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

Nash Q-Learning Multi-Agent Flow Control for High-Speed Networks

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

***Abstract*—For the congestion problems in high-speed networks, a multi-agent flow controller (MFC) based on Q-learning algorithm conjunction with the theory of Nash equilibrium is proposed. Because of the uncertainties and highly time-varying, it is not easy to accurately obtain the complete information for high-speed networks, especially for the multi-bottleneck case. The Nash Q-learning algorithm, which is independent of mathematic model, shows the particular superiority in high-speed networks. It obtains the Nash Q-values through trial-and-error and interaction with the network environment to improve its behavior policy. By means of learning procedures, MFCs can learn to take the best actions 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.**

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 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 to be 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 case, the reinforcement learning shows its particular superiority, which just needs very simple information such as estimable and critical information, "right" or "wrong" [3]. It 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, it has been played a very important role in the flow control in high-speed networks [4-7]. The Q-learning of reinforcement learning is easy for application and has a firm foundation in the theory. In [8], we combined the Q-learning and simulated annealing to solve the problems of ABR flow control in ATM networks. In [9], authors extended the environment to multi-bottleneck network and propose a cooperative multi-agent congestion controller. But, the convergence of the controller was not demonstrated.

In this paper, based on the Q-learning algorithm, a multi-agent flow controller (MFC) for high-speed networks is proposed. The fuzzy inference system is adopted to generate the reward signal in Q-learning. In the controller proposed, each learning agent in bottleneck node 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. 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. The convergence of the proposed learning algorithm is demonstrated and some 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.

II. Design of Controller

A. Architecture of MFC

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 [10]. However, it is hard to achieve high system performance by reactive AIMD scheme because of the propagation delay and the dynamic nature of

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.

Yuanwei Jing, Xin Li, Yan Zheng, 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. Whereas the proposed MFC can behave optimally only rely on the interaction with unknown environment and provide the best action for a given state. The architecture of MFC in n node case is shown in Fig.1.

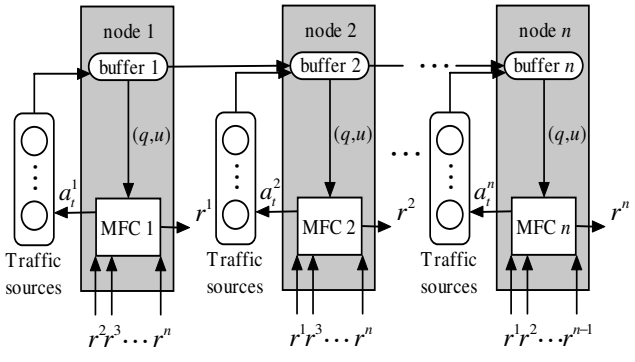

Fig.1. Architecture of the proposed MFC in n node case

The high-speed network has n bottleneck nodes, with controllable sources. Each MFC has its own state variables (S), which is composed of the current queue length q, the current rate of queue length change $\dot{q}$, and the current rate of source sending rate change $\dot{u}$ irrespective of other MFCs in the network. The output of each MFC is the feedback signal a to the traffic sources, which is the ratio of the sending rate. It determines the sending rate u of traffic sources. 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 rewards and actions taken previously. The sample for each MFC is same. By way of multi-agent strategy-search learning, the proposed MFC could converge to Nash equilibrium. In contrast to single-agent flow controller (SFC), MFC uses joint-action learning algorithms to learn values for joint actions to avoid the bias in individual action decision.

The sending rate is controlled by the feedback control signal a_t periodically. The controlled sending rate is defined by the equation

$$u_t = a_t FL\,, \tag{1}$$

where $a_t \in [0.2,1.0]$ is the feedback signal by the flow controller, F is a relative value in the ratio of source offered load to the available output bit rate, L denotes the outgoing rate of link, and $u_t \in [0.2 \cdot FL, FL]$ is the controlled sending rate at sample time t.

B. Reward Signal

Q-learning is to learn what to do and how to map situations to actions, so as to maximize the reward signal r. The reward 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. r is in the range $[0,1]$, the larger r is, the better control affects.

In high-speed networks, the fuzzy reward evaluator (FRE) [11] evaluates the reward for environmental states. FRE relies on three parameters $(q,\dot{q},\dot{u})$ to generate a reward or a punishment for the action (the ratio of the source sending rate) in a state. If the state of the network is enhanced toward positive evolution, the action will be rewarded; otherwise punished.

The term set should be determined at an approximate level of granularity to describe the values of linguistic variables. For queue length, the term set is defined as $f(q)$={Low (L), Medium (M), High (H)}, which is used to describe the degree of queue lengths. The term set for the rate of queue length change is defined as $f(\dot{q})$={Decrease (D), Increase (I)}, which describes the rate of queue length change as "Decrease" or "Increase". The term set for the rate of source transmission rate change is defined as $f(\dot{u})$={Negative (N), Positive (P)}, which describes the rate of source transmission rate change as "Negative" or "Positive". In order to provide a precise graded reward in various states, the term set for reward is defined as $f(y_r)$={Penalty More (PM), Penalty Slightly (PS), No Reward (NR), Reward Slightly (RS), Reward More (RM)}. The membership functions (MFs) of the term set are shown in Fig.2.

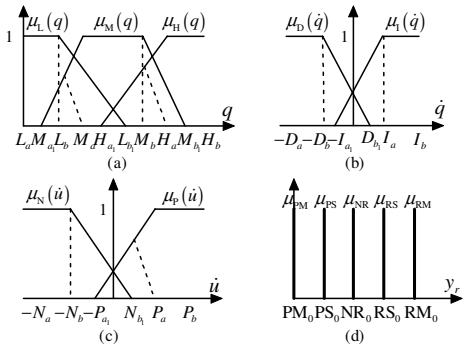

Fig. 2. MFs of the term set (a) $f(q)$, (b) $f(\dot{q})$, (c) $f(\dot{u})$, and (d) $f(y_r)$

The fuzzy rule base is a reward knowledge base, characterized by a set of linguistic statements in the form of "if-then" rules that describe the fuzzy logic relationship between the input variables and the reward (penalty) y_r. According to fuzzy set theory, the fuzzy rule base forms a fuzzy set with dimensions 3×2×2=12. Table I shows a total of twelve inference rules in the fuzzy rule base under various system states. For example, rule 1 can be linguistically started as "if the queue length is low, the queue length change rate is decreased, and the transmission change rate is negative, then give more penalty."

TABLE I
RULE TABLE OF FRE

Rule	q	$\dot{q}$	$\dot{u}$	y_r	Rule	q	$\dot{q}$	$\dot{u}$	y_r
1	L	D	N	PM	7	M	I	N	RS
2	L	D	P	PS	8	M	I	P	RM
3	L	I	N	NR	9	H	D	N	PS
4	L	I	P	NR	10	H	D	P	PM
5	M	D	N	RS	11	H	I	N	PS
6	M	D	P	PM	12	H	I	P	PM

The proposed FRE adopts the max-min inference method for the inference engine because it is designed for real-time operation.

C. The Nash Q-learning Multi-Agent 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. The proposed flow controller in each bottleneck node acts as a learning agent. During the learning process, an agent uses its experience to improve its estimate by blending new information into its prior experience [13].

The difference between single-agent and multi-agent system exists in the environments. In multi-agent systems, other adapting agents make the environment no longer stationary and violate the Markov property that traditional single-agent behavior learning relies upon. Based on the framework of stochastic games, the single-agent Q-learning is extended to multi-agent systems.

In general form, an n-agent is defined by a tuple $<n,\mathrm{S},\mathrm{A}^1,\ldots,\mathrm{A}^n,r^1,\ldots,r^n,p>$, where n is the number of agents, in another word the number of bottleneck nodes in high-speed networks; S is a set of discrete state space of high-speed networks composed of $(q,\dot{q},\dot{u})$; $\mathrm{A}^1,\ldots,\mathrm{A}^n$ 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); r^i is the reward function for agent 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 π^i be the action strategy of agent i. For a given initial state s, agent i tries to maximize

$$v^i\left(s,\pi^1,\cdots,\pi^n\right)=\sum_{t=0}^{\infty}\beta^t E\left(r_t^i\middle|\pi^1,\cdots,\pi^n,s_0=s\right). \qquad (2)$$

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

The Nash equilibrium solution is a tuple of n strategies $(\pi_*^1,\cdots,\pi_*^n)$ such that for all $s\in\mathrm{S}$ and $\pi^i\in\Pi^i$,

$$v^i\left(s,\pi_*^1,\cdots,\pi_*^n\right)\geq v^i\left(s,\pi_*^1,\cdots,\pi_*^{i-1},\pi^i,\pi_*^{i+1},\cdots,\pi_*^n\right) \qquad (3)$$

where Π^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 ith agent is defined over $(s,a^1,\ldots,a^n)$ as the sum of its current reward plus its future rewards when all agents follow a joint Nash equilibrium strategy. That is,

$$Q_*^i\left(s,a^1,\cdots,a^n\right)=r^i\left(s,a^1,\cdots,a^n\right)+\beta\sum_{s'\in S}p\left(s'\middle|s,a^1,\cdots,a^n\right)v^i\left(s',\pi_*^1,\cdots,\pi_*^n\right) \qquad (4)$$

where $(\pi_*^1,\cdots,\pi_*^n)$ is the joint Nash equilibrium strategy, $r^i(s,a^1,\ldots,a^n)$ is agent i's one-period reward in state s under joint action $(a^1,\ldots,a^n)$, $v^i(s',\pi_*^1,\cdots,\pi_*^n)$ 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 $Q_0^i(s,a^1,\cdots,a^n)=0$ for all $s\in S$, $a^1\in A^1,\cdots,a^n\in A^n$. 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 $\pi^1(s')\ldots\pi^n(s')$ for $(Q_t^1(s'),\cdots,Q_t^n(s'))$, and updates its Q-values according to

$$Q_{t+1}^i\left(s,a^1,\cdots,a^n\right)=(1-\alpha_t)Q_t^i\left(s,a^1,\cdots,a^n\right)+\alpha_t\left[r_t^i+\beta NashQ_t^i\left(s'\right)\right] \qquad (5)$$

where $\beta\in[0,1)$ is the discount factor, if β 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 β 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 α. If α is small, the convergence rate will be slow but it will easily tend to stabilize. If α is large, the convergence rate will be fast but it will not easily tend to stabilize.

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

$$NashQ_t^i\left(s'\right)=\pi^1\left(s'\right)\cdots\pi^n\left(s'\right)Q_t^i\left(s'\right). \qquad (6)$$

In order to calculate the Nash equilibrium $(\pi^1(s')\ldots\pi^n(s'))$, agent i would need to know $Q_t^1(s'),\cdots,Q_t^n(s')$. 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, $Q_0^j(s,a^1,\cdots,a^n)=0$ 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 (5) it applies to its own,

$$Q_{t+1}^j\left(s,a^1,\cdots,a^n\right)=(1-\alpha_t)Q_t^j\left(s,a^1,\cdots,a^n\right)+\alpha_t\left[r_t^j+\beta NashQ_t^j\left(s'\right)\right]. \qquad (7)$$

Note that $\alpha_t=0$ for $(s,a^1,\cdots,a^n)\neq(s_t,a_t^1,\cdots,a_t^n)$. Therefore (7) 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.

D. Convergence of the Proposed Controller

In this section, we would like to demonstrate the convergence of Q_t^i to the Nash equilibrium Q_*^i for agent i. The value of Q_*^i is determined by the joint strategies of all agents. That means the agent has to learn Q-values of all the agents and derive strategies from them. The learning objective is $(Q_*^1,\cdots,Q_*^n)$, and we have to show the convergence of $(Q_t^1,\cdots,Q_t^n)$ to $(Q_*^1,\cdots,Q_*^n)$.

The convergence proof requires a basic assumption that every state and action should be visited infinitely often [14]. The proof relies on the following lemma (Corollary 5 in [15]), which establishes the convergence of a general Q-learning process updated by a pseudo-contraction operator. Let Q be the space of all Q functions.

Lemma. If the mapping P_t:Q→Q satisfies

$$\left\|P_tQ-P_tQ_*\right\|\leq\gamma\left\|Q-Q_*\right\|+\lambda_t \qquad (8)$$

for all $Q\in\mathrm{Q}$ and $Q_*=E[P_tQ_*]$, where $0<\gamma<1$ and $\lambda_t\geq 0$

converging to zero with probability 1, then the iteration defined by

$$Q_{t+1} = (1-\alpha_t)Q_t + \alpha_t [P_t Q_t] \quad (9)$$

converges to Q_* with probability 1.

If the Nash optimal Q-value Q_* matches the lemma with an existing operator P_t, then we can have the conclusion that the iteration converges to Nash optimal Q-value Q_* with probability 1.

For Nash Q-learning, we define the operator P_t as follows.

Let $Q=(Q^1,\ldots,Q^n)$, where $Q^k \in \mathrm{Q}^k$, for k=1,…,n, and $\mathrm{Q}=\mathrm{Q}^1\times\ldots\times\mathrm{Q}^n$. $P_t Q = (P_t Q^1,\cdots,P_t Q^n)$, where

$$P_t Q^k(s,a^1,\cdots,a^n) = r_t^k(s,a^1,\cdots,a^n) + \beta\pi^1(s')\cdots\pi^n(s')Q^k(s') \quad (10)$$

Let $v^k(s',\pi_*^1,\cdots\pi_*^n)$ is agent k's Nash equilibrium reward for $(Q_*^1(s')\cdots Q_*^n(s'))$, and $(\pi_*^1(s')\cdots\pi_*^n(s'))$ is the Nash equilibrium solution, we have

$$\begin{aligned}
&Q_*^k(s,a^1,\cdots,a^n)\\
&= r^k(s,a^1,\cdots,a^n) + \beta\sum_{s'\in S} p(s'|s,a^1,\cdots,a^n)v^k(s',\pi_*^1,\cdots\pi_*^n)\\
&= r^k(s,a^1,\cdots,a^n) + \beta\sum_{s'\in S} p(s'|s,a^1,\cdots,a^n)\pi_*^1(s')\cdots\pi_*^n(s')Q_*^k(s')\\
&= \sum_{s'\in S} p(s'|s,a^1,\cdots,a^n)\left(r^k(s,a^1,\cdots,a^n) + \beta\pi_*^1(s')\cdots\pi_*^n(s')Q_*^k(s')\right)\\
&= E\left[P_t^k Q_*^k(s,a^1,\cdots,a^n)\right]
\end{aligned}$$

for all $s,a^1,\ldots,a^n$. Thus $Q_*^k = E[P_t^k Q_*^k]$. Since this holds for all k, $E[P_t Q_*] = Q_*$.

During the learning process of MFC in high-speed networks, every stage game $(Q_t^1(s),\cdots,Q_t^n(s))$, for all t and s, holds Nash equilibrium, and agents' reward in this equilibrium are used to update their Q-functions. In the Nash equilibrium, each agent effectively holds a correct expectation about other agents' behaviors, and acts rationally with respect to this expectation. Acting rationally means the agent's strategy is a best response to the others' strategies. Any deviation would make that agent worse off. That is,

$$\pi^k(s)\pi^{-k}(s)Q^j(s) \ge \hat{\pi}^k(s)\pi^{-k}(s)Q^j(s), \quad (11)$$

$$\pi^k(s)\pi^{-k}(s)Q^j(s) \le \pi^k(s)\hat{\pi}^{-k}(s)Q^j(s), \quad (12)$$

for all k, and all $\hat{\pi}^k \in \pi(\mathrm{A}^k)$, $\hat{\pi}^{-k} \in \pi(\mathrm{A}^{-k})$. Where

$$\pi^{-k}(s) = \pi^1(s)\cdots\pi^{k-1}(s)\pi^{k+1}(s)\cdots\pi^n(s). \quad (13)$$

For all $Q,\hat{Q} \in \mathrm{Q}$ we have

$$\begin{aligned}
&\left\|P_t Q - P_t\hat{Q}\right\|\\
&= \max_j \left\|P_t Q^j - P_t\hat{Q}^j\right\|_{(j)}\\
&= \max_j \max_s \left\|P_t Q^j(s) - P_t\hat{Q}^j(s)\right\|_{(j,s)}\\
&= \max_j \max_s \left|\beta\pi^1(s)\cdots\pi^n(s)Q^k(s) - \beta\hat{\pi}^1(s)\cdots\hat{\pi}^n(s)\hat{Q}^k(s)\right|\\
&= \max_j \max_s \beta\left|\pi^k(s)\pi^{-k}(s)Q^k(s) - \hat{\pi}^k(s)\hat{\pi}^{-k}(s)\hat{Q}^k(s)\right| \quad (14)
\end{aligned}$$

Based on the property of Nash equilibrium [16], if $\pi^k(s)\pi^{-k}(s)Q^j(s) \ge \hat{\pi}^k(s)\hat{\pi}^{-k}(s)\hat{Q}^j(s)$, we have

$$\begin{aligned}
&\pi^k(s)\pi^{-k}(s)Q^k(s) - \hat{\pi}^k(s)\hat{\pi}^{-k}(s)\hat{Q}^k(s)\\
&\le \pi^k(s)\pi^{-k}(s)Q^k(s) - \pi^k(s)\hat{\pi}^{-k}(s)\hat{Q}^k(s)\\
&\le \pi^k(s)\hat{\pi}^{-k}(s)Q^k(s) - \pi^k(s)\hat{\pi}^{-k}(s)\hat{Q}^k(s)\\
&\le \left\|Q^k(s) - \hat{Q}^k(s)\right\|
\end{aligned}$$

where

$$\left\|Q^k(s) - \hat{Q}^k(s)\right\| = \max_{a^1,\cdots,a^n}\left|Q^k(s,a^1,\cdots,a^n) - \hat{Q}^k(s,a^1,\cdots,a^n)\right| \quad (15)$$

If $\pi^k(s)\pi^{-k}(s)Q^j(s) \le \hat{\pi}^k(s)\hat{\pi}^{-k}(s)\hat{Q}^j(s)$, proof is similar.

So we have

$$\left|\pi^k(s)\pi^{-k}(s)Q^k(s) - \hat{\pi}^k(s)\hat{\pi}^{-k}(s)\hat{Q}^k(s)\right| \le \left\|Q^k(s) - \hat{Q}^k(s)\right\| \quad (16)$$

Combining (14) and (16), we have

$$\begin{aligned}
&\left\|P_t Q - P_t\hat{Q}\right\|\\
&\le \max_j \max_s \beta\left\|Q^k(s) - \hat{Q}^k(s)\right\|\\
&= \beta\left\|Q - \hat{Q}\right\|
\end{aligned}$$

Because $\hat{Q} \in \mathrm{Q}$, so the Nash optimal Q-value Q_* satisfies (8).

Above all, we have the result that the proposed Nash Q-learning process converges to Nash Q-values.

III. Simulation and Comparison

We assume that all packets are with a fixed length of 1000bytes, and adopt a finite buffer length of 20packets in each 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.

From the knowledge of evaluating system performance, the parameters of the membership functions for input linguistic variables in FRE are selected as follows. For $\mu_L(q)$, $\mu_M(q)$, and $\mu_H(q)$, $L_a = 0$, $L_b = 6$, $L_{b_1} = 10$, $M_{a_1} = 2$, $M_a = 8$, $M_b = 12$, $M_{b_1} = 20$, $H_{a_1} = 9$, $H_a = 14$, $H_b = 20$, and $H_{b_1} = 20$; for $\mu_D(\dot{q})$ and $\mu_I(\dot{q})$, $D_a = 4$, $D_b = D_{b_1} = 2$, $I_{a_1} = I_a = 2$, and $I_b = 4$; for $\mu_N(\dot{u})$ and $\mu_P(\dot{u})$, $N_a = 0.8$, $N_b = 0.4$, $N_{b_1} = 0.2$, $P_{a_1} = 0.2$, $P_a = 0.4$, and $P_b = 0.8$. Also, the parameters of the membership functions for output reward linguistic variables are given by $PM_0 = 0$, $PS_0 = 0.25$, $NR_0 = 0.5$, $RS_0 = 0.75$, and $RM_0 = 1$.

In the simulation, three schemes of flow control agent, AIMD, MFC with general Q-learning algorithm and Nash Q-learning algorithm proposed 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 rates are decreased by a multiple of 0.8 of the previous sending rate to avoid congestion. Finally, for the second and third schemes, the sending rates are controlled by the feedback control signal periodically. For assuring MFCs 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 status of the input multiplexer's buffer in each node reflects the degree of congestion resulting in possible packet losses. For

simplicity, packets' mean delay only takes into consideration the processing time at each node plus time needed to transmit packets. The details are delineated in the following.

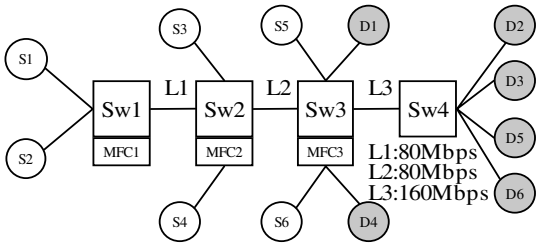

Fig.3. The configuration of high-speed networks with four switches

The configuration of high-speed networks with multi-node, as shown in Fig.3, is composed of four switches Sw1, Sw2, Sw3, and Sw4 in cascade. Three nodes Sw1, Sw2, and Sw3 are implemented with its own control agent, respectively; hence, the sending rates of sources S1 and S2, S3 and S4, and S5 and S6 are regulated by MFC1, MFC2 and MFC3, respectively. Consequently, three agents jointly interact to reach a common goal with high system performance. For case of three schemes, each node has the same training loading pattern, which is 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.

We take Sw2 as an example to consider the performance of the flow control schemes adopted in simulation. Fig.5 shows the throughputs of Sw2 controlled by three different kinds of control agents individually. Analogously, because of the reactive control, the throughput of nodes for the AIMD method decrease seriously at loading of about 0.9. Conversely, the MFC methods remain a higher throughput even though the offered loading is over 1.0. Fig.4-7 shows the PLR, buffer utilization and mean delay of Sw2 controlled by four different kinds of control agents individually. It is obvious that the PLR of no control is high, even though we adopt the AIMD method. However, using the MFC with Nash Q-learning method can decrease the PLR enormously with high throughput and low mean delay. The MFC with Nash Q-learning algorithm has a better performance over MFC with general Q-learning in PLR, buffer utilization and mean delay. It demonstrates once again that MFC with Nash Q-learning possesses the ability to predict the network behavior in advance.

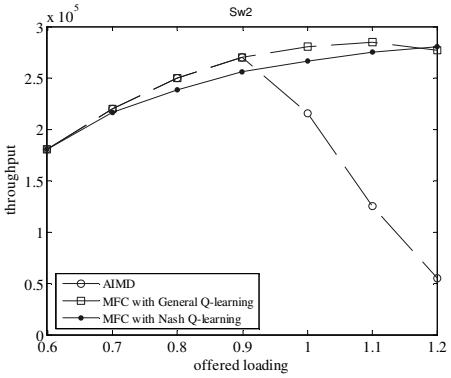

Fig.4. Throughput versus various offered loading at Sw2

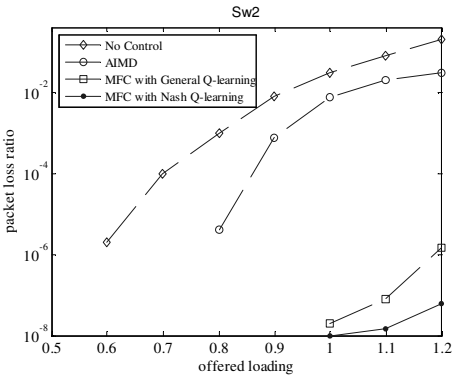

Fig.5. PLR versus various offered loading at Sw2

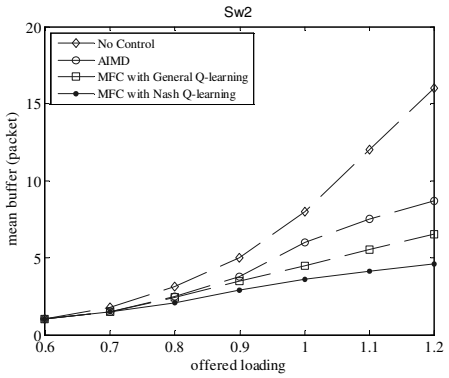

Fig.6. Mean buffer versus various offered loading at Sw2

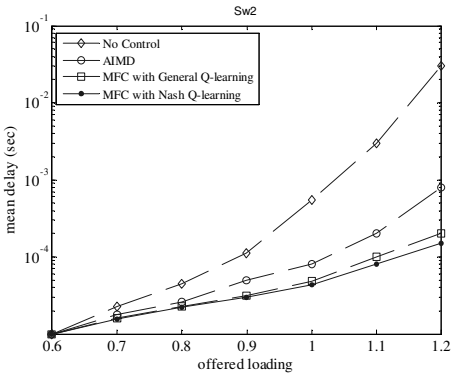

Fig.7. Mean delay versus various offered loading at Sw2

IV. 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 Nash Q-learning algorithm of reinforcement learning can cope with the prediction problems. The proposed MFC, which is applicable to a network of multi-bottleneck, can respond to the networks' dynamics. Through a proper training process, MFC 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 and the convergence is demonstrated. Simulation results have shown that the proposed method can increase the utilization of the buffer and decrease PLR and delay

simultaneously. Therefore, MFC with Nash Q-learning algorithm not only guarantees low PLR for the existing links, but also achieves high system utilization.

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] 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.

[9] 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.

[10] 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.

[11] M. L. Littman, "Value-function reinforcement learning in Markov games," *Journal of Cognitive System Research*, vol. 2, no. 1, pp. 55-66, 2001.

[12] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 237-285, 1996.

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

[14] J. Hu, and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, pp. 1039-1069, Nov. 2003.

[15] C. Szepesvari, and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Computation*, vol. 11, no. 8, pp. 2017-2060, Nov. 1999.

[16] J. F. Nash, "Non-cooperative games,", *Annals of Mathematics*, vol. 54, no. 2, pp. 286-295, Sep. 1951.