

***Management of Telecommunication Projects***

Network Mode Standard

v1.0

**Fall 2021/22**

*MTP GROUPS A, B AND C*

Table of Contents

[**Document Versions**](#_uknxtf24f2ib) **2**

[**Introduction and Scope**](#_dtoswm8tm8fn) **2**

[**RF / Hardware specification**](#_v9k51cwh3zzt) **2**

[**Software specification**](#_v558mik6yo55) **3**

[**Packet Structure**](#_vfcioaocutyh) **7**

# Document Versions

| ***Version*** | ***Date*** | ***Comments*** | ***Responsible*** |
| --- | --- | --- | --- |
| 1.0 (Draft) | 12/11/2021 | First Draft | Standard Committee |

# Introduction and Scope

This document intends to define the specifications of a protocol for the transmission of 0.5KB of data, codified in UNICODE, from a randomly selected transceiver to a receiver passing through different intermediate nodes within 5 minutes.

To achieve the objective, the protocol and the functionality for the application described above is defined through different flow diagrams and state machines. Additionally, the structure and packets required for the transmission are detailed.

Note that the protocol described within this document is designed for having minimum three transceivers : one transmitter, one receiver and one intermediate node. Therefore, having only 2 transceivers is out of the document’s scope.

In order for the protocol to be able to tell itself that all nodes have received the file, all devices in all groups must participate. However, the protocol is robust enough to send all nodes the file regardless if one is missing.

We assume that each device counts with at least one button for transmission start and one button for transmission stop, in addition to some mechanism for indicating that it’s the first of the chain, and hence, must write the text file from the USB memory.

# RF / Hardware specification

(I) The transmission data rate is set to 250kbps, in order to improve sensitivity of the receiver

(II) The transmission power is set to -6dBm in order to meet the emission regulations.

(III) The 2.4GHz RF channel selected is the 6th one, bandwidth of 1 MHz.

(IV) The modulation used to transmit is GFSK (Gaussian Frequency-Shift Keying)

(V) The Nordic Semiconductor NRF24L0+ chip is used to assure compatibility between nodes.

# Software specification

The protocol is based on the Stop & Wait mechanism, with a token-based multiple access control to avoid collision between packets on the air. The communication always links just two nodes at the same time and on the air there is just one packet being sent.

So, the main idea is to implement a simple protocol that enables the transfer of a file through all the nodes in the network in an easy and reliable manner. The protocol is based on a typical *Stop & Wait* system mixed with a dynamic *Neighbour Discovery* method.

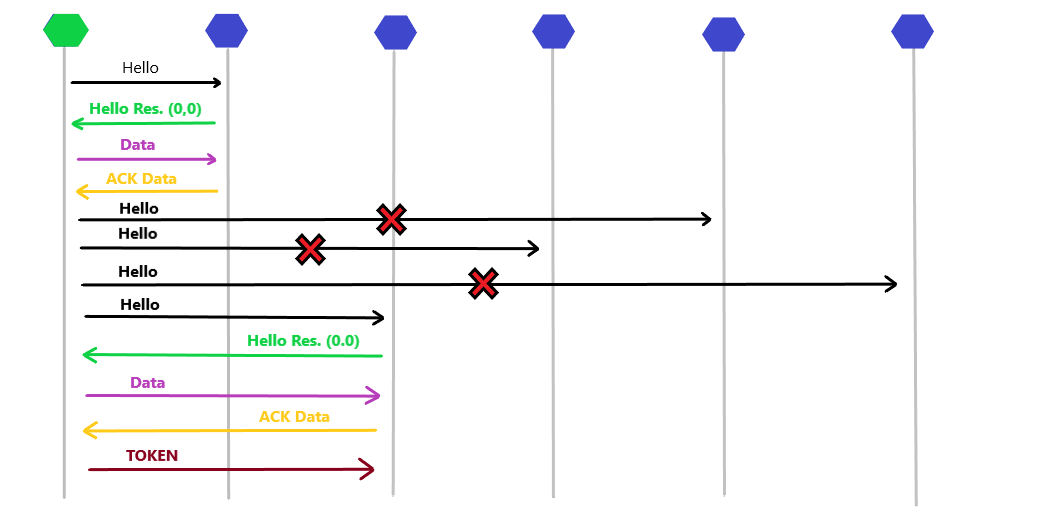
The protocol uses a token that indicates who is the user able to transmit, only the user holding the token, let it be called *token user* for the purpose of this document, can transmit at that time. Any other user is only allowed to answer to the token user with a *Hello response* or an *ACK*, either a *Data\_ACK or a Token\_ACK* depending on the circumstances. Aside from handling who is able to transmit at all times, the token also has the purpose to keep track of how many users have the file at the moment. To pass the token to another user, the message *Token* is used as described above.

One focus of the protocol is to try to send the file to as many users as possible, so the user holding the token will try to contact all the other users in the network using a *Hello* message. For every user who answers with a *Hello Response* message indicating that it does not have the file, the token user will send the file to them.

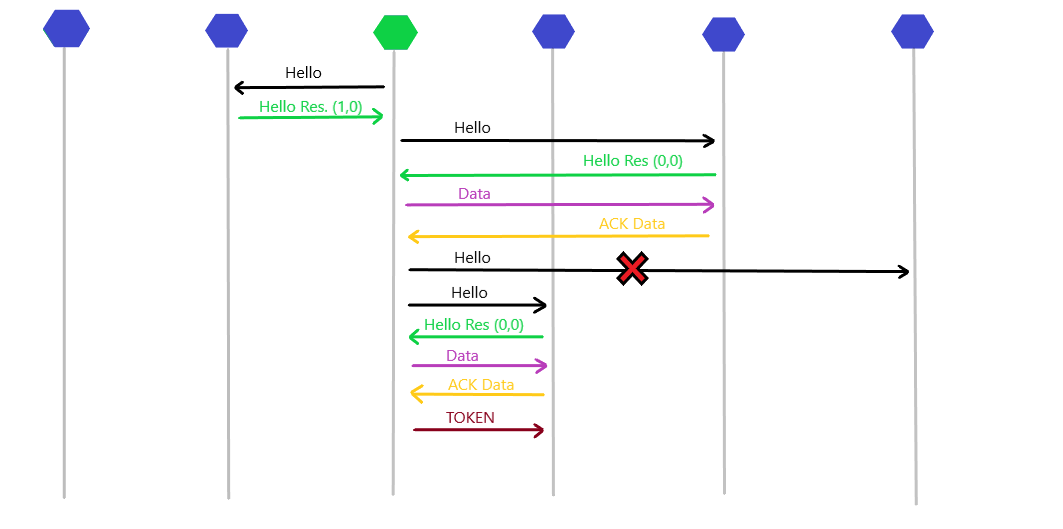
To send the messages, the token user will follow an order to avoid collisions. It will start by sending the hello message to one user, wait for an answer and then two situations might happen; it answers that it does not have the file, or it answers that it already has got the file from a previous transmission or does not answer. If it does not answer or answers it already has the file, the token user will try to contact the next user and send a *Hello* message to him. If it answers that it does not have the file it will send it to him and once finished send *Hello* to the next user. This process is repeated until the token user has tried to establish contact with all the users in the network, one by one. Finally, the token user will send the token to the last user with whom it has successfully established contact and sent the file. The token will indicate to the other user that it is his turn to transmit and also how many users already have the file.

The communication to send the file, starts when the token user receives a *Hello Response* indicating that the user does not have the file. Then the user will start transmitting the file in different *Data* packets, waiting for an *ACK* response for every packet, before sending the next one. If the packet fails, either it receives a negative response or it doesn’t receive any after an established timeout, it sends it again. The last packet will be indicated with the *End of Transmission* bit.

When a user receives the token and it indicates that six users already have the file, it will stop trying to transmit any packet and the process will finish. Otherwise, it can happen that some users for some reason have not been able to establish a connection properly with their neighbours and have been skipped in the transmission process. In this case, a user will receive the token and indicate a number different from six of the people who have the file. When that happens it will follow the previous process of trying to contact all users but it might happen that he is too far away from the missing users and does not receive any answer from them. Then the user will send the token to any user who answers and has not had the token yet, so they can try to reach them. Also, it might happen that all the answers indicate that they already have the file and already have had the token. In this situation the system will pass the token randomly to a user, so it tries again to reach the missing users. This situation will be repeated until it finally contacts the missing users and the transmission succeeds or it is ended by the five minutes timeout.



*Figure 1: Flow diagram 1*



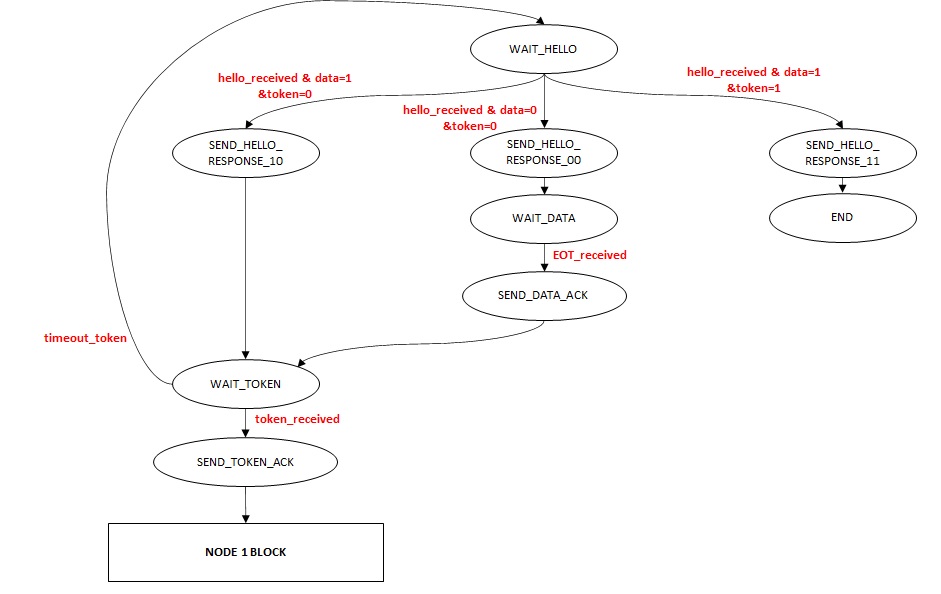
*Figure 2: Flow diagram 2*

Next it is explained the state machine diagram based on the explanation done before.The diagram for Node 1 is the following:



*Figure 3: State machine diagram Node 1*

The state machine diagram for the other nodes is shown below. Once the node has the token, it has the same behaviour as Node 1.



*Figure 4: State machine diagram other nodes*

The diagram depicts a series of retries and timeouts whose values should be defined not to surpass the 5 minutes transmission limit. The proposed values are described in the table below. In this table retry is the number of retransmissions of the same trace that can be carried out. On the other hand, timeout is the maximum ACK trace waiting time. If this time is surpassed, and the number of retries is below the maximum, a retransmission of data should occur.

| ***Parameter*** | ***Time or number*** | ***Description*** |
| --- | --- | --- |
| *Hello and Token* retries | 10 | *Hello and Token* packets are used for establishing the connection between the nodes. Since the number of destination nodes is unknown a lot of retransmissions need to be done to be sure the next node is nonexistent or not operational. |
| *Hello and Token* timeout | 1s | Since the destination node is unknown a high ACK wait time is set. |
| Data retries | 5 | Data packets have a greater number of bits than *hello* and *token* packets and, consequently, are more prone to transmission errors. However, a previous communication with the receiver device has been established and communication should be fluid. In this retry number, it has been taken into consideration that packets might not be received correctly (due to interferences or other causes). In this case, no ACK trace is received and transmission retry is going to be used. |
| Data timeout | 2s | Data packets have a significant big number of bits compared to the Token and Hello ones. Consequently, the time to reach the destination needs to be greater than the one used in the *Hello* and *Token* traces. |

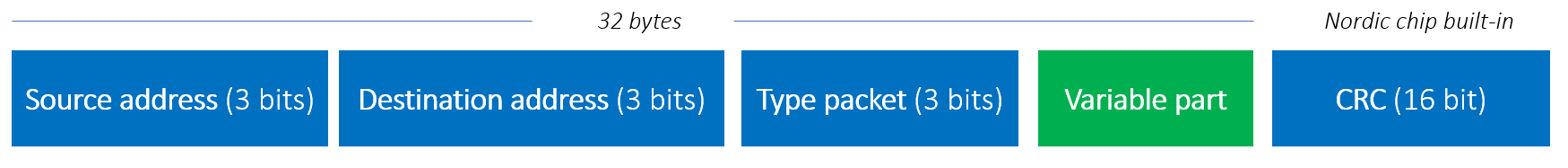
*Table 1: Retransmissions and timeouts specificaton*

# Packet Structure

The following defines the different packages used and their structure. Note that the packets marked with an S are the packets sent by the node that has the token. And the ones marked with R are the ones sent by the nodes that do not have the token in response to each of the S packets.

* *(S)* **HELLO packet:** Packet sent by the node that has the token to discover the nodes next to it.
* *(R)* **HELLO RESPONSE packet:** A packet sent by a node that has received a HELLO packet, to indicate to its transmitter that it is within range. With this package the node will also inform if it has had the token, and if it has the data.
* *(S)* **DATA packet:** Packet sent by the node that has the token containing a part of the text file to be sent. Multiple data packets will transmit the total file.
* *(R)* **DATA ACK packet:** Packet sent by the node that has received a DATA packet to inform that it has correctly (or incorrectly) received the packet.
* *(S)* **TOKEN packet:** Packet sent by the node that has the token to pass the token to another node.
* *(R)* **TOKEN ACK packet:** Packet sent by the node that has received a TOKEN packet to inform that it has correctly (or incorrectly) received the TOKEN.

Each of the previously introduced packet types has a **fixed part**, same for all types, and a **variable part,** which depends on each type. *\*Note: to simplify the protocol, packets will always be 32 bytes long, so the packet definitions below will fill the first bytes required, and the rest will be padded with zeros.*

****

*Figure 5: Complete packet structure*

The **Source address** and **Destination address**, as we have 6 nodes, they will go from 0 to 5. Each node on the network will have a fixed address within this **0-5 range**. Each of the packet’s types defined before has the following code (which is identified in the type packet field).

| ***Packet Type*** | ***Code*** |
| --- | --- |
| *HELLO* | *000* |
| *HELLO RESPONSE* | *001* |
| *DATA* | *010* |
| *DATA ACK* | *011* |
| *TOKEN* | *100* |
| *TOKEN ACK* | *101* |

*Table 2: Packet Types Codes*

The variable part for each packet type is defined below.

1. **HELLO packet**

*No variable part*

1. **HELLO RESPONSE packet**

****

*Figure 6: Hello Response variable part packet*

The **have data** field indicates if the node has the data (1) or not (0). The **had token** field indicates if the node has had the token (1) or not (0).

1. **DATA packet**

****

*Figure 7: Data variable part packet*

The **length** field indicates how many bytes is the payload actually. This is useful for the last packet, that can have a different length than the rest of the packet. The **End Of Transmission** (EoT) field, if 1 indicates the last packet. The **ID** allows identificate the packet for the Stop&Wait mechanism. The **payload** contains the actual text file data.

1. **DATA ACK packet**

****

*Figure 8: Data ACK variable part packet*

The **ID** identifies the packet that we are acknowledging for the stop & wait protocol. And **ACK** (1) indicates the packet has been received correctly, and **NACK** (0) the contrary, and is required a retransmission.

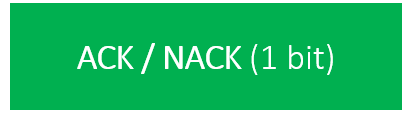
1. **TOKEN packet**

****

*Figure 9: Token variable part packet*

The **Num. Recv. Data** indicates how many nodes have received the data. If a node receives a token with this field at 6, it can stop the transmission as all the nodes have the data.

1. **TOKEN ACK packet**

****

*Figure 10: Token ACK variable part packet*

And **ACK** (1) indicates the packet has been received correctly, and **NACK** (0) the contrary, and is required a retransmission.