COSC 264 - Assignment

User codes: 48056985, //Morgan’s

*Source Code*

*Questions*

1. **The protocol between sender and receiver as described above has (at least) one**

**weakness: it has a deadlock. Please explain the notion of a deadlock in the**

**context of networking protocols and describe the particular deadlock situation in**

**our case. A guiding question is: what can go wrong and when in case certain**

**packets are lost?**

A deadlock occurs when two processes sharing a singular resource. When the use of this conflicts such that no process can access the resource it becomes a deadlock.

A deadlock refers to the problem the occurs when two programs are using the same resource and therefore, prevent each other from using the resource as both are contesting it. In our case, the sender is sending programs from the same port that is is being read from. I our case, a deadlock that can occur is if enough packets are lost in quick succession such that both sender and receiver are waiting on packets and therefore do not send any packets themselves. This will cause both to time-out, resulting in both sending packets at similar times and therefore not sending the packet in the end. This problem can be resolved by staggering the delay between the sender and the receiver. However, this will not remove the chance for a deadlock to occur, but rather allow it be self-resolve provided that the packet isn’t completely dropped within this time frame.

1. **What is the magicno field good for?**

The magicno field is a random hexadecimal number that gives an easy way to ensure that the packet is what is being expected. By acting as a random number that would be only known to the sender and receiver, it allows their communication to be secure. Therefore, it is also useful in the checksum as it is a value that is expected to be constant throughout all packets.

1. **How have you solved the issue with the bit errors? Please explain what you have**

**added to the packet and to the sender and receiver modules**

To solve the problem with bit errors we added a header checksum to the packet type. This checksum is simply an addition of all other header fields. This meant that in the sender and receiver we had to add a check that the packet checksum was indeed the same as the sum of all other header fields for each packet. If not the packet must have had a bit error introduced and would be dropped.

1. **Please explain what the select() function is doing and why it is useful for the**

**channel (and in another way for the sender).**

The select function allows a program to wait on file descriptors or similar (such as sockets in our case) until they become ready. That is, when it becomes possible to perform I/O operations without blocking

The select function (select.select) in python mimics the select function in linux in that it allows for easy waiting of inputs and outputs. This is very useful for the channel as without it it becomes difficult to properly buffer any inputs and send them to the receiver. For sender this is useful as it makes sure that the packets will arrive in the same order sent assuming that they are buffered properly by the channel. This waiting on inputs is what makes the channel function properly as otherwise it would be difficult to guarantee in sequence delivery of the packets from sender.

1. **Please explain how you have checked whether or not the file was transferred cor-**

**rectly (i.e. the receivers copy is identical to the transmitters copy).**

To check that the file was copied exactly we looked at the properties window in the file view for the outfile and made sure it had the exact number of bytes as the infile. To complement this, we also had “START” and “END” in the files respectively and checked to make sure these were copied where they should be.

1. **We consider different packet loss probabilities of P f0:0; 0:01; 0:05; 0:1; 0:2; 0:3g**

**and a source file of length M = 512 \* 100 = 51,200 bytes (you need to create such**

**a file). For each value of P make ten repetitions of the file transfer and for each repetition record how many packets the sender has sent in total. Draw a graph that shows the different values of P on the x-axis and for each such value the average number of total packets (the average being taken over the ten repetitions) on the y-axis. Explain the results.**

//Chuck in data from 10 runs into excel? Seems like a good enough solution

1. **Assume the following:**

* **The probability to lose an individual packet (either a data packet or an acknowledgement packet) is P.**
* **Packet loss events are statistically independent of each other.**
* **The size of the file requires N packets.**

**Please derive and justify an expression for the average total number of packets that need to be sent (including retransmissions) to transmit the entire file. Compare this to the (average) total number of packets you have observed in your experiments.**

Since we need to send N packets of data and therefore N acknowledgement packets as well, the total number of packets sent is 2N. As the probability of packets being lost is P, we expect 2N \* P packets to be lost and therefore need to be retransmitted. The 2N \* P packets can again be lost when being resent and therefore suffer an additional 2N \* P \* P packets required to transmit them. Therefore, we will need to send the following number of packets:

Total Packets = 2N +

Comparison to our average:

*Contribution Form*

//Just a table? Don’t need to mention specifics so just a percentage value. Maybe just a couple of lines of text?