroller based configurations. On the contrary, the base is considered resourceful. This should be used to lower the node requirements.

6.3 Low power consumption

TODO

6.4 Low memory footprint

TODO