Midterm Update

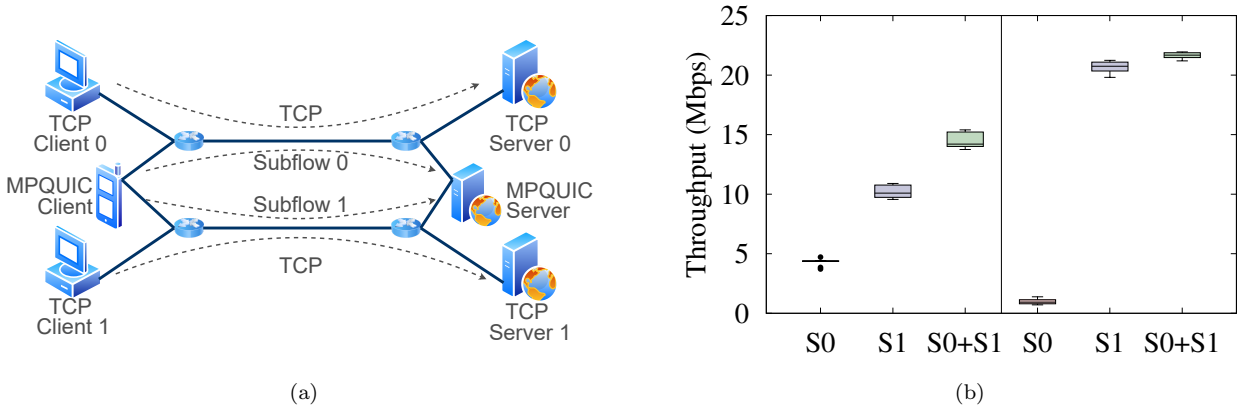
Shirley Shu, Tianfang Chang

1 Simulation Platform

Recall that our design will be based on the up-to-date multipath transport-layer protocol, MPQUIC. Since the experiments on transport-layer protocols are difficult to realize with the hardware in the real world, network simulator is a valuable choice with its various modules, high precision, and low cost. Exploring the popular simulators, we selected ns-3, a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. Benefiting from the open source contribution, we can use the ns-3 QUIC module [1] with some multipath extension to implement our design. The multipath extension of QUIC module is a previous research work, which is able to have the basic data transmission over two paths.

2 Problem Verification

To encounter the transmission close to the real world, our experiment scenarios simulate the transmission similar to mobile devices, which contains both Wi-Fi and LTE network and is able to do packets transmission with both interfaces at the same time. From our daily experience, we know that mostly Wi-Fi have a higher data rate than LTE. So, we consider the topology in Fig.1a that two subflows each with TCP background traffic respectively. The data rate for subflow 0 is 10 Mbps with 10 ms delay, which is used to simulate the LTE connection. The data rate for subflow 1 is 50 Mbps with 10 ms delay, which is used to simulate the Wi-Fi connection. We tested for two widely used scheduler algorithms, Round-Robin and Min-RTT, with 10 rounds for each transmission. Fig. 1b illustrates the throughput comparison for Round-Robin (left-side) vs. Min-RTT (right-side). For Round-Robin, the throughput of subflow 0 is around 5 Mbps, which is close to the ideal data rate, but subflow 1 is not fully used. For Min-RTT, the throughput of subflow 1 is around 20 Mbps, which is close to the ideal data rate, but subflow 0 is not fully used. Thus, we verified the problem that the transmission could not fully utilize both subflows with either Round-Robin or Min-RTT when two subflows have distinct data rates.



(a) (b)

Figure 1: (a) Topology of two subflows each with TCP background traffic respectively, (b) Throughput comparison for Round-Robin vs. Min-RTT with the different data rates of subflows (S0: 10 Mbps, S1: 50 Mbps)

3 Bandit Algorithm Analysis

The most important feature distinguishing reinforcement learning from other types of learning is that it uses training information that evaluates the actions taken rather than instructs by giving correct actions. This is what creates the need for active exploration, for an explicit search for good behavior.
The multi-armed bandit problem is a classic problem that well demonstrates the exploration vs exploitation dilemma. Imagine you are in a casino facing multiple slot machines and each is configured with an unknown probability of how likely you can get a reward at one play.
Exploration will give up some known reward information and try some new options - that is, in a certain state, the algorithm may have learned what action to choose to make the reward larger, but it cannot be done every time. The same choice, maybe another choice that has not been tried will have a greater reward, that is, Exploration hopes to explore more potential information. Exploitation refers to maximizing rewards based on known information.
The difference can also be simply understood that the Exploration algorithm searches for the global optimal solution and is not based on the existing experience; the Exploitation algorithm searches for the local optimal solution and maximizes the use of the existing experience information.

- We have k-armed bandit with winning probabilities, $p_1, ..., p_k$.
- At each time step t, we take an action(pull a arm) on the machine and receive a reward r_t.
- A is a set of actions, each referring to the interaction with one arm. The value of action a is the expected reward, $Q(a) = \mathbb{E}[r|a] = p$. If action a_t at the time step is on the k_i arm, then $Q(a_t) = p_i$.
- R is a reward function. In the case of a k-armed bandit, we observe a reward r in a stochastic fashion. At the time step t, $r_t = R(a_t)$ may return reward 1 with a probability $Q(a_t)$ or 0.

Let $Q_t(a)$ denotes the estimated value of action $a(k_i$ arm) at time step t.
If $Q_t(a)$ is an accurate estimate of $q_*(a)$, we can select actions using $Q_t(a)$. Greedy action estimate of $q_*(a)$: $A_t \doteq argmaxQ_t(a)$
Sample average estimate of $q_*(a)$:

$$Q_t(a) \doteq \frac{sum\ of\ rewards\ by\ choosing\ a\ prior\ to\ t}{number\ of\ times\ of\ choosing\ a\ prior\ to\ t}$$

If a is chosen more often, the estimate becomes more accurate. Exploratory action selection to improve value estimation.
Greedy action selection exploits current knowledge of action value. Exploitation maximizes the expected reward for the current step.
Non-greedy action selection explores other actions and improves their value estimates. Exploration may improve total reward in the long run.
Constant memory requirement and constant per-time-step computation:

$$NewEstimate = OldEstimate + StepSize[NewObervation - OldEstimate]$$

Sample average estimate:

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{t}[R_t(a) - Q_{t-1}(a)]$$

$R_t(a)$ is the reward after selecting action a at time step t.
To solve the non-stationary problem, the updated samples will be significant, so we can use a constant discount factor alpha, we can rewrite the update equation like this:

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{\alpha}[R_t(a) - Q_{t-1}(a)]$$

Note that we have replaced t with a constant alpha, which ensures that the most recent samples have a higher weight, and these most recent samples more determine the delta.

3.1 Epsilon Greedy

Using the epsilon probability, we will choose a random action (exploration) and choose the action with the largest $Q_t(a)$ with probability $1 - \epsilon$.

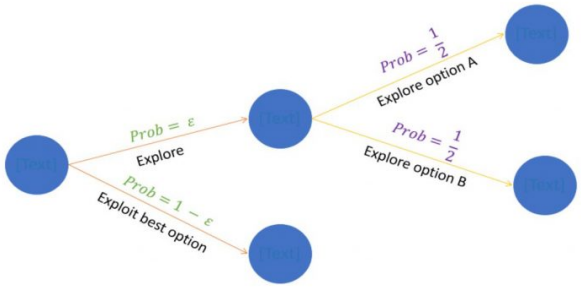


Figure 2: $\epsilon-$greedy

With probability $1 - \epsilon$, we choose the action with the maximum value ($argmax\ Q_t(a)$)
With probability ϵ, we randomly choose an action A a set of all actions.
For example, if we have a problem with two actions A and B, then the epsilon greedy algorithm works as follows:
Upper Confidence Bound
The UCB algorithm keeps a track of the mean reward for each arm up to the present trial and also calculates the upper confidence bound for each arm. The upper bound indicates the uncertainty in our evaluation of the potential of the arm.
The algorithm is highly unsure of an arm's potential if it has a very high upper confidence bound and hence chooses the arm because of a great exploration opportunity.

$$a_t^{UCB} = argmax\hat{Q}_t(a) + \sqrt{\frac{2logt}{N_t(a)}}$$

Each time a is chosen, the uncertainty may decrease: $N_t(a)$ increases, and, when it appears in the denominator, the uncertainty term decreases. On the other hand, every time an action other than a is chosen, t increases, but $N_t(a)$ does not; because t appears in the numerator, the uncertainty estimate increases.
Using natural logarithms means that the increase will be smaller over time; eventually all operations will be selected, but operations with lower value estimates or already frequently selected will be selected less frequently over time, which ultimately leads to the final repetition of selecting the best move.

References

[1] A. De Biasio, F. Chiariotti, M. Polese, A. Zanella, and M. Zorzi, "A quic implementation for ns-3," in *Proceedings of the 2019 Workshop on Ns-3*, ser. WNS3 2019, 2019, p. 1–8.