

File Transfers – Windows vs. FTP

By Michael J. Sepcot and Robert D. Brozyna

BACKGROUND

METHODOLOGY

RESULTS

Our results are broken up into two sections: the transfer rates between small to medium sized text and executable files and transfer rates between a single large file and multiple medium sized files. We measured a variety of statistics during file transfer ranging from the total time to transfer files to the total number of packets sent and the average number of megabits received per second on the client side. We also measured the processor utilization in the client and server computers using both the FileZilla file transfer protocol and the built in Windows transfer protocol.

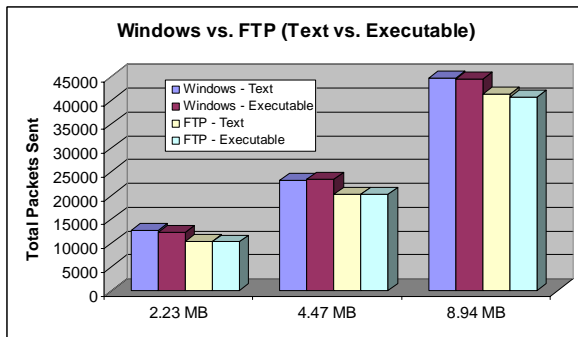


Table 1 – Total Packets Sent – Text vs. Exe

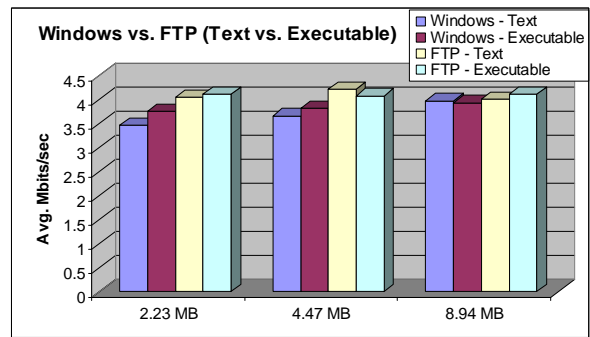


Table 2 – Average Mbits/sec – Text vs. Exe

Table 2 shows the text versus executable comparison of the average number of megabits per second sent via Windows and FileZilla. Table 4 shows the comparison between a single large file and multiple medium sized files in the same statistic scenario. It appears that FileZilla throughput rates remain more or less stable regardless of size for medium sized files. For Windows, throughput increase with file size for smaller text files, but remains stable for binary files. The throughput for both Windows and FileZilla drop substantially for large file sizes. This is most likely due to the reduction of the window size to reduce the number of retransmits required in transferring the files.

Table 1 illustrates the total number of packets sent from the server to the client when transferring medium sized text and binary files in the Windows and FileZilla environments. With the window size lower in the Windows environment, it is no surprise that across the board, Windows needs to transfer more packets in order to complete the file transfer.

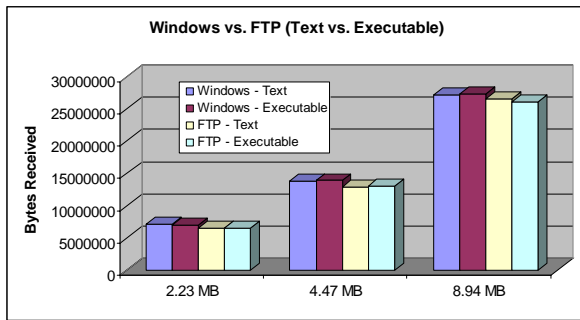


Table 3 – Bytes Received – Text vs. Exe

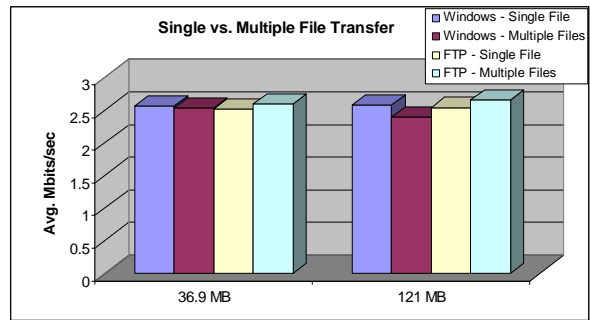


Table 4 – Average Mbits/sec – Single vs. Multiple

Table 3 illustrates the number of bytes received at the client side in the text versus binary comparison. The total number of bytes received doubles as expected as file size doubles. The Windows environment has a slightly higher number of total bytes transferred. This is probably due to the increased number of packets that needs to be sent because of lower window sizes in the Windows environment. Low window size means that more packets need to be sent. Since more packets are being sent, a higher overhead leads the Windows environment to having a slightly higher number of bytes received on the client side.

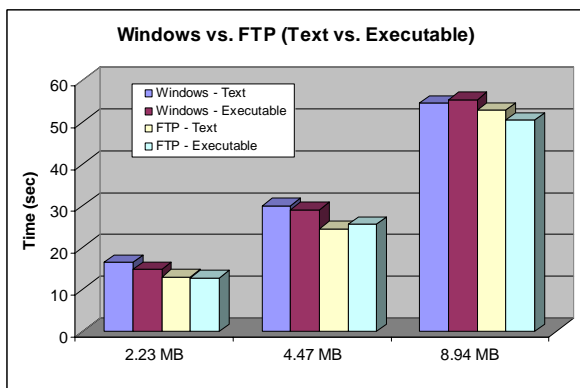


Table 5 – Time (sec) – Text vs. Exe

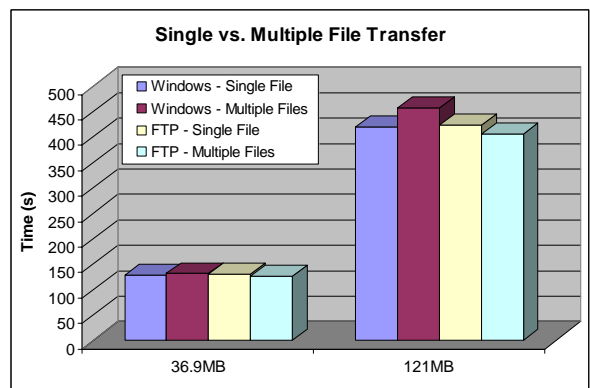


Table 6 – Time (sec) – Single vs. Multiple

Table 5 shows the total delivery time comparison between Windows and FileZilla for both text and executable files. The graph in table 5 shows the total delivery time doubling as expected as file size doubles. FileZilla is shown to be faster at transferring medium sized text and executable files than Windows alone. But, comparison between text and binary transfer rates themselves is inconclusive. It is not clear whether or not a different transfer protocol is being used to transfer text and binary files.

Table 6 shows the total delivery time comparison between Windows and FileZilla for a single large file and multiple medium size files. From the graph, we can conclude that it takes Windows longer to transfer multiple medium files than it does to transfer the same amount of data as a single large file. The reverse is true for FileZilla's file transfer protocol; it takes longer for the single large file to be transferred than multiple medium sized files. This might be because the FileZilla client was downloading the medium files concurrently (two at a time). This would

better utilize the available bandwidth and complete the download quicker. Windows, on the other hand, downloaded the medium sized files in sequence allowing one to completely download before starting the next. While Table 5 showed that FileZilla was better at transferring single medium sized files, Table 6 shows that as the file size increases, Windows becomes better at transferring single files.

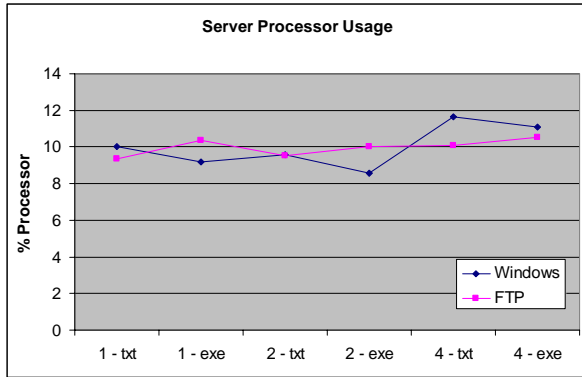


Table 7 – Processor Percentage - Server

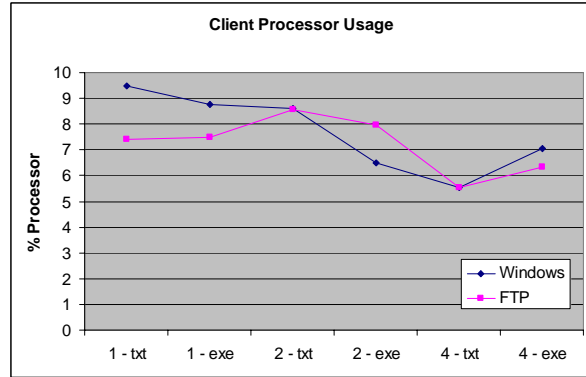


Table 8 – Processor Percentage - Client

Table 7 shows the processor utilization on the server side, and Table 8 shows the processor utilization on the client side. Both tables graph the comparison between Windows and FileZilla. Neither graph shows any conclusions between Windows and FileZilla processor utilization on the client or server side. Text and executable comparison is also not conclusive.

CONCLUSIONS

FURTHER WORK

To further expand this work, we would like to expand the file sizes to identify the point in which (if one exists) where Windows transfer protocol is better than FileZilla's implementation of the file transfer protocol. We would like to expand our comparison of transfer rates to different FTP clients such as SecureFTP which handles binary and text transfers differently. Expanding even further would take us to compare operating systems including Linux, Unix, Mac OSX, and others.