**Score: \_\_\_\_\_**

**NA2 – Physical and Data Link Layers**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

In today’s class we discussed the physical and data link layers. We saw several methods by which the 0’s and 1’s that are being transmitted can be encoded into electrical signals (the NRZ and Manchester encodings) and several ways they can be encoded on radio signals (frequency shift keying and quadrature phase shift keying). We then moved to the data link layer and discussed Ethernet, seeing that uses a broadcast medium and discussed how access to that medium is controlled using Carrier Sense Multiple Access with Collision Detection (CSMA-CD) and how exponential backoff is used in that process. Finally, we discussed parity bits as a way to detect transmission errors across a link. This homework will revisit those ideas and guide you in a deeper exploration of some of them.

**The Physical Layer:**

🔑 1. Data can be transmitted serially (one bit at a time) or in parallel (multiple bits simultaneously). Use human transportation as an analogy to explain the difference between serial and parallel data transmission.

🔑 2. When transmitting data serially over a physical medium the 0’s and 1’s in the data must be encoded into signals on the medium. For example, in class we saw the Nonreturn To Zero (NRZ) and Manchester encodings for transmitting 0’s and 1’s over a wire. There are many others as well.

a. Give the string of bits that are encoded in the signal below assuming that it is using NRZ encoding.



b. Give the string of bits that are encoded in the signal below assuming that it is using Manchester encoding.



🏆 c. There are many other encodings in addition to NRZ and Manchester. Watch the following video that describes the NRZ-I encoding:

* <https://www.youtube.com/watch?v=Kxndom8GaUQ> (1:24)

Draw in the Data signal in the figure below so that it encodes the same bits of data as the NRZ signal from part a.



🏆 3. The Manchester encoding is clearly more complicated than the NRZ encoding. However, the added complexity of the Manchester encoding has an advantage in that it helps with clock synchronization between the sender and receiver. This question explores that advantage. Refer to the Data traces for the NRZ and Manchester encodings in parts a and b above to help in answering the following questions.

a. Will an NRZ encoding of a long string of 0’s or a long string of 1’s contain any transitions from ground to high voltage or vice versa?

b. When the destination receives a long string of 0’s or 1’s, will that data contain any information that could help the destination synchronize its clock to the sender’s clock? What could happen as a result?

c. Briefly explain how data transmitted using the Manchester encoding mitigates the problem you described in part b.

**Ethernet in the Data Link Layer:**

🔑 4. Ethernet use Carrier Sense Multi-Access with Collision Detection (CSMA/CD) over a broadcast media when transmitting data.

a. All Ethernet frames are broadcast to every node connected to the local area network (LAN). How does the frame being sent get processed by its intended recipient and not by the other nodes?

b. What does an Ethernet device do before it begins transmitting a frame? Why does it do that?

c. When two Ethernet devices begin the process of transmitting a frame at almost exactly the same time, they may both decide that it is okay to transmit. Explain at a high level, what happens when they both begin to transmit their frames.

d. What is the purpose of transmitting the preamble with each Ethernet frame?

🔑 5. When two or more Ethernet devices detect a collision, they each wait a random amount of time before trying to retransmit their frame. The random amount of time that an Ethernet device waits before attempting to retransmit a frame after a collision is determined using an exponential backoff algorithm. This question explores that algorithm a little bit.

a. How long (in clock ticks) does an Ethernet device wait on average the first time it detects a collision when trying to transmit a frame? Briefly, explain your answer.

b. Consider two nodes having a collision the first time they try to transmit a frame. What are the chances these two nodes will then have a second collision?

c. Do the chances of two nodes having another collision increase or decrease with each successive collision?

d. How many collisions in a row would have to occur before the maximum time an Ethernet device would wait to try to send a frame again exceeded 16,000 clock ticks?

🏆 e. What is the longest time (in seconds) that a 100Mbs Ethernet device would wait after a collision, due to its exponential backoff, before trying to resend a frame?

6. The exponential backoff algorithm is so named because the range of possible times that an Ethernet device will wait after a collision grows exponentially with successive collisions (up to 10 collisions). This question explores one of the reasons why Ethernet (and other network protocols) use exponential backoff. Keep in mind as you answer these questions that Ethernet, as we have studied it, uses a broadcast medium and thus, all nodes must share the available bandwidth of that medium.

a. Will collisions become more or less frequent as the network becomes busier (i.e. more nodes are trying to transmit frames)? Briefly explain your answer.

b. Based on your answer to part a, what is likely to happen to the *data rate* of an individual node as the network becomes busier? Briefly explain your answer.

c. Based on your answer to part b, briefly summarize how the exponential backoff algorithm helps to ensure that nodes adjust their *data rates* to share the *bandwidth* of the broadcast medium without requiring coordination.

🏆 7. Using what we know about the ethernet frame format (i.e. don’t try to look up an answer to this question, you will not find the correct answer). If the *bandwidth* of the communication link being used at the physical layer were 1Mbs, what would the maximum data rate for Ethernet transmissions over this physical link? Hint: Find the amount of useful data that would be transmitted assuming the entire payload (data part) of the frame is useful.

**Error Detection and Correction:**

🔑 8. A parity bit is an example of an *error detecting code*. This means that by adding a parity bit we can detecting transmission errors (e.g. a 0 being flipped to a 1, or vice versa).

a. Complete the following table by indicating the value of the parity bit for each given byte using odd or even parity as indicated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Parity Type** | **Byte** | **Parity Bit** |  |
|  | Odd | 1010 0011 |  |  |
|  | Odd | 0010 0101 |  |  |
|  | Even | 0101 0111 |  |  |
|  | Even | 1100 0110 |  |  |
|  |  |  |  |  |

b. Each of the following data values has been received by a device after having been transmitted using either even or odd parity as indicated. Fill in the “Error Detected” column with either “Yes” or “No” to indicate if an error can be detected in the transmitted Data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Parity****Type** | **Data** | **Error Detected** |  |
|  | Odd | 010111011 |  |  |
|  | Odd | 101101010 |  |  |
|  | Even | 001000000 |  |  |
|  | Even | 111100111 |  |  |
|  |  |  |  |  |

9. Using a parity bit allows us to detect some errors but not all errors. This question explores those limits. Imagine that the byte 0110 0011 were transmitted from node A to B with an even parity bit added.

a. Give a byte plus a parity bit that might be received by B if *exactly two* transmission errors occurred.

b. Will B be able to detect when received bytes contain exactly two transmission errors have occurred?

c. Give two examples showing byte with a parity bit that could be received at B, one containing *exactly three* transmission errors and another showing *exactly five* transmission errors. Would B be able to detect these errors?

d. Generalize from what you observed in parts b and c to briefly summarize the types of transmission errors that a parity bit will allow node B to detect and those that it will not allow node B to detect.

10. By using more parity bits we can not only improve our ability to detect transmission errors we can actually correct some of them! Two-dimensional (2D) parity and is an example of an *error correcting code*. This question explores how 2D parity can detect and correct some types of transmission errors.

a. With 2D parity a block of data being transmitted is broken into rows and columns and a parity bit is computed for each row and each column. Imagine that node A wishes to transmit the following four bytes to node B using 2D parity:

1010 0110 0011 0100 0101 1100 0111 1110

One way to do so would be to arrange the bytes into a 2D grid as shown in the table below. Then a parity bit can be computed for each row and each column.

Fill in the table below by adding an *even parity bit* for each row and each column. The parity bit for the first row and first column are given as examples.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | Row Parity |  |
|  |  | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | **0** |  |
|  |  | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |  |  |
|  |  | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |  |  |
|  |  | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |  |  |
|  | Column Parity | **1** |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

b. Node A would transmit the 4 bytes of data plus the 12 parity bits in part a to node B. Node B would then check all of the parity bits.

Imagine that node B receives the data and even parity bits shown below. This data contains a detectable transmission error. Highlight the row and column that contains the detectable transmission error.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | Row Parity |  |
|  |  | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | **1** |  |
|  |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | **1** |  |
|  |  | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | **0** |  |
|  |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | **0** |  |
|  | Column Parity | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

c. Briefly summarize how node B would be able to correct the transmission error that occurred in part b.

d. 2D parity can also correct some instances in which there are multiple transmission errors. Give an example using the grid below showing an example where node B would be able to detect and correct two transmission errors. Highlight the rows and columns in which the transmission errors occurred.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | Row Parity |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | Column Parity |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

e. While 2D parity can correct some instances in which there are multiple transmission errors it cannot correct them all. Give an example using the grid below showing an example where there are two transmission errors, where node B would be able to detect but not correct two transmission errors. Highlight the rows and columns in which the transmission errors occurred.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | Row Parity |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | Column Parity |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

🏆 11. Optional: Other layers of the network stack also used error detection mechanisms. One that is commonly used at the network and transport layers is called the Internet Checksum. Watch the video below by Jacob Schrum at Southwestern University in which he explains how to compute an Internet Checksum:

* Error Detection and Correction 1: Internet Checksum
	+ <https://www.youtube.com/watch?v=EmUuFRMJbss> (9:07)

Then compute the internet checksum that would be used to transmit the 4 bytes of data transmitted in question #10a. Show your work. There are numerous Internet Checksum calculators available on-line that you can use to check your answer.

🏆 🏆 12. Optional: If you have taken some more advanced mathematics courses and

* Error Detection and Correction 2: Cyclic Redundancy Check
	+ <https://www.youtube.com/watch?v=6gbkoFciryA> (12:20)

Then compute a 3-bit CRC value (the frame check sequence, FCS) that would be used to transmit the following 6 bits of data:

 101111

using the pattern 1001 as P. Show your work.

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.