

**HKUSTx: ELEC1200.3x A System View of Communications: From Signals to Packets (Part 3)**

Bookmarks

► Pre-course
Materials

► Topic 1: Course
Overview

▼ Topic 2: The Link
Layer

2.1 Link Layer

Week 1 Quiz due Jan 25,
2016 at 15:30 UTC

**2.2 Multiple Access
Protocols**

Week 1 Quiz due Jan 25,
2016 at 15:30 UTC

**2.3 Aloha Protocol**

Week 1 Quiz due Jan 25,
2016 at 15:30 UTC

**2.4 Efficiency of Slotted
Aloha**

Week 1 Quiz due Jan 25,
2016 at 15:30 UTC



Topic 2: The Link Layer > 2.5 Lab 1: Link Layer > LAB 1 - TASK 1



Bookmark

LAB 1 - TASK 1 (EXTERNAL RESOURCE) (1.0 points possible)

2.5 Lab 1: Link Layer

Lab due Jan 25, 2016 at
15:30 UTC



- ▶ MATLAB download and tutorials

LAB 1 - TASK 1

This task provides an overview of the slotted ALOHA system. Your task is to compute the efficiency system, which is defined as the ratio between the number of successful transmissions and the number of slots.

INSTRUCTIONS

In our simulation of the slotted ALOHA protocol we study here, we assume a population of **n_users** nodes (nodes with data to send) access a shared channel by transmitting a frame in each slot with probability $0 < p < 1$. There is a nonzero probability that only one of the nodes will use the channel avoiding collision and enabling a successful transmission. Since access to the channel is controlled by random transmission by the nodes, we call this a random access scheme. This is a slight simplification of the original ALOHA protocol, since we do not consider retransmission, but rather that frames are lost if collisions occur, and we do not explicitly detect collisions.

The MATLAB code in the below window simulates a slotted ALOHA system with **n_users=4** nodes. Each node is identified by an index **id** ranging from 1 to **n_users**, and transmits a data frame in each slot with probability **p=0.1**. The code simulates the transmission for **n_slots=1000** time slots. At the beginning of each slot, the code initializes the variable **slot**, which is the state of the shared channel in the current slot, to zero. Then, each user checks if its frame is empty. Its frame will be empty if the user transmitted its frame in the previous time slot. If it gets a new datagram (function **getNewDatagram**) and then creates and stores the frame (function **createFrame** and **updateFrame**). Then, the user determines whether to transmit the frame or not by generating a Bernoulli random variable with parameter **p=0.1**. If the user transmits its frame, the code updates the previously stored frame (function **getFrame**), updates the variable **slot**, and then resets (function **resetFrame**) so that it will get a new frame in the next iteration. If the user does not transmit, the frame is retained for future transmission. In this simulation, we assume a binary channel where the channel state, **slot**, is obtained by taking the logical **OR** operation among all frames transmitted. In a practical system, this would be implemented if each node was connected to a common wire with a pull down resistor to state at zero if no nodes are transmitting, and each node is connected to the wire with a pull up transistor.

The correctness of the received frame is checked using the checksum (function **checkReceivedFrame**).

will be explained in details in Task 3. In this code, errors in the transmission are only due to multiple transmissions in the same time slot (collision), not to noise. If the checksum is correct, we increment of successful transmissions (**n_succ**). If the checksum is incorrect, we increment the number of collisions (**n_coll**). If the slot is empty (zero), we increment the counter of empty frames (**n_empty**).

At the end of the simulation, we display the total number of time slots, empty slots, collisions and successfully transmitted frames. We also plot in Fig. 1 the transmission traffic for the first 50 slots to illustrate how the protocol works in controlling the access of multiple users to the same shared channel. For each user, we show a circle at the time it transmits a frame. When we observe more than one circle in a time slot (e.g., 4, 14, ...), there is a collision. When there are no circles (e.g., slots 1, 5, ...) there is an empty frame.

Your task is to compute the efficiency of the system and store the result inside the variable **efficiency**. You may need to revise the code between the lines

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

and

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

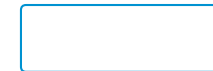
Do not change other parts of the code.

Your Solution

 Reset

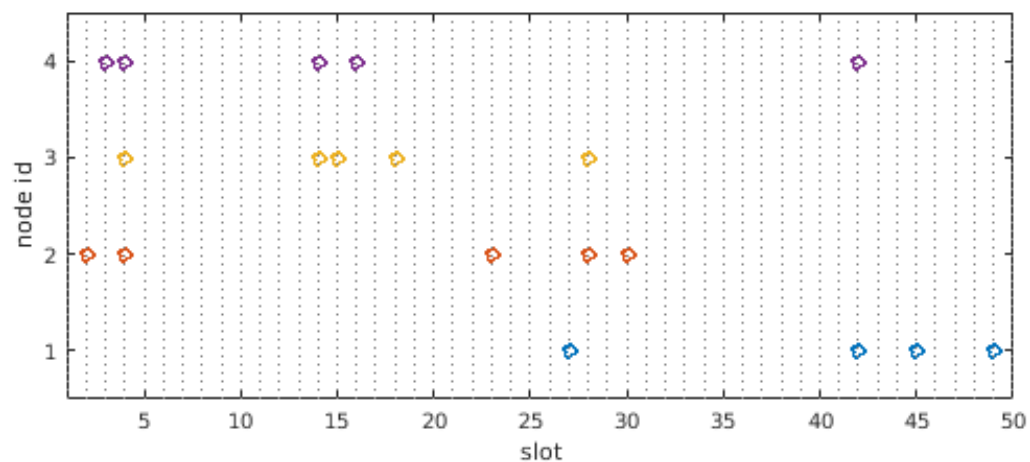
 MATLAB Documentation (<https://www.mathworks.com>)

```
63 display(['Collisions: ' num2str(n_coll) '']);
64 display(['Frame transmitted successfully: ' num2str(n_succ) '']);
65
66 % plot
67
68 fh = figure(1);
69 window_size = 50;
70 plotTraffic(fh, slot_status, n_users, window_size);
```

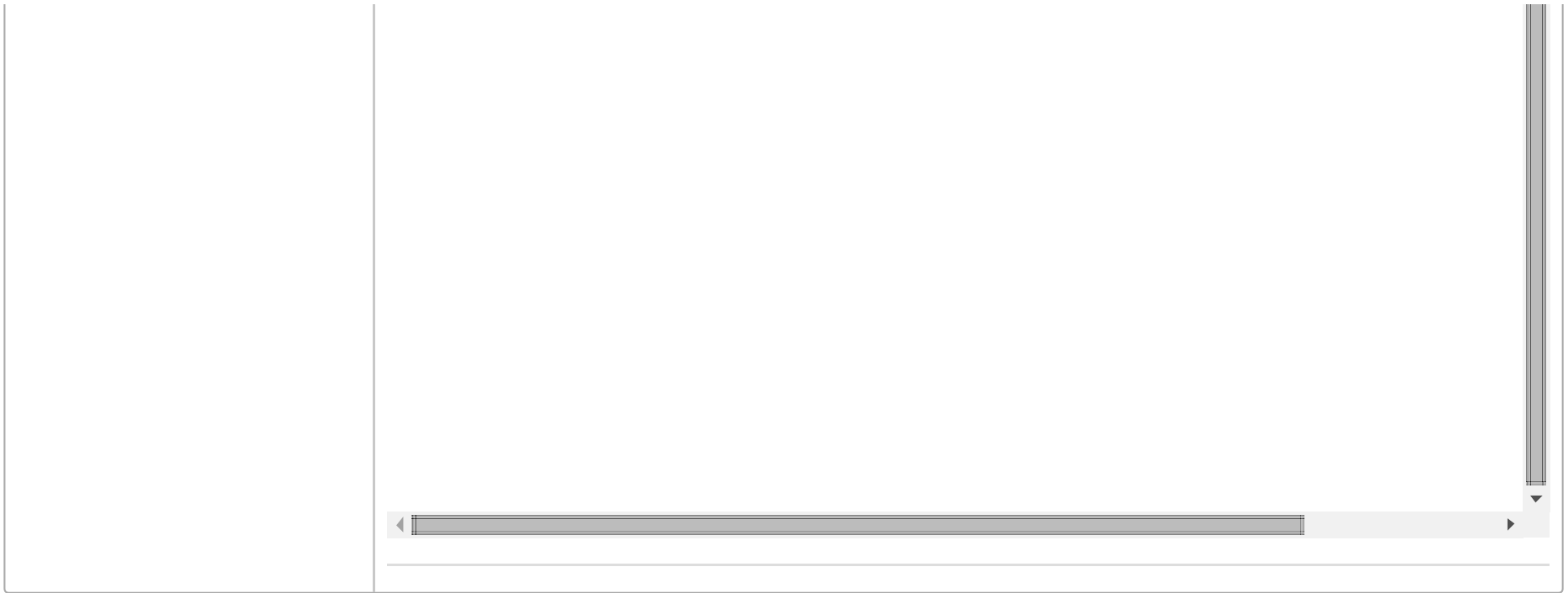


Output

Total number of slots: 1000
Empty slots: 636
Collisions: 48
Frame transmitted successfully: 316



Assessment Tests



© All Rights Reserved



© edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.

POWERED BY
OPENedX



