Remy

Congestion control has been a fundamental challenge in a networking domain for the past thirty years. Previously, the researchers have explored the end-to-end congestion control algorithm over heterogeneous pack-switched networks, which typically computes a congestion window as well as the round-trip time (RTT). However, each of such end-to-end approaches has its own domain of the problem in which it outperforms over other algorithms. Said differently, a majority of the classical congestion algorithms lack the ability to adapt itself to new scenarios. To ensure a generalizable solution to congestion control, Remy exploits the idea of leaving a machine to automatically adapt to an objective of congestion control, which is specified by the end-user. More specifically, given a user-specified objective, Remy models the congestion control problem by expressing prior assumptions (Markovian) about the network, defining a traffic model for the offered load given an endpoints and maximizing the objective function, measured over the set of defined network and traffic model.

Regarding Remy, we have initiated the discussion by asking the following question: Why are machines simply better at producing these algorithms? Given a large scale dataset with a set of defined "rules" like Go and Chess, the machine learning approaches outperform the ability of human. Yet, the variables in the networking, such as signal attenuation, data-rates, channel capacity, are not as finite as the number of legal moves on a board and constantly changing. Therefore, Remy may not be an optimal solution to congestion control, which requires some amount of general intelligence and adaptability. Furthermore, when deployed, it has less transparency in how the mechanism works, making it potentially more difficult to debug when failure happens. The fact that Remy is an offline optimization tool exacerbates this failure issue because retraining is required whenever unknown failure happens. During the talk, we have emphasized that this is a critical limitation to the work and transferring the ideas of Remy into a real-time realm may be more practical. Another weakness of the paper we have addressed is its lack of evaluation on the impact of the parameters that are left open to the end users. Although the weight factor for fairness-vs.-efficiency tradeoff is defined, it is underexplored.

Overall, the design of Remy resembles Reinforcement learning. First, both are based on the Markov decision process with a set of possible states (ack_ewma, send_ewma, rtt_ratio), a set of possible actions (to send the packet and not to send the packet), the distribution of reward (the flow's score) given (state, action), distribution over the next (state, action) pair. Also, both algorithms try to find the optimal policy that maximizes cumulative reward via gradient descent (i.e. Cartesian product). Yet, to train a typical reinforcement learning algorithm, many AI researchers leverage a random minibatch sampling to uncorrelated the batches of consecutive samples. By doing so, Remy may enhance its ability to generalize with unseen data.