An Adaptive PID Controller for AQM with ECN Marks Based on Neural Networks

Chuan Zhou, Lu Zhang, Qingwei Chen

Abstract—**Nowadays Congestion control problem of the intermediate nodes in the Internet has received extensively attention in networking and control community. In this paper, a novel adaptive PID (Proportional-Integral-Differential) controller based on neural networks for the problem of AQM with ECN marks is presented. Considering a previously developed nonlinear dynamic model of TCP/AQM system and the queue management mechanism of intermediate nodes, the parameters of AQM controller based on neural networks is tuned online by using gradient-descent algorithm, and the packet drop/mark probability is determined adaptively in time to avoid congestion, so that the quality of service (QoS) of network and the transient performance can be improve greatly especially when the network parameters are time-varying. Finally, the proposed algorithm is verified by using NS-2 simulator, and simulation results show that the integrated performance of this proposed controller is obviously superior to those of typical RED and PID controller especially on the queue stability and mean time delay. Furthermore, this AQM algorithm has simple structure and can be implemented easily.**

I. INTRODUCTION

CONGESTION control for the networks with TCP/IP has received extensively attention as the Internet continues to expand in size, diversity and reach. The early TCP flow control is one of the most important congestion control mechanism for IP networks. In order to ensure better quality of service, some end to end flow control strategies have been presented, such as Tahoe and Reno. The basic idea of these protocols is for a source to probe the network for spare capacity by linearly increasing its rate and exponentially reducing its rate when congestion is detected. These methods play the key role in the end system. Active queue management (AQM) is an effective congestion control mechanism at the intermediate nodes, which can keep the best-effort service with low delay. The main goal of active queue management is congestion avoidance. Comparing the drop tail mechanism at router, it can make a compromise between high throughput and low delay by dropping the packets purposely and randomly at the router before the buffer of queue becoming full. Random early detection [1] (RED) is one of the typical algorithms for active queue management, which attempts to drop packets with a certain probability that is a function of the average queue length. Although RED is widely used in networks, it is difficult to tune RED parameters for different network traffic conditions and even readily leads to TCP global synchronization and queue oscillation at router. In order to overcome the shortcomings of RED, many modified algorithm were presented, for example adaptive RED [2] (ARED), Stabilize RED (SRED) and so on. BLUE[4][9] is a self-configuring AQM mechanism which can adjust drop probability adaptively according to packets loss and link utilization. From the point of control system view, AQM can be regarded as a typical regulation problem in control theory. Since the objective of AQM is to keep the queue size at router nearly around the reference value. Traditional PI and PID controller were induced to the design problem of AQM, and the performance was improved, such as queue stability [8][11]. However, it is difficult to tune the controller's parameters, especially under the uncertain and time-varying network environment.

In this paper, an adaptive PID controller based on neural networks is presented for AQM with ECN mechanism. The AQM controller integrates the merits of both PID controller and neural networks, which introduces the gradient descend algorithm to tune the PID controller parameters. It can improve the performance and robustness of AQM control system. The rest of the paper is organized as follows. Section two gives the dynamical model of TCP/AQM system and its linearization. An adaptive PID controller for AQM based on neural networks is presented in section three. In section four, the implementation of AQM algorithm based on NEURAL NETWORKS with ECN is discussed briefly. Next, in section five we give the simulation results to verify the proposed method by using NS-2 simulator. Finally, a conclusion is given in section six.

II. TCP/AQM SYSTEM MODEL

A dynamic model of TCP behavior was developed by Misra and Hollot using fluid-flow and stochastic differential equation analysis [5][6]. The dynamics of TCP is described by the following nonlinear differential equations:

$$\frac{dW(t)}{dt} = \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t)) \qquad (1.a)$$

$$\frac{dq(t)}{dt} = \frac{W(t)}{R(t)}N(t) - C \qquad (1.b)$$

Manuscript received February 10, 2009. This work was supported in part by the Natural Science Foundation of Jiangsu province under Grant BK2007206, China.

Chuan Zhou is with the School of Automation, Nanjing University of Science and Technology, Nanjing, 210094 China (phone: 86-25-84315872; e-mail: njust_zc@yahoo.com.cn).

Lu Zhang is with the School of Automation, Nanjing University of Science and Technology, Nanjing, 210094 China

Qingwei Chen is with the School of Automation, Nanjing University of Science and Technology, Nanjing, 210094 China

where W is TCP window size (packets), q is queue length at router, $R(t) = T_p + q(t)/C$ is round-trip time, T_p is propagation delay, C is link capacity (packets/sec), $N(t)$ is the number of TCP sessions, $p(t)$ is the drop/marking probability and $p(t) \in [0,1]$ .The queue length q and window size W are positive and bounded variables, i.e., $q \in [0, \bar{q}]$, $W(t) \in [0, \bar{W}]$, where $\bar{q}$ and $\bar{W}$ denote buffer capacity and maximum window size respectively.

Taking $(W(t), q(t))$ as the state variables and $p(t)$ as input, the operating point (W_0, q_0, p_0) is defined by setting $\dot{W} = 0, \dot{q} = 0$. Assuming TCP sessions $N(t) \equiv N$ and RTT $R(t) \equiv R_0$, so that the equilibrium point (W_0, q_0, p_0) is uniquely corresponding to the networks parameters (N, C, T_p) . We linearize equation (1) at the above operating point and perform a Laplace transform on the small-signal linearized differential equations, then open-loop transfer function from p to q is determined by:

$$P(s) = \frac{C^2 e^{-sR_0}/2N}{(s + 2N/R_0^2 C)(s + 1/R_0)} \tag{2}$$

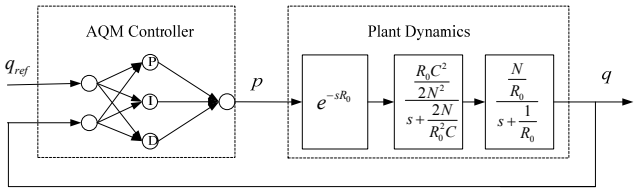


Fig.1. AQM control system based on neural networks

The block diagram of AQM control system based on neural networks is illustrated in Figure 1. The reference input of the system is desired queue length q_{ref} at router, and output is actual queue length q , the plant to be controlled is composed by TCP window-control model and queue dynamics. The goal of AQM controller design is to provide a stable closed-loop system and keep actual queue length q as near as possible around the desired queue length q_{ref} , even when there exist variations in networks parameters and modeling errors.

III. AQM ALGORITHM BASED ON NEURAL NETWORKS

PID (Proportional-Integral-Differential) controller is very pervasive in many control area, since it is easy to understand and has the merits of simple structure and good performance. But tuning of controller parameters is not a easy work when the system to be controlled is nonlinear and time-varying. In this paper, we present a novel adaptive PID controller based on neural networks to realize the objective of AQM. The AQM scheme combines the merits of both PID controller and neural networks and the controller parameter can be tuned adaptively by using online learning algorithm of neural networks. Therefore the AQM controller is robust to the variations in network parameters and modeling errors.

AQM controller based on neural networks as shown in Figure 1 is formed by a three layers feed-forward neural networks with 2-3-1 structure. The input layer of neural networks has two input nodes, one is desired queue length at router and the other is the feedback signal of actual queue length, the activation function of both neurons are pure-line. There are three neurons in the hidden layer and they are proportional, integral and differential neurons respectively. That means these neurons implement proportional, integral and differential manipulation for queue error. The third layer has only one node and its activation function is also pure-linear, and its output is the linear weighting of the hidden neurons' output. For the sake of simplification, Let weights of input layer are $w_{1i}^{(1)} = 1, i = 1,2$, then at the sampling time constant k , the outputs of the input nodes are:

$$y_i^{(1)}(k) = f(x_i^{(1)}(k)) = x_i^{(1)}(k) = \begin{cases} q_{ref} & i = 1 \\ q(k) & i = 2 \end{cases} \tag{3}$$

The hidden layer of neural networks has three neurons and set the weights from input layer to hidden layer are $w_{1i}^{(2)} = 1$, $w_{2i}^{(2)} = -1$ respectively, then the input of i^{th} neuron of hidden layer is:

$$x_i^{(2)}(k) = x_1^{(1)}(k) - x_2^{(1)}(k) = q_{ref} - q(k) \tag{4}$$

where $i = 1,2,3$. Let $e(k) = q_{ref} - q(k)$, then the output of proportional neuron in the hidden layer is

$$y_1^{(2)}(k) = x_1^{(2)}(k) = e(k) \tag{5}$$

the output of integral neuron in the hidden layer is:

$$y_2^{(2)}(k) = y_2^{(2)}(k-1) + x_2^{(2)}(k) = \sum_{j=1}^{k} e(j) \tag{6}$$

the output of differential neuron in the hidden layer is:

$$y_3^{(2)}(k) = x_3^{(2)}(k) - x_3^{(2)}(k-1) = e(k) - e(k-1) \tag{7}$$

the output of overall neural networks is linear weighting of the hidden layer neurons' output, that is:

$$y_{NN}(k) = \sum_{i=1}^{3} w_i x_i^{(3)}(k) = \sum_{i=1}^{3} w_i y_i^{(2)}(k) \tag{8}$$

where w_i are the weights from hidden layer to output layer.

Finally, the output of AQM controller determines the mark/drop probability $p(k)$, that is,

$$p(k) = \begin{cases} 0 & y_{NN}(k) \le 0 \\ y_{NN}(k) & 0 < y_{NN}(k) < 1 \\ 1 & y_{NN}(k) \ge 1 \end{cases} \tag{9}$$

From (9), we know that the output of AQM controller is determined by the output of neural networks, which is furthermore rely on the connection weights of neural networks. As for the AQM controller presented in this paper, its output mainly depend on the weights from hidden layer to output layer. Since the objective of AQM control system shown in figure 1 is to determine an appropriate mark/drop probability $p(k)$ so that actual queue length q is kept near around the desired queue length q_{ref} . Define the cost function as:

$$E(k)=\frac{1}{2}[q_{ref}(k)-q(k)]^2 \tag{10}$$

The tuning rule of weights from hidden layer to output layer is as follows:

$$w_i(k+1)=w_i(k)-\eta\frac{\partial E(k)}{\partial w_i(k)} \tag{11}$$

where η is learning rate, equation (11) is essentially the gradient descend method and gradient can be rewritten by:

$$\frac{\partial E(k)}{\partial w_i(k)}=\frac{\partial E(k)}{\partial q(k)}\frac{\partial q(k)}{\partial p(k)}\frac{\partial p(k)}{\partial y_{NN}(k)}\frac{\partial y_{NN}(k)}{\partial w_i} \tag{12}$$

where $\partial E(k)/\partial q(k)=-e(k)$, $\partial p(k)/\partial y_{NN}(k)=1$, $\partial y_{NN}(k)/\partial w_i(k)=y_i^{(2)}(k)$, while (12) contains the term of $\partial q(k)/\partial p(k)$, which is difficult to be obtained because of nonlinearity and uncertainties of the TCP dynamics. Therefore this term is instead of its sign function:

$$\delta(k)\triangleq\frac{\partial q(k)}{\partial p(k)}=\mathrm{sgn}(\frac{q(k)-q(k-1)}{p(k-1)-p(k-2)}) \tag{13}$$

The above substitution is feasible since the term of $\partial q(k)/\partial p(k)$ is one of the factors of (12), its sign determine the direction of weights tuning, while the its value can be compensated by selecting appropriate learning rate η. From (11), (12), (13), the tuning rule of neural networks weights is as follows:

$$w_i(k+1)=w_i(k)+\eta e(k)\delta(k)y_i^{(2)}(k) \tag{14}$$

Where $y_i^{(2)}(k)$ can be obtained from (5),(6),(7), and $i=1,2,3$.

IV. IMPLEMETATION OF AQM ALGORITHM BASED ON NEURAL NETWORKS WITH MODIFIED ECN

In current TCP/IP networks, TCP relies on packet drop as the indication of congestion, which is an implicit signal. However, unnecessary dropped packet may result in waste of resource, and increasing of delay. In order to avoiding unnecessary packet drops, Explicit Congestion Notification (ECN) has been proposed to provide early indication to sources about imminent congestion [12]. ECN marking is a mechanism to provide information quickly and unambiguously about the network to the user. Over smaller time scales, TCP's response to ECN can be less conservative than its response to a dropped packet as an indication of congestion.

Before any ECN-enabled data exchange can take place between two endpoints, they first have to successfully negotiate the use of ECN. ECN negotiation happens during the TCP connection setup phase. The ECN bit include ECN-Capable (ECT) and Congestion Experienced (CE) bit in the IP header, and ECN-Echo bit in the TCP header. The negotiation procedure is as follows.

The sender set the ECN-Echo bit in the packet header, and transmits the packet to the receiver. Once receiving the packet, the receiver sets ECN-Echo bit in the ACK packet's TCP header, and sends the ACK packet back to the sender. When the sender receives the ACK packet, the ECN Capability is negotiated, and both endpoints start an ECN-capable transport by setting the ECT bit in the IP header of data packets. When congestion happens, the router will mark ECT-enable packets by setting the CE bit in the IP header. After the packet is received, receiver sets the ECN-Echo bit in the header of the corresponding ACK packet, and sends it to sender. Sender decreases the cwnd (congestion window) after received the ACK packet. The ECN mechanism in TCP/IP is shown in Fig.2.

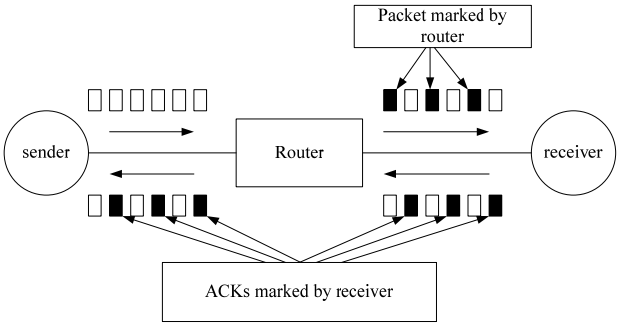


Fig.2. ECN mechanism in TCP/IP

The Explicit Congestion Notification (ECN) allows active queue management mechanisms such as RED to probabilistically mark (rather than drop) packets when the average queue length lies between two thresholds, if both the sender and receiver are ECN capable. The main objective of the proposed AQM algorithm is to modify ECN response in order to increase throughput, reduce rate fluctuations and delays. The pseudo-code for AQM algorithm with modified ECN is given in Table Ⅰ.

TABLE Ⅰ
PSEUDO-CODE FOR ROUTER WITH MODIFIED ECN

```
At router
When a pakect arrive at router
    Calculate queue length q
if  q ≥ q_max
   Drop the packet
else if  q_min < q < q_max
    if   ECT=1
         Calculate mark probability p(k) by (9)
         Set CE=1 with the probability p(k)
    else  Calculate drop probability p(k) by (9)
          drop the packet with the probability p(k)
else  packet pass normally
end
```

The AQM controller will compute the mark/drop probability at router so that the TCP sources will regulate their sending rates to keep the queue of router tracking the desired value. The adaptive PID algorithm based on neural networks with modified ECN can improve the transient performance of closed-loop control system and enhance the robustness to time-varying network situations.

V. SIMULATIONS

We verify the presented AQM algorithm based on neural networks via simulation by using the NS-2 simulator [13]. In the simulation, we consider the network topology as shown in Figure 3. The bottleneck link lies between node R1 and node R2, and its capacity is 15 Mbps (3750 packets/sec) and delay is 20ms. All the other links in the network have 10 Mbps capacity and 2ms propagation delay. The size of queue buffer is 300 packets. The TCP sources are all running Reno algorithm with packet size 500Bytes. We focus on the performance of instantaneous queue length and average delay for each algorithm when the load level of networks is constant or time-varying respectively.

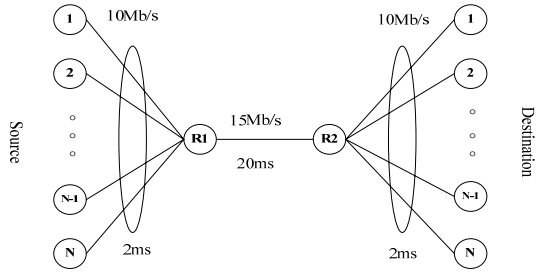
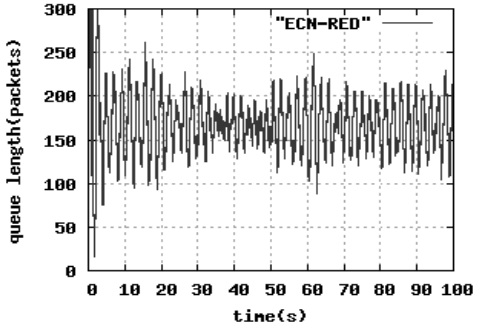


Fig.3. Network topology

Firstly, we assume that load level of network is constant, i.e. the given TCP sessions *N*=250. Simulation runs for 100 seconds and the reference queue length is $q_{ref} = 150$ packets. The proposed PID controller based on neural networks (PIDNN) will be compared with the typical RED and PID algorithm. The initial parameters of PID algorithm are chosen as $K_p = 2.58392e-6$, $K_i = 2.9829e-6$, $K_d = 3.50735e-8$, the thresholds of RED are set as $Th_{\min} = 80$ and $Th_{\max} = 150$, the learning rate of PIDNN is $\eta = 3.5\text{e-}7$.

The Fig.4 is the time evolution of the instantaneous queue length by using RED, PID and PIDNN algorithm with ECN marks respectively. The Simulation results show that ECN-RED and ECN-PID have slow transient response and the large value of fluctuation around the desired queue length. While the ECN-PIDNN algorithm keeps better stability of queue length with small fluctuation, and the transient response is fast. Therefore the proposed algorithm is superior to the other two AQM schemes on the aspects of queue stability and transient response.



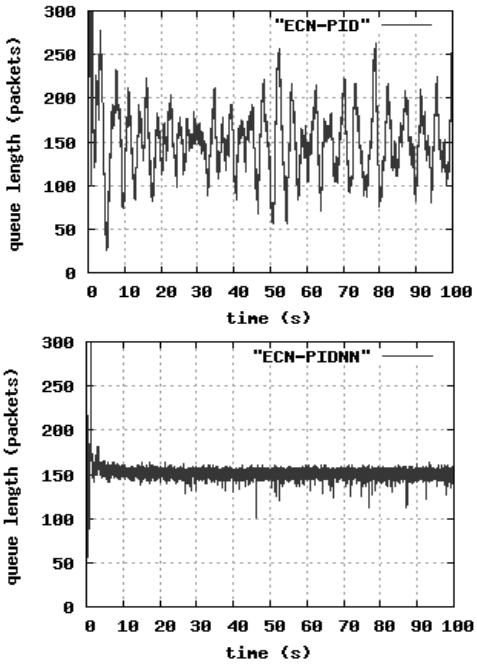


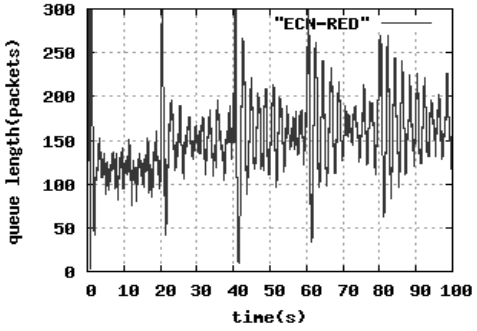
Fig.4. Queue length evolution for constant load level

Table Ⅱ shows the average delay for each algorithm when the load level is constant. It is obvious that the average delay for the presented algorithm is the smallest comparatively.

TABLE Ⅱ
AVERAGE DELAY FOR CONSTANT LOAD

Algorithm	ECN-RED	ECN-PID	ECN-PIDNN
Average delay (ms)	96.603	84.108	83.655

Next we compare the presented AQM scheme with RED and PID algorithms when the load level of networks is time-varying. Assume that 50 TCP flow is started at the time instant of 0s, 20s, 40s, 60s, 80s respectively, that means the number of TCP sessions is from *N*=50 to *N*=250. The initial parameters of the algorithms are same as those in above simulation. Fig.5 is the time evolution of the queue length for each of the above AQM schemes when the load level is time varying. The Simulation results show that the PIDNN algorithm achieves better stability and robustness to varying load than the other two schemes in dynamic network environment. Furthermore, it can lead to fast transient response.



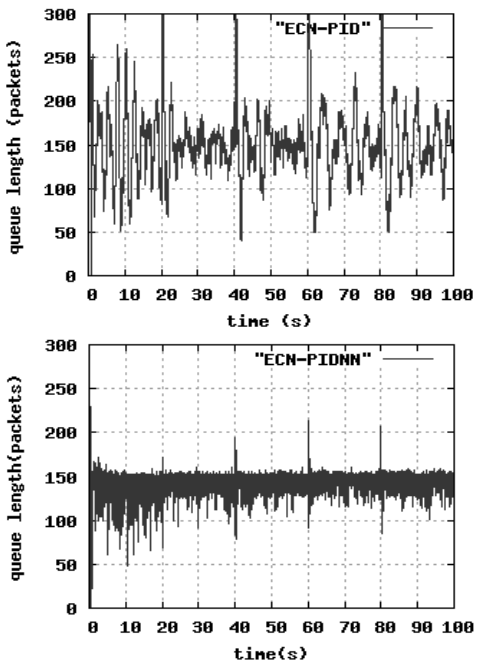


Fig.5. Queue length evolution for time-varying load level

Table Ⅲ shows the average delay for each algorithm when the load level is time-varying. We can see that the average delay for ECN-PIDNN algorithm is still the smallest among those AQM schemes.

TABLE Ⅲ
AVERAGE DELAY FOR TIME-VARYING LOAD

Algorithm	ECN-RED	ECN-PID	ECN-PIDNN
Average delay (ms)	85.191	83.894	77.425

From the mechanism analysis, we know that PIDNN is an adaptive AQM algorithm based on neural networks which can online tune the controller parameters according to the variations of network load and parameters. Hence it can keep the stability of queue length by adjusting the mark/drop probability in time. But the traditional PID controller and RED may lead to big fluctuations about operating level of queue for fixed parameters. On the other hand the mean time delay for the presented AQM algorithm is smaller than that for RED and PID controller.

VI. CONCLUSION

Active queue management is a very active research area in the networking community, which intends to achieve small queuing delay and high throughput by purposely dropping/marking the packets at the intermediate nodes. A lot of research works for AQM based on control theory has been presented, for example PI or PID controller etc. But these methods can not guarantee the good performance robustness to the time-varying and uncertain network environment for the fixed controller parameters. In this paper, an adaptive neural networks-based PID controller for AQM with modified ECN is presented. It can overcome the shortcoming of the traditional PID controller by introducing the gradient descend algorithm for the tuning of PID controller parameters, which can improve the performance and robustness of AQM control system greatly. The simulation results also demonstrate the proposed algorithm is superior to typical RED and PID controller.

REFERENCES

[1] S Floyd, V Jacobson, "Random early detection gateway for congestion avoidance," *IEEE／ACM Transactions on Networking,* vol. 4, no.1, pp. 397-413, 1993..

[2] S Floyd, Gummadir Shenkers, etal. "Adaptive RED: an algorithm for increasing the robustness of RED active queue management," [EB/OL].[2005-01-10]. http://www.icir.org/floyd/

[3] Athuraliya, S, Li, V.H, Low, S.H, et al. "REM: active queue management," *IEEE Transactions on Networking*, vol.15, no.3，pp. 48-53, 2001

[4] W C. Feng , D.D.Kandlur, D.Saha,K,G.Shin., "The BLUE active queue management algorithms," *IEEE/ACM transaction on Networking*, vol. 10, No.4, pp.513-528, August 2000.

[5] Misra V, Gong W B, Towsley D, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," *Proceedings of the ACM／SIGCOMM 2000*, pp.386-399, 2000

[6] C. Hollot, V Misra, D. Towsley and W. B. Gong, "A control theoretic analysis of RED," *Proceedings of IEEE INFOCOM,* 2001

[7] C.Hollot, V.Misra, "On designing improved controllers for AQM routers supporting TCP flows," *Proc. of INFOCOM2000*, Anchorage: IEEE Communications Society, pp.1726-1734, 2001.

[8] Ren Fengyuan, Wang Fubao, Ren Yong, et al, "PID controller for active queue management," *Journal of Electronics and Information Technology*, vol. 25, no.1, pp. 94-99, 2003.

[9] Wei Jiaolong, Qian Jingjing, "An enhanced BLUE algorithm based on highest-rate-flow discards," *Proc. of SPIE* Vol. 5985, 59851Y, 2005

[10] Eitan Altman, Konstantin Avrachenkov, and Chadi Barakat, " A Stochastic Model of TCP/IP With Stationary Random Losses," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, April 2005

[11] Yuedong Xu, Huifang Deng, et al, "Tuning PUPID Active Queue Management Controllers Supporting TCP/IP Flows," *IEEE,*, 2005.0-7803-9015-6

[12] Kuzmanovic. A. "The Power of Explicit Congestion Notification," ACM/SIGCOMM, Aug., pp. 61-72, 2005

[13] USC/ISI, Los Angeles, CA. The NS simulator and the documentation. [Online] Available: http://www..isi.edu/nsnam/ns/.