2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

A Q-Learning Model-Independent Flow Controller for High-speed Networks

Xin Li, Georgi M. Dimirovski, *Senior Member*, *IEEE*, Yuanwei Jing, and Siying Zhang

***Abstract*—For the congestion problems in high-speed networks, a Q-learning model-independent 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. In this case, the Q-learning, which is independent of mathematic model and prior-knowledge, has good performance. In this paper, the flow with higher priority in the network is considered. The competition of the flows with different priorities is regarded as a two-player game. Through learning process, the proposed controller can achieve the optimal sending rate for the sources with lower priority while the sources with higher priority existing. Simulation results show that the proposed controller can learn to regulate source flow with the features of high throughput and low packet loss ratio, and can avoid the occurrence of congestion effectively.**

I. 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 to be 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 high-speed networks [1, 2]. But the priori-knowledge of network to train the parameters in the controller is hard to achieve for high-speed networks.

In this case, the reinforcement learning (RL) shows its particular superiority, which just needs very simple information such as estimable and critical information, "right" or "wrong" [3]. 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. Because of the advantages above, RL has been played a very important role in the flow control in high-speed networks [4-7]. The Q-learning algorithm of RL is easy for application and has a firm foundation in the theory [8]. In [9], we combined the Q-learning and simulated annealing to solve the problems of ABR flow control in ATM networks.

In order to satisfy various classes of flow with different QoS requirements, high-speed networks must support different service categories, for example the real-time applications and nonreal-time applications. Among them, the real-time applications have higher priority then the nonreal-time applications. Link bandwidth is first allocated to the flow with higher priority and the remaining bandwidth is given to the flow with lower priority we need to control. We can control the sending rates of the flow by feedback mechanism. In this paper, we consider the competition of the flows with different priorities as a two-player game for controller optimization.

In this paper, a Q-learning model-independent flow controller for high-speed networks is proposed. The proposed controller can behave optimally without the mathematic model of the network environment, only relying on the interaction with the unknown environment and provide the best action for a given state. 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 controller can avoid the occurrence of congestion effectively with the features of high throughput, low PLR, low end-to-end delay, and high utilization.

II. Design of Controller

A. Architecture of Flow Controller Proposed

This section gives the detailed architecture of the proposed flow controller as shown in Fig.1.

In high-speed networks, the proposed controller in bottleneck node acts as a flow control agent with flow control ability. The inputs of controller are state variable in

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 Grant20020145007.

Xin Li, Yuanwei Jing, and Siying Zhang are with Faculty of Information Science and Engineering, Northeastern University, Shenyang, Liaoning, 110004, P.R. of China (e-mail: lixin820106@126.com).

Georgi M. Dimirovski is with Faculty of Engineering, Computer Engg. Dept, Dogus University of Istanbul, TR-347222 Istanbul, Rep. of Turkey (e-mail: gdimirovski@dogus.edu.tr).

high-speed networks composed of the current queue length in the buffer and the available bandwidth for the controlled traffic sources. The output of controller is the feedback signal to the traffic sources we need to control, which is the determined source sending rate. The learning agent and the network environment interact continually in the learning process.

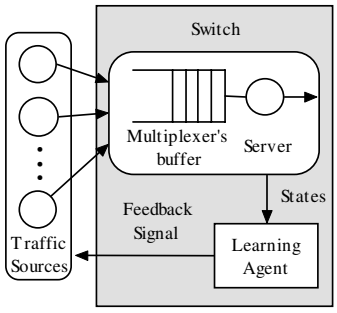

Fig. 1. Architecture of the proposed flow controller

B. Theoretical Framework

In this section, we consider the following discrete time system as the model of the high-speed networks

$$x_{k+1} = f(x_k, u_k, v_k) = Ax_k + Bu_k + Cv_k \ , \tag{1}$$

with feedback control

$$u_k = Lx_k \ , \tag{2}$$

and

$$v_k = Kx_k \ , \tag{3}$$

where x_k is the state of the high-speed networks, composed of the queue length in the buffer and the available bandwidth for the controlled traffic sources; u_k and v_k are the control inputs of the controller, u_k means the controlled sending rate of traffic sources with lower priority; v_k means the sending rate of traffic sources with higher priority. L and K are the control policies for u_k and v_k respectively. Here A, B, L, and K are the matrices of dimensions 2×2, 2×1, 1×2, and 1×2 respectively. L and K are chosen so that the matrix $A+BL+CK$ has all of its eigenvalues strictly within the unit circle.

From (1), we can see that if A,B,C in high-speed networks is known, we can design the flow controllers adopting many model-dependent control method. But because of the difficulty to devise a mathematic model for high-speed networks, the value of A,B,C is hard to achieve or hard to achieve accurately. So, we need model-independent control method to deal with the above-mentioned difficulties. Q-learning is a model-independent control method which has good performance.

Associated with the network system we define a one step cost as

$$c_k = c(x_k, u_k, v_k) = x_k^T R x_k + u_k^T E u_k - v_k^T F v_k \ . \tag{4}$$

where R is a symmetric positive semidefinite matrix of dimensions 2×2.

The total cost of a state x_k under the control policy (L,K), $V(x_k)$, is defined as the discounted sum of all costs that will be incurred by using (L,K) from time k onward, i.e.,

$$V(x_k) = \sum_{i=0}^{\infty} \beta^i c_{k+i} \ . \tag{5}$$

where $0 \le \beta \le 1$ is the discount factor.

In Q-learning, the learning agent in the flow controller tries to optimize the function $V(x_k)$ when the sources with lower priority taking the sending rate u_k while the sources with higher priority taking the sending rate v_k at state x_k. The definition of $V(x_k)$ implies the recurrence relation

$$V(x_k) = c(x_k, Lx_k, Kx_k) + \beta V(x_{k+1}) \ . \tag{6}$$

$V(x_k)$ is a quadratic function in the state and therefore can be expressed as

$$V(x_k) = x_k^T P x_k \ , \tag{7}$$

where P is the 2×2 cost matrix for policy (L,K). (L^*,K^*) denotes the policy which is optimal in the sense that the total discounted cost of every state is minimized. P^* represents the cost matrix associated with (L^*,K^*).

It is a simple matter to derive (L^*,K^*) if accurate models of the network and cost function are available. The problem we address is how to derive (L^*,K^*) without access to such models.

Watkins defined the Q-function for a stable control policy (L,K) as

$$Q(x,u,v) = c(x,u,v) + \beta V(f(x,u,v)) \ . \tag{8}$$

The value of $Q(x,u,v)$ is the sum of the one step cost incurred by taking action (u,v) from state x, plus the total cost that would accrue if the fixed policy (L,K) were followed from the state $f(x,u,v)$ and all subsequent states. The function Q can also be defined recursively as

$$Q(x_k, u_k, v_k) = c(x_k, u_k, v_k) + \beta Q(x_{k+1}, Lx_{k+1}, Kx_{k+1}) \ , \tag{9}$$

by noting that

$$Q(x, Lx, Kx) = c(x, Lx, Kx) + \beta V(f(x, Lx, Kx)) = V(x) \ . \tag{10}$$

The Q-function can be computed explicitly by

$$\begin{aligned}
&Q(x,u,v) \\
&= c(x,u,v) + \beta V(f(x,u,v)) \\
&= x^T R x + u^T E u - v^T F v + \beta (Ax + Bu + Cv)^T P (Ax + Bu + Cv) \\
&= x^T (\beta A^T P A + R) x + u^T (\beta B^T P B + E) u + \\
&\quad v^T (\beta C^T P C - F) v + \beta x^T A^T P B u + \beta x^T A^T P C v + \\
&\quad \beta u^T B^T P A x + \beta u^T B^T P C v + \beta v^T C^T P A x + \beta v^T C^T P B u \\
&= \begin{bmatrix} x \\ u \\ v \end{bmatrix}^T \begin{bmatrix} \beta A^T P A + R & \beta A^T P B & \beta A^T P C \\ \beta B^T P A & \beta B^T P B + E & \beta B^T P C \\ \beta C^T P A & \beta C^T P B & \beta C^T P C - F \end{bmatrix} \begin{bmatrix} x \\ u \\ v \end{bmatrix} \\
&= \begin{bmatrix} x \\ u \\ v \end{bmatrix}^T \begin{bmatrix} H_{(11)} & H_{(12)} & H_{(13)} \\ H_{(21)} & H_{(22)} & H_{(23)} \\ H_{(31)} & H_{(32)} & H_{(33)} \end{bmatrix} \begin{bmatrix} x \\ u \\ v \end{bmatrix} \\
&= \begin{bmatrix} x \\ u \\ v \end{bmatrix}^T H \begin{bmatrix} x \\ u \\ v \end{bmatrix} = y^T H y \ .
\end{aligned} \tag{11}$$

where $y = [x^T, u^T, v^T]^T$ is the column vector concatenation of

x, u, and v. H is a symmetric positive definite matrix of dimensions $(2+1+1)\times(2+1+1)$.

Given the policy (L_j, K_j) and the value function V, we can find an improved policy, (L_{j+1}, K_{j+1}), by minimized Q. So we can find the minimizing u and v by taking the partial derivative of $Q(x,u,v)$ with respect to u and v respectively. Taking the derivative we get

$$\begin{cases} \dfrac{\partial Q(x,u,v)}{\partial u} = 2\left(\beta B^T PB + E\right)u + 2\beta B^T PCv + 2\beta B^T PAx \\ \dfrac{\partial Q(x,u,v)}{\partial v} = 2\beta C^T PBu + 2\left(\beta C^T PC - F\right)v + 2\beta C^T PAx \end{cases}. \quad (12)$$

Setting that to zero and solving for u and v yields

$$\begin{aligned} u &= \left(\beta B^T P_j B + E - \beta^2 B^T P_j C\left(\beta C^T P_j C - F\right)^{-1} C^T P_j B\right)^{-1} \times \\ &\quad \left(\beta^2 B^T P_j C\left(\beta C^T P_j C - F\right)^{-1} C^T P_j A - \beta B^T P_j A\right)x \quad (13) \\ &= L_{j+1}x, \end{aligned}$$

and

$$\begin{aligned} v &= \left(\beta C^T P_j C - F - \beta^2 C^T P_j B\left(\beta B^T P_j B + E\right)^{-1} B^T P_j C\right)^{-1} \times \\ &\quad \left(\beta^2 C^T P_j B\left(\beta B^T P_j B + E\right)^{-1} B^T P_j A - \beta C^T P_j A\right)x \quad (14) \\ &= K_{j+1}x. \end{aligned}$$

Since the new policy (L_{j+1}, K_{j+1}) does not depend on x, it is the minimizing policy for all x. Using (11), (L_{j+1}, K_{j+1}) can be written in the terms of H as

$$\begin{cases} L_{j+1} = \left(H_{j(22)} - H_{j(23)}H_{j(33)}^{-1}H_{j(32)}\right)^{-1} \times \\ \qquad \left(H_{j(23)}H_{j(33)}^{-1}H_{j(31)} - H_{j(21)}\right) \\ K_{j+1} = \left(H_{j(33)} - H_{j(32)}H_{j(22)}^{-1}H_{j(23)}\right)^{-1} \times \\ \qquad \left(H_{j(32)}H_{j(22)}^{-1}H_{j(21)} - H_{j(31)}\right) \end{cases}. \quad (15)$$

Note that (15) depends only on the H matrix, and they are needed to find the control gains. If H is known, then the network model is not needed to compute the controller gains [10].

C. Learning Process of the controller

In this section, we show how the function Q can be directly estimated using recursive least squares (RLS). It is not necessary to identify either the network model or the one-step cost function separately.

First, we define the "overbar" function for vectors so that $\overline{y}$ is the vector whose elements are all of the quadratic basis functions over the elements of y, i.e.,

$$\overline{y} = (y_1^2, \cdots, y_1 y_n, y_2^2, y_2 y_3, \cdots, y_{n-1}^2, y_{n-1} y_n, y_n^2). \quad (16)$$

where n is the sum of dimensions of x, u, and v.

Next, we define the vector function Θ for square matrices. $\Theta(H)$ is the vector whose elements are the n diagonal entries of H and the $n(n+1)/2 - n$ distinct sums $(H_{ij} + H_{ji})$.

The elements of $\overline{y}$ and $\Theta(H)$ are ordered so that

$$Q(x,u,v) = Q(y) = y^T Hy = \overline{y}\Theta(H). \quad (17)$$

Finally, we rearrange equation (9) to yield

$$\begin{aligned} & c(x_k, u_k, v_k) \\ &= Q(x_k, u_k, v_k) - \beta Q(x_{k+1}, Lx_{k+1}, Kx_{k+1}) \\ &= \left[x_k^T, u_k^T, v_k^T\right]^T H\left[x_k^T, u_k^T, v_k^T\right] - \\ &\quad \beta\left[x_{k+1}^T, Lx_{k+1}^T, Kx_{k+1}^T\right]^T H\left[x_{k+1}^T, Lx_{k+1}^T, Kx_{k+1}^T\right] \\ &= \overline{\left[x_k^T, u_k^T, v_k^T\right]}^T \Theta(H) - \beta\overline{\left[x_{k+1}^T, Lx_{k+1}^T, Kx_{k+1}^T\right]}^T \Theta(H) \\ &= \phi_k^T \theta, \end{aligned}$$

where

$$\phi_k = \overline{\left[x_k^T, u_k^T, v_k^T\right]} - \beta\overline{\left[x_{k+1}^T, Lx_{k+1}^T, Kx_{k+1}^T\right]}, \quad (18)$$

and

$$\theta = \Theta(H). \quad (19)$$

Recursive Least Squares (RLS) can now be used to estimate θ. The recurrence relations for RLS are given by

$$\hat{\theta}_j(i) = \hat{\theta}_j(i-1) + \frac{U_j(i-1)\phi_k\left(c_k - \phi_k^T\hat{\theta}_j(i-1)\right)}{1 + \phi_k^T U_j(i-1)\phi_k}, \quad (20)$$

$$U_j(i) = U_j(i-1) - \frac{U_j(i-1)\phi_k\phi_k^T U_j(i-1)}{1 + \phi_k^T U_j(i-1)\phi_k}, \quad (21)$$

$$U_j(0) = U_0. \quad (22)$$

where $U_0 = \alpha I$ for some large positive constant α. $\theta_j = \Theta(H_j)$ is the true parameter vector for the function Q_j. $\hat{\theta}_j(i)$ is the i^{th} estimate of θ_j. The subscript k and the index i are both incremented at each time step.

It is shown that this algorithm converges to the true parameters if θ_j is fixed and ϕ_k satisfies the persistent excitation condition

$$\varepsilon_0 I \le \frac{1}{N}\sum_{i=1}^{N}\phi_{k-i}\phi_{k-i}^T \le \overline{\varepsilon}_0 I \quad \text{for all} \quad k \ge N_0 \quad \text{and} \quad N \ge N_0, \quad (23)$$

where $\varepsilon_0 \le \overline{\varepsilon}_0$, and N_0 is a positive number.

The policy improvement starts with the network system in some initial state x_0 and with some stabilizing controller (L_0, K_0). j keeps track of the number of policy iteration steps. k keeps track of the total number of time steps. i counts the number of time steps since the last change of policy. When $i = N$, one policy improvement step is executed.

Each policy iteration step consists of two phases: estimation of the Q-function for the current controller, and policy improvement based on that estimate. Consider the j^{th} policy iteration step. (L_j, K_j) is the current controller. $\hat{\theta}_j = \hat{\theta}_j(N)$ is the estimate of θ_j at the end of the parameter estimation interval. Each estimation interval is N time-steps long. The algorithm is initialized at the start of the j^{th} estimation interval by setting $U_j(0) = U_0$ and initializing the parameter estimates for the j^{th} estimation interval to the final parameter estimates from the previous interval, i.e., $\hat{\theta}_j(0) = \hat{\theta}_{j-1}(N)$. The index i used in equation (20) and (21) counts the number of time steps since the beginning of the estimation interval. After identifying the parameters $\Theta(H_j)$ for N time-steps, one policy improvement step is taken based on the estimate $\hat{\theta}_j$. This produces the new controller (L_{j+1}, K_{j+1}), and a new policy iteration step is begun.

III. SIMULATION RESULTS

The simulation model of high-speed networks, as shown in Fig.2, is composed of two switches, Sw1 with a control agent and Sw2 with no controller are cascaded. The constant output link L is 80Mbps. The sending rates of the sources are regulated by the flow controllers individually.

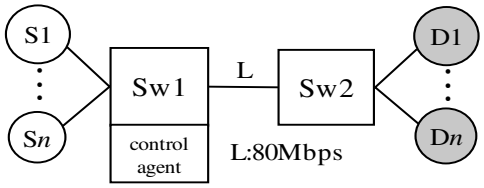

Fig. 2. The simulation model of network with two switches

In the simulation, we assume that all packets are with a fixed length of 1000bytes, and adopt a finite buffer length of 20packets in the node. The traffic flow with higher priority exists during the whole course of learning process. 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. For the link of 80Mbps, the theoretical throughput is 62.5K packets.

In the simulation, four schemes of flow control agent, AIMD, the model based PID flow controller, neural network (NN) flow controller with learning ability, and the proposed flow controller are implemented individually in high-speed networks. 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 [11]. Finally, for the other schemes, the sending rate is controlled by the flow controller.

For assuring controller 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 performance comparison of throughput, PLR, buffer utilization, and mean delay controlled by four different kinds of agents individually are shown in Fig.3-6. The throughput for AIMD method decrease seriously at loading of 0.9. Conversely, the controller proposed remains a higher throughput even though the offered loading is over 1.0. It is obvious that PLR is high, even though we adopt AIMD scheme. However, the controller proposed can decrease the PLR enormously with high throughput and low mean delay. The controller proposed has a better performance over PID flow controller and NN flow controller in PLR, buffer utilization, and mean delay. It demonstrates once again that controller proposed possesses the ability to predict the network behavior in advance.

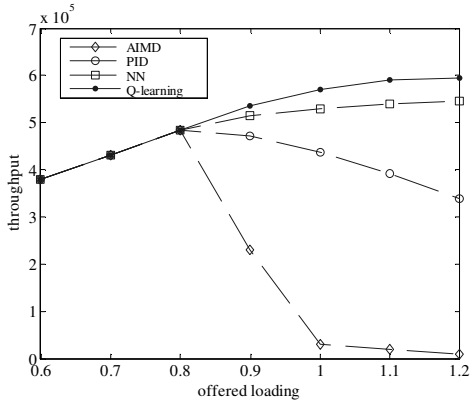

Fig. 3. Throughput versus various offered loading

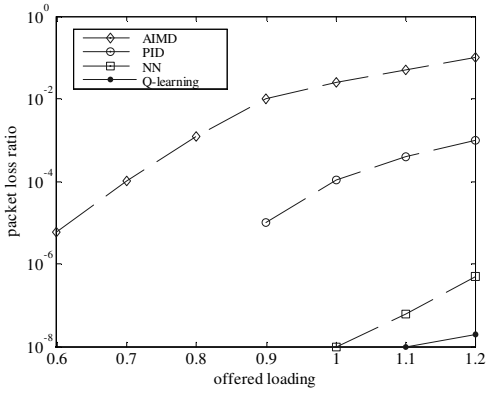

Fig. 4. PLR versus various offered loading

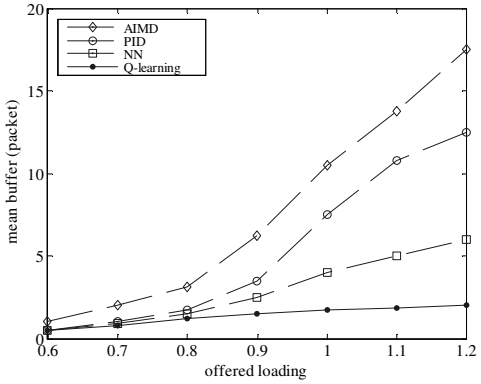

Fig. 5. Mean buffer versus various offered loading

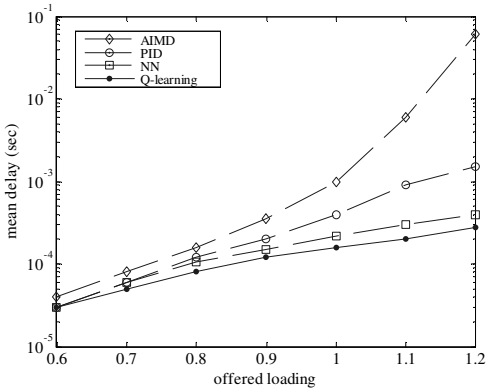

Fig. 6. Mean delay versus various offered loading

IV. CONCLUSION

In this paper, a Q-learning model-independent flow controller is proposed to deal with the congestion problems in high-speed networks. Because of the interaction with the environment, the proposed controller has good performance in the congestion control of high-speed networks considering the competition of the flows with different priorities. It is seen that the proposed controller indeed performs an efficiently flow control, and is capable of learning the system behavior. The simulation results show that the proposed controller is superior to the other flow controller in comparison with respect to the performance of high-speed networks.

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, pp. 111-121, 1999.

[2] M. Lestas, A. Pitsillides, P. Ioannou, and G. Hadjipollas, "Adaptive congestion protocol: a congestion control protocol with learning capability," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 51, no. 13. pp. 3773-3798, Sep. 2007.

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

[4] A. Chatovich, S. Okug, and G. Dundar, "Hierarchical neuro-fuzzy call admission controller for ATM networks," *Computer Communications*, vol. 24, no. 11, pp. 1031-1044, Jun. 2001.

[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, pp. 181-202, Jan. 2005.

[6] K. S. Hwang, S. W. Tan, M. C. Hsiao, and C. S. Wu, "Cooperative multiagent congestion control for high-speed networks," *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, vol. 35, no. 2, pp. 255-268, Apr. 2005.

[7] X. Li, X. J. Shen, Y. W. Jing, and S. Y. Zhang, "Simulated annealing-reinforcement learning algorithm for ABR traffic control of ATM networks," in *Proc. of the46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, Dec. 2007, pp. 5716-5721.

[8] C. J. C. H. Watkins, and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279-292, May 1992.

[9] X. Li, Y. C. Zhou, G. M. Dimirovski, and Y. W. Jing, "Simulated annealing Q-learning algorithm for ABR traffic control of ATM networks," in *Proc. of the 2008 American Control Conf.*, Seattle, Washington, USA, Jun. 2008.

[10] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," *Proceedings of the American control conference*, pp.3475-3479, 1994.

[11] P. Gevros, J. Crowcoft, P. Kirstein, and S. Bhatti, "Congestion control mechanisms and the best effort service model," *IEEE Network*, vol. 15, no. 3, pp. 16-26, May/Jun. 2001.