Research on SDN Congestion Control Based on Reinforcement Learning

Limin Zhang1*, Xiaolu Tian1

1 Information Technology College, Guang Dong Polytechnic College, Zhao Qing, Guang Dong, 526100, China

*Corresponding author’s e-mail: liminzhang26@163.com

Abstract. Aiming at the congestion control problem of SDN, this paper designs a network congestion control scheme based on Q-learning algorithm of reinforcement Learning. According to the usage of link bandwidth in the network, the algorithm adjusts the transmission rate of the source data stream continuously, and finally obtains the optimal data stream allocation scheme. Experimental simulation evaluates the performance of the algorithm, the proposed control algorithm based on reinforcement learning congestion, can effectively improve the link utilization, reduce network congestion. The experimental results show that the congestion control algorithm proposed in this paper can allocate the optimal rate for each data stream, reduce network congestion, and has better performance than the traditional SDN algorithm.

1. Introduction

Compared with the traditional network, software defined network (SDN) separates the control and of data and provides convenience for data transmission. But with the rapid growth of network, network traffic is also growing rapidly, which easily leads to network congestion. Therefore, this paper designs a congestion control method based on reinforcement learning to optimize network services.

Tomovic introduces the concept of fairness in SDN[1]. This framework is based on resource reservation, optimal path selection and access control using pox controller, which purpose is to ensure the QoS required by priority flow and minimize the descent of best effort flow. Guillen proposes a hybrid load balancing scheme[2], which provides higher throughput for distributed storage system by using the multi-path function based on SDN. Yoon and Kamal propose a mixed integer linear programming model and a heuristic algorithm for local optimization based on simulated annealing[3], which can ensure the required QoS and minimize the energy consumption in SDN network. Lu Yifei proposed a scheme to limit the sending rate of the flow to avoid congestion and packet loss[4]. If the controller receives the congestion information, it adjusts the receiving window of the elephant stream ACK by calculation to limit the sending rate of the response stream, but it also increases the network transmission delay.

Reinforcement learning is the interaction between agent and environment to realize dynamic decision management. It has been widely used in path optimization, TCP network congestion control and ACM control, but it is rarely used in SDN congestion control. Therefore, aiming at the SDN congestion control strategy, this paper proposes to apply the reinforcement learning algorithm to the SDN congestion control, and make a decision on the flow sending rate in the SDN according to the characteristics of the algorithm. The algorithm has the following advantages: the algorithm can control the transmission of network flow in a fine-grained way, and dynamic congestion control according to the optimal Q-value function. In this paper, the performance of the algorithm is verified and analyzed by simulation software,
and experimental results show that the proposed SDN congestion control algorithm based on reinforcement learning can avoid congestion effectively.

2. Reinforcement learning

The standard form of Reinforcement Learning (RL) includes agent and interactive environment. The purpose of reinforcement learning is to enable the agent to obtain the largest cumulative reward in the process of interaction with the environment. The environment is usually described as a series of states $S^{\alpha \in Q} = S_{s, a}, R, P(s, a, r)$. RL usually uses Markov Decision Process (MDP) for problem modeling. MDP can be defined as a four-tuple $(S, A, R, P)$, the specific meaning is as follows:

1. $S$ represents the set of all states of the interactive environment, $s_t$ represents the state of the agent at time $t$;
2. $A$ represents the set of all actions that the agent can perform, and $a_t$ represents the action taken by the agent at time $t$;
3. $R$ represents the reward obtained by the agent after performing the action, $r_t = R(s_t, a_t)$ represents the instant reward obtained by the agent performing the action at in the state $s$;
4. $P(s'|s, a)$ represents the state transition probability, which represents the probability of the agent transitioning to the next state $s'$ after executing the action $a_t$ in the state $s_t$;

The deterministic strategy is the mapping from state to action: $\pi: S \rightarrow A$, which means that the action is taken in the $s$ state. The agent starts by sampling an initial state $s_0$ in each episode. The agent will generate an action based on the current state at each time step $t$: $a_t = \pi(s_t)$, and then get the reward $r_t = R(s_t, a_t)$ and the new state of the environment from the distribution $p(s'|s, a)$. The sum of discounts for future rewards is called the return:

$$R_t = \sum_{i=0}^{\infty} \gamma^{i-t} r_i$$ (1)

Among them, $\gamma \in [0, 1]$ represents the discount rate, and the reward from the subsequent state feedback is multiplied by the discount coefficient, indicating that the current reward is more important than the reward in the future. The goal of the agent is to maximize its expected reward $E_{S_0}[R_t|s_0]$, the Q function is defined as follow:

$$Q^\pi(s_t, a_t) = E[R_t|s_t, a_t]$$ (2)

Let $\pi^*$ denote the optimal strategy, for every $s \in S, a \in A$ and any strategy $\pi$, $Q^\pi(s, a) \geq Q^\pi(s, a)$. All optimal strategies have the same Q function, which is called the optimal Q function, and is marked as $Q^*$. It satisfies the following Bellman equation:

$$Q^*(s, a) = E_{s' \sim P_c(s, a)} [r(s, a) + \gamma \max_{a' \in A} Q^*(s', a')]$$ (3)

3. Q-Learning algorithm

3.1 introduce of algorithm

Reinforcement learning can be divided into two types: model-independent method and model-based method. One of the more commonly used in model-independent method is Q-learning algorithm.

Q-learning is value-based algorithm, it represents the expected value of taking action $A$ in some state $S$. The core concept of Q-learning is to store the value of $Q$ obtained by taking different actions $A$ in a certain state $S$ and form a table, and select the action with the highest payoff according to the value of $Q^\pi$.

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$ (4)

The above is the updated formula of Q-learning. $\alpha$ is the learning rate ($\alpha \in [0, 1]$). According to the next state $S'$, select the maximum $Q(s', a')$ value multiplied by the discount rate $\gamma$ plus the true return value as $Q$ reality, and take $Q(S, A)$ in the past $Q$ table as $Q$ estimation. It is to select the action by maximizing the value of the next state, so Q-learning belongs to the off-policy algorithm. Q-learning algorithm pseudocode is as follows:
Table 1. Q-learning algorithm

| Q-Learning Algorithm |
|-----------------------|
| • Initialize Q-function $Q(s, a)$, target Q-function $Q' = Q$ |
| • Repeat (for each episode) |
| • Initialize $S$ |
| • For each time step $t$ |
| • Given state $s_t$, take action $a_t$ based on $Q$ (epsilon greedy) |
| • Obtain reward $r_t$, and reach new state $s_{t+1}$ |
| Every steps reset $Q(s, a) = Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ |
| • $S \leftarrow s_{t+1}$ |
| • Until $S$ is terminal |

3.2 Improved Q-learning algorithm

At present, the reinforcement Learning has been used in the TCP network congestion control, but rarely applied to SDN congestion control. In this section, reinforcement learning is innovatively applied to SDN congestion control, it combines the feature of SDN network and the characteristics of the congestion control, controls the source side sends data flow rate and maximizes the link utilization and meets the demand of data flow by using the improved Q Learning algorithm to avoid congestion.

According to the reinforcement learning model described in 1.2, environment refers to the bandwidth usage of each link in SDN, and the controller is regarded as the agent. The controller changes the link bandwidth by controlling the transmission rate of the data stream at the source side, so as to obtain a reward $r$, and then updates the mapping table of the flow and rate. In this way, the learning process is repeated continuously, and the algorithm eventually converges, and finally a better mapping table is obtained. Where $a$ is an action output.

$$\pi'(s) = \arg \max_a Q^\pi(s, a)$$  \hspace{1cm} (5)

Table 2. Improved Q-Learning Algorithm

| Improved Q-Learning Algorithm |
|-----------------------------|
| Initialize Q-function $Q$, target Q-function $Q' = Q$, actor $\pi$, |
| set the environment rewards in matrix $R$, target actor $\pi' = \pi$ |
| • Repeat (for each episode) |
| • Initialize link state $S$ |
| • For each time step $t$ |
| • according to the current state and path of the flow, take action $a_t$ based on $\pi$ (exploration) |
| • Obtain reward $r_t$, and reach new state $s_{t+1}$ |
| • Store $(s_t, a_t, r_t, s_{t+1})$ into buffer |
| • Sample $(s_i, a_i, r_i, s_{i+1})$ from buffer (usually a batch) |
| • Target $y = r_t + Q'(s_{t+1}, \pi'(s_{t+1}))$ |
| • Update the parameters of $Q$ to make $Q(s_i, a_i)$ close to $y$ (regression) |
| • Update the parameters of $\pi$ to maximize $\pi(s_i, \pi(s_i))$ |
| • Every steps reset $Q' = Q$, $\pi' = \pi$ |
| • If link congestion, end |
| • Until $S$ is end state |

After the above steps, the algorithm finally converges to obtain a $Q$ matrix, through which a suitable rate can be allocated to each flow. It not only guarantees link utilization, but also avoids congestion.
3.3 Congestion control process

Section 2.2 describes how to use the Q-Learning algorithm for congestion control in SDN. This section will describe how to use the algorithm for congestion control.

The congestion control process based on Q-Learning is as follows:

1. Set multiple data streams for the current network, and obtain the path usage and flow rate requirements of these streams;
2. Obtain the initial bandwidth occupancy of the current link;
3. According to the Q matrix obtained in section 2.2, select the action \( a \) with the largest reward value as the flow distribution rate;
4. Recording the distribution rate of each flow, and finally form a \(<flow, rate>\) mapping table;
5. Determine whether the rate has been allocated for each stream. If the allocation is not completed, continue to step (3); otherwise, the controller will make a control decision based on the rate mapping table of each stream.

4. Experimental analysis

4.1 Experimental environment

This experiment is run in Vmware virtual machine. The Ubuntu operating system was installed in the virtual machine, and then Mininet emulator was installed to create a network topology environment.

The experimental topology used in this article is shown in Figure 1. There are 7 switches in the topology, S1 is the core switch, and the other switches are connected to the core switch. There are 6 links in total, namely L1, L2, L3, L4, and L5, L6, the bandwidth of each link is 30G.

![Network topology diagram](image)

Figure 1 Network topology diagram

The initial action set \( A = \{1G, 2G, 3G, 4G\} \) taken in the experiment, that is, the agent will randomly select a value from these four rates and assign it to each data stream. The initial value of \( \varepsilon \) is set to 0.99, and the final value is 0.01. That is, at a certain moment, the agent will select an action from \( A \) with a probability of 99\% to execute, and with a probability of 1\%, it will execute with the learned optimal value. As the algorithm converges, the value of \( \varepsilon \) keeps decreasing, down to 0.01.

For the setting of this article uses a single-peak reward function to set the rewards after the agent performs an action, and the link bandwidth threshold is set to 25G. When the link capacity exceeds the threshold, the reward is a negative value, indicating that the network is congested at this time; When the capacity does not exceed the threshold, the reward value is positive. After the agent performs the action, if the link utilization rate increases, the reward value will increase relatively.

4.2 Performance comparison

Figure 2 shows the bandwidth changes of the 6 links and the rate distribution of each data stream. It can be seen from the figure that after the fourth data stream is allocated, the link L1 is congested. Therefore, if the congestion control method is not adopted, but the required rate is directly allocated to each data stream, the allocation of 10 data streams cannot be completed.
Figure 2 Changes in link bandwidth allocated on demand

Figure 3 Link bandwidth changes based on Q-Learning algorithm

Figure 3 shows the change of link bandwidth after using the Q-Learning algorithm. It can be seen from the figure that after the rate of each data stream is allocated, the link bandwidth does not exceed 30G, and the network is not congested.

In summary, it can be concluded that if each data stream meets its rate requirements, the link will be congested. After the algorithm proposed in this paper allocates rates for 10 data streams, the bandwidth of all links exceeds the threshold limit, and the link does not appear to be congested. Therefore, the congestion control method based on reinforcement learning proposed in this paper is feasible.

5. Conclusion

In this paper, the knowledge of reinforcement learning is applied to SDN congestion control, so that the controller can allocate an appropriate rate for each flow through training. Compared with traditional congestion control methods, this algorithm combines the knowledge of artificial intelligence and has a certain degree of innovation. This article only uses the improved Q-Learning algorithm. There are many other algorithms for reinforcement learning. Future work will introduce other algorithms for further comparative analysis.

Acknowledgments

Young Innovative Talents Project of Guangdong Province (2018KQNCX312)

References

[1] S. Tomovic, N. Prasad, and I. Radusinovic. \“SDN control framework for QoS provisioning,” in Proc. 22nd Telecommun. Forum Telfor (TELFOR), Belgrade, Serbia, Nov. 2014, pp. 111–114.

[2] L. Guillen, S. Izumi, T. Abe, T. Suganuma, and H. Muraoka. “SDN–based hybrid server and link load balancing in multipath distributed storage systems,” in Proc. NOMS-IEEE/IFIP Netw. Operations Manage. Symp. Apr. 2018, pp. 1–6.

[3] M. S. Yoon and A. E. Kamal, “Power minimization in fat-tree SDN datacenter operation,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), San Diego, CA, USA, Dec. 2015, pp. 1–7.

[4] Lu Fei, Zhu Shuhong et al. Research and implementation of TCP congestion control mechanism based on SDN in data center network [J]. Journal of Computers, 2017, 40(9): 2167–2180.

[5] Liu C, Ju W, Zhang G, et al. A SDN-based active measurement method to traffic QoS sensing for smart network access[J]. Wireless Networks, 2020(2).

[6] Che Xiangbei, Kang Wenqian, Ouyang Yuhong, Yang Kehan, Li Jian. SDN routing optimization algorithm based on reinforcement learning [J/OL]. Computer Engineering and Applications: 1-7 [2020-07-29].

[7] Tuo Xin. Research on SDN Data Center Network Congestion Control Method Based on Reinforcement Learning [D]. Zhejiang Gongshang University, 2018.