# **LAB 1 - TASK 2**

In this task, you will learn how to create the frame that is used at the link level to send the datagram.

INSTRUCTIONS

The MATLAB code in the below window is similar to the code described in task 1 where we simulate the performance of the slotted ALOHA protocol. The only difference in the code is that here we do not use the function **createFrame** to generate the frame structure but implement the function in detail.

Here, we consider a modified (simplified) frame structure, which consists four blocks, each with four bits. The total frame length is 16 bits. The four blocks contain the preamble, the node ID, the datagram, and the checksum. An example of the frame is shown as follows:

[ 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1]

The preamble, which is "1010" in this example, is a fixed sequence of bits utilized to indicate the arrival of a frame. The node ID, which is "0001" in this example, is the binary representation of the user's **id**. Here, the user **id** is 1, indicating that the frame is from the first user. The conversion of the user id from decimal to binary is achieved by using the function **num2bin**. Given that the frame structure only uses four bits to represent the id, we can simulate a system with a maximum of 16 nodes. The third block, "1010", is the datagram of the user, which is provided by the function **getNewDatagram**.

The final block contains the checksum bits, which are "0001" in this example. The checksum bits are utilized to check whether errors occured during the transmission (recall the channel coding schemes we learned in Part I). In this simulation, the checksum bits are generated as the even parity bits for the first three blocks of the frame. Specifically, we first divide the first 12 bits in the frame into 4 groups, each with 3 bits, as shown by the columns below:

[ 1 0 1 0 ]

[ 0 0 0 1 ]

[ 1 0 1 0 ]

Then, the 4 checksum bits are computed by performing the "exclusive or" operation over all three bits in each group (one column in the above array). Equivalently, we can obtain the checksum bits by binary addition. The exclusive or of the bits in the first three (leftmost) columns are 0. The exclusive or of the bits in the rightmost column is 1. As a result, the checksum bits for the above example are [0 0 0 1]. Another way to compute the checksum is counting the number of 1s in the words and set the bit of the checksum to 1 if this number is odd, and to 0 otherwise. For example, for the leftmost bits we have two ones, which is even, so the checksum for that bit is 0.

Your task is to create the frame for the datagram and store the result inside the variable **frame**. In order to obtain the binary represenation of the user's **id**, you can use the function **num2bin(id,4)**, where the second argument indicates the length of the binary representation. Set the preamble to be **[1 0 1 0]** for this simulation. Please, revise the code between the lines

% % % % Revise the following code % % % %

and

% % % % Do not change the code below % % % %

Do not change other parts of the code and do not use the function **createFrame**.

Total number of slots: 1000 Empty slots: 664 Collisions: 335 Frame transmitted successfully: 1

