TARGET TRACKING BASED NETWORK ACTIVE QUEUE MANAGEMENT

Shane F. Cotter

ECE Department, Union College
Schenectady, NY 12308
cotters@union.edu

Manohar N. Murthi *

ECE Department, University of Miami
Coral Gables, FL 33146
mmurthi@miami.edu

ABSTRACT

Active Queue Management (AQM) methods attempt to predict and control network router queue levels and provide feedback regarding network congestion to data sources through packet marking/dropping. AQM methods have not employed statistical signal processing principles largely due to the requirement of low complexity. In this paper, we apply optimal filtering and target tracking methods to the design of AQM. In particular, we develop Kalman Filter based AQM which results in router queues with reduced queue level variance. To account for networks with more bursty traffic, we use Interacting Multiple Models (IMM) which similarly result in reduced queue variance in simulations with both long-term and bursty short-term traffic. In comparisons with other AQM methods, these low complexity target tracking-based AQM methods give a more constant queue length without any loss in source throughput.

Index Terms— Networking, active queue management, Kalman Filter

1. INTRODUCTION

In high speed networks such as the Internet, network congestion occurs when data transmitting sources sharing a router overwhelm the router with too many packets to process. This causes the queue level and queueing delay at the router to increase and ultimately a router may drop packets from its buffer if the sources sharing the buffer do not reduce their sending bit-rates. To provide sources with feedback regarding network congestion, Active Queue Management (AQM) methods are commonly employed. An AQM scheme in a router attempts to control the queue level, and detect and control congestion. Upon detection of congestion, the AQM scheme can warn sources by randomly marking packets. If the congestion continues, the AQM scheme actually drops packets, usually in some random manner.

The Random Early Detection (RED) algorithm [1] was the first AQM method. Building upon this early work, many variants of RED have been proposed [2, 3, 4, 5]. In RED, the dropping probability is based on an average queue size which is calculated as a weighted average of the current and past queue values (see section 2). Based on this calculated probability, *future* packets arriving at the router are queued or dropped. As pointed out in [4], one of the drawbacks of RED is its inability to react to bursty traffic which is typical in networks with a large number of active TCP sources. This may be addressed by an algorithm which seeks to react quickly based on a shorter time window and is able to predict the future queue value.

A natural question is whether statistical signal processing methods can be applied to the key prediction and control problems of AQM. In [6], the NLMS algorithm [7] from adaptive filtering was used in the APACE AQM method. In particular, NLMS was used to adapt the weights on previous measurements of the queue length in order to predict the queue value at a future time. Simulation results in [6] showed that the APACE method was better able to control the instantaneous queue than RED [1], SRED [2] and AVQ [8].

In some respects, the prediction and control of a queue level can be viewed as a target tracking problem. Motivated by the success of optimal filtering and estimation principles such as the Kalman Filter and the Interacting Multiple Models (IMM) method in target tracking, we explore the application of these principles to AQM in networks. In particular, we first propose a novel AQM algorithm which uses the Kalman Filter to estimate both the next value of a queue and the rate of change of the queue size. Estimates of these two quantities are incorporated into a new dropping function which is updated at uniform sample times. This method, which we term KF-AQM, is low complexity and predicts the future queue value accurately when the traffic is not overly bursty. To account for very bursty network traffic, we expand our approach to use the IMM target tracking method, which allows us to use other process models to account for greater variations in the buffer size. This approach is termed KF/IMM-AQM and to keep the complexity of the method reasonably low (a requirement for AQM), we limited the number of models to two in our experiments.

Through simulations using long and short duration TCP Reno connections, our proposed methods are shown to give a very stable queue value (i.e., reduced queueing delay variance) at the router, with improvements over both RED and APACE. From an applications viewpoint, a reduced queueing delay variance leads to better applications performance (e.g., better jitter buffer design and performance in streaming video). Moreover, the performance indicates that the KF/IMM-AQM methods produces better queue dynamics without any loss in source throughput. Consequently, this paper shows the potential for the utilization of optimal filtering and target tracking principles in network AQM.

2. RANDOM EARLY DETECTION (RED)

On each packet arrival at the router, the new average queue length q_{ave}^{new} is updated using the current queue length q and the old value of the average queue length q_{ave}^{old} as

$$\begin{aligned} q > 0 \quad &: \quad q_{ave}^{new} = (1 - w_q) q_{ave}^{old} + w_q q \\ q = 0 \quad &: \quad q_{ave}^{new} = (1 - w_q)^m q_{ave}^{old} \end{aligned} \tag{1}$$

The parameter w_q determines the weight given to the new queue measurement and when the queue is empty the average queue value drops off exponentially with the parameter m derived from the idle time between packet arrivals. Using the value q_{ave}^{new} packets are marked (by setting a bit in the header of the packet) or dropped from

*This work was partially supported by National Science Foundation Awards CCF-0347229, and CNS-0519933

the queue. Since the effect on the sender is the same, we will just consider these as dropped packets in our description. The dropping probability p_{drop}^{RED} is calculated based on minimum and maximum queue thresholds m_q and M_q respectively which are usually set as some percentage of the maximum buffer capacity:

$$\begin{aligned} q_{ave}^{new} < m_q : p_d = 0 \quad &; \quad q_{ave}^{new} > M_q : p_d = 1 \\ m_q \leq q_{ave}^{new} \leq M_q \quad &: \quad max_p \,.\, \frac{q_{ave}^{new} - m_q}{M_q - m_q} \end{aligned} \tag{2}$$

The parameter max_p determines how aggressively packets are dropped as the queue builds. The parameter settings m_q, M_q, w_q and max_p were tuned through simulation and are documented in the original RED paper [1].

3. APPLICATION OF TARGET TRACKING TO AQM

It has been well established through a number of studies that network traffic has the property of self-similarity and long-range dependence [10]. However, for small buffers and over the short time scales where we wish to predict future queue values, this characteristic will have no impact on our model [11]. As demonstrated through network measurements in [12], some network flows' packet arrivals (e.g., FTP) cannot be adequately modeled using the Poisson distribution as the burstiness of the data transfer is not captured. While one can use sophisticated traffic models in AQM algorithms, the fact is that one must design low-complexity algorithms suitable for real-time implementation in hardware. Consequently, to capture the burstiness in network traffic, we propose to model the variability using Gaussian statistics with different variances. In particular, by utilizing multiple models (e.g., IMM) from target tracking, one can obtain low complexity AQM algorithms that capture network traffic variability. To start, we first consider a simple Kalman Filter model for predicting future queue arrivals.

3.1. KF-AQM

We consider a uniform sampling of the queue length at the router and denote the queue length at time kT by q_k where T is the sampling period. We frame the problem in AQM as that of determining the future queue length based on the current queue length, past queue lengths and some rate of change in the queue length. The problem posed is very similar to that faced in tracking a moving target which has an unknown velocity and the Kalman Filter has been very successfully applied to this problem [13]. In common with many papers, we assume that any acceleration is taken into account by the noise samples in the model equations. Denoting the rate of change in the queue length at time kT by v_k, we model the evolution of the queue length as:

$$\begin{aligned} q_k &= q_{k-1} + v_{k-1}.T + \delta_{k-1} \\ v_k &= v_{k-1} + \epsilon_{k-1} \end{aligned} \tag{3}$$

Taking the state vector at time kT as $x_k = [q_k \;\; v_k]^T$, we can rewrite (3) in the form

$$x_k = Ax_{k-1} + \varepsilon_{k-1} \text{ where } A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{4}$$

and $\varepsilon_{k-1} = [\delta_{k-1} \;\; \epsilon_{k-1}]^T$ is a Gaussian noise with distribution $\mathcal{N}(0, V_{k-1})$.

The queue length z_k is the number of complete packets in the buffer at time kT and the measurement model is given as

$$z_k = q_k + e_k. \tag{5}$$

The deviation of the measurement z_k from the true q_k is due to the fact that packets are arriving and departing from the queue at any given time point and so a measurement in packets does not capture the partial packets which should be subtracted or added to give the true length of the queue. The error is assumed to be Gaussian and distributed as $e_k \sim \mathcal{N}(0, R_k)$. Expressing (5) in terms of the state vector x_k, $z_k = Hx_k + e_k$ where $H = [1 \;\; 0]$.

Assuming the prior distribution of the state vector is $x_0 \sim \mathcal{N}(m_0, P_0)$ where m_0 is the mean and P_0 is the covariance, the Kalman Filter can be summarized using a prediction step which gives the values m_k^- and P_k^- and an update step which gives the next mean and covariance estimates m_k and P_k [7, 13]:
Prediction:

$$m_k^- = Am_{k-1} \;;\; P_k^- = AP_{k-1}A^T + V_{k-1} \tag{6}$$

Update:

$$\begin{aligned} u_k &= z_k - Hm_k^- \\ K_k &= P_k^- H^T S_k^{-1} \text{ where } S_k = (HP_k^- H^T + R_k) \\ m_k &= m_k^- + K_k u_k \;;\; P_k = P_k^- - K_k S_k K_k^T \end{aligned} \tag{7}$$

When a new measurement of the queue length q_k is made, (6) is used to obtain m_{k+1}^- which gives a prediction of the future queue length q_{k+1}^- and rate of change in the queue length v_{k+1}^-. We incorporate both components of the next state into a new dropping/marking probability. Given a buffer size, B, and a target fractional occupancy of the queue α, we calculate this probability as

$$\alpha B < q_{k+1}^- < B : p_{drop}^{KF} = \frac{q_{k+1}^- - \alpha B}{B - \alpha B} + \gamma_v v_{k+1}^- \tag{8}$$

where γ_v is a weighting parameter which must be chosen. If $q_{k+1}^- < \alpha B$, $p_{drop}^{KF} = 0$ and if $q_{k+1}^- \geq B$, $p_{drop}^{KF} = 1$. This sets the dropping probability for any packets which arrive between the current queue measurement and the next queue measurement.

In practice, we consider both noise variances V_k and R_k to be time-invariant; V_k is diagonal with equal variances denoted by η and $R_k = \rho$. The setting of these parameters will be discussed in the simulation section.

3.2. KF/IMM-AQM

While a single model of the queue dynamics can adequately capture the behavior of the queue under a variety of traffic conditions, the bursty nature of network traffic means that the behavior of the queue will at times diverge from one particular model. Drawing from the Kalman Filter literature, we can effectively deal with these sudden changes by using a number of different models for the queue dynamics and associating measurements with the best matching model. The Interacting Multiple Model (IMM) Kalman Filter [13] combines the estimates from different tracking models at the start of each estimation cycle and a transition probability matrix, T, is used to determine the interaction. A single cycle of the IMM can be best described as three different stages: interaction, filtering, and combination [14].

We denote each model by $M^j, j = 1, \cdots, N$ and the prior probability of each model at the kth time step by $\mu_k^i, i = 1, \cdots, N$. The different models are described by

$$x_k = A_{k-1}^j x_{k-1} + \delta_{k-1}^j \;;\; z_k = H_k^j x_k + e_k^j, j = 1, \cdots, N \tag{9}$$

Interaction: The mixing probabilities are first calculated as

$$c_j = \sum_{i=1}^{N} T_{ij}\mu_{k-1}^{i} \; ; \; \mu_k^{i|j} = \frac{1}{c_j} T_{ij}\mu_{k-1}^{i}. \tag{10}$$

Using these probabilities, the initial mixed input means and covariances for each filter are obtained as:

$$m_{k-1}^{0j} = \sum_{i=1}^{N} \mu_k^{i|j} m_{k-1}^{i} \tag{11}$$

$$P_{k-1}^{0j} = \sum_{i=1}^{N} \mu_k^{i|j} \{P_{k-1}^{i} + (m_{k-1}^{i} - m_{k-1}^{0j})(m_{k-1}^{i} - m_{k-1}^{0j})^T\} \tag{12}$$

Filtering: For each model, the means and covariances calculated in the interaction step along with the system equations in (9) are put through a Kalman Filter detailed in (6) and (7) to give the new means and covariances for each model at the kth step: $m_k^i, P_k^i, i = 1, \cdots, N$. For each model, the likelihood of the measurement for each filter is computed as

$$\Lambda_k^i = \mathcal{N}(u_k^i; 0, S_k^i) \tag{13}$$

where u_k^i is the measurement residual and S_k^i is the covariance matrix obtained from the Kalman Filter update (see (7)). The new model probabilities are obtained as

$$\mu_k^i = \frac{\Lambda_k^i c_i}{\sum_{i=1}^{N} \Lambda_k^i c_i} \tag{14}$$

where c_i is obtained from (10).

Combination: Finally, the updated mean and covariance is computed by combining the estimates from the different models:

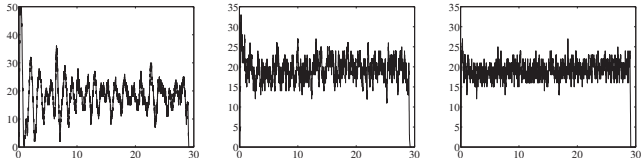
$$m_k = \sum_{i=1}^{N} \mu_k^i m_k^i \tag{15}$$

$$P_k = \sum_{i=1}^{N} \mu_k^i \{P_k^i + (m_k^i - m_k)(m_k^i - m_k)^T\} \tag{16}$$

4. SIMULATIONS

As a common testbed for AQM algorithms [1, 5, 6] we consider a network with a dumbbell topology where each source node at one side of a bottleneck link is paired with a sink node at the other side of the bottleneck link. The bottleneck link has a capacity of 1Mbps while the other links in the network have a capacity of 3Mbps so that this simulation is identical to the APACE simulation in [6]. The propagation delay on each of the links is set to 1ms which means that the packet delay is mainly due to queueing delay. The buffer size at the bottleneck node is set to 50 packets. All packets are sent over TCP Reno connections and the size of all packets in the network is set to 552 bytes. The performances of the KF-AQM and KF/IMM-AQM methods are compared to both the RED [1] and APACE [6] methods using simulations in ns-2 [9]. APACE was run with a filter length of $M = 10$ and lookahead of $P = 6$ and the fractional occupancy of the buffer was set to 0.3 for both APACE and KF-AQM.

In KF-AQM, the sampling frequency was varied over the range 100-200Hz and the difference in performance was found to be minimal so the sampling frequency was set to 100Hz. The weight parameter γ_v in p_{drop}^{KF} from (8) was also varied and a value of 0.05 was found to give the best performance across multiple simulations. The measurement noise variance ρ was set to 0.5 while the process noise variance was set to $\eta = 1.0$.



(a) RED (b) APACE (c) KF-AQM

Fig. 1. Comparison of the router instantaneous queue length in packets (y-axis) over 0-30 seconds (x-axis) with 40 nodes transmitting over FTP using (a) RED, (b) NLMS, and (c) KF-AQM.

Nodes	KF-AQM		APACE		RED	
	μ	σ	μ	σ	μ	σ
10	13.97	1.75	14.31	2.30	15.44	3.15
15	15.92	1.81	16.08	2.23	16.53	4.63
20	16.94	1.83	17.23	2.25	17.18	4.33
25	17.17	1.91	17.76	2.51	17.61	3.51
30	18.53	1.77	17.95	2.65	18.01	4.53
35	19.12	1.83	18.67	2.74	18.25	5.79
40	19.15	1.80	19.02	2.65	18.53	4.84
45	19.95	1.83	20.06	3.06	19.01	4.56
50	19.90	1.82	20.69	3.15	18.82	6.52

Table 1. Mean (μ) and standard deviation (σ) of queue length for each of the AQM methods KF-AQM, APACE, and RED as the number of FTP connections is increased.

4.1. Long-lived TCP Flows

In this experiment, each of the nodes in the network constantly streams packets over FTP which corresponds to long file transfers from each source to destination node through the bottleneck link. As the number of nodes increases, the congestion at the bottleneck router increases leading to increased packet drops. The number of transmitting nodes was increased from 10 to 50 in steps of 5 and the effect on the bottleneck queue was examined.

To illustrate the difference in the instantaneous queue when using RED, APACE, and KF-AQM, the queue evolution with 40 nodes transmitting is shown in figure 1. At the very start of the simulation, it is seen that RED is unable to control the queue and quickly reaches the buffer limit. The APACE algorithm improves on the RED performance but the KF-AQM method is even more effective in rapidly bringing the queue length to a stable level. As the packet arrivals from the different flows fluctuate, the queue size using RED shows large oscillations while APACE is more successful in limiting the queue variance. KF-AQM further reduces the queue variance maintaining a more constant queue length at the router.

Similar behavior was observed with different numbers of transmitting nodes and table 1 gives the mean and standard deviation of the queue length over the time range 5-25 seconds (which avoids the changes in the queue size at the start and end of the simulation). For all methods, the mean queue size increases slowly as the number of FTP sources increases. However, the standard deviation illustrates that KF-AQM is much more effective than APACE or RED in keeping the queue size close to a constant value as the number of sources increases. Indeed the deviation from the mean queue size for KF-AQM is less than 2 packets in each case. In table 2, the number of drops is summarized for each method and this shows that the superior performance of KF-AQM is not at the expense of an increased drop rate over RED or APACE indicating that a high throughput is maintained. Even though the total number of packets dropped is similar in all cases, the times at which packets are dropped are different

Nodes	RED	NLMS	KF-AQM
10	734	772	749
15	1076	1129	1015
20	1295	1341	1330
25	1492	1606	1465
30	1596	1617	1664
35	1720	1705	1746
40	1784	1888	1867
45	2020	1975	1999
50	1973	2148	2111

Table 2. Marked/dropped packets recorded for each of the AQM methods KF-AQM, APACE, and RED as the number of FTP connections is increased.

Method	μ	σ	Drops
RED	19.95	6.87	4415
APACE	22.87	3.95	2817
KF/IMM-AQM	22.92	3.52	2754

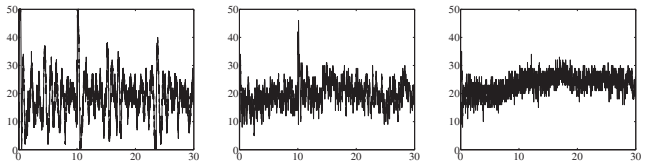
Table 3. Mean (μ) and standard deviation (σ) of queue length and drops for each of the AQM methods RED, APACE, KF/IMM-AQM with short and long TCP flows.

leading to the observed differences in queue lengths at the router. The better performance of KF-AQM can be attributed to its ability to reliably predict the future queue length and base its drop decisions on this prediction.

4.2. Long-lived and Short-lived TCP Flows

The other form of TCP transfer prevalent in the internet is short-lived flows resulting from short HTTP requests and responses. The number of long-lived flows will also be time-varying. In this experiment, we simulate one such scenario where we start with 10 long-lived TCP flows and at 10 seconds 20 more long-lived TCP flows are added to the router traffic over a period of 0.5 seconds. In addition to this, 6 nodes generate short TCP transfers which pass through the bottleneck link to the destination nodes. The sizes of the short TCP transfers are governed by a Pareto distribution with mean size 10KB and a shape parameter of 1.5. The sessions arrive from the source nodes according to a Poisson distribution with a mean arrival rate of $\lambda = 100$ sessions/second. The behaviors of the router queue using RED, APACE and KF/IMM-AQM were compared. In this case, we found that the bursty nature of the traffic was modeled better by using an IMM with two models. The parameter settings were kept the same as the first simulation for APACE and RED. For KF/IMM-AQM, A and H were kept the same in each model (see section 3.1) but the process noise variances were set to $\eta_1 = 1.0$ to account for small changes in the queue and $\eta_2 = 25.0$ to capture the behavior of the queue when sudden bursts of traffic arrived at the router.

The instantaneous queues for each method are shown in figure 2. It is readily apparent that the KF/IMM-AQM method absorbs the new flows that arrive at 10 seconds much better than either the APACE or RED methods. In table 3, the mean and standard deviation of the queue length along with the packet drops for each AQM methods are summarized. The RED algorithm fails to work with this mix of traffic types as seen from the queue oscillations and number of packet drops. The APACE method controls the queue size much better than RED and the packet drops were significantly reduced. The new KF/IMM-AQM method performs best of all as seen from the reduced variance in queue length in figure 2(c) compared to 2(a) and 2(b) and also confirmed by the values in table 3.



(a) RED (b) APACE (c) KF/IMM-AQM

Fig. 2. Comparison of the router instantaneous queue length in packets (y-axis) over 0-30 seconds (x-axis) with long and short TCP transfers using (a) RED, (b) NLMS, and (c) KF/IMM-AQM.

5. CONCLUSIONS

We have applied optimal filtering principles drawn from statistical signal processing to the problem of network congestion control and queue management in network routers. We introduced Kalman Filter based AQM (KF-AQM) and extended this to Interacting Multiple Model AQM (KF/IMM-AQM) to account for increased burstiness at the router under certain network traffic conditions. The complexity of these AQM methods is reasonably low and simulations showed that these methods succeeded in keeping the queue length close to a target fractional buffer occupancy level. Simulations further revealed that our methods result in reduced queue variance in comparison to other AQM methods such as RED and APACE which is a very desirable characteristic from an applications viewpoint. Future work will examine the complexity performance trade-off associated with increased numbers of models in KF/IMM-AQM.

6. REFERENCES

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", *IEEE Trans. on Net.*, pp. 397-413, Aug. 1993.

[2] T.J. Ott et al., "SRED: Stabilized RED", Proc. IEEE INFOCOM, New York, NY, pp. 1346–1355, March 1999

[3] S. Floyd et al., "Adaptive RED", ACIRI Technical Report, http://www.icir.org/floyd/adaptivered/, 2001

[4] W. Feng et al., "The BLUE active queue management algorithms", *IEEE Trans. on Net.*, pp. 513–528, Aug. 2002

[5] C. V. Hollet et al., "On designing improved controllers for AQM routers supporting TCP flows", Proc. IEEE INFOCOM, Anchorage, AK, Vol. 3, pp. 1726–1734, April 2001

[6] A. Jain et al., "Adaptive prediction based approach for congestion estimation (APACE) in active queue management ", *Computer Communications*, pp. 1647–1660, Oct. 2004

[7] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, 4th Ed., 2001

[8] S.S. Kunniyur and R. Srikant, "An adaptive virtual queue (AVQ) algorithm for active queue management", *IEEE Trans. on Net.*, Vol. 12(2), pp. 286–299, April 2004

[9] The network simulator ns-2, http://www.isi.edu/nsnam/ns/

[10] K. Park, *Self-Similar Network Traffic and Performance Evaluation* Wiley-Interscience, 2000

[11] M. Grossglauser and J. Bolot "On the relevance of long-range dependence in network traffic", *IEEE Trans. on Net.*, pp. 629-640, Oct. 1999

[12] V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling", *IEEE Trans. on Net.*, Vol. 3(3), pp. 226-244, June 1995

[13] Y. Bar-Shalom et al., *Estimation with Applications to Tracking and Navigation*, Wiley-Interscience, 2001

[14] E. Mazor et al., "Interacting multiple model methods in target tracking: a survey", *IEEE Trans. on AES*, pp. 103-123, Jan. 1998.