# 1) This protocol has a deadlock. Describe what a deadlock is and describe the particular deadlock situation in this protocol.

A deadlock in networking is where transmission of new data ceases due to the system state. A deadlock occurs in this protocol when the last acknowledgement packet from the receiver is lost. The receiver closes after it sends the last acknowledgement packet, so if this packet is lost, the sender thinks that the last data packet was lost and keeps resending the last packet. The sender is thus in a deadlock state: it tries to retransmit the last packet continuously, preventing it from transmitting any more data it may need to.

# 2) What is the magicno field good for?

Ensuring that the message came from this protocol and that some other program has not connected to one of the programs by mistake.

# 3) How have you solved the bit errors? Please explain what you have added to the packet and to the sender and receiver modules.

Bit errors were solved by adding a checksum to the header. This was calculated when a packet was created, by taking the last three digits of the sum of the other header fields. When the sender and receiver received a packet, they checked if the checksum was correct and threw out the packet if it was incorrect.

# 4) Please explain what the select function is doing and why it is useful for channel and in another way for sender.

The select function, or in Python the select.select method waits until data can be read from the socket, or a timeout is exceeded. In channel, this method is used so that it channel can wait until it can read data from its sockets, and only then read from sockets that have new data. This is useful as it reduces code complexity and prevents the channel program from using CPU resources unnecessarily. In sender, the select method is used to enact the timeout mechanism for re-sending data if an acknowledgement packet isn’t received.

# 5) Please explain how you **have checked the file was transferred correctly.**

To check that the two files were the same, diff <input\_file> <output\_file> was run. This compares the contents of the two files and outputs nothing if they are identical.

# 6) Packet loss measurement question

The number of packets required to send a 512,000 byte long file with different packet loss rates was tested experimentally. To do this the programs were run 10 times for each packet loss rate. The results of this are shown in table 1 below, and are summarised in figure 1.

<Explanation goes here>

Table 1 - Number of packets required to send 512,000 byte long file for

different packet loss probabilities.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Packet Loss  Probability | 0 | 0.01 | 0.05 | 0.1 | 0.2 | 0.3 |
| Packets Sent | 134 | 126 | 142 | 162 | 185 | 227 |
| 123 | 136 | 139 | 154 | 191 | 202 |
| 128 | 125 | 138 | 151 | 178 | 272 |
| 122 | 119 | 153 | 131 | 188 | 293 |
| 119 | 130 | 137 | 159 | 167 | 269 |
| 124 | 115 | 141 | 156 | 208 | 269 |
| 124 | 128 | 139 | 147 | 219 | 292 |
| 121 | 123 | 136 | 145 | 173 | 215 |
| 121 | 126 | 142 | 146 | 183 | 274 |
| 118 | 136 | 140 | 150 | 179 | 253 |
| Average | 123.4 | 126.4 | 140.7 | 150.1 | 187.1 | 256.6 |

|  |
| --- |
|  |
| Figure 1 – The average number of packets required to transmit a file 512,000 bytes long. |

# 7) Average packets required derivation

Let *P* be the probability that a packet is dropped or has its data length field changed, and *N* be the number of packets that is needed to transmit some arbitrary file. Assuming that each transmission is statistically independent of all others, each transmission can be defined as a Bernoulli trial. Let the event the a given packet is dropped be a success and the event that a packet is successfully transmitted be a failure.

If a series of these Bernoulli experiments are performed then the results will have a negative binomial distribution. That is, have a probability mass function of,

|  |  |
| --- | --- |
|  | (1) |

where *k* is the number of successes, *r* is the number of failures, and *γ* is the probability of success. This distribution has the expected value of:

|  |  |
| --- | --- |
|  | (2) |

where *µ* is the expected value (Pennsylvania State University, 2017), (Weisstein, nd).

The negative binomial distribution measures the number of success before *r* failures occurs. Because of this definition, the average number of packets that must be sent to transmit successfully transmit the file is,

|  |  |
| --- | --- |
|  | (3) |

There should be constant of one as overhead from the protocol - an empty data packet is sent to signal the end of transmission - this packet can be lost and need retransmission. However, this is ignored as the last packet must pass through unmolested for successful transmission to occur.

However, the parameter *γ* is not the same as *P* - it is in fact,

|  |  |
| --- | --- |
|  | (4) |

This is because if the acknowledgement packet from receiver is lost then a retransmission also occurs.

Thus, the average number of packets needed to transmit a file that can be split into *N* packets is,

|  |  |
| --- | --- |
|  | (5) |

this can be simplified to,

|  |  |
| --- | --- |
|  | (6) |

Table 2 below, shows that the predicted average number of transmissions closely matches the experimental average. The difference will be due to the small sample size of 10 and the fact that the random number generator in Python is only pseudo-random and not truly random.

Table 2 – The predicted average number of packets required to send a 100 packet file compared to an experimental average.

|  |  |  |
| --- | --- | --- |
| **Packet Drop Rate** | **Predicted Average** | **Experimental Average** |
| 0 | 123.5 | 123.4 |
| 0.01 | 126.0 | 126.4 |
| 0.05 | 136.8 | 140.7 |
| 0.1 | 152.4 | 150.1 |
| 0.2 | 192.9 | 187.1 |
| 0.3 | 252.0 | 256.6 |

**References**

Pennsylvania State University (2017). Key Properties of a Negative Binomial Random

*Variable*. Retrieved on 30th August 2017 from

https://onlinecourses.science.psu.edu/stat414/node/79

Weisstein, Eric W. (nd). *Negative Binomial Distribution.* Retrieved 30th August 2017 from

http://mathworld.wolfram.com/NegativeBinomialDistribution.html