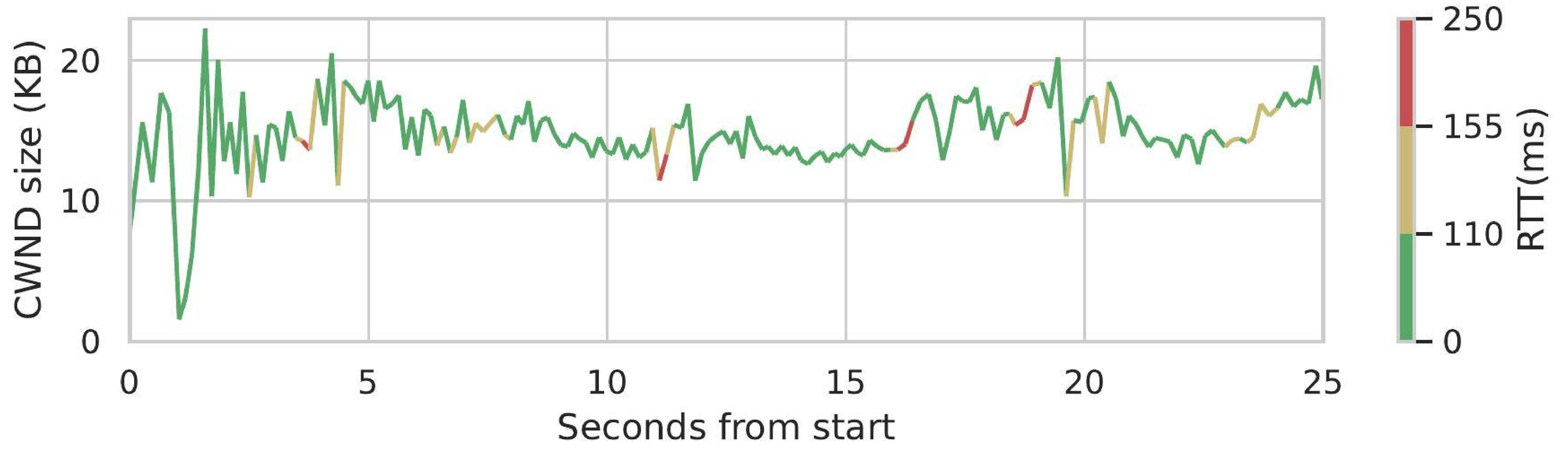

Figure 3: The rolling averaged CWND size is plotted against the time from the beginning of the episode, while segment color represents the RTT observed between two actions. This plot was obtained from the best run of the second MARLIN model, i.e. 5b, which completed the transfer in 24.84s.

that drove the design of the QUIC protocol [25] (for additional details on Mockets, the reader can refer to [26]). Mockets implements a very aggressive CC algorithm whose purpose is to fully utilize the available bandwidth in presence of variable communication latency and elevated packet loss. While it presents several advantages over TCP and other transport protocols in degraded environments [27], Mockets' existing CC fails to share link capacity with other communication flows going through the same links and cannot adapt quickly to changes in the available bandwidth. Therefore, Mockets could benefit significantly from MARLIN, which could make it a viable choice also for traditional network scenarios.

To have an accurate representation of the state of the environment within MARLIN, we improved the RTT estimation in Mockets. To do so, the sender keeps track of the transmission time of each packet. When an ACK arrives, the sender computes the RTT by calculating the difference between the current time and the transmission time of the last packet received by the receiver before generating the ACK message and then subtracting the ACK processing time from it. To make this calculation possible, the receiver appends the identifier of the last received packet (a strictly increasing unsigned integer) and the processing time (in microseconds) to all ACK messages.

IV. Experimental Results

A. Hardware Involved

The testbed described in Figure 2 is implemented using Commercial Off-The-Shelf (COS) devices. The agent is trained on a workstation equipped with an Intel i7-8700 CPU, 32GB DDR4 RAM, an NVIDIA GeForce RTX 2060 GPU, and an Intel Wireless-AC 9560 Network Interface Card (NIC). The "Receiver Application" resides on a 4GB Raspberry Pi 4 Model B (RPi4B). The two AP used are a TP-Link EAP245 and a Netgear WAC505, connected to two Netgear GS108E switches. A 2015 Apple MacBook Pro with macOS Catalina v10.15.6 generates background traffic, which is sent to the receiving application running on a Dell Latitude 7000 laptop (E7470). Both the desktop machine and E7470 run the Ubuntu 20.04.5 LTS OS. All NIC-related optimizations, e.g., TCP Segmentation Offload (TSO) or Large Receive Offload (LRO), are enabled, as per default, on all systems. A Ubiquiti Networks EdgeRouter X with EdgeOS routes packets between the two subnetworks.

B. Training

The agent is trained for 1M steps on an infinite-horizon task with partial episodes lasting 200 steps. When the last step of a partial episode is encountered, a *truncated* signal is emitted and PEB is performed, recalling the agent that additional steps and rewards would actually be available thereafter [12].

Due to time constraints, the optimization steps involved (for actor, critic, target networks, and learned entropy coefficient) happen at the end of every partial episode. Taking training steps between two actions on a real network is a luxury we cannot afford, as we observed it would introduce delays in the order of 800-1000ms. Training at the end of an episode allows us to avoid such effect, limiting the overhead to the sole pipeline, which we report being around 25 to 30 ms.

At the beginning of training, in order to increase exploration, actions coming from a uniform random distribution are taken for 10K steps. Subsequently, SAC starts its normal exploration-exploitation strategy, led by its entropy coefficient.

We trained three different models for MARLIN. For the first one, we used Eq 1 (5a). For the second model, we used the reward function in Eq. 2 (5b). In both cases, the background traffic pattern was fixed, with elephant flows following this order: $[100\ \text{UDP}, 200\ \text{TCP}, 100\ \text{UDP}, 50\ \text{UDP}]$. For the last model (5c), we changed the background traffic to use all permutations presented in Section II-B.

The mean training reward shows a promisingly increasing trend in each of the three setting, as shown in Figure 4. It is straightforward to interpret the performance showed during training by taking as a reference our ideal target, which, in this case, would result in -100 mean training reward.

C. Results

Following training, we tested all the three MARLIN models by transferring a 3MB file on the same network with the background traffic pattern $[100\ \text{UDP}, 200\ \text{TCP}, 100\ \text{UDP}, 50\ \text{UDP}]$. It is important to note that file transfer is a significantly different problem from the one that the agent faces during