Standard protocol for supporting the Network Mode test of MTP course

Group A, B and C

28/12/2021

ETSETB, UPC

Managament of Telecommunications Projects (MTP, 230655)

Table of contents

1 Document versions **2**

2 Introduction and scope **3**

3 Procedure specification **4**

4 Packet structure **8**

4.1 HELLO . . . 9

4.2 HELLO RESPONSE . . . 9

4.3 DATA . . . 9

4.4 DATA ACK . . . 9

4.5 TOKEN . . . 10

4.6 TOKEN ACK . . . 10

A Error probability and timeouts study **11**

A.1 Introduction . . . 11

A.1.1 Scope . . . 11

A.2 Error probability . . . 11

A.2.1 Required number of retries . . . 11

A.2.2 Bit error probability p_{eb} . . . 12

A.2.3 Numerical example for the NM competition scenario . . . 14

A.3 Timeouts . . . 19

A.3.1 Theoretical explanation . . . 19

A.3.2 Numerical example for the NM competition scenario . . . 20

1. Document versions

Version	Date	Comments	Responsible
1.0	12/11/2021	First draft	Standard Committee
2.0	2/12/2021	Final document	Standard Committee
2.1	28/12/2021	Fixed typos, format change	Standard Committee

Table 1. Document versions

2. Introduction and scope

This document intends to define the specifications of a protocol for the transmission of 0.5KB of data, encoded 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 a flow diagram. 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. In a generic way, the protocol will be defined taking into account that there will be N nodes. The protocol described must be robust enough to send the file to all N available nodes.

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. The first device will be in charge of reading the text file from the USB memory; this will indicate it's the first device.

3. Procedure 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.

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 Network OSI layer is based on packet transmission (TCP based) whose content is based in a mix between Stop & Wait and Neighbour Discovery.

The reader can refer to Figure 1 and Figure 2 for a visual explanation of the packets to be sent and received.

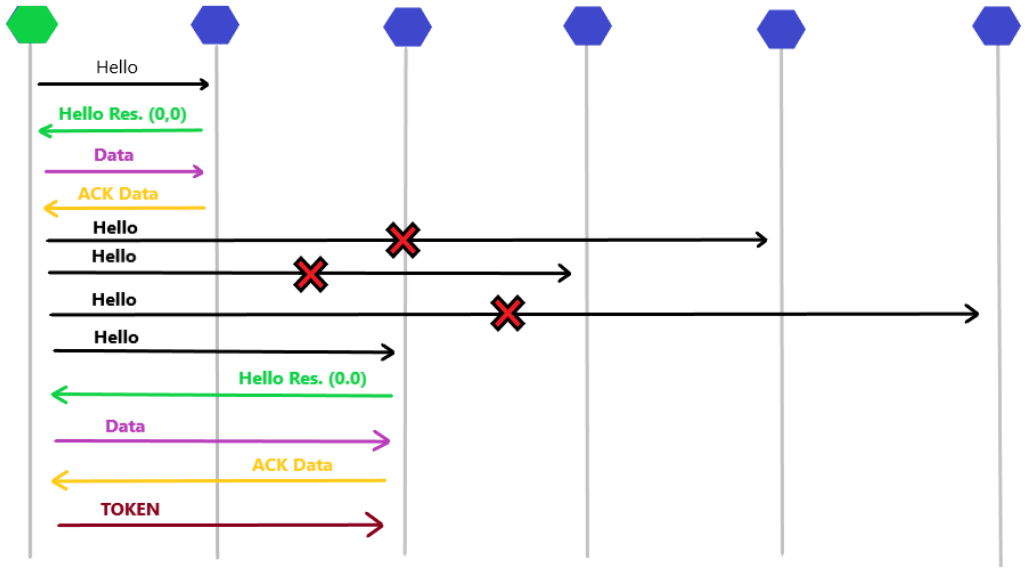

Figure 1. Flow diagram, 1

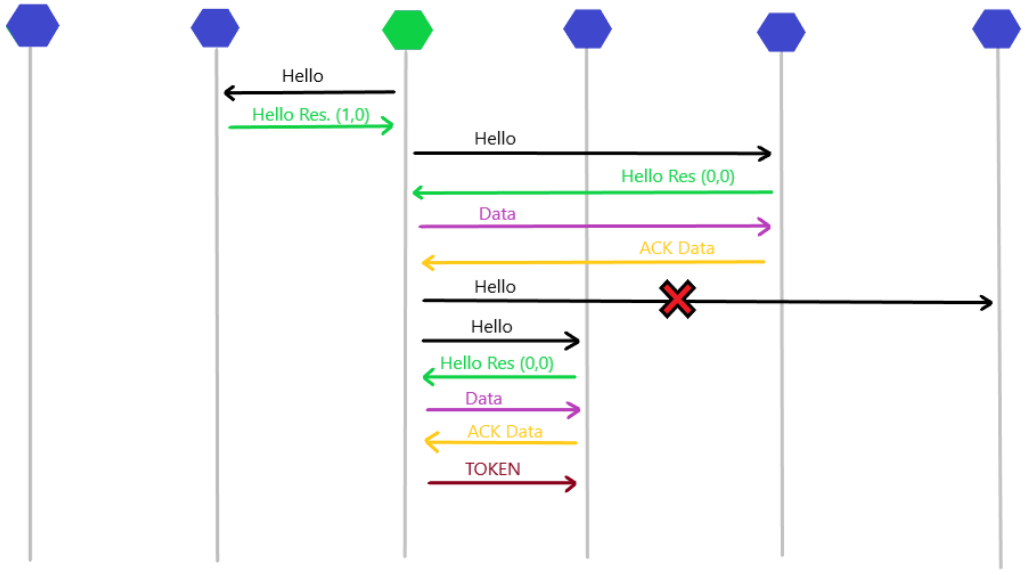

Figure 2. Flow diagram, 2

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.

One of the protocol objectives is 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 can transmit. Together with the token a message containing the IDs of the nodes who have already had the data will be sent.

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, i.e., either the transmitter receives a negative response or it doesn't receive any after an established timeout, the packet is sent again. The last packet will be indicated with the *End of Transmission* bit.

When a user receives the token and it indicates that all the expected users (or users in the network) 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 skipped the transmission process. In this case, a user will receive the token indicating that a number of users different from the expected in the network have the file. When that happens the process will start again and the token will try to contact all users again. However, it might happen that the node 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. On the other hand, 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.

The reader can check Appendix A to gain insight on the delays and retries imposed.

Figure 3 depicts a state machine for one node.

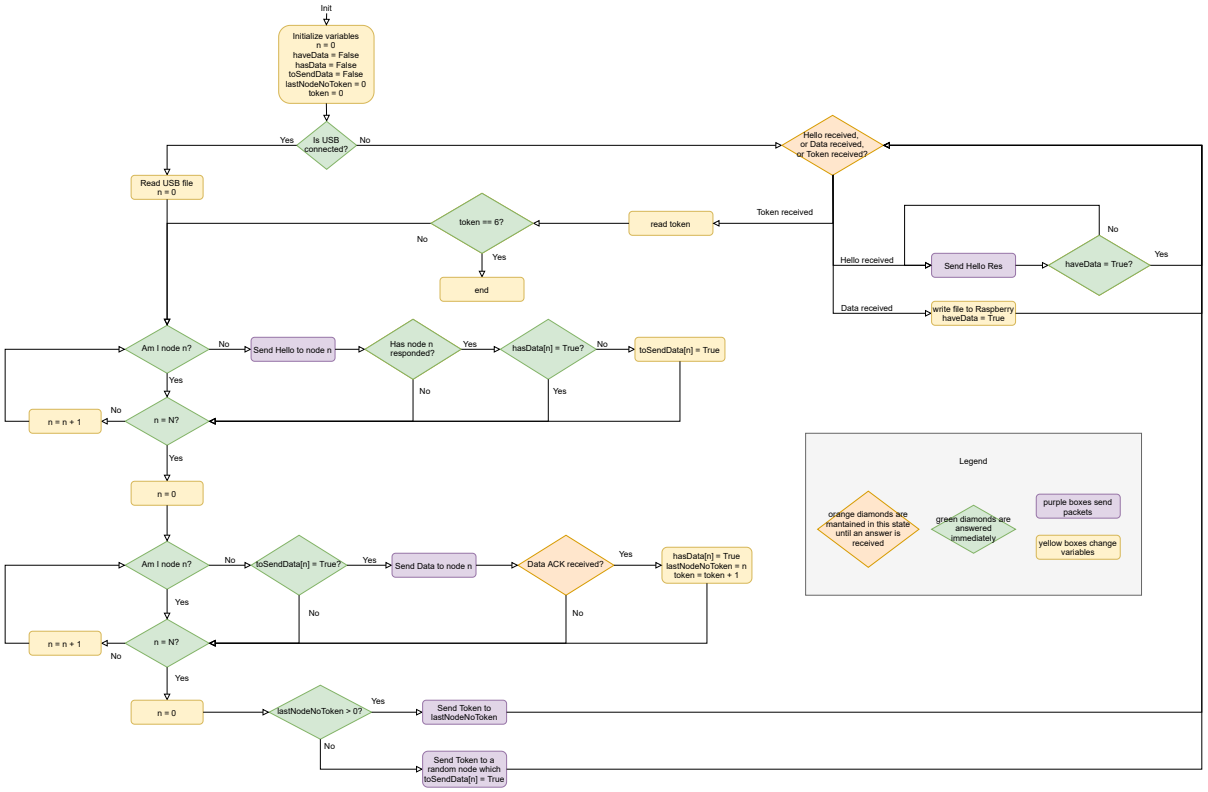

Figure 3. Flowchart

Figure 4 shows the same figure in a landscape page, to appreciate better the text and details.

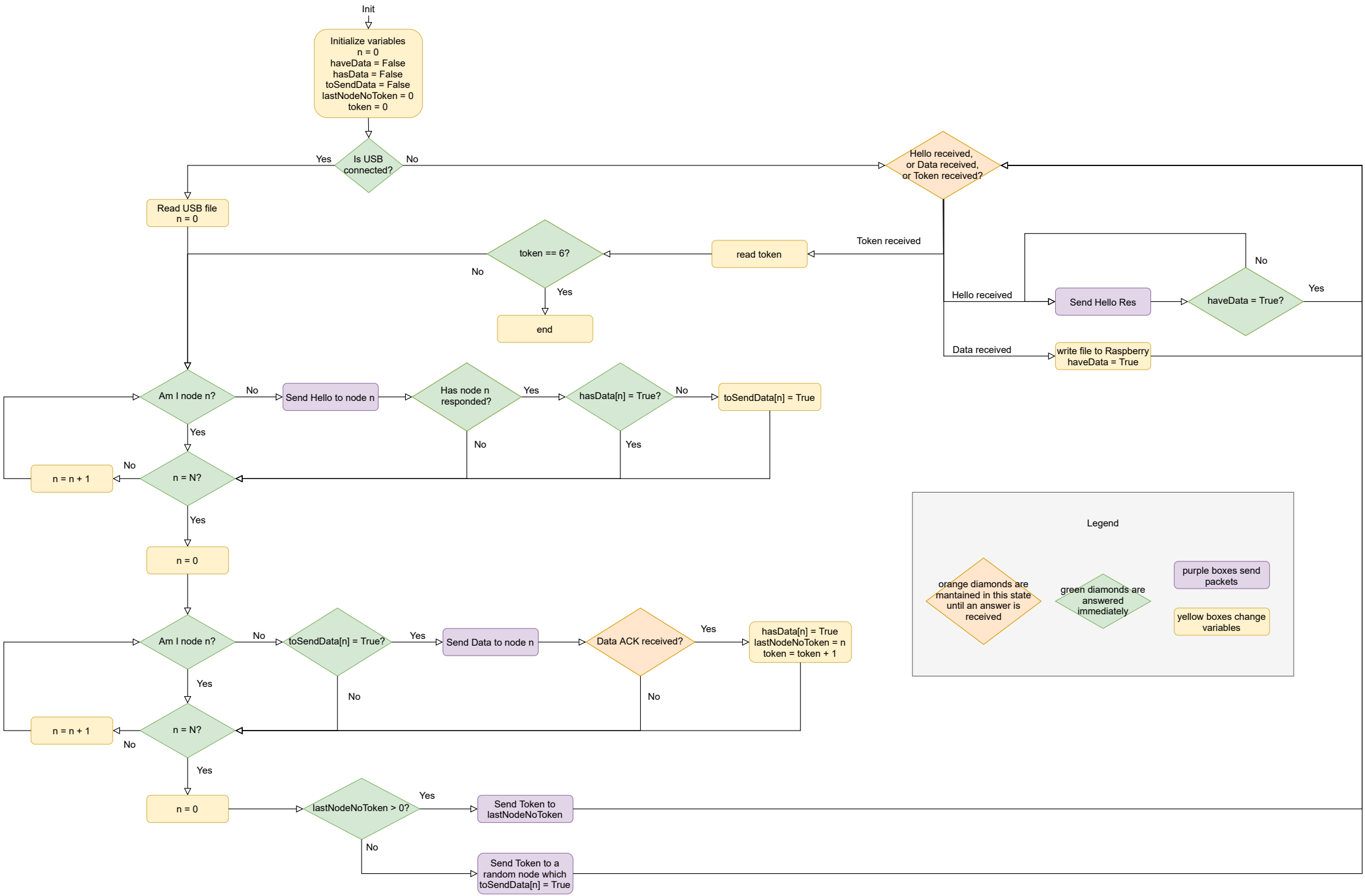

Figure 4. Flowchart, landscape view

4. 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 sent by the node that has the token to discover the nodes next to it.
- (R) *HELLO RESPONSE*: 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 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 sent by the node that has received a *DATA* packet to inform that it has correctly (or incorrectly) received the packet.
- (S) *TOKEN*: Packet sent by the node that has the token to pass the token to another node.
- (R) *TOKEN ACK*: 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, will range 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), shown in Table 3.

Packet type	Code
HELLO	000
HELLO RESPONSE	001
DATA	010
DATA ACK	011
TOKEN	100
TOKEN ACK	101

Table 2. Packet type codes

The variable part for each packet type is defined below.

4.1. HELLO

No variable part.

4.2. HELLO RESPONSE

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).

Figure 6. *HELLO RESPONSE* packet variable part

4.3. DATA

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 packets. The *End Of Transmission* (EoT) field, if 1 indicates the last packet. The *SN (Sequence Number)* allows identificate the origin of the transmission packet. Finally, the *payload* contains the actual text file data.

Figure 7. DATA packet variable part

4.4. DATA ACK

The *SeqNum (Sequence Number)* 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.

Figure 8. DATA ACK packet variable part

4.5. TOKEN

The *Num. Recv. Data* indicates the nodes that 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.

Figure 9. *TOKEN* packet variable part

4.6. TOKEN ACK

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

Figure 10. *TOKEN ACK* packet variable part

A. Error probability and timeouts study

A.1. Introduction

This annex has been written with the purpose to quantify the error probability that transcievers that implement the Network Mode standard can suffer, as well as to study the minimum timeout requirements. A proper analysis can lead to optimal values than enhance the performance without hurting the robustness of the standard.

A.1.1. Scope

This document establishes the minimum number of retries and the minimum timeout to achieve a successful Network Mode standard operation. It starts from generic explanations and equations to later focus on the Network Mode particular case. For both chapters there are, first, theoretical sections; and after a section that shows calculations for a concrete application.

This document is applicable to transcievers that implement the Network Mode standard in its totality. Numerical results given as examples are made for the Network Mode competition constraints; the reader should perform his own calculations if he brings the Network Mode standard to other scenarios.

A.2. Error probability

A.2.1. Required number of retries

The bit error rate (BER) can be understood as the expectancy for a bit, sent by a transmitter, to be flipped at the receiver side, by unit of time. To avoid time dependency this document makes use of bit error probability, defined as p_{eb}. If multiple bits are exchanged between a transmitter and a receiver, then the following holds:

$$p_{eb} = \mathrm{E}\left\{\frac{\#\text{ erroneous bits}}{\#\text{ total bits}}\right\} = \frac{\mathrm{E}\{\#\text{ erroneous bits}\}}{\#\text{ total bits}} \ . \tag{1}$$

The Network Mode standard defines how packets of bits are exchanged between nodes. Here, a packet is considered erroneous if at least one of its bits is incorrect. Then, the probability for a packet to have at least one error can be understood as a particular binomial distribution case in which the number of trials is equal to the number of trials in which the bit hasn't been flipped.

$$P(x)|_{x=n} = \binom{n}{x} p^x \left(1-p\right)^{n-x} = p^x \ . \tag{2}$$

Where p is the event probability, n the number of trials and x the number of trials for which the event occurs.

The probability for a packet to differ, at least by 1 bit, with respect to the one at the transmitter

side is defined as p_{ep}. For N bits in a packet, with the bit error probability being p_{eb}, p_{ep} is defined as 1 minus the probability of having N out of N bits correct.

$$p_{ep} = 1 - (1 - p_{eb})^N \ . \tag{3}$$

It's already conceived in the standard that some received packets may contain errors, and thus the packet can be resent, i.e., a retry is made from the transmitter side. The probability to fail after T tries is named p_{fail} and follows

$$\begin{aligned} p_{\text{fail}} &= (p_{ep})^T \\ p_{\text{fail}} &= \left(1 - (1 - p_{eb})^N\right)^T \ . \end{aligned} \tag{4}$$

It's a goal of the standard to achieve a robust communication, which means the probability of failing at sending a certain packet of N bits after T tries should be lower than a desired threshold.

$$p_{\text{fail}} = \left(1 - (1 - p_{eb})^N\right)^T < p_{\text{fail threshold}} \ . \tag{5}$$

The number of packet bits N is fixed by the defined packets, and p_{eb} can be derived after link budget calculations. The fail probability $p_{\text{fail threshold}}$ is set to an arbitrary value. The unknown is the number of tries required, T, which can only take integer values equal or higher than 1. Because the number inside the parenthesis must always be lower than 1, T must respect a minimum value.

$$T > \frac{\ln\left(p_{\text{fail threshold}}\right)}{\ln\left(1 - (1 - p_{eb})^N\right)} \ . \tag{6}$$

The number of retries is just

$$R = T - 1 \ . \tag{7}$$

A.2.2. Bit error probability p_{eb}

RF modules manufacturers typically provide the expected bit error probability (usually named as BER) for a single received power. To solve (6) for T, link budget calculations are required, and the calculated received power signal at the receiver end should be used to determine p_{eb}.

Complex equations could be used to express $p_{eb}\,(E_b/N_0)$, where E_b is the bit energy and N_0 is two times the noise added by the channel; but this job has already been done by Luque, Morón, and Casilari [1]. In the paper, the authors study a Bluetooth system, which works with Gaussian frequency-shift keying (GFSK). The Network Mode proposed standard uses GFSK. The bit error probability depends on the modulation index h. The p_{eb} is plotted against the E_b/N_0 (dB) ratio in Figure 11.

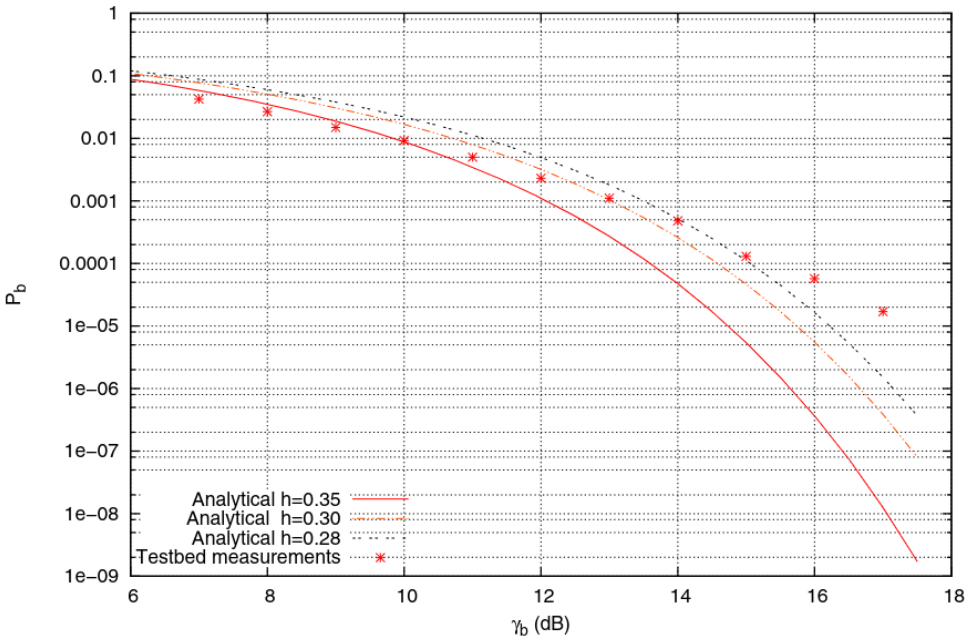

Figure 11. p_{eb} as a function of E_b/N_0 (dB), from Luque, Morón, and Casilari [1]

The standard doesn't specify a concrete h value. The testbed measurements will be used for future calculations because it's the worst case for large E_b/N_0 values and shows quite a linear trend in the logarithmic axes.

The number of symbols of a codification, named M, follows

$$M = 2^{R_b/BW} \ , \tag{8}$$

where BW is the bandwidth and R_b the bandwidth efficiency. In GFSK there are only two symbols possible: 0 or 1. This means $R_b = BW$.

The signal-to-noise ratio SNR at the output of the receiver demodulator is defined in the paper as

$$SNR = \frac{E_b}{N_0 \cdot BW \cdot T_b} \ , \tag{9}$$

where $T_b = \frac{1}{R_b}$ is the bit transmission period. Substituting (8) leads to

$$SNR = \frac{E_b}{N_0} \ , \tag{10}$$

for the particular GFSK case. By definition the receiver sensitivity is the power level for which a certain received signal leads to a certain BER, and this means

$$SNR = \frac{E_b}{N_0}\,[\text{dB}] = P_{RX}\,[\text{dBm}] - (P_{RX,\,\text{sensitvity}}\,[\text{dBm}] - P_{\text{offset}}\,[\text{dBm}]) \ , \tag{11}$$

where P_{offset} is the power difference between the noise floor and the sensitivity that has been specified for a certain BER smaller than 0.5.

In order to find P_{RX} link budget calculations are necessary. Assuming quite and ideal case with

line of sight (LOS) and no ground reflections,

$$P_{RX}\,[\text{dBm}] = P_{TX}\,[\text{dBm}] + G_{\text{antenna TX}}\,[\text{dB}] - 20\log\left(\frac{4\pi d}{\lambda}\right)[\text{dB}] + G_{\text{antenna RX}}\,[\text{dB}] - G_{\text{losses}}\,[\text{dB}] \quad (12)$$

Where d is the distance between antennas, and the gain that antennas can offer and extra possible losses has been considered.

Using the previous definitions (6) can be solved. As a summary, P_{RX} should be calculated from the link budget, and $P_{RX,\,\text{sensititivy}}$ can be either calculated from LNA noise figure (NF) and input thermal noise, or taken from the receiver manufacturer specifications. Then, if P_{offset} is known it's straightforward to get $\frac{E_b}{N_0}$. Next, a plot such as the one in Figure 11 can be used to get the bit error probability, p_{eb}. Finally, this value is substituted in (6) together with the N bits of the packet and an arbitrary $p_{\text{fail threshold}}$,

A.2.3. Numerical example for the NM competition scenario

The purpose of this section is to provide some application-dependant orientative numbers to solve for T.

DATA packet, neighbour distance

First, for the link budget it's taken into account that nRF24 output power is set to $-6\,\text{dBm}$. The nRF24l01 comes, often, soldered in a board together with a 20 dB power amplifier (PA), leading to a transmitted power of 14 dBm. Moreover, it contains a low noise amplifier (LNA) that provides 10 dB. The sensitivities given by the manufacturer are referenced to the LNA output, meaning that at the LNA input they are 10 dB lower.

Both the transmitter and the receiver antennas show the same gain: $G_{\text{antenna RX}} = G_{\text{antenna TX}} = 2\,\text{dB}$. G_{losses} is set to 12 dB to account for the ground reflection effect and the fact that, after some measurements, it was seen that the output power was a few dB shy of the one specified by the manufacturer.

It's assumed 3 transcievers will be placed at the 260 m Campus Nord side, meaning the worst case distance between transcievers is $d = 130\,\text{m}$. Up to 4 devices are expected on the short side, with the corner device being taken into account. The central frequency of operation must be between 2.4 GHz and 2.525 GHz. Channel 27 was chosen because of interference reasons, meaning the central frequency is 2.427 GHz.

With all the considered values (12) is solved.

$$\begin{aligned} P_{RX}\,[\text{dBm}] &= 14\,\text{dBm} + 2\,\text{dB} - 20\log\left(\frac{4\pi 130\,\text{m}}{\frac{3\times 10^{8}\,\text{m}\,\text{s}^{-1}}{2.427\,\text{GHz}}}\right)[\text{dB}] + 2\,\text{dB} - 12\,\text{dB} \\ P_{RX} &= -76.42\,\text{dBm} \end{aligned} \quad . \tag{13}$$

For the nRF24, at 250kbps the sensitivity that achieves $p_{eb} = 1 \times 10^{-3}$ is $-94\,\text{dBm} - 10\,\text{dBm} = -104\,\text{dBm}$. The difference between the received power at the LNA input and the mentioned sensitivity is $-76.42\,\text{dBm} - (-104\,\text{dBm}) = 27.58\,\text{dB}$.

Now, from Figure 11 and testbed measurements it's derived that to get $p_{eb} = 1 \times 10^{-3}$, $\frac{E_b}{N_0} = 13\,\text{dB}$. If there's a 27.58 dB difference between the received power and the sensitivity for which $p_{eb} = 1 \times 10^{-3}$, then p_{eb} is obtained by looking at $\frac{E_b}{N_0} \approx 40\,\text{dB}$.

The testbed measurements plot is quite linear in a logarithmic scale, with an approximate slope of ≈ 3.5 decades for every 10 dB increase in $\frac{E_b}{N_0}$. Thus, in this numeric case $p_{eb} \approx 1 \times 10^{-13}$.

The DATA packet defined in the protocol requires 30 bytes plus 7 bits. This payload is fitted inside the nRF24 packet. This means a preamble of 1 byte, an address of up to 5 bytes, a packet control field of 9 bits and a CRC of up to 2 bytes are also required. This adds up to

$$N = (1 + 5 + 2 + 30)\ \text{bytes} \times \frac{8\ \text{bits}}{1\ \text{byte}} + 9\ \text{bits} + 7\ \text{bits} = 320\ \text{bits}\ . \tag{14}$$

Substituting the N number of bits and the bit error probability p_{eb} into (3) leads to the following p_{ep} packet error:

$$\begin{aligned} p_{ep} &= 1 - (1 - p_{eb})^{N} \\ p_{ep} &= 1 - \left(1 - 1 \times 10^{-13}\right)^{320} \\ p_{ep} &= 3.20 \times 10^{-11} \end{aligned} \quad . \tag{15}$$

A failure threshold of $p_{\text{fail threshold}} = 1 \times 10^{-16}$ is imposed and T is found using (6).

$$\begin{aligned} T &> \frac{\ln\left(p_{\text{fail threshold}}\right)}{\ln\left(1 - (1 - p_{eb})^{N}\right)} \\ T &> \frac{\ln\left(1 \times 10^{-16}\right)}{\ln\left(1 - \left(1 - 1 \times 10^{-13}\right)^{320}\right)} \\ T &> 1.525 \\ T &\to 2 \end{aligned} \quad . \tag{16}$$

This means that the number of retries is just

$$R = 1\ , \tag{17}$$

due to the low probability of getting a packet wrong. The fact that the 250kbps datarate is set means the sensitivity is very low and leads to a very robust communication for the distances the application requires, according to theoretical calculations.

DATA packet, maximum distance

The previous case considered the distance between transmitter and receiver was just $d = 130\,\text{m}$. Although this is expected to be the most common case, the low sensitivity makes it possible to send data at higher distances. Here, $d = 260\,\text{m}$, which is the maximum line of sight (LOS) distance expected, is taken.

From (12), it's enough to calculate the received power decrease because of the distance increase.

$$\begin{aligned} \Delta P_{RX} &= -20\log\left(d_{\text{current}}\right) + 20\log\left(d_{\text{previous}}\right) \\ \Delta P_{RX} &= -20\log\left(260\,\text{m}\right) + 20\log\left(130\,\text{m}\right) \\ \Delta P_{RX} &= -6.02\,\text{dB} \end{aligned} \quad . \tag{18}$$

Thus,

$$\begin{aligned} P_{RX} &= P_{RX,\,\text{previous}} + \Delta P_{RX} \\ P_{RX} &= -76.42\,\text{dBm} - 6.02\,\text{dB} \\ P_{RX} &= -82.44\,\text{dBm} \end{aligned} \quad . \tag{19}$$

In this case, the difference of the received power with respect to the senstivity level is $-82.44\,\text{dBm} - (-104\,\text{dBm}) = 21.56\,\text{dB}$. Adding up the $13\,\text{dB}$, the Figure 11 must be analyzed at $34.56\,\text{dB}$. In this case, by extrapolating the testbed measurements plot a safe value to take is

$$p_{eb} = 1 \times 10^{-11} \; . \tag{20}$$

Considering the $N = 320$ bits of the DATA packet, the packet error probability is:

$$\begin{aligned} p_{ep} &= 1 - \left(1 - p_{eb}\right)^{N} \\ p_{ep} &= 1 - \left(1 - 1 \times 10^{-11}\right)^{320} \\ p_{ep} &= 3.2 \times 10^{-9} \end{aligned} \quad . \tag{21}$$

The difference with the previous case is around 4 orders of magnitude. Again, a failure threshold of $p_{\text{fail threshold}} = 1 \times 10^{-16}$ is imposed and T is found using (6).

$$\begin{aligned} T &> \frac{\ln\left(p_{\text{fail threshold}}\right)}{\ln\left(1 - \left(1 - p_{eb}\right)^{N}\right)} \\ T &> \frac{\ln\left(1 \times 10^{-16}\right)}{\ln\left(1 - \left(1 - 1 \times 10^{-11}\right)^{320}\right)} \\ T &> 1.88 \\ T &\to 3 \end{aligned} \quad . \tag{22}$$

Just to be safe, as the result is close to its ceiling integer, the result has been approximated to 3 tries. This means that the number of retries must be

$$R = 2 \; . \tag{23}$$

Thanks to the exponential dependence of p_{ep} on the number of tries T the 4 orders of magnitude

between the two cases don't require a tries difference.

TOKEN packet, neighbour distance

The two previous cases detailed in subsections A.2.3 and A.2.3 refer to the DATA packet, which is the largest packet specified by the standard. However, the TOKEN packet should also be considered. As mentioned in the standard, a transmitter needs to make sure if a node is available of not. The TOKEN packet contains a variable part of 3 bits, larger than the 2 bits of the HELLO RESPONSE packet. The TOKEN packet is analyzed because it's the largest of the two.

Thanks to previous calculations the P_{RX} power are already known. Changes appear, now, at the packet number of bits, N.

$$N = (1 + 5 + 2)\ \text{bytes} \times \frac{8\ \text{bits}}{1\ \text{byte}} + 9\ \text{bits} + 3\ \text{bits} = 67\ \text{bits}\ . \tag{24}$$

For the neighbour distance

$$P_{RX} = -76.42\,\text{dBm}\ . \tag{25}$$

Recall this lead to

$$p_{eb} = 1 \times 10^{-13}\ . \tag{26}$$

Considering the $N = 67$ bits of the TOKEN packet, the packet error probability is:

$$\begin{aligned} p_{ep} &= 1 - (1 - p_{eb})^N \\ p_{ep} &= 1 - \left(1 - 1 \times 10^{-13}\right)^{67}\ . \\ p_{ep} &= 6.70 \times 10^{-12} \end{aligned} \tag{27}$$

In this case, the packet failure threshold is decreased by two orders of magnitude with respect to the DATA packet, i.e., $p_{\text{fail threshold}} = 1 \times 10^{-18}$. This number is arbitrary, but the authors want to emphasize the importance of sending the TOKEN packet properly. (6) is solved for T.

$$\begin{aligned} T &> \frac{\ln\left(p_{\text{fail threshold}}\right)}{\ln\left(1 - (1 - p_{eb})^N\right)} \\ T &> \frac{\ln\left(1 \times 10^{-18}\right)}{\ln\left(1 - \left(1 - 1 \times 10^{-13}\right)^{67}\right)}\ . \\ T &> 1.610 \\ T &\to 2 \end{aligned} \tag{28}$$

On one hand, the threshold has been decreased; on the other the number of bits N is lower. This means that the number of retries must be just 1 for the neighbour distance.

$$R = 1\ . \tag{29}$$

TOKEN packet, 3 nodes distance

The main difference with respect to subsection A.2.3 is that now $P_{RX} = -82.44\,\text{dBm}$, which means

$$p_{eb} = 1 \times 10^{-11} \ . \tag{30}$$

Recall the defined TOKEN packet contains $N = 67$ bits. Then,

$$\begin{aligned} p_{ep} &= 1 - (1 - p_{eb})^N \\ p_{ep} &= 1 - \left(1 - 1 \times 10^{-11}\right)^{67} \ . \\ p_{ep} &= 6.70 \times 10^{-10} \end{aligned} \tag{31}$$

There's been a 2 orders of magnitude increase in p_{ep} with respect to subsection A.2.3. Again, the packed failure threshold is set to $p_{\text{fail threshold}} = 1 \times 10^{-18}$. (6) is solved for T.

$$\begin{aligned} T &> \frac{\ln\left(p_{\text{fail threshold}}\right)}{\ln\left(1 - (1 - p_{eb})^N\right)} \\ T &> \frac{\ln\left(1 \times 10^{-18}\right)}{\ln\left(1 - \left(1 - 1 \times 10^{-11}\right)^{67}\right)} \ . \\ T &> 1.962 \\ T &\to 3 \end{aligned} \tag{32}$$

On one hand, the threshold has been decreased; on the other the number of bits N is lower. Again, because the value is very close to its ceiling integer the number of tries has been made 3.

$$R = 2 \ . \tag{33}$$

Summary

The obtained results are summarized below:

Packet	Distance (m)	$p_{\text{fail threshold}}$	Number of tries T	Number of retries R
DATA	130 m	1×10^{-16}	2	1
DATA	260 m	1×10^{-16}	3	2
TOKEN	130 m	1×10^{-18}	2	1
TOKEN	260 m	1×10^{-18}	3	2

Table 3. Number of tries and retries, summary

In conclusion, it's been justified that for both the DATA and TOKEN packets a total of $T = 3$ tries, i.e., $R = 2$ retries are required, to meet the set thresholds for the 260 m distance.

A.3. Timeouts

A.3.1. Theoretical explanation

As important as the number of tries and retries are, it's also critical to determine the timeouts required for a transmitter to switch to a receiver and wait for acknowledgements (ACKs) from potential receivers that switch to transmitters after they have received a package.

It should be made clear that the timeout is defined here as the maximum loop time in which a transmitter will be stuck waiting for an acknowledgement signal. Too small, and there won't be time to receive the ACK. Too large, and the transmitter will waste time waiting for no signal. This means there's an optimal timeout. It's the aim of this chapter to find it.

Moreover, it can be noticed from the given definition that the timeout is independent of the package length that has been sent. It can, however, be made dependant on the ACK packet length.

The timeout at the receiver side must meet the following equation.

$$\begin{aligned} t_{\text{timeout}} &\geq t_{\text{air}} + t_{\mu\text{C, RX}} + t_{\text{RTAD}} + t_{\text{RX}} + t_{\text{air}} + t_{\mu\text{C, TX}} + t_{\text{RTAD}} \\ t_{\text{timeout}} &\geq 2t_{\text{air}} + t_{\text{RX}} + (t_{\mu\text{C, RX}} + t_{\mu\text{C, TX}}) + 2t_{\text{RTAD}} \end{aligned} . \tag{34}$$

Where t_{air} is the amount of time the last bit sent by the transmitter TX is received by the receiver TX, t_{TX} is the duration of time the receiver requires to send the ACK packet, $t_{\mu\text{C, RX}}$ and $t_{\mu\text{C, TX}}$ are the required times at the RF modules microcontrollers to interpret the received packet and take a decision from there, and t_{RTAD} is the radio turn around delay, which is the time a transciever requires to switch from transmitter to receiver or viceversa.

The packets are sent at the speed of light, meaning t_{air} is

$$t_{\text{air}} = \frac{d}{c} . \tag{35}$$

Where d is the distance between transmitter and receiver, and c is the speed of light $c \approx 3 \times 10^8 \,\text{m}\,\text{s}^{-1}$.

The time required by the receiver to send the ACK packet, t_{RX}, is

$$t_{\text{RX}} = \frac{\text{Packet size}\,[\text{bits}]}{\text{Datarate}\left[\frac{\text{bits}}{\text{s}}\right]} . \tag{36}$$

The time spent spent at the RF module microcontroller, $t_{\mu\text{C}}$, is proportional to the clock periods to fetch, decode and execute the required instructions to process the message and take a decision. This depends on the integrated circuit internal operation and clock frequency.

The t_{RTAD} time is typically specified by the RF module manufacturer.

A.3.2. Numerical example for the NM competition scenario

Once the timeout equation (34) has been specified, numerical values can be substituted to calculate the timeout required for a real application.

The maximum expected line of sight communication distance during the NM competition is $d = 260\,\mathrm{m}$. Thus,

$$\begin{aligned} t_{\mathrm{air}} &= \frac{d}{c} \\ t_{\mathrm{air}} &= \frac{260\,\mathrm{m}}{3 \times 10^8\,\mathrm{m\,s^{-1}}} \\ t_{\mathrm{air}} &= 86.67\,\mathrm{ns} \end{aligned} \quad . \tag{37}$$

The time required to transmit an ACK package depends on its length. Although all packages defined in the standard have very similar lengths, the largest one is considered here. It's the DATA ACK packet, that shares the length with the HELLO RESPONSE packet. Both have 2 bits of information plus the fixed bytes and bits of the nRF24.

$$N = (1 + 5 + 2)\ \text{bytes} \times \frac{8\ \text{bits}}{1\ \text{byte}} + 9\ \text{bits} + 2\ \text{bits} = 66\ \text{bits}\ . \tag{38}$$

The standard specifies Datarate = 250 kbps.

$$t_{\mathrm{RX}} = \frac{66\,[\text{bits}]}{250 \times 10^3\left[\frac{\text{bits}}{\text{s}}\right]} = 264\,\mu\mathrm{s}\ . \tag{39}$$

The $t_{\mu\mathrm{C}}$ times are not specified by Nordic, the nRF24 manufacturer, and one may suspect these times are be already included in t_{RTAD}. In case of doubt, the following assumption is made:

$$t_{\mu\mathrm{C,\,RX}} = t_{\mu\mathrm{C,\,TX}} = t_{\mu\mathrm{RTAD}}\ . \tag{40}$$

According to the nRF24 specifications

$$t_{\mathrm{RTAD}} = 130\,\mu\mathrm{s}\ . \tag{41}$$

Finally, numerical values are plugged into (34).

$$\begin{aligned} t_{\mathrm{timeout}} &\geq 2t_{\mathrm{air}} + t_{\mathrm{RX}} + (t_{\mu\mathrm{C,\,RX}} + t_{\mu\mathrm{C,\,TX}}) + 2t_{\mathrm{RTAD}} \\ t_{\mathrm{timeout}} &\geq 2 \times 86.67\,\mathrm{ns} + 264\,\mu\mathrm{s} + 2 \times 130\,\mu\mathrm{s} + 2 \times 130\,\mu\mathrm{s} \\ t_{\mathrm{timeout}} &\geq 784\,\mu\mathrm{s} \end{aligned}\ . \tag{42}$$

This is the minimum time required between two transmissions at the transmitter side. There's a high chance the nRF24 already implements some of the timeout sumands. Anyways, the communication should work fine if the above timeout is imposed to wait for all the possible responses.

Bibliography

[1] J. R. Luque, M. J. Morón, and E. Casilari, "Analytical and empirical evaluation of the impact of gaussian noise on the modulations employed by bluetooth enhanced data rates," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, Mar. 2012. DOI: 10.1186/1687-1499-2012-94. [Online]. Available: https://doi.org/10.1186/1687-1499-2012-94.