지능형 능동 큐 관리 제어기

*김재만, **최윤호, *박진배
*연세대 전기전자공학과 **경기대학교 전자공학부

Intelligent AQM Controller

*Jae Man Kim, **Yoon Ho Choi, *Jin Bae Park
*Dept. Electrical & Electronic Eng. Yonsei Univ. **School of Electronic Eng. Kyonggi Univ.

Abstract - In this paper, we present the wavelet neural network (WNN) controller as an active queue management(AQM) in end-to-end TCP network. AQM is important to regulate the queue length and short round trip time by passing or dropping the packets at the intermediate routers. As the role of AQM, the WNN controller adaptively controls the dropping probability of the TCP network and is trained by gradient-descent algorithm. We illustrate our result that WNN controller is superior to PI controller via simulations.

1. Introduction

Congestion control for TCP network is an essential problem in the Internet, which has been an main issue over the last decade [1],[2],[3]. In TCP network, the purpose of congestion control is to retrieve from the congested network condition, or to inhibit an incipient congestion. If a packet needs to be transmitted across a link but finds the link is busy, the packet must wait in the output buffer. Since amount of buffer space is finite, an arriving packet may find that the buffer is completely filled with other packets waiting for transmission. If there is not enough memory to buffer an incoming packet, a decision must be made to either drop the arriving packet or remove one or more already-queued packets to make room for the newly arrived packet.

The active queue management(AQM) has been proposed for the end-to-end congestion control in the Internet. AQM primarily responds to network congestion when a queue begins to increase and maintains a queue size at a predefined level in the router. By keeping the average queue size small, AQM has the ability to preserve efficient queue utilization and reduce the delays seen by the network flow, which is very particularly important for real-time interactive applications. RED, PI algorithms have been proposed to AQM scheme [4]. But these algorithms are too sensitive to configure the parameters and take the network as the linear and constant system though the actual network is nonlinear system. Therefore, we present the wavelet neural network (WNN) controller for AQM in the TCP network router. The proposed WNN can apply to the linear system as well as the time-varying nonlinear system.

The WNN controller is trained to minimize the error between the desired value and the system output value, which is adaptive than RED and PI/PID controller. Also we present GD algorithm for training the WNN controller.

2. Preliminaries

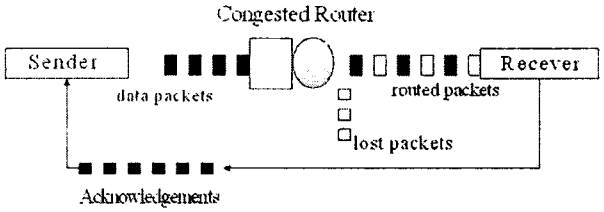
2.1 Dynamic System of TCP Network

Based on fluid-flow and stochastic differential theory, a nonlinear dynamic model for TCP flow control is as follows:

$$\dot{W}(t)=\frac{1}{R(t)}-\frac{W(t)W(t-R(t))}{2R(t-R(t))}p(t-R(t)), \quad (1)$$

$$\dot{q}(t)=\frac{W(t)}{R(t)}N(t)-C, \quad (2)$$

where W denotes TCP window size, q is a queue length at a router, $R(t)$ is a RTT, calculated by $q(t)/C+T_p$. Here C is a link capacity, T_p is a propagation delay, N is the number of TCP sessions(load factor) and p is the probability of a packet loss. These coupled differential equations show that the problem of AQM algorithm can be converted into the design of the controller. This is very important itself, because the algorithm based on control theory is certainly superior to the RED algorithm on the performance, such as a robustness, stability and responsibility. The AQM controller using the simplified and inaccurate linear constant model should not be optimal, because the real network is rapidly changed, that is, the network parameters, including RTT, the number of TCP sessions, and the capacity of the link are hardly kept at the constant values for a long time.



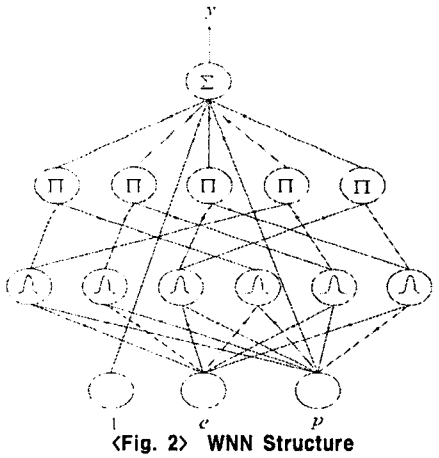
<Fig. 1> Link structure between Sender and Receiver

2.2 WNN Controller

The WNN is designed as a four-layer structure. Each layer has one or more nodes Fig. shows the schematic diagram of the four-layer WNN. Fig.1 shows the schematic diagram of the four-layer WNN. In this paper, the control objective is to design the WNN controller which adjusts the queue length of TCP network to the command value. To accomplish this control objective, the error $e=q^*-q$ should be minimized, where q and q^* represent the actual queue lengths and the desired queue length, respectively. The inputs of WNN controller are e and p, which are previous probability of a packet loss. In the mother each node performs a wavelet ϕ_j that is derived from its mother wavelet. For the jth node,

$$\phi(z_{jk})=\phi(\frac{x_k-m_{jk}}{d_{jk}}), \quad (3)$$

where m_{jk} and d_{jk} are the translation and dilation in the jth term of the kth input to the node of mother wavelet layer, respectively. In this paper, the first derivative of a Gaussian function, $\phi(z)=-zexp(-0.5z^2)$, is selected as a mother wavelet. The output of a WNN, which is a control input of TCP model, is composed by each wavelet and parameters as follows:



<Fig. 2> WNN Structure

$$y=\Phi(x,\theta)=\sum_{i=1}^{N_w}c_i\Phi_i(x)+\sum_{k=1}^{N_i}a_kx_k+a_0, \quad (4)$$

where, a_0 and a_k are connection weight between input nodes and output nodes, respectively. N_i and N_w represent the number of nodes in the

input layers and the product layers, respectively. c_l is a connection weight between wavelet nodes and output nodes, and W is the set of adjustable parameters:

$$W=\{m_{jk}, d_{jk}, c_l, a_k, a_0\}, \tag{5}$$

2.3 Training Algorithm

In this paper, we use a direct adaptive controller based on WNN. As shown in Fig. , the error, that is the difference between the length of queue, is the input of direct adaptive controller. To describe the training algorithm of the WNN using the GD method, first the cost function is defined as

$$J(W(n))=\frac{1}{2}(y_d(n)-y(n))^2=\frac{1}{2}e^2(n), \tag{6}$$

where, $y(n)$ is the output value of the WNN and $y_d(n)$ is the desired output value. The training method is based on the minimization of the cost function. The minimization is performed by the following GD method:

$$\begin{aligned} W(n+1) &= W(n)-\Delta W(n) \\ &= W(n)-\eta\frac{\partial J(W(n))}{\partial W(n)}, \end{aligned} \tag{7}$$

where, η is the learning rate of a WNN.
The partial derivative of the cost function with respect to weighting vector is

$$\frac{\partial J(W(n))}{\partial W(n)}=-e(n)\frac{\partial y(n)}{\partial W(n)}, \tag{8}$$

$$\frac{\partial y(n)}{\partial a_0}=1, \tag{9}$$

$$\frac{\partial y(n)}{\partial a_k}=x_k, \tag{10}$$

$$\frac{\partial y(n)}{\partial c_l}=\Phi_l(x), \tag{11}$$

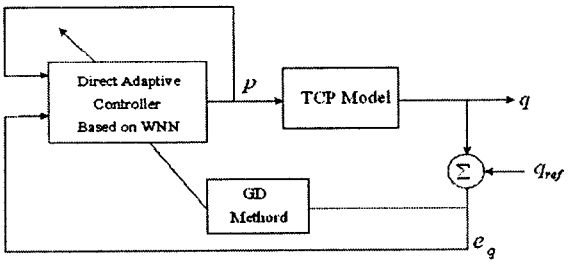
$$\frac{\partial y(n)}{\partial m_{jk}}=-\frac{c_l}{d_{jk}}\frac{\partial \Phi_l(x)}{\partial z_{jk}}, \tag{12}$$

$$\frac{\partial y(n)}{\partial d_{jk}}=-\frac{c_l}{d_{jk}}z_{jk}\frac{\partial \Phi_l(x)}{\partial z_{jk}}, \tag{13}$$

where,

$$\frac{\partial \Phi_l(x)}{\partial z_{jk}}=\phi(z_{j1})\phi(z_{j2})\cdots\dot{\phi}(z_{jk})\cdots\phi(z_{jNi}), \tag{14}$$

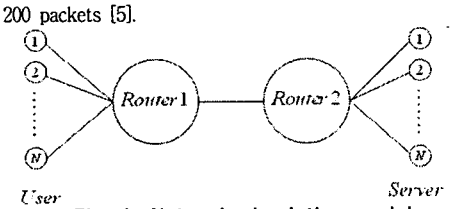
$$\frac{\partial \Phi_l(x)}{\partial z_{jk}}=\phi(z_{j1})\phi(z_{j2})\cdots\dot{\phi}(z_{jk})\cdots\phi(z_{jNi}). \tag{15}$$



<Fig. 3> The block diagram of TCP control

3. Simulations

In this section, we verify the performance of WNN controller through computer simulations, and then compare its results with PI controller. We use a simple bottleneck network topology as shown in Fig. 4. The capacity of bottleneck link between the router 1 and 2, is 15Mbps (3750 packets/sec, packet size is 500 bytes). And the propagation delay is 15ms. All link between users and router 1 have 10Mbps capacity and those propagation delay is also 15ms. The size of all queues is 300 packets. The number of TCP sessions is 120, and the reference queue length is 200 packets [5].

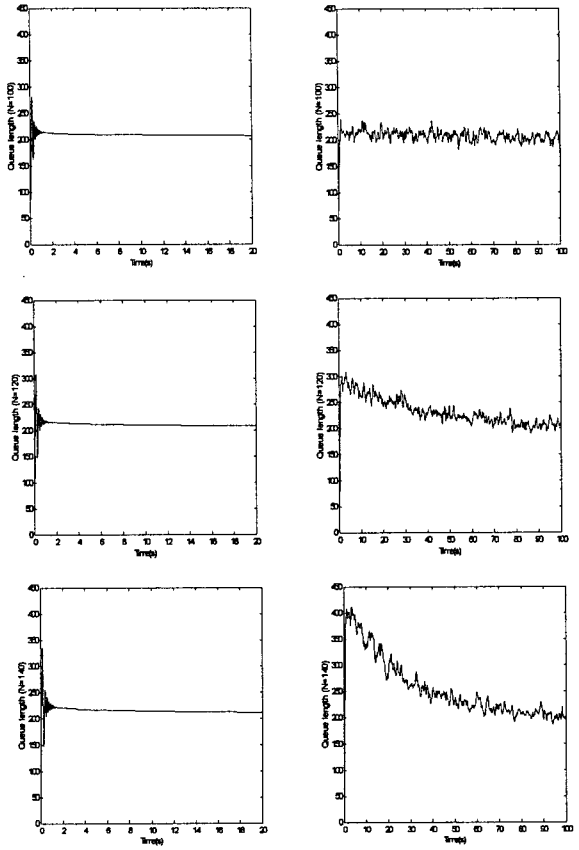


<Fig. 4> Network simulation model

We compare the control performance of the PI controller and the WNN controller under three different number of TCP sessions. The instantaneous queue lengths for PI controller and WNN controller are shown in Fig. 5. From the result of Fig. 5, we can know that the WNN controller have better performance than PI controller under three different number of sessions. Table 1 shows the mean square error(MSE) of each controllers.

<Table 1> Compare of mean square error for each controller

	PI controller	WNN controller
N=100	0.2845	0.2332
N=120	0.3262	0.2652
N=140	0.6204	0.2965



(a) WNN controller (b) PI controller

<Fig. 5> Queue lengths of WNN controller and PI controller

4. Conclusion

The WNN controller was proposed for AQM in TCP network. The WNN controller was operated as a direct adaptive controller, where the output was a dropping probability of packets at router. The WNN controller maintained the actual queue size close to a reference queue value, which was trained by the GD method. Through simulation results, it was shown that the proposed control method was more adaptive in the dynamic TCP network system.

[Reference]

[1] X. Deng, Y. Sungwon, G. Kesidis, C. R. Das, "A Control Theoretic Approach for Designing Adaptive AQM Schemes", GlOBECOM, vol. 5, pp. 2947-2951, 2003.

[2] F. Yanfei, R. Fengyuan, L. Chuang, "Design a PID Controller for Active Queue Management", Proc. of IEEE International Symposium on Computers and Communication, vol. 2, pp. 985-990, 2003.

[3] P. F. Quet, H. Ozbay. "On the Design of AQM Supporting TCP Flows Using Robust Control Theory", IEEE Trans. on Automatic Control, vol.49, pp. 1031-1036, 2004

[4] C. V. Hollet, V. Misra, D. Towsley, and W. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows", IEEE Trans. Automatic Control, Vol. 47, pp. 945-959, 2002.

[5]R. Fengyuan, R. Yong, S. Xiuming, "Design of a Fuzzy Controller for Active Queue Management", Computer Communication, vol. 25, pp. 874-883, 2001.