**Chapter 12: Multiple Access**

Table of Contents

[Throughput for Pure ALOHA 3](#_Toc86834818)

[Vulnerable Times 3](#_Toc86834819)

[Derivation 4](#_Toc86834820)

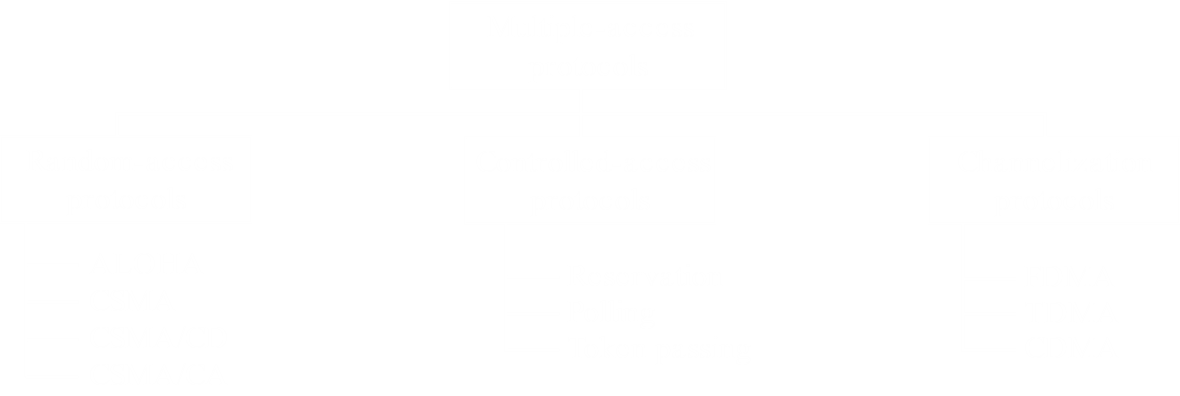
[Average Transfer Delay for Pure ALOHA 7](#_Toc86834821)

[Throughput for Slotted ALOHA 9](#_Toc86834822)

[Average Transfer Delay for Slotted ALOHA 10](#_Toc86834823)

We have previously discussed **multiple access techniques**, or **MAC protocols**, which are needed when resources are being shared between users and must be accessed mutually exclusively.

The MAC protocols we have studied are shown in the diagram below:



One of the first protocols that we studied was ALOHA, more specifically, **pure ALOHA**. We saw that, comparatively, it was a terrible technique, with only **18 frames** out of every 100 attempts being sent successfully. This is known as the **throughput** of the ALOHA protocol and that is the first topic we will be discussing now.

## Throughput for Pure ALOHA

The throughput for pure ALOHA is given by

where is the **load**.

For pure ALOHA, at most, which gives . This is how we know that only 18 frames out of every 100 attempts can be delivered successfully at most. However, we still need to discuss how this equation came to be.

### Vulnerable Times

Before we can get to the derivation, we need to go over **vulnerable times** again. The vulnerable time for pure ALOHA is twice the propagation time of one frame. This is because, when one frame is being sent, a **collision** may occur with the very **last bit** of the **first frame** and the very **first bit** of the **second frame**. This is the worst-case scenario and would cause the propagation times for both the first and the second frames to be lost.

The first improvement made over pure ALOHA, **slotted ALOHA**, actually worked to improve the vulnerable time. By providing slots in which frames would need to be sent, **partial collisions** like the one discussed above could not occur. Only complete overlaps of frames could occur. This meant that the vulnerable time was reduced to just the propagation time of one frame.

### Derivation

With the concept of vulnerable time in mind, we can begin to derive the **equation for throughput**.

For a single frame to be sent successfully, we need to ensure that not even a single bit from a different frame overlaps with the first bit of this frame, or with the last bit of this frame. If is the **transmission time**, or the time taken to transmit a single frame, we are essentially saying that for one frame to be successfully sent, no frames can be sent for time before that frame or for time after that frame.

A single transmission is not dependent on anything else. The probability of a single transmission occurring at a given moment is completely independent. As such, the probability model is a **Poisson distribution**.

The probability of transmissions occurring within the next time units is given by:

Here, is the **arrival rate**, or the number of frames being sent per unit time.

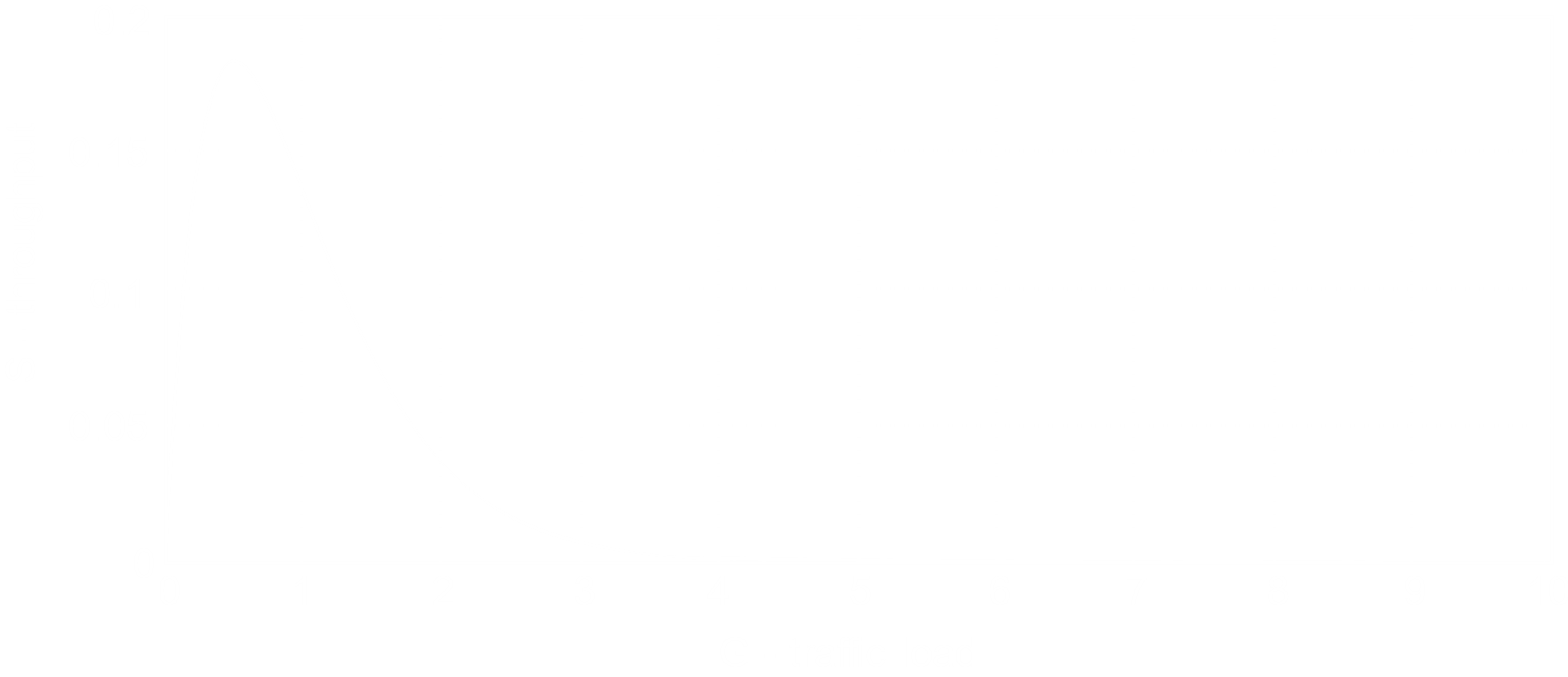
Thus, the probability that a single frame will be successful, , which requires frames to be sent in time before and after that frame is given by:

essentially tells us the number of frames that are being sent in every unit of transmission time. This can be written as the **load**, or the amount of the totally capacity of the system currently being used, .

**Throughput** is the number of **successful transmissions** per unit time. If we consider the time unit to be the transmission time, the rate at which successful transmissions occur is given by:

The maximum throughput can be found by differentiating this.

If we consider the graph of throughput against load for pure aloha, it looks like this:



Clearly, the throughput **increases at first**, which makes sense, since the more packets we transmit, the higher the transmission rate. However, after is crossed, if we keep sending more packets, the throughput will **decrease**. This is because the number of collisions, and consequently the number of retransmissions, increases to such a high extent that fewer and fewer packets are successfully transmitted.

## Average Transfer Delay for Pure ALOHA

If tells us the number of attempted transmissions per unit time and tells us the number of successful transmissions per unit time, then tells us the number of **transmissions** that had to be attempted for a **single successful** transmission to occur on average.

Thus, there are transmission attempts by each packet before they are successfully transmitted, on average. This is denoted as , the **number of retransmissions**.

For every failed attempt, we need to take into account the **transmission time** for that attempt, , and the **backoff time** after the attempt fails, . Thus, on average, the total time required to successfully transmit a packet, or the **average transfer delay**, is:

Here, is the **average backoff time**. The backoff time is the transmission time multiplied by some multiple, , the value of which ranges from to . Thus,

## Throughput for Slotted ALOHA

For slotted ALOHA, the **vulnerable time** is . Thus,

## Average Transfer Delay for Slotted ALOHA

From the equation for **throughput**, we can derive

For the average transfer delay, we need to take into account four things:

1. How long does it take to get to the next slot? A particular station could request a transmission at any time, but it will not be allowed to transmit until the next time slot starts, which could be a maximum of time later. If the probability of a transmission request is a uniform distribution, the average delay here is .
2. How long does it take to actually transmit the frame? This is just the frame transmission time, .
3. The total time wasted due to retransmissions, . Here, is the time taken to realize that a collision has occurred. is the smallest integer greater than . is the propagation time, as opposed to , which is the frame transmission time.
4. The propagation time, . The is because we assume, on average, the distance between any two stations is 1/3rd of the total length of the network.