Optimization of TFRC Loss History Initialization

Guillaume Jourjon, Emmanuel Lochin, and Laurent Dairaine

***Abstract*— This letter deals with the initialization of the loss history structure in the TFRC (TCP-Friendly Rate Control) mechanism. This initialization occurs after the detection of the first loss event after every slowstart phase. The loss history is crucial for the algorithm since it returns the packet loss rate estimation. This estimation is used in the TFRC equation to compute the sending rate. In this letter, we propose a new method to compute the packet loss rate which is more computationally efficient and remains as accurate as the classical commonly used method. The motivation of this work is to reduce the computation time and formulate a unified computation scheme. This method is based on the Newton's algorithm issued from numerical analysis of the TCP throughput equation. This proposal is evaluated analytically and the results show a significant improvement in terms of the computation time.**

***Index Terms*— Optimization, TFRC, transport.**

I. Introduction

TFRC is a congestion control mechanism for unicast flows operating in a best-effort Internet environment and competing with TCP traffic [1]. This mechanism has been integrated into DCCP [2] as one of the possible congestion control method. The main characteristic of TFRC is the use of a TCP equation model which provides a much lower throughput variation over time than TCP. As a result, it is more suitable for multimedia applications such as audio/video streaming or voice over IP.

In addition to round trip time and received estimated throughput,TFRC algorithm needs an estimation of the packet loss rate. This loss estimation is computed at the receiver side and sent periodically to the sender where the rate control is performed. The initial packet loss estimation is crucial as it allows the sender to compute the sending rate as described in [1] and the receiver to correctly initialize its loss history events. Indeed, the subsequent estimations of the packet loss rate are based on a weighted moving average using this history. As a consequence, the initialization of this structure has an impact on the sending throughput and on TFRC overall performances (for further details see section 6 in [1]).

Since TFRC equation can not be analytically solved due to the higher exponent, the method proposed in [3] is based on a binary search process. In every iteration, this process computes the solution of the initialization by halving the range of study; by the end, the result will be chosen as the middle of the last range. This method is used by all the early available TFRC implementations such as in ns-2 or DCCP implementation. To the best of our knowledge, no efficient method has been formulated. In this letter, we show that the binary search is not efficient and needs a large number of iterations to achieve the 5% accuracy required by the TFRC RFC [1]. Then, we propose a method which computes faster than the binary search method.

This letter is structured as follows. Section II states the problem and explains our proposed method. Section III provides numerical results and analytical analysis of them. Finally, section IV provides conclusions and perspectives.

Manuscript received October 19, 2006. The associate editor coordinating the review of this letter and approving it for publication was Prof. Samuel Pierre.

G. Jourjon and E. Lochin are with National ICT Australia Ltd., Alexandria, NSW 1435, Australia (email: guillaume.jourjon@nicta.com.au).

L. Dairaine is with the University of Toulouse.

Digital Object Identifier 10.1109/LCOMM.2007.061707.

II. Optimization of the Loss Rate Computation

In this section, we present the initialization problem and provide the algorithm of our solution.

A. Problem statement

TFRC uses a TCP throughput model given by the following equation (1).

$$X = \frac{s}{(RTT \cdot \sqrt{\frac{p \cdot 2}{3}} + RTO \cdot \sqrt{\frac{p \cdot 27}{8}} \cdot p \cdot (1 + 32 \cdot p^2))} \quad (1)$$

The sending rate (X) depends on the packet loss rate (p), the mean packet size (s) and the Round Trip Time. RTO refers to the TCP retransmission timeout value.

During the initialization phase, TFRC acts like the TCP slow start algorithm. This slow start phase can also occur during the transfer if the RTO timeout expires [1]. This phase is followed by a congestion avoidance phase as soon as the receiver detects a loss. In order to compute the sending rate X, TFRC needs an estimation of the current loss event rate. This estimation is achieved using a loss history structure which records the number of packets between successive loss events. From this structure, the loss rate can be computed, as described in [1].

When the slow start phase is over, the loss history has to be initialized. The number of packets transmitted during the slow start phase cannot be used to estimate the loss event rate since it does not reflect the underlying packet drop rate of the connection [3]. For this reason, existing TFRC implementations use a simple binary search process in which the receiver measures the receiving rate ($X_{measured}$) and then starts the estimation of the corresponding loss event rate. This estimation is performed by computing the packet loss rate that should have allowed the sender to transmit at the rate $X_{measured}$ using (1). We claim that the loss history's initialization can be improved with the use of a numerical analysis based on Newton's algorithm of the TFRC equation.

B. Newton's algorithm rate estimation method

The binary search algorithm is well known for its easily programmable properties but also for its slow convergence. Usually, numerical problems are solved using more complex algorithms, such as the gradient method or Newton's algorithm or primal-dual method. Because of the relatively simple equation used by TFRC, we propose to set a Newton's algorithm in order to estimate the packet loss rate. To compute this algorithm, during the loss history initialization phase, (1) has to be used as a simple function of p as follows:

$$F(p) = RTT \cdot \sqrt{\frac{p \cdot 2}{3}} + RTO \cdot \sqrt{\frac{p \cdot 27}{8}} \cdot p \cdot (1 + 32 \cdot p^2) - \frac{s}{X_{measured}} \quad (2)$$

Then, the problem becomes to solve $F(p) = 0$. (2) is a polynomial function which can be derived as follows:

$$F'(p) = \frac{RTT \cdot \sqrt{\frac{2}{3}}}{2 \cdot \sqrt{p}} + RTO \cdot \sqrt{\frac{27}{8}} \cdot (1.5 \cdot \sqrt{p} + 112 \cdot p^{\frac{5}{2}}) \quad (3)$$

Newton's algorithm is known to converge to the solution in a $O(n^2)$ compared to a $O(log(n))$ for the binary search algorithm [4]. Thanks to (3), we propose using Newton's algorithm starting with an under solution (value of p for the first iteration). Newton's iterative process is performed while the convergence criterion is not reached and is computed as follows:

$$p_{i+1} = p_i - \frac{F(p_i)}{F'(p_i)} \quad (4)$$

Newton's algorithm is constrained by the existence of $F(p_i)$ and $F'(p_i)$ for all $p_i \in [a, b]$, where $[a, b]$ is the study interval with $a = 0$ and $b = 1$. This constraint leads us to exclude values $p_i \leq 0$. In our method, we claim that we have to start the process with the value $a = 10^{-7}$. This value comes from the analysis of the function $F(p)$. Indeed, $F(p)$ has a double concavity. This double concavity can result in a negative value for the next iteration of the algorithm. Nevertheless, this double concavity has a stationary point for $p_0 = 2.84 * 10^{-2}$, for all RTT, s, and $X_{measured}$ under the hypothesis $RTO = 4RTT$ [1]. In an obvious way, the stationary point is no longer identical for all RTT, s, and $X_{measured}$ if $RTO \neq 4RTT$. According to this stationary point and the restriction $p > 0$ due to the square root, we have to take a starting point inferior to p_0. Thanks to the convergence property of Newton's algorithm [4], we know that it will find the result from any starting point.

In order to compare the two methods, we introduce the following convergence criterion:

$$\left| \frac{X_{measured} - X_{computed}}{X_{measured}} \right| \leqslant \alpha \quad (5)$$

where $X_{measured}$ is the rate measured by the receiver, $X_{computed}$ is the rate computed with (1) and α is the computation accuracy criterion (also called stopping criterion). According to [1], this accuracy is recommended to be at least 5%, meaning that when $X_{computed}$ equals to $X_{measured}$ more or less 5% the process stops. The 5% tolerance is introduced for two main reasons: firstly (1) is difficult to invert, and secondly the numerical computation cost of p [1]. The next section will evaluate the convergence pace of both algorithms.

TABLE I
SUMMARY OF THE NUMBER OF ITERATIONS FOR BOTH ALGORITHMS

	average	min	max	standard deviation
Binary search	21.7379	1	29	4.1656998
Newton	5.00034	3	13	0.0688248

TABLE II
EXAMPLE OF α' VALUE

RTT	$X_{measured}$	s	α'
$100ms$	$1Mbit/s$	$8Kbits$	0.0006 ($n = 11$)
$400ms$	$2Mbit/s$	$8Kbits$	0.000015 ($n = 17$)

III. NUMERICAL RESULTS AND INTERPRETATION

A. Numerical results

This section presents the numerical results of our proposed loss rate computation method compared to the classical binary search method. We have implemented these two algorithms in C++ and evaluated analytically the number of iterations to compute p for given X and RTT values. All the computations have been performed on a Pentium IV processor. The study is performed over a large scale of bandwidth and delay values, with an RTT ranging from $1ms$ to $1000ms$ and a bandwidth ranging from $1KB/s$ to $100MB/s$.

Table I shows the summary of the number of iterations required to reach the 5% accuracy in this context.

The results concerning the binary search are expected as this method tends to converge with a number of iterations n [4] with:

$$n = \left\lceil log_2(\frac{|b-a|}{\alpha}) \right\rceil \quad (6)$$

where $[a, b]$ is the study interval with $a = 0$ and $b = 1$, α is the convergence criterion and $\lceil x \rceil$ denotes the ceiling of x (i.e., the smaller integer $\geq x$). Nevertheless in (6), the convergence criterion α is supposed to be function of the iterative parameter (in our case p). In this study the convergence criterion is function of $X_{computed}$. In order to explain the number of iterations needed by the binary search, we have to use the equation (7).

$$n = \left\lceil log_2(\frac{|b-a|}{\alpha'}) \right\rceil \quad (7)$$

where α' is the equivalent convergence criterion on p for $\alpha = 0.05$. For the same reason, as it is impossible to solve directly the equation, the translation from α to α' cannot be done with a simple function. In order to illustrate this translation, we give two examples as shown in Table II.

Our large range of numerical experiments tends to show that there is a correlation between RTT, $X_{measured}$ and the new accuracy on p. Indeed, in our experiments, the number of iterations needed to compute the binary search increase with the RTT and $X_{measured}$.

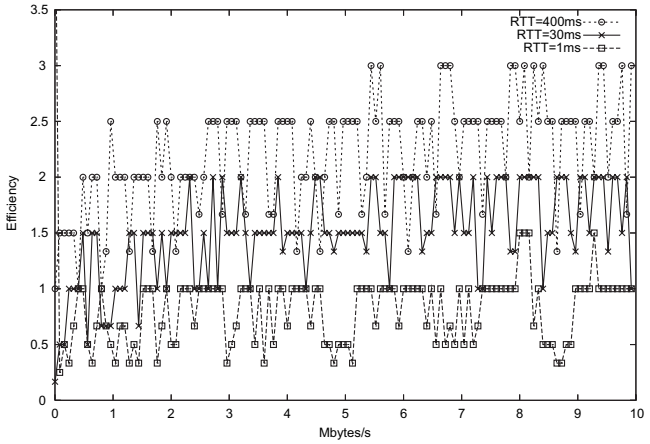

Fig. 1. Comparison of binary search and Newton's algorithm efficiency.

The results concerning Newton's algorithm are also linked to the properties of this algorithm. Indeed, Newton's algorithm converges monotonically from any starting point [4]. Nevertheless, there is no general inferior or superior boundary for Newton's algorithm.

Obviously, the number of iterations is linked to the computation time needed by both methods. However, even if Newton's algorithm requires a stable number of iterations, we need to verify if it is more computationally efficient. In the next section, we therefore compare the computation time of both algorithms. We focus on the range where the binary search algorithm is efficient.

B. Interpretation and discussion

In this section, we study the efficiency of Newton's algorithm over the binary search algorithms in terms of computation time. We define the efficiency criterion ϵ as the ratio of the binary search algorithm computation time (t_{dicho}) to Newton's algorithm computation time (t_{Newton}):

$$\epsilon = \frac{t_{dicho}}{t_{Newton}} \tag{8}$$

The results are shown in Fig. 1. When the efficiency criterion is lower than one it corresponds to a better efficiency of the binary search algorithm.

In Fig. 1, we represent the same study case as in Table I. We focus on the smaller range of bandwidth than in Table I as this range is less favorable to Newton's algorithm. In this study range, our study shows that Newton's algorithm is more efficient for more than 95% of the cases, as efficient for less than 4% and less efficient for less than 1% of the cases.

In addition, we propose to study the correlation between the number of iterations and the number of CPU cycles needed to reach the 5% of accuracy. In order to explore the way these two characteristics interact, we proceed to an algorithmic study of both algorithms. We present in Table III the number of elementary operations in both algorithms and the theoretical number of CPU cycles for every operation[1].

Usually, the division and the square root need the highest number of CPU cycles. We see that Newton's algorithm needs more elementary operations per iteration. But as shown previously, it also needs less iteration for a given accuracy. Next we study these two algorithms for the worst case. For the binary search, the worst case is when the result of the computation is outside the range $[0, 1]$[2]. In this case, the maximum number of iterations should be fixed *a-priori* as a static variable. According to results obtained and summarized in Table I, this variable is equal to 30. The Newton's worst case has been evaluated to 13 iterations by our computation scheme. In these conditions the worst case study results are presented in Table IV.

We show that the number of cycles needed in both cases is largely in favor of Newton's algorithm.

TABLE III
LIST OF THE NUMBER OF ELEMENTARY OPERATIONS
($n = number\ of\ iterations$)

	$+$	$*$	$/$	$sqrt$
Binary search	$4n+4$	$8n+8$	$2n+2$	n
Newton	$5n$	$14n$	$3n$	n
Number of CPU cycles	0.5	15	70	70

TABLE IV
WORST CASE STUDY

	Iterations	CPU cycles
Binary search	30	10222
Newton	13	6402.5

IV. CONCLUSION

In this letter, we present an efficient method to compute the packet loss rate of the TFRC equation. We show that this method outperforms the 5% accuracy in terms of number of iterations and computation time. The initial results presented in this letter show the efficiency of the proposed method. Due to its low computation needs, it is particularly well-suited to mobile devices with low processing power. Furthermore, due to the low standard deviation of the number of iterations, it is well-suited to real time processing. In future work, we will evaluate the performance implications of using this mechanism in a TFRC implementation.

ACKNOWLEDGMENT

We thank Jean-Luc Lamotte for his advice concerning the algorithmic study of both algorithms. We would like to thank anonymous reviewers and Sébastien Ardon for their valuable remarks.

REFERENCES

[1] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control TFRC: protocol specification," IETF, Request for Comments.
[2] E. Kohler, M. Handley, and S. Floyd, "Datagram congestion control protocol (DCCP)," IETF, Request for Comments 4340, Mar. 2006.
[3] J. Widmer, "Equation-based congestion control," Ph.D. dissertation, University of Mannheim, Feb. 2000.
[4] W. Gautschi, *Numerical Analysis: An Introduction*. Boston: Birkhäuser, 1997.

[1]According to Intel PIV documentation.

[2]These rare cases have been voluntary excluded from the Table I.