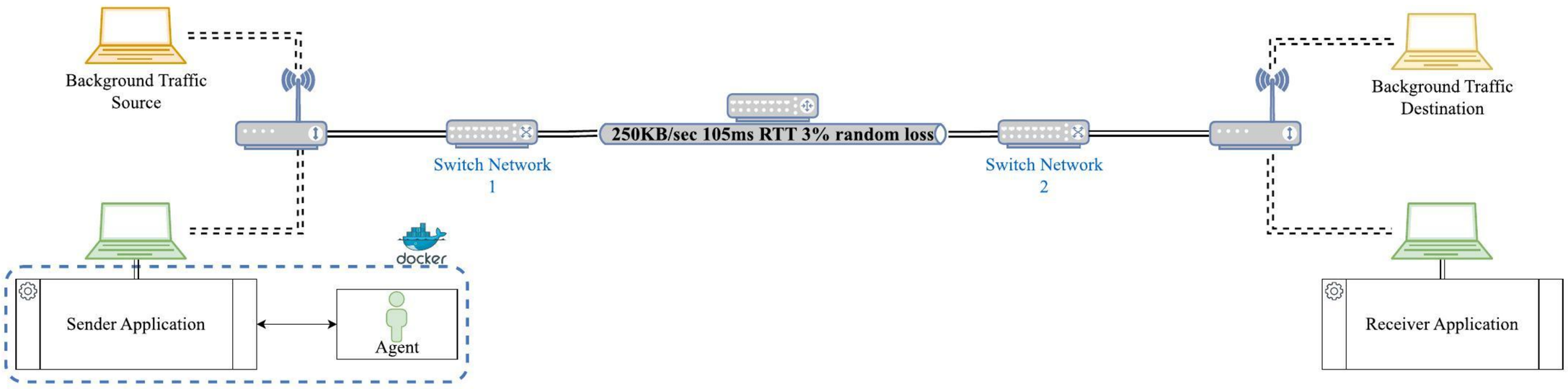

Figure 2: A dumbbell network is used for training and testing. Background and agent traffic are transmitted from one sub-network to the other over the bottleneck link. Senders and receivers are all connected to a WiFi Access Point.

the action taken can be inferred. These events will take, at the very least, one RTT to complete. This raises a fundamental question for anybody who wants to apply RL to the problem of CC: *when should the next action be taken?* While other similar approaches, like [16, 17], gather statistics during a fixed amount of time to compute the state and take the next action, MARLIN implements a different heuristic algorithm. Every time the transport protocol makes new network statistics available to MARLIN, if at least the last reported SRTT has elapsed since the time the last action was taken, data collected up to that time is refined, the reward r_t with regards to action a_{t-1} is assigned, a new state s_t is fed to the agent, and a new action a_t is taken. Waiting for at least one SRTT between actions prevents that a new action is taken before the impact of the previous one can be sensed by the agent.

B. Training Environment Setup

A key design point of MARLIN is to be trained and evaluated on real networks. Utilizing real components allowed us to understand the challenges that the CC domain brings to RL, avoiding the risk of failing a "sim-to-real" transposition, which could be caused, by inaccurate emulated/simulated patterns and weak assumptions, at the cost of a slower training time. Relying on real hardware and protocol implementation intrinsically diversifies training experience thanks to the stochasticity of the environment and it prevents the agent from overfitting on artificial patterns that would not be found in the real world.

To this end, we built the network shown in Fig. 2, which comprises two sources of traffic and two receivers, two WiFi Access Point (AP), two network switches, and one router connected in a dumbbell topology. Traffic sources are connected to the WiFi AP located at one end of the dumbbell, while traffic receivers are connected to the AP at the other end. The "Background Traffic Source", on the left, is responsible for generating background and send it to the "Background Traffic Receiver" node, on the right. The "Sender Application" transmits traffic to the "Receiver Application" via a custom transport protocol implementation that uses MARLIN as its CC algorithm.

The need for implementation flexibility led us to integrate MARLIN with Mockets, a protocol originally designed for communication environments characterized by limited bandwidth, typically found in tactical and wireless sensor networks. Further details regarding Mockets are presented in Section III. To avoid running into situations where MARLIN could underperform due to the protocol implementation, we shaped the network scenario around the typical Mockets use case for this first iteration. We set a Smart Queue policy on the router to limit the maximum amount of traffic flowing between the two subnetworks to 250 KB/s; we also used TC NetEm - Traffic Control Network Emulator (TC-NETEM) on the "Sender Application" node to introduce 100 ms of latency and 3% random packet loss that only affect the traffic generated by the application. Note that the router's manual reports that it effectively limits the actual rate to 95% of the specified value when Smart Queue is enabled [18].

The background traffic patterns generated can be divided into elephant flows, i.e., long-lived data transfers that represent a large percentage of the total traffic (imagine large file transfers or video streaming), and mice flows, i.e., short-lived data transfers at low throughput (e-mail or RPC calls). Roughly 87% of the background traffic in our testbed is made by elephant flows, consisting of four different flows that alternate each other over the link every two seconds, and two mice flows, which continuously generate very short-lived (in the order of milliseconds) traffic bursts with intervals that follow a Poisson distribution.

The Background Traffic Source generates traffic using Multi-Generator Network Test Tool (MGEN) [19]. Elephant flows consist of two UDP communications transferring data at 100 KB/s, one UDP communication transferring data at 50 KB/s, and one TCP connection producing 200 KB/s of traffic. One TCP and one UDP mice flow introduce an extra 17 KB of traffic in average over the link each second. Each elephant flow is assigned to a different temporal slot of a 2 second duration and they repeat every 8 seconds. Mice flows start with the first elephant flow and continue until traffic generation is stopped.

C. Agent Design

The interface between MARLIN and the transport protocol, shown in Figure 1, is implemented using *gRPC*. The gRPC middleware sitting between the transport protocol and MAR-