Pricing Scheme based Nash Q-Learning Flow Control for Multi-user Network

Xin Li\textsuperscript{a}, Haibin Yu\textsuperscript{b}

Shenyang Institute of Automation, Chinese Academy of Sciences
Shenyang, Liaoning, 110016, China
\textsuperscript{a}xinli@sia.cn, \textsuperscript{b}yhb@sia.cn

Keywords: pricing scheme, Nash Q-learning, multi-user.

Abstract. For the congestion problems with multi-user existing in high-speed networks, a pricing scheme based Nash Q-learning flow controller is proposed. It considers a network with a single service provider, and some non-cooperative users. The pricing scheme is introduced to the design of the reward function in the learning process of Q-learning. Because of the uncertainties and highly time-varying, it is not easy to accurately obtain the complete information for high-speed networks. The Nash Q-learning, which is independent of mathematic model, shows particular superiority. It obtains the Nash Q-values through trial-and-error and interaction with the environment to improve its behavior policy. By means of learning process, the proposed controller can learn to take the best actions to regulate source flow with the features of high quality of service. Simulation results show that the proposed controller can promote the performance of the networks and avoid the occurrence of congestion effectively.

Introduction

The growing interest on congestion problems in high-speed networks arise from the control of sending rates of traffic sources. Congestion problems result from a mismatch of offered load and available link bandwidth between network nodes. Such problems can cause high packet loss ratio (PLR) and long delays, and can even break down the entire system because of congestion collapse. Therefore, high-speed networks must have an applicable flow control scheme not only to guarantee the quality of service (QoS) for the existing links but also to achieve high system utilization.

The flow control of high-speed networks is difficult owing to the uncertainties and highly time-varying of different traffic patterns. The flow control mainly checks the availability of bandwidth and buffer space necessary to guarantee the requested QoS. A major problem here is the lack of information related to the characteristics of source flow. Devising a mathematical model for source flow is the fundamental issue. However, it has been revealed that this is a very difficult task, especially for broadband sources.

In order to overcome the above-mentioned difficulties, the control schemes with learning capability have been employed in flow control [1, 2]. The basic advantage is the ability to learn the source traffic characteristics from sufficiently big and representative data samples. But it is obvious that the accurate data, needed to train the parameters, are hard to get for the disturbance and error in instrument measuring. In this circumstance, the reinforcement learning (RL), with the ability of self-learning, shows its particular superiority. It is independent of mathematic model and just needs very simple information such as estimable and critical information, “right” or “wrong” [3]. It obtains the knowledge through interaction with environment to improve its behavior policy. In [4], authors proposed a call admission controller combined RL with neuro-fuzzy network. In [5], authors proposed a congestion controller for single-bottleneck high-speed network. But all the users considered in the literatures above are noncompetitive, in another word, the users are cooperative. The users share the available bandwidth fairly.
In this paper, we proposed a pricing scheme based Nash Q-learning flow controller considered the network with \( n \) non-cooperative users. In the proposed controller, each user has a separate memory structure to explicitly implement its own objectives to achieve Nash equilibrium Q-values. The Nash equilibrium solution serves as the optimal sending rate of traffic sources. The reward values are determined by the bandwidth price the users should pay for. By means of learning process, the controller adjusts the source sending rate to the optimal value to reduce the average length of queue in the buffer. The simulation results show that the proposed controller can avoid the occurrence of congestion effectively with the high network performance.

**Theoretical Framework**

1) Architecture of the Proposed Controller

In high-speed networks, it is hard to achieve high system performance by reactive AIMD scheme because of the propagation delay and the dynamic nature. Whereas the proposed controller can behave optimally only rely on the interaction with unknown network environment and provide the best action for a given state. The architecture of controller with \( n \) non-cooperative users is shown in Fig.1.

![Fig. 1 The network model with \( n \) non-cooperative users](image)

The input of the controller is the state variable \( (S) \), which is composed of the current queue length \( (b) \) and the current change of queue length \( (\Delta b) \). The output of the controller is the action variable \( (U) \), which is the feedback signals \( (u) \) determining the controllable source sending rates. However, all agents have incomplete but perfect information, meaning agents do not know other agents’ reward functions and state transition probabilities, but they can observe other agents’ immediate payoffs and actions taken previously. By way of multi-agent policy-search learning, the proposed controller uses joint-action learning algorithms to learn values for joint actions, so that it could converge to Nash equilibrium and avoid the bias in individual action decision.

2) Architecture of the Reward Function

In this paper, the network model we considered is composed of \( n \) non-cooperative users and a single bottleneck link. All the users want to have the maximum network bandwidth when they have packets preparing to transmit. For the purpose of having good network performance, the pricing scheme is introduced in the flow control in many literatures [6, 7].

In order to reach the co-ordination of supply and demand, the manager of the network sets different level of price for different kinds of users. In this paper, the users of network are divided into two kind by the different QoS. One kind of user is uncontrollable, the price of which is a constant. The other kind of users is controllable, the price of which is determined by the unavailable bandwidth. The curves of the two pricing scheme are shown in Fig. 2, where \( B \) is the buffer length of the switch, \( b^* \) is the desired queue length of buffer, \( p\% \) is the reference price determined by the manager of the network.
For the kind of user 2, when the queue length is larger than the desired value $b^*$, the price for using the network is increased rapidly along with the increasing of the queue length.

![Fig. 2 The curves of the two pricing scheme](image)

For user 1, the price is a constant $\tilde{p}$ regardless of the situation of the network. For user 2, the price is given as following

$$p_i(b) = \begin{cases} 
\tilde{p}_i & \text{if } 0 \leq b \leq b^* \\
\tilde{p}_i + \hat{p}_i(b) & \text{if } b^* \leq b \leq B 
\end{cases}$$

(1)

where $\hat{p}$ is the punish function given to the user by the network manager considered the situation of buffer. In this paper, we take $\hat{p}$ as

$$\hat{p}_i(b) = c_i \left( \frac{b}{B} \right)^{m_i}.$$  

(2)

In the Q-learning based learning process, the reward value $r$ is the only information for the controller to judge whether the sending rate taken is good or bad, so it is vital to choose an appropriate $r$. In this paper, based on the requirement and experience of the buffer and the pricing scheme, the reward function $r'$ for user $i$ is defined as

$$r' = r' \cdot U(u_i) - (p u_i + k \bar{u}_i).$$

(3)

where $p_i$ and $u_i$ is the bandwidth price and the source sending rate for user $i$, $\bar{u}_i$ is the total source sending rate for all the users. $U(u_i)$ is the income at the sending rate $u_i$ for user $i$, it is determined by the QoS of user. In this paper, we take the form as [8]

$$U(u_i) = u_i \cdot \log_2 u_i.$$  

(4)

In equation (3), $r'$ is the factor reflected the situation of buffer usage, it has the form as following

$$r' = \begin{cases} 
0 & b \geq 1.1b^* \text{ or } b \leq 0.9b^* \\
\frac{1.1b^* - b}{0.1b^*} & b^* < b < 1.1b^* \\
\frac{b - 0.9b^*}{0.1b^*} & 0.9b^* < b < b^* \\
1 & b = b^* 
\end{cases}.$$  

(5)
Refer to (5), if the value of queue length \( b \) is less than \( 0.9b^* \) or more than \( 1.1b^* \), the control result should be considered bad. If the value of \( b \) is equal to \( b^* \), \( r' = 1 \), it can be thought that the control result is good. Otherwise, \( r' \) is in the range \( (0, 1) \), the larger \( r' \) is, the better control affects.

3) Design of Nash Q-learning Flow Controller

Q-learning is a value learning version of model-free reinforcement learning that learns utility values (Q-values) of state and action pairs [12]. The objective of Q-learning is to estimate Q-values for an optimal strategy.

In general form, an n-agent is defined by a tuple \( < n, S, A^1, ..., A^n, r^1, ..., r^n, P > \), where \( n \) is the number of agents, in another word the number of users in high-speed networks; \( S \) is a set of discrete state space of high-speed networks composed of \( (b, \Delta b) \); \( A^1, ..., A^n \) is a collection of actions (feedback control signal to traffic sources) available to each agent (\( A^i \) is the discrete action space available to agent \( i \)); In this paper, we take \( u' \in \{ 0.25u_{\max}', 0.5u_{\max}', 0.75u_{\max}', u_{\max}' \} \), where \( u_{\max}' \) is the maximum sending rate for user \( i \); \( P \) is the transition probability map.

In Nash Q-learning flow controller, the objective of each agent is to maximize the discounted sum of rewards, with discount factor \( \beta \in [0,1] \). Let \( \pi' \) be the action strategy of agent \( i \). For a given initial state \( s \), agent \( i \) tries to maximize

\[
v'(s, \pi^1, ..., \pi^n) = \sum_{t=0}^{\infty} \beta^t E \left( r_t \left| \pi^1, ..., \pi^n, s_0 = s \right. \right)
\]

(6)

A strategy \( \pi = (\pi_0, ..., \pi_t, ...) \) is defined over the whole course of learning process. \( \pi_t \) is called the decision rule at sample time \( t \).

The Nash equilibrium solution is a tuple of \( n \) strategies \( (\pi^1, ..., \pi^n) \) such that for all \( s \in S \) and \( \pi^i \in \Pi^i \),

\[
v'(s, \pi^1, ..., \pi^n) \geq v'(s, \pi^1, ..., \pi_i^{*1}, \pi^i', \pi_i^{*i1}, ..., \pi^n)
\]

(7)

where \( \Pi^i \) is the set of strategies available to agent \( i \). The definition of Nash equilibrium requires that each agent’s strategy is a best response to the other’s strategy.

The Nash Q-function for the \( i \)th agent is defined over \( \left( s, a^1, ..., a^n \right) \) as the sum of its current reward plus its future rewards when all agents follow a joint Nash equilibrium strategy. That is

\[
Q(s, a^1, ..., a^n) = r'(s, a^1, ..., a^n) + \beta \sum_{s' \in S} P(s'|s, a^1, ..., a^n) v'(s', \pi^1, ..., \pi^n)
\]

(8)

where \( (\pi^1, ..., \pi^n) \) is the joint Nash equilibrium strategy, \( r'(s, a^1, ..., a^n) \) is agent \( i \)’s one-period reward in state \( s \) under joint action \( (a^1, ..., a^n) \). \( v'(s', \pi^1, ..., \pi^n) \) is agent \( i \)'s total discounted reward over infinite periods starting from state \( s' \) given that agents follow the equilibrium strategies.

The learning agent, indexed by \( i \), learns about its Q-values by forming an arbitrary guess at time 0. One simple guess would be letting \( Q(s, a^1, ..., a^n) = 0 \) for all \( s \in S, a^1 \in A^1, ..., a^n \in A^n \). At each time \( t \), agent \( i \) observes the current state, and takes its action. After that, it observes its own reward, actions taken by all other agents, others’ rewards, and the new state \( s' \). It then calculates a Nash equilibrium \( \pi^1(s')...\pi^n(s') \) for \( (Q^1(s'), ..., Q^n(s')) \), and updates its Q-values according to

\[
Q_{t+1}(s, a^1, ..., a^n) = (1 - \alpha) Q_t(s, a^1, ..., a^n) + \alpha \left[ r'_t + \beta \text{Nash} Q(s') \right]
\]

(9)

where \( \beta \in [0,1] \) is the discount factor, if \( \beta \) is large, systems will easily tend to follow the current strategy so that it will not have more opportunities to find a better strategy; if \( \beta \) is small, systems will not easily follow a strategy so that it will do explorations all the time. This will cause the convergence rate to be slow. On the other hand, \( \alpha \in [0,1] \) is the learning rate. The convergence rate is determined.
by the value of $\alpha$. If $\alpha$ is small, the convergence rate will be slow but it will easily tend to stabilize. If $\alpha$ is large, the convergence rate will be fast but it will not easily tend to stabilize.

The NashQ$^i$ ($s'$) is defined as

$$\text{NashQ}^i (s') = \pi^1 (s') \cdot \pi^n (s') Q^i (s') .$$

(10)

In order to calculate the Nash equilibrium ($\pi^1 (s') \ldots \pi^n (s')$), agent $i$ would need to know $Q^i (s'), \ldots, Q^n (s')$. Information about other agents' Q-values is not given, so agent $i$ must learn about them too. Agent $i$ forms conjectures about those Q-functions at the beginning of learning, for example, $Q^i (s, a^1, \ldots, a^n) = 0$ for all $j$ and all $s, a^1, \ldots, a^n$. As the learning process, agent $i$ observes other agents’ immediate rewards and previous actions. That information can then be used to update agent $i$’s conjectures on other agents’ Q-functions. Agent $i$ updates its beliefs about agent $j$’s Q-function according to the same updating rule (9) it applies to its own,

$$Q^i_{t+1} (s, a^1, \ldots, a^n) = (1 - \alpha_t) Q^i_t (s, a^1, \ldots, a^n) + \alpha_t [r'_j + \beta \text{NashQ}^j (s')].$$

(11)

Note that $\alpha=0$ for $(s, a^1, \ldots, a^n) \neq (s, a^1, \ldots, a^n)$. Therefore (11) does not update all the entries in the Q-functions. It updates only the entry corresponding to the current state and the actions chosen by the agents.

**Simulation Results**

In the simulation, we adopt a single bottleneck network model shown in Fig. 3. We assume that all packets are with a fixed length of 1000 bytes, and adopt a finite buffer length of 50 packets in the switch. The desired queue length in the buffer is set at 10 packets, and the bandwidth of the bottleneck link is 80 Mbps.

In the network, there are four kinds of sources. S1 is the uncontrollable user as user 1 in Fig. 2. S2, S3, and S4 are the controllable user as user 2 in Fig. 2. The bandwidth is first allocated to the S1, and the other is allocated to the other users. The income functions for S2, S3, and S4 are selected as $U_2 (u) = u \cdot \log u$, $U_3 (u) = u \cdot \ln u$, and $U_4 (u) = u \cdot \lg u$.

In the simulation process, the sources generate packets randomly and there still have packets waiting to be transmitted all the time. The curve of queue length in the buffer is shown in Fig. 4. From the simulation result we can see that the queue length in the buffer reaches the desired value 10 packets quickly at about 2s, and then oscillating slightly. There is no congestion occurred in the network and the network resources are used sufficiently. So we can have the conclusion that the proposed controller has a good network performance with the non-cooperative users existing.

After the learning process is being executed 6000 steps, the statistics of probability of sending rate selection for users is shown in Fig. 5. From the simulation result we can see that the users have the ability to select higher sending rate as possible as they can by the adjusting scheme of proposed controller.
Conclusion

In high-speed networks, most packet losses result from the dropping of packets owing to congested nodes. In order to cope with the congestion problems, we proposed a pricing scheme based Nash Q-learning flow controller. Considering the existing of non-cooperative users, the pricing scheme is introduced in the design of the reward function of Nash Q-learning. Through a proper training process, the proposed controller can learn empirically without prior information on the environmental dynamics. The sending rate of traffic sources can be determined by the well-trained Nash Q-values. Simulation results have shown that the proposed method can increase the utilization of the buffer and avoid the congestion occurrence simultaneously.

References

[1] R. G. Cheng, C. J. Chang and L. F. Lin: A QoS-provisioning Neural Fuzzy Connection Admission Controller for Multimedia High-speed Networks. IEEE/ACM Transactions on Networking, vol. 7, no. 1 (1999), p. 111-121.

[2] M. Lestas, A. Pitsillides, P. Ioannou, and G. Hadjipollas: Adaptive Congestion Protocol: a Congestion Control Protocol with Learning Capability. Computer Networks, vol. 51, no. 13 (2007), p. 3773-3798.

[3] R. S. Sutton and A. G. Barto: Reinforcement Learning an Introduction. (MIT Press, Cambridge, 1998).

[4] A. Chatovich, S. Okug, and G. Dundar: Hierarchical Neuro-fuzzy Call Admission Controller for ATM Networks. Computer Communications, vol. 24, no. 11 (2001), p. 1031-1044.

[5] M. C. Hsiao, S. W. Tan, K. S. Hwang, and C. S. Wu: A Reinforcement Learning Approach to Congestion Control of High-speed Multimedia Networks. Cybernetics and Systems, vol. 36, no. 2 (2003), p. 181-202.

[6] F. P. KELLY, A. MAULLOO and D. TAN: Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. Journal of the Operations Research Society, vol. 49 (1998), p. 237-252.

[7] J. Hou, J. Yang, and S. Papavassiliou: Integration of Pricing with Call Admission Control to Meet QoS Requirements in Cellular Networks. IEEE Transactions on Parallel and Distributed System, vol.13, no.9 (2002), p.898-910.

[8] Basar T, Srikant R. Revenue Maximizing Pricing and Capacity Expansion in a Many-Users Regime. Proceedings of the IEEE INFOCOM, New York, p. 321-329 (2002).