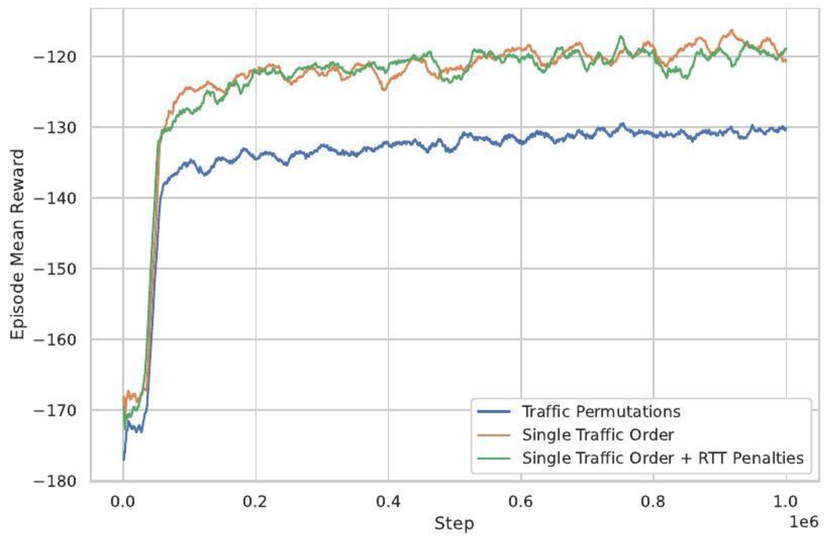

Figure 4: Mean training reward over the last 100 partial episodes on the three training runs.

training: here, the agent is evaluated on completing a task longer than the partial episodes seen during training, which are limited to 200 steps, and a faster completion coincides better results.

We repeated the experiment 100 times for each of the three trained agents. Results are then compared to the performance of Secure Copy Protocol (SCP), which in our system uses TCP CUBIC as the transport protocol, on the same file transfer task. Figure 5 shows the optimal behaviour, computed from the a-priori knowledge of the background traffic, along with the batch of results obtained from the trained agents. The average, best, and worst performance of TCP CUBIC are also shown under the same conditions after having repeated the transfer 100 times as well. Figure 3 presents the variation of the CWND size during MARLIN's best run across all the three experiments. The color of each segment represents the RTT of the communication measured during a period of one SRTT immediately following the action that took the CWND size to the value represented by the rightmost end of the segment.

Runs with significant performance degradation were aborted after 80 seconds, with 5a reporting 4 aborted experiments, 5b reporting 20, and 5c reporting 2. TCP CUBIC completes the file transfer in 45.03s in average, with its fastest transfer achieved in 33.97s, and a worst performance of 70.78s. Excluding aborted runs, MARLIN reached the target in 46.1s in average in 5a, 50.31s in 5b, and 46.9s in 5c. All agents' best performance was faster than TCP CUBIC's best. In 5a, 48% of the experiments achieve a performance equal or better than the average accomplished by TCP CUBIC, 31% in 5b, and 47% in 5c. The fastest transfer was completed by 5b in 24.84s, 27% faster than the fastest SCP transfer and 45% faster than its average. For comparison, the available bandwidth on the link allows file transfers to be completed in 25s (see Figure 5); faster transfers can still occur if the traffic injected by MARLIN causes other flows to slow down.

D. Discussion

Despite the increased decision making complexity brought by continuous actions in a problem as intricate as CC in a real network and a training budget of 1M steps, which is modest when compared to other RL-based CC agents, results are promising and already comparable to TCP CUBIC in our environment. We believe that part of it is due to the sample efficiency of SAC. Another reason can be found in the shaping of the CC problem as an infinite-horizon task with strictly negative rewards, which enables the agent to exploit its acquired experience in tasks longer than the partial episodes seen during training and with different goals.

Permuting the order of the background traffic patterns during training did not deteriorate the performance during evaluation. In fact, evaluation runs exhibited lower variance, a better fastest transfer, and similar average transfer time compared to the experiments shown in 5a. Moreover, fewer experiments had to be aborted. These results suggest that MARLIN's robustness might be further improved by feeding data from diverse scenarios to the algorithm.

Figure 3 shows the CWND size during one of the evaluation runs. Note that most of the time, following an increased RTT reading, the agent has learned to respond by reducing the CWND size or decreasing its growth speed in the next step. This behavior is compatible with many RTT-based CC algorithms.

Finally, although most trajectories obtained during evaluation express promising and valuable results (Figure 5), they also present a significant degree of instability, which warrants additional research. This behavior is particularly evident in 5b, where the model is trained using the function in Eq. 2. This model has had several transfers significantly faster than the ones in 5a and 5c, as well as those obtained with SCP; nonetheless, the model has proved considerably slower in average.

V. Related Work

Research efforts on the design and optimization of TCP CC have historically been very different from the approach discussed in this paper. Traditional CC algorithms aim at achieving full bottleneck link utilization by applying diverse heuristic strategies that increase the CWND size until a congestion sign emerges, such as a packet loss, e.g., NewReno [28] and CUBIC [29] (the default CC algorithm in modern Linux Kernels and recent Windows operating systems), or an increase in latency, e.g., Vegas [30] and, more recently, BBR [31]. These approaches have been shown to work well under specific network conditions, but underperform or experience other types of issues in other scenarios [32, 33, 34].

Due to the exceptional degree of heterogeneity, complexity, and intrinsic dynamism of networking environments, and following the reinvigorated interest in ML that captivated the scientific community in the last two decades, several recent efforts have focused on learning-based transport protocol optimization techniques. Some approaches focus on addressing very specific problem of transport protocols, such as accurately identifying congestion events, but do not aim at replacing CC algorithms. For instance, researchers have successfully built ML-based models that can distinguish between losses caused