# UTFRC - Utility-driven TCP-Friendly Rate Control for Multimedia Streams

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

## Abstract

*This paper describes UTFRC - Utility-driven TCP-Friendly Rate Control, a congestion control mechanism more suitable for streaming layered scalable video streams in best-effort networks than TFRC [2]. UTFRC relies on the original TFRC for achieving a stable throughput, but uses the throughput outputted by TFRC only as a guideline and shapes this throughput, on a coarser granularity scale, according to media characteristics of the video stream. This way, UTFRC is TCP-friendly when computing available bandwidth, but it is also media-sensitive (i.e., media-friendly). We present two versions of UTFRC, a simple one based only on bitrate properties of the video stream and an advanced one which takes into account also the client prefetching buffer value, but other media characteristics of the stream can also be considered. UTFRC improves the perceived quality of video streams in best-effort network conditions.*

## 1. Introduction

In the last decade, multimedia communication and real-time streaming over the Internet has received much attention from the scientific community. Due to their high bandwidth demands and almost isochronous communication, multimedia streaming applications like VoD (Video on Demand), video conferencing and live broadcasting applications still face many challenges nowadays. The Internet does not guarantee a constant level of service because of its heterogeneity and best-effort nature, forcing streaming applications to continuously adapt their bitrate/quality demands to changing QoS parameters of the network (i.e., they must perform congestion control). 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.

TCP's AIMD (Additive Increase Multiplicative Decrease) congestion control is not well suited for multimedia streams [8] due to its highly fluctuating throughput. Consequently, other congestion control algorithms which offer a smoother throughput were developed [1], [4], [5], [6], [7], perhaps the most well known being TFRC (*TCP-Friendly Rate Control*) [2]. All these smooth congestion controls have a 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 don't take into consideration media characteristics of the stream (i.e. they are not media-friendly).

The rest of the paper is organized as follows. In section 2 we review related work. Then section 3 provides an intuitive description of the ideas and goals behind Utility-driven TFRC. The main body of the paper is in sections 4 and 5 which describe in details a simple Utility-driven TFRC and an advanced Utility-driven TFRC, respectively. The paper continues with section 6 which describes experiments that prove UTFRC usefulness for multimedia streaming and ends with conclusions in section 7.

## 2. Related work

The TCP-Friendly Rate Control [1], [2] is a rate-based congestion control that has two main components: the throughput function and the WALI (i.e., Weighted Average Loss Intervals) mechanism for computing the loss rate. The throughput function is the throughput equation of a TCP-Reno source [3]:

$$X(p) = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}, \qquad (1)$$

where $X$ is the sending rate in bytes/sec, $s$ is the packet size, $R$ is the round-trip time (RTT), $p$ is the steady-state loss event rate and $t_{RTO} = 4 * R$ is the TCP retransmit timeout value. This throughput function is behind TCP-friendliness of TFRC. WALI, the mechanism for computing the loss rate as a weighted average of the last 8 loss intervals, is responsible for the smoothness of throughput. Studies reveal that indeed TFRC's throughput is smoother than the

throughput of TCP [9], [10], [11], but it also has some limitations [12], [13] which will be further discussed in the next section. We would like to mention that other proposals for smooth TCP-friendly congestion control exist [4], [5], [6], [7].

The work presented in [14] is the closest to our work. Authors develop a media- and TCP-friendly congestion control based on TFRC 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 an utility function obtained from the rate-distortion characteristics of the stream. We have two observations related to this paper. First, the utility function it is used was developed for MPEG FGS (Fine Granularity Scalable) video streams, so the algorithm does not seem to work for other type of streams, while UTFRC is general and works for any video streams (i.e. it can use various stream characteristics). 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.

A total different approach in congestion control derived from optimization theory is taken in [15], [16], [17], [18]. Kelly et al. formulates the problem of sharing bandwidth in a best-effort network as an optimization problem [15] and derives two gradient-like algorithms, a primal algorithm and a dual, to control the congestion in the Internet in an end-to-end way and to obtain optimal bandwidth allocation among competing sources. The primal algorithm is a window-based congestion control, while the dual is rate-based. Low et al. also provides a dual-type algorithm for achieving optimal bandwidth allocation and congestion control [16]. However, all aforementioned papers use general utility functions and they don't consider the specific characteristics of multimedia streaming applications.

## 3. Motivation and rationale behind UTFRC

TCP throughput equation (1) used in TFRC is supposed to be an upper bound on the sending rate of a TCP flow. However, several arguments determine us to say that this TCP throughput equation should be thought only as a guideline (not upper bound nor lower bound) of the throughput achieved by a TCP flow:

- the TCP throughput equation is derived in [3] under simplified assumptions;
- the TCP throughput equation characterizes a TCP Reno flow, while other TCP flows like TCP-Sack can be more aggressive and TCP-Reno is not used any more in modern TCP stacks;
- the loss rate $p$ from the equation is computed differently in [3] than the loss rate from TFRC;
- Milan Vojnovic et al. [19] present some experiments which show that sometimes TCP overshoots or undershoots the throughput predicted by the TCP Reno equation;

Also, because the WALI mechanism computes the loss rate as an average over several RTTs, thus making TFRC's throughput a smoothed average version of the throughput achieved by a TCP connection in the same network conditions, the instantaneous throughput and sometimes the long-run throughput of TFRC can be sometimes smaller, sometimes higher than the throughput of a TCP flow [12].

Hence, the throughput given by TFRC can be considered as a guideline throughput for a smoothed averaged TCP throughput.

In the view of the aforementioned observations, we argue that for a multimedia streaming application following blindly the transmission rate given by TFRC is good from a network perspective and *good but not optimal* from the application's perspective. In other words, TFRC is too much "network friendly" and less "media-friendly".

On the other hand, VBR codecs like MPEG can vary a lot the outputted bitrate of a video stream between scene changes in order to preserve a relative constant quality throughout the video (e.g. the bitrate can vary 20 times from one stream second to the next one) [20]. In other words, the bitrate of such a stream is certainly not smooth. You can see in Fig. 1 the bitrate evolution for a part (between stream seconds 40 and 180) of a typical MPEG-4 video stream.

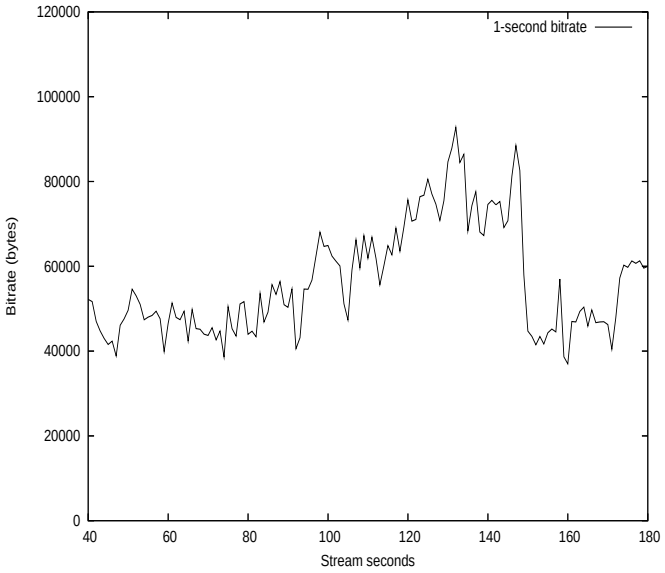


Figure 1. The 1-second bitrates of a typical MPEG-4 stream

Please note that bitrate variability is higher for other video streams with more abrupt scene changes. 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 transmission rate a congestion control can give is not necessarily a smooth one, but a transmission rate that tracks the evolution of media characteristics (e.g. bitrate) across the stream. Of course, this transmission rate must also obey network-related characteristics (i.e. must have a TCP-friendly shape). This is especially true for live streaming that doesn't afford to maintain a reasonable prefetch buffer at the client. An intuitive example of what we have in mind can be seen in Fig. 2 where the bitrate of the stream is used as an utility vector.

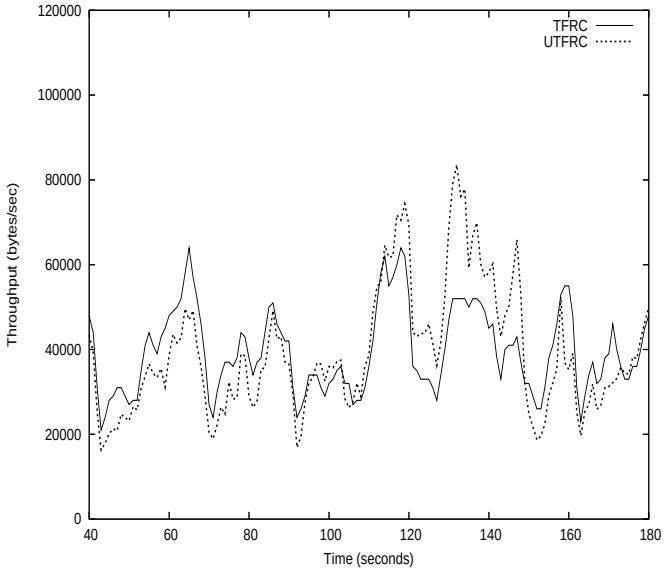


Figure 2. TFRC throughput modified according to the bitrate of the stream

In Fig. 2 we see with continuous line a typical evolution of the transmission rate (throughput) of a TFRC flow and with interrupted line the same throughput modified to track the evolution of bitrate of the stream from Fig. 1. We can see that TFRC's throughput is increased between seconds 112-150 because the bitrates of seconds 112-150 are higher in Fig.1, but in the same time, throughput is decreased in seconds 42-90, because the bitrates between these seconds are low in Fig. 1.

In the lines above we have used only a simple metric of utility - that is the 1-second bitrates of the stream, but other media characteristics can be used in quantifying the utility of a throughput: bitrate averaged over a scene, quality measures, PSNR values, client buffer value etc. In conclusion, by the name UTFRC (*Utility-driven TCP-Friendly Rate Control*) we refer to a congestion control which computes the transmission rate in the following way:

$$X_{UTFRC}(t) = U(q) * X_{TFRC}(t) \tag{2}$$

where $t$ is time, $X_{TFRC}(t)$ is the transmission rate computed by TFRC at time $t$ using equation (1), $U(q)$ is an utility function with values in the interval $[0.8, 1.2]$ and $q$ is a vector variable that includes various media characteristics. The function $U(q)$ embodies the usefulness of increasing TFRC's throughput passed the rate computed with (1) to the streaming application. We have chosen the interval $[0.8, 1.2]$ because we think that the increase or decrease of TFRC's throughput should be within maximum 20 % of the throughput in order to remain TCP-friendly on the short term. We haven't studied utility values greater than 1.2 or smaller than 0.8.

We want to make sure that our congestion control algorithm described by equation (2) remains TCP-Friendly on the long term. That is, if we have two flows sharing the same network path and the first flow uses UTFRC, but the second one uses TFRC, we want to make sure that the bandwidth used by the first flow is approximately equal to the bandwidth used by the second flow throughout their existence, assuming they last a significant period of time. If we consider the transmission rate instances of both TFRC and UTFRC and we take the expectations of both we get:

$$E[X_{UTFRC}(t)] = E[U(q) * X_{TFRC}(t)] = \\ E[U(q)] * E[X_{TFRC}(t)]$$

where the last equality results from the fact that $U(q)$ is statistically independent of $X_{TFRC}(t)$ as the first one includes only media parameters and the second one includes only network parameters. If we further choose the utility function $U(q)$ in such a way that $E[U(q)] = 1$ then the expected transmission rate (bandwidth) of TFRC is equal with the expected transmission rate (bandwidth) of UTFRC.

## 4. Simple utility-driven TFRC

A very simple variant of utility-driven TFRC can be obtained by multiplying the throughput computed by TFRC with the value $\frac{b}{b_{avg}}$, where $b$ is the bitrate measured in bytes for the current second of stream and $b_{avg}$ is the average bitrate over the whole stream (This rule was used in the throughput of UTFRC from Fig.2, for example).

$$X_{UTFRC}(t) = \left(\frac{b_i}{b_{avg}}\right) X_{TFRC}(t) \tag{3}$$

for the i-th stream second.

Among the advantages of this simple UTFRC we mention: it is very simple to implement and is not time consuming during streaming, especially if the bitrate for each second is computed off-line (the average bitrate has to be computed off-line). However it has some disadvantages: it doesn't consider other media characteristics besides bitrate like client buffer value, PSNR and also, we don't have a strict control over the increase and decrease of TFRC's throughput - that is, it is very possible that $\frac{b}{b_{avg}}$ is greater than 1.2 or smaller than 0.8 and we think that increase or decrease

should be within maximum 20 % of the throughput in order to remain TCP-friendly on the short term.

## 5. Advanced utility-driven TFRC - UTFRC

A more advanced utility-driven TFRC can be built if we consider also the client buffer value besides bitrate in the following intuitive way:

- if the client prefetch buffer is small (e.g. smaller than a threshold), the utility should be high, because if we don't have a large enough throughput we might get an empty buffer at the client and stream playing can freeze;
- if the buffer is large then the utility should be small, but still it should follow the slope of the bitrate;

The throughput of our advanced UTFRC is depicted in the equation below:

$$X_{UTFRC}(t) = U(b, \Delta) * X_{TFRC}(t) \tag{4}$$

where $b$ is the bitrate for the current stream second and $\Delta$ is the number of stream seconds saved in the client's prefetch buffer. $U(b, \Delta)$ has values in the interval $[0.8, 1.2]$ and has the following form:

$$U(b, \Delta) = 0.8 + U_b(b) + U_\Delta(\Delta) \tag{5}$$

$U_b(b)$ is responsible for $3/4$ of the total utility and is a linear mapping of current second's bitrate $b$ from the interval $[b_{min}, b_{max}]$ of all possible 1-second bitrates of the stream ($b_{min}$ refers to the minimum 1-second bitrate from the whole stream and $b_{max}$ to the maximum 1-second bitrate) to the interval $[0, 0.3]$:

$$U_b(b) = \frac{3}{10} * \frac{b - b_{min}}{b_{max} - b_{min}} \tag{6}$$

Function $U_\Delta(\Delta)$ is responsible for $1/4$ of the total utility and takes values in the interval $[0.05, 0.1]$ when $\Delta$ is smaller than a threshold value and in the interval $[0, 0.05]$ when $\Delta$ is greater than the threshold value. The expression of the function is given bellow:

$$U_\Delta(\Delta) = \begin{cases} \frac{1}{10}\left(1 - \frac{\Delta}{2threshold} - max\left(0, \frac{threshold - E[\Delta]}{2threshold}\right)^+\right) \\ \qquad\qquad , \Delta \leq threshold \\ \frac{1}{20}\left(1 - \frac{\Delta - threshold}{\Delta_{max} - threshold}\right) \qquad , \Delta > threshold \end{cases} \tag{7}$$

where $E[\Delta]$ is the average buffer value from the beginning of the streaming session up to now and $\Delta_{max}$ is a reasonable maximum buffer value. The term $max\left(0, \frac{threshold - E[\Delta]}{2threshold}\right)$ was introduced as an incentive for applications which maintain a lower buffer in order to get a higher utility and, thus, a larger throughput; for those applications the value $E[\Delta]$ is small and because of this term $U_\Delta(\Delta)$ gets smaller with time.

One final note about the advanced UTFRC. The utility function uses the value of the client buffer, $\Delta$, which is computed by the client, but it must be available at the server. This value must be fed back to the server which introduces a delay of one round-trip time in the decision of the server, but this is no worse than TFRC as it reacts to congestion also after a round-trip time.

## 6. Experiments

We have performed some simulations using the ns-simulator with the simple UTFRC modified so that the utility is incremented by 0.8 when the buffer becomes really small. We have a single 7.1 Mbps link shared by 16 TCP-Reno flows and one multimedia flow. In the first experiment we have used TFRC for the multimedia flow, while in the second experiment we have used the simple UTFRC modified as stated above for the multimedia flow. In the TFRC and UTFRC source we simulated the prefetch client buffer which is loaded according to the throughput computed by TFRC/UTFRC and consumed by the player according to the bitrate of the stream (see Fig. 5). The results are shown in Fig. 3 and Fig. 4. Fig. 5 presents the bitrate of the stream.

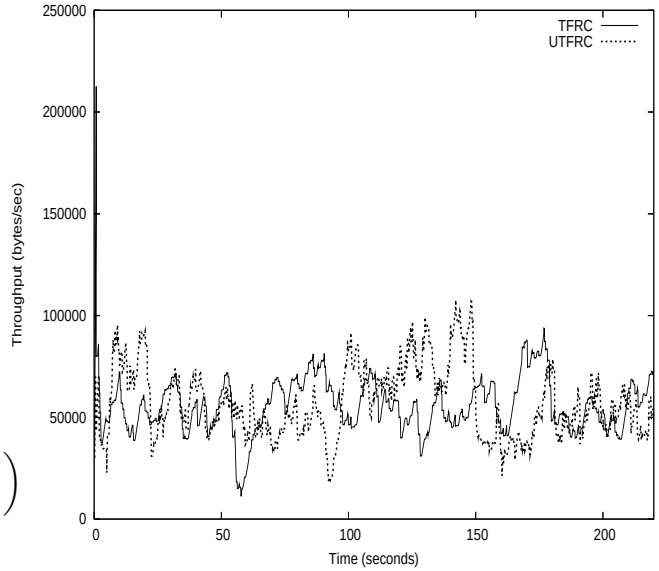


Figure 3. TFRC throughput versus simple UTFRC throughput

We can see in Fig. 3 the evolution of the throughput of TFRC and UTFRC and the evolution of the buffer for TFRC and UTFRC in Fig. 4. The benefits of using UTFRC are clear in Fig. 4 where we can see that the client prefetch buffer becomes empty several times (e.g. seconds 25, 60, 140) when TFRC is used, thus causing video to freeze at the client, while for UTFRC this situation does not happen.

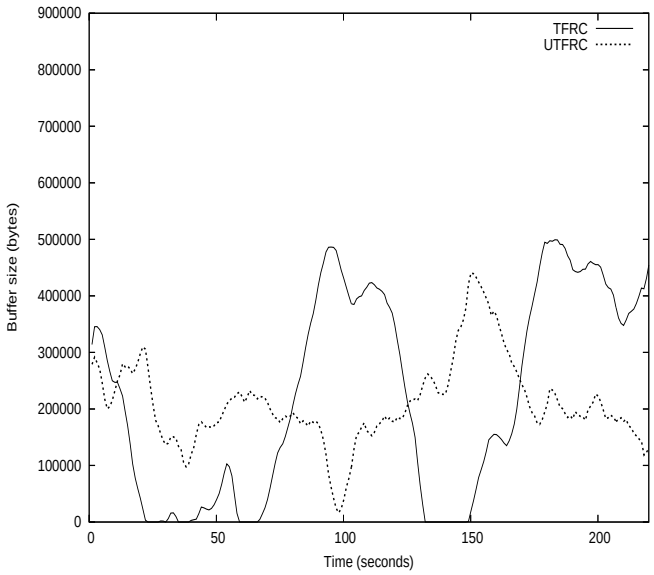


Figure 4. Buffer evolution for TFRC and UTFRC throughput

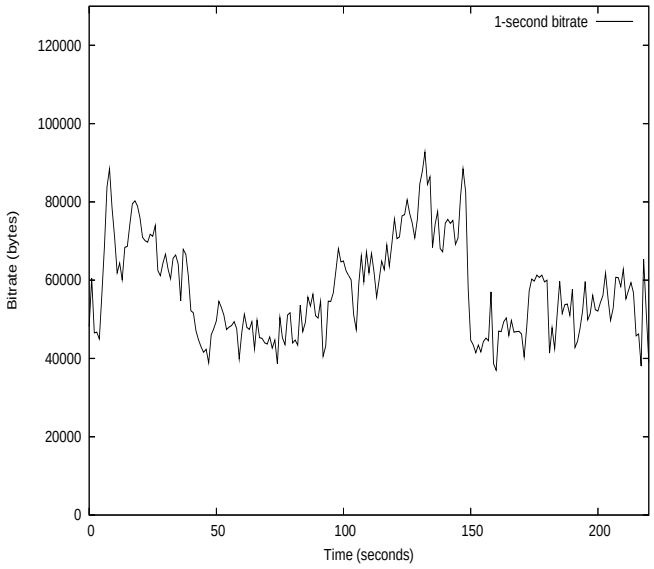


Figure 5. The bitrate of the stream

## 7. Conclusions and Future Work

We have presented in this paper the concept of Utility-driven TFRC and discussed its usefulness for multimedia streaming applications. We have presented also two variants of Utility-driven TFRC, a simple one and an advanced one, that can improve the quality of service for streaming applications. Initial simulations show that UTFRC can improve the perceived quality of multimedia streams, but further tests in real streaming frameworks with both versions of UTFRC are necessary in order to fully assess their value or limitations.

## References

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

[2] S. Floyd, M. Handley, J. Padhye, J. Widmer, *TCP Friendly Rate Control*, RFC 3448, January 2003.

[3] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, SIGCOMM Symposium on Communications Architectures and Protocols, Aug. 1998.

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

[5] R. Rejaie, M. Handley, and D. Estrin, *An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet*, In Proceedings of INFOCOMM 99, 1999.

[6] D. Sisalem and H. Schulzrinne, *The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaption Scheme*, In Proceedings of NOSSDAV98, 1998.

[7] I. Rhee, V. Ozdemir, and Y. Yi, *TEAR: TCP Emulation at Receivers Flow Control for Multimedia Streaming*, April 2000. NCSU Technical Report.

[8] D. Tan and A. Zakhor, *Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly Transport Protocol*, IEEE Transactions on Multimedia, May 1999.

[9] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, *Dynamic behavior of slowly-responsive congestion control algorithms*, In Proc. of ACM SIGCOMM01, San Diego, California, USA, August 2001.

[10] Y. Richard Yang, M. Sik Kim, and Simon S. Lam, *Transient Behaviors of TCP-friendly Congestion Control Protocols*, In Proc. of IEEE Infocom2001, March 2001.

[11] S. Floyd, M. Handley and J. Padhye, *A Comparison of Equation-Based and AIMD Congestion Control*, ACIRI, February 2000, http://www.aciri.org/tfrc/.

[12] I. Rhee, L. Xu, *Limitations of Equation-based Congestion Control*, In Proc. of ACM SIGCOMM'05, Philadelphia, Pennsylvania, USA, August 2005.

[13] Z. Wang, S. Banerjee and S. Jamin, *Media-Friendliness of A Slowly-Responsive Congestion Control Protocol*, In Proceedings of NOSSDAV'04, 2004.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] M. Vojnovic, J. Y. Le Boudec, *On the long-run behavior of equation-based rate control*, IEEE/ACM Transactions on Networking, Volume 13, Issue 3, pp.568 - 581, 2005.

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