Standard for the Network Competition

Network team from groups A,B,C,D

Abstract: The aim of the document is to set and define a protocol standard shared with all the teams so as to achieve a transmission of a text file of 0.5 kB through a network composed by 8 devices (2 devices per team).

1. Introduction

1.1. Assumptions

- The first device will have a USB connected with a file before switching on the devices whilst the last one will not.
- All teams will participate in the competition. However, the protocol is designed to handle the circumstance under which a team is not able to participate.
- The session will end when a node receives all *Reply_NO* and the token has passed through all the nodes that have received the data, or if the 8 nodes have received the data although they have not received the token..
- First device should work.
- The file to read from the USB should be .txt.

2. Hardware Requirements

2.1. RF constraints

- All devices should work in the same frequency band and channel.
- Transmitted power should be less than 20dBm.
- Devices will work at a rate of 1 Mbps in order to discover neighbours as fast as possible.

2.2. Hardware requirements

Each device should have implemented a start button and a led to control the information exchanged. This button will be in charge of telling it should start the execution thread. The first node will detect that a usb is connected, and therefore understand that it is the first node in the chain.

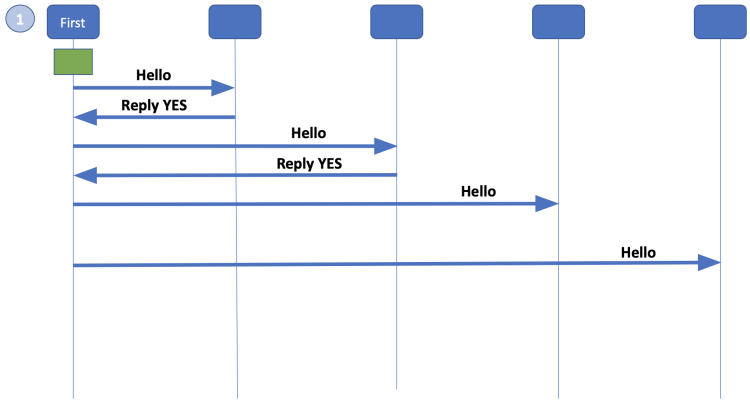
3. Protocol Set

Our protocol, between each pair of nodes (in the link layer), will be based on Stop&Wait implementation. Neighbour discovery is going be implemented dynamically, so we will be sure that we have reach every possible node.

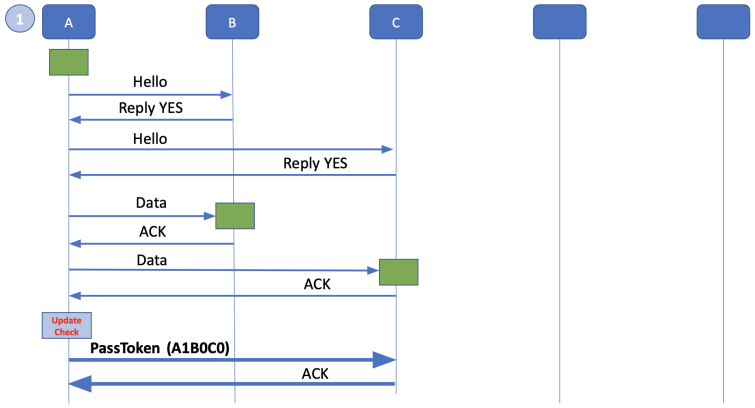
3.1. Information Flow diagram

In this section we are going to explain the procedure of this Network Protocol and the different steps in order to send the data properly along all the devices.

- *Hello* message is the first thing that is needed to be done by the first transmitter node. It will follow the polling technique, sending Hello to all the devices one by one and waiting for a $Reply\_YES$ message. This reply means that they do not have the file already and that they are ready for the transmission. Those who are not reachable will not reply any Reply packet in their specific time slot. The transmitter node will save the nodes IDs in its routing table (saving the neighbors locally in each node) in order to send the file separately to each node.



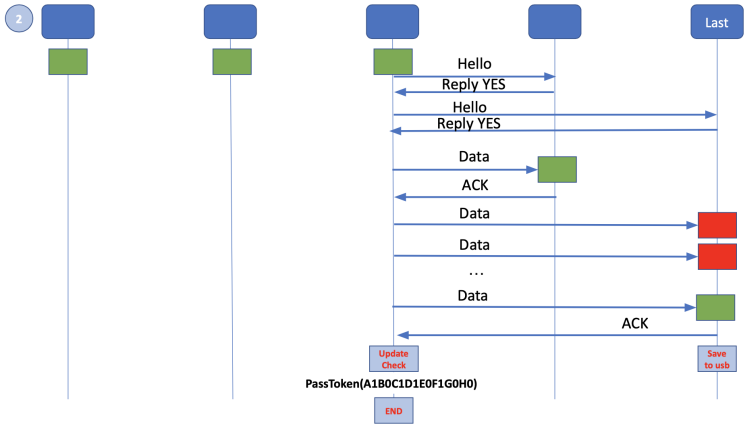
This protocol will use a token or a transmission role, which means that the device who owns this token will be the only one able to start the connection with other nodes and to transmit the file.



Once the transmission between the nodes is completed, the transmitter will send the token or the transmitter role to one of the nodes to whom it has sent the data correctly.
It is important to record in the payload of the token a table with the node identifiers (ids) that have received the data and if those nodes have already received the token or not. The node sending the token will update the token table, saying that he has received the token and adding the nodes that have received the data. However, after updating the token and before sending it, the node will check if all the nodes (8) have received the data, and in that case, the objective will be completed and therefore the session can finish. If not, it will send the token and continue with the communication.

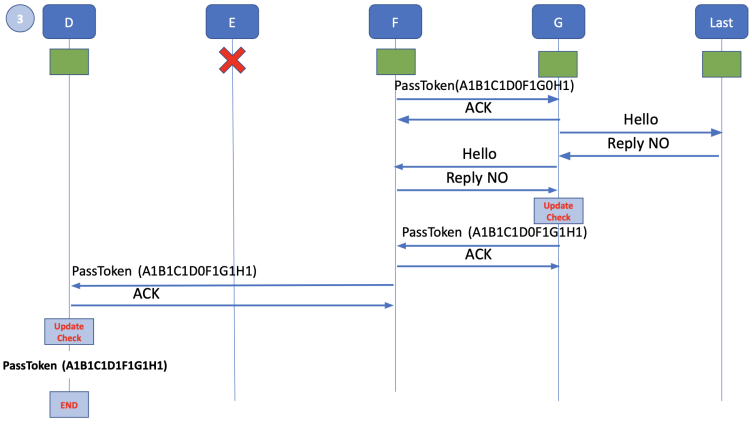
- Once the token has been transmitted to another node, this new node repeats the same procedure as before. The nodes that do not have the file will answer with a *Reply_YES* and the nodes that already have the file will answer with a *Reply_NO* message, letting the transmitter know that they do not want the file to be received again.

  In the case that the data is not transmitted correctly, the transmitter will start the retransmission procedure several times until the counter ends or the file is transmitted properly.



- Once a node only receives *Reply_No* messages, checks if the maximum number of nodes (8) have received the data. Independently if they have received the token or not, this will mean that all of them have the data and that the session can finish.

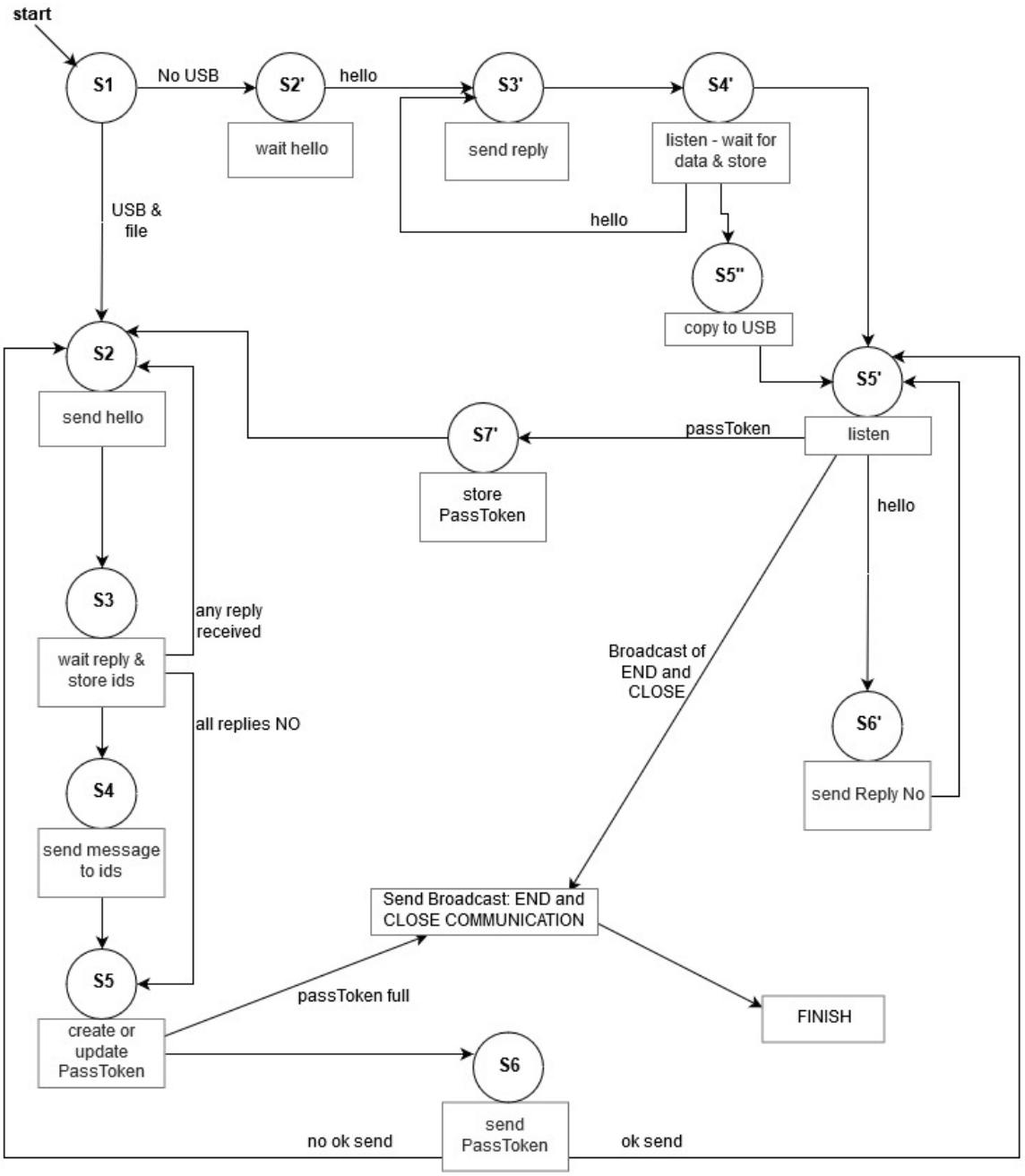
  In case that not all the nodes (8) have the data, checks in the token table if there is at least one node that has not had the token, but has received the data. If this is the case, the node will send the token message to this node, taking into account that the destination network address will be the one of this node, and probably the destination link address will be another different, because the source node may not be capable of sending the packet directly to the destination node.



  In the case that every node that has received the data correctly has had the token, the transmission will be finished.

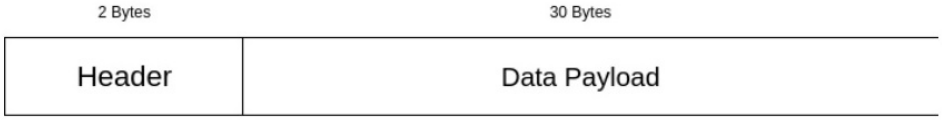
3.2. General State Diagram

The thread initialized is all the nodes will follow this structure. Initially they will check if they have a USB connected and follow the corresponding path, waiting for hello or starting the communication (if it is the first node in the chain).



3.3. Frame Structure

The frame has a maximum length of 32 bytes where we have to allocate both the data and the different control fields in order to be able to achieve the communication between the 8 devices. In the next figure we can see how our frame distribution is made. First we have a 2 Bytes Header part, where we are going to allocate different information about the transmission (addresses, ACK, sequence number, type of packet), corresponding to the link layer communication as well as for the network layer. In the second part of 30 Bytes, data payload will be allocated.



@Source (Link Layer)	@Dest. 1 (Link Layer)	ISACK	SN	Network		
				@Dest. 2 (Network Layer)	Type	Data
4 bits	4 bits	1 bit	1 bit	4 bits	2 bits	30 bytes
1 byte		1 byte				

- **@Source**: address of the source device.
- **@Destination1**: destination address of the packet in the link layer.
- **ISACK**: 1 bit to know if is ACK
- **SN**: In this case we only need one bit, as we only need to know if the received packet is the expected or not (0 or 1).
- **@Destination2**: destination address of the packet in the network layer.
- **Type**: packet identification. This field is used by the receiver in order to know which type of packet it is receiving.

3.3.1. Addresses

The network will be formed by a maximum of 8 devices. As we need to identify each device, each team will have 2 addresses previously assigned:

- Group A: 1, 5
- Group B: 2, 6
- Group C: 3, 7
- Group D: 4, 8
- Broadcast: 0

3.3.2. Packet type

There are 3 different types of packets used in this protocol:

- *Control*
  This packet type includes three different kind of packets which are related to the notification of the device presence. This kind is represented by **00**. They are the following:
  - Hello: First packet sent by the transmitter in order to know how many devices will be listening and waiting for the transmission of the data.
  - Reply YES: First packet sent by the receiver in order to let the transmitter know that we are listening and waiting for the transmission of the data. This header is only sent by the devices that do not have the file.
  - Reply NO: First packet sent by the receivers that have the file in order to let the transmitter know that they already have it.
  - EOT: This packet will be sent by broadcast and indicates the end of the network session.
- *Data*
  This type of packet indicates that we are sending data. This kind of packet is represented by **01**.
- *PassToken*
  This kind of packet is used to pass the token or the transmitter role to another device. This packet will carry the information about the nodes which have received the data and the ones that have received the token. This kind of packet is represented by **10**. And in the first phase of the protocol, when you receive *Reply_Yes*, it is sent randomly to a reachable node in the link layer (to whom you have sent the data before). But in the second phase (when all the replies of a node are *Reply_No*), the network destination address will usually be unreachable on the link layer, and will be sent to the predecessor (the one that has sent to you the token in the first place).
  The payload of this packet will indicate the ids of the nodes that have received the data and whether they have received the token (1) or not (0).
  Example: 11203150 $->$ Nodes 1,2,3,5 have the data. Nodes 1,3 have received token.