A Note on the Stability Requirements of Adaptive Virtual Queue

Dina Katabi
MIT-LCS
dk@mit.edu

Charles Blake
MIT-LCS
cb@mit.edu

Abstract

Choosing the correct values for the parameters of an Active Queue Management (AQM) scheme is a well-known hard problem. The Adaptive Virtual Queue (AVQ) attempts at solving this problem by using stability requirements to devise a rule for setting its parameter. This memo shows that the AVQ rule for setting its parameter is impractical for many real-life situations.

1 Background

Active Queue management (AQM) refers to queuing schemes in which the router signals anticipated congestion to TCP sources by dropping or marking their packets.

An important problem facing the deployment of various AQMs is the inability to identify the correct values of the corresponding parameters. Random Early Discard (RED [4]), REM [1], Blue [3], PI-controller [5] all rely on a set of parameters whose values dramatically affect the performance of the queuing scheme [7, 2]. Experiments show that in many cases the effective values for the parameters change with the number of sources, the capacity and the feedback delay. However, none of these AQMs de£ne a systematic rule for setting its parameters.

In contrast to other AQMs, the recently proposed Adaptive Virtual Queue (AVQ [6]) provides a systematic rule for setting its parameter α.

The AVQ algorithm maintains two variables called *the virtual capacity* and *the virtual queue*. The virtual capacity takes values smaller than the actual capacity of the link. When a packet arrives, it is queued in the real queue and further the virtual queue is updated to re¤ect a new arrival. Packets are drained from the virtual queue according to the virtual capacity. Packets in the real queue are marked or dropped when the virtual buffer over¤ows.

The virtual capacity $\tilde{C}$ is updated according to the following rule:

$$\dot{\tilde{C}} = \alpha(\gamma C - \lambda), \tag{1}$$

where C is the link capacity, λ is the input traf£c arrival rate, γ is a constant set to 0.98, and α is a constant whose value is chosen to achieve some stability requirements.

Based on a stability analysis, the authors of AVQ devise a rule for setting α. In particular, when the designer can specify a lower bound on the number of sources traversing the link N, and an upper bound on the round trip delay d, α should be set to a value that satis£es:

$$\omega d + \arctan(\frac{\omega}{K_{11}}) = \frac{\pi}{2}, \tag{2}$$

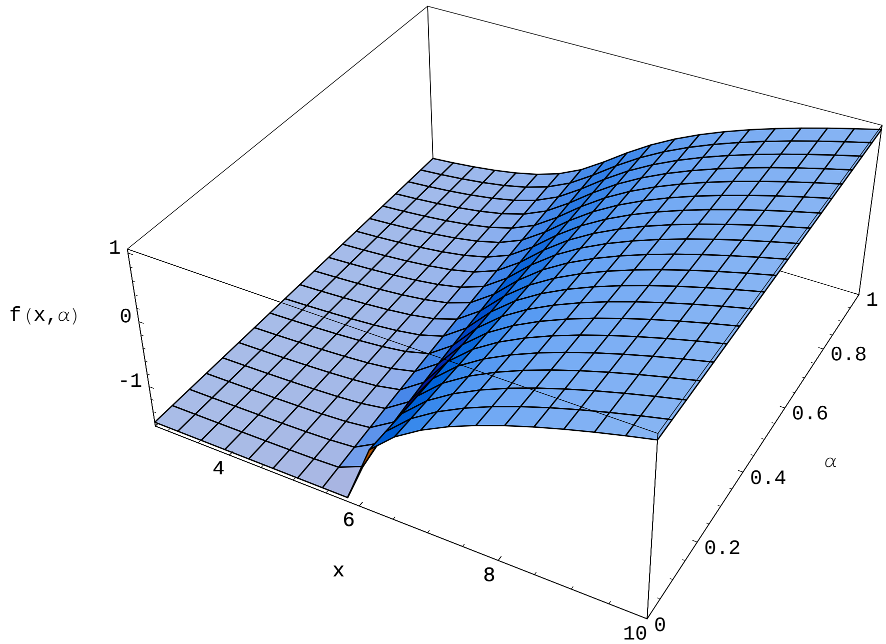

Figure 1: Impracticality of the AVQ parameter setting rule. The values of α that satisfy the stability requirements are those for which $f = 0$. However, f is highly insensitive to α. It is zero only for $x \approx 7$, which means an unacceptably low per-¤ow throughput.

The term ω is computed as:

$$\omega(\alpha, d, N, \gamma) = \frac{1}{\sqrt{2}} sqrt(K_{12}^2 - K_{11}^2 + sqrt((K_{12}^2 - K_{11}^2)^2 + 4K_2^2\alpha^2)). \quad (3)$$

The parameters K_1 and K_2 are de£ned as follows.

$$K_1 = \frac{N}{\gamma C d^2} \quad \text{and} \quad K_2 = \frac{2}{3}\frac{\gamma C}{N}.$$

Below, we show that such a design rule is impractical for high capacity links.

2 Solving the AVQ Equation

In this section, we show that as the capacity of the link increases, Equation 2 becomes unsolvable for almost all practical situations.

First, we note that capacity, C, and number of ¤ows, N, are coupled in Equation 2. In particular, we can substitute $\frac{\gamma C}{N}$ by a single variable $x = \frac{\gamma C}{N}$. As a result,

$$K_1 = \frac{1}{xd^2} \quad \text{and} \quad K_2 = \frac{2}{3}x.$$

Substituting the values of K_1 and K_2 in Equation 3, it becomes clear that ω is a function of d, x, and α.

Next, we examine all the values of α that satisfy Equation 2. To do so, we de£ne the following function:

$$f(\alpha, x, d) = wd + \arctan(\frac{w}{K_{11}}) - \frac{\pi}{2}.$$

Clearly, the values of α for which $f = 0$ are the solutions of Equation 2, and thus are the values that should be used by the network administrator.

Using Mathematica, we plot f as a function of both α and x for a speci£c delay. For example, Let us assume the maximum delay is $d = 0.21$ seconds, as used in [6]. Figure 1 shows the function $f(\alpha, x)$ for that delay. As seen from the £gure, the function f is highly insensitive to the values of α. Its value is zero only for values of x that are less than 7 packets/second per ¤ow.[1]

The above result can be restated also as follows., if the administrator wants to choose α according to the rule in [6], then he has to design the system so that the average per ¤ow throughput is 7 packets/sec, and hence the average window size is around 2 packets/RTT (i.e., 7×0.21). For example, If the maximum delay is around 0.21 seconds, and the link capacity is larger than 1 Gb/s, then to design for stability, the ISP needs to design for a simultanuous number of ¤ows larger than $16,000$ ¤ows.

The insensitivity of $f = 0$ to the value of α means that the network administrator cannot choose the minimum number of ¤ows, the capacity, γ, and the maximum delay that correspond to its network, and compute the appropriate α, as suggested in [6]. Since the maximum delay, the link capacity, and the minimum number of sources are not design parameters, but rather parameters whose values are implied by the environment, the AVQ equation in many cases becomes unsolvable.

Apart from the practicality issue, the above result shows a ¤aw in the stability analysis used by the authors of [6]. In particular, their analysis, assumes that the TCPs sharing the link are always performing Additive-Increase Multiplicative-Decrease. However, the average window size imposed by the stability analysis is less than 4 packets, the minimum window required to fast retransmit. Therefore, most of the TCP ¤ows sharing the link will be in timeout or slow start rather than in the AIMD mode.

Although, the results in Figure 1 are for a maximum round trip delay of 0.21 seconds, they can be generalized to the range of practical network delays. In general, for the a maximum round trip delay in [0.1s, 0.5s], $f = 0$ is insensitive to α and depends mostly on $x = \frac{\gamma C}{N}$. Further, for $d \in$ [0.1s, 0.5s] the values of x for which $f = 0$ satisfy $x \times d \approx 2$. This means that regardless of the maximum delay, the average congestion window that satis£es the stability rule cannot be larger than few packets. The average per-¤ow throughput, however, depends on the maximum delay. Figures 2 and 3 show f as a function of α and x for $d = 0.1$s and $d = 0.5$s respectively. The £gures show that if the administrator designs his network for a maximum delay of 0.1s, then the per-¤ow throughput is around 15 packets/s, whereas, A maximum delay of 0.5s results a per-¤ow throughput of 3 packets/s.

3 Conclusions

This memo shows that the rule provided for setting the parameter of Adaptive Virtual Queue is impractical, and that this issue remains as problematic as it is for any other AQM scheme.

References

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. Rem: Active queue management. *IEEE Network*, 2001.

[1]Plotting f for a larger range of x shows clearly that f does not reach zero for larger values of x. The above range was chosen for visual convenience.

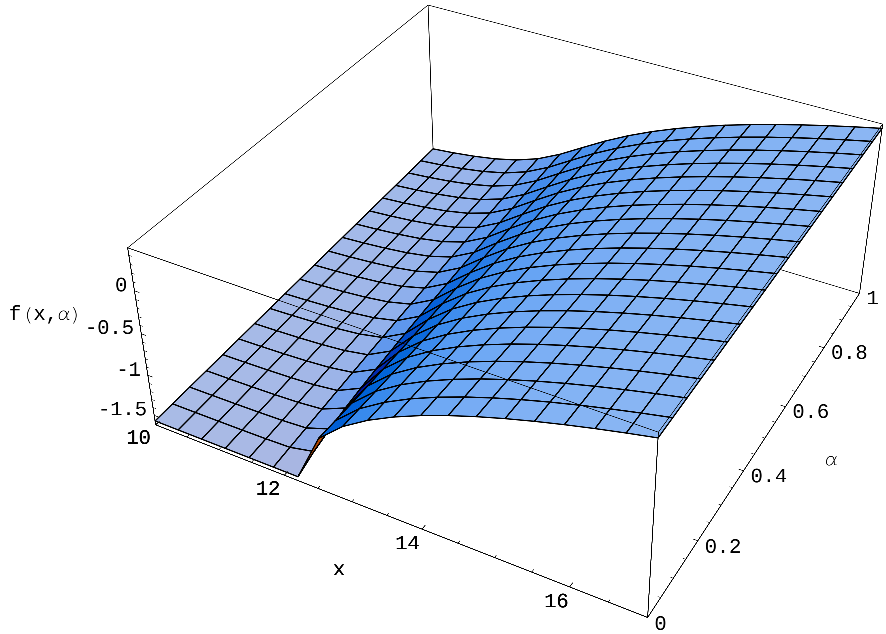

Figure 2: f as a function of α and x for d=0.1

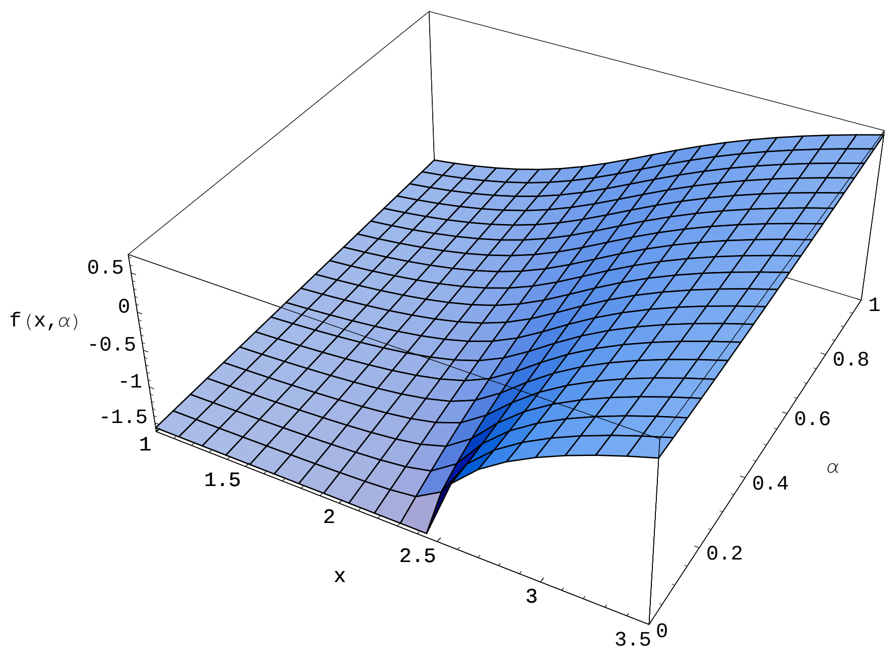

Figure 3: f as a function of α and x for d=0.5

[2] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. Tuning red for web traf£c. In *Proc. of ACM SIGCOMM '00*, Aug. 2000.

[3] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, Apr. 1999.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. In *IEEE/ACM Transactions on Networking, 1(4):397–413*, Aug. 1993.

[5] C. Hollot, V. Misra, D. Towsley, , and W. Gong. On designing improved controllers for aqm routers supporting tcp ¤ows. In *Proc. of IEEE INFOCOM*, Apr. 2001.

[6] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue. In *Proc. of ACM SIGCOMM*, 2001.

[7] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy red, June 1999.