AIMD Window Flow Control using Reinforcement Learning

Kento OTANI[†a)], **Shota INOUE**[†b)], **Keita GOTO**[†c)], ***Student Members***, ***and*** **Hiroyuki OHSAKI**[†d)], ***Member***

1. Introduction

Machine-learning-based traffic control techniques that do not require prior knowledge of the internal state of the network have been actively studied. Sacco et al. [1] proposed Owl, a window flow control method that uses Q-learning, a type of reinforcement learning. In Owl, each sending host performs Q-learning with the current window size, round-trip time, etc. as the state and the amount of window size change as the action. Each sending host updates the Q-value corresponding to (state, action) based on the throughput and packet loss rate resulting from the window size change.

Owl, a window flow control scheme based on reinforcement learning, changes the window size additively (i.e., addition and subtraction) as an action of the sending host. Conventional window flow control schemes, such as TCP NewReno, use Additive-Increase and Multiplicative-Decrease (AIMD) type window size adjustment. We believe that AIMD-type window size adjustment is also effective in window flow control schemes using reinforcement learning.

This study proposes Q-learning-based AIMD window flow control (Q-AIMD) to change the congestion window in an AIMD fashion and clarifies its effectiveness through experiments. Specifically, the state of Q-learning in the conventional Owl method is limited to the window size, throughput, and packet loss rate, and the action of Q-learning is given by the addition and multiplication of window sizes. Furthermore, through simulations, we show that Q-AIMD performs well when two flows compete in a dumbbell topology.

2. Q-AIMD

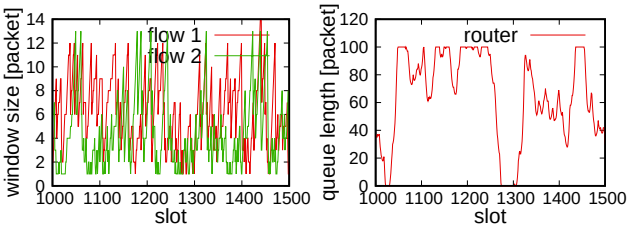
Q-AIMD is a window flow control scheme that operates on the sending host of the TCP transport protocol. It determines the number of packets (i.e., window size) that can be sent out into the network during the round-trip time by Q-learning. Although the basic idea of Q-AIMD is based on Owl [1], the action in Q-learning is different.

The sending host in Q-AIMD updates the Q-table at intervals of Δ and increases or decreases the window size based on the current Q-table. The state s of each flow in Q-AIMD is the window size, and the action a of each flow is of two types: adding α_n and multiplying by β_n. α_n and β_n ($n \geq 1$) are control parameters. The action a in the conventional Owl method only corresponds to the case with different α_n.

Similar to Owl, Q-AIMD calculates the reward for the current state and action pair (s, a) by measuring the current throughput and the packet loss rate from the ACK returned by the receiving host, and it updates the corresponding Q-value in the Q-table. The reward r is given by the following equation based on the measured throughput λ and packet loss rate p[1].

$$r = \lambda - \delta \lambda (1 - p)^{-1}, \tag{1}$$

where δ is a parameter that adjusts the weights of the throughput and packet loss rate.



(a) Variation of window size (b) Variation in the number of packets in the buffer

Fig. 1 Q-AIMD simulation results

3. Simulation

In a dumbbell network consisting of two flows and a single router running Q-AIMD, we measured the evolution of window sizes when two flows continuously send data to their corresponding receiving hosts.

For both flows, the control interval was set to Δ = 100 [ms]; round-trip propagation delay was set to 100 [ms]; minimum window size was set to 1 [packet]; and the parameters for changing the window size were set to $\alpha_1 = 0$, $\alpha_2 = 1$, and $\beta_1 = 1/2$. The learning rate for Q-learning was set to 0.1, the discount rate was set to 0.1, and the weight of the reward r was set to $\delta = 0.5$. The router bandwidth was set to 0.1 [packets/ms], and the buffer size was set to 100 [packets].

Figure 1 shows the evolution of the window sizes of two flows running Q-AIMD and the queue length of the router. These two figures show the temporal variation of the window size and number of packets in the router's buffer for 1,000–1,500 slots, where Q-learning is advanced to some extent. This ensures that the window size of the two flows is properly controlled by the Q-AIMD.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 19H04104.

References

[1] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Owl: congestion control with partially invisible networks via reinforcement learning," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 2021)*, pp. 1–10, May 2021.

[†]The author is with Kwansei Gakuin University, Uegahara, Gakuen Sanda, Hyogo 669-1330, Japan.
a) E-mail: kento-o@kwansei.ac.jp
b) E-mail: shota-i@kwansei.ac.jp
c) E-mail: kei-gt@kwansei.ac.jp
d) E-mail: ohsaki@kwansei.ac.jp