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?

In networking, a deadlock is when two programs both are waiting for input from each other, and so neither can complete their respective actions. In our instance, when the final ACK packet is dropped/lost, the "Sender" program cannot tell that its last packet has been received, and so it keeps attempting to send packets, getting no acknowledgements since the receiver has already finished.

2. What is the magicno field good for?

MagicNo is used to make sure it is our program that sent the received packet. Since we are using UDP, any packets sent to our port will be accepted regardless of the sender.

MagicNo can also serve sort of as a checksum value. If, during transmission, the MagicNo field is changed, then we should drop the packet since we cannot be sure. However, it is much more likely that the data is modified than the MagicNo field, so it can't guarantee the data isn't modified.

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

select() is a blocking call that waits on one or more sockets until they have data ready to be read. In other words it multiplexes sockets for synchronous IO. It is useful for the channel because the channel has two incoming sockets, and we only want to wait until one of them has data. It is useful for the sender because it allows us to set a timeout to wait for an acknowledgement, whereas recvfrom() would block forever.

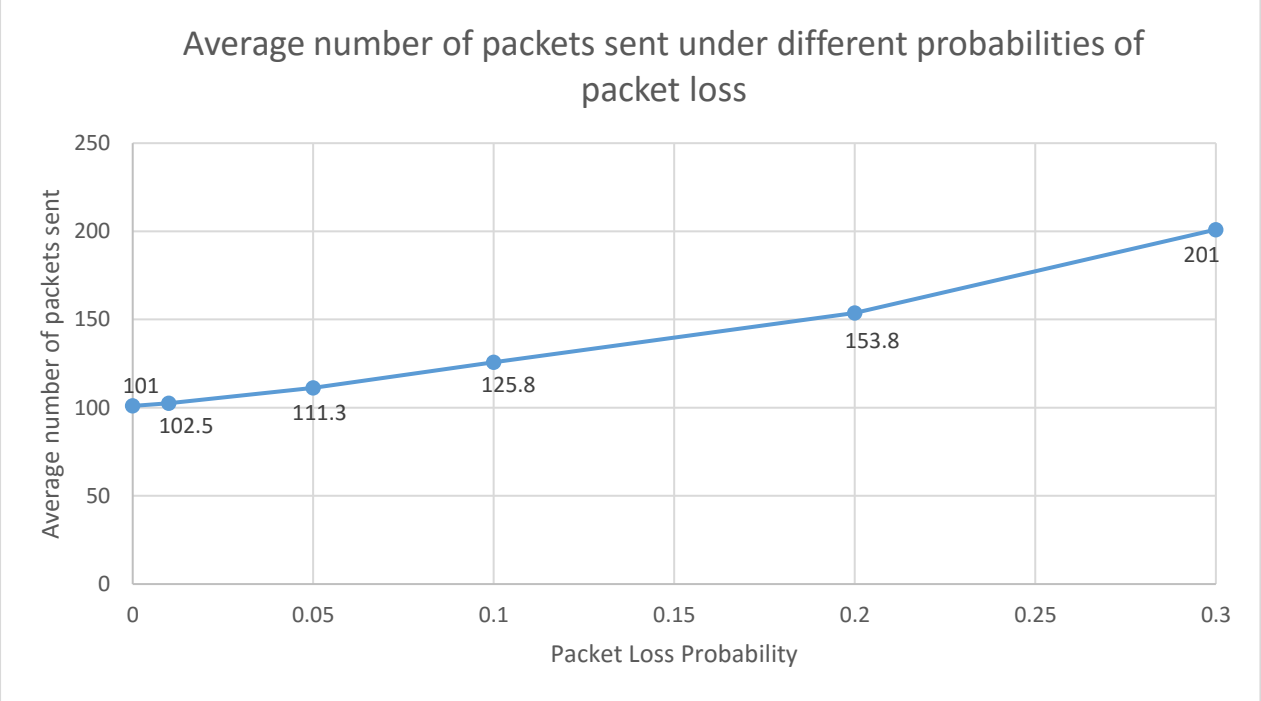
4. Please explain how you have checked whether or not the file was transferred correctly (i.e. the receivers copy is identical to the transmitters copy).

We have used an md5 checksum on the input file and the output file. This gets run after transmission by our "test.py" program. if the md5 sum is the same then it is almost ceratin that the files are the same.

5. We consider different packet loss probabilities of P = 0.0, 0.01, 0.05, 0.1, 0.2, 0.3, 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. Note: To produce graphs, the tool gnuplot can be useful under Linux. Its main advantage is that it allows for script-based (i.e. non-interactive) creation of graphs, but admittedly its command syntax needs some getting used to. However, you are free to use any tool you like (including Excel, Matlab, etc.) for producing graphs. Under all circumstances you need to make sure that axes and curves are properly labeled.

Table 1: Number of packets sent from 10 trials of each packet drop probability.

P	Avg	1	2	3	4	5	6	7	8	9	10
0	101	101	101	101	101	101	101	101	101	101	101
0.01	102.5	103	101	102	104	105	102	102	102	102	102
0.05	111.3	112	121	111	108	104	110	112	112	110	113
0.1	125.8	122	116	122	130	124	129	131	123	126	135
0.2	153.8	153	154	148	151	148	159	168	155	145	157
0.3	201	184	226	238	208	197	202	173	201	182	199



6. Assume the following:
-- The probability to loose an individual packet (either a dataPacket or an ackPacket) is P.
-- Packet loss events are statistically independent of each other.
-- The size of the file to be transmitted 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.

require a resend when either DATA or ACK packet dropped

drop probability = P

probablity of a successful send is the probablity that we successfully sent both the DATA and ACK packets

probability of not dropping a packet = (1-P)

probability of successful send = (1-P)(1-P)

The average number of times we will need to send a packet is a discrete random variable following a geometric distribution.

The geometric distribution formula is: $E(X = k) = (1-P)^{k-1}P$

Thus the number of sends (including all failures and the first success) is:

resend count = 1/(probability of successful send) = 1/(1-P)(1-P)

Therefore the average number of packets we will need to send is:

av number of packets = N*(resend count) = **N/(1-P)(1-P)**

Using some examples to demonstrate:

N = 101, P = 0.3

N/(1-P)(1-P) = 101/(1-0.3)(1-0.3) = 101/(0.7)(0.7) = 101/0.49 = 206.122

Experimental result: 201

N = 101, P = 0.1

N/(1-P)(1-P) = 101/(1-0.1)(1-0.1) = 101/(0.9)(0.9) = 101/0.81 = 124.69

Experimental result: 125.8

N = 101, P = 0

N/(1-P)(1-P) = 101/(1-0)(1-0) = 101/1 = 101

Experimental result: 101