An Adaptive Neuron AQM for a Stable Internet

Jinsheng Sun and Moshe Zukerman

The ARC Special Research Centre for Ultra-Broadband Information Networks, Department of Electrical and Electronic Engineering, The University of Melbourne, Victoria, 3010, Australia. {j.sun,m.zukerman}@ee.unimelb.edu.au

Abstract. Recognizing that Internet congestion control is a complex nonlinear system, we propose here to use an intelligent controller to improve its stability and performance. In particular, we propose here a new, powerful, easy-to-configure and robust active queue management (AQM) scheme called adaptive neuron AQM (AN-AQM). We present extensive simulation results for AN-AQM, over a wide range of network conditions and scenarios, that demonstrate its attributes. We demonstrate its robustness in various realistic environments involving bursty HTTP connections and non-responsive UDP connections. Comparison with other AQM schemes has demonstrated the superiority of AN-AQM over well-known AQM schemes in achieving faster convergence to queue length target, and smaller queue length jitter.

Key words: Congestion control, Active queue management, Neuron, AQM

1 Introduction

Internet congestion control aims to achieve efficient resource utilization, acceptable packet loss and stable operation. It is based on two parts: 1) the end-to-end Transport Control Protocol (TCP) and 2) buffer management. The traditional algorithm for buffer management is Drop-Tail, which drops packets only if buffer overflows. This passive behavior may create a synchronized process alternating between periods of excessive packet loss and under utilization which in turn results in unacceptable queueing delay, inefficient (low) link utilization and instability. To mitigate such problems, active queue management (AQM) has been introduced [2] to improve performance by actively trigger packet drops before a buffer overflows.

AQM has been a very active research area, and many AQM schemes have been proposed (see for example [1, 4–8, 10–16, 19, 20] and references therein).

However, the many published AQM proposals fail to achieve optimal congestion control operation because they use fixed parameters, which is inadequate because the network state varies with time. Accordingly, an intelligent AQM controller is required for the Internet, which is complex, highly nonlinear and time varying.

In this paper, a novel neuron-based AQM scheme is proposed. It is called *Adaptive Neuron AQM* (AN-AQM). We apply the ideas of [3, 22], where an

adaptive neuron PID controller is designed for a multi-model plant. Extensive simulation results over a wide range of scenarios show that AN-AQM can control queue length and achieves fast queue-length convergence to a desirable target. These performance attributes are still maintained following significant changes to network conditions, even for long-delay networks. We also demonstrate by simulations that AN-AQM is more efficient and stable than other well-known AQM schemes.

The remainder of the paper is organized as follows. In Section 2, we describe the AN-AQM scheme in details. Then, in Section 3, we present simulation results to demonstrate that AN-AQM is effective, robust and outperforms other well-known AQM schemes. Finally, we conclude the paper in Section 4.

2 The AN-AQM Scheme

The AN-AQM scheme can be described by the following equation,

$$p(k) = p(k-1) + \Delta p(k) \tag{1}$$

where $p(k)$ is the packets dropping probability, $\Delta p(k)$ is the increment of packets dropping probability given by an adaptive neuron

$$\Delta p(k) = K \sum_{i=1}^{6} w_i(k) x_i(k) \tag{2}$$

and $K > 0$ is the neuron proportional coefficient; $x_i(k)(i = 1, 2, ..., 6)$ denote the neuron inputs, and $w_i(k)$ is the connection weight of $x_i(k)$ determined by the learning rule.

Let

$$e(k) = q(k) - Q_T \tag{3}$$

denote the queue length error, where $q(k)$ is the queue length, and Q_T is the target queue length. Let

$$\gamma(k) = \frac{r(k)}{C} - 1 \tag{4}$$

denote the normalized rate error, where $r(k)$ is the input rate of the buffer at the bottleneck link, and C is the capacity of the bottleneck link. The inputs of AN-AQM scheme are $x_1(k) = e(k) - e(k-1)$, $x_2(k) = e(k)$, $x_3(k) = e(k) - 2e(k-1) + e(k-2)$, $x_4(k) = \gamma(k) - \gamma(k-1)$, $x_5(k) = \gamma(k)$, and $x_6(k) = \gamma(k) - 2\gamma(k-1) + \gamma(k-2)$. According to Hebb [3], the learning rule of a neuron is formulated by

$$w_i(k+1) = w_i(k) + d_i y_i(k) \tag{5}$$

where $d_i > 0$ is the learning rate, and $y_i(k)$ is the learning strategy. The associative learning strategy given in [3] is as follows:

$$y_i(k) = e(k)p(k)x_i(k). \tag{6}$$

where $e(k)$ is used as teacher's signal. This implies that an adaptive neuron, which uses integrated Hebbian Learning and Supervised Learning, makes actions and reflections to the unknown outsides with associative search. It means that the neuron self-organizes the surrounding information under supervision of the teacher's signal $e(k)$. It also implies a critic on the neuron actions.

The AN-AQM scheme is based on the following nine parameters.

1. Sampling time interval T; an appropriate value is $T = 0.001s$.
2. Target queue length Q_T; this parameter decides the steady-state queue length value, which will affect the utilization and the average queueing delay. A high target will improve link utilization, but will increase the queueing delay. The target queue length Q_T should be selected according the Quality of Service (QoS) requirements.
3. The neuron proportional coefficient K; a suggested value is $K = 0.01$.
4. The learning rate d_1; a suggested value is $d_1 = 0.00001$.
5. The learning rate d_2; a suggested value is $d_2 = 0.00001$.
6. The learning rate d_3; a suggested value is $d_3 = 0.00001$.
7. The learning rate d_4; a suggested value is $d_4 = 0.0001$.
8. The learning rate d_5; a suggested value is $d_5 = 0.0001$.
9. The learning rate d_6; a suggested value is $d_6 = 0.0001$.

The above values of K and $d_i (i = 1, 2, ..., 6)$ have been chosen based on trial and error by a few simulations. Nevertheless, Zhang *et al.* [22] showed that an adaptive neuron system is very robust and adaptable, so the choice of values of K, $d_i (i = 1, 2, ..., 6)$ is not that critical. This is also confirmed by simulation results presented in Section 3.9. The initial values of $w_i (i = 1, 2, ..., 6)$ do not affect the performance significantly; we use: $w_i = 0.00001, (i = 1, 2, ..., 6)$.

3 Performance Evaluation and Comparison

In this section, we conduct extensive simulations using *ns2* [9] to demonstrate the performance attributes of AN-AQM and its superiority over other AQM schemes such as ARED [5], PI [7] and REM [1]. Our simulations and comparisons will cover the following attributes:

1. the ability to control the queue length to quickly converge to a given queue length target;
2. robustness to traffic loading under fixed and dynamic scenarios, to bottleneck link capacity, and to impact of traffic noise (UDP and HTTP);
3. robustness to Round Trip Propagation Time (RTPT) and effectiveness for long delay network;
4. performance under multiple bottleneck topology.

Many of the above attributes that we achieve for AN-AQM are the result of its first attribute, namely, stabilizing queue length at a target value Q_T. As mentioned above, if we can control the queue to stay close to a desirable target, we can achieve high throughput, predictable delay and low delay jitter. The low delay jitter also enables meeting QoS requirements for real time services especially when the queue length target is achieved independently of traffic conditions [21].

3.1 Single bottleneck topology

The single bottleneck network topology used in the simulation is shown in Figure 1. The only bottleneck link is the Common Link between the two routers. The other links are assumed to have sufficient capacity to carry their traffic. The sources use TCP/Reno. In the following simulations, unless mentioned otherwise, the following parameters are used: the packet size is 1000 bytes, the common link capacity is 45 Mb/s, the round trip propagation delay is 80 ms, the buffer size is 900 packets (twice the bandwidth-delay product of the network). The TCP connections always have data to send as long as their congestion windows permit. The receiver's advertised window size is set sufficiently large so that TCP connections are not constrained at the destination. The ack-every-packet strategy is used at the TCP receivers. The target queue length is set at 300 packets.

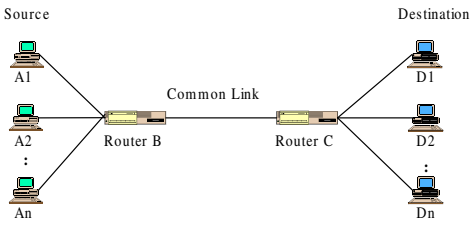

Fig. 1. The single bottleneck topology.

3.2 Performance for constant number of TCP connections

In this simulation experiment, we test whether AN-AQM can control and stabilize the queue length at an arbitrarily chosen target for different loads and link capacities. Figure 2 presents the instantaneous queue lengths for 300 and 1500 TCP connections. All sources start data transmission at time 0. We can see that AN-AQM is effective at stabilizing and keeping the queue length around the target Q_T.

In order to further demonstrate this ability of AN-AQM, we set Q_T at 50 and 500 for 800 TCP connections. The results are depicted in Figure 3. Again, we can see that AN-AQM is indeed successful in controlling the queue length at any arbitrary chosen target.

In order to test the performance of AN-AQM for different link capacities, we vary the capacity from 45 Mb/s to 15 Mb/s and to 115 Mb/s while the other parameters remain the same. The simulation results for 800 TCP connections are given in Figure 4. Again, we can see that the queue length is stable.

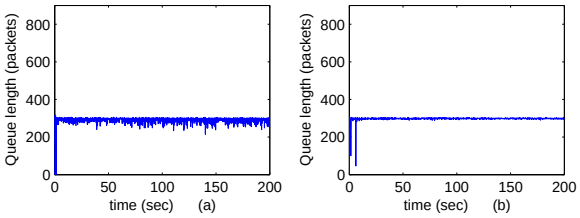

Fig. 2. Queue length variations for different numbers of greedy TCP connections: (a) 300 TCP connections (b) 1500 TCP connections.

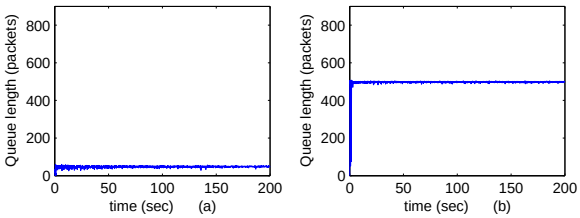

Fig. 3. Queue length variations for the following queue length targets: (a) $Q_T = 50$ (b) $Q_T = 500$.

3.3 Performance for different round trip propagation time

We now investigate the impact of round trip propagation time (RTPT) on the performance indices. Two simulations have been performed. In both there are 800 TCP connections with different RTPTs. In the first, the RTPTs are uniformly distributed between 20 and 140 ms, and in the second, they are uniformly distributed between 150 and 250 ms. Figure 5 presents the queue lengths for these two simulations. The results demonstrate that AN-AQM is still effective to stabilize queue length around the target with TCP connections having different RTPTs.

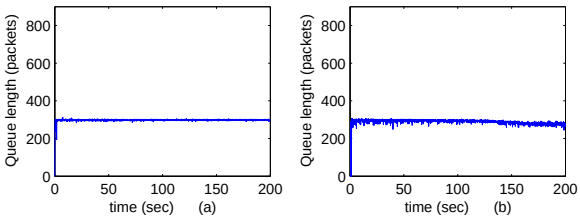

Fig. 4. Queue lengths variations for the following link capacities: (a)15 Mb/s (b)115 Mb/s.

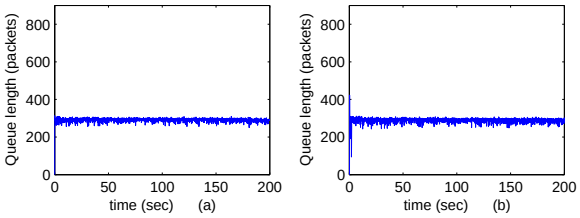

Fig. 5. Queue length variations in the following scenarios: (a) RTPTs are uniformly distributed between 20 and 140 ms (b) RTPTs are uniformly distributed between 150 ms to 250 ms.

3.4 Performance for TCP connections randomly start and stop

In this experiment, we investigate the impact of TCP connections randomly start and stop on the performance indices. We first present results of two simulation runs where we dynamically vary the number of active TCP connections. The number of TCP connections is varied from 500 to 1500 in the first and from 1500 to 500 in the second. In each of the runs, a group of 100 connections start (or stop), at the same time, at each 10 seconds interval. The instantaneous queue lengths are plotted in Figure 6. We can clearly see that AN-AQM is able to stabilize the queue length around the control target when the number of connections dynamically varies over time.

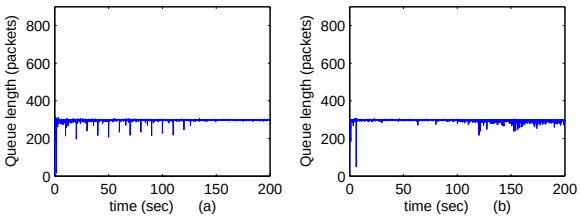

Fig. 6. Queue length variations for two cases: (a) Number of TCP connections varies from 500 to 1500 (b) Number of TCP connections varies from 1500 to 500.

Next, we perform two simulations involving random start and stop times, thus simulating staggered connection setup and termination. In the first, the initial number of connections is set to 300 and, in addition, 1200 connections have their start-time uniformly distributed over a period of 100 seconds. In the second simulation, the initial number of connections is set to 1500, out of which 1200 connections have their stop-time uniformly distributed over a period 100 seconds. The instantaneous queue lengths are plotted in Figure 7. We can clearly see that AN-AQM is able to stabilize the queue length around the control target.

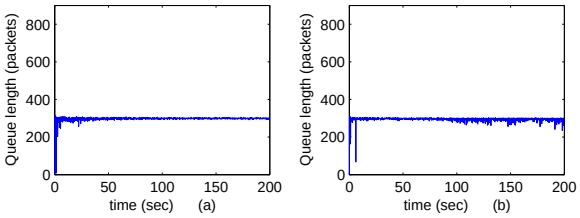

Fig. 7. Queue length variations under varying number of TCP connections (a) Number of TCP connections varies from 300 to 1500 (b) Number of TCP connections varies from 1500 to 300.

3.5 Performance for long delay network

Recalling that simulations in [11] have reported that AQMs, such as PI, RED and REM, are unstable when RTPT is 400 ms. In this experiment, we investigate here the performance of AN-AQM for a long delay network. In the simulation, there are 800 TCP connections, and the RTPTs are 500 ms. Figure 8 presents the queue length for the AN-AQM. The results demonstrate that AN-AQM is still effective in stabilizing the queue length around the target for TCP connections with long RTPTs.

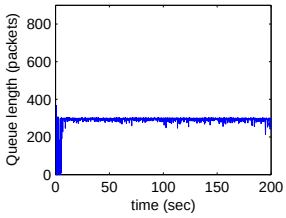

Fig. 8. Queue length variations for 800 TCP connections with the RTPT of 500 ms.

3.6 Performance for TCP connections mixed with both HTTP and UDP connections

In this simulation experiment, we investigate the performance impact of the disturbances caused by HTTP as well as UDP connections. We have considered 800 TCP connections with different RTPTs. These RTPTs are uniformly distributed between 50 and 500 ms. The bursty HTTP traffic involves 400 sessions (connections), and the number of pages per session is 250. The RTPTs of the HTTP connections are uniformly distributed between 50 and 300 ms. There are 400 UDP flows with propagation delay uniformly distributed between 30 to 250

ms. Each of the UDP sources follows an exponential ON/OFF traffic model, both the idle and the burst times have mean of 500 ms. The packet size is set at 500 bytes, and the sending rate during on-time is 64 kb/s. Figure 9 provides the queue lengths, which demonstrate that AN-AQM is robust to the disturbances caused by HTTP as well as UDP connections.

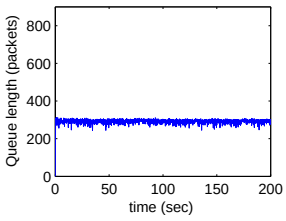

Fig. 9. Queue length variations for 800 TCP connections in the presence of additional UDP and HTTP flows.

3.7 Multiple bottlenecks

Here we extend the simple single bottleneck topology to a case of multiple bottlenecks. We consider the network topology presented in Figure 10. There are two bottlenecks in this topology. One is between Routers B and C, and the other is between Routers D and E. The link capacity of the two bottlenecks is 45 Mb/s and the capacity of other links is 100 Mb/s. There are three traffic group. The first group has N TCP connections traversing all bottleneck links, the second group has N_1 TCP connections traversing the bottleneck link between Routers B and C, and the third group has N_2 TCP connections traversing the bottleneck link between Routers D and E. The RTPTs of the first group are 80 ms, and for the second and third groups, they are 100 ms and 150 ms, respectively. Two simulation tests have been performed. In the first, $N = 500$, $N_1 = 200$, and $N_2 = 200$, and in the second, $N = 100$, $N_1 = 800$, and $N_2 = 400$. Figure 11 presents the queue lengths for these two case. The results demonstrate that AN-AQM is effective in stabilizing the queue length around the target for TCP connections in multiple bottleneck network.

3.8 Comparison with other AQMs

In this section, we perform simulations to compare the performance of AN-AQM with ARED [5], PI controller [7], and REM [1]. The network topology used in the simulation is the same as in Figure 1. The target queue length is set at 300 packets for all AQM algorithms. For ARED, we set the parameters: $min_{th} = 15$, $max_{th} = 585$ and $w_q = 0.002$, and other parameters are set the same as in [5]: $\alpha = 0.01$, $\beta = 0.9$, $intervaltime = 0.5s$. For PI controller, we use the default

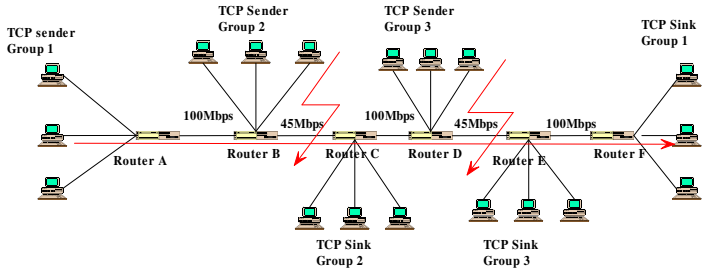

Fig. 10. The multiple bottleneck network topology.

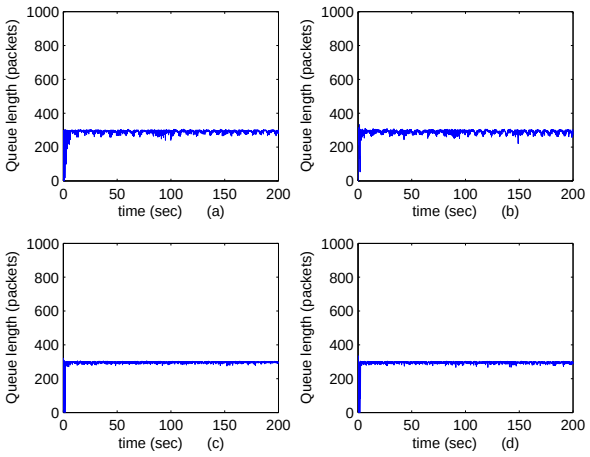

Fig. 11. Queue length variations in multiple bottleneck network: (a) Router B for $N = 500$, $N_1 = 200$, and $N_2 = 200$ (b) Router D for $N = 500$, $N_1 = 200$, and $N_2 = 200$ (c) Router B for $N = 100$, $N_1 = 800$, and $N_2 = 400$ (d) Router D for $N = 100$, $N_1 = 800$, and $N_2 = 400$.

parameters in *ns2*: $a = 0.00001822$, $b = 0.00001816$ and the sampling frequency $w = 170$. For REM, the default parameters of [1] are used: $\phi = 1.001$, $\gamma = 0.001$. The parameters of AN-AQM are set as specified in the previous Section.

In the simulation, the initial number of connections is set to 300, and 1200 additional connections have their start-times uniformly distributed over a period of 100 seconds. Figure 12 presents the queue lengths for all four AQMs. We can see that AN-AQM reacts and converges to the target queue occupancy of 300 faster than all other three AQMs.

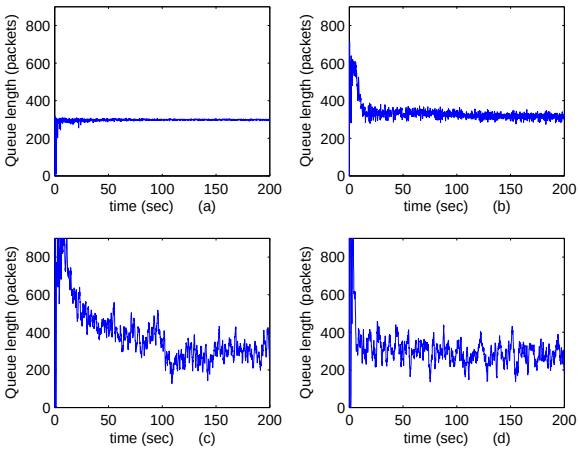

Fig. 12. Comparison of queue length variations: (a) AN-AQM (b) ARED (c) PI (d) REM.

In order to evaluate the performance in steady-state, we calculate, the average and the standard deviation of the queue length for the last 150 seconds. The results are presented in Table 1. We observe that AN-AQM queue length has the mean of 298.9, which is the closest to the target of 300, and achieved the lowest standard deviation of all AQMs.

Table 1. Mean and standard deviation of the queue length for various AQMs

	AN-AQM	ARED	REM	PI
Mean	298.9	322.2	321.0	285.3
Standard deviation	2.3	15.6	71.2	50.5

3.9 Comments on parameter robustness and complexity

We have examined the robustness and sensitivity of AN-AQM to parameters setting. To this end, we performed a set of simulations using again the single bottleneck network topology shown in Figure 1. In each simulation experiment, we fixed all parameters except one, and measure the sensitivity and the performance of AN-AQM resulting from changing the single parameter. The results which are presented in the extended version of this paper [18] demonstrate that AN-AQM is robust to neuron proportional coefficient misconfiguration and learning rate gain misconfiguration as it achieves stability in all cases in the similar way.

A thorough computational analysis of AN-AQM and its comparison with other AQM is planned for future study. However, since AN-AQM conducts computation every sample interval and not every packets (such as RED for example) we do not expect the computational complexity to be an impediment.

4 Conclusions

We have introduced a novel AQM scheme called AN-AQM. We have demonstrated by simulations that AN-AQM is able to maintain the queue length around the given target under different traffic loading and scenarios, different RTPTs, and different bottleneck link capacities. The numerous simulation results have also demonstrated that AN-AQM is powerful, easy to configure, and robust to bursty HTTP connections and non-responsive UDP connections. Comparison with other well-known AQM schemes has demonstrated the superiority of AN-AQM in achieving faster convergence to target queue length, and then maintaining the queue length closest to the target.

Acknowledgments. This work was jointly supported by grants from the Australian Research Council (Grant DP0559131), and the Natural Science Foundation of Jiangsu Province, China (No. BK2004132).

References

1. Athuraliya S., Low S.H., Li V.H. and Yin Q.: REM: Active Queue Management. IEEE Network Mag. **15** (2001) 48–53
2. Braden B., et al.: Recommendations on Queue Management and Congestion Avoidance in the Internet. IETF RFC2309 (1998)
3. Du Y. and Wang N.: A PID Controller with Neuron Tuning Parameters for Multimodel Plants. Proceedings of 2004 International Conference on Machine Learning and Cybernetics **6** (2004) 3408–3411
4. Feng W., Kandlur D., Saha D., Shin K.: The Blue Queue Management Algorithms. IEEE/ACM Transactions on Networking **10** (2002) 513–528
5. Floyd S., Gummadi R. and Shenker S.: Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. http://www.icir.org/floyd/red.html (2001)

6. Floyd S. and Jacobson V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Trans. Networking **1** (1993) 397–413
7. Hollot C.V., Misra V., Towsley D. and Gong W.: On Designing Improved Controllers for AQM Routers Supporting TCP Flows. Proceedings of IEEE INFOCOM 2001 **3** (2001) 1726–1734
8. Kunniyur S.S. and Srikant R.: An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. IEEE/ACM Transactions on Networking **12** (2004) 286–299
9. The NS Simulator and the Documentation, http://www.isi.edu/nsnam/ns/
10. Ranjan P., Abed E.H., and La R.J.: Nonlinear Instabilities in TCP-RED. IEEE/ACM Transactions on Networking **12** (2004) 1079–1092
11. Ren F., Lin C. and Wei B.: A Robust Active Queue Management Algorithm in Large Delay Networks. Computer Communications **28** (2005) 485–493
12. Sun J., Chen G., Ko K.T., Chan S. and Zukerman M.: PD-Controller: A New Active Queue Management Scheme. Proceedings of IEEE Globecom **6** (2003) 3103–3107
13. Sun J., Ko K.T., Chen G., Chan S. and Zukerman M.: PD-RED: to Improve the Performance of RED. IEEE Communications Letters **7**(2003) 406–408
14. Sun J., Chan S., Ko K.T., Chen G. and Zukerman M.: Neuron PID: A Robust AQM Scheme. Proceedings of ATNAC 2006 (2006) 259–262
15. Sun J., Zukerman M. and Palaniswami M.: Stabilizing RED Using a Fuzzy Controller. Proceedings of ICC 2007 (2007)
16. Sun J. and Zukerman M.: Improving RED by a Neuron Controller. Proceedings of ITC20 (2007)
17. Sun J. and Zukerman M.: RaQ: A Robust Active Queue Management Scheme Based on Rate and Queue Length. to appear in Computer Communications (2007)
18. Sun J. and Zukerman M.: An Adaptive Neuron AQM for a Stable Internet (extended version). unpublished report, available on-line: http://www.ee.unimelb.edu.au/staff/mzu/AN_AQM_extended_version.pdf
19. Wydrowski B. and Zukerman M.: GREEN: An Active Queue Management Algorithm for a Self Managed Internet. Proceedings of ICC 2002 **4** (2002) 2368–2372
20. Wang C., Li B., Hou Y.T., Sohraby K. and Long K.: A Stable Rate-based Algorithm for Active Queue Management. Computer Communications **28** (2005) 1731–1740
21. Wydrowski B. and Zukerman M.: QoS in Best-effort Networks. IEEE Communications Magazine **40** (2002) 44–49
22. Zhang J., Wang S. and Wang N.: A New Intelligent Coordination Control System for a Unit Power Plant . Proceedings of the 3rd World Congress on Intelligent Control and Automation (2000) 313–317