Project Report II

Jason Parkinson and Jan Komorowski

Reinforcement Learning-Based TCP Protocol Over NS3

Our Approach

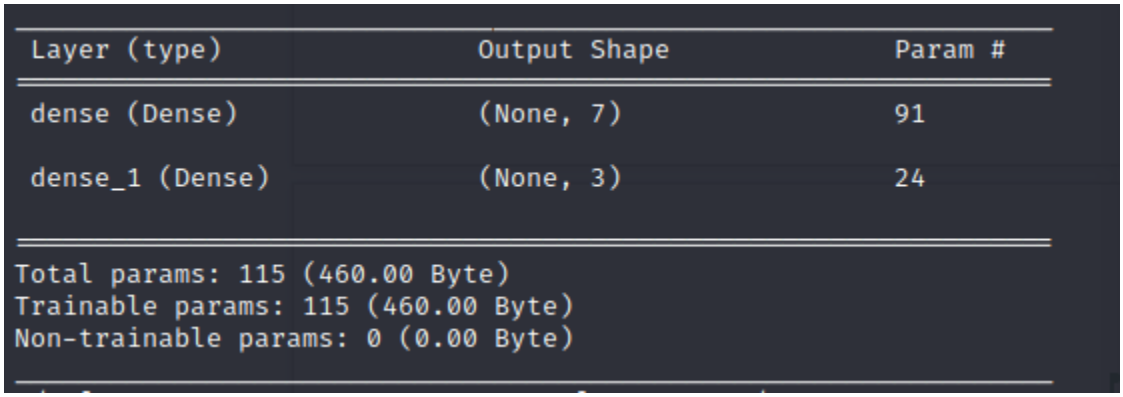
Reinforcement Learning (RL) can solve a lot of the problems of rule-based TCP described in the article given to us on the topic. It can enable an RL agent to learn from what sort of congestion has occurred in the network prior, without the need for rules of any knowledge from past networking scenarios or implementation from the last congestion window. The RL-based agent that we developed interacts with the network scenarios and different network environments and explores the optimal sender rate by varying the size of the congestion window and giving feedback as it continues to move towards the optimal goal of reducing user congestion and optimizing latency. We implemented our reinforcement based model using the popular python libraries with current and past inputs being the state size and the action size of the current users congestion window. This in theory will allow us to better learn from the past sizes and actions taken by the user, use our model to create better outcomes for the user, and reduce the overall latency of TCP.

We used Keras API to define our neural network model and the epsilon-greedy algorithm with “cross entropy” loss function and adam optimizer at learning rate of 0.0012. We simulated our packets using the NS3 development environment as well as the popular proxy library for NS3, ns3 open gym. Together using this as well as other visualization libraries, we were able to simulate successful packet transmission as well as successfully build a model that attempts to optimize the protocol.

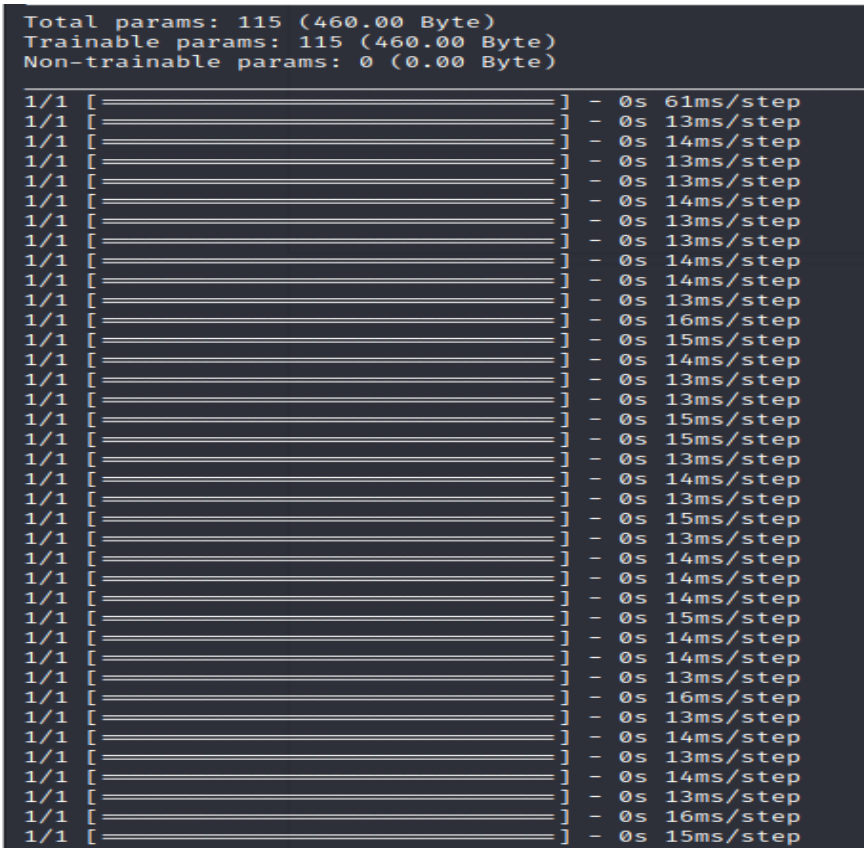
Our Model

As mentioned in the last paragraph, there was a lot that went into the model/simulation that we used to complete our task. We also simulated the existing TCP Reno that is implemented inside of NS3. For our model, we constructed an efficient neural network that uses the popular python-based libraries Tensorflow and Keras. The model is a single Dense layer and uses the ‘relu’ as an activation function for declaring an output for each layer. These activations perform aggregation over the inputs, such as taking the mean, minimum or maximum. The final output layer of the model uses a softmax which declares the concluding decision for optimizing the size of the congestion window as well as decreasing and determining latency. To determine the individual output and decide which cwnd size was going to be declared as an optimal output and which size was going to be declared a sub-optimal output, we used

TCP Reno theory as well as a threshold to determine if the window size that was currently being accepted into the model was above or below said threshold. This would allow us to train our model on inputs sent through the NS3 development environment. From there, we would continue to train our model through reinforcement learning once we ran our actual tests by using the same logic of TCP reno and our optimal threshold.

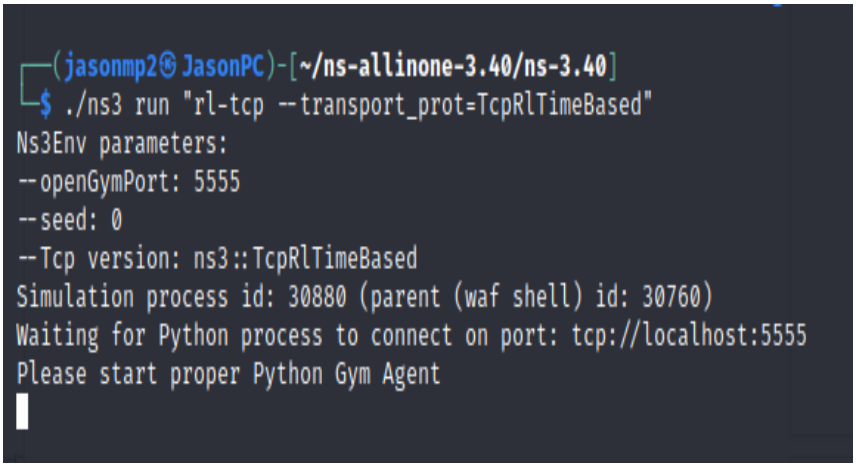


The above photo shows the total number of nodes we used inside of our model before training while the below photo shows our model being trained inside of the linux terminal window. After completely training our model on 115 parameters sent to us (picture below), we were ready to apply our model to live data using NS3.

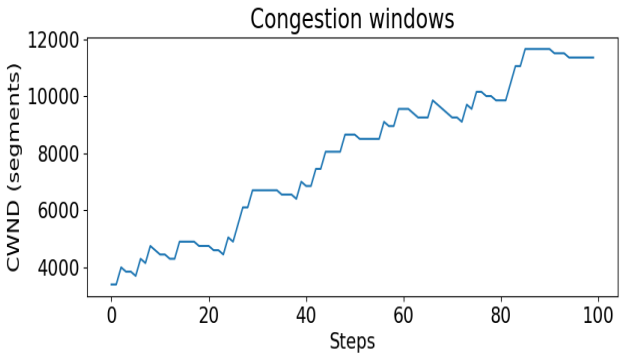


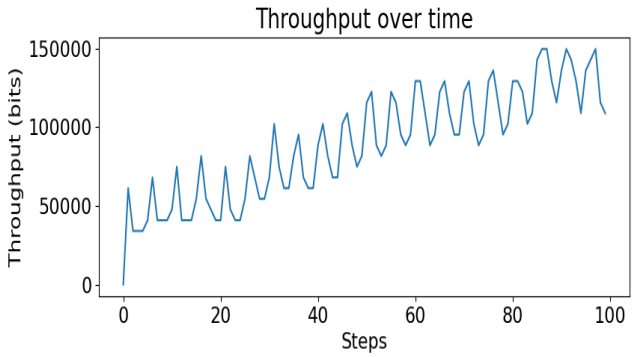
Our Results

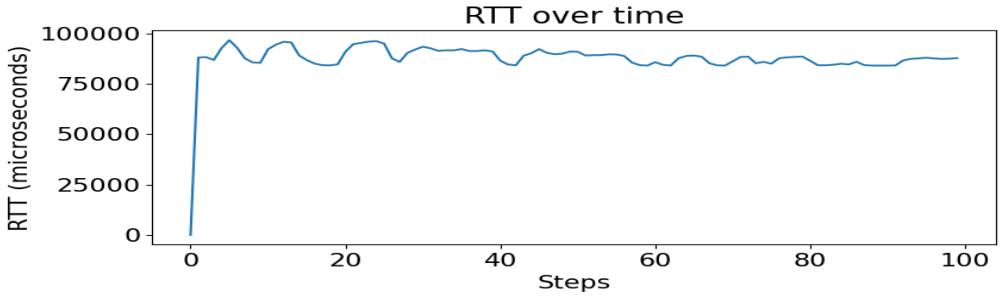
Overall, the test for our new reinforcement based model was run by opening up a port to connect to for implementing the library Open Gym which allows us to enter simulated packets into our RL model. The below screenshot shows the command for opening up the TCP time based port in NS3 version 3.40.



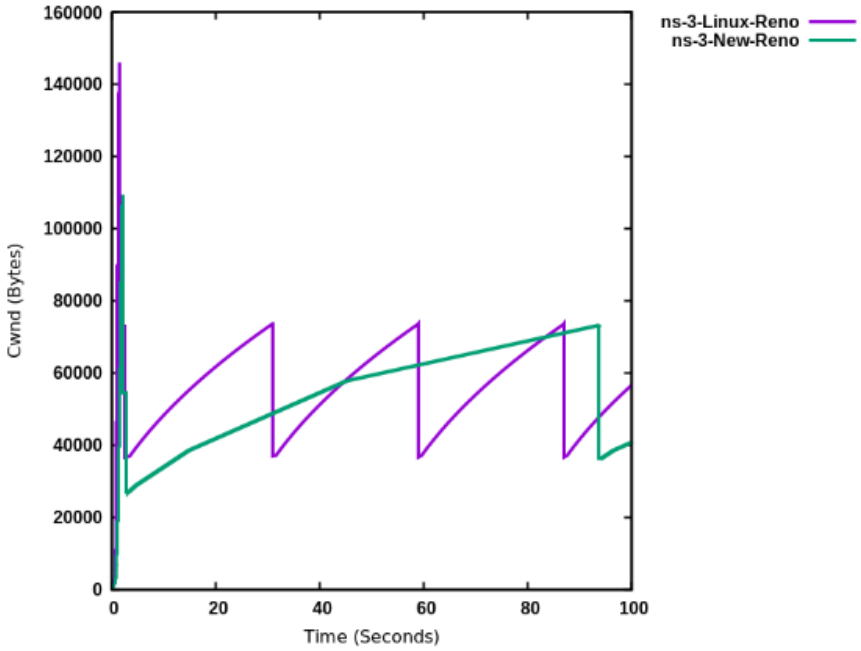
From there we would go on to train our model as described in the previous paragraph. With the help of NS3 Open Gym and tensorflow capabilities, we were able to construct a model that allowed us to successfully monitor the size of the current congestion windows, the throughput over time through one node, and the round trip time of each packet. Below are our results.







As we can see from the above photos, the model did successfully emulate correct behavior relative to how TCP should be implemented and what the congestion window size and throughput should look like. We also ran TCP Reno and simulated the current version through NS3 and below are the following results.



ns-3 TCP NewReno vs. ns-3 TCP Linux Reno

As we can see from the above image, the existing ns-3 TCP Linux Reno and TCP NewReno are both relatively more efficient than the RL model that we used to simulate the existing algorithms written and used by network professionals today.

Analysis

As mentioned above, the model that we created was *NOT* as efficient as existing TCP Reno and NewReno algorithms today and this could have been for a few reasons.The first reason was insufficient training/layers in our model. The more a model is trained, the more it is likely to make the correct prediction on the model. We only trained our data on 115 different scenarios before allowing it to begin making predictions which would most definitely hinder its performance. Additionally, if we added more layers to our model, it potentially could have picked up more complex patterns and generalize better. This is most definitely another way that the model could have been improved to help make better predictions on the congestion window size as well as throughput optimization. The last improvement that could be made to the system is a better understanding of the NS3 system itself. Getting to optimally adjust the TCP packets being sent would be a great help and would definitely lead to more optimized results. The existing file used to run the test on NS3 was already optimized which could have led them to better results as well.

Overall, we definitely have a greater understanding of TCP Reno and how it can be manipulated to optimize performance in a network. We also had the chance to experiment and test with a very difficult yet intriguing software in NS3. I think that this will help improve our overall networking knowledge as well as enhance our capabilities in the field.

Acknowledgements

The full complex idea of this simulator agent was inspired by Kenan, Sharif and Piotr Gawlowicz.

W. Li, F. Zhou, K. R. Chowdhury and W. Meleis, "QTCP: Adaptive Congestion Control with Reinforcement Learning," in IEEE Transactions on Network Science and Engineering, vol. 6, no. 3, pp. 445-458, 1 July-Sept. 2019, doi: 10.1109/TNSE.2018.2835758.

Jay, N., Rotman, N. H., Brighten Godfrey, P., Schapira, M., & Tamar, A. (2019). A deep reinforcement learning perspective on internet congestion control. In *36th International Conference on Machine Learning, ICML 2019* (pp. 5390-5399). (36th International Conference on Machine Learning, ICML 2019; Vol. 2019-June). International Machine Learning Society (IMLS).