Report Challenge 4 Made by 2198312 - Wenjie Zhao and 2356546 - Fátima González

For the fourth challenge of the Network Systems course, we designed and implemented a fast longest-prefix lookup algorithm for IPv4. The aim of this report is to discuss our approach and understanding of an IPv4 longest-prefix lookup implementation. With the help of this challenge, we managed to get an insight into the scalability of internet forwarding and addressing procedure. We worked closely with the website "networkingchallenges.ewi.utwente.nl", where our results were displayed and an analysis of them could be done more facile and preponderant.

Firstly, we decided to implement a linear search procedure. To do that we worked on the lookup() and addRoute() methods. We created three array lists where we stored the routeIpList, the prefixLengthList and the portNumberList. Inside the lookup() method, a for loop was made, we iterate all addresses given by routes.txt, each one with index i, after shifting the routeIP (32 minus prefixLength) bits from left, we get the fixed part(also called the network part of one IP address). Then, we compared it with another value achieved from shifting the lookup-ip (32 minus prefixLength) bits from left. If the two results are the same, the port number with index i should be the correct one, also we noticed the longest prefix problem, if there are multiple IPs matches the lookup-IP, the one with highest prefix length should be the most specific one and the one which should be returned as well. If none of IPs in routes.txt matches the lookup-IP, "-1" will be returned which means it was not found. After testing our implementation we found out two problems, the code written was correctly working without errors but the speed is relatively low and incapable to complete all tests in 5 seconds. So we decided to optimize our first code, which achieved a score with 1217 points. We did not come up with an idea that we can create HashMap in addRoute() method to combine all the variables together. As we continue using linear search, the speed will definitely stay low. We divided IPs from routes.txt into many slots, the program will decide which slot the lookup-IP belong to and do less linear search and reduce the requested time. Then we reached our highest scored 32143.

If we had to start this challenge again, we would first focus on how to make the program as fast as possible. As we have learnt, linear search is not the most efficient method. Binary shifting or the tree sorting methods would have enabled us to achieve a higher speed mark. Nevertheless, we don't regret starting with linear search, as we have slowly progressed and after having realised an analysis our results, we have gotten a better understanding of longest-prefix algorithms.

To conclude, in this challenge we not only learned precious knowledge about longest-prefix algorithms but withal a practical way to implement it through java. We tried multiple methods to amend our code in order to both increase our speed score. In the end, we enjoyed learning more about implementing an IPv4 longest-prefix lookup and got inspired by how to improve our code to make it work better.