Edge-based active queue management

L. Zhu, N. Ansari, G. Cheng and K. Xu

Abstract: In the paper, a new framework of active queue management, namely, edge-based active queue management (EAQM), is proposed. Conventional AQM schemes are required to be deployed at all routers in the network, and this involves significant upgrading to current drop-tail routers. The new approach only needs to modify the edge routers and at the same time provides similar or better performance as compared to conventional AQM schemes. Furthermore, EAQM can reduce the throughput bias against TCP connections with longer round trip times, and this cannot be achieved by current available AQM approaches.

1 Introduction

Active queue management (AQM) has been a very active research area in recent years. Many AQM mechanisms have been proposed, e.g., random early detection (RED) [1], random exponential marking (REM) [2], PI controller [3], adaptive virtual queue (AVQ) [4], state feedback control (SFC) [5], and virtual queue and rate-based scheme (VQR) [6]. The main goals of these schemes are to reduce and stabilise queuing delay, avoid global synchronisation, and achieve high link utilisation. RED is a standard queue length based AQM that drops packets with probability proportional to the current average queue length. AVQ is a typical rate based scheme. REM, PI, and SFC use the queue length and the aggregate flow rate to compute the dropping probability. VQR uses both the virtual queue length and aggregate flow rate to determine the dropping probability.

AQM schemes have not been widely deployed in the current Internet because all the proposed AQM schemes need to be implemented in all routers (or at least bottleneck routers), and this demands significant upgrading to routers in the current networks. Another reason is that the performance of these AQM schemes have not been fully investigated and understood, and it is not clear which scheme performs the best. Therefore, it is still risky to deploy them in the entire network.

In this paper, we propose the framework of edge-based AQM (EAQM). EAQM is only deployed at the network edge, and the drop-tail core routers are kept unchanged. Compared with traditional AQM schemes, it is able to provide comparable or even better performance; thus, it is more practical and economically feasible. It is well known that TCP's throughput is inversely proportional to the round trip time (RTT); considerable unfairness occurs to TCP connections with longer RTTs. As one of the key features, EAQM alleviates this type of unfairness, while current available AQM schemes fail to do so.

© IEE, 2006
IEE Proceedings online no. 20050312
doi:10.1049/ip-com:20050312
Paper received 27th July 2004

The authors are with Advanced Networking Lab, Electrical and Computer Engineering Department, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA
E-mail: nirwan.ansari@njit.edu

2 Framework of EAQM

In this paper, we use the term 'aggregate' to refer to all the TCP connections, which enter the network at the same ingress node and leave the network at the same egress node. Note that any edge node can be the ingress or egress node for multiple aggregates at the same time, and each aggregate can be identified with one pair of ingress and egress nodes uniquely. Each pair of ingress and egress nodes in each aggregate exchange information about the aggregate membership and network congestion condition by sending feedback packets to each other. At ingress nodes, packets are classified into aggregates. Based on the received congestion information, ingress nodes compute dropping probabilities for the corresponding aggregates, and drop packets based on the computed probabilities. The rest of this Section will explain the feedback protocol between ingress and egress nodes, operations at ingress and egress routers, and how the dropping probabilities are computed.

We adopt a feedback protocol between ingress and egress nodes similar to [7]. There are two types of feedback packets: forward packets, which are sent from ingress nodes to egress nodes, and backward packets, which are sent from egress nodes to ingress nodes. Both types of packets contain a timestamp field and aggregate information field. The timestamp field serves to calculate the RTTs between the two edge nodes, and the one-way queueing delay from the ingress node to the egress node. Based on the variation of RTTs, ingress nodes calculate the dropping probabilities. Each ingress node fills in the aggregate information field with necessary information about the TCP flows that enter the network through itself. This allows the egress nodes to know at which ingress node each TCP flow enters the network. Similarly, the aggregate information field in backward packets enables ingress nodes to know at which egress node each TCP flow leaves the network. Based on the information in backward packets, ingress nodes are able to know to which aggregate each TCP flow belongs, and this allows ingress nodes to classify TCP flows into the correct aggregates.

In order to calculate RTTs and one-way queuing delays, ingress nodes continue to send out forward packets after a certain time interval T. In this paper, we set T for each aggregate to be 100 ms. The timestamp field in the forward packets is always available, but the aggregate information field is optional. The aggregate information field is needed only when a new flow enters the network through an ingress node, or one existing flow becomes inactive. When an egress node receives one forward packet, it immediately sends a backward packet back to the corresponding ingress node. The backward packet contains the original timestamp of the forward packet, and the time the corresponding forward packet was received. Again, the aggregate information field of the backward packet is not always required, and it is filled in only when the egress node detects new flows or an existing flow becomes inactive. Figure 1 shows the architecture of the ingress node in EAQM.

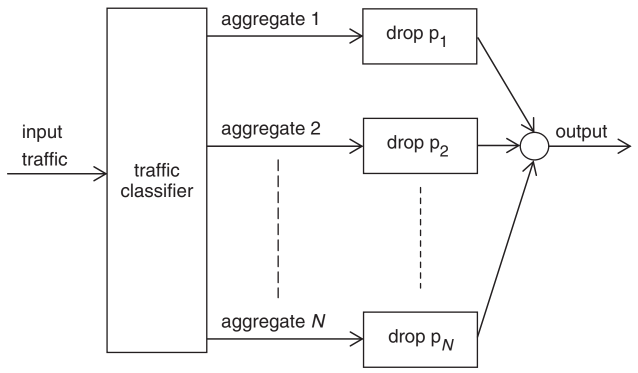

Fig. 1 *Architecture of the ingress node in EAQM*

Upon arrival of a backward packet, each ingress node uses its timestamp field to compute the RTT and one-way queuing delay q for the corresponding aggregate. Then, the average RTT is updated

$$RTT_{ave} \leftarrow (1 - w_{RTT})RTT_{ave} + w_{RTT}RTT \tag{1}$$

where w_{RTT} is the smoothing factor. One method to compute the one-way queuing delay q is to use *BaseRTT* to track the minimum RTT, and compute q as $q = RTT - BaseRTT$. This approach is implemented in the current TCP Vegas, but the performance degrades significantly when there is congestion in the reverse path from the egress node to the ingress node. This disadvantage was pointed out and the remedy was proposed in [8]. In this paper, we adopt the same approach in [8] to compute the one-way queuing delay, and interested readers are referred to [8] for more details.

In EAQM, one simple approach to compute the dropping probability p is

$$p \propto q_{ave} \tag{2}$$

where q_{ave} is the average one-way queuing delay. There is similarity between this approach and RED. In RED, p is a linear function of the queue length at each router, and (2) indicates that p is proportional to the average one-way queuing delay q. The throughput of TCP can be expressed as [9]

$$R = \frac{1}{d\sqrt{\frac{2p}{3}} + RTO \min\left(1, 3\sqrt{\frac{3p}{8}}\right) p(1+32p^2)} \tag{3}$$

where d is the round trip time between the TCP sender and receiver, and RTO is the retransmission timeout, which is set to $4d$ in this paper. Note that d is slightly different from RTT_{ave}, which is the round trip time between the two edge routers of the corresponding aggregate. When the dropping probability p is very small, (3) is reduced to

$$R = \frac{1}{d}\sqrt{\frac{3}{2p}} \tag{4}$$

We can see that there is considerable unfairness against TCP flows with longer round trip time from (3) and (4). Our approach is based on the following intuition: if the dropping probability is computed as a function of d so that d can be cancelled in the right-hand side of (3), the TCP throughput will no longer depend on d. Therefore, we adopt the following way to compute the dropping probability p such that

$$\sqrt{\frac{1}{\beta q_{ave}}} = \frac{1}{RTT_{ave}\sqrt{\frac{2p}{3}} + 4RTT_{ave}\min\left(1, 3\sqrt{\frac{3p}{8}}\right) p(1+32p^2)} \tag{5}$$

is satisfied, and thus eliminating the dependence on d. Here, β is a constant. We can compute the dropping probability p by solving (5). In (5), we use RTT_{ave} rather than d, since only RTT_{ave} can be measured by the edge routers. Note that if $RTT_{ave} = d$, by substituting (5) into (3), the throughput of TCP flows equal to $\sqrt{(1/(\beta q_{ave}))}$, which is independent of their round trip time. Thus, the unfairness is eliminated. Since no explicit expression of p from (5) is available, p has to be numerically determined, thus leading to considerable computing overhead at each ingress node. Hence, we adopt the following approximation by replacing $\min(1, 3\sqrt{(3p/8)})$ with $3(\sqrt{(3p/8)})$ and neglecting the p^2 term in (5)

$$\sqrt{\frac{1}{\beta q_{ave}}} = \frac{1}{RTT_{ave}\sqrt{\frac{2p}{3}} + 12RTT_{ave}\sqrt{\frac{3p}{8}}\,p} \tag{6}$$

Thus, (6) can be reduced to a cubic equation of p, and p is set to be the unique positive root of (6).

When p is very small, (6) can be simplified to

$$p = \frac{3\beta q_{ave}}{2RTT_{ave}^2} \tag{7}$$

Substituting it into (4)

$$R = \sqrt{\frac{1}{\beta q_{ave}}} \times \frac{RTT_{ave}}{d} \tag{8}$$

We make a reasonable assumption that the propagation delay between TCP senders and the ingress node, and that between the egress node and TCP receivers are much smaller than the round trip time (RTT_{ave}) between the two edge nodes. Therefore, we have $RTT_{ave}/d \approx 1$, and (8) can be rewritten as

$$R \approx \sqrt{\frac{1}{\beta q_{ave}}} \tag{9}$$

Thus, the throughput unfairness against TCP flows with longer round trip times is eliminated. We will use simulations to demonstrate this improvement in the next Section.

In the next Section, we will study the performance of EAQM and compare it with RED by simulations. The reasons that we decide to compare EAQM with RED are twofold: first, RED is one of the most classic and well studied AQM schemes; second, both RED and EAQM are core-stateless AQM schemes, and it is thus appropriate to compare their performances. We will show that EAQM can provide similar or even better performance as compared to RED in terms of queue length stability, responsiveness to traffic dynamics, and TCP throughput fairness. There are other possible ways in EAQM to compute the dropping probability other than using (6), e.g., using both queuing delay q and its changing rate $\dot{q}$. The focus of this paper is the introduction of the EAQM framework, which can potentially be adopted in enhancing current Internet, and will prompt more relevant follow-up research activities.

3 Simulations

A simple network shown in Fig. 2 is adopted in this Section for simulations. The link from node N_1 to N_2 is the only bottleneck. There are two TCP aggregates, and each consists of 30 long-lived TCP flows: the first aggregate is from S_1 to R_1, and the second aggregate is from S_2 to R_2. E_1, and E_2 are ingress nodes, and E_3 and E_4 are egress nodes. TCP packet size is 500 bytes, and the buffer size at N_1 is 800. All simulations are 200 seconds long, and the throughputs are computed based on the last 100 seconds of the simulations.

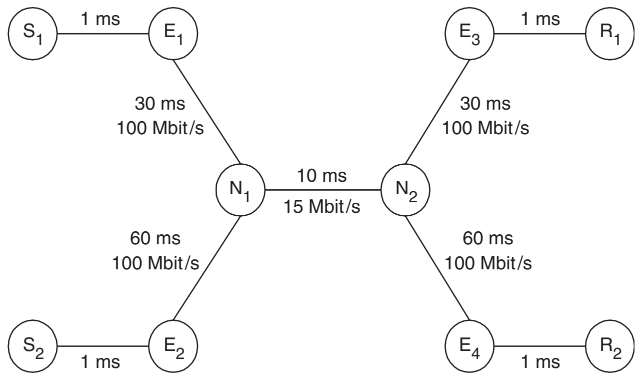

Fig. 2 *Single bottleneck link network*

In the Appendix, we provide the guideline to set the parameters of EAQM to ensure stability. Here, the parameters of EAQM are: $\beta = 0.015$, $w_{RTT} = 0.01$, and $T = 0.1$ s. We follow the guideline in [10] to set the parameters of RED to ensure it is stable: $min_{th} = 50$ packets, $max_{th} = 500$ packets, $p_{\max} = 0.1$, and the weight w for queue averaging is 10^{-6}. All links except the bottleneck link are drop-tails. In all simulations, we use RED at the bottleneck link from N_1 to N_2 first; then we enable EAQM at ingress and egress nodes ($E_1 \ldots E_4$) and the bottleneck link from N_1 to N_2 is set back to simple drop-tail.

In the first set of simulations, we compare the performance between RED and EAQM with respect to the queue length stability. The evolution of the queue length at the bottleneck N_1 is plotted in Fig. 3. We can see that the instantaneous queue length under EAQM oscillates less and is more stable.

In the second set of simulations, we investigate the responsiveness of EAQM to the traffic dynamics. The propagation delay between E_2 and N_1 and that between N_2 and E_4 are changed to 30 ms. At time zero, 30 TCP connections start in the first aggregate, and no traffics are in the second aggregate. At the 100th second, 30 TCP connections start in the second aggregate. The evolution of the queue length is plotted in Fig. 4. It is clear that the instantaneous queue length under RED fluctuates significantly between the 100th and 140th seconds, while EAQM is much less vulnerable to the traffic dynamics.

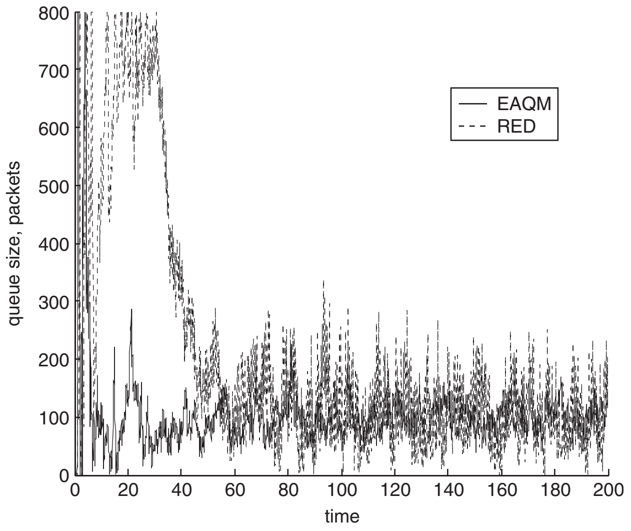

Fig. 3 *Queue stability of EAQM*

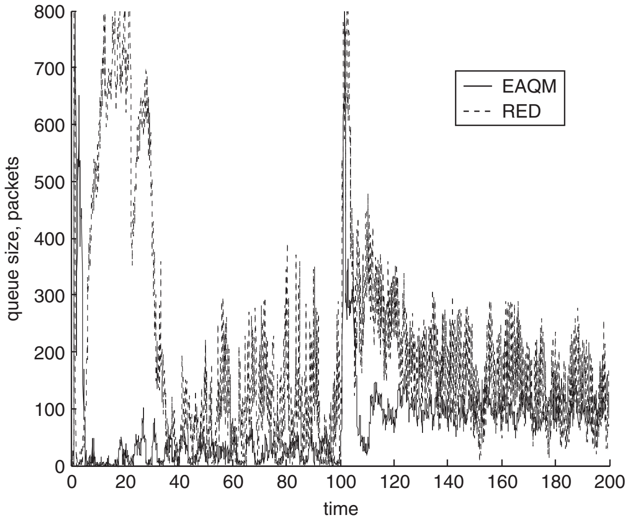

Fig. 4 *Responsiveness to traffic dynamics of EAQM*

In the third set of simulations, we want to demonstrate that EAQM is immune from the throughput bias against TCP flows with longer round trip times. We keep the propagation delay between E_1 and N_1 identical to that between N_2 and E_3, and change their values from 10 ms to 60 ms. We calculate the fairness indices for both RED and EAQM schemes. The fairness index F is defined as

$$F = \frac{\min(r_1, r_2)}{\max(r_1, r_2)} \tag{10}$$

where r_1 and r_2 are the throughput of aggregate one and two, respectively. The closer the index to one, the fairer among all TCP flows. The results are shown in Fig. 5. We can see that EAQM greatly reduces the unfairness.

We also conduct simulations based on the network with two bottleneck links shown in Fig. 6. Each of S_1, S_2, and S_3 represents 30 TCP senders, and R_1, R_2, and R_3 are the corresponding TCP receivers. The link between N_1 and N_2 and the link between N_1 and N_2 are the bottlenecks. We use RED at the bottlenecks first; then we enable EAQM at ingress and egress nodes ($E_1 \ldots E_5$) and the bottleneck links are set back to simple drop-tail. All the RED and EAQM

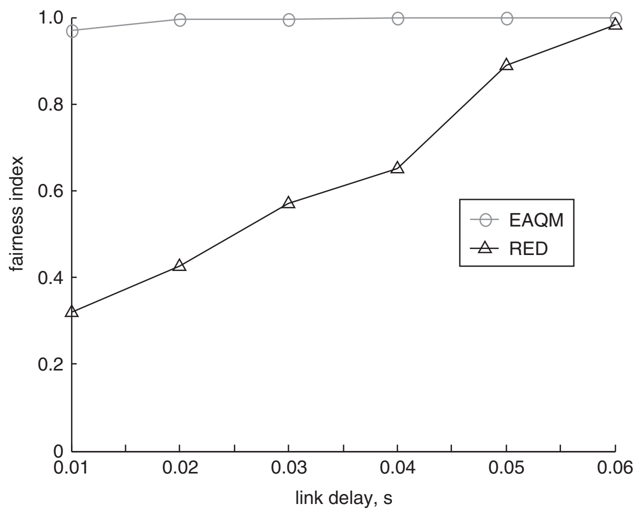

Fig. 5 *Fairness index in a single bottleneck link*

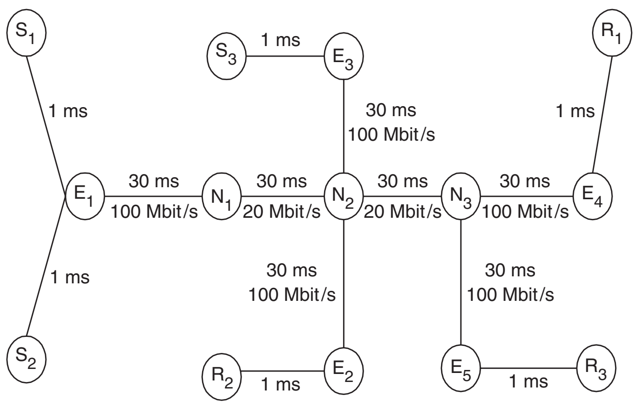

Fig. 6 *Multiple bottleneck link network*

parameters are the same as before. We keep the propagation delay between E_1 and N_1 identical to that between N_2 and E_3, and change their values from 10 ms to 60 ms. We calculate the fairness indices for both the RED and EAQM schemes. The fairness index F is defined as

$$F = \frac{\min(r_1, r_2, r_3)}{\max(r_1, r_2, r_3)} \tag{11}$$

where r_1, r_2, r_3 are the throughput of aggregate one, two and three, respectively. The results are shown in Fig. 7.

Again, it is clear that the fairness index of EAQM is higher than that of RED. We can see that the fairness index of EAQM in Fig. 7 is around 0.7 instead of 1.0 as in Fig. 5. This is not because that EAQM does not eliminate the throughput unfairness against TCP flows with longer RTTs. It is because that aggregate one (from S_1 to N_1) traverses two bottleneck links and each of the remaining two aggregates only traverses one bottleneck link. Thus, aggregate one experiences more queuing delay, and as a result, exhibits a larger dropping probability than that of aggregate two and three. Owing to the symmetry, the average queue length at each bottleneck, denoted as Q_{ave}, is equal to each other. From (9), the throughput of aggregate one is

$$R_{1,eaqm} = \sqrt{\frac{1}{2\beta Q_{ave}}} \tag{12}$$

and the throughput of aggregate two or three is

$$R_{2,eaqm} = R_{3,eaqm} = \sqrt{\frac{1}{\beta Q_{ave}}} \tag{13}$$

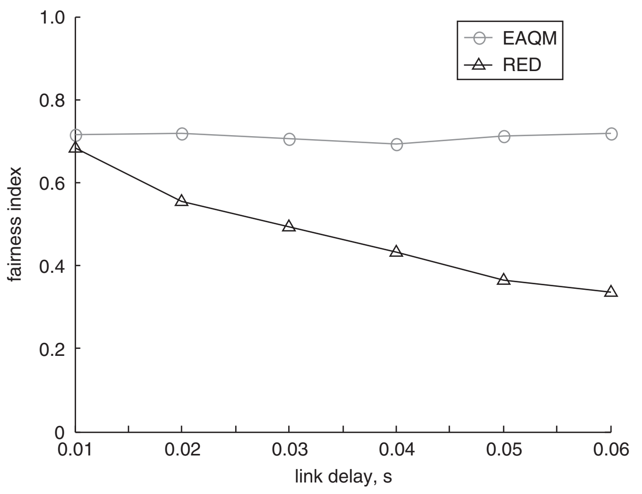

Fig. 7 *Fairness index in multiple bottleneck link network*

From (12) and (13), the fairness index for EAQM is

$$F_{eaqm} = \frac{R_{1,eaqm}}{R_{2,eaqm}} = \frac{1}{\sqrt{2}} \approx 0.707 \tag{14}$$

and this is very close to the result in Fig. 7.

A similar situation occurs if RED is adopted. Owing to the symmetry, the equilibrium dropping probabilities at all bottleneck links are the same, and we denote them as P_{ave}. Thus, the dropping probability for aggregate one is $2P_{ave}$, and P_{ave} for aggregate two and three. From (4)

$$R_{1,red} = \frac{1}{d_1}\sqrt{\frac{3}{4P_{ave}}} \tag{15}$$

$$R_{2,red} = \frac{1}{d_2}\sqrt{\frac{3}{2P_{ave}}} = R_{3,red} = \frac{1}{d_3}\sqrt{\frac{3}{2P_{ave}}} \tag{16}$$

Here, d_1, d_2 and $d_3 = d_2 \le d_1$ are the RTTs for aggregate one, two and three, respectively. The fairness index is

$$F_{red} = \frac{R_{1,red}}{R_{2,red}} = \frac{d_2}{d_1\sqrt{2}} \approx 0.707\frac{d_2}{d_1} \tag{17}$$

Similarly, it can be shown that the factor $1/\sqrt{2}$ also exists under other traditional AQM schemes.

4 Conclusions

There exist some other AQM schemes such as flow random early drop (FRED) [11] and core-stateless fair queuing (CSFQ) [12], which address the fairness issue. However, all of them require the deployment at both edge routers and core routers. FRED is a core-stateful rather than a core-stateless approach. In FRED, core routers need to maintain $qlen_i$ and $strike_i$ for every flow i. This drawback makes it very difficult to implement FRED in the core routers. CSFQ needs extra field in the IP header to encode traffic rate of each flow and other necessary information, which requires the introduction of new protocols.

In this paper, we have proposed a new framework of AQM, namely, EAQM, which is only required to be deployed at the edge of the network. It has been shown that EAQM can achieve similar or better performance as compared to the typical AQM scheme, RED. Furthermore, EAQM can reduce the throughput bias against TCP connections with longer round trip times. EAQM is a more practical and economical approach to improve the current Internet.

5 Acknowledgments

This work has been supported in part by the New Jersey Commission on Higher Education via the NJI-TOWER project, and the New Jersey Commission on Science and Technology via NJWINS.

6 References

1 Floyd, S., and Jacobson, V.: 'Random early detection gateways for congestion avoidance', *IEEE/ACM Trans. Netw.*, 1993, **1**, (4), pp. 397–413
2 Athuraliya, S., Low, S.H., Li, V.H., and Yin, Q.: 'REM: active queue management', *IEEE Netw. Mag.*, 2001, **15**, (3), pp. 48–53
3 Hollot, C.C., Misra, V., Toqaley, D., and Gong, W.: 'On designing improved controllers for AQM routers supporting TCP flows'. Proc. IEEE INFOCOM 2001, Alaska, USA, April 2001, Vol. 3, pp. 1726–1734
4 Kunniyur, S., and Srikant, R.: 'Analysis and design of an active virtual queue (AVQ) algorithm for active queue management'. Proc. ACM SIGCOMM 2001, San Diego, USA, August 2001, pp. 123–134
5 Gao, Y., and Hou, J.C.: 'A state feedback control approach to stabilizing queues for ECN-enabled TCP connections'. Proc. IEEE INFOCOM 2003, San Diego, USA, April 2003, Vol. 3, pp. 2301–2311
6 Zhu, L., and Ansari, N.: 'Local stability of a new adaptive queue management (AQM) scheme', *IEEE Commun. Lett.*, 2004, **8**, (6), pp. 406–409
7 Albuquerque, C., Vickers, B.J., and Suda, T.: 'Network border patrol: preventing congestion collapse and promoting fairness in the internet', *IEEE/ACM Trans. Netw.*, 2004, **1**, (1), pp. 173–186
8 Chan, Y.C., Chan, C.T., and Chen, Y.C.: 'An enhanced congestion avoidance mechanism for TCP vegas', *IEEE Commun. Lett.*, 2003, **7**, (7), pp. 343–345
9 Padhye, J., Firoiu, V., Towsley, D., and Kurose, J.: 'Modeling TCP throughput: a simple model and its empirical validation'. Proc. ACM SIGCOMM 1998, Canada, 1998, pp. 303–314
10 Hollot, C.C., Misra, V., Toqaley, D., and Gong, W.: 'A control theoretical analysis of RED'. Proc. IEEE INFOCOM 2001, Alaska, USA, April 2001, Vol. 3, pp. 1510–1519
11 Lin, D., and Morris, R.: 'Dynamics of random early detection'. Proc. ACM SIGCOMM 1997, Palais des festivals, Cannes, France, September 1997, pp. 127–137
12 Stoica, I., Shenker, S., and Zhang, H.: 'Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high speed networks'. Proc. ACM SIGCOMM 1998, Vancouver, Canada, September 1997, pp. 118–130

7 Appendix

We follow the similar procedures and the criteria in [10] to set the EAQM parameters β, w_{RTT} and T. Consider N TCP connections with propagation delay d_0 in a single bottleneck network. The linearised dynamic model in the Laplacian domain is illustrated in Fig. 8. Readers are referred to [3, 10] for more details on this model.

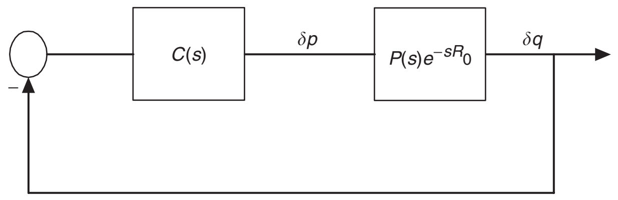

Fig. 8 *Linearised model of TCP/AQM*

In Fig. 8, $P(s)$ is expressed as

$$P(s) = \frac{1}{C} P_{tcp}(s) P_{queue}(s) \tag{18}$$

where

$$P_{tcp}(s) = \frac{\dfrac{R_{ave}C^2}{2N^2}}{s + \dfrac{2N}{R_{ave}^2 C}} \tag{19}$$

and

$$P_{queue}(s) = \frac{\dfrac{N}{R_{ave}}}{s + \dfrac{1}{R_{ave}}} \tag{20}$$

Here, $P_{tcp}(s)$ and $P_{queue}(s)$ represent the dynamics of TCP and queue, respectively. Note that there is an additional term $1/C$ in (18) compared to that in [10], and this is because q in this paper is the queuing delay instead of queue length used in [10]. Here, C is the capacity of the link. From (7)

$$p = \frac{3\beta q_{ave}}{2RTT_{ave}^2}$$

we have [10]

$$\begin{aligned}\delta p &= \frac{3}{2RTT_{ave}^2}\left(1 - \frac{2q_{ave}}{RTT_{ave}}\right)\delta q_{ave} \\ &= \frac{3}{2RTT_{ave}^2}\left(1 - \frac{2q_{ave}}{RTT_{ave}}\right)\frac{1}{\dfrac{s}{K}+1}\delta q \end{aligned} \tag{21}$$

where $K = -\ln(1 - w_{RTT})/T$, w_{RTT} is a smoothing factor for RTT, and T is the time interval, after each of which the ingress nodes send out forward packets to probe the one-way queuing delay of each aggregate. Combining (18)–(21), the frequency response for the open loop transfer function is

$$L(j\omega) = \frac{3\beta RTT_{ave}\left(\dfrac{C}{2N}\right)^2\left(1 - \dfrac{2q_{ave}}{RTT_{ave}}\right)e^{-j\omega R_{ave}}}{2\left(\dfrac{j\omega}{K}+1\right)\left(\dfrac{j\omega RTT_{ave}^2}{2N}+1\right)(j\omega RTT_{ave}+1)} \tag{22}$$

According to Nyquist Theorem [10], the above system is stable if $L(j\omega)$ crosses the real axis on the right hand side of -1. Using the results of Proposition 1 and 2 in [10], we have

Proposition 1: If

$$\frac{3}{2}\beta R^{+}\left(\frac{C}{2N^{-}}\right)^2 \leq \sqrt{\frac{\omega_g^2}{K^2}+1} \tag{23}$$

where $\omega_g = 0.1\min\{2N^{-}/(R^{+})^2C, 1/R^{+}\}$, then the linearised feedback system is stable for $N \geq N^{-}$ and $R_{ave} \leq R^{+}$. Furthermore, the above feedback system is also stable if either $RTT_{ave} < 15R^{+}$ or $N > (1/5\pi)N^{-}$.

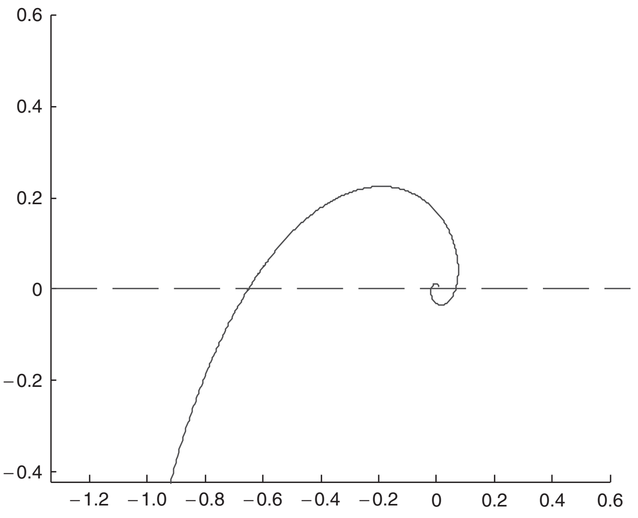

Fig. 9 *Nyquist plot of $L(j\omega)$ for $N = 60$, $RTT_{ave} = 0.6$ s*

The proof of Proposition 1 is similar to the proofs of Proposition 1 and 2 in [10], and it is thus omitted here. Interested readers are referred to [10] for details.

In all simulations in this paper, we use $N \geq 60$, and $RTT_{ave} \leq 0.6\,\text{s}$. Let $N^- = 60$, $R^+ = 0.04\,\text{s}$, $\beta = 0.015$, $w_{RTT} = 0.01$, and $T = 0.1\,\text{s}$. Using Proposition 1, we can verify that the linearised EAQM system is stable with these parameter settings. We also show the Nyquist plot of $L(j\omega)$ with these parameter settings in Fig. 9. We can see that $L(j\omega)$ crosses the real axis on the right hand side of -1. Therefore, stability is guaranteed.