# Programming Assignment 3

**Data Communication Networks – Fall 2021**
**Investigating Backoff Protocols**

**Due Date:** November 20, 2021 by 11:59pm CST (local time in Starkville)

**Carefully follow the instructions for submitting your solution.**

## 1. Assignment Objective

The objective of this assignment is to evaluate through simple simulation of three backoff protocols. Your simulation code can be written in any language you wish; you do not need to submit your code, and you may work on your local machine if you wish (you do not need to use Pluto). You are also responsible for submitting a document plotting your resulting data and explaining your findings (as a pdf); described in more detail later.

## 2. Overview and Description of Backoff Protocols

This section provides a description of the **three** backoff protocols to be implemented. Your implemented simulation must align with this description. There is no requirement to simulate true concurrency, nor should you pursue such a goal (you are building a discrete-event simulation).

In your simulation, time is assumed to be discretized: a unit of time in the simulation is referred to as a ***slot***. Slots are indexed starting at 1. Assume that any data transmitted by a device can fit within a single slot. If only one device transmits in a slot, then the transmission is ***successful***. Else, if two or more devices transmit in a slot, then this results in a ***collision***; that is, this the respective transmissions ***fail*** for all transmitting devices in that slot.

There are assumed to be ***N devices*** (see details for setting this later on in this specification). Each device begins the simulation with a single packet to transmit. Once this packet is successfully transmitted, the device is no longer active and you need not track its status in the simulation further.

Your simulation should not model the internal details of packets or the data transmitted (that is, this simulation is not worried about frame structure or contents). For your simulations, you are only concerned with whether 0,1, or more devices send within any particular slot.

The general structure for any backoff protocol is as follows. Starting from an initial time slot, contiguous slots are grouped into ***windows*** $W_1$, $W_2$, $W_3$… and so on. The size of a window $W_j$ is the number of slots in the window and this is denoted by $|W_j|$. In each window, a device selects a single slot uniformly at random to attempt a transmission of its packet. If the transmission is successful in this slot, the device terminates the protocol. Else, the transmission fails and the device waits until the end of the current window and repeats the same actions for the next window.

**Note:** It is **not** the case that a device sends in each slot with some non-zero probability. Each device picks 1 slot uniformly at random to send in (there is a difference).

The three backoff protocols you will analyze are different only in how the windows are defined. Keeping this in mind should help simplify the structure of your code. The remainder of this section succinctly defines the three protocols to be simulated.

### 2.1 Linear Backoff

You will start by implementing a backoff protocol named ***linear backoff***. In this protocol, the initial window size is 2 slots. Each subsequent window increases by 1 slot.

For example, $|W_1| = 2$, $|W_2| = 3$, $|W_3|=4$, etc.

### 2.2 Binary Exponential Backoff

The next backoff protocol to be implemented is the familiar ***binary exponential backoff*** protocol. The initial window size is 2 slots. Each subsequent window increases by a multiplicative factor of 2.

For example $|W_1| = 2$, $|W_2| = 4$, $|W_3|=8$, etc.

|

### 2.3 LogLog Backoff

The final backoff protocol is called ***loglog backoff***. Here, the initial window size is 4 slots. Each subsequent window is defined as follows. For a current window $W_j$ which has just ended, the next window $W_{j+1}$ has $(1 + 1/\log_2\log_2(|W_j|))*|W_j|$ slots; when this value is not an integer, you should take the floor of the value. Note that the logarithm is base 2.

For example, $|W_1| = 4$, $|W_2| = (1 + 1/\log_2\log_2(4))*4 = 8$, $|W_3| = (1 + 1/\log_2\log_2(8))*8 = 13$ , etc.

## 3. Aspects of Implementation

To understand how to build your simulation, it may be helpful to explain the execution with some concrete numbers.

Time proceeds in times slots (or just ``slots"). Let's say the very first slot is indexed by 1. In each slot, a device can try to send a message. You are NOT modeling any of the message contents, or physical-layer effects (like fading, interference, etc. that we discussed in class)

Windows are a conceptual construct used to break up slots into groups. For example, if I have slots 1 to 100, linear backoff would use windows consisting of slots:

W1 = slots 1, 2
W2 = slots 3, 4, 5
W3 = slots 6, 7, 8, 9
W4 = slots 10, 11, 12, 13, 14

Let's focus on W1 and consider N=500 devices. Each device will pick uniformly at random either slot 1 or slot 2 to attempt to transmit.

If slot 1 is chosen by exactly 1 device, then that device "succeeds". However, most likely, you'll get roughly 250 devices picking slot 1, and the remainder picking slot 2. So, no device succeeds in W1, and all devices proceed to W2.

You can implement this simulation any way you like. For example, you might simply have an integer array of size 2, and a random number generator (RNG) that lets you generate 500 random values in the set {0,1}. Each time 0 comes up from the RNG, increment index 0 in your array; otherwise, increment index 1. Then, at the end, you can see how many devices "chose" slot 1 (index 0 in your array) or slot 2 (index 1 in your array).

For W2, all N=500 devices pick a slot uniformly at random from the set of slots {3,4,5}. Again, it's unlikely any device by the kind of probabilistic analysis we did in class, right? You expect N/3 for each slot (there will be variance of course, due to this being a random process). Here, you could use an array of size 3, keeping a record of how many devices pick each slot.

So all devices proceed to window W3, where each device again picks uniformly at random from {6,7,8,9}...and so on.

You probably won't see any devices succeed until you get up to a window containing ~500 slots or so. At that point, you'll need to update the number of devices that still haven't succeeded (it will be less than 500 if some succeed). So you are going to update N (it will decrease) once devices start finishing/succeeding; you can do this at the end of each window.

Finally, let's assume that the very last device finishes/succeeds inside of window W700 (I'm just making up 700 to be concrete). And let's assume it finishes in slot 10 of W700. Then, the latency for this execution of linear backoff is the total number of slots up to and including this slot 10. Or put alternatively: the sum of all prior window lengths W1, W2, ..., W699 plus 10.

## 4. Your Simulation Code

This section provides advice on how to structure the output of your code (this is optional), as well as the amount of data to produce (this is required).

Your code should produce output for each of the three protocols defined above in Section 2. The output of interest is the number of slots required to have all devices succeed; we will refer to this metric as ***latency***. It may be helpful to generate three separate files, one per protocol:

```
linearLatency.txt
binaryLatency.txt
loglogLatency.txt
```

We will describe the format of this output momentarily.

Let us say that your executable is called `myBackoff.` The execution of `myBackoff` will run trials of these protocols. A trial is a single execution of one protocol for N devices.

For each protocol, the first trial will use N=100 devices, and each subsequent trial will increase this by 100 more devices. For example, an execution will run linear backoff with 100 devices, then with 200 devices, then 300 devices, etc. all the way up to **N=6000 devices**.

Each trial will be repeated 10 times per N value. For example, the first trial for linear backoff will run 10 simulations with 100 devices, then 10 simulations with 200 devices, and so on. This allows us to avoid data that is too biased by unlucky executions of the randomized protocol.

For each trial, the corresponding text file should contain the average latency only. For example, for the first trial of linear backoff, the ***average latency*** over 10 repetitions will be written to `linearLatency.txt`. I recommend writing each value on its own line, since this will make plotting easier in most cases.

The experiments should run fairly quickly, in less than a minute. So, if you're seeing much longer run times, then you're likely misunderstanding something; in that case, come talk with me.

For measuring the latency in your simulation, remember to count up to the slot that the last device finishes (note this is different from the entirety of the current window if all devices finish prior to this point). For example, consider linear backoff executing in a window of size 5. If the last device succeeds in the 3rd slot of this window, then that is the point (including this slot) up to which you should count the latency, rather than counting all 5 slots in the current window. In particular, the latency would be 2 + 3 + 4 + 3 = 12 since the initial window size is 2, the next is 3, the next is 4, and 3 slots of the last window were required before all devices finished. The number 12 would be written on a single line in `linearLatency.txt`.

## 4. Your Writeup

You must present the findings of your simulations. In particular, you must plot your data for the three protocols on a single graph where the x-axis is the number of devices (increasing by 100 in each trial) and the y-axis is the average latency. Both axes must be labeled.

It should be clear which points correspond to which protocol and there should be a legend with the necessary labeling to make this clear. You may join the points within each protocol by a "trend line" if you wish, but it is not necessary. No error analysis is necessary.

You may plot this using any program you wish; however, you are **not** to plot this by hand. So long as the plot meets the requirements mentioned above, and is of sufficient size to be read clearly, the particulars of how you accomplish this is up to you.

In addition to the plot, you should give a brief report on your findings; this should be at most a couple paragraphs. This report should answer the following questions:

1. Which backoff protocol appears to give the best (lowest) average latency value as the number of devices tends to infinity?

2. Which backoff protocol appears to give the worst (highest) average latency value as the number of devices tends to infinity?

3. Why do you think the worst (in terms of latency) protocol is behaving so poorly compared to the other two?

## 5. Your Submitted Solution

### 5.1 Due Date

This assignment is due on November 20, 2021 by 11:59pm CST (local time in Starkville).

### 5.2 Hand in Instructions

***(i)*** You must hand in the following through Canvas:

- **Your report on your findings (as described in Section 4) with your plot embedded in the document; if the plot is separate, you will lost 5 points. This must be in pdf format and the file must be of reasonable size (less than 5 MB)**