Assignment P2

1 Introduction

1.1 Background

In the second assignment we implement some basic networking functionalities. Namely, we dive into the world of wireless packet networks. The assignment assumes some underlying packet radio interfaces that supports broadcasting short messages. In our case, we use the ESP-Now protocol for simplicity, but equally well, the radio channel could be based on some other protocol(s).

1.2 High-level functionalities

What is provided: The starting point for this assignment includes the following functionalities and specifications:

1. Configuring and setting up the WiFi for ESP-Now protocol.
2. Specification for the packet format for short messages (200 bytes).
3. Sample Network Time client application.
4. Sample CRC checksum computation routine.

The provided code is elementary and by no means optimized or elegant in some parts. You are expected to study it, and encouraged to improve it, without introducing any new bugs! It might be a good idea to use `#ifdef`'s in order to be able to toggle between the original code and your changes.

Your task: The following functionalities constitute the main programming assignment:

1. Complete the missing parts of Chat client that offers both public chats and direct messaging based on the node id numbers.
2. Implement the Ping command and including the responses to ping messages sent by other devices.
3. Complete the corresponding commands in the serial user interface.

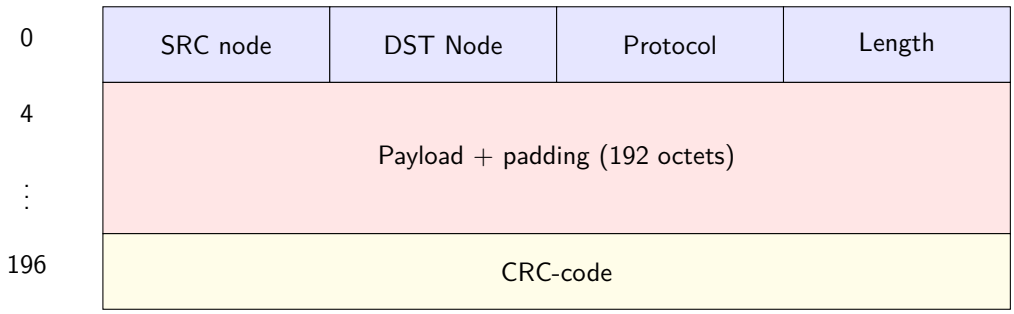
2 Specification

In this section, we specify the packet format and other necessary details for each task.

2.1 Packet format

As said, we use ESP-Now, but equally well the wireless networking could be LoRa, IP, or some mix of them. For now we assume that all devices are in close contact so that radio transmissions reach all nodes. Thus no routing functionality is needed, and the setup is essentially Layer-2 networking.

The packet format is as follows:



Note that:

- Each row corresponds to 32 bits.
- `Source` and `Destination` node id's are one byte each (`uint8_t`). For broadcast messages the destination field is set to `0xFF`.
  Note that you can acquire the node id of your device via the system call `lownet_get_device_id(void)`.
- `Protocol` defines the application. These are defined in the provided include files.
- `Length` corresponds to the length of actual data in the payload.
- `CRC` computation is based on the given 24-bit CRC generator polynomial,

$$G(x) = x^{24} + x^{10} + x^6 + x^5 + x + 1,$$

  so that the 8 MSBs are zero (32 bits reserved for CRC code in the packet).

2.2 API for lownet

The **public interface** to our networking stack is in the `lownet.h` header file. It contains the packet structure, protocol types, and utility functions such as `lownet_get_device_id(void)`.

Receive: The lownet must be initialized before the network can used. A callback function must be passed to the `lownet_init()`. This function is responsible for the asynchronous processing of the incoming packets.

Messages will be delivered via message queue, i.e., the callback function will be called whenever a packet has arrived.

Transmit: Transmission of packets is handled by `lownet_send()`. The sender needs to prepare the four header fields and the payload according to the particular protocol. If the message is a broadcast message, the destination field shall have the value `0xFF`. Otherwise the node id of the destination is specified. The source node id should be the one the node's MAC address is associated with.

The `lownet_send()` is responsible for an appropriate padding in the unused payload, if any. Then it computes the CRC for the whole packet, and sends to message in the air.

2.3 Serial console

The textual interface to interactive applications is via the serial console.

- Lines starting with `/` are commands. At this point, the UI must support the following two commands. You are allowed to implement more should you find some commands useful.
- `/ping` needs one argument that is the target node id. If the target node is `0xFF`, this will trigger a ping that all nodes in the vicinity will respond. An example:

  ```
  /ping 0xa0
  ```

  It can be useful to print out informational messages to serial UI about the outgoing and incoming ping messages.
- `/date` prints out the current time in seconds since the epoch with 10ms accuracy. The format shall be as follows:

  ```
  /date
  100000.5 sec since the course started.
  ```

  Use `lownet_get_time()` to obtain the network time in `lownet_time_t` structure that contains two fields: `seconds` and `parts`, where the unit for the latter quantitity is 1/256 second. If the network time is not available, this function returns zeroes in both fields. In this case, the command will respond as follows:

  ```
  /date
  Network time is not available.
  ```

- Lines starting with `@` are direct messages. The recipients node id should follow immediately without any white-space, and the remaining of the line is the message. An example:

  ```
  @0xab Hello!
  ```

  sends the message *"Hello!"* to the node `0xAB` (in hex).
- Any other line is considered as the public chat message, and broadcasted to all nodes (this behavior is likely to be changed in later assignments).

Note that the destination field defines if a message is meant for the given device. Either the the packet destination field must match device's node ID, or the destination field must be `0xFF` (broadcast message). In other cases, the message should be ignored by the recipient.

2.4 Chat protocol

The Chat protocol allows messaging via the serial UI.

- Source field corresponds to the node id of the device.
- Destination field is either `0xFF` or a node id, depending on the type of the chat message (to all, or to a specific node).
- Protocol field has value `LOWNET_PROTOCOL_CHAT`
- Length field defines the length of the actual message.
- Message itself is carried in the payload. One line of text without newline or a nul-terminator. Call the function `util_printable(char)` to check if a particular character is acceptabled for chat payload.

2.5 Time protocol

The network shall have a special node, "time authority", that provides *the network time*. The network time is measured in seconds and fraction of seconds since the course started, August 21st, 2023, at 6am UTC.

The network time packet is as follows:

- Source field corresponds to the node id of the device.
- Destination field is (typically) `0xFF`, i.e., this information is meant for everyone.
- Protocol field has value `LOWNET_PROTOCOL_TIME`
- Length field defines the length of the actual message (5 bytes).
- 32-bit integer for full seconds, and one byte for the fraction of seconds (1/256 seconds), similarly as in the `lownet_time_t`. The time provided is the time since the epoch started.
- The digital signature for authentication may follow the timestamp in future. Such signature is present if the length field has a value greater than 5. For now this reservation can be ignored.

2.6 Ping protocol

Ping messages are responded with "Pong" messages.

- Source field corresponds to the node id of the device.
- Ping messages are processed only if the destination field has the correct node id. If the destination field is `0xFF`, the message has been sent to everyone. Either way, the response back is unicast, i.e. the destination field shall be the node id of the origin of the ping message.
- Protocol field has value `LOWNET_PROTOCOL_PING`
- Length field has a value of 11 or more (see below).
- The payload starts with the following:
  `lownet_time_t`: time stamp of the origin (5 bytes)
  `lownet_time_t`: time stamp of the response (5 bytes)
  `uint8_t`: node id of the origin (the node that initiated the ping)
  The payload may contain additional data that the node responding to ping shall keep intact.

3 Other information

We will host in Gróska a test node with node id `0xF0` that is operational 24/7 and will broadcast the time information, respond to ping messages, and also acts as a silly bot for testing Chat functionality. The device will be near the room Alan.

You can thus visit Gróska anytime to test your code. Similar opportunity may be provided also during the next meeting.