Computer networks

In this document I'm showing the result of my network. The results will be split between the different modes : RELIABLE, UNRELIABLE

All tests happened in localhost using ip: 127.0.0.1

All expensive operations (creating the file, closing the file stream, writing to the file) happen after the receiver receives the End of Transmission flag. I made sure that instead of reading the entire file we read parts that we can (depends on MDS) so that the sending of the packet happens as soon as possible so that we do not trigger the timeout exception.

Execution time is the time that we start the sender after sending the first packet minus the time that we start printing the result in the console. So we can use the execution time as a near estimate of the actual execution time. We can most likely speed the exécution up by removing all console logs and/or by optimising the code.

For all incoming packets we use a TreeMap<Integer, byte[]> to make sure that all packets are received in order. If we are too late to send an ACK we will receive the same packet so by using a TreeMap we can make sure that the old value will be replaced by the new value.

Reliable mode results

Timeout in µs	MDS in bytes	File size in bytes	Execution time in ms	#packets(lost)
20000	400	1219	19	5(0)
20000	200	1219	33	8(0)
20000	100	1219	41	15(0)
20000	400	202006	585	519(0)
20000	200	202006	825	1065(0)
20000	100	202006	925	2246(0)

Unreliable mode results

Timeout in µs	MDS in bytes	File size in bytes	Execution time in ms	#packets(lost)
20000	400	1219	2	5(0)
20000	200	1219	5	8(0)
20000	100	1219	31	15(1)
20000	400	202006	1244	519(51)
20000	200	202006	2495	1065(106)
20000	100	202006	5169	2246(224)

Conclusion

The timeout doesn't impact the transfer time when you can guarantee that there will be no packet loss. But the moment there is any you can expect a wait time equal to the timeout because that is the time that the Sender resends a packet. This means that the bigger the timeout the longer the execution time will be.

With the MDS you can reduce the amount of packets sent by packing more data in each packet to the Receiver which will in turn lead to smaller execution times.