Design and Analysis of a Rate-based Algorithm for Active Queue Management

Chonggang Wang
University of Arkansas
Fayetteville, AR, U.S.A.
E-mail: cgwang@uark.edu

Bo Li
The Hong Kong Univ. of Sci. & Tech.
Hong Kong, China
E-mail: bli@cs.ust.hk

Y. Thomas Hou
Virginia Tech
Blacksburg, VA, U.S.A.
E-mail: thou@vt.edu

Kazem Sohraby
University of Arkansas
Fayetteville, AR, U.S.A.
E-mail: sohraby@uark.edu

Weiwen Tang
Sichuan Communication Research Planning
& Designing Co., Ltd., Chengdu, China
E-mail: tangww@sctele.com

Abstract

This paper proposes a rate-based active queue management algorithm or RAQM. It uses the aggregated traffic input rate to calculate packet drop probability according to an exponential rule. We analyze the stability and investigate practical implementation issues of the RAQM. Simulations are carried out to study RAQM performance and to compare with other AQM algorithms, in particular PI and REM schemes. The results demonstrate that RAQM achieves better stability and faster response as it can quickly regulate the queue length to the expected value with small overshoot. RAQM also obtains better tradeoff between link utilization and queuing delay, and obtains higher goodput with the same buffer size as in PI and REM schemes. Finally RAQM has O(1) complexity, thus independent of the number of flows.

1. Introdoction

Buffer management plays an important role in congestion control, whose primary objectives are high link utilization and low packet queuing delay. However, these two objectives generally conflict, since small buffer leads to lower queuing delay but also lower link utilization, and vice versa. Therefore, one of the key design issues in the buffer management is to strike a good tradeoff between them.

This work was supported in part by grants from Research Grants Council (RGC) under contracts HKUST6196/02E and HKUST6204/03E, a grant from NSFC/RGC under contract N_HKUST605/02, and a grant from Microsoft Research under contract MCCL02/03.EG01. Please send all correspondences to Bo Li at bli@cs.ust.hk, or Fax: +852 2358 1477.

The traditional algorithm for buffer management is First-In-First-Out (FIFO) with Tail-Drop, which drops packets only if buffer overflows. This passive behavior results in long queuing delay, and is often the cause of the correlation among packet drops, resulting in the well-known "TCP synchronization" problem [1]. That is, many TCP flows may lose packets and decrease sending rate simultaneously. To mitigate such problems, AQM [1] has been introduced in recent years. Comparing with FIOF Tail-Drop, AQM algorithms can actively trigger packet drops before buffer overflows, which can avoid "TCP synchronization" and achieve better link utilization and reduce the possibility of congestion. The primary objectives of AQM include: 1) High link utility and high goodput (exclude the duplicate packets received). On one hand, buffer emptiness should be avoided; on the other hand, it is also necessary to prevent buffer overflow or longer queue length. Otherwise, the timeout of TCP flows would occur frequently, resulting in unnecessary retransmissions, thus low goodput for them. 2) Low queuing delay. This is helpful to prevent TCP timeout and also attractive for the ever-increasing real-time applications. 3) Simple and efficient. AQM must be simple and scalable to be deployed in high-speed routers. 4) Stability and robustness. AQM should be stable and robust under dynamic environments, *i.e.*, retain good performance even if the network parameters, such as the number of flows *N*, round-trip time *RTT*, as well as the characteristics of the flows, such as short-lived or long-lived, and responsive or unresponsive, change frequently without being known *a priori*.

The key issues in the design of AQM algorithms are how to measure link-congestion degree and how to determine the behavior of packet dropping (or mark if explicit congestion notification (ECN) [2] is enabled [1]. In the existing AQM algorithms, link-congestion can be estimated through (average) queue length, input rate, event of buffer overflow or buffer empty, or and combination of them. Among them, queue length is the most widely used index, e.g., in the original RED [3][4] and most of its variants, such as S-RED [5] and ARED [6]. The event of buffer overflow or buffer empty is used in BLUE [7] to estimate link-congestion level, and the input rate is used in AVQ [8]. Recent algorithms, such as PI [9], REM [10], and SFC [11], jointly use queue length and input rate to estimate link-congestion.

The motivation in this paper is to design a more stable algorithm with faster response or convergence rate since the traffics in Internet may be often varied and hard to be known exactly, and slow response will lead to buffer overflow or emptiness and corresponding large queuing delay and low link utility [12]. Also in order to flexibly adjust queuing delay, AQM algorithms should be capable to effectively regulate the queue length to an expected value. The main contribution in this paper is the proposed rate-based algorithm or RAQM, which obtains good stability and fast response under diverse network environment. RAQM can regulate the queue length to an expected value and achieve a better tradeoff between goodput and queuing delay. RAQM has two modes of operations, *queue-independent* and *queue-dependent modes.* In the *queue-independent* mode, RAQM periodically measures aggregated traffic input rate, and iteratively computes the packet drop probability, so as to make the input rate close to the expected value r_0. This mode does not use any information on the queue length. In the *queue-dependent* mode, RAQM also uses instantaneous queue length to further adjust packet drop probability besides the traffic input rate, and can regulate queue length to the expected value q_0. The stability analysis and design rules for RAQM are also given out in this paper, which jointly with extensive simulations demonstrate the advantages of RAQM.

The rest of this paper is organized as follows. Section II presents RAQM algorithm, stability analysis and discussions on the implementation. Section III presents the numerical results obtained using simulation and comparison with other AQM algorithms. Section IV concludes the paper and highlights several possible extensions.

2. RAQM algorithm

RAQM uses aggregated traffic input rate to regulate it to the expected value r_0. It needs to periodically measure the aggregated traffic input rate r_k, and iteratively updates the packet drop probability p_k according to an *exponential* rule. The rational for using the exponential rule is that it can achieve the stability and leads to faster convergence of the packet drop probability to the expected value. RAQM can also jointly use the instantaneous queue length and further refine the packet drop probability more each time when a new packet arrives, so as to regulate queue length to the expected value q_0.

2.1. Algorithm description

Like most of other AQM algorithms, RAQM also selects the tail-packet (or the arriving packet just now) when it needs to drop packet. In RAQM, packet drop probability is periodically computed according to the measured aggregated traffic input rate r_k. Intuitively when r_k increases, packet drop probability p_k has to be set at a larger value, and vice versa. First, the aggregated traffic input rate r_k is periodically measured using the mechanism proposed in [13] as:

$$r_k = (1 - e^{-T/f})\frac{l_k}{T} + e^{-T/f} r_{k-1} \tag{1}$$

where T is the measure period, l_k is the total bytes arriving during the *k-th* measure period, r_k is the input rate measured in the end of *k-th* measure period, and *f* is a constant that will influence the correctness of measure result.

For simplicity, we can rewrite Eq. (1) as:

$$r_k = (1 - f')\frac{l_k}{T} + f' r_{k-1} \tag{2}$$

where $f'(= e^{-T/f} < 1)$ is a constant.

After obtaining the aggregated traffic input rate r_k, packer drop probability p_k in RAQM is periodically and iteratively calculated at the end of *k-th* period using the exponential rule as:

$$p_k = e^{\alpha(r_k - r_0)} p_{k-1} \tag{3}$$

where r_0 is the expected traffic input rate, and $\alpha > 0$ is a parameter to be configured. r_0 can be set as the link capacity C or γC (where $0 < \gamma \le 1$). α needs to be carefully considered since it determines the stability and response time of RAQM. For example, if α is too large, p_k will oscillate. On the contrary, p_k will adapt slowly if α is too small. This will be analyzed in the next sub-section.

[1] For easy of exposition, we use "drop" for both cases in the rest of this paper, unless explicitly indicated.

2.2. Performance Analysis

This sub-section analyzes the stability of RAQM and determines how to configure α. Assume that there are N long-lived TCP flows, and each flow i has round-trip time d_i, packet size L_i, and throughput r_k^i. According to the TCP throughput model deduced in [14], the relationship between the aggregated traffic input rate r_k and packet drop probability p_k can be represented as:

$$r_k^i = \frac{\eta \times L_i}{d_i \times \sqrt{p_{k-1}}} \tag{4}$$

$$r_k = \sum_{i=1}^{N} r_k^i = \frac{\eta}{\sqrt{p_{k-1}}} \sum_{i=1}^{N} \frac{L_i}{d_i} = \frac{\eta H}{\sqrt{p_{k-1}}} \tag{5}$$

where η is a constant, and $H = \sum_{i=1}^{N} (L_i / d_i)$. From Eq. (5), we have:

$$\frac{r_k}{r_{k-1}} = \sqrt{\frac{p_{k-2}}{p_{k-1}}} \tag{6}$$

Lemma 1: The expected traffic input rate r_0 is one of the stable points for the system defined by Eqs. (3)-(5).

Proof: assume $r_k = r_0$. From Eq. (3), we can have $p_k = p_{k-1}$. Then according to Eq. (6), there is: $r_{k+1} = r_k = r_0$ Q.E.D

Lemma 2: The stable point, for the system defined by Eqs. (3)-(5), is unique.

Proof: assume there is another stable point r'. Let $r_{k+1} = r_k = r'$. From Eq. (10), we have $p_k = p_{k-1}$. Then it is sure to be $r' = r_0$, according to Eq. (3).

Q.E.D

Theorem 1: RAQM is stable iff $0 < \alpha < \hat{\alpha}$, where

$$\hat{\alpha} = \frac{\ln(2r_0 - r_k) - \ln r_k}{0.5(r_0 - r_k)}$$

Proof: using Lyapunov Theorem. Firstly, define Lyapunov function as:

$$V_k = (r_k - r_0)^2 \tag{7}$$

Then its discrete differentiated equation can be written as:

$$V_{k+1} - V_k = (r_{k+1} - r_k)(r_{k+1} + r_k - 2r_0) \tag{8}$$

From Eqs. (3) and (6), we can have:

$$\frac{r_{k+1}}{r_k} = e^{-0.5\alpha(r_k - r_0)} \tag{9}$$

Substituting Eq. (9) into Eq. (8), the differentiated equation $V_{k+1} - V_k$ can be calculated as:

$$V_{k+1} - V_k = (r_{k+1} - r_k)(r_k(e^{-0.5\alpha(r_k - r_0)} + 1) - 2r_0) \tag{10}$$

Because of $V_k > 0$, we only need to prove that its differentiated equation Eq. (10) meets: $V_{k+1} - V_k < 0$. Then RAQM (or Eq. (3)) is sure to be stable according to Lyapunov theorem.

Let $V_{k+1} - V_k = 0$. Assume $r_0 \neq r_k$ and $2r_0 - r_k > 0$, then:

$$\alpha = \frac{\ln(2r_0 - r_k) - \ln r_k}{0.5(r_0 - r_k)} = \hat{\alpha} \tag{11}$$

If $2r_0 - r_k > 0$ and $r_0 \neq r_k$, it can be easily proved that $\hat{\alpha} > 0$ whenever $r_0 > r_k$ or $r_0 < r_k$.

Case1: $r_k > r_0$. From Eq. (9), we can get $r_{k+1} < r_k$. According to Eqs. (10) and (11), there are:

If
$0 < \alpha < \hat{\alpha} \Rightarrow r_k(e^{-0.5\alpha(r_k - r_0)} + 1) - 2r_0 > 0 \Rightarrow V_{k+1} - V_k < 0$;
If
$\alpha > \hat{\alpha} > 0 \Rightarrow r_k(e^{-0.5\alpha(r_k - r_0)} + 1) - 2r_0 < 0 \Rightarrow V_{k+1} - V_k > 0$.

Case2: $r_k < r_0$. From Eq. (9), we can get $r_{k+1} > r_k$. According to Eqs. (10) and (11), there are:

If
$0 < \alpha < \hat{\alpha} \Rightarrow r_k(e^{-0.5\alpha(r_k - r_0)} + 1) - 2r_0 < 0 \Rightarrow V_{k+1} - V_k < 0$;
If
$\alpha > \hat{\alpha} > 0 \Rightarrow r_k(e^{-0.5\alpha(r_k - r_0)} + 1) - 2r_0 > 0 \Rightarrow V_{k+1} - V_k > 0$.

Combining the above discussions and the fact that $V_k > 0$, we conclude: 1) If $0 < \alpha < \hat{\alpha}$, then $V_{k+1} - V_k < 0$, and RAQM is stable; 2) If $\alpha > \hat{\alpha} > 0$, then $V_{k+1} - V_k > 0$, RAQM becomes unstable; 3) If $\alpha = \hat{\alpha}$, then $V_{k+1} - V_k = 0$, we can't determine the stability of RAQM according to Lyapunov function in Eq. (7).

Q.E.D

Theorem 2: The convergence rate of RAQM is determined by value of α. The bigger value of α, the faster convergence rate will be obtained.

Proof: it can be seen from proof of Theorem 1 that the absolute value of $V_{k+1} - V_k$ will decrease when α grows. Therefore, the convergence rate of RAQM will increase.

Q.E.D

2.3. Queue-dependent mode

RAQM mode defined by Eq. (3) aims to control the aggregated rate to the expected value r_0. We refer this as *queue-independent* mode. We can further use instantaneous queue length to regulate queue length to the expected value q_0, and in turn to regulate the queuing delay. This is called as *queue-dependent* mode in RAQM, where the packet drop probability p is calculated each time a new packet arrives as:

$$p = \max(0, \min(1, p_k \frac{q}{q_0})) \qquad (12)$$

where q is the instantaneous queue length and p_k is the packet drop probability in *queue-independent mode*, calculated from Eq. (3). Fig. 1 presents the detailed procedures, respectively, for *queue-independent mode* and *queue-dependent mode*.

2.4. Implementation issues

RAQM needs to periodically measure the aggregated traffic input rate, and use it to calculate packet drop probability. The period T can be set as several *RTTs* to track the variation of input rate, since the control cycle in TCP protocol is one *RTT*. This will be verified in the next section through simulations. Another parameter to be configured is α, which not only determines the stability of the algorithm, but also influences the convergence rate. Theorem 1 has deduced the upper bound of α to maintain the stability of RAQM. In fact, $\hat{\alpha}$ also has its lower bound. Assuming $r_k = x.r_0$ ($x \in (0,1) \bigcup (1,2)$), $\hat{\alpha}$ can be simply calculated from Eq. (11) as:

$$\hat{\alpha} = \left(\frac{2\ln((2-x)/x)}{1-x}\right) r_0^{-1} = c(x)\, r_0^{-1} \qquad (13)$$

The dynamics of *c(x)* is presented in Fig. 2. It can be observed and easily proved that: 1) *c(x)>4*, when $x \in (0,1) \bigcup (1,2)$; 2) when x is close to 1, *c(x)* has limited value of 4; 3) when x is close to 0 or 2, *c(x)* has no limited value or its limited value is $+\infty$. Since $c(x) \geq 4$, $\hat{\alpha}$ is not smaller than $4 r_0^{-1}$ according to Eq. (13).

As a result, two ways can be used to configure α : 1) *Static configuration*-We can statically set α in a value that is smaller than $\hat{\alpha}$. In order to meets this condition, this static value needs to be conservatively set using a small value, which will cause slow convergence and response according to Theorem 2. Since $\hat{\alpha}$ has its lower bound ($4 r_0^{-1}$) according to the discussions in the last paragraph, we can set $\alpha = m\, r_0^{-1}$ (0<*m*<4). 2) *Adaptive configuration*-In this way α can be adaptive re-configured with $\hat{\alpha}$, if and only if it is a bit smaller than $\hat{\alpha}$, so as to keep stability and fast convergence simultaneously. We can set $\alpha = \varepsilon \times \hat{\alpha}$ directly using Eq. (11), where ε is smaller than 1 but close to 1.

```
/* Initialization */
dropPb=0.0002;     arrByte=0;      inputRate=0;
/* InputRateMeasure()-Called periodically every T seconds */
        inputRateTemp=arrByte/T;
        inputRate=(1-f') inputRateTemp+f'*inputRate;
        mismatch=inputRate-r0;
        dropPb=max(0, min(1, dropPb*e^mismatch));
        arrByte =0;
/* Enqueue()-Called each time a new packet arrives */
        arrByte+=Length(arrival packet);
        if(queue-independent mode)   p=dropPb;
        if(queue-dependent mode)     p=max(0, min(1, dropPb*(qlen/q0)));
        random=uniformRandom(0, 1);
        if(buffer is full) {
           Drop the packet;
        }  else if(random>p)  {  Enqueue the packet;
        }  else  {
           Drop the packet;
        }
```

Figure 1. RAQM procedures

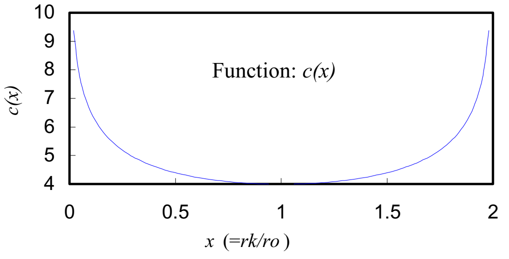


Figure 2. Dynamics of *c(x)* in Eq. (13)

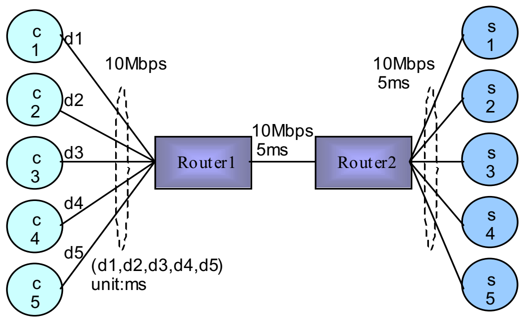


Figure 3. Network topology

3. Simulation results

Among the proposed algorithms, PI [9] and REM [10] have excellent stability property and can effectively regulate queue length to the expected value, like RAQM. SFC [11] can also regulate the queue length to the expected value, but it provides no detailed procedures (for example, the frequency to estimate the traffic rate) to estimate traffic input rate (Fig. 5 in [11]), then it is hard to precisely implement it and make comparisons with RAQM and other algorithms. Therefore we only choose PI and REM in this paper as comparison, and use the default value in NS2 [15] for their parameters[2]. The *queue-dependent* mode of RAQM is implemented and its parameters are set as

[2] In PI, a= 0.00001822, b= 0.00001816, sampling frequency is 170. In REM, $\phi = 1.001$, $\alpha = 0.1$, $\gamma = 0.001$, sampling interval is 2ms.

follows. First, the measure period (T) and the constant f' in Eq. (2) are set at 1.0s and 0.1 respectively. The α in Eq. (3) is calculated using *adaptive configuration* according to Theorem 1, and its initial value is set at $2r_0^{-1}$, r_0 is set to the link capacity. The initial packet drop probability can be very small value (such as 0.0002 in Fig. 1 and the later simulations).

The network topology is the commonly used dumb-bell (see Fig. 3), where there is only a single congestion link lying from Router 1 to Router 2, and capacity of each link is 10Mbps. The link delay (unit in ms) between client *c(i)* and Router 1 is labeled as a 5-tuple $(d_1, d_2, d_3, d_4, d_5)$. All flows are uniformly configured between pairs of client *c(i)* and server *s(i)*, and issue packets from client to server. Packet size is 500 bytes, and buffer size is 100 packets throughout this section. Simulation time is configured at 200 seconds.

Although AQM can integrate with ECN [2] mechanism, this paper does not considers packet marking, but focuses on packet dropping. In simulations, we have collected goodput (exclude the duplicated packets in receivers), instantaneous queue length, and packet drop ratio. We carry out simulations in four cases in order to investigate the performance of RAQM in a variety of environment. Small *RTT* and large *RTT* are configured respectively in case 1 and case 2, where each flow has nearly the same *RTT*. In case 3, the link delay between clients and Router 1 is configured using different value. Therefore flows have diverse and different *RTT*. In case 4, a dynamic network environment is constructed through introducing unresponsive UDP flows and short-lived TCP flows to emulate the real network at the best.

3.1. Case 1-small round-trip time (RTT)

In this case, we configure the topology with small round trip time through setting *di (i=1~5)*=10ms. The expected queue length q_0 for PI, REM, and RAQM is 50 packets. All the traffic flows are long-lived TCP flows. We collect the queue length at Router 1, and present it in the Fig. 4, which shows that RAQM can quickly and effectively regulate the queue length around the expected value q_0. However, PI and REM cause slow response and big overshoot, especially when the number of TCP flows is large (=400). The queue length under RAQM is effectively regulated around q_0 with much shorter overshoot.

3.2. Case 2-large round-trip time (RTT)

In this case, *di (i=1~5)* is set at 100ms, so as to investigate the performance of RAQM under large round trip time. The other parameters are the same to that in case 1. The responding results about queue length are illustrated in Fig. 5, which shows that RAQM still obtains better stability and much faster response than PI and REM, as that under the small round trip time. Moreover, REM is nearly out of control in this case when the number of TCP flows is equal to 40.

3.3. Case 3-diverse round trip time (RTT)

In this case, the link delay between clients and Router 1 are configured at $(d_1, d_2, d_3, d_4, d_5)$=(10, 50, 100, 150, 200) ms. First we also set the expected queue length q_0 at 50 packets and the number of TCP flow is 40 and 400. The queue length is illustrated in Fig. 6 for PI, REM, and RAQM. RAQM still achieves better stability and much faster response than PI and REM, since it is capable to quickly regulate the queue length around q_0 with much shorter overshoot.

Second, we fix the number of long-lived TCP flows at 100, and vary the expected queue length q_0 from 20, to 30, 40, 50, 60, 70, 80, and 90. We collect the sum of goodput at all servers and packet drop ratio at Router 1. Fig. 7 presents the results about goodput and packet drop ratio vs. average queue length, and shows that RAQM can obtain higher goodput and lower packet drop ratio simultaneously at the cost of the same average queue length. Even if the average queue length is about 20, RAQM still obtains high goodput (about 0.96). In PI and REM, when the expected queue length q_0 is bigger than 60, it causes decreased goodput on the contrary like the behavior of traditional Tail-Drop. But this phenomenon is not observed in RAQM and shows that RAQM is capable to more effectively regulate queue length and keep higher goodput even if the expected queue length is close to the buffer size.

3.4. Case 4-hybrid traffics

In this case, hybrid traffics are introduced, such as long-lived TCP flows, short-lived TCP flows, and unresponsive UDP flows. The long-lived TCP flows are active during time interval [0s, 200s], but short-lived TCP and UDP flows become active only during [50s, 150s]. The short-lived TCP is emulated using a Poisson process, and its mean arrival rate equals to 10. The duration of each short-lived TCP flow is uniformly distributed in [1.0s, 2.0s]. The UDP is the kind of ON/OFF CBR flow, with average rate 40Kbps. The duration of ON and OFF state is exponentially distributed with mean value 1.0s. The total number of UDP flow is 50. Other parameters are the same to that in case 3. We collect the queue length for each AQM algorithm of interest (See Fig. 8), and observe that

RAQM still obtains better stability and faster response than PI and REM under hybrid traffics.

4. Conclusions

A new algorithm (RAQM) for active queue management is proposed in this paper, which uses the aggregated traffic input rate to iteratively compute the packet drop probability according to an exponential rule, in order to regulate the input rate to an expected value. RAQM can jointly use instantaneous queue length to adjust packet drop probability and regulate the queue length to its expected value. The stability and design rules of RAQM are analyzed, which with the extensive simulations demonstrate that RAQM achieves better stability and faster response and than PI and REM under diverse environments. RAQM also obtains higher goodput at the same cost of average queue length, and better tradeoff between goodput and queuing delay. Although RAQM needs to measure traffic input rate, yet this operation is for the aggregated flows and there is no need to differentiate each micro flow. It can be executed at scale of second, which is easy for the commercial routers. The total complexity of RAQM is independent of the number of micro-flows and is in the order of $O(1)$.

There are several other issues currently under investigation. This paper only investigates the RAQM performance under single-hop environment. We will evaluate its performance under multi-hop environments such as DiffServ capable networks in our future works. Another issue is priority supporting or bandwidth sharing. If configuring different expected rate value for different sub-aggregated flows and separately computing packet drop probability for them, RAQM can be easily extended to provide bandwidth guarantee for different sub-aggregated TCP flows using a single physical queue.

References

[1] B. Braden, et al., "Recommendations on queue management and congestion avoidance in the Internet," *IETF RFC2309*, Apr. 1998.

[2] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10-23, Oct. 1994.

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

[4] S. Floyd, "Recommendation on using the 'gentle_' variant of RED," http://www.icir.org/floyd/red/gentle.html, Mar. 2000.

[5] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED Gateway," In *Proceedings of IEEE INFOCOMM*, vol. 3, pp. 1320-1328, 1999.

[6] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for increasing the robustness of RED's active queue management," *http://www.icir.org/floyd/papers/adaptiveRed.pdf*, Aug. 2001.

[7] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "The blue active queue management algorithms," *IEEE/ACM Trans. on Networking*, vol. 10, no. 4, pp. 513-528, Aug. 2002.

[8] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," In *Proceedings of ACM SIGCOMM'01*, pp. 123-134, Aug. 2001.

[9] C. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Trans. on Automatic Control*, vol. 47, pp. 945-959, Jun. 2002.

[10] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active queue management," *IEEE Network Magazine*, vol. 15, pp. 48-53, May 2001.

[11] Y. Gao and J. C. Hou, "A state feedback control approach to stabilizing queues for ECN-enabled TCP flows," In *Proceedings of IEEE INFOCOM,* 2003.

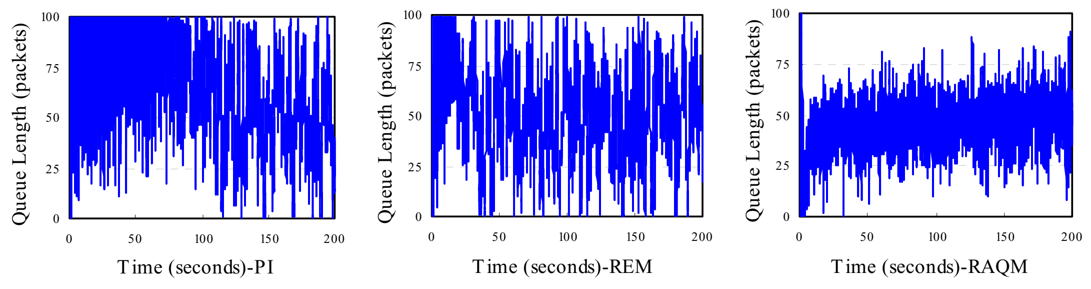
[12] C. Wang, B. Li, K. Sohraby, and Y. Peng, "AFRED: An adaptive fuzzy-based algorithm for active queue management," *The 28th Annual IEEE Conference on Local Computer Networks (LCN)*, Bonn, Germany, October 20-24, 2003.

[13] I. Stoica, S. Shenker and H. Zhang, "Core-stateless fair queuing: Achieving approximately fair bandwidth allocations in high-speed networks," in *Proceedings of SIGCOMM 1998*, pp.118-130, 1998

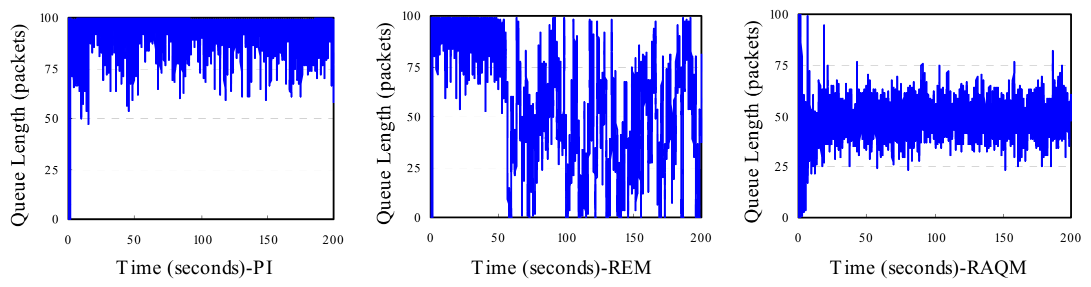
[14] J. Padhye, V.Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation", In *Proceedings of ACM SIGCOMM'98*, 1998.

[15] UCN/LBL/VINT, Network simulator-ns2, *http://www-mash.cs.berkeley.edu/ns*.

Appendix: Figure 4-Figure 8

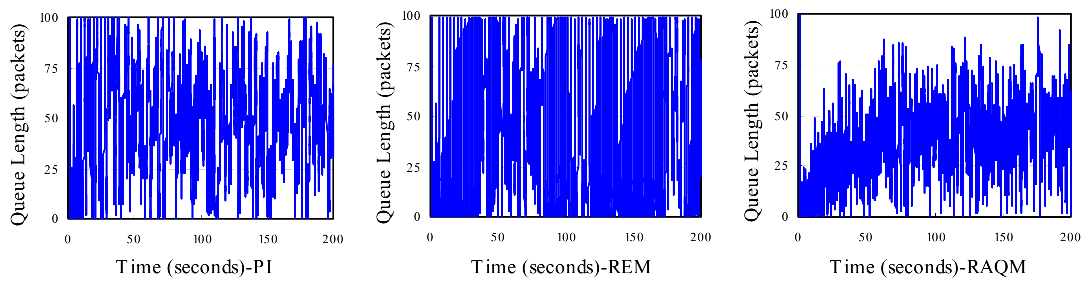


(a): The number of long-lived TCP flows is 40

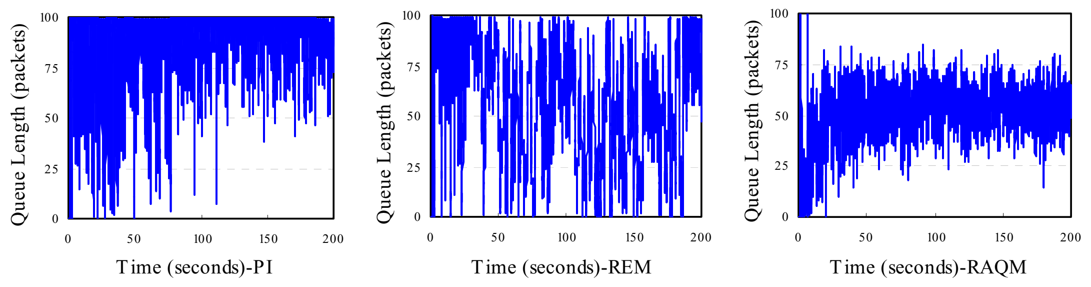


(b): The number of long-lived TCP flows is 400

Figure 4. Queue length in case 1 with small RTT

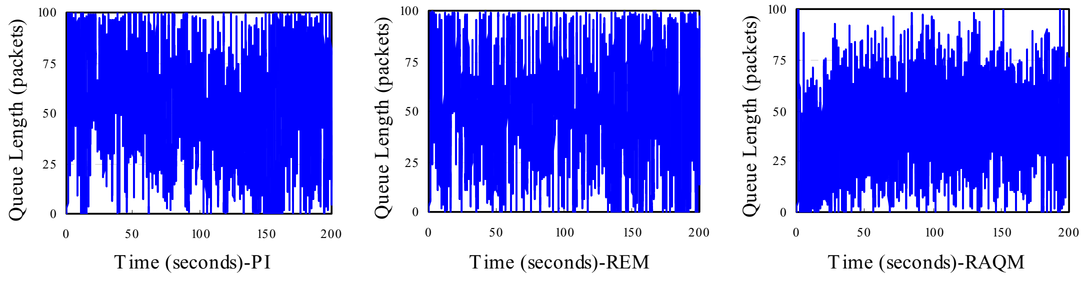


(a): The number of long-lived TCP flows is 40

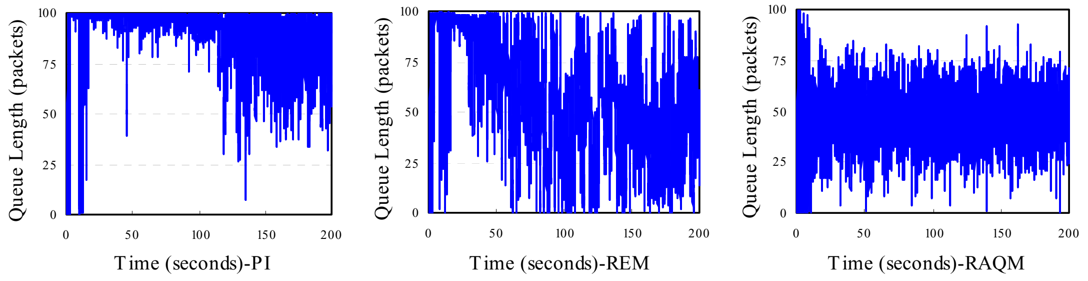


(b): The number of long-lived TCP flows is 400

Figure 5. Queue length in case 2 with large RTT

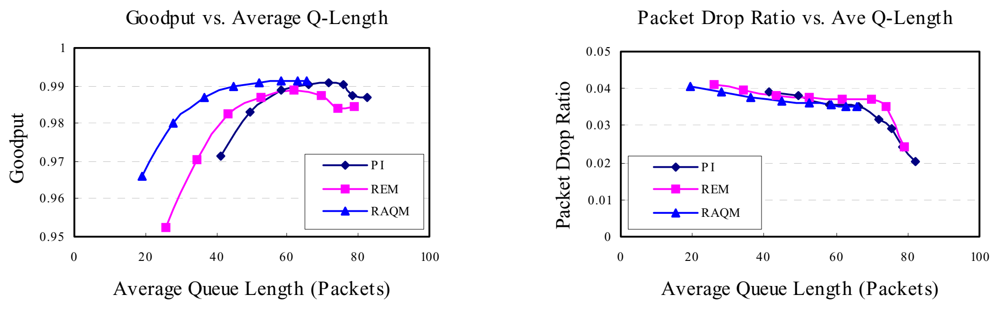


(a): The number of long-lived TCP flows is 40



(b): The number of long-lived TCP flows is 400

Figure 6. Queue length in case 3 with diverse RTT



(a): Goodput vs. Average queue length (b): Packet drop ratio vs. Average queue length

Figure 7. Goodput and packet drop ratio in case 3 with diverse RTT

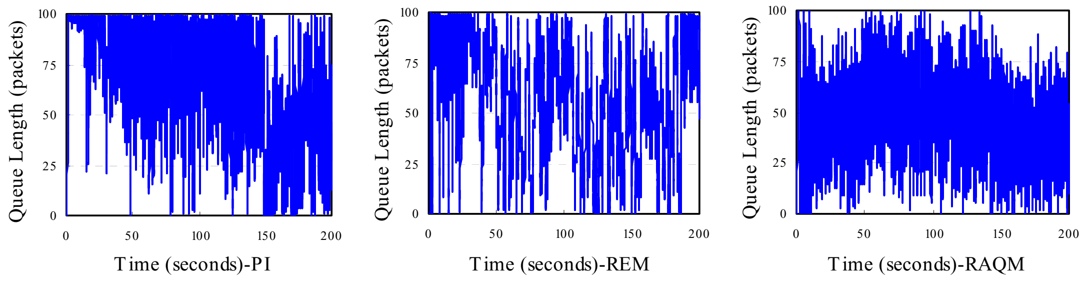


Figure 8. Queue length in case 4 with hybrid traffics