Additive Increase Adaptive Decrease Congestion Control: a Mathematical Model and Its Experimental Validation[1]

L.A. Grieco
Dipartimento d'Ingegneria dell'Innovazione
Università di Lecce, Italy
alfredo.grieco@unile.it

S. Mascolo
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari, Italy
mascolo@poliba.it

R. Ferorelli
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari, Italy

Abstract

Due to the fundamental end-to-end design principle of the TCP/IP for which the network cannot supply any explicit feedback, today TCP congestion control algorithm implements an additive increase multiplicative decrease (AIMD) algorithm. It is widely recognized that the AIMD mechanism is at the core of the stability of end-to-end congestion control. In this paper we describe a new mechanism we call Additive Increase Adaptive Decrease (AIAD). The key concept of the adaptive decrease mechanism is to adapt congestion window reductions to the bandwidth available at the time the congestion is experienced. We propose Westwood++ TCP as an implementation of the AIAD paradigm, and we consider Reno TCP as an example of the AIMD mechanism for comparison. We derive a mathematical model of the throughput of the AIAD mechanism that shows that Westwood++ is stable, is friendly to Reno and increases the fairness in bandwidth utilization. To confirm the validity of the theoretical model Internet measurements are reported.

1. Introduction

The stability of the Internet requires that flows use some form of end-to-end congestion control to adapt the input rate to the available bandwidth. TCP Congestion control was introduced into the Internet in the late 1980s by Van Jacobson after that the Internet had experienced congestion collapse [1], [2]. Due to the fundamental end-to-end design principle of the TCP/IP for which the network is a black box that cannot supply any explicit feedback [4], [5], [8] the TCP congestion control algorithm was designed by following the additive increase multiplicative decrease paradigm (AIMD) [17]. The *additive increase* phase aims at gradually increasing the flow input rate until the network available bandwidth is hit and a congestion episode happens. The sender becomes aware of congestion via the reception of duplicate acknowledgments (DUPACKs) or the expiration of a timeout. Then it reacts to light congestion (i.e. 3 DUPACKs) by halving the congestion window and to heavy congestion (i.e. timeout) by reducing the congestion window to one. This behavior is the *multiplicative decrease* phase, which aims at reducing the input rate after congestion.

In this paper, we propose a new mechanism that adaptively set the congestion window after a congestion episode by taking into account the bandwidth used when congestion is experienced. We call this mechanism *Additive Increase Adaptive Decrease* (AIAD).

Vegas TCP can be considered the first significant example of congestion control algorithm that partially departs from the AIMD paradigm [12],[14]. In fact, during congestion avoidance, Vegas exploits an *additive increase additive decrease* mechanism [10]. Another example of control algorithm that partially departs from of the AIMD paradigm is Westwood TCP [3], which uses a sophisticated end-to-end bandwidth estimation to set the control windows after light congestion. Both Vegas and Westwood preserve the standard multiplicative decrease behavior after a timeout.

In this paper, we propose Westwood++ TCP, which is based on a complete exploitation of the AIAD paradigm. The main contribution of the paper is a simple stochastic model of the throughput of Westwood++ which shows that Westwood++ (1) is stable, (2) is friendly to Reno and (3) increases the fairness in network usage.

[1] This work was partially supported by the MIUR Research Project 488/92 C22 entitled "e-service to the Citizens"

Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)
1530-1346/02 $17.00 © 2002 IEEE

We test Westwood++ on the Internet in order to validate the proposed equation model and compare Westwood++ with respect to Reno TCP. Experiments show that the model predicts the throughput of Westwood++ within a mean error ranging from 11.7% to 43.9% and that Reno obtains a slight smaller throughput than Westwood++.

The paper is organized as follows: in Section 2 an equation model of Westwood++ TCP is developed; in Section 3 we validate the model via Internet measurements and we provide a comparison with Reno TCP; finally the last section draws the conclusions.

2. Westwood++ TCP

In this section we describe Westwood++ TCP and develop a mathematical model for its steady state throughput.

2.1. Westwood++ TCP

First, we briefly resume the Westwood TCP algorithm [3] and then we introduce Westwood++. A TCP connection is characterized by the following variables: (*cwnd*) Congestion Window; (*ssthresh*) Slow Start Threshold; (*RTT*) Round Trip Time of the connection; (RTT_{min}) Minimum Round Trip Time measured by the sender; (*Seg_size*) Size of the delivered segments.

The main idea of Westwood TCP is to perform an end-to-end estimate of the bandwidth *B* available along a TCP connection by measuring and low-pass filtering the rate of returning ACKs. The estimate is then used to properly set the congestion window and the slow-start threshold after a congestion episode as described below:

a) When 3 DUPACKs are received by the sender:
   $ssthresh = (B*RTT_{min})/seg\_size$;
   $cwnd = ssthresh$;
b) When coarse timeout expires:
   $ssthresh = (B*RTT_{min})/seg\_size$;
   $cwnd = 1$;
c) When ACKs are successfully received, TCPW increases *cwnd* accordingly to Reno algorithm.

The idea of Westwood++ TCP is to exploit the bandwidth estimate also after a timeout. In fact, the mechanism of bandwidth estimation, which is based on the returning flow of ACKs, ensures back-off when several ACKs are lost. In other words, the bandwidth estimate as used in step (a) is sufficient to react also to heavy congestion. Thus, by unifying steps (a) and (b), the Westwood++ algorithm completely exploits the AIAD paradigm as follows:

a) When 3 DUPACKs are received OR a timeout expires:
   $ssthresh = (B*RTT_{min})/seg\_size$;
   $cwnd = ssthresh$;
b) When ACKs are successfully received, TCPW increases *cwnd* accordingly to Reno algorithm.

It should be noticed that the AIAD mechanism can improve the stability property of the Additive Increase Multiplicative Decrease algorithm (AIMD). In fact the *adaptive decrease* phase ensures that the window is reduced enough in the presence of congestion whereas the by half reduction of Reno could not be sufficient.

2.2. A Stochastic Model of Westwood++ throughput

In this section a mathematical model of the Westwood++ throughput is developed. To derive the model, we follow arguments similar to the ones developed in the excellent paper [7].

Theorem 1: Let us consider a Westwood++ TCP flow and suppose that the drop probability of a segment is *p*, the bandwidth available for the flow is *B*, the mean Round Trip Time is *RTT* and the minimum Round Trip Time is RTT_{min}. By letting r^{West} be the steady state mean throughput of the flow, it holds:

$$r^{West} = \frac{B \cdot RTT_{\min}}{2 \cdot RTT} + \sqrt{\left(\frac{B \cdot RTT_{\min}}{2 \cdot RTT}\right)^2 + \frac{1-p}{p \cdot (RTT)^2}} \qquad (1)$$

Proof: The congestion window is updated upon ACK reception. Each time an ACK is received back by the sender the *cwnd* is increased by 1/*cwnd*, whereas after a segment loss the congestion window is set equal to $B \cdot RTT_{min}$ so that the change in *cwnd* is $B \cdot RTT_{\min} - cwnd$. Thus, the variation of congestion window $\Delta cwnd$ is a discrete random variable with the following probability function:

$$p(\Delta cwnd) = \begin{cases} 1-p & when\ \Delta cwnd = \dfrac{1}{cwnd} \\ p & when\ \Delta cwnd = B \cdot RTT_{\min} - cwnd \end{cases}$$

The expected increment of the congestion window *cwnd* per update step is then:

$$E[\Delta cwnd] = \frac{1-p}{cwnd} + (B \cdot RTT_{\min} - cwnd) \cdot p \qquad (2)$$

Since the time between update steps is about $RTT / cwnd$, by recalling Eq. (2), the expected change in the rate *r* per unit time is approximately:

$$\frac{\partial r(t)}{\partial t} = \frac{1-p}{RTT^2} + p \cdot r(t) \cdot \frac{B \cdot RTT_{\min}}{RTT} - r^2(t) \cdot p \qquad (3)$$

Eq. (3) is a separable variable differential equation that can be written as follows:

Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)
1530-1346/02 $17.00 © 2002 IEEE

$$\frac{\partial r\ (t)}{r^2(t) - r(t)\cdot\frac{B\cdot RTT_{\min}}{RTT} - \frac{1-p}{p\cdot RTT^2}} = -p\cdot\partial t \qquad (4)$$

and by integrating each member the following solution can be obtained

$$r(t) = \frac{r_1 - r_2\cdot C\cdot e^{-p\cdot t\cdot(r_1 - r_2)}}{1 - C\cdot e^{-p\ \cdot t\cdot(r_1 - r_2)}}$$, where $r_{1,2}$ are the roots of $r^2 - r\cdot\frac{B\cdot RTT_{\min}}{RTT} - \frac{1-p}{p\cdot RTT^2} = 0$ and C depends on the initial conditions.

The steady state throughput of Westwood++ is then obtained as follows: $r^{West} = \lim_{t\to\infty} r(t)$.

3. Model based evaluation of Westwood++

The goal of this section is to analyze stability, fairness and friendliness of Westwood++ and to develop a comparison with Reno TCP by using their respective throughput equation models. Finally, we validate the model of Westwood++ using Internet measurements.

3.1. Stability and full bandwidth utilization

We start by showing that Westwood++ is stable, that is, its steady state input rate is bounded by the available bandwidth *B*.

Corollary 1: The Westwood++ control algorithm is stable, that is:

$$r^{West} \le B \qquad (5)$$

Proof: From Eq. (1) we can argue that r^{West} can never be greater than *B*. In fact, by contradiction, let us assume that $r^{West} > B$. This assumption would lead to congestion collapse so that the drop probability *p* would increase up to 1. Thus, from Eq. (1) it would result $r^{West} = B\cdot RTT_{\min} / RTT$. Since under congestion $RTT_{\min} < RTT$, it would result $r^{West} < B$, which would contradict the assumption. Therefore we can conclude that $r^{West} \le B$.

Corollary 2: Let $\hat{B}$ be an estimate of *B*. The westwood++ throughput satisfies the bound:

$$r^{West} > \alpha\cdot\hat{B} \qquad (6)$$

where $0 < \alpha \le 1$.

Proof: From Eq. (1) it is straightforward to notice that $r^{West} > \alpha\cdot\hat{B}$, where $\alpha = RTT_{\min} / RTT \le 1$.

3.2. Fairness and friendliness of Westwood++

Equation (1) gives the throughput as a function of the given capacity *B*. However, in real networks the available bandwidth *B* is unknown. Thus, it is necessary to employ a mechanism of end-to-end bandwidth estimation. The basic idea is to estimate the available bandwidth by means of the flow of returning ACKs. Due to delays and ACKs compression, the flow of returning ACKs must be low-pass filtered in a proper way [3], [9], [11], [13], [15], [18], [19]. In [3], a sample of available bandwidth $b_k = d_k / (t_k - t_{k-1}) = d_k / \Delta_k$ is computed every time t_k the sender receives an ACK, the amount d_k of data acknowledged by an ACK is determined by a proper counting procedure and samples b_k are low-pass filtered using the time-varying filter $\hat{b}_k = \frac{2\tau_f - \Delta_k}{2\tau_f + \Delta_k}\hat{b}_{k-1} + \Delta_k\frac{b_k + b_{k-1}}{2\tau_f + \Delta_k}$, where τ_f is the filter time constant (a typical value is $\tau_f = 0.5\,\text{sec}$). In this paper we use a slightly modified version of the filter used in [3] since that filter overestimates the available bandwidth because of ACK compression [15]. To overcome this problem, we compute bandwidth samples every *RTT*. More precisely, we count all data d_k acknowledged during the last *RTT* and compute the bandwidth sample $b_k = d_k / \Delta_k$, where Δ_k is the last *RTT*. To avoid aliasing effects [6], when $\Delta_k > \tau_f / 4$, we interpolate and re-sample using $N = int(4\cdot RTT / \tau_f)$[2] virtual samples b_k arriving with interarrival time $\Delta_k = \tau_f / 4$.

It is worth noticing that, because of the flow conservation principle, the estimator provides a value of *B* close to the input rate *cwnd/RTT*. By assuming the approximation *B=cwnd/RTT* for the estimate, we are ready to show the following result.

Theorem 2: The steady state throughput of Westwood++ using the bandwidth estimate *B=cwnd/RTT* is

$$r^{West} = \frac{1}{\sqrt{RTT\cdot T_q}}\cdot\sqrt{\frac{(1-p)}{p}} \qquad (7)$$

where $T_q = RTT - RTT_{min}$ is the mean queuing time experienced by the segments of the connection.

Proof: By assuming the following estimate of the available bandwidth:

$$B = cwnd / RTT = r(t) \qquad (8)$$

and by substituting Eq. (8) into Eq. (3), the following differential equation is obtained:

[2] $int(\cdot)$ stands for the integer part of $(\cdot)$

$$\frac{\partial r(t)}{\partial t} = \frac{1-p}{RTT^2} + p \cdot r^2(t) \cdot \frac{RTT_{\min}}{RTT} - r^2(t) \cdot p \qquad (9)$$

By separating variables, Eq. (9) can be written as:

$$\frac{\partial r(t)}{r^2(t) - \frac{1-p}{p \cdot T_q \cdot RTT}} = -p \cdot \frac{T_q}{RTT} \partial t \qquad (10)$$

and by integrating each member the solution is:

$$r(t) = \sqrt{\frac{1-p}{p \cdot RTT \cdot T_q}} \frac{1 + C \cdot e^{-2pt \frac{T_q}{RTT}\sqrt{\frac{1-p}{pRTT \cdot T_q}}}}{1 - C \cdot e^{-2pt \frac{T_q}{RTT}\sqrt{\frac{1-p}{pRTT \cdot T_q}}}} \qquad (11)$$

Where, C depends on the initial conditions. The steady state throughput (7) is then obtained for $t \to \infty$.

Now we are ready to compare the Westwood++ throughput (7) with the Reno throughput as reported in [16], which is:

$$r^{Reno} = \frac{1}{RTT\sqrt{\frac{2p}{3}} + T_0 \cdot \min\left(1, \sqrt{\frac{27p}{8}}\right) \cdot p \cdot \left(1 + 32p^2\right)} \qquad (12)$$

where T_0 is the timeout. When $p << 1$, a good approximation of (12) is (see [16])

$$r^{Reno} = \frac{\sqrt{1.5}}{RTT\sqrt{p}} \qquad (13)$$

With reference to friendliness, by comparing (7) and (13) it can be noticed that both throughputs of Westwood++ and Reno depend on $1/\sqrt{p}$, that is Westwood++ is friendly to Reno. Moreover, from (7) it can be noticed that flows with different *RTTs* experience the same mean queuing time T_q. Therefore, the throughput of Westwood++ depends on round trip time as $1/\sqrt{RTT}$ whereas the throughput of Reno as $1/RTT$. Thus, with reference to fairness it can be concluded that Westwood++ increases fair sharing of network capacity between flows with different RTTs.

3.3. Modeling the impact of window limitation

So far, we have assumed no bound on the *cwnd* size. This subsection aims at modeling the Westwood++ throughput under the assumption that a maximum bound W_{MAX} on the congestion window value exists. Therefore we assume that during the data transfer the *cwnd* reaches its maximum value so that the available bandwidth estimated at the time a loss is experienced is equal to $B = W_{MAX} / RTT$. Thus, the *cwnd* is set equal to $cwnd = B \cdot RTT_{\min} = W_{MAX} RTT_{\min} / RTT$ after that a loss is experienced. Under this assumption we obtain the time domain behavior of *cwnd* pictured in Fig. 1.

Let us assume the following notation:

S_1 : segments sent during the time interval $[0, T_1]$.

S_2:segments sent during the interval $[T_1, T_1 + \Delta T]$.

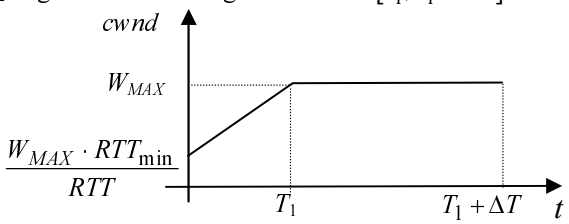

Figure 1. Congestion Window behavior

From Fig. (1) it follows:

$$Cwnd = \begin{cases} \frac{W_{MAX} RTT_{\min}}{RTT} + \frac{t}{RTT} & if\ cwnd \le W_{MAX} \\ W_{MAX} & elsewhere \end{cases} \qquad (14)$$

From Eq. (14) we can compute T_1 by using the relation $\frac{W_{MAX} \cdot RTT_{\min}}{RTT} + \frac{T_1}{RTT} = W_{MAX}$:

$$T_1 = W_{MAX} \cdot (RTT - RTT_{\min}) \qquad (15)$$

By considering that $r = cwnd / RTT$ is the connection input rate, the number of segment sent during the time interval $[0, T_1]$ is given by the integral $S_1 = \int_0^{T_1} r(\tau) d\tau$. Then from Fig. 1 it holds:

$$S_1 = \frac{W_{MAX}^2}{2 \cdot RTT^2} \cdot (RTT^2 - RTT_{\min}^2) \qquad (16)$$

Following the same arguments used to compute S_1, the number of segment sent during the time interval $[T_1, T_1 + \Delta T]$ is:

$$S_2 = W_{MAX} \cdot \Delta T / RTT \qquad (17)$$

By considering that a segment drop probability p means that a segment is dropped approximately after that $1/p$ segments have been sent, it follows:

$$\frac{1}{p} = S_1 + S_2 \qquad (18)$$

By considering Eqs. (16), (17) and (18) it results:

$$\Delta T = \frac{RTT}{p \cdot W_{MAX}} - \frac{W_{MAX}}{2 \cdot RTT} \cdot (RTT^2 - RTT_{\min}^2) \qquad (19)$$

By considering Eqs. (15), (18) and (19), the mean throughput of TCP Westwood++ can be computed as:

$$r^{West} = \frac{S_1 + S_2}{T_1 + \Delta T} = \frac{\frac{W_{MAX}}{RTT}}{1 + \frac{p}{2}\left(W_{MAX} - \frac{W_{MAX} RTT_{\min}}{RTT}\right)^2} \qquad (20)$$

Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)
1530-1346/02 $17.00 © 2002 IEEE

By using the same arguments used in [16], when $p << 1$, Eq. (20) may be approximated as:

$$r^{West} = W_{MAX} / RTT \quad (21)$$

and by considering Eqs. (1) and (21) it follows that

$$r^{West} = \min\left[\frac{W_{MAX}}{RTT}, B\frac{RTT_{\min}}{2RTT} + \sqrt{\left(B\frac{RTT_{\min}}{2RTT}\right)^2 + \frac{1-p}{pRTT^2}}\right] \quad (22)$$

whereas the TCP Reno throughput reported in [16] is

$$r^{Reno} = \min\left[\frac{W_{MAX}}{RTT}, \frac{1}{RTT\sqrt{\frac{2p}{3}} + T_0 \min\left(1, \sqrt{\frac{27p}{8}}\right)\cdot(p+32p^3)}\right] \quad (23)$$

3.4. Empirical validation of the model

In order to validate the proposed equation models we test Westwood++ on the Internet. We carry out a set of 30 ftp uploads using three different servers. Our client runs a Linux 2.2.20 implementation of Westwood++ TCP and is located at "Politecnico di Bari", which is in the South of Italy. The first server is located at the "Politecnico di Torino", in the North of Italy, the second one is located at the "Uppsala University", Sweden, and the third one is located at the University of California, Los Angeles. On 17 November 2001, we transferred 4 Mbytes for 30 consecutive uploads on each server. At the end of each transfer we have measured: (1) the mean value of the bandwidth estimate B; (2) the drop probability p computed as the ratio between the number of retransmitted segment and the total number of sent segments; (3) the mean RTT and (4) the minimum RTT_{min}. Then we have computed the mean throughputs as predicted by Eqs. (22) and (7) and we have compared them to the real measured throughput. For each transfer, we have computed the *relative error* as follows: $Relative\_Error = \left|\frac{Predicted\_Throughput}{Real\_Throughput} - 1\right|$ and for each server the *normalized error* as follows:

$$Normalized\_error = \frac{\sum_i Relative\_Error_i}{Total\_number\_of\_measurements}$$

We report collected data for Westwood++ and Reno in the tables I, II, III and IV. We have noticed that the connections were not limited by W_{MAX}.

From Tables I, II and III we observe that the normalized errors obtained by using the equation model (22), are 37.5% for the server located in Torino, 44% for the server located in Sweden and 11.7% for the server located in Los Angeles. These errors are comparable to the ones that have been reported in [16] to validate the Reno equation model (23). By using the approximated equation model (7), the normalized errors are 59.9% for the server in Torino, 92% for the server in Sweden and 41.7% for the server in Los Angeles. In [16] comparable errors were found for the throughput of TCP Reno when predicted using equation model (13).

Table IV reports measurements for Reno connections. Data show that Reno obtains a slight smaller throughput than Westwood++.

4. Conclusions

We have proposed the new Westwood++ algorithm for Internet congestion control. We have shown via mathematical modeling that Westwood++ TCP is stable, improves the fairness with respect to Reno TCP and it is friendly to Reno TCP. Finally, the model has been validated via live Internet measurements.

5. Acknowledgments

We thank Profs. Mario Gerla, Mikael Sternad and Claudio Casetti who allowed us to collect live Internet measurements using servers located at their Universities in Los Angeles, Uppsala and Torino, respectively.

6. References

[1] V. Jacobson, "Congestion Avoidance and Control", *ACM SIGCOMM '88 conference*, August 1988, pp. 314-329.

[2] V. Jacobson, "Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno", *18th Internet Engineering Task Force*, University of British Colombia, Vancouver, BC, September 1990.

[3] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, "TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks", *ACM Mobicom 2001*, July, Rome, Italy. To appear in ACM Wireless Networks (WINET), Special Issue on Wireless Networks with selected papers from MOBICOM 2001.

[4] D. Clark, "The design philosophy of the DARPA Internet protocols", *ACM SIGCOMM '88 conference,* August 1988, pp. 106-114.

[5] S. Floyd, K. Fall, "Promoting the use of end-to-end congestion control in the Internet", *IEEE/ACM Trans. on Networking*, August 1999, pp. 458-472.

[6] K.J. Astrom , B. Wittenmark, *Computer controlled systems*, Prentice Hall, Englewood Cliffs, N. J., 1997.

[7] F. Kelly, "Mathematical modelling of the Internet", *Fourth International Congress on Industrial and Applied Mathematics*, July 1999.

[8] S. Mascolo, "Congestion control in high-speed communication networks", *Automatica*, Special Issue on Control Methods for Communication Networks, December 1999.

[9] W. Stevens, "TCP/IP illustrated", *Addison Wesley*, Reading, MA, 1994.

[10] L. Peterson & B. Davie, "Computer Networks a Systems Approach", Morgan Kaufmann, 1996

[11] S. Q. Li and C. Hwang, "Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks", *IEEE/ACM Trans. Networking*, February 1995, pp. 10-25.

[12] L.S. Brakmo, S.W. O'Malley, and L. Peterson, "TCP Vegas: End-to-end congestion avoidance on a global Internet", *IEEE Journal on Selected Areas in Communications (JSAC)*, 1995, pp. 1465-1480.

[13] J.C. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", *ACM Sigcomm 1996 conference*, pp. 270-280.

[14] S. H. Low, L. Peterson and L. Wang, "Understanding Vegas: A Duality Model", *ACM Sigmetrics*, Boston, MA, June 2001.

[15] A. Capone, F. Martignon, "Bandwidth Estimates in the TCP Congestion Control Scheme", *Tyrrhenian IWDC 2001*, Taormina, Italy, September 2001.

[16] J. Padhye, V. Firoiu, Don Towsley, J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", *ACM Sigcomm 1998 conference*.

[17] Dah-Ming Chiu, R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", *Computer Networks and ISDN Systems*, 10 June 1989, pp. 1-14.

[18] M. Allman and V. Paxson, "On Estimating End-to-End Network Path Properties", *ACM Sigcomm 1999 conference*.

[19] K. Lai and M. Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay", *ACM Sigcomm 2000 conference*.

TABLE I. Westwood++ connections from Bari to Torino

Torino West++	Total Byte sent (Mbyte)	Byte retransm. (Mbyte)	Drop prob. (10^{-3})	RTT (ms)	RTT_{min} (ms)	Bandwidth estimation (MByte/s)	Predicted throughputs (Mbyte/s)		Measured Goodput (Mbyte/s)	Measured Throughput (Mbyte/s)	Relative Errors %	
							Eq. (22)	Eq. (7)			Eq. (22)	Eq. (7)
MAX	4.688	0.493	105	168	136	0.168	0.114	0.106	0.174	0.184	45.2	71.3
AVG	4.488	0.294	65	148	93.3	0.144	0.104	0.067	0.157	0.167	37.5	59.9
MIN	4.400	0.205	47	128	68	0.094	0.096	0.048	0.137	0.149	32.3	40.4

TABLE II. Westwood++ connections from Bari to Uppsala

Uppsala West++	Total Byte sent (Mbyte)	Byte retransm. (Mbyte)	Drop Prob. (10^{-3})	RTT (ms)	RTT_{min} (ms)	Bandwidth estimation (MByte/s)	Predicted throughputs (Mbyte/s)		Measured Goodput (Mbyte/s)	Measured Throughput (Mbyte/s)	Relative Errors %	
							Eq. (22)	Eq. (7)			Eq. (22)	Eq. (7)
MAX	6.143	1.948	317	148	81	0.338	0.242	0.037	0.281	0.399	60.6	94.8
AVG	5.976	1.781	298	114.5	73.1	0.314	0.206	0.028	0.258	0.369	43.9	92
MIN	5.709	1.515	265	96	70	0.299	0.150	0.02	0.232	0.341	34	90

TABLE III. Westwood++ connections from Bari to Los Angeles

Los Angeles West++	Total byte sent (Mbyte)	Byte retransm. (Mbyte)	Drop Prob. (10^{-3})	RTT (ms)	RTT_{min} (ms)	Bandwidth estimation (MByte/s)	Predicted throughputs (Mbyte/s)		Measured Goodput (Mbyte/s)	Measured Throughput (Mbyte/s)	Relative Errors %	
							Eq. (22)	Eq. (7)			Eq. (22)	Eq. (7)
MAX	4.633	0.439	95	235	199	0.104	0.12	0.172	0.104	0.109	15.7	64.2
AVG	4.425	0.231	52	231.7	195	0.097	0.093	0.072	0.097	0.102	11.71	41.7
MIN	4.239	0.045	10.6	228	189	0.092	0.083	0.041	0.089	0.095	2.38	8.5

TABLE IV. Reno connections

Reno	Measured Goodput (Mbyte/s)			Measured Throughput (Mbyte/s)		
	Torino	Uppsala	Los Angeles	Torino	Uppsala	Los Angeles
MAX	0.174	0.263	0.094	0.185	0.304	0.0949
AVG	0.155	0.234	0.084	0.164	0.277	0.0864
MIN	0.127	0.206	0.067	0.135	0.245	0.0736