TCP & UDP ANALYSIS

A comparison of the two most popular protocols.

TCP may not be as slow as most people think, while UDP may not be as unreliable as previously thought when using smaller packet sizes.

Contents

Introduction

TCP and UDP are the most common protocols used in modern society.

TCP is known as the reliable, albeit slow, protocol of the two. It is traditionally used in situations where you cannot afford to miss data, and where every packet matters.

UDP is known as the fast, albeit unreliable, protocol of the two. It is traditionally used in situations where speed is more important than missing data, or where missing data is irrelevant.

The goal of this project was to put these common beliefs to a test and truly analyze the differences between TCP and UDP in identical applications. This document represents the findings from data gathered via the Protocol Analyzer. The findings here are derived from the data gathered, and not any other data. This means that the findings may not be perfect, as they are at the liberty of the implementation of TCP and UDP within the Protocol Analyzer tool. Further testing is required to guarantee the findings are indeed accurate.

Executive Summary

TCP was the faster protocol in every single comparison between TCP and UDP. TCP was not only faster, but had the huge benefit of no packet loss, which hurt UDP tremendously.

Analysis

The following sections will compare various aspects of both TCP and UDP. First, we will look at how the two compare how their efficiency changes depending on their packet sizes. We will then compare the two together and see how efficient the two are.

Afterwards, we will compare how they behave in regards to the amount of packets being sent and see if how the amount of packets changes their behavior.

Finally, we will look at the packet loss of UDP among all of these variables.

Comparison of Size

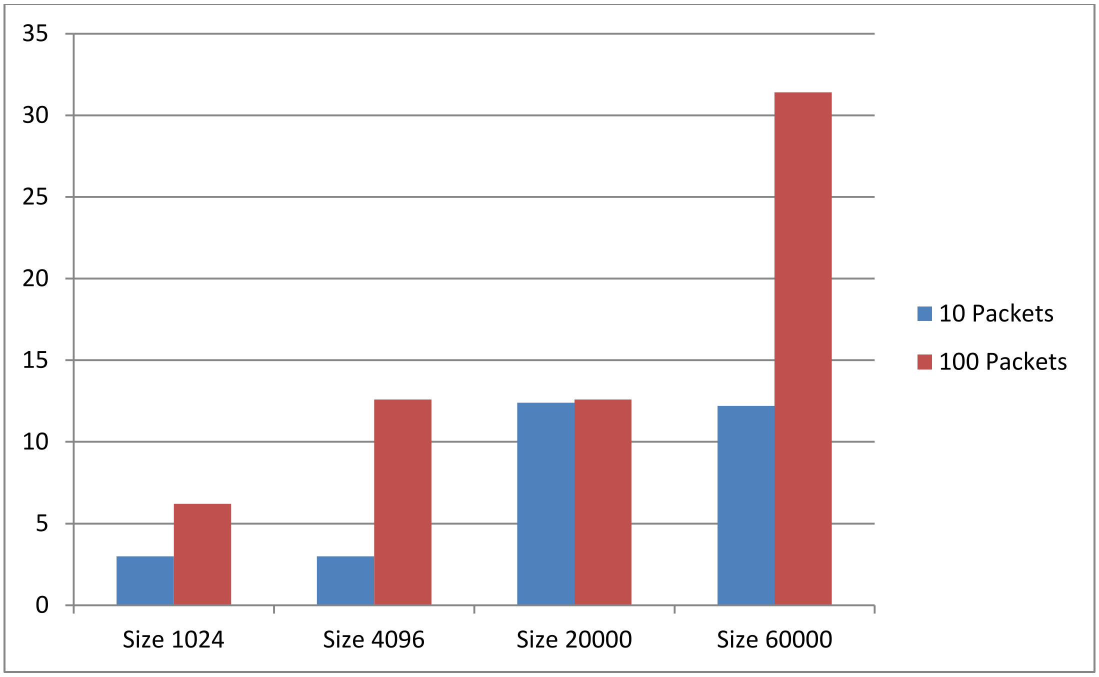

Figure 1. TCP Transfer Time

Figure 1 above will show us the average transfer time of the TCP protocol over four different packet sizes and two different quantities of packets. We will be focusing, at this point in time, on the sizes. As to be expected, the larger the packet, the longer it takes to send data.

Focusing on those of packet size 10, we can see that the difference in transfer time between size 1,024 and 60,000 is roughly a difference of 400%. What is peculiar, however, is there is virtually no difference between 1,024 and 4,096 in transfer time, as well as 20,000 and 60,000, despite the former being a 400% difference and the latter being a 300% difference.

On the 100 packets side, we see a similar happenstance, however 4,096 and 20,000 are the two points that are virtually identical. Regardless, in TCP we see the increase in packet size has a consistent increase in transfer time.

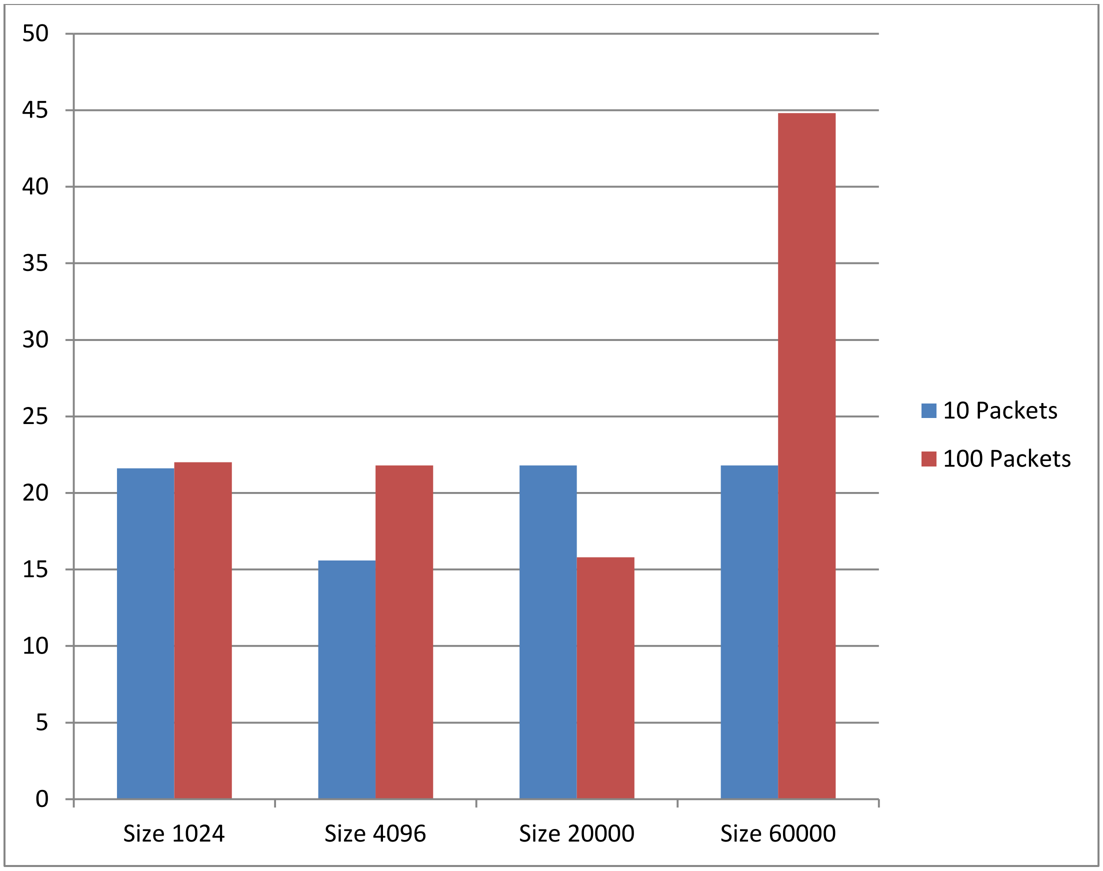

Figure 2. UDP Transfer Time

Figure 2 above will show us the average transfer time of the UDP protocol over four different packet sizes and two different quantities of packets.

Unlike TCP, UDP does not see a consistent increase in transfer time when sending bigger packets. When focusing on the blue bars, which represent 10 packets being sent, we see that transfer time actually decreased on averaged between the 1,024 byte sizes over to the 4,096 byte sizes. Afterwards, it increases back to the speed we had with 1,024 for both the 20,000 packets and 60,000 packets.

What this behavior tells us is most likely not that 4,096 is somehow faster. It is most likely an error in the program used to demonstrate this data, or a sign of too small of sample size. What it does tell us, however, is that the packet size difference in UDP does not seem to make a huge difference when sending view packets. The overhead of setting up the send and receive calls is higher than the transfer time of the data, and thus they all have very similar showings.

On the 100 packets sent side, we see similar behavior until the 60,000 bytes mark in size. The overhead of setting up the transfer is most likely more work than the transfer itself, meaning we see similar performance across most transfers. The outlier, the 100 packets transfer of 60,000 bytes being higher, is additional proof of this idea. Once we sent 60 times the smallest amount of data, we saw an increase in transfer time of 200%.

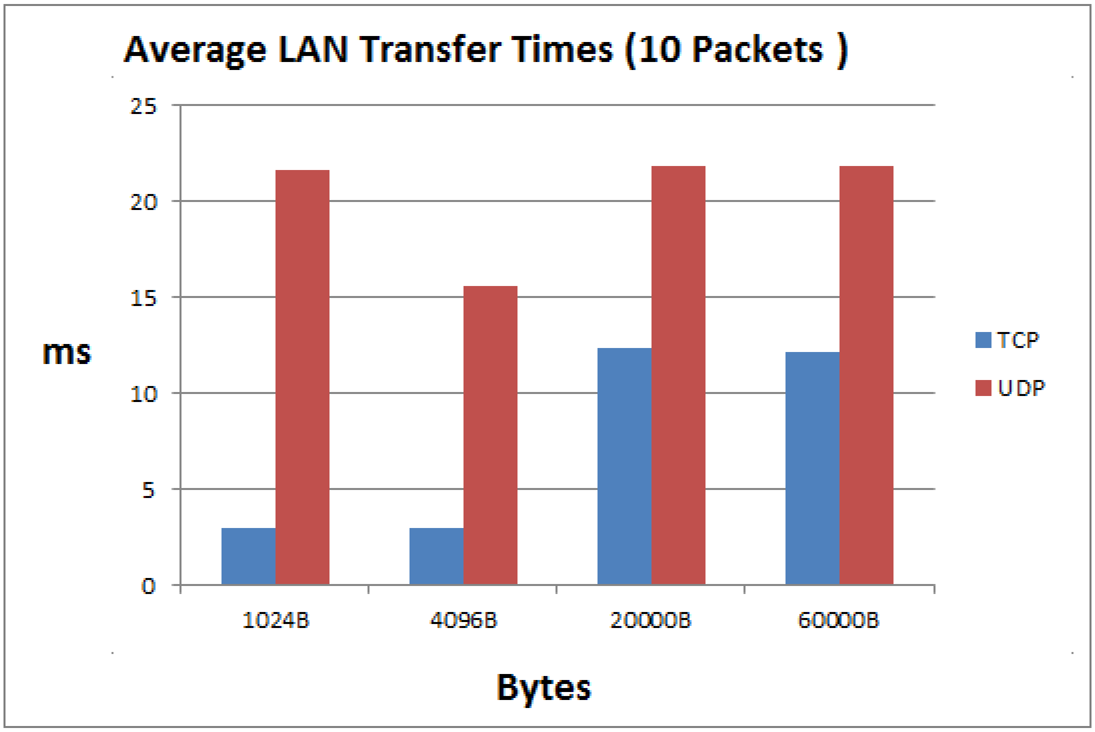

Figure 3. Average LAN Transfer Times (10 Packages)

Figure 3 above shows us an average comparison of TCP and UDP in the situation of sending a 10 packet burst. This is the same data from Figure 1 and Figure 2.

Once we align both TCP and UDP beside each other, we see that UDP has higher transfer times across the board. UDP, as stated before, does not seem to differ in transfer time when we increase packet size. TCP on the other hand increases drastically as packet size increases. From this data we can concur that TCP has less overhead when sending data after being connected, while UDP has the same overhead regardless of the quantity of packets being sent. Assuming implementations were equally efficient; TCP is also more efficient in every situation listed above.

Comparison of Quantity

Continuing on the figures 1, 2 and 3 above, we will now look at the difference in how the protocols behave in regards to the amount of packets being sent.

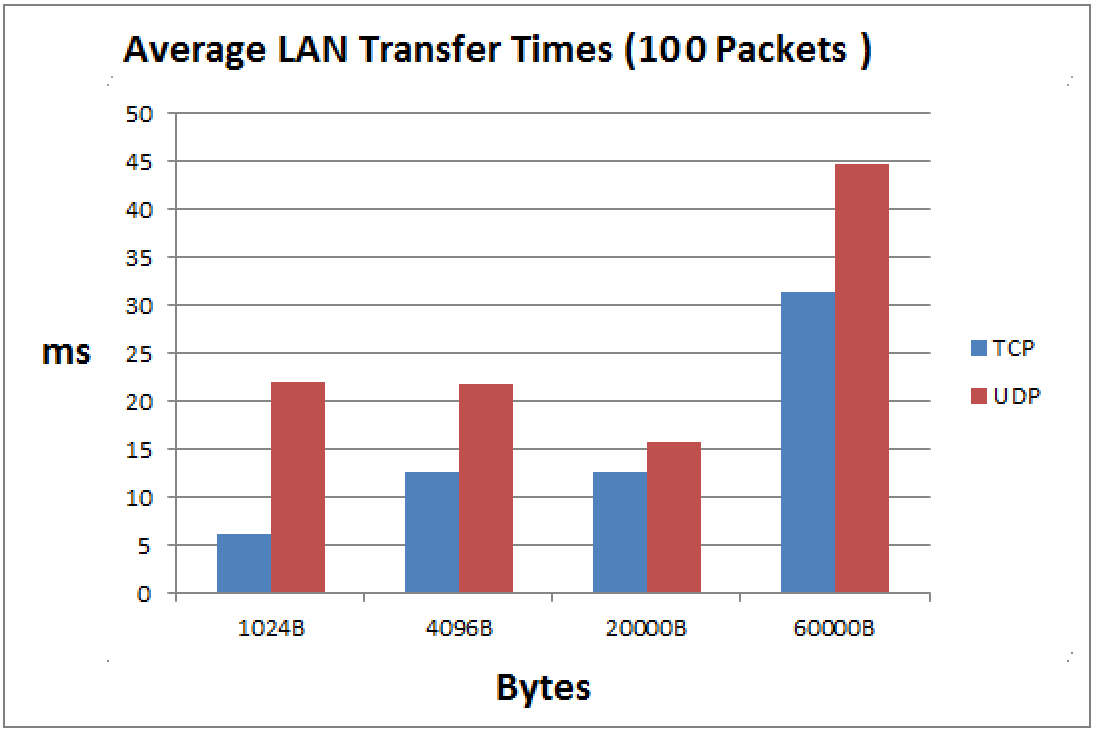

Figure 4. Average LAN Transfer Times (100 Packets)

Figure 4 above shows the difference between TCP and UDP transfer times when sending 100 packets with various packet sizes. This data is the same as that in Figures 1 and 2.

When we compare figure 4 to figure 3, we see how both TCP and UDP act differently between sending bursts of 10 and 100 packets. In figure 4, TCP and UDP are closer in efficiency then shown in figure 3. This is most likely due to UDP's high initial overhead. Once we get multiple packets as such, we see that they are truly much closer than expected in speed. Regardless, TCP is still more efficient in every test to date.

Having additional packets pushes TCP and UDP closer in efficiency than smaller amount of packets. We can concur from this that TCP and UDP will, most likely, be closer in efficiency once we send even more packets.

UDP Packet Loss

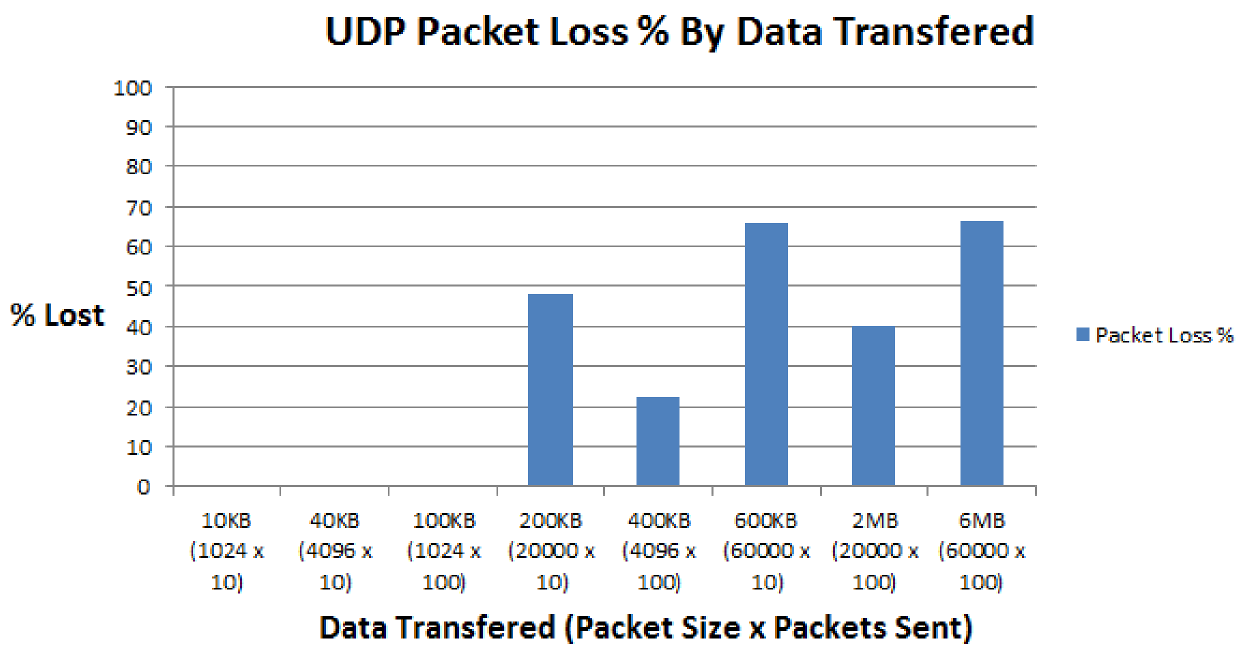

Figure 5. UDP Packet Loss % by Data Transferred

Figure 5 shows us the traditional major drawback of UDP – packet loss. UDP suffers in comparison to TCP in regards to packet loss, which TCP does not have to deal with as it is a connection oriented language. The bars are organized by total data transferred, not packet size nor packets sent.

In the tests above, we see that packet loss is not an issue for lower amounts of data. Once we reach the 200KB size, packet loss becomes apparent. On average, once packet loss occurred, we jumped to an average of 50% packet loss, a ridiculous amount of missing data.

In proper implementations of UDP, under proper packet sizes, packet loss is a lesser issue. However, in strength tests, like above, it becomes more of an issue. If the data we are sending is critical UDP would be a risky choice and not recommended, especially if we are sending mass amounts of it as we did above.

That being said, we received 0% packet loss when sending 1024 byte packets in 100 packet bursts. This test is shown in figure 2, where we see it took 22ms to send. That means we had an effective transfer rate of 4.5MB per second. We can achieve a good, reliable transfer rate with UDP if we keep the packet size low with no loss of packets.

Conclusion

We can conclude from the tests and data shown in figures one through five that TCP is the more efficient and reliable protocol, assuming the data gathered is accurate.

TCP was more efficient in every single test in varying packet sizes and varying quantities of packets sent. In some cases UDP and TCP were close, however not a single case did UDP outperform TCP.

TCP also has the benefit of having zero packet loss, a huge drawback in UDP.

In practice, UDP is known to be slightly faster than TCP, while TCP is known as being more reliable. If the data gathered from this program is accurate, then we can safely say that the above statement was not the case; TCP was the faster protocol.