University of Nevada, Reno
Computer Science and Engineering Department

# Analyzing Time Requirements and Packet Transfer Failure in TCP and UDP When Utilizing Multiple Path Packet Transfer

CPE 400 -- Fall 2021

Instructor:
Shamik Sengupta

Authors:
Cooper Flourens
Mark Graham

## Abstract:

The objective of this paper is to analyze the effectiveness and viability of using Multiple Path Packet Transfer in order to reduce the chance of packet loss by nodal failure. To meet this goal, a network was generated using the NetworkX library in Python, on which we performed a k-shortest paths algorithm to find a realistic set of k paths that would be used in an actual implementation of multiple path packet transfer. This information was exported to a C++ driver to analyze metrics important in networking.

## Introduction and Theory:

Packet loss is a major issue in multiple online spaces, such as video streaming and online gaming. This may be caused by an interruption between the source and destination nodes, whether that be because of network congestion or hardware failure. While it is difficult to manage all online traffic in order to reduce network congestion, or to manage all the nodes in a network for hardware or software failures, an alternate solution would be to receive the packet over two separate paths. While these paths may share some nodes, travelling in this way would reduce the likelihood of any given nodal failure.

There are applications where the network utilization is less important than the reliability of packet data, such as in military and cybersecurity. While it would be much more computationally expensive to send the same packets over multiple paths repeatedly, these examples are instances where Multiple Path Packet Transfer (MPPT) would be most useful.

A practical implementation of MPPT would likely utilize either a k-shortest or k-unique paths algorithm, which checks for all unique nodes in between. As it is hard to construct k-unique paths between a small simulated network due to the lack of nodes, we will be using an abstracted dijkstra's algorithm to find our k-shortest paths.

It should also be noted that there is a difference in the implementation between UDP and TCP. For UDP, we are able to multicast to different nodes at once, meaning that the time requirement for UDP will be significantly less than in TCP, where we can only unicast. For the purpose of our study, we assume that reused nodes will wait to send their packets until all packets with the same destination are received. This would necessitate a maximum number of packets and a new storage mechanism to be included in the protocol, but these are concerns outside the scope of this paper.

## Implementation of Theory:

In order to perform our simulated operations, our python driver would randomly generate a network, assigning weights to each connection that would be the time (in ms) to transfer a packet between the nodes. An example of one of our generated networks and its weights is shown in Figure 1.

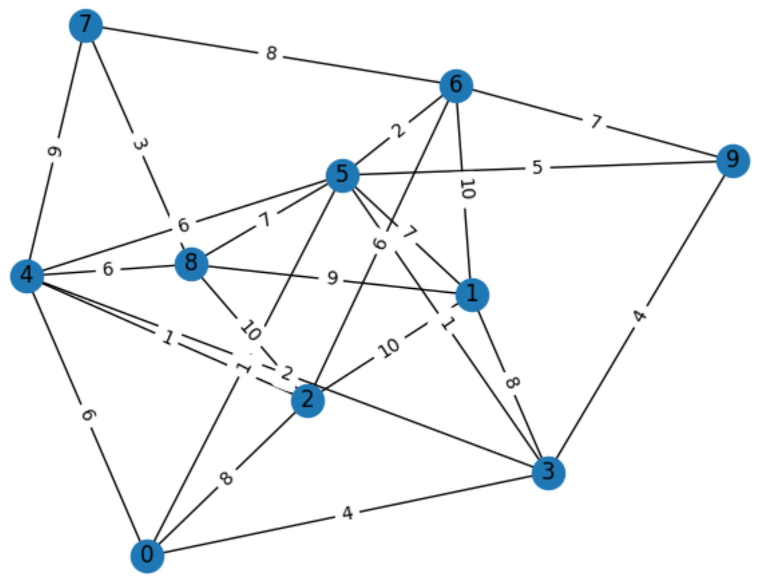


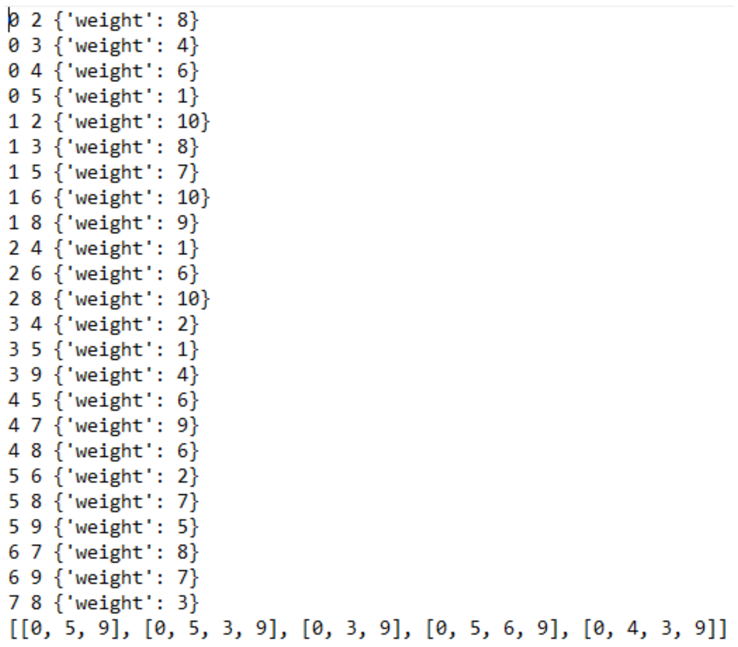
Figure 1 - This figure shows an example of one of our generated networks.

On our generated network, we perform a k-shortest paths algorithm. We will compare the chance of failure of  k = 1 through 5. We will be checking the shortest paths between 0 and 9. Keep in mind that, due to the small size of our network, nodes may be reused between the paths.

Our shortest paths are calculated by selecting the k top results of the NetworkX library's shortest_simple_path() function. This is also done in our python driver file. These paths as well as all the nodes and weights of its edges are exported to our "output.txt" file. The format can be seen in Figure 2 below.

This file is then read in by our C++ driver, which does further calculation on the error rates and time requirements of each k path. This is done by assuming a rate of nodal failure at 10% per node and applying that failure rate to each node in the path. Note that we are assuming a high rate of failure to further emphasize the benefit of MPPT.

The rates of failure and time to transfer over multiple paths are then calculated by comparing every possible combination of the paths. This allows us to easily compare the time to transfer and the reduction in error rate.



```
0 2 {'weight': 8}
0 3 {'weight': 4}
0 4 {'weight': 6}
0 5 {'weight': 1}
1 2 {'weight': 10}
1 3 {'weight': 8}
1 5 {'weight': 7}
1 6 {'weight': 10}
1 8 {'weight': 9}
2 4 {'weight': 1}
2 6 {'weight': 6}
2 8 {'weight': 10}
3 4 {'weight': 2}
3 5 {'weight': 1}
3 9 {'weight': 4}
4 5 {'weight': 6}
4 7 {'weight': 9}
4 8 {'weight': 6}
5 6 {'weight': 2}
5 8 {'weight': 7}
5 9 {'weight': 5}
6 7 {'weight': 8}
6 9 {'weight': 7}
7 8 {'weight': 3}
[[0, 5, 9], [0, 5, 3, 9], [0, 3, 9], [0, 5, 6, 9], [0, 4, 3, 9]]
```

Figure 2 - This figure shows the export of our Python driver file for Figure 1.

The results for each of these tests are then exported to our pathAnalysis.txt file, which can be seen in Figure 3 through Figure 6.

As stated in the Introduction and Theory section, this protocol would wait to send out any packets until all packets to be sent to the same destination were received. This would mean that the time requirement for the UDP would be max([path_times]). This is what we used to project the time requirement for UDP.

TCP has no such benefits with our protocol. As TCP can not multicast, we assume that MPPT would have to wait until it has sent the previous packet to send the next packet. To simulate this, we assume the worst-case scenario of waiting for the previous send to finish completely before sending the next one.

## Results:

Assumptions made for this analysis are that the probability of a single nodal failure is 10%, compared to a realistic probability of a single nodal failure of 1-2%. We also assume a connection time between nodes of 3ms for TCP connections. As UDP may multicast, we assume that multicasting has occurred to reduce the amount of time required for multiple paths.

The paths given by the k-unique paths algorithm are the most time efficient out of all paths possible. The following table shows the paths, their time requirement for UDP and TCP, as well as their probability of failure.

| **Path** | **UDP Time (ms)** | **TCP Time (ms) (UDP + ms [for connection time])** | **Probability of Failure (10% / node)** |
|---|---|---|---|
| 0 - 5 - 9 | 6 | 12 | 30% |
| 0 - 5 - 3 - 9 | 6 | 15 | 40% |
| 0 - 3 - 9 | 8 | 14 | 30% |
| 0 - 5 - 6 - 9 | 10 | 19 | 40% |
| 0 - 4 - 3 - 9 | 12 | 21 | 40% |

Table 1 - This table shows the paths, rate of failure, and TCP and UDP time requirements for each individual path.

Table 2 shows the combination of paths with the total probability of failure and the time requirements if those paths were to be used sequentially in UDP and TCP. This table shows this information for all our considered values of k from 2 to 5.

| **K value** | **Path** | **UDP Time (ms)** | **TCP Time (ms) (UDP + ms [for connection time])** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 2 | 0 - 5 - 9<br>0 - 5 - 3 - 9 | 12 = 6 + 6 | 27 = 12 + 15 | 12% = 30% * 40% |
| 2 | 0 - 5 - 9<br>0 - 3 - 9 | 14 = 8 + 6 | 26 = 14 + 12 | 9% = 30% * 30% |

Previous Table Continued:

| **K value** | **Path(s)** | **UDP Time (ms)** | **TCP Time (ms)** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 2 | 0 - 5 - 9<br>0 - 5 - 6 - 9 | 16 | 31 | 12% |
| 2 | 0 - 5 - 9<br>0 - 4 - 3 - 9 | 18 | 33 | 12% |
| 2 | 0 - 5 - 3 - 9<br>0 - 3 - 9 | 14 | 29 | 12% |
| 2 | 0 - 5 - 3 - 9<br>0 - 5 - 6 - 9 | 16 | 34 | 16% |
| 2 | 0 - 5 - 3 - 9<br>0 - 4 - 3 - 9 | 18 | 36 | 16% |
| 2 | 0 - 3 - 9<br>0 - 5 - 6 - 9 | 18 | 33 | 12% |
| 2 | 0 - 3 - 9<br>0 - 4 - 3 - 9 | 20 | 35 | 12% |
| 2 | 0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 22 | 40 | 16% |
| 3 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9 | 20 | 41 | 3.6% |
| 3 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 5 - 6 - 9 | 22 | 46 | 4.8% |
| 3 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 4 - 3 - 9 | 24 | 48 | 4.8% |
| 3 | 0 - 5 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9 | 24 | 45 | 3.6% |
| 3 | 0 - 5 - 9<br>0 - 3 - 9<br>0 - 4 - 3 - 9 | 26 | 47 | 3.6% |

Previous Table Continued:

| K value | Path(s) | UDP Time (ms) | TCP Time (ms) | Probability of Failure (10% / node) |
|---|---|---|---|---|
| 3 | 0 - 5 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 28 | 52 | 4.8% |
| 3 | 0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9 | 24 | 48 | 4.8% |
| 3 | 0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 4 - 3 - 9 | 26 | 50 | 4.8% |
| 3 | 0 - 5 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 28 | 55 | 6.4% |
| 3 | 0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 30 | 54 | 4.8% |
| 4 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9 | 30 | 60 | 1.44% |
| 4 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 4 - 3 - 9 | 32 | 62 | 1.44% |
| 4 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 34 | 67 | 1.92% |
| 4 | 0 - 5 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 36 | 66 | 1.44% |

Previous Table Continued:

| **K value** | **Path(s)** | **UDP Time (ms)** | **TCP Time (ms)** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 4 | 0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 36 | 69 | 1.92% |
| 5 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 42 | 81 | 0.576% |

Table 2 - This table shows the relation to multiple paths being used, the time complexity for both UDP and TCP connections, and the probability of failure.

However, the following tables demonstrate that if the paths were to run simultaneously, then the slowest time in the paths usage would be the maximum value used for the complete path packet transfer. As noted in the theory section, when utilizing UDP, the protocol would indicate how many packets to wait to receive before sending, in case a path utilizes a single node twice. For example, in our sample where 3 receives from 0 and 5, it would wait to receive the packet from both before sending on to 9. Note that the time requirements for TCP remain unchanged, as we continue to assume the worst case scenario of sequential sending.

| **K value** | **Path** | **UDP Time (ms)** | **TCP Time (ms) (UDP + ms [for connection time])** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 2 | 0 - 5 - 9<br>0 - 5 - 3 - 9 | 6 | 27 = 12 + 15 | 12% = 30% * 40% |
| 2 | 0 - 5 - 9<br>0 - 3 - 9 | 8 | 26 = 14 + 12 | 9% = 30% * 30% |
| 2 | 0 - 5 - 9<br>0 - 5 - 6 - 9 | 10 | 31 | 12% |
| 2 | 0 - 5 - 9<br>0 - 4 - 3 - 9 | 12 | 33 | 12% |

Previous Table Continued:

| **K value** | **Path(s)** | **UDP Time (ms)** | **TCP Time (ms)** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 2 | 0 - 5 - 3 - 9<br>0 - 3 - 9 | 8 | 29 | 12% |
| 2 | 0 - 5 - 3 - 9<br>0 - 5 - 6 - 9 | 10 | 34 | 16% |
| 2 | 0 - 5 - 3 - 9<br>0 - 4 - 3 - 9 | 12 | 36 | 16% |
| 2 | 0 - 3 - 9<br>0 - 5 - 6 - 9 | 10 | 33 | 12% |
| 2 | 0 - 3 - 9<br>0 - 4 - 3 - 9 | 12 | 35 | 12% |
| 2 | 0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 40 | 16% |
| 3 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9 | 8 | 41 | 3.6% |
| 3 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 5 - 6 - 9 | 10 | 46 | 4.8% |
| 3 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 4 - 3 - 9 | 12 | 48 | 4.8% |
| 3 | 0 - 5 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9 | 10 | 45 | 3.6% |
| 3 | 0 - 5 - 9<br>0 - 3 - 9<br>0 - 4 - 3 - 9 | 12 | 47 | 3.6% |
| 3 | 0 - 5 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 52 | 4.8% |

Previous Table Continued:

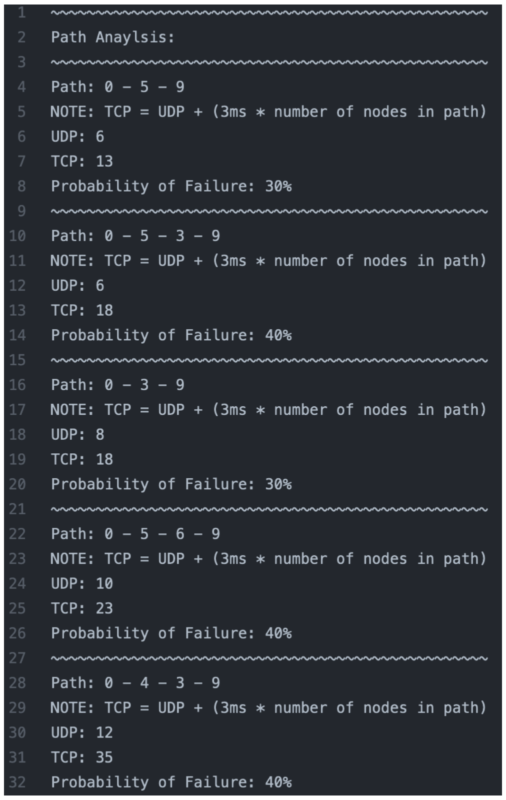
| **K value** | **Path(s)** | **UDP Time (ms)** | **TCP Time (ms)** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 3 | 0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9 | 10 | 48 | 4.8% |
| 3 | 0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 4 - 3 - 9 | 12 | 50 | 4.8% |
| 3 | 0 - 5 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 55 | 6.4% |
| 3 | 0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 54 | 4.8% |
| 4 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9 | 10 | 60 | 1.44% |
| 4 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 4 - 3 - 9 | 12 | 62 | 1.44% |
| 4 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 67 | 1.92% |
| 4 | 0 - 5 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 66 | 1.44% |
| 4 | 0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 69 | 1.92% |

Previous Table Continued:

| **K value** | **Path(s)** | **UDP Time (ms)** | **TCP Time (ms)** | **Probability of Failure (10% / node)** |
|---|---|---|---|---|
| 5 | 0 - 5 - 9<br>0 - 5 - 3 - 9<br>0 - 3 - 9<br>0 - 5 - 6 - 9<br>0 - 4 - 3 - 9 | 12 | 81 | 0.576% |

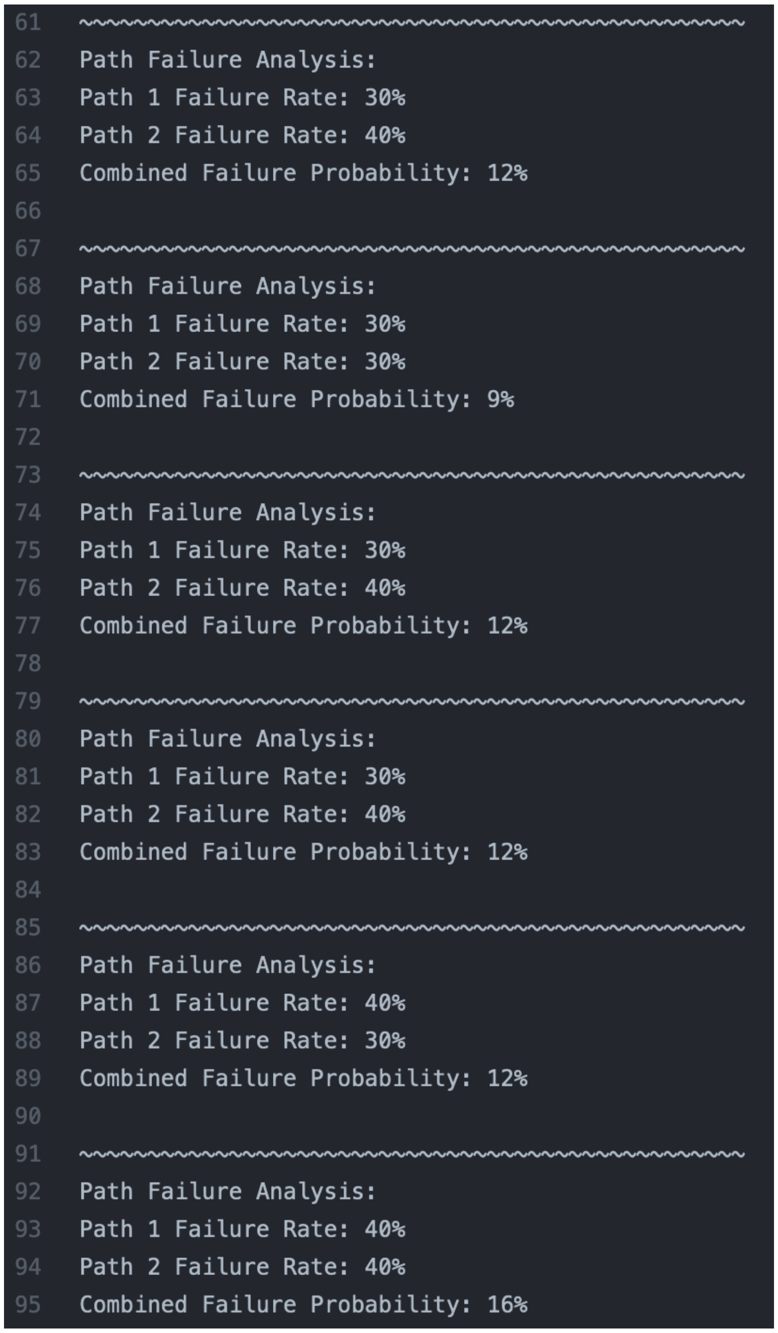
Table 3 - This table shows the relation to multiple paths being used, the time complexity for UDP while paths are used simultaneously, the time complexity for TCP connections, and the probability of failure.

To further confirm this analysis, the following figures are snapshots from the pathAnalysis.txt, which is the output of our code. The code calculates the time to traverse each path in the UDP and the TCP. It also calculates the probability of failure over multiple paths.



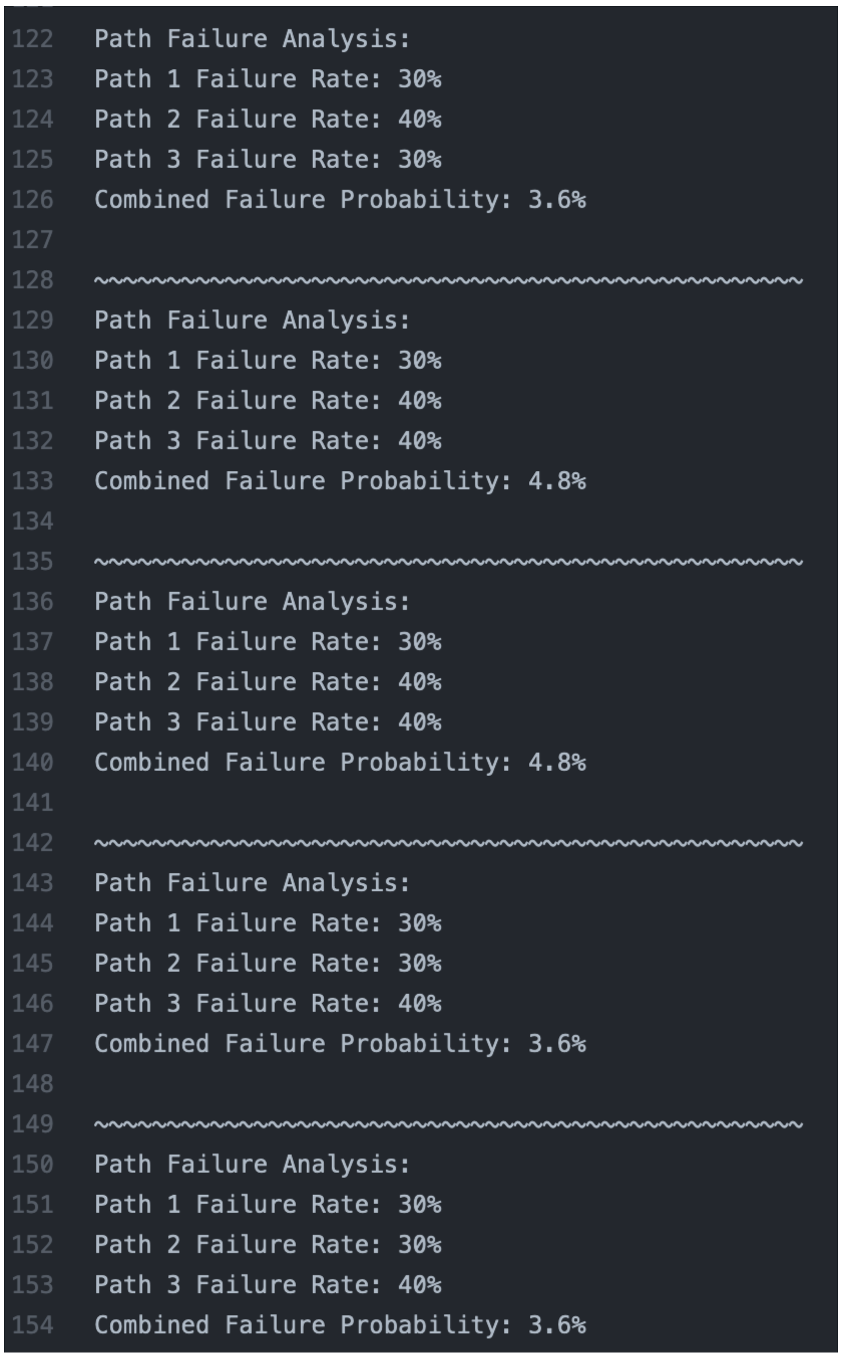
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Anaylsis:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path: 0 - 5 - 9
NOTE: TCP = UDP + (3ms * number of nodes in path)
UDP: 6
TCP: 13
Probability of Failure: 30%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path: 0 - 5 - 3 - 9
NOTE: TCP = UDP + (3ms * number of nodes in path)
UDP: 6
TCP: 18
Probability of Failure: 40%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path: 0 - 3 - 9
NOTE: TCP = UDP + (3ms * number of nodes in path)
UDP: 8
TCP: 18
Probability of Failure: 30%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path: 0 - 5 - 6 - 9
NOTE: TCP = UDP + (3ms * number of nodes in path)
UDP: 10
TCP: 23
Probability of Failure: 40%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path: 0 - 4 - 3 - 9
NOTE: TCP = UDP + (3ms * number of nodes in path)
UDP: 12
TCP: 35
Probability of Failure: 40%
```

Figure 3 - This figure shows the paths, their time requirements for TCP and UDP, and their individual probability of failure.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Combined Failure Probability: 12%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 30%
Combined Failure Probability: 9%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Combined Failure Probability: 12%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Combined Failure Probability: 12%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 40%
Path 2 Failure Rate: 30%
Combined Failure Probability: 12%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 40%
Path 2 Failure Rate: 40%
Combined Failure Probability: 16%
```

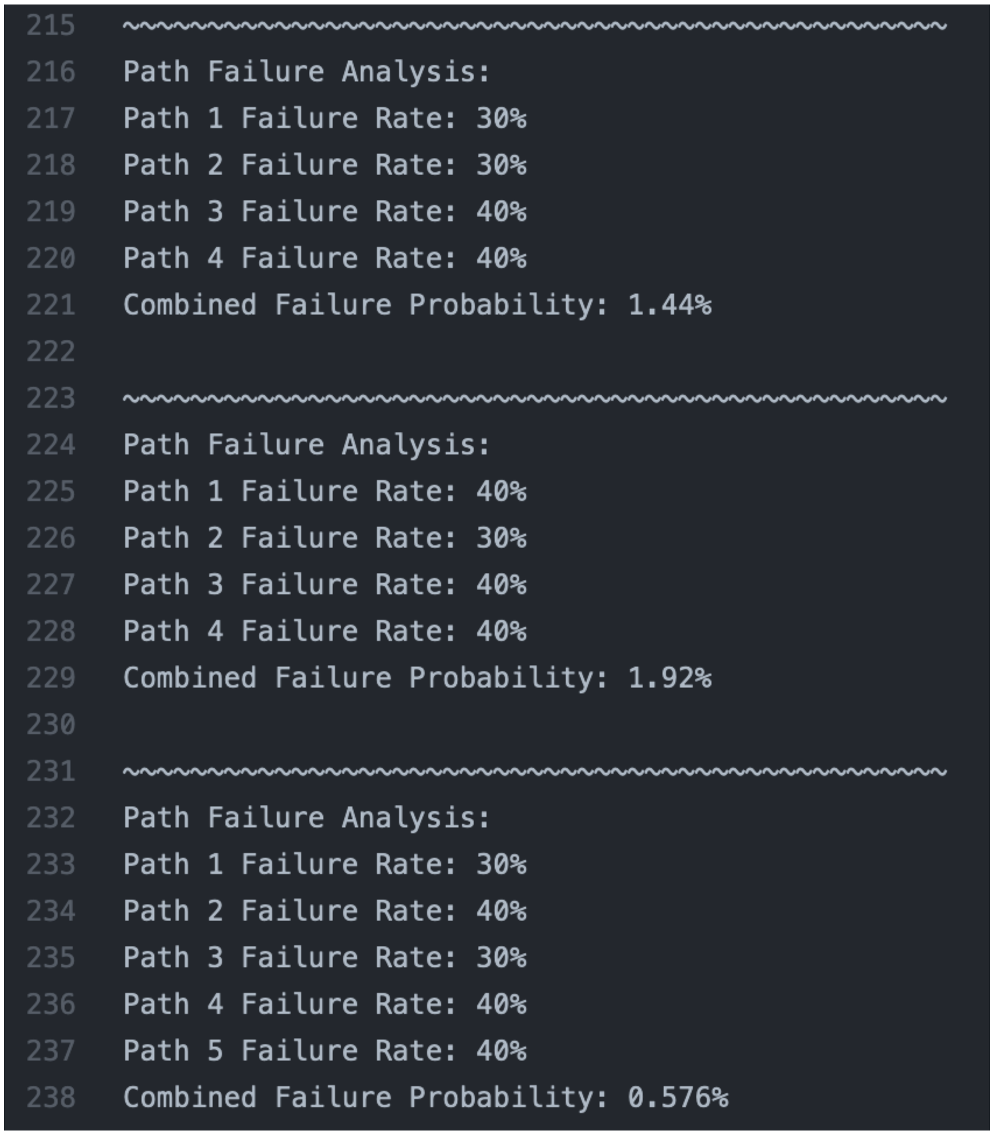
Figure 4 - This figure shows the probability of failure for some combinations of two paths. For the full document, reference the pathAnalysis.txt file.



```
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Path 3 Failure Rate: 30%
Combined Failure Probability: 3.6%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Path 3 Failure Rate: 40%
Combined Failure Probability: 4.8%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Path 3 Failure Rate: 40%
Combined Failure Probability: 4.8%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 30%
Path 3 Failure Rate: 40%
Combined Failure Probability: 3.6%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 30%
Path 3 Failure Rate: 40%
Combined Failure Probability: 3.6%
```

Figure 5 - This figure shows the probability of failure for some combinations of three paths. For the full document, reference the pathAnalysis.txt file.

.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 30%
Path 3 Failure Rate: 40%
Path 4 Failure Rate: 40%
Combined Failure Probability: 1.44%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 40%
Path 2 Failure Rate: 30%
Path 3 Failure Rate: 40%
Path 4 Failure Rate: 40%
Combined Failure Probability: 1.92%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Path Failure Analysis:
Path 1 Failure Rate: 30%
Path 2 Failure Rate: 40%
Path 3 Failure Rate: 30%
Path 4 Failure Rate: 40%
Path 5 Failure Rate: 40%
Combined Failure Probability: 0.576%
```

Figure 6 - This figure shows the probability of failure for a couple combinations of four paths, as well as for all five paths at once. For the full document, reference the pathAnalysis.txt file.

## Analysis and Discussion:

As predicted, Multiple Path Packet Transfer (MTTP) leads to a significant reduction in the probability of failure. This benefit, however, comes at the expense of time and network usage.

In the case of UDP, we saw a time complexity equal to that of the longest time requirement of an individual path, with the k = 5 path being only 12ms for the reduction of error from 30% down to .576%. If, for instance, we were to send the same packet over the same path twice, for the same time requirement of 12ms we would only reduce the error rate to 9%.

That being said, it is also a trade off between network usage and reliability. Obviously, using more paths is going to require significantly more network bandwidth.

In the case of TCP for the five packet transfer, we saw a worst-case maximum time complexity of 81ms, with the same error reduction as the UDP. This is not worth the increase, as we could reduce the error rate by the same amount by sending over the shortest path 5 times instead of the 5 shortest paths once. This is, however, assuming the worst case, where we would have to wait until the previous packet has completed before sending the next one.

From the analysis of both the tables and the output from our code, the results conclude that the more paths included in packet transfer, the more reliable the transfer will be. However, the results also show that the more paths included will have a higher TCP connection time and cause more packet delay.

Some future improvements that can be made to our project and theory is to approach TCP in a different fashion as currently our TCP connection analysis is assuming the worst case i.e. the establishment of connection is only after the previous connection is released. However, TCP connection can occur prior to the release process, which would reduce the amount of time required for the transfer.

It would also be beneficial to analyze the network usage for different size packets to further ensure the practicality of usage in UDP. While our results show that the time requirements do not increase significantly when using MPPT with UDP, it is sure that the network usage will increase exponentially, which may blunt the benefits yielded in our analysis.

## Conclusion:

The conclusion of our project and theory proves that the idea of reliability given multiple paths and that using paths simultaneously reduces the probability of failure in packet transfer. A side product that we assumed was that time for TCP connection would increase, but after deliberation discussed earlier, the best course of action is to use two paths because it would increase the reliability factor as well as keep the time usage for TCP connection to a feasible value. However, should a protocol meeting the specifications laid out in the theory section be introduced for UDP, it would have practical applications in use cases where packet reliability is of maximum importance.