"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVII – 2014 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in:
PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

THE PROTOCOL MODEL FOR THE CONGESTION CONTROL

Ioan POPOVICIU
Assistant professor, PhD Naval Academy "Mircea cel Batran", Constanta

***Abstract**: The current Internet allows applications to use the network with arbitrary data rates and congestion response, potentially in a harmful way. Protection of the public network may not be a practically important problem when the majority of Internet applications uses TCP. However, it becomes serious with the growth of delay sensitive applications such as streaming media, which often prefer UDP over TCP as their transport protocol choice. A non-TCP protocol is called TCP-friendly when it yields the same throughput as traditional TCP. TCP-friendly protocols are generally used for multimedia/real-time applications. This paper proposes a TCP-friendly protocol model for the streaming media based on Additive Increase / Multiplicative Decrease Control algorithm (AIMD) .*
***Keywords:** TCP, congestion, AIMD, network, friendly*

1. Introduction

Network congestion is characterized by presence of a large number of packets (load) being routed in all or portions of the subnet that exceeds its link and router capacities (resources) resulting in a performance slowdown

A network is considered congested when too many packets try to access the same router's buffer, resulting in an amount of packets being dropped. In this state, the load exceeds the network capacity. During congestion, actions need to be taken by both the transmission protocols and the network routers in order to avoid a congestion collapse and furthermore to ensure network stability, throughput efficiency and fair resource allocation to network users. Indeed, during a collapse, only a fraction of the existing bandwidth is utilized by traffic useful for the receiver. Congestion collapse is considered, in general, as a catastrophic event. However, congestion itself is associated with different properties, depending on the characteristics of the underlying networks, the mechanisms of the transmission protocols, the traffic characteristics of the contenting flows, the level of flow contention, and the functionality of network routers. Therefore, the impact of congestion may be temporary and easily controllable; or it may be catastrophic. Consider, for example, a high speed network which hosts a number o competing flows that increases or decreases. The window of each flow also increases and decreases. However, unlike the traditional networks, the time it takes for the flows to exploit the available bandwidth is certainly longer; the amount of loss upon congestion is certainly higher; and the duration of congestion itself throughout the overall communication time may be relatively smaller. Since the nature of acceptable congestion cannot be prescribed or even accurately defined in general, congestion control becomes a complex task. Furthermore, complexity increases due to the multipurpose-task of congestion control algorithms. They need to control congestion and avoid collapses, maximize bandwidth utilization, guarantee network stability, and ensure fair resource allocation. Considering the network as a black box that only provides a binary feedback to network flows upon congestion, shifts all the burden to end users and calls for solutions that are more generic and perhaps less responsive. That is, a binary congestion signal does not reflect the particular network state. Each sender operates independently and goals to adjust its rate (or window) in a manner that the total bandwidth of the network will be expended fairly and effectively. From its algorithmic perspective the above problem is challenging because the distributed entities (sources) do not have any prior or present knowledge of the other entities' states; nor do they know the system's capacity and the number of competitors. Hence, the goal of fairness and efficiency appears initially difficult to attain. However, if the system is entitled to a prescribed behavior and the entities agree on common transmission tactics, convergence1 to fairness becomes feasible. AIMD (additive increase/multiplicative decrease), the traditional congestion control algorithm of theInternet, operates within that scope: it increases additively the rate of the senders (by a value α) until the system reaches congestion. Upon congestion, all senders decrease their rate multiplicatively using a decrease ratio β. On the other hand, one can measure network conditions, estimate the available bandwidth or even flow contention, and obtain some knowledge about the network. However, measurements are taken at time-instances which may not necessarily represent current network dynamics, or may not correspond to the overall conditions; consequently, protocols may not manage to accurately estimate the load and predict its duration, resulting in either wrong estimations or wrong recovery strategies. Furthermore, some generic questions cannot really be addressed with certainty: How frequently should we measure the network? How far can we trust our measurements? How responsive should the recovery strategy be? How shall we associate the instantaneous measurements of congestion the instantaneous measurements of congestion with the network load over some sufficiently long but also sufficiently recent time period? Consequently, the network may not be a black box but it is certainly not better than grey, involving occasionally a considerable risk. One can go beyond the blind algorithms or the high risk of estimations and actually ask the network2 for help. Of course, precision comes at some cost. Besides the practical difficulty of layer collaboration and the issue of convincing people to add functionality (and invest money) to their network, the issue of recovery strategies remains. That is, even when the network is really a green box (which practically is very difficult), changes
may be so rapid and unpredictable that our costly and painful effort to obtain some information may go wasted.
Steps of closed-loop congestion control:

- congestion detection: system monitoring
- transmit the information to parts of the network where corrective measures are possible.
- adjust network operation parameters (routing procedures etc.) to correct the problem.

For congestion detection we can utilize two technique:

- Notification from packet switches (routers).
- Infer congestion from packet loss:
  - Packet loss can be used to detect congestion because packet loss due hardware failure is very rare.
  - Sender can infer congestion from packet loss through missing acknowledgments.
  - Rate or percentage of lost packets can be used to gauge degree of congestion.

Congestion control methods:

- Traffic Shaping:
  - Heavily used in VC subnets including ATM networks.
  - Avoid bursty traffic by producing more uniform output at the hosts.
  - Representative examples: Leaky Bucket, Token Bucket.
- Admission Control:
  - Used in VC subnets.
  - Once congestion has been detected in part of the subnet, no additional VCs are created until the congestion level is reduced.
- Choke Packets:
  - Used in both datagram and VC subnets.

- When a high level of line traffic is detected, a choke packet is
- sent to source host to reduce traffic.
- Variation Hop-by-Hop choke packets.

• Load Shedding:

- Used only when other congestion control methods in place fail.
- When capacity is reached, routers or switches may discard a number of incoming packets to reduce their load.

Exemples of congestion control methods:

a)**The leaky bucket** (Figure 1): a traffic shaping method that aims at creating a uniform transmission rate at the hosts.

• Used in ATM networks.

• An output queue of finite length is connected between the sending host and the network.

• Either built into the network hardware interface or implemented by the operating system.

• One packet (for fixed-size packets) or a number of bytes (for variable-size packets) are allowed into the queue per clock cycle.

• Congestion control is accomplished by discarding packets arriving from the host when the queue is full.

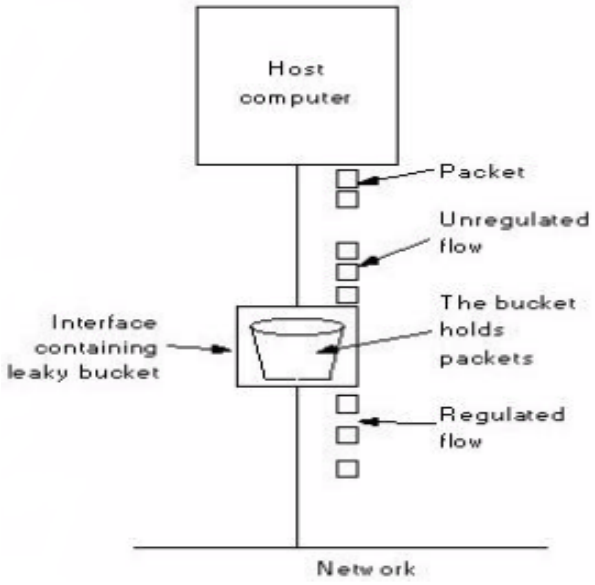


Figure 1

b) **The token bucket** (Figure 2):

• An output queue is connected to the host where tokens are generated and a finite number is stored at the rate of DT

• Packets from the host can be transmitted only if enough tokens exist.

• When the queue is full tokens are discarded not packets.

• Implemented using a variable that counts tokens.

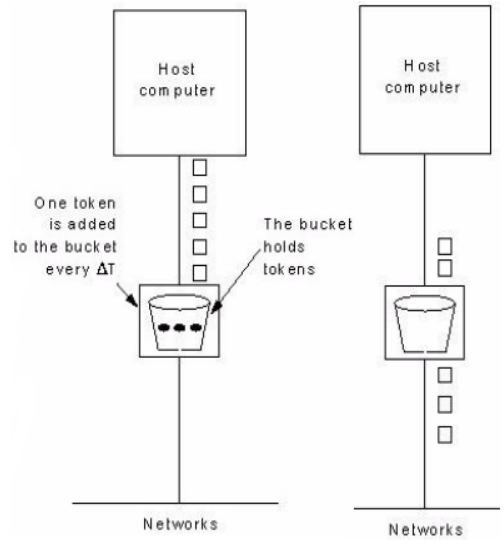


Figure 2

c) **Choke packets**:

• Used in both VC and datagram subnets.

• A variable "*u*" is associated by the router to reflect the recent utilization of an output line: $u = au_{old} + (1-a)f$

• When "*u*" goes above a given threshold, the corresponding line enters a warning state.

• Each new packet is checked if its output line is in warning state if so:

– The router sends a choke packet to the source host with the packet destination.

– The original packet is tagged (no new choke packets are generated).

• A host receiving a choke packet should reduce the traffic to the specified destination.

• A variation (Hop-by-Hop Choke Packets) operate similarly but take effect at each hop while choke packets travel back to the source**.**

2. Goals and metrics

The congestion window determines the number of packets that can be outstanding at any time. That is, the number of packets that can be sent without having received the corresponding ACK packets. It is incorporated into the transport layer and controls the number of packets put into the network. The rate describes packets per second or bits per second. The window or rate can be dynamically adjusted as the total load on the system changes, however, the former is strictly based on ACKs. A cycle is the phase between two seriate feedbacks of 1 (indicating congestion). Hence, a cycle consists of one decrease step triggered by congestion and a number of additive increase steps. A step describes a single window adjustment in response to a single feedback (either 0 or 1). The system is in an equilibrium state, when resource usage of all flows in a bottleneck is balanced. AIMD-based congestion control algorithms guarantee convergence to equilibrium [4]. In congestion avoidance algorithms this is not always guaranteed. A non-TCP protocol is called TCP-friendly when it yields the same throughput as traditional TCP. TCP-friendly protocols are generally used for multimedia/real-time applications. Although the sources might discover their fair-share early on, the dynamics of real systems in practice prohibit a straightforward adjustment, but instead, they call for continuous oscillations as a means of discovering the available bandwidth. The metrics for the system performance are as follows:

• **Efficiency**: Efficiency is the average flows throughput per step (or per RTT- round-trip time), when the system is in equilibrium.

• **Fairness**: Fairness characterizes the fair distribution of resources between flows in a shared bottleneck link. A well-known metric is: $F(x) = \frac{\sum (x_i)^2}{\sum (x_i^2)}$. This index is bounded between 0 and 1.

• **Convergence Speed**: Convergence speed describes time passed till the equilibrium state.

• **Smoothness**: Smoothness is reflected by the magnitude of the oscillations during multiplicative decrease. It depends on the oscillations size.

• **Responsiveness**: Responsiveness is measured by the number of steps (or RTTs- round-trip time) to reach an equilibrium (i.e., to equate the windows in order to be in a fair state).

The difference between Responsiveness and Convergence Speed is that the former is related to a single flow and the latter to the System.

Goals in the evaluation process of a congestion avoidance/control algorithm are:

• To achieve high bandwidth utilization.
• To converge to fairness quickly.
• To minimize the amplitude of oscillations.
• To maintain high responsiveness.
• To coexist fairly and be compatible with traditional widely-used (AIMD based) protocols.

3. TCP Congestion Control

TCP uses a form of end-to-end flow control. In TCP, when a sender send a packet, the receiver acknowledges receipt of the packet. A sending source can use the acknowledgement arrival rate as a measure of network congestion. When it successfully receives an acknowledgment, a sender knows that the packet reached its destination. The sender can then send new packets on the network. Both the sender and the receiver agree on a common **window size** for packet flow . The window size represents the number of bytes that the source can send at a time. The window size varies according to the condition of traffic in the network to avoid congestion . Generally, a file of size f with a total transfer time of Δ on a TCP connection results in a **TCP transfer throughput** denoted by **r** and obtained from equation

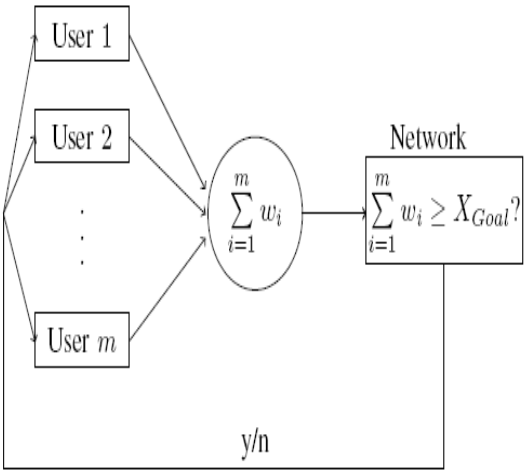
$$\mathbf{r = f / \Delta} \qquad (1)$$

We can also derive the **bandwidth utilization,** P_u , assuming that the link bandwidth is **B**, by equation

$$P_u = \mathbf{r / B} \qquad (2)$$

TCP has three congestion-control methods: **additive increase, slow start,** and **retransmit.**

Chiu and Jain [4] have formulated the congestion avoidance problem as a resource management problem and proposed a distributed congestion avoidance mechanism named 'additive increase/multiplicative decrease' (AIMD). In their work, as a network model they use a "binary feedback" scheme with one bottleneck router (Figure 3).



Synchronous control system model of m users sharing a network.

Figure 3

It consists of a set of m users each of which send data in the network at a rate $2w_i$. The data send by each user are aggregated in a single bottleneck and the network checks whether the total amount of data send by users exceeds some network or bandwidth threshold *goal X* (we can assume that *goal X* is a value between the knee and the cliff and is a characteristic of the network). The system sends a binary feedback to each user telling whether the flows exceed the network threshold. The system response is 1 when bandwidth is available and 0 when bandwidth is exhausted. The feedback sent by the network arrives at the same time to all users. The signal is the same to all users and they take the same action when the signal arrives. The next signal is not send until the users have responded to the previous signal. Such a system is called synchronous feedback system or simply synchronous system. The time elapsed between the arrival of two consecutive signals is discrete and the same after every signal arrival. This time is referred also as RTT. The system behavior can be defined the following time units: a step (or round-trip time – RTT) is the time elapsed between the arrival of two consecutive signals. A cycle or epoch is the time elapsed between two consecutive congestion events (i.e., the time immediately after a system response 0 and ending at the next event of congestion when the system response is again 0). This network model is quite simple and its assumptions have been evaluated in the Internet for several years. In practice the parameter *goal X* is the network capacity (i.e. the number of packets that the link and the routers' buffer can hold – or in-the-fly packets). When the aggregate flows' rate exceeds the network capacity the flows start to lose packets. If the transport protocol provides reliability mechanisms (e.g. as in TCP) it can detect the packet loss or congestion event. Since the majority of the applications use reliable transport protocols (e.g. TCP), the binary feedback mechanism has an implicit presence: a successful data transmissionis interpreted as available bandwidth, and a packet loss is interpreted as congestion event . Although the system had a strong impact on the evaluation of congestion avoidance mechanisms (e.g. AIMD), there are some limitations. First, the system considers the responses to be synchronous, which, in terms of real networks means that all flows have the same RTT. This assumption is not real. A second assumption and limitation is that the network response arrives at the same time to all users, even when they have the same RTT. The above assumption is supported by Jacobson experimentally in a low bandwidth network with congestion avoidance mechanisms (TCP-Tahoe) and where flows have the same RTT . Whatever the argument, this assumption is not true for a reason which is the third limitation of the system. The system has only one bottleneck. In reality a connection might go through none, one, or more than one router or bottlenecks. If a flow traverses more than one bottleneck, then it is not guaranteed that at each

bottleneck congestion will happen at the same time. Nevertheless, these limitations do not prevent the mechanisms from controlling flows' data rate and avoid congestion which was the major concern in the early stages of the Internet .

Streaming media is sensitive to delay and jitter, but can tolerate some data loss. Thus, TCP with reliable transmission service at the cost of potentially large delay at congestion may not be an optimal choice for streaming applications. Recent research has proposed rate-based TCP-Friendly protocols in the hope that streaming media applications will use them, but such protocols are not yet widely part of most operating system distributions. For these reasons, streaming media applications often use UDP as a transport protocol rather than TCP. Moreover, with the use of repair techniques, UDP packet losses can be partially or fully concealed, reducing the impact of loss on the quality of the media by the user, and thus reducing the incentive for multimedia applications to lower their bitrate in the presence of packet loss during congestion. Moreover, as the end-user Internet connection capacity offered by Internet Service Providers (ISP) has significantly increased (up to 3 Mbps for typical cable modem services), even the highest quality media, about 2-4 Mbps for broadcast quality video, can be streamed without imposing congestion at the local Internet connection links. Thus, high-bandwidth Internet connections are pushing the streaming media performance bottleneck closer to the servers threatening the well-being of the public Internet.

4.Additive Increase / Multiplicative Decrease Control algorithm (AIMD) and TCP-friendly protocol for the streaming media

The basic idea of the algorithms to reduce the sending rate/window of the flows when the system bandwidth is exhausted and to increase the sending rates/windows when bandwidth is available. As mentioned in the previous section, when bandwidth is available (i.e. the aggregate rates of the flows do not exceed the network threshold: $\sum_i w_i < goal\ X$) the system attaches the signal 1 to the acknowledgment of each packet. In response, flows increase by one (packet) their windows. A continuous series of positive signals will cause a linear increase in the flows' rate. Obviously, the increase is not unlimited because the bandwidth is fixed. When flows' rate exceed the bandwidth limit (i.e. $\sum_i w_i \geq goal\ X$) the system attaches the 0 signal to the acknowledgment of each packet and flows respond to congestion by a decrease in their sending rates/windows. A. Lahanas and V. Tsaoussidis prove that a linear increase/exponential decrease policy is a condition for the increase/decrease algorithms to set (or converge) quickly the system in a fair state where the load oscillates around some equilibrium. The equilibrium state determines also the fairness and efficiency of the mechanism.

The TCP congestion control is classified as Additive-Increase Multiplicative-decrease (AIMD) mechanism. Following the notation AIMD(a, b), TCP is AIMD(1, 1/2). The parameter a represents the factor to be added to the congestion window each round trip time in absence of congestion, that is congestion_window + a. On the other hand, the parameter b represents the complement to 1 that should be multiplied to the congestion window when congestion is detected, that is (1-b) congestion window.

An AIMD control algorithm may be expressed as:

Increase:

$$W_{t+R} \leftarrow W_t + a\ , a > 0$$

Decrease:

$$W_{t+\delta t} \leftarrow (1-\beta) W_t\ , 0 < \beta < 1$$

A generalizations of AIMD is binomial control:

Increase:

$$W_{t+R} \leftarrow W_t + \frac{a}{W_t^k}\ , a > 0$$

Decrease:

$$W_{t+\delta t} \leftarrow W_t - \beta W_t^l\ , 0 < \beta < 1$$

where "Increase" refers to the increase in window as a result of the receipt of one window of ACKs within a single RTT, "Decrease" refers to the decrease in window upon detection of congestion by the sender, Wt the window size at time t, R the flow's RTT, and a, b, k, l are constants. For example, for k=0, l=1 we get AIMD. There is, in the (k, l) space, two AIMD variations (which are also TCPcompatible).IIAD (with k=1 and l=0) and SQRT (with k=1/2 and l=1/2) algorithms. The first is called Inverse Increase Additive Decrease (IIAD) because its increase rule is in inverse proportion to the current window. The second is called SQRT because both its increase is inversely proportional and decrease proportional to the square-root of the current window. Note that a binomial algorithm is TCP-compatible if and only if $k + l = 1$ and $l \leq 1$ for suitable α and β.

Other Generalizations of AIMD generalizes AIMD congestion control by parameterizing the additive increase value α and multiplicative decrease ratio β. Authors of [15], [7] extended the throughput equation for standard TCP, proposed in [13], to include parameters

$$T_{\alpha,\beta}(p, RTT, T_0, b) = \frac{1}{RTT\sqrt{\frac{2b(1-\beta)}{\alpha(1+\beta)}p} + T_0 \min\left(1, \sqrt{\frac{(1-\beta^2)b}{2\alpha}p}\right) p\left(1+32p^2\right)} \quad (1)$$

where p is the loss rate; T_0 is the retransmission timeout value; b is the number of packets acknowledged by each ACK. The overall throughput of TCP-Friendly (α, β) protocols is bounded by the average throughput of standard TCP (α = 1, β = 0.5), which means that equation (23), which is derived from (22) ([15], [7]) could provide a rough guide to achieve friendliness.

$$T_{a,b}(p, RTT, T_0, b) = T_{1,0.5}(p, RTT, T_0, b) \quad (2)$$

Authors of [15] derive from (1) and (2) a simple relationship for α and β:

$$\alpha = \frac{4(1-\beta^2)}{3} \quad (3)$$

Based on experiments, they proposed a $\beta = 7/8$ as the appropriate value for the reduced the window (i.e., less rapidly than TCP does). For $\beta = 7/8$, (3) gives an increase value $\alpha = 0.31$.

AIMD and binomial controls are memoryless since the increase and decrease rules use only the current window size w_t and constants ($\alpha, \beta, k\ and\ l$). The window size at the end of the last congestion epoch is useful, not only as an indicator of the current congestion level of the network, but also as a good predictor of the congestion state for the next sequence. Thus, our proposed scheme maintains such a state variable $w_{\max}$, which is updated at the end of *each* congestion sequence. In addition, let w_0 denote the window size after the decrease. Given a decrease rule, w_0 can be obtained from w_t, and *vice versa*. For example, for AIMD,

$w_0 = (1-\beta)w_{\max}$. Henceforth, for clarity, we use both $w_{\max}$ and w_0.

We propose to adopt the following window increase function:

$$w(t) = w_0 + c * t^u, u, c > 0 \quad (4)$$

where $w(t)$ is the continuous approximation of the window size at time t (in RTTs- round-trip time) elapsed since the window started to increase. By definition, $w(0) = w_0$. This window increase function is equivalent to the following window increase rule:

$$w_{t+1} \leftarrow w_t + \alpha / (w_t - w_0)^k, \alpha > 0 \quad (5)$$

where $k > -1$ and α is independent of t. In particular, $u = 1/(k+1)$ and $c = ((k+1)\alpha)^u$. We are interested in congestion control schemes that have various window size increase patterns (different $u's$, or equivalently, different $k's$). Consider three cases. First, if $-1 < k < 0$, the congestion window increases super-linearly. The window is increased cautiously just after the detection of packet loss, and the increase becomes more and more aggressive when no more loss occurs. Second, if $k = 0$, the window increases linearly, i.e., additive increase. The aggressiveness does not change with time. Third, if $k > 0$, the window increases sublinearly. The connection approaches the previously probed window size fast, but it becomes less aggressive beyond that. These various schemes possess different degrees of aggressiveness, and may satisfy different applications. For example, super-linear increase can support applications that need to quickly acquire bandwidth as it becomes available. Therefore, we consider the following control rules:

Increase

$$w_{t+1} \leftarrow w_t + \alpha(w_{\max}) / (w_t - w_0)^k, \alpha(w_{\max}) > 0 \quad (6)$$

Decrease:

$$w_t \leftarrow w_t - \beta w_t^l, 0 < \beta < 1$$

Note that we write α as a function of $w_{\max}$ since this is required in the derivation of TCP-friendliness. In the remainder of this paper, we simply write α for clarity. We use the same decrease rule as binomial controls. For the increase rule, we consider $k > -1$, since otherwise the window size increases exponentially or faster and we consider it unstable. For the decrease rule, we consider $l \le 1$, since otherwise $(w_t - \beta w_t^l)$ can be negative when w_t is large enough.

We show that this control can be TCP-friendly by appropriately defining α as a function of the constant β and the state variable $w_{\max}$. This control is radically different from binomial controls, because binomial controls generalize AIMD, but they are still in the memoryless space.

We show that this control scheme using the control rules in (6) can be TCP-friendly. The notion of TCP-friendliness refers to the relationship between throughput and packet loss rate. We consider a random loss model, where the losses are Bernoulli trials; packets are dropped uniformly with a fixed probability, and following definition of α to make congestion control scheme TCP-friendly:

$$\alpha = \frac{3}{2(k+1)\left(1 - \frac{1}{k+2}\beta w_{\max} l - 1\right)} \left(\frac{\beta}{\Gamma\left(\frac{1}{k+1}+1\right)} \right)^{k+1} w_{\max}^{kl+l-1} \quad (7)$$

where the Gamma function $\Gamma(.)$ is a constant. According to formula (4), c in (4) is defined as a function of α and we have

$$c = \left(\frac{3}{2\left(1 - \frac{1}{k+2}\beta w_{\max}^{l-1}\right)} \right)^{\frac{1}{k+1}} \frac{\beta}{\Gamma\left(\frac{1}{k+1}+1\right)} w_{\max}^{l-\frac{1}{k+1}} \quad (8)$$

When the window size variation is small, i.e., the window decrease is small, $\beta w_{\max}^l \le w_{\max}$, we can simplify α and c as

$$\alpha \approx \frac{3}{2(k+1)} \left(\frac{\beta}{\Gamma\left(\frac{1}{k+1}+1\right)} \right)^{k+1} w_{\max}^{kl+l-1} \quad (9)$$

$$c \approx \left(\frac{3}{2}\right)^{\frac{1}{k+1}} \frac{\beta}{\Gamma\left(\frac{1}{k+1}+1\right)} w_{\max}^{l-\frac{1}{k+1}} \quad (10)$$

α is a constant factor of $w_{\max}^{kl+l-1}$ and c is a constant factor of $w_{\max}^{l-(l/k+1)}$. Table 1 gives several special cases, control rules and the window increase functions. When $k = 0$ and $l = 1$, from (7) we have $\alpha_{AIMD} = 3\beta(2-\beta)$. If $\beta \le 1, \alpha_{AIMD} \approx 3\beta/2$ it degenerates to the memoryless TCP-friendly AIMD control. When $k = -0.5$ and $l = 1$ we have square-increase/multiplicative-decrease (SIMD)

$$\alpha_{SIMD} = \frac{3\sqrt{\beta}}{\left(1 - \frac{2\beta}{3}\right)\sqrt{2w_{\max}}} \quad (11)$$

If $\beta \le 1, \alpha_{SIMD} = \frac{3\sqrt{\beta}}{\sqrt{2w_{\max}}}$. In this case, the window size decreases multiplicatively upon the detection of packet loss, but increases in proportion to the *square* of the time elapsed since the detection of the last loss event (cf. Table 1). We call this control square-increase/multiplicative-decrease (SIMD).

(k,l)	Increase rule	Decrease rule	Increase function
$k = 0, l = 1$ *AIMD*	$w_{t+1} = w_t + 3\beta/(2-\beta)$	$w_t = w_t - \beta w_t$	$w(t) = w_0 + (3\beta/(2-\beta))t$

Table 1

$k = -1/2,$ $l = 1,$ *SIMD*	$w_{t+1} = w_t + \frac{3\sqrt{\beta}}{\sqrt{2}\left(1-2\beta/3\right)} x \sqrt{\frac{w_t - w_0}{w_{\max}}}$	$w_t = w_t - \beta w_t$	$w(t) = w_0 + \frac{9\beta}{8\left(1-2\beta/3\right)^2} \frac{1}{w_{\max}} t^2$
$k = 0,$ $l = 1/2$	$w_{t+1} = w_t + 3\beta / \left(2\sqrt{w_{\max}} - \beta\right)$	$w_t = w_t - \beta\sqrt{w_t}$	$w(t) = w_0 + \left(3\beta / \left(2\sqrt{w_{\max}} - \beta\right)\right) t$

5. Conclusion

We proposed a TCP-friendly protocol model for congestion controls. They are TCP-friendly and TCP-compatible under queue management. They possess different smoothness, aggressiveness, and responsiveness tradeoffs. Thus, instances from this applications can be chosen as the transport schemes of various applications, for example, streaming applications on the Internet which are required to be TCP-friendly. In particular, we presented simulation results of SIMD, AIMD as special instances. Analysis were used to demonstrate the TCP-friendliness and TCP-compatibility of congestion controls and they can solve the problem raised by slowly responsive congestion controls.

REFERENCES

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management", IEEE Network, vol. 15, no. 3, May–Jun. 2001, pp. 48 – 53.
[2] P. C. Attie, A. Lahanas, and V. Tsaoussidis, "Beyond AIMD: Explicit fair-share calculation", in Proc. ISCC 2003, Antalya, Turkey, June 2003.
[3] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms", in Proc. IEEE INFOCOM'01, Anchorage, Alaska, USA, April 2001.
[4] D. Chiu and R. Jain, "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks", Journal of Computer Networks and ISDN, vol. 17, no. 1, June 1989, pp. 1–14.
[5] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A New Class of Active Queue Management Algorithms", University of Michigan, Tech. Rep. CSE-TR-387-99, April 1999.
[6] S. Floyd, M. Handley, and J. Padhye, "A Comparison of Equation- Based and AIMD Congestion Control", May 2000.
[7] S. Jin, L. Guo, I. Matta, and A. Bestavros, "TCP-friendly SIMD Congestion Control and Its Convergence Behavior", in Proc. ICNP'2001, Riverside, USA, November 2001.
[8] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker, "Optimization Problems in Congestion Control", in Proc. IEEE Symposium on Foundations of Computer Science, Redondo Beach, USA, November 2000.
[9] A. Lahanas and V. Tsaoussidis, "Additive Increase Multiplicative Decrease - Fast Convergence (AIMD-FC)", in Proc. Networks 2002, Atlanta, USA, August 2002.
[10] A. Lahanas and V. Tsaoussidis, "Exploiting the Efficiency and Fairness Potential of AIMD-based Congestion Avoidance and Control", Computer Networks, Elsevier, vol. 43, no. 2, October 2003, pp. 227–245.
[11] A. Lahanas and V. Tsaoussidis, "_-AIMD for Asynchronous Receiver Feedback", in Proc. IEEE ISCC 2003, Antalya, Turkey, June 2003.
[12] S. Low, F. Paganini, and J. C. Doyle, "Internet Congestion Control", IEEE Control Systems Magazine, vol. 22, 2002.
[13] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", in Proc. ACM SIGCOMM, Vancouver, Canada, August 1998.
[14] D. X. Wei, C. Jin, S. Low, and S. Hedge, "FAST TCP: Motivation, Architecture, Algorithms, Performance", IEEE/ACM Transactions on Networking, 2007, in press.
[15] Y. Yang and S. Lam, "General AIMD Congestion Control", in Proc. IEEE International Conference on Network Protocols (ICNP), Osaka, Japan, November 2000.