Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Metropolis Criterion Based Q-Learning Flow Control for High-Speed Networks

Xin Li*. Yuanwei Jing*. Georgi M. Dimirovski. Siying Zhang***

**Institute of Information Science & Engineering,*
Northeastern University, Shenyang, Liaoning, 110004, P.R. of China
(e-mail: lixin820106@ 126.com; ywjjing@ mail.neu.edu.cn; zhangsiying@ ise.neu.edu.cn).
***Dogus University of Istanbul, Faculty of Engineering, TR-347222 Istanbul,*
Rep. of Turkey, and with SS (e-mail: gdimirovski@ dogus.edu.tr)

Abstract: For the congestion problems in high-speed networks, a Metropolis criterion based Q-learning flow controller is proposed. Because of the uncertainties and highly time-varying, it is not easy to accurately obtain the complete information for high-speed networks. The Q-learning algorithm, which is independent of mathematic model, shows the particular superiority in high-speed networks. It obtains the optimal Q-values through interaction with the environment to improve its behavior policy. The Metropolis criterion of simulated annealing algorithm can cope with the balance between exploration and exploitation in Q-learning. By means of learning procedures, the proposed controller can learn to take the best action to regulate source flow with the features of high throughput and low packet loss ratio. Simulation results show that the proposed method can promote the performance of the networks and avoid the occurrence of congestion effectively.

1. 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 network system because of the 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 flow control scheme with learning capability has been employed in flow controller of high-speed network (Cheng *et al*., 1999; Lestas *et al*., 2007). Of all the learning algorithms, the reinforcement learning (RL) shows its particular superiority, which just needs very simple information such as estimable and critical information, "right" or "wrong" (Sutton *et al*., 1998). RL is independent of mathematic model and priori-knowledge of system. It obtains the knowledge through trial-and-error and interaction with environment to improve its behavior policy. So it has the ability of self-learning. Chatovich proposed a call admission controller combined RL with neuro-fuzzy network (Chatovich *et al*., 2001). Hsiao proposed a RL congestion controller for high-speed network, consisting of two subsystems called action-value evaluator and stochastic action selector (Hsiao *et al*., 2003, 2005).

The Q-learning algorithm of RL is easy for application and has a firm foundation in the theory. But the balance between exploration and exploitation is one of the key problems of action selection in Q-learning (Kaelbling *et al*., 1996). The Metropolis criterion of simulated annealing (SA), simulating an annealing process on computer, is a powerful way to solve hard combinatorial optimization problems (Metropolis *et al*., 1953). So it is applied to control the balance between exploration and exploitation (Guo *et al*., 2004).

In this paper, based on the Metropolis criterion based Q-learning algorithm, a SA-Q-learning flow controller (SAQFC) for high-speed networks is proposed. In the controller proposed, the learning agent has a memory structure to explicitly implement its objectives to achieve optimal Q-value. The optimal Q-value serves as the optimal sending rate of traffic sources. By means of learning procedures, the proposed controller adjusts the source sending rate to the optimal value to reduce the average length of queue in the buffer. Simulation results show that the proposed method can avoid the occurrence of congestion effectively with the features of high throughput, low PLR, low end-to-end delay, and high utilization.

2. THEORETICAL FRAMEWORK

2.1 Architecture of the Proposed Flow Controller

In the AIMD case, the agent senses the network system's states and makes a decision based on a rate control scheme to

avoid packet losses and increase the utilization of multiplexer's output bandwidth (Gevros *et al*., 2001). However, it is hard to achieve high system performance by reactive AIMD scheme because of the propagation delay and the dynamic nature of high-speed networks. Whereas the proposed SAQFC can behave optimally only rely on the interaction with the unknown environment and provide the best action for a given state. The detailed architecture of the proposed SAQFC is shown in Fig.1.

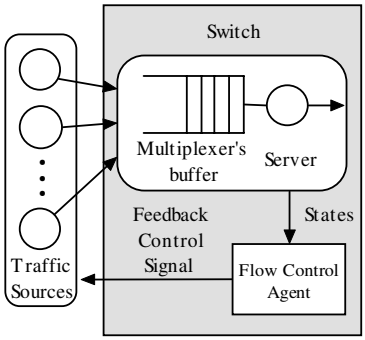

Fig. 1. Architecture of the proposed SAQFC

In high-speed networks, SAQFC in bottleneck node acts as a flow control agent with flow control ability. The inputs of SAQFC are state variables (S) composed of queue length (q) and sending rate (u) in the network. The output of SAQFC is the feedback signal (a) to the traffic sources, which means the probability mass function (PMF) of sending rate. The controlled sending rate u_t at sample time t is defined by $u_t = a_t \cdot u_{\max}$, where $u_{\max}$ is the maximum sending rate of the traffic sources. Reinforcement signal r for a specified state is denoted as r=1 for reward, and r=0 for penalty (Littman, 2001). The learning agent and the network environment interact continually in the learning process. At the beginning of each time step of learning, the controller senses the states for the network and gets the reinforcement signal (reward). Then it selects an action to make decision on which rate the sources should use to reduce the PLR and increase the link utilization. After the controller selects an action, the network responds to the action by changing its state and giving a new reward to the controller. Then the next step of learning begins.

2.2 Associative State Space Partitioning

The input state variables (q,u) for a node are equally divided by ten partitions numbered 0-9, respectively. The number of state variables of the system is 10×10=100. The vector of quantized input values specifies a discrete state and is used to generate the addresses for retrieving information from memory for this state.

In order to reduce the number of state variables of the system, a series of mapping as S→I→H is performed. Where S is the input state vector, I denotes a set of intermediate variables, H is the state vector through transforming. In S→I mapping, each of variables q and u has three different ways of block decomposition, each with four blocks. For state s_{55}, as depicted in Fig. 2, u=5 is mapped into the set I_1={B,G,K}, and q=5 is mapped into the set I_2={b,g,k}. In I→H mapping, each hypercube is assigned a unique name that is the combination of intermediate variables. In this case, the intermediate variables are mapped into a set of three hypercubes. i.e., I_1 and I_2 are mapped to H={Bb,Gg,Kk}, where Bb, Gg, and Kk specify three hypercubes. Furthermore, the number of state variables of the system is reduced from 100 to 4×4×3=48.

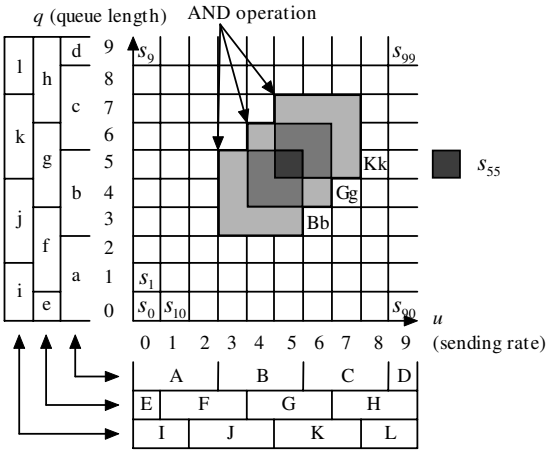

Fig. 2. The mapping for state variables in the network node

2.3 The Q-learning Algorithm

Q-learning is an algorithm that belongs to the class of reinforcement learning algorithms. Reinforcement learning is concerned with the problem how an agent can learn to behave optimally from interactions with its environment. The general idea of reinforcement learning is as follows. An agent interacts repeatedly with its environment. During each interaction, the agent first observes the state of the environment $s \in$ S. The agent then decides to execute an action a$\in$A. This results in a payoff r that is received by the agent and in a transition of the state of the environment from the old state s to the new state s'. Because the state of the environment changes as a result of the action that was executed by the agent, the choice of an action may not only influence the agent's immediate payoff r but also its payoffs in future periods. The agent's payoff and the new state of the environment only depend (either deterministically or stochastically) on the old state of the environment and on the action that was executed by the agent. In reinforcement learning it is typically assumed that the agent has no prior knowledge of the payoff function $r(s,a)$ and the state transition function, so the agent has no model of its environment. The goal of the agent is to find an optimal policy π^*: S→A for choosing actions. A policy $\pi(s)$ is optimal if in each state $s \in$ S it selects an action $a \in$ A that maximizes the agent's cumulative payoff, which is the sum of its immediate payoff and its future payoffs.

In Q-learning algorithm, the objective of learning agent is to maximize the discounted sum of rewards, with discount factor $\beta \in [0,1)$. Let π be the policy of learning agent, for a given initial state s, agent tries to maximize

$$v(s,\pi)=\sum_{t=0}^{\infty}\beta^t E\left(r_t\left|\pi,s_0=s\right.\right). \quad (1)$$

The policy π is defined over the whole course of the learning process. π_t is the decision rule at time t. We can rewrite (1) as

$$v(s,\pi)=r(s,a_\pi)+\beta\sum_{s'}p(s'|s,a_\pi)v(s',\pi), \quad (2)$$

where a_π is the action determined by policy π, $r(s,a)$ is the reward for taking action a at state s, s' is the next state, and $p(s'|s,a)$ is the probability of transiting to state s' after taking action a in state s. It has been proved that there exists an optimal policy π such that for any $s\in$S, the following Bellman equation holds

$$v(s,\pi^*)=\max_a\left\{r(s,a)+\beta\sum_{s'}p(s'|s,a)v(s',\pi^*)\right\}, \quad (3)$$

where $v(s,\pi^*)$ is called the optimal value for state s.

The basis idea of Q-learning is that we can define the Q-function as

$$Q^*(s,a)=r(s,a)+\beta\sum_{s'\in S}p(s'|s,a)v(s',\pi^*). \quad (4)$$

By this definition, $Q^*(s,a)$ is the total discounted reward attained by taking action a in state s and then following the optimal policy thereafter. Then by (3) we have

$$v(s,\pi^*)=\max_a Q^*(s,a). \quad (5)$$

If we know $Q^*(s,a)$, then the optimal policy π^*, which is always taking an action so as to maximize $Q^*(s,a)$ under any state s, can be found. The problem is then reduced to finding the function $Q^*(s,a)$ instead of searching for the optimal value of $v(s,\pi^*)$.

Q-learning provides us with a simple updating procedure, in which the learning agent starts with arbitrary initial values of $Q(s,a)$ for all $s\in$S, $a\in$A, and updates the Q-values as

$$Q_{t+1}(s_t,a_t)=(1-\alpha)Q_t(s_t,a_t)+\alpha\left[r_t+\beta\max_a Q_t(s_{t+1},a)\right], \quad (6)$$

where $0\le\alpha<1$ is the learning rate. The convergence rate is determined by the value of α. If α is small, the convergence rate will be slow but it will easily tend to stabilize. On the other hand, if α is large, the convergence rate will be fast but it will not easily tend to stabilize. The update rule allows an agent that does not know the payoff function and the state transition function to learn the values of the Q function and, consequently, to find an optimal policy for choosing actions. Watkins and Dayan has proved that sequence (6) converges to optimal $Q^*(s,a)$ with probability 1 (Watkins *et al.*, 1992).

However, finding the proper balance between exploration and exploitation in Q-learning is one of the key problems in action selection. On the one hand, an agent may want to explore unknown states and actions to collect new information about its environment. On the other hand, an agent may want to exploit its current knowledge of the environment by executing the action that is expected to maximize the cumulative payoff. When the selection is greedy (i.e., exploitation), it will lead to locally optimal policies that possibly differ from a globally optimal one. On the other hand, excessive exploration will drastically decrease the performance of a learning algorithm, and in some cases might be even harmful with respect to the learning results themselves.

A popular strategy proposed to deal with this problem is the ε-greedy strategy (with $0\le\varepsilon<1$). In the ε-greedy strategy, the action with the highest Q-value is selected with probability $1-\varepsilon$. With probability ε an action is selected randomly using a uniform distribution over all actions. Here, the value of ε has obviously a great impact on the algorithm. Sutton compared the performance of learning for different ε values and concluded that the result for a nonzero ε is usually better than that for ε equal 0, which means that ε-greediness is effective (Sutton *et al.*, 1998). However, excessive exploration becomes unnecessary after a period of an initial interaction between the agent and the environment.

Therefore, we have decided to explore the possibility of improving the simple ε-greedy strategy by appropriately reducing ε during the learning process. This will not only improve the ability of agent to acquire new knowledge, but will also allow the algorithm to avoid performance decrease due to the constant value of ε. Thus, the task of finding the optimal policy in Q-learning is transformed into searching for an optimal solution in a combinatorial optimization problem. Then the Metropolis criterion, the core of simulated annealing is applied to the searching procedure in order to control the balance between exploration and exploitation.

2.4 The Metropolis Criterion of Simulated Annealing

Simulating the annealing process of solids, the SA algorithm is one kind of the computational processes resembling nature and has been shown to be an effective approximate algorithm to solve combinatorial optimization problems. An important characteristic of the SA algorithm is that it does not require specialist knowledge about how to solve a particular problem. This makes the algorithm generic in the sense that it can be used in a variety of optimization problems without changing the basic structure of computations. The procedure employs methods that originated from statistical mechanics to find global minima of systems with very large degrees of freedom. The major advantage of SA algorithm over the pure local search method is the ability to avoid becoming trapped in a local solution.

In SA algorithm, the transition probability $P(i\to j)$ of the Metropolis criterion is used to decide whether the transition from the current state (solution) i to the new state j occurs. $P(i\to j)$ can be defined as follows:

$$P(i\to j)=\begin{cases}1, & \text{if } f(j)\ge f(i)\\ \exp\left(\left(f(j)-f(i)\right)/T\right), & \text{otherwise}\end{cases}, \quad (7)$$

where T is the temperature of annealing process, $f(i)$ and $f(j)$ are the values of the cost function of the optimization problem. $f(i)$ and $f(j)$ can be compared to the energy of tstates i and j, respectively, in an annealing solid. Obviously, the SA algorithm does not greedily reject all the suboptimal solutions, and the optimal state can eventually be reached.

2.5 The SA-Q-Learning Algorithm

In order to avoid blind exploration and repeatedly learning after finding optimal solution, the Metropolis criterion was introduced into Q-learning algorithm to select action, and the SA-Q-learning algorithm was proposed.

Let P be a policy space, and $p \in P$ be a policy, the value function $V(p)$ of the policy p is the sum of all the Q-values obtained for this policy.

$$V(p) = V\left((a_1, a_2, \cdots, a_n)\right) = \sum_{k=1}^{n} Q^p(s_k, a_k) \tag{8}$$

If the policy $p_1=(a_1,\ldots,a_{i-1},a_i,a_{i+1},\ldots,a_n)$ transits to the policy $p_2=(a_1,\ldots,a_{i-1},b_i,a_{i+1},\ldots,a_n)$, then the difference between their values is

$$V(p_2) - V(p_1) = Q(s_i, b_i) - Q(s_i, a_i) \ . \tag{9}$$

The policy p_1 is considered superior to p_2 if $V(p_1)>V(p_2)$. The value of a policy is compared to the energy of the microcosmic state in solid annealing.

The detailed of SA-Q-learning algorithm is as follows:

1. Initiate arbitrarily all $Q(s,a)$ values;

2. Repeat (for each episode);

(a) Choose a random (initial) state s;

(b) Repeat (for each step in the episode);

i. Select an action a_r in A(s) arbitrarily;

ii. Select an action a_p in A(s) according to the ε-greedy strategy;

iii. Let $a \leftarrow a_p$;

iv. Generate a random value $\xi \in (0,1)$;

v. If $\xi < \exp((Q(s,a_r) - Q(s,a_p))/Temperature)$, then let $a \leftarrow a_r$;

vi. Execute the action a, receive immediate reward r, then observe the new state s';

vii. Update $Q(s,a)$ by (6);

viii. Let $s \leftarrow s'$.

Until s is one of the goal states.

(c) Recalculate temperature by the temperature-dropping criterion.

Until the desired number of episodes has been researched.

Although the temperature-dropping criterion can be in general arbitrary, we use the geometric scaling factor criterion to reduce temperature according to the following

$$T_{k+1} = \lambda \cdot T_k \ . \tag{10}$$

The cooling rate $\lambda<1$ decides the decreasing velocity of the temperature. Usually λ is a constant factor close to 1, in order to guarantee a slow decay of the temperature factor in the algorithm.

In SA-Q-learning algorithm, no more than one action of the policy is modified in steps 2(b)i–2(b)vi, and this corresponds to the transition from one policy to another. Therefore, the Q-learning algorithm can be regarded as the combinatorial optimization in the policy space. Hence, the introduction of the Metropolis criterion is rational.

Comparing the ε-greedy Q-learning algorithm, there are only two additional steps in SA-Q-learning algorithm: the randomized selection of action and the evaluation of the Metropolis criterion. Therefore, there is no substantial increase of complexity between them, as the two steps take the constant time to evaluate. However, SA-Q-learning algorithm eliminates the disadvantage of the probability ε remaining constant, and the exploration will gradually be reduced with the dropping temperature.

3. SIMULATION AND COMPARISON

We assume that all packets are with a fixed length of 1000bytes, and adopt a finite buffer length of 20packets in the node. On the other hand, the offered loading of the simulation varies between 0.6 and 1.2 corresponding to the systems' dynamics; therefore, higher loading results in heavier traffic and vice versa.

In the simulation, four schemes of flow control agent, AIMD, Q-learning flow controller (QFC) with general (0-greedy) Q-learning and 0.1-greedy Q-learning algorithm, SAQFC proposed are implemented individually in high-speed network. The first scheme AIMD increases its sending rate by a fixed increment (0.11) if the queue length is less than the predefined threshold; otherwise the sending rate is decreased by a multiple of 0.8 of the previous sending rate to avoid congestion. Finally, for the other schemes, the sending rate is controlled by the feedback control signal periodically. For assuring SAQFC proposed applied to high-speed networks to be achievable and feasible, comparisons among those schemes are analyzed. Four measures, throughput, PLR, buffer utilization, and packets' mean delay, are used as the performance indices. The throughput is the amount of received packets at specified nodes (switches) without retransmission. The status of the input multiplexer's buffer in node reflects the degree of congestion resulting in possible packet losses. For simplicity, packets' mean delay only takes into consideration the processing time at node plus the time needed to transmit packets. The details are delineated in the following.

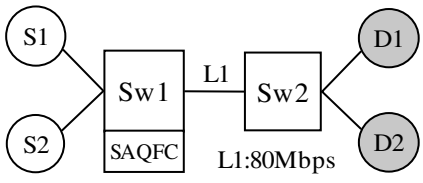

Fig. 3. The simulation model of network with two switches

The simulation model of high-speed network, as shown in Fig.3, is composed of two switches, Sw1 with a control agent and Sw2 with no controller are cascaded. In addition, two sources, S1 and S2, are connected to Sw1, and two destinations, D1 and D2, are connected to Sw2. The constant output link L1 is 80Mbps. The sending rates of the sources S1 and S2 are regulated by SAQFC. Training input epochs are generated by a shuffle of loading pattern (0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2), each of which lasts for 0.6s; i.e., a training epoch will last for a period of 4.2s.

As shown in Fig.4, the PLR is decrease as the training epochs go on. After ten training epochs, the PLR for an epoch is less than 2.5×10^{-5}. It is worth mentioning that no packet loss occurs in offered loading less than 1.0. In order to explore the reliability and robustness of the controlled structure, an extra loading 2.0 is incorporated as a disturbance at the 20th training epoch. As shown in Fig.4, the disturbance deteriorates the PLR temporarily, but SAQFC can adaptively learn to remain at low PLR when the disturbance is removed.

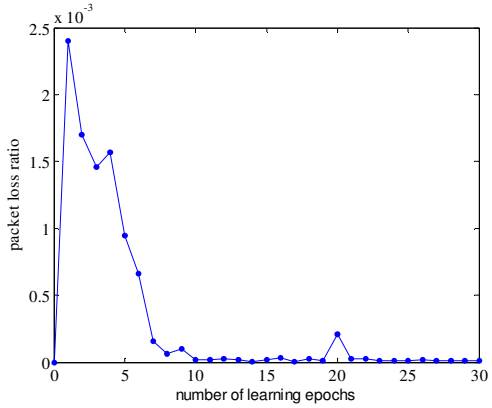

Fig. 4. PLR of SAQFC proposed, an extra loading 2.0 is incorporated as a disturbance at the 20th training epoch

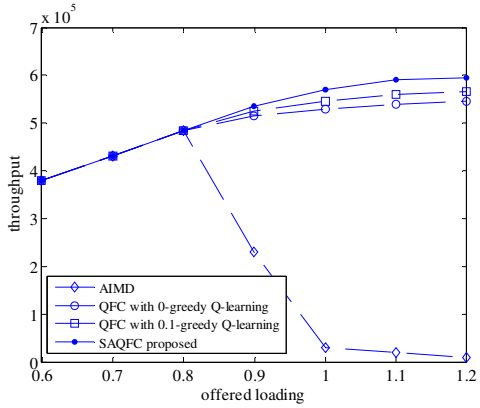

Fig. 5. Throughput versus various offered loading

The performance comparison of throughput, PLR, buffer utilization, and mean delay controlled by four different kinds of agents individually are shown in Fig.5-8. Analogously, because of the reactive control, the throughput for the AIMD method decrease seriously at loading of 0.9. Conversely, the SAQFC methods remain a higher throughput even though the offered loading is over 1.0. It is obvious that PLR of no control is high, even though we adopt the AIMD method. However, the SAQFC proposed can decrease the PLR enormously with high throughput and low mean delay. The SAQFC proposed has a better performance over QFC with 0-greedy and 0.1-greedy Q-learning in PLR, buffer utilization, and mean delay. It demonstrates once again that SAQFC possesses the ability to predict the network behavior in advance.

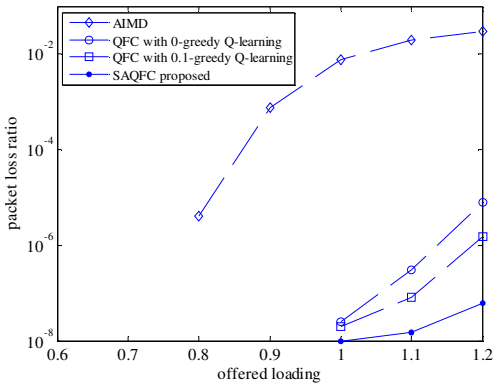

Fig. 6. PLR versus various offered loading

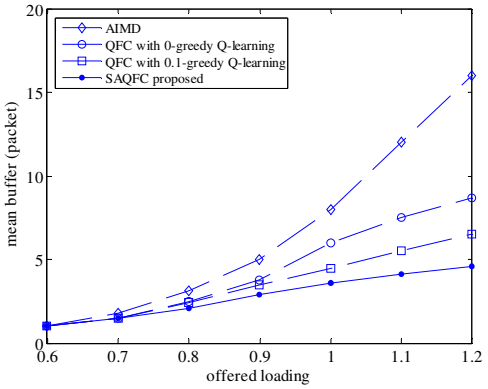

Fig. 7. Mean buffer versus various offered loading

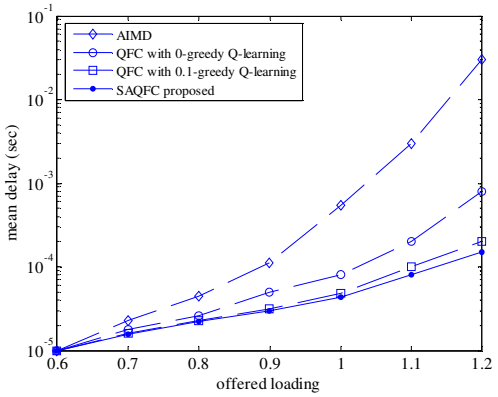

Fig. 8. Mean delay versus various offered loading

4. CONCLUSION

In high-speed networks, most packet losses result from the dropping of packets owing to congested nodes. The reactive scheme AIMD could not accurately respond to a time-varying environment due to the lack of prediction capability. In contrast, the Q-learning algorithm of reinforcement learning can cope with the prediction problems and the Metropolis criterion of simulated annealing can cope with the balance between exploration and exploitation in Q-learning. The proposed SAQFC, based on the Metropolis criterion based Q-learning algorithm, can respond to the networks' dynamics. Through a proper training process, SAQFC can learn empirically without prior information on the environmental dynamics. The sending rate of traffic sources can be determined by the well-trained optimal Q-values. Simulation results have shown that the proposed method can increase the utilization of the buffer and decrease the PLR simultaneously. Therefore, the SAQFC proposed not only guarantees low PLR for the existing links, but also achieves high system utilization.

ACKONWLEDGMENT

This work is supported by the National Natural Science Foundation of China, under grant 60274009, and Specialized Research Fund for the Doctoral Program of Higher Education, under grant 20020145007.

REFERENCES

Chatovich, A., S. Okug and G. Dundar (2001). Hierarchical neuro-fuzzy call admission controller for ATM networks. *Computer communications*, **24** (11), 1031-1044.

Cheng, R. G., C. J. Chang and L. F. Lin (1999). A QoS-provisioning neural fuzzy connection admission controller for multimedia high-speed networks. *IEEE/ACM Transactions on Networking*, **7** (1), 111-121.

Gevros, P., J. Crowcoft, P. Kirstein and S. Bhatti (2001). Congestion control mechanisms and the best effort service model. *IEEE Network*, **15** (3), 16-26.

Guo, M. Z., Y. Liu and J. Malec (2004). A new Q-learning algorithm based on the metropolis criterion. *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, **34** (5), 2140-2143.

Hsiao, M. C., K. S. Hwang, S. W. Tan and C. S. Wu (2003). Reinforcement learning congestion controller for multimedia surveillance system. In: P*roc. of the 2003 IEEE International Conf. on Robotics and Automation*, Taipei, Taiwan. 4403-4407.

Hsiao, M. C., S. W. Tan, K. S. Hwang and C. S. Wu (2005). A reinforcement learning approach to congestion control of high-speed multimedia networks. *Cybernetics and Systems*, **36** (2), 181-202.

Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, **4** (1), 237-285.

Lestas, M., A. Pitsillides, P. Ioannou and G. Hadjipollas (2007). Adaptive congestion protocol: a congestion control protocol with learning capability. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, **51** (13), 3773-3798.

Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Journal of Cognitive System Research*, **2** (1), 55-66.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21** (6), 1087-1092.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: an Introduction*. MIT. Cambridge, MA. USA.

Watkins, C. J. C. H. and P. Dayan (1992). Q-learning. *Machine Learning*, **8** (3), 279-292.