File Transfer Protocols - Marking Scheme

	Registration Number	Surname	Forename	% Contribution
Student 1	000839875	Turon	Kacper	25%
Student 2	000871936	Young	Jack	25%
Student 3	000878481	Kallon	Kevin	25%
Student 4	000874782	Basharat	Usman	25%

Task	Mark	Grade
Task 1 - Table 1	5	
Task 2 - Table 1	5	
Task 3 - Table 1	5	
Task 4 - Graph 1	20	
Task 5 - Graph 2	20	-
Task 6 - Graph 3	30	
Task 7 - Reflection	15	
Total	100	

COMP1587 Communication Systems

Week 4 Laboratory

File Transfer Protocol

Usman Basharat - 000874782

Kevin Kallon - 000878481

Jack Young - 000871936

Kacper Turon - 000839875

Contents

Abstract

File transferring over two PCs is one of the first and still the most important part of computer networks. The purpose of this laboratory is to get grips with how to transfer files over two PCs (Greenwich, 2015). The key aspect of this laboratory was motivation. Task 1 was about recording the status of the breakout box before wiring; Task 2 was after and Task 3 was whilst the Procomm Plus was running. As we moved on, we completed each task as a group. We did everything on time. Whilst one tasks were complete, one member of the group was on to the calculation of it. We kept moving at a good pace and did everything on time.

Introduction

In this report, we will explain how we transferred files over two PCs using different protocols. We all participated in each task and by following the guide, it was easy to understand and we all knew what to do. Once these tasks were complete, we had to reflect upon how each task went and how to improve on it.

Methods and Materials (or Equipment)

Equipment/Materials used:

- EIA-232 Breakout Box
- Two PCs
- Patch Lead cables
- Null modem cable
- Guide (Greenwich, 2015)

Software used:

- Procomm Plus – we used this to transfer files from one PC to another.
- Stop Watch – we used this to time how long each file took in seconds.

Experimental Procedure

For this laboratory session, we followed a guide provided by the University of Greenwich that told us what to do step by step. All steps were followed; no extra steps were taken when doing the tasks that were based on the guide.

Results

Task 1, 2, 3 - Table 1

Acronym	Function	Status Before Wiring		Status After Wiring		Final Status with Procom Plus Running	
		Left	Right	Left	Right	Left	Right
CD	Carrier Detect	0	0	-	-	+	-
RD	Receive Data	0	0	-	-	-	-
TD	Transmit Data	-	-	-	-	-	-
DTR	Data Terminal Ready	-	-	-	-	-	+
SG	System Ground	0	0	0	0	0	0
DSR	Data Set Ready	0	0	-	-	+	-
RTS	Request to Send	-	-	-	-	-	+
CTS	Clear to Send	0	0	-	-	+	-
RI	Ring Indicator	0	0	-	-	+	-

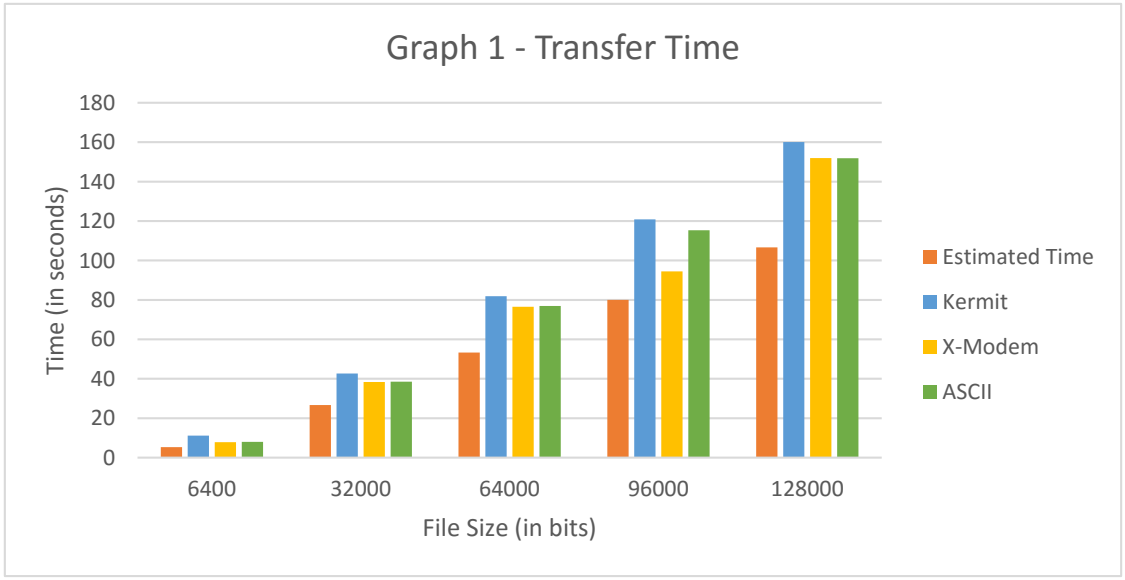
Key	
-	Negative Result
+	Positive Result
0	No Voltage

Table 1 Voltage output of the left and right side of the breaker box

Task 4 - Graph 1

File	File Size (in bits)	Time (s) - Calculated	Time (s) - Kermit	Time (s) - X-Modem	Time (s) - ASCII
800.DAT	6400	5.333	11.138	7.817	7.974
4000.DAT	32000	26.666667	42.64	38.37	38.511
8000.DAT	64000	53.333	81.904	76.565	76.927
12000.DAT	96000	80	120.867	94.449	115.362
16000.DAT	128000	106.6666667	160.09	151.932	151.917

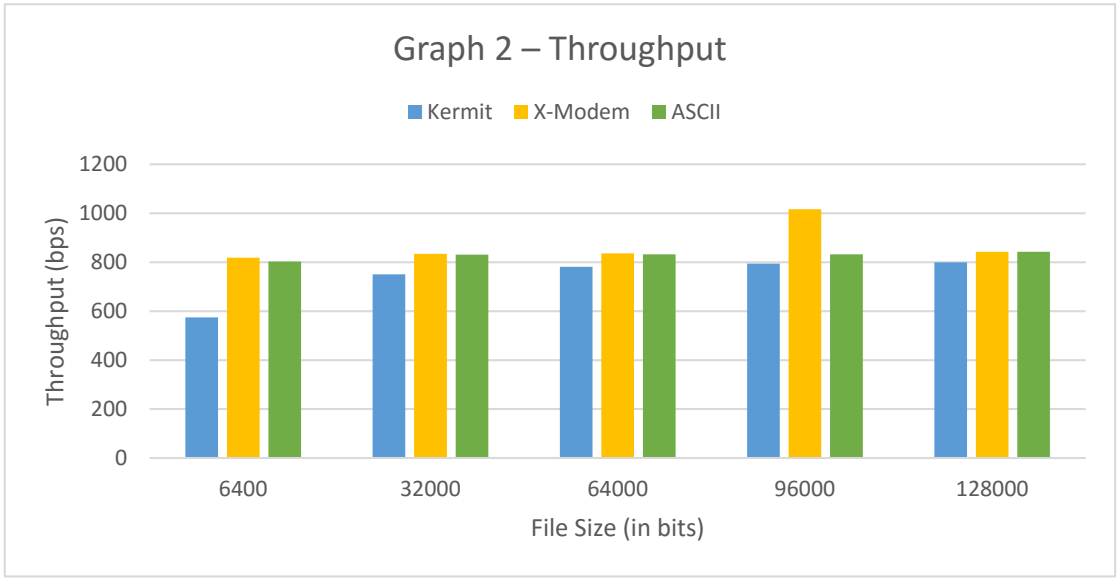
Table 2 - Transfer Time of different file sizes



Task 5 - Graph 2

File	File Size (in bits)	Line Speed (bps)	Throughput (bps) - Kermit	Throughput (bps) - X-Modem	Throughput (bps) - ASCII
800.DAT	6400	1200	574.609	818.728	802.609
4000.DAT	32000	1200	750.469	833.984	830.931
8000.DAT	64000	1200	781.402	835.891	831.958
12000.DAT	96000	1200	794.261	1016.422	832.163
16000.DAT	128000	1200	799.55	842.482	842.565

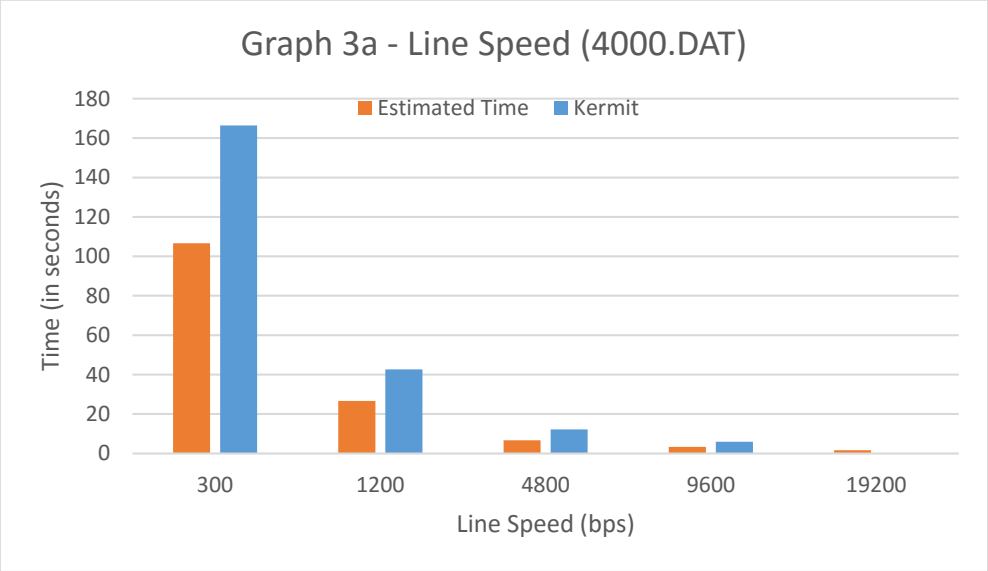
Table 3 - Throughput of different file sizes

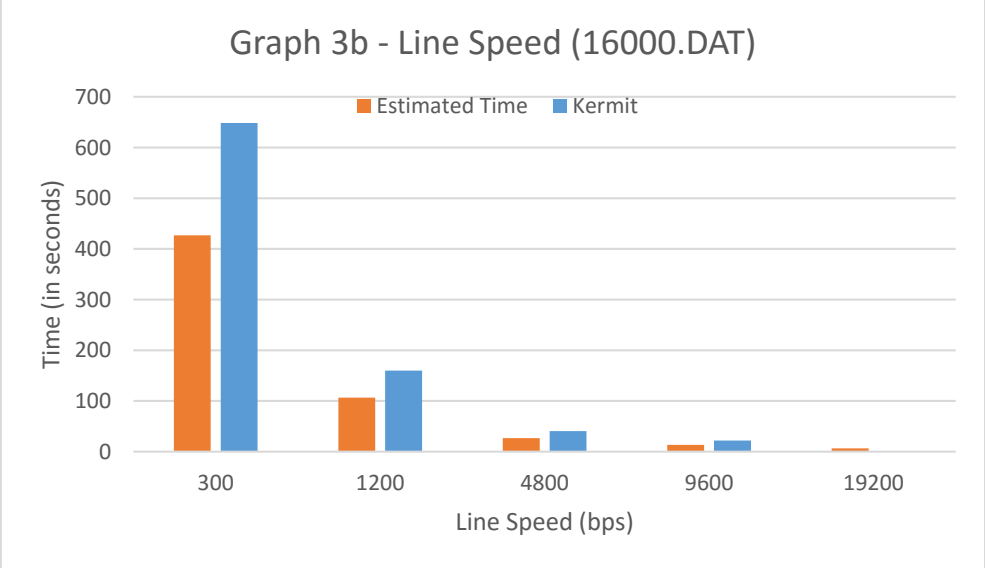


Task 6 - Graph 3

Line Speed (bps)	4000.DAT Time (s) - Calculated	4000.DAT Time (s) - Kermit	16000.DAT Time (s) - Calculated	16000.DAT Time (s) - Kermit
300	106.6666667	166.428	426.6666667	648.573
1200	26.66666667	42.64	106.6666667	160.09
4800	6.666666667	12.179	26.66666667	40.668
9600	3.333333333	5.963	13.33333333	22.088
19200	1.666666667	Timed Out	6.666666667	Timed Out

Table 4 - Different Line Speeds for transferring 4000.DAT and 16000.DAT





Discussion

Task 7 – Reflection

"One of the first and still one of the most important uses of computer networks is the ability to transfer files from one system to another" (Greenwich, 2015).

This laboratory session on file transfer protocols helped us understand the basics of file transfer. During the session, we used 3 different protocols called: Kermit, X-Modem and ASCII. Each protocols had different ways of transferring files but the output was the same, the file that reaches the end computer will be the same as the start computer no matter what protocol you use. This will help us in our degrees as this is our first time that we physically setup a system which transfers files locally across cables. As a group, we have all transferred files; whether it is through a USB stick or as an attachment to an email, but in this laboratory session, we learnt how to use the breakout box to transfer files with different functions that the box has.

For *Task 1*, we were tasked with connecting a breakout box to two computers. To do this, we had to connect two null modem cables into the opposite ends of the breakout box; this served as the basis of powering up the breakout box as well as confirming that the box would transfer data between the two PCs as shown on Figure 1.

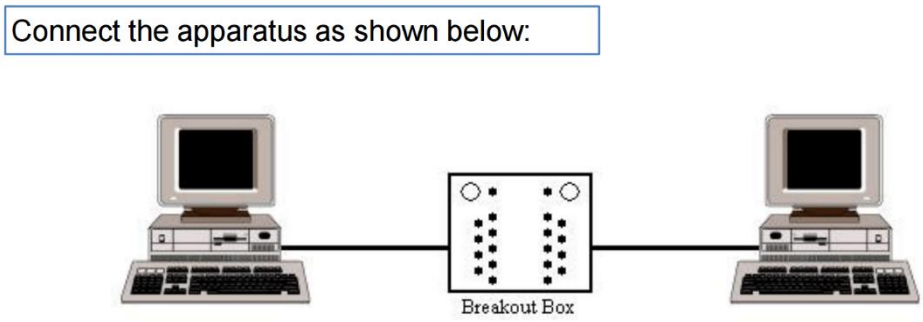


Figure 1 (Greenwich, 2015)

After completing this, we had to complete the first column of Table 1, which required us to record the status of the breakout box before wiring. To do this, we connected just one patch lead cable onto the very first hole, we were uncertain if the breakout box was functional as the first few holes did not give any results. Later, we realised that those holes just had no voltage. Completing the rest of this task was relatively simple as we tested out every hole and recorded the results. We noticed that although we tried out every hole, there was no result that yielded a positive voltage. To rectify this, we started the tests all over again however we simply got the same results as before so naturally, we had to assume that this was an intended feature of Table 1.

We had no troubles in doing Task 1, and there was little that we actually needed to improve about the way we went about completing this task.

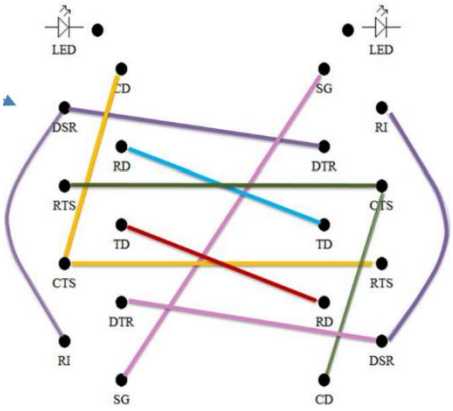


Figure 2 (Greenwich, 2015)

For *Task 2*, this activity was where things got a bit tricky. At first, we were fixated on the idea that we had to wire the box and gain results one at a time, but we quickly figured out that was not the case. Referring to Figure 2, as one of us brought up the idea that we must replicate the diagram as shown in the diagram from our brief, we quickly got confirmation by reading the brief again, it said, “Wire the breakout box as a Null Modem” (Greenwich, 2015). A line at the end of that small sentence pointed towards the same diagram and from there the

task got a little easier. After completing the wiring, we did tests that allowed us to complete the second column of Table 1 which required us to record the status of the breakout box after wiring but without ProComm plus running. We noticed that some of the holes had changed their results from the previous table. At the end of this task, we noticed that quite a few of the tables had stayed the same such as the TD function, the DTR function, the SG function, and the RTS function.

In this task, we could improve more by reading the brief more clearly before attempting the work as if our colleague did not interject we could've been set back by incorrect data.

For *Task 3*, we had to record the status of the breakout box whilst the Procomm Plus software was running. We ran the program and all we did was record the results in the third column of Table 1. Whilst the software was running, we were unsure if it was working as nothing appeared on the computer screen. However, to our surprise, this was intentional and what the software actually did was change the voltage outputs of the breakout box. We repeated what we did for Task 1 and 2 and test to see if the voltage was positive, negative or no voltage. We got different results from Task 1 and 2 which we assumed that this was correct, we did another test to check for any errors that may have occurred.

For *Task 4*, the first step we did is saving the files that we had to transfer in our own area for us to send the files to the other PC. Once this was complete, we used different protocols and timed each one with the given stopwatch software for the files to be successfully transferred. Some of the protocols took slightly longer than other protocols. Looking at Graph 2, we can see that the X-Modem protocol took less time than Kermit and ASCII. We read the guide to help us on anything that we were stuck on. For example, how to transfer the files. It said on the guide what to press, PageUP on one computer to upload a file and PageDOWN to download the file on a different computer.

To improve this, we would put more protocols to test. Maybe another slow protocol to test so we can compare each one. This would take longer, but we can analyse how file transfer can be done with four protocols used.

For *Task 5*, we used data from the previous task to calculate the file size of the given .DAT files. We divided the file size in bits by the time we have recorded on our stopwatch in seconds to get the answer in bits per second and we followed those steps with every transmission protocol (Kermit, X-modem and ASCII). We used the formula provided in the guide to calculate the throughput of the data.

"**Throughput = (File Size * 8) / Time Taken**" (Greenwich, 2015)

So for example, the first file was 800.DAT which was 800 bytes, times this by 8 to give us 6400 bits. It took us 11.138 seconds to transfer via Kermit protocol, so we divided 6400 by 11.138 which calculated to be 574.609 bits per second (bps). This data can be found on Table 3 and is shown on Graph 2.

b) Kermit first sends a Send-Initiate packet to specify its parameters (length, timeout, etc.). Then the receiver sends ACK (Acknowledgement) packet. Then sender sends File-Header packet after that sender sends contents of the file, after all data has been set, the sender sends an End-Of-File packet the receiver ACKs it.

XModem sends first initiation packet then sender sends the data with checksum, then sender sends the ACK packet and again sender sends data with checksum then sender sends packet till the end of the file.

c) Different protocols have different speeds because they use different algorithms. For example X-modem is using checksum/CRC to validate that the data received isn't corrupted or didn't all go through so it sends packets in blocks of 128/1024 bytes each which slow down the transfer overall.

Kermit by default, does not assume that control characters pass through transparently, nor that large buffers are available. It does not even assume a full-duplex connection. The trade-off is speed. Kermit is supposed to work under any conditions without any tweaking in comparison to X-modem/Z-modem etc.

ASCII is quite fast since it doesn't use any protocol 'handshake' or error correction or data compression it just sends through a string of characters for 7 bit connections.

For *Task 6*, to calculate how long it would take to transfer 4000 and 16000 bytes files we changed to bits hence the line speed was given in bps (300 bps, 1200 bps, 4800 bps…). 4000 and 16000 bytes equal to 32000 and 128000 bits respectively and then we divided it by given line speed values 300 bps, 1200 bps etc. so 32000 divided by 300 bps gives us 106.666 seconds and 128000 divided by 300 bps gives us 426.666 seconds. We repeated this equation for the rest of the line speeds. For the column that asked for Kermit protocol transfer time, we changed the line speed in the Procom Plus Software to each bps in the table and sent the 4000.DAT file and used stop watch to get the time in seconds so for 4000 bytes with Kermit protocol and 300 bps line speed it took 166.428 seconds. All this data can be found on Table 4 and is shown on two graphs (Graph 3a and 3b) which shows the time in seconds of how long it took for the data file to transfer using different line speeds.

When we tried to transfer the .DAT files for both 4000 and 16000 using the 19200 bps speed, it gave us an error message displaying "Timed Out". This means that the two PCs are unable to make a connection in a given time. A possible reason for this may be because the break out box does not support speeds that go to 19200 bps as it may be too fast for the box to handle and therefore does not allow us to transfer between PCs.

For our data in particular, when the line speeds double, the time it takes roughly halves. For example, if we take the 300 bps data for 4000.DAT on Table 4, the number is 166.428. To get from 300 to 1200, we double twice. So, to estimate how long the 1200 bps will take using the 300 bps result, we half it twice which will give us 41.607. The actual result for the 1200 bps is 42.64. This means that the line speeds took roughly half the time if it is double the speed. We would probably use the 9600 bps line speed to transfer files as it faster and more reliable because it did not give us any error message when we were timing how long it took. This means that we do not have to wait around for a file to be transferred as it does not take that long at all.

To improve the laboratory session for next year, I would skip timing how long it took for the .DAT files to transfer using the 300 bps speed as it wasted valuable time.

Conclusion

In conclusion, we felt that we understood the concepts of file transfer protocols and know how to use the breaker box to connect two PCs together. In addition to this, we got familiar with using the Procom Plus software which enabled us to transfer the files.

References

Greenwich, University of. (2015) *File Transfer Protocol*, 1st ed, Greenwich, University of Greenwich, [online] Available at: http://staffweb.cms.gre.ac.uk/~lg47/lectures/COMP1587/COMP1587Lab4.pdf (Accessed 27 October 2015).