Primal Congestion Control Algorithms for Multimedia Streams

Adrian Sterca
Babes-Bolyai University
Department of Computer Science
Cluj-Napoca, M. Kogalniceanu No.1, Romania
forest@cs.ubbcluj.ro

Abstract

We investigate in this paper congestion control algorithms that are TCP-friendly and media-friendly at the same time, so they are more suitable for multimedia streaming than TCP's AIMD. We use in this investigation the optimization framework developed by Kelly et al. in [3]. Kelly et al. derived two gradient-like algorithms, a primal algorithm and a dual one, to control the congestion in a computer network in an end-to-end way and to obtain optimal bandwidth allocation among competing sources. Starting with Kelly et. al's paper, several other papers proposed different variants of the original primal and dual algorithms, but none of these considers the specific characteristics of multimedia streams. The aim of this paper is to build window-based, primal congestion control algorithms for optimizing the perceived quality of multimedia streams in best-effort networks. We build these media-friendly primal congestion control algorithms using the following recipe: 1) first we construct a window-based congestion control algorithm that is TCP-friendly, but has a much more stable throughput and then 2) we shape this stable throughput according to media characteristics (e.g. bitrate). We present two such primal congestion control algorithms and prove their stability around the equilibrium.

1. Introduction

New multimedia capable equipment and improved codec specifications make multimedia presentations most wanted on a wide range of devices from ordinary home computers to different types of cell phones and hand held devices. Due to many practical and economical reasons (great storage demands, costs etc.) the multimedia data will not always be available on the local hard disk, but instead it is stored on dedicated network servers or on the nodes of a peer-to-peer system and is streamed on demand or broadcasted to interested users. Fighting for a large audience, these services are deployed on the biggest net available, the Internet. Consequently, multimedia data make for a high percentage of the data transferred over the Internet lately. However, the heterogeneity and the best-effort nature of the Internet pose great challenges to multimedia streaming applications. This is because no QoS guarantees can be established on the Internet. Due to the time constraints related to multimedia data, multimedia streaming applications need almost isochronous communication and a guaranteed high bandwidth over time and, ideally, they should never suffer from lacking bandwidth. But these can not be achieved due to the best-effort nature of the Internet, so streaming servers must continuously adapt to changing network conditions.

TCP deals with these inconsistencies of the network by implementing congestion control and guaranteed retransmission. But it does this on the expense of bandwidth (i.e., bandwidth is sacrificed for retransmissions) and the timeliness of the transferred data (i.e., it trades timeliness over reliability: it is more important that data arrives safely and in-order than it is to arrive in time). This philosophy is counterproductive for multimedia streams, for which timeliness is more important than reliability. A multimedia data packet that arrives safely at the receiver past its presentation time is useless. To arrive at the destination with a small delay is more important than to arrive at all. Further more, TCP's congestion control algorithm [1] incurs a steep variation in the sending bitrate, a variation that is not well supported by current codecs. Steep degradations in the sending bitrate of a multimedia stream has very bad consequences on the quality perceived by the receiver. Because it does not have a byte-stream semantics and guaranteed retransmissions, UDP is faster than TCP. This is why, multimedia streaming applications often choose to use UDP, instead of TCP, as the transport-level protocol. But applications that do not rely on TCP's congestion control must perform congestion control themselves. If applications fail to do so or choose not to do so, the perceived quality of service at the end user is decreased and it also might lead to congestion collapse, damaging the work of other network flows.

A congestion-unresponsive application also manifests unfairness to other flows competing for the same network resources.

With increasing multimedia traffic on the Internet comes the need for new congestion control mechanisms different than TCP's AIMD (*Additive Increase Multiplicative Decrease*) and more suitable for bandwidth-demanding real-time data. These congestion control mechanisms must be TCP-friendly [12], but they also must consider multimedia streams' characteristics in establishing the sending bitrate (i.e. they must be media-friendly).

The rest of the paper is organized as follows. In section 2 we review related work. Then section 3 provides a description of the methodology we are using for building TCP-friendly and media-friendly primal congestion controls. Section 4 presents two such congestion control algorithms: a *log*-based one and a *sqrt*-based one, and proves they are stable around the equilibrium using a Lyapunov argument. The paper continues with section 5 which deals with packet-level details of the implementation of *mLOG* and *mSQRT* congestion controls and, finally, section 6 which presents simulations of the performance of both congestion control algorithms. The paper ends with conclusions and future work of the author in section 7.

2. Background and Related Work

Kelly et al.'s seminal paper [3] generated in latest years a significant amount of active research on Internet congestion control derived from optimization theory (see [14] for a survey). Kelly et al. derived two gradient-like algorithms, a primal algorithm and a dual one, to control the congestion in the Internet in an end-to-end way and to obtain optimal bandwidth allocation among competing sources. A primal congestion control algorithm is an algorithm which uses a dynamic law (i.e. differential equation) to control the throughput at the source and a static function for computing the price of using the network (i.e. congestion price/measure incurred on the network by bandwidth demands of flows using the network) at the links. A dual algorithm does the opposite. Starting with Kelly et. al's paper, several other papers proposed different variants of the original primal and dual algorithms [4, 5, 6, 7, 8, 9, 10, 11] and proved their stability and speed of convergence properties, but none of these considers the specific characteristics of multimedia streams.

Authors develop in [13] a media- and TCP-friendly congestion control based on TFRC (*TCP-Friendly Rate Control*) [12] using a two-timescale approach: they compute the long term average of throughput according to TFRC, but they modify this throughput on a smaller timescale according to the rate of increase/decrease of a utility function obtained from the rate-distortion characteristics of the stream. We have two observations related to this paper. First, the utility function that is used was developed for MPEG FGS (Fine Granularity Scalable) video streams, so the algorithm does not seem to work for other types of streams. Second, since the rate-distortion utility function is not scaled with TFRC's throughput we are not sure that the derivative of the utility function will have significant influence on TFRC's throughput (even on small timescales) in all network scenarios.

3. Two-step Methodology for Building TCP-friendly and Media-friendly Congestion Controls

TCP's AIMD congestion control is not well suited for multimedia streams [1] due to its highly fluctuating throughput. Consequently, other congestion control algorithms which offer smoother throughput were developed, perhaps the most well known being TFRC [12]. All these smooth congestion controls have more stable throughput than TCP's AIMD because they are less aggressive than TCP in using new available bandwidth, but they are also slower responsive to congestion than TCP. Because they offer a more stable throughput, multimedia streams, especially CBR (*Constant Bit Rate*) ones, but also VBR (*Variable Bit Rate*) ones, can be better adapted to predictable bandwidths by the streaming servers. However, although smooth congestion controls improve the delivery of multimedia streams, they are not the optimal solution, because they do not take into consideration media characteristics of the stream (i.e. they are not media-friendly).

On the other hand, VBR codecs compliant to MPEG video coding standards can vary a lot the output bitrate of a video stream between scene changes in order to preserve a relatively constant quality throughout the video (e.g. the bitrate can vary 20 times from one stream second to the next one) [15]. In other words, the bitrate of such a stream is certainly not smooth.

If the bitrate of a stream is not smooth and, generally, if media characteristics are not smooth (relatively constant) across the stream, then perhaps the best send rate a congestion control can give is not necessarily a smooth one, but a send rate that tracks the evolution of media characteristics (e.g. bitrate) across the stream. Of course, this send rate must also obey network-related characteristics (i.e. must have a TCP-friendly shape).

In this paper we use the following technique to obtain a TCP-friendly and media-friendly primal congestion control:

- first we construct a window-based congestion control algorithm that is TCP-friendly, but has a much more stable throughput;

- we shape this stable throughput according to media characteristics (e.g. bitrate).

Using this methodology we will develop in the next section two such congestion control algorithms: mSQRT and mLOG.

4. Primal Congestion Control for Multimedia Streams

We are using the network model developed in [3] where the network is seen as a set of resources or links which are shared by a set of sources or users. The goal is to split bandwidth among sources in such a way that a social optimum is attained for all users sharing the network.

The problem of bandwidth allocation among flows reduces to finding the solution to the following concave optimization problem:

$$\begin{cases} \max_{x>0} \sum_{s\in S} U_s(x_s) & , x = (x_1, ..., x_n) \\ & S = \{s_1, ..., s_n\} \\ \text{subject to: } \sum_{s\in S(l)} x_s \leq c_l & \forall l \in L \end{cases} \quad (1)$$

In this model the network is abstracted as a set of links $l \in L$ and each link l has the capacity c_l. The network is shared by sources $s \in S$ and each source s transmits data at rate x_s. When the source s sends data at rate x_s, it gets a utility $U_s(x_s)$ which is assumed to be a non-decreasing concave function twice differentiable. Also, let $S(l)$ denote the set of sources which use link $l \in L$ and $L(s)$ the set of links that source s uses.

In order to build a TCP-friendly and media-friendly congestion control, we need first, according to the methodology outlined in section 3, a primal congestion control which is TCP-friendly and achieves a smoother throughput than TCP's AIMD. We start with TCP's AIMD algorithm which can be expressed as [2]:

$$\begin{cases} I : W_{t+RTT} = W_t + \alpha & , \alpha = 1 \\ D : W_{t+\delta} = W_t - \beta W_t & , \beta = 1/2 \end{cases}$$

where W_t is the congestion window at time t, RTT is the round-trip time for that connection and δ is the length of a small time interval containing a congestion event. I stands for increase operation and D for decrease operation. The evolution of TCP's congestion window can also roughly be described by the following equation which combines the above two operations:

$$W_{t+\delta} = W_t + \frac{\alpha}{W_t}(1 - q_t) - \beta W_t q_t \quad (2)$$

assuming δ is much smaller than RTT and q_t is the probability that a packet is dropped in time interval $[t, t + \delta]$. If we multiply this equation by $1/RTT$ and we consider the throughput at time t to be $x_t \approx W_t/RTT$ we get:

$$x_{t+\delta} = x_t + \frac{\alpha}{RTT\, W_t}(1 - q_t) - \beta x_t q_t$$

Then by taking δ to be small and dividing both sides of equation by it, we obtain the differential (rate) evolution of the throughput:

$$\dot{x}(t) = \frac{\alpha}{\delta RTT^2 x(t)}(1 - q(t)) - \beta x(t) q(t)$$

After considering δ equal to a time unit and dropping the RTT^2 term as we consider it a constant and it can be incorporated into the α constant, we interpret $q(t)$ as the price charged by the network for using $x(t)$ of its resources [3] (when there is no congestion $q(t) = 0$) and we obtain the final form for the AIMD throughput evolution:

$$\dot{x}(t) = \frac{\alpha}{x(t)} - \beta x(t) q(t) \quad (3)$$

Next, we need to smooth out the TCP's throughput evolution given in equation (3), but at the same time to keep it TCP-friendly. Generally, we are looking for a primal congestion control of the following form:

$$\dot{x}(t) = \frac{\alpha}{I(x(t))} - \beta D(x(t)) q(t) \quad (4)$$

where $I(x)$ is the increase factor and $D(x)$ is the decrease factor and which has a smoother throughput evolution than the one of the control given in (3).

The mSQRT Control

The mSQRT control is a smooth TCP-friendly congestion control introduced in [2] which has the following form:

$$\dot{x}(t) = \frac{\alpha}{\sqrt{x(t)}} - \beta\sqrt{x(t)} q(t)$$

The mSQRT control is smoother than TCP as it increases throughput less aggressively than TCP (i.e. $\frac{\alpha}{\sqrt{x(t)}}$), but also it decreases it less drastically than TCP (i.e. $\beta\sqrt{x(t)}$).

Next, we need to make mSQRT's throughput track the media characteristics of the stream. We will use here only bitrate and client prefetch buffer values, but other media characteristics can be used too: bitrate averaged over a scene, quality measures, PSNR values etc. So, the final version of our mSQRT which is smooth TCP-friendly, but also media friendly, formulated as a primal congestion control is:

$$\begin{cases} \dot{x_s}(t) = m_s(t) * \frac{\alpha}{\sqrt{x_s(t)}} - \beta\sqrt{x_s(t)} q_s(t) \\ where\ m_s(t) = \frac{b(t)}{b_{avg}}\left(1 + \frac{1}{\Delta(t)}\right) \end{cases} \quad (5)$$

$$q_s(t) = \sum_{l \in L(s)} p_l \left(\sum_{r:r \in S(l)} x_r(t) \right) \quad (6)$$

where $x_s(t)$ is the throughput of source s, $b(t)$ is the bitrate measured in bytes for the current second of stream, b_{avg} is the average bitrate over the whole stream and $q_s(t)$ is given in the form from [3] as the total price for using network resources from $L(s)$; $p_l(y)$ is a continuous, non-negative increasing function of y, the price for using only resource l. If $y < c_l$ then $p_l(y) = 0$. This way, mSQRT increases its throughput when there is no congestion according to the bitrate needs of the stream and also it increases more when the buffer $\Delta(t)$ is small to avoid an empty buffer at the client. Function $m_s(t)$ quantifies the utility or the need for bandwidth of the streaming application.

It can be proved that the control given in (5) is stable and the next proposition establishes this.

Proposition 1. The strictly concave function

$$V_1(x) = \sum_{s \in S} \alpha\, m_s(t)\, log(x_s(t)) - \beta \sum_{l \in L} \int_0^{\sum_{r:r \in S(l)} x_r(t)} p_l(y) dy$$

is a Lyapunov function for the dynamical system (5)-(6), hence the system is stable around the equilibrium.

The stability proof follows the one given in [3].

Proof: We first observe that $V_1(x)$ is strictly concave on $x \geq 0$ (note that $m_s(t) > 0 \, \forall t$) and also $V_1(x) \to -\infty$ for $||x|| \to 0$. Next, all we have to do is verify that $\dot{V}_1(x) > 0$. We have

$$\frac{\partial V_1}{\partial x_s}(x) = \frac{\alpha\, m_s(t)}{x_s(t)} - \beta \sum_{l \in L(s)} p_l \left(\sum_{r:r \in S(l)} x_r(t) \right)$$

$$\frac{\partial V_1}{\partial t}(x(t)) = \sum_{s \in S} \frac{\partial V_1}{\partial x_s} \frac{\partial x_s}{\partial t}(t) = \sum_{s \in S} \sqrt{x_s(t)} \left(\frac{\alpha\, m_s(t)}{x_s(t)} - \beta \sum_{l \in L(s)} p_l \left(\sum_{r:r \in S(l)} x_r(t) \right) \right)^2$$

Because $V_1(x)$ is strictly concave on $x \geq 0$, it has a unique maximum and since $\dot{V}_1(x) \geq 0$ with equality only for the unique x maximizing $V_1(x)$, $V_1(x)$ is strictly increasing with time up to its maximum and once it reaches this maximum, it stays there as t goes to ∞. We can conclude the system (5)-(6) is asymptotically stable in the sense of Lyapunov. The unique value maximizing $V_1(x)$ is a stable point of the system (5)-(6) to which all trajectories converge.

The mLOG Control

The mLOG control is the following smooth TCP-friendly congestion control:

$$\dot{x}(t) = \frac{\alpha}{log(x(t))} - \beta log(x(t)) q(t)$$

The mLOG control is smoother than TCP as it increases throughput less aggressively than TCP (i.e. $\frac{\alpha}{log(x(t))}$), but also it decreases it less drastically than TCP (i.e. $\beta log(x(t))$). Taking into account media characteristics like we did for mSQRT we obtain the primal form of the mLOG congestion control:

$$\begin{cases} \dot{x_s}(t) = m_s(t) * \frac{\alpha}{log(x_s(t))} - \beta log(x_s(t)) q_s(t) \\ where\ m_s(t) = \frac{b(t)}{b_{avg}} \left(1 + \frac{1}{\Delta(t)} \right) \end{cases} \quad (7)$$

$$q_s(t) = \sum_{l \in L(s)} p_l \left(\sum_{r:r \in S(l)} x_r(t) \right) \quad (8)$$

Proposition 2. The strictly concave function

$$V_2(x) = \sum_{s \in S} \alpha\, m_s(t) \left(li(x_s(t)) - \frac{x_s(t)}{log(x_s(t))} \right) - \beta \sum_{l \in L} \int_0^{\sum_{r:r \in S(l)} x_r(t)} p_l(y) dy$$

where $li(x_s(t))$ is the logarithmic integral [1] is a Lyapunov function for the dynamical system (7)-(8), hence the system is stable around the equilibrium.

Proof: The proof is similar to the one for mSQRT, the only differences being the derivatives of $V_2(x)$ and $x_s(t)$:

$$\frac{\partial V_2}{\partial x_s}(x) = \frac{\alpha\, m_s(t)}{log^2(x_s(t))} - \beta \sum_{l \in L(s)} p_l \left(\sum_{r:r \in S(l)} x_r(t) \right)$$

$$\frac{\partial V_2}{\partial t}(x(t)) = \sum_{s \in S} \frac{\partial V_2}{\partial x_s} \frac{\partial x_s}{\partial t}(t) = \sum_{s \in S} log(x_s(t)) \left(\frac{\alpha\, m_s(t)}{log^2(x_s(t))} - \beta \sum_{l \in L(s)} p_l \left(\sum_{r:r \in S(l)} x_r(t) \right) \right)^2$$

5. Implementation Details and Fairness Issues

The first step in getting a practical implementation of mLOG and mSQRT is the discretization of differential equations (5) and (7). We get the following discrete versions (we drop the "s" subscript for convenience):

$$mSQRT : x[t+1] = x[t] + m(t) * \frac{\alpha}{\sqrt{x(t)}} - \beta \sqrt{x(t)} q(t)$$
$$mLOG : x[t+1] = x[t] + m(t) * \frac{\alpha}{log(x(t))} - \beta log(x(t)) q(t)$$
$$where \quad m(t) = \frac{b(t)}{b_{avg}} \left(1 + \frac{1}{\Delta(t)} \right)$$

After considering $q(t)$ as the probability of a lost event in time interval $[t, t+1]$, we get the following window-based characterizations of mLOG and mSQRT which are more useful for a packet-level implementation:

mSQRT:

$$\begin{cases} I : W_{t+RTT} = W_t + m(t) * \frac{\alpha}{sqrt(W_t)} \\ D : W_{t+\delta} = W_t - \beta sqrt(W_t) \end{cases}$$

[1] $li(x) = \int_0^x \frac{dt}{log(t)}$

mLOG:

$$\begin{cases} I : W_{t+RTT} = W_t + m(t) * \frac{\alpha}{log(W_t)} \\ D : W_{t+\delta} = W_t - \beta log(W_t) \end{cases}$$

Using the above window-based characterizations we can specify the increment and decrement rules for mSQRT and mLOG which are similar to TCP's AIMD ones. For every acknowledged packet mSQRT increments it's congestion window by $m(t) * \frac{\alpha}{W_t sqrt(W_t)}$ and after a congestion indication it decrements the congestion window by $\beta sqrt(W_t)$. For mLOG, every acknowledged packet increments the congestion window by $m(t) * \frac{\alpha}{W_t log(W_t)}$ and a congestion indication decrements it by $\beta log(W_t)$. In the next section, during simulations, we have used for α and β the same values as TCP's AIMD uses for these factors ($\alpha = 1, \beta = 1/2$).

Although we have proved in the previous section that both controls are stable around the equilibrium, it remains to be tested whether mLOG and mSQRT are fair to TCP and are fair to themselves (i.e. a mSQRT/mLOG flow uses the same amount of bandwidth as another mSQRT/mLOG flow sharing the same network conditions). Because we encountered some unfairness issues during our simulations, which will be further detailed, we suggest for a practical implementation of mLOG and mSQRT, the following scaled version of the media-friendly function $m(t)$:

$$m(t) = k * \frac{b(t)}{b_{avg}} \left(1 + \frac{1}{\Delta(t)}\right) \quad (9)$$

where k is a scaling factor. During our simulations we have come to the following rules for assuring fairness of mSQRT and mLOG:

- the media-friendly function, $m(t)$, should participate at the increment of the congestion window at most once per RTT, otherwise the control tends to use more bandwidth or less bandwidth than TCP;
- the value of $m(t)$ should be "very close to 1" in order to achieve fairness, especially if $m(t) > 1$ and the congestion window is large; we have obtained good results when the media characteristics are scaled with the inverse of the congestion window (i.e. $k = 1/W_t$).

6. Simulations

In this section, we present some simulations we have conducted using ns-2 [16] in order to prove mSQRT's and mLOG's usefulness for multimedia streaming applications. We have used a network topology consisting of a single 9.5Mbps congested link with one-way delay of 50ms and which uses RED (Random Early Detection) queue management with default parameter settings. This link is shared by 11 long-run TCP-Reno flows and 2 multimedia streaming flows. These 2 multimedia streaming flows will use, during a simulation, either mSQRT or SQRT (mSQRT without the media-friendly, $m(t)$, factor) or mLog or LOG (mLOG without the media-friendly, $m(t)$, factor) for controlling congestion. We have also deployed 4 TCP-Reno flows in the reverse direction sharing the bottleneck link to eliminate any synchronization between flows.

The bitrate values ($b(t)$ and b_{avg}) we have used for computing $m(t)$ are taken from a real MPEG-4 video stream and are depicted in Fig. 1. Also, in the case of mSQRT and mLOG we used $m(t)$ to increment/decrement the congestion window once per 5 RTTs. An average RTT for our setting was 110 ms. The value of $m(t)$ was scaled, as specified in the previous section, in order to achieve fairness to $1/W_t$ for mSQRT and $1/50$ for mLOG.

First, we have tested the variations in congestion window of a TCP flow against the variations obtained when using SQRT and LOG. As we can see in Fig. 2 and Fig. 3, as expected, SQRT and LOG have smoother variations of congestion window, thus, they get more stable throughput than TCP which makes them more suitable for streaming applications because a smooth throughput is better supported by codecs than a highly fluctuating one. Also we can see that both LOG and SQRT are fair to TCP (i.e. they get approximately the same bandwidth as TCP) with a slight unfairness manifested by LOG.

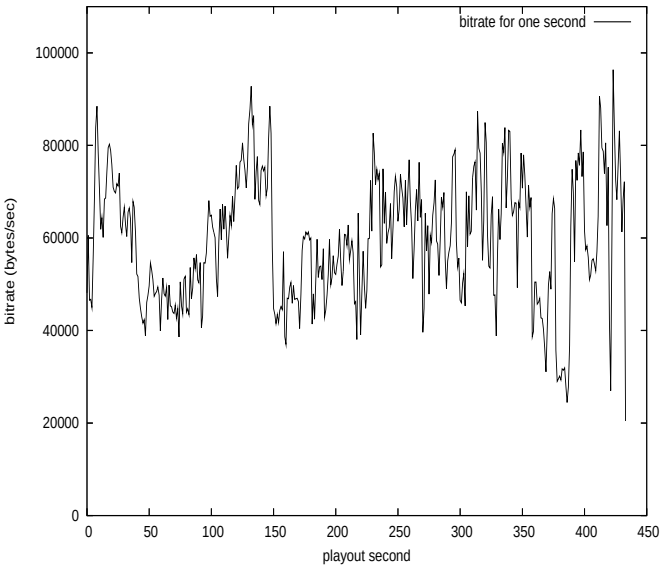

Figure 1. The 1 second bitrates of the stream

In the next simulations, we wanted to see how much does the media-friendly factor, $m(t)$, influences the throughput of SQRT and LOG. Therefore, we ran the simulation again and we used first mSQRT for the two multimedia flows and then on a second simulation we used mLOG. We compared

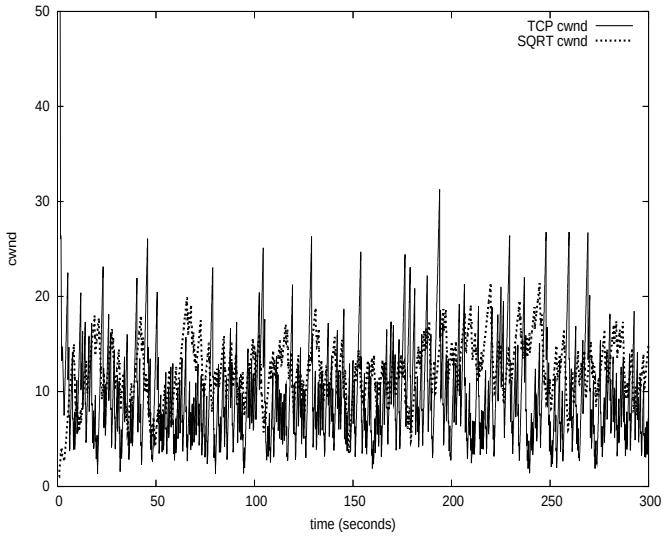

Figure 2. Congestion window evolution for TCP and SQRT

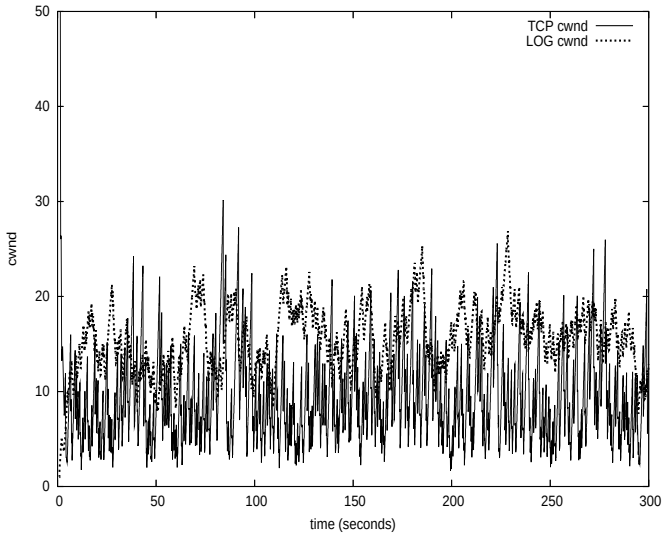

Figure 3. Congestion window evolution for TCP and LOG

the evolution of congestion window of mLOG and mSQRT, respectively, to the evolution of congestion window of LOG and SQRT, respectively, obtained in the previous simulations. These comparisons are depicted in Fig. 4 and Fig. 5.

The final simulations tries to prove that mSQRT and mLOG are better for multimedia streaming applications than TCP. This is especially true for live broadcasting or videoconferencing applications where we have to maintain a low buffer, so when the throughput of the application follows its bitrate demand, we get a higher chance to avoid an empty buffer at the client. We have implemented the buffer consumption of the stream from Fig. 1 into the ns-2 simulator and the size of buffer obtained when TCP is used for streaming is compared to the size of buffer when mSQRT is used for streaming. As we can see in Fig. 6, TCP gets an empty buffer several times as opposed to mSQRT, thus causing the stream to freeze at the client in a real world scenario.

7. Conclusions and Future Work

We have presented in this paper a methodology for building TCP-friendly congestion control algorithms suitable for multimedia streaming applications and derived two such algorithms: a *log*-based one and a *sqrt*-based one. We also proved that these two congestion control algorithms are stable around the optimal bandwidth allocation. Detailed simulations showed they are more suitable for multimedia streaming than TCP. However, to fully assess their value and limitations they must be implemented and tested in real streaming scenarios. As future work, we plan to test mLOG and mSQRT in real streaming scenarios and to devise a general rule for choosing the value of the scaling factor k which should fit all possible network topologies and bandwidths.

8. Acknowledgments

This work was partially supported by CNCSIS's National Program II grants no. TD-371/2007 and P4 11-052/2007.

References

[1] V. Jacobson, *Congestion avoidance and control*, ACM Comput. Commun. Rev., vol. 18, pp. 314-329, 1988.

[2] D. Bansal, H. Balakrishnan, *Binomial Congestion Control Algorithms*, IEEE Infocom 2001.

[3] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, *Rate control for communication networks: Shadow prices, proportional fairness and stability*, J. Oper. Res. Soc., vol. 49, no. 3, pp. 237-252, Mar. 1998.

[4] S. H. Low and D. E. Lapsley, *Optimization flow control I: Basic algorithm and convergence*, IEEE/ACM Transactions on Networking, vol. 7, pp. 861-874, Dec. 1999.

[5] K. Kar, S. Sarkar, and L. Tassiulas, *A simple rate control algorithm for maximizing total user utility*, in Proc. IEEE INFOCOM, Apr. 2001, pp. 133-141.

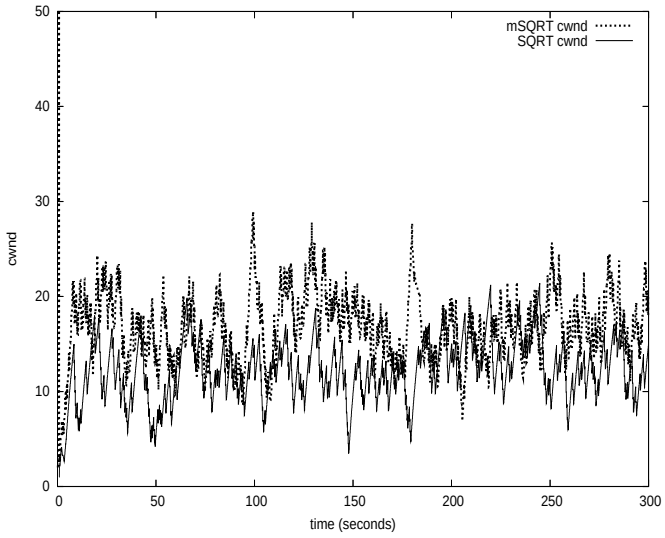

Figure 4. Congestion window evolution for SQRT and mSQRT

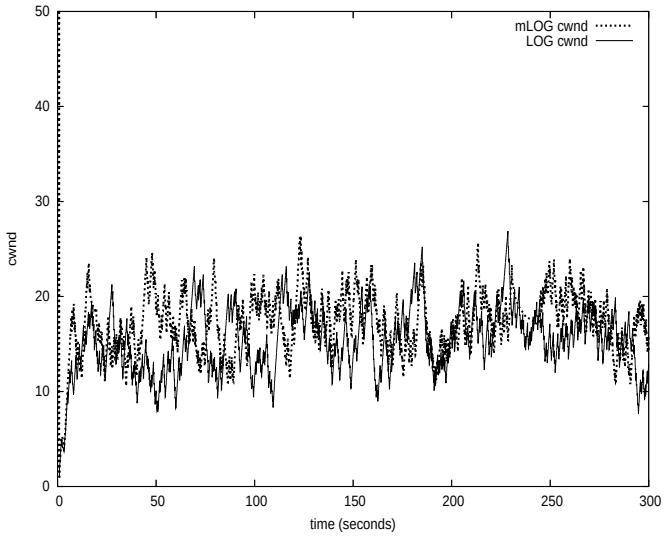

Figure 5. Congestion window evolution for LOG and mLOG

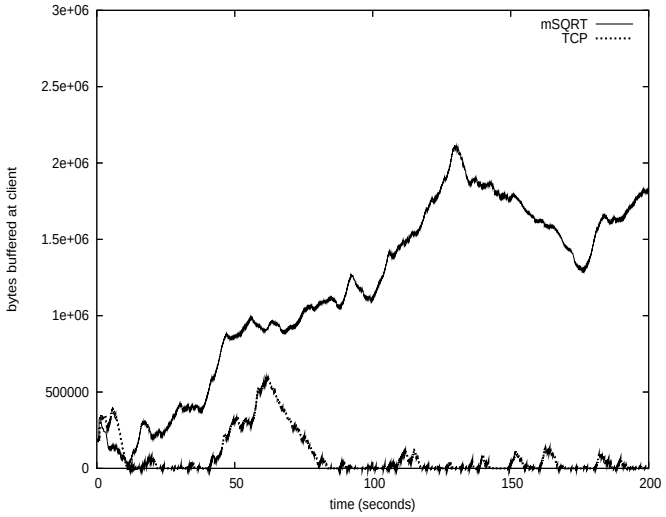

Figure 6. Buffer size for a TCP flow and a mSQRT flow

[6] R. Johari and D. K. H. Tan, *End-to-End congestion control for the Internet: Delays and stability*, IEEE/ACM Transactions on Networking, vol. 9, no. 6, pp. 818-832, Dec. 2001.

[7] S. Kunniyur and R. Srikant, *End-to-End congestion control schemes: Utility functions, random losses and ECN marks*, IEEE/ACM Transactions on Networking, vol. 11, no. 5, pp. 689-702, Oct. 2003.

[8] G. Vinnicombe, *On the stability of end-to-end congestion control for the Internet*, Cambridge Univ., Cambridge, U.K., Tech. Rep. CUED/F-INFENG/TR.398, Dec. 2000.

[9] G. Vinnicombe, *On the stability of networks operating TCP-like protocols* , in Proc. IFAC, Aug. 2002.

[10] Y. Zhang, S. R. Kang, and D. Loguinov, *Delay-Independent Stability and Performance of Distributed Congestion Control*, IEEE/ACM Transactions on Networking, vol. 15, no. 5, pp. 838-851, Oct. 2007.

[11] L. Ying, G. E. Dullerud, and R. Srikant, *Global Stability of Internet Congestion Controllers With Heterogeneous Delays*, IEEE/ACM Transactions on Networking, vol. 14, no. 3, pp.579-591, Jun. 2006.

[12] S. Floyd, M. Handley, J. Padhye, J. Widmer, *Equation-Based Congestion Control for Unicast Applications*, ACM SIGCOMM 2000.

[13] J. Yan, K. Katrinis, M. May, B. Plattner, *Media- and TCP-Friendly Congestion Control for Scalable Video Streams*, IEEE Transactions on Multimedia, Vol. 8, No. 2, April, 2006.

[14] R. Srikant, *The Mathematics of Internet Congestion Control*, Cambridge, MA: Birkhauser, 2004.

[15] F. Pereira, T. Ebrahimi, *The MPEG-4 Book*, Prentice Hall PTR, ISBN 0130616214, 9780130616210, 2002.

[16] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns.