CS3099 Superground C Protocol Spec

Document Created by: Paul Lancaster

November 15, 2018

Protocol Types

address_t A 32 bit unsigned integer. Used to hold the destination and source addresses within a microbit packet.

messageId_t A 32 bit unsigned integer. Used to hold the unique message Id of a message.

checkSum_t A 32 bit unsigned integer. Used to hold the hash / checksum of the message.

Message Layout

destination	source	depth	flags	messageId	bodyLength	checksum
address_t	address_t			messageId_t	bodyLength_t	checkSum_t
32 bit	32 bit	4 bit	4 bit	32 bit	16 bit	32 bit
152 bit fixed length						

Figure 1: A diagram showing the layout of the fixed length message header

destination The unique ID of the destination microbit or 0 to broadcast to every microbit.

source The unique ID of the sending microbit.

depth A limit on the number of times a message should be resent by the microbits. This should be decremented by 1 on each resend. If a message is received with a depth of 1 or less then the message should not be resent.

flags Indicates the purpose of the message.

First bit If set to 1 then the message is meant for a microbit. If not set then the message is meant for a server or other entity.

Second bit and Third Bit If neither bit is set to 1 then the message is an original / initial message. If only the second bit is set then the message is an acknowledgement response to a previous message. If the third bit is set then the message is a refusal of a previous message (regardless of the second bit).

Fourth bit Currently unused.

messageId A unique ID for this message. This in combination with the sender address should uniquely identify a message.

bodyLength The length of the body which immediately follows the header in bytes.

checksum A hash of the message body used to check that the data hasn't been corrupted. Implemented as the 32-bit version of the MurmurHash3 hash.

On Message Recieve

In-order to stop messages being re-broadcast every message contains a depth value. Each time a message is recieved by a microbit that isn't the destination microbit it should decrease this value by 1. If the resulting value is equal to 0 then the message should not be re-broadcast. If the microbit is the destination microbit then it should not rebroadcast the message. The exception to this is the case where the destination address is set to 0 (broadcast) in which case the message should be rebroadcast (with the depth still decremented by 1) as long as the depth is greater than 0.

Each message contains a messageId which in combination with the senderId must uniquely identify a message. The protocol itself makes no attempt to log previously recieved messages so a microbit may recieve the same message multiple times. In-order to prevent this causing issues each microbit should

have a way of know which messages have already been recieved so that if it is recieved again it can be ignored.

Discovery

If a message has the discovery flag set then the recieving microbit should send a message to the sender of the discovery message with the discovery and acknowledgement flags set and with the body containing the name of the microbit. This name is choosen arbitarly but a human readable descriptive name such as C2-Microbit1 is recommended. In this way a microbit can discover the names and ID's of all microbits in the mesh.

Message Id

MessageId should not be dependant on the body of a message because it should allow for the same sender to send the same message (with the same body) multiple times. Theses messages should each have a unique message Id relative to the sender which is why it cannot be dependant on the body.

Hashing

Included with each message is a hash of the body. This hash is included to validate that the body of the message hasn't become corupted. This only hashes the body and not the header and should not be used for a message Id for the reasons above. The hashing algorithm used is the 32-bit MurmurHash3 algorithm and this was choosen due to it being a very efficient / quick algorithm. This algorithm is not considered secure so should only be used for data integrity validation.

Microbit send message procedure

- 1. Send a discover request to all microbits.
- 2. Recieve a number of discovery acknowledgements and thereby discover all connected microbits and their names.

- 3. Select the ID of the microbit that you want to send a message to or send to 0 to broadcast to every microbit.
- 4. Create a unique ID for the message.
- 5. Send the message.
- 6. Message sent! Acknowledgements / TCP type communication is not implemented by default apart from for discovery messages.

References

1 https://github.com/aappleby/smhasher/wiki/MurmurHash3 (14/11/2018)