4th IFAC Workshop on Time Delay Systems (TDS'03)
Rocquencourt, France, September 8-10, 2003

MODELING THE INTERNET CONGESTION CONTROL AS A TIME DELAY SYSTEM: A ROBUST STABILITY ANALYSIS

Saverio Mascolo

Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari
Via Orabona 4, 70125 Italy

Abstract: In this paper control theoretic analysis is employed to model the Internet flow and congestion control as a time delay system. It is shown that the self-clocking principle, which is known to be a key component of the Transmission Control Protocol (TCP), corresponds to implement a simple proportional controller (P) plus a Smith predictor (SP). It is also shown that different variants of the TCP congestion control algorithms, such as classic TCP Reno or the recent Westwood TCP, can be modelled by proper input shaping of the reference signal. Finally, the robustness of the control algorithm in the face of uncertain round trip time delay is analysed.

Keywords: Telecommunication, Internet, Time-delay systems.

1. INTRODUCTION AND RELATED WORKS[1]

The Internet has shown a great capability of exponential growing, without incurring congestion collapse, thanks to the TCP/IP congestion control algorithm. The stability of the Internet and in particular the prevention of congestion requires that flows use some form of end-to-end congestion control to adapt the input rate to the available bandwidth (Jacobson, 1988; Floyd *et al*., 1999).

The Transmission Control Protocol (TCP) has two feedback mechanisms to tackle congestion: the *flow control* and the *congestion control*. The TCP *flow control* aims at avoiding the overflow of the receiver's buffer and is based on explicit feedback. In particular, the TCP receiver sends to the source the Receiver's Advertised Window, which is the buffer available at the receiver. The TCP *congestion control* aims at avoiding the flooding of the network and is based on implicit feedback such as timeouts, duplicate acknowledgments (DUPACKs), round trip time measurements. In the latter case, the source infers the network capacity using an additive-increase/multiplicative-decrease (AIMD) probing paradigm. The *increase* phase aims at increasing the flow input rate until the network available capacity 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 (*fast recovery*) and sending again the missing packet (*fast retransmit*), and to heavy congestion (i.e. timeout) by reducing the congestion window to one. Both the flow and congestion control implements the self-clocking principle, that is, when a packet exits a new one enters the network. The described mechanisms form the core of the classic Internet congestion control algorithm known as Tahoe/Reno TCP (Jacobson, 1988; Allman *et al*., 1999). It is interesting to notice that these mechanisms continue to be the core of all enhanced TCP congestion control algorithms that are successful.

Research on TCP congestion control is still active in order to improve its efficiency and fairness, especially in new environments such as the wireless Internet (Mascolo *et al*., 2001) or the high-speed Internet (Jacobson et al., 1992; Floyd, 2003). An overview of the most significant modifications that

[1]This work was supported by the MIUR-FIRB project n. RBNE01BNL5 "Traffic Models and Algorithms for Next Generation IP networks Optimisation (TANGO)"

have been proposed, such as New Reno, Vegas, Santa Cruz etc., will be given in the extended version of this paper.

TCP Westwood (Mascolo *et al*., 2001) uses an end-to-end estimation of the available bandwidth to adaptively set the control windows after congestion. In (Gerla *et al*., 2002) the concept of *generalized advertised window* has been proposed to provide an explicit indication of the network congestion status. Recently, non linear stochastic differential equations have been proposed to model the TCP dynamics (Grieco *et al*., 2002; Kelly, 2001; Misra *et al*. 2000. In particular, the dynamics of the expected value of the *cwnd* has been expressed as a function of the packet dropping probability through a non-linear differential equation.

In this paper a general control theoretic framework to model the TCP flow and congestion control along with its variants such as Reno and Westwood is proposed. The work is organized as follows: Section 2 outlines the TCP flow and congestion control algorithm; Section 3 models the TCP flow and congestion control using transfer functions and a proportional controller plus a Smith predictor; Section 4 shows that the Smith predictor enforces the *self-clocking* principle and provides stability; Section 5 analyses the robust stability in the face of uncertain delays; Section 6 models the TCP Reno and Westwood control algorithms by properly setting the controller input; finally, Section 7 draws the conclusions. In the extended version of this paper it will be also shown that simple *PID* controllers provide a too sluggish behaviour when used in packet communication networks such as in the case of the Internet.

2 TCP/IP FLOW AND CONGESTION CONTROL

A TCP connection is a virtual pipe between the send socket buffer and the receive socket buffer (see Fig. 1). The TCP has two feedback mechanisms to tackle congestion: the *flow control mechanism* that prevents the sender from overflowing the receiver's buffer, and the *congestion control mechanism* that prevents the sender from overloading the network.

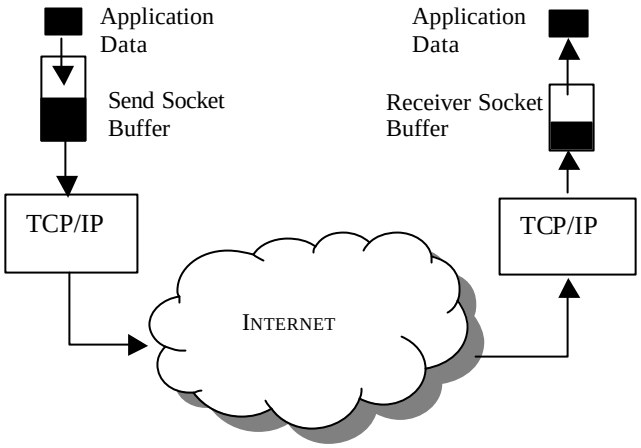

Fig. 1. Schematic of a TCP connection.

2.1 The flow control algorithm

The TCP flow control is based on *explicit feedback*. In particular, the TCP receiver sends to the source the Receiver's *Advertised Window*, which is the buffer available at the receiver. Let *MaxRcvBuffer* be the size of the receiver buffer in bytes, *LastByteRcvd* the last byte received and *NextByteRead* the next byte to be read. On the receive side TCP must keep

$$LastByteRcvd - NextByteRead \leq MaxRcvBuffer$$

to avoid overflow. Therefore, receiver advertises a window size (*AdWnd*) of

$$AdWnd = MaxRcvBuffer - (LastByteRcvd - NextByteRead)$$

which represents the amount of free space remaining in the receiver buffer. The TCP on the send side computes an Effective Window W

$$W = AdWnd - (LastByteSent - LastByteAcked) \quad (1)$$

which limits the number of outstanding packets.

2.2 The congestion control algorithm

Congestion control aims at inferring "some measurement" of the network capacity at the end nodes using implicit feedback such as timeout and acknowledgments.

Today TCP estimates the best effort capacity of the network using a variable called *congestion window* (*cwnd*). In particular, the TCP learns the appropriate value of the *cwnd* by using an additive-increase/multiplicative-decrease (AIMD) paradigm. The increasing process goes through two phases: the *slow start* and the *congestion avoidance*. During the slow start phase the *cwnd* is exponentially increased until the *slow start threshold* (*ssthresh*) value is reached. This phase is intended to quickly grab available bandwidth. After the *ssthresh* value is reached, the *cwnd* is linearly increased to gently probe for extra available bandwidth. This phase is called *congestion avoidance*. At some point the TCP connection starts to lose packets. After a timeout *cwnd* is drastically reduced to one and the slow start, congestion avoidance cycle repeats. After 3 DUPACKs *cwnd* is halved and the congestion avoidance phase is entered (Jacobson, 1988).

The TCP sender computes the minimum of the congestion window and the advertised window in order to implement both flow and congestion control. In particular, it computes the *Effective Window W* as follows

$$W = MIN(Cwin, AdvWin) - OutstandingPackets \quad (2)$$

where

$$OutstandingPackets = LastByteSent - LastByteAcked$$

are the in flight packets.

3. MODELING THE TCP FLOW AND CONGESTION CONTROL

Van Jacobson (1988) clearly states: "A packet network is to a large extend a linear system made of *gains*, *delays* and *integrators*". In particular this work makes a model of a TCP/IP connection using: (a) two integrators: one is used to model the receiver buffer, the other is employed to model the storage capacity of the network viewed as a whole; (b) two delays: one is the forward delay T_{fw} from the sender to the receiver, the other is the backward delay T_{fb} from the receiver to the sender; (c) a gain k for the proportional controller; (d) the Smith predictor.

The dynamic equation of a generic buffer length $x(t)$ is:

$$x(t) = \int_{-\infty}^{t} u(t) - b(t) - o(t)\,dt$$

where $u(t)\geq 0$ is the data arrival rate, $b(t)\geq 0$ is the data depletion rate, i.e. the used bandwidth, and $o(t)\geq 0$ is the overflow data rate, i.e. the data that are lost when the buffer is full and the input rate exceeds the output rate. The proposed linear model of the TCP flow and congestion control algorithm is shown in the functional block diagram reported in Fig. 2. In particular, Fig. 2 shows:

1) The receiver buffer x_r and the network storage capacity x_n that modelled by two integrators with Laplace transfer function 1/s. The receiver buffer receives the inputs $u_r(t)$, $b_r(t)$, $o_r(t)$, which represent the input rate, the depletion rate and the overflow data rate, respectively. Analogously, the buffer that models the network storage capacity receives the inputs $u_n(t)$, $b_n(t)$, $o_n(t)$; notice that the network depletion rate $b_n(t)$ represents the bandwidth used by the TCP connection, which is the best-effort bandwidth left available by the Internet coexisting traffic; the depletion rate $b_n(t)$ reaches the receiver buffer as an input rate, i.e. $u_r(t)=b_n(t)$; the network data arrival rate $u_n(t)$ is equal to the delayed TCP sending rate, i.e. $u_n(t)= u_s(t-T_{fw})$;
2) The forward and backward propagation delays are modelled in the Laplace domain by the transfer functions $e^{-sT_{fw}}$ and $e^{-sT_{fb}}$;
3) The receiver queue length x_r and the receiver capacity r_1 provide the *Advertised Window, i.e.* $r_1 - x_r$ which reaches the sender after the delay T_{fb};
4) The set point $r_2(t)$ represents a threshold for the network storage capacity modelled by queue $x_n(t)$;
5) The minimum block takes the minimum between the *Advertised Window* and $r_2(t) - x_n(t - T_{fb})$;
6) The controller transfer function

$$G(s) = \frac{k}{1 + \frac{k}{s}(1 - e^{-sT})},$$

contains the proportional gain k and the Smith predictor $(1 - e^{-sT})/s$, where $T=T_{fw}+T_{fb}$; the role of the Smith predictor is to overcome the delay T, which is inside the feedback loop and is harmful for the stability of the closed-loop control system (Mascolo, 1999).

It should be noted that the inner feedback loop controls the packets stored in the network, i.e. it implements the TCP congestion control, whereas the outer loop controls the receiver buffer, i.e. it implements the TCP flow control. A detailed dynamic model of a generic TCP/IP connection that considers all routers in the connection path will be reported elsewhere. The analysis shows that a TCP connection interacting with the whole Internet can indeed be modelled, without loss of generality, as it is shown in Fig. 2.

In order to show that the block diagram in Fig. 2 models the TCP/IP flow and congestion control, first we will assume that the bottleneck is at the receiver and then that the bottleneck is inside the network.

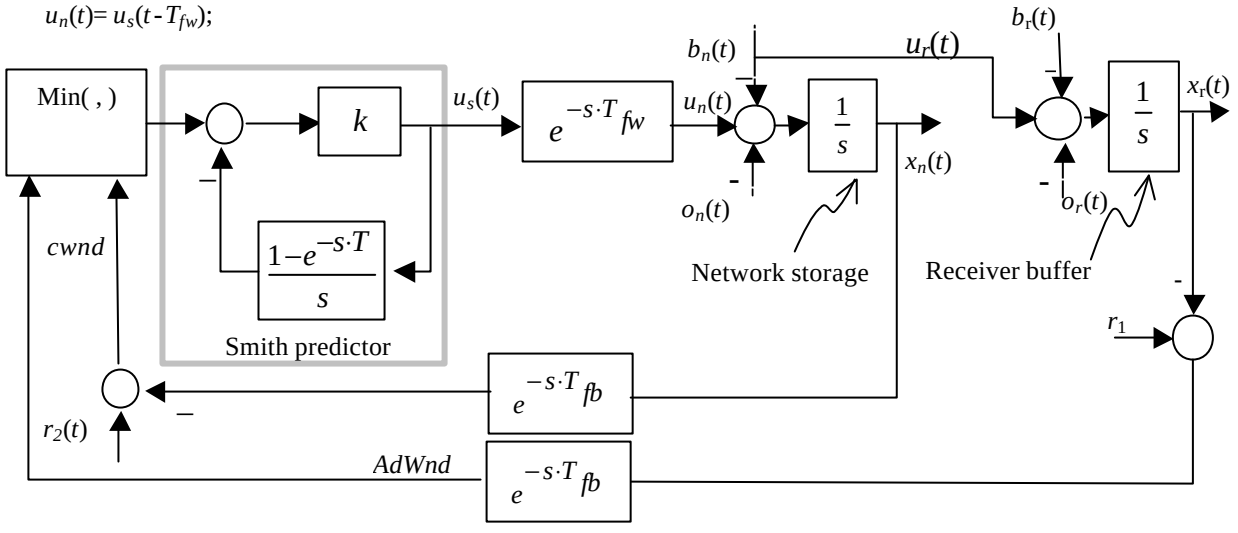

Fig. 2: Functional block diagram of the TCP/IP flow and congestion control

3.1 The flow control

By assuming that the bottleneck is at the receiver, we can ignore the inner feedback loop. To find the input rate $u_s(t)$ computed by the TCP sender we use standard Laplace techniques, that is, we compute the Laplace transform of the input rate that is:

$$U_S(s) = [R_1(s) - X_r(s)]e^{-sT_b} \frac{k}{1 + k\left(\frac{1 - e^{-sT}}{s}\right)}$$

that can be written as

$$U_S = -kU_S\left(\frac{1 - e^{-sT}}{s}\right) + k[R_1 - X_r]e^{-sT_{fb}}$$

By transforming back to time domain it results:

$$\frac{u_S(t)}{k} = r_1(t - T_{fb}) - x_r(t - T_{fb}) - \int_{t-T}^{t} u_S(t)dt \quad (3)$$

By considering that

$$r_1(T - T_{fb}) - x_r(T - T_{fb}) = \text{Advertised window}$$

and that

$$\int_{t-T}^{T} u_S(t)dt = \text{Outstanding packets}$$

Equation (3) gives the classic window-based flow control equation (1), where $W = u_s(t)/k$. By considering the relation $u_s(t) = W/T$ that relates the rate and the window size of a window-based control, it results that $1/k = T$.

3.2 The congestion control

Now, by assuming that bottleneck is localized in the network, we can ignore the outer feedback loop. The queue length x_n represents the packets stored in the network. The input rate computed by the controller is now

$$u_S(t)T = r_2(t) - x_n(t - T_{fb}) - \int_{t-T}^{t} u_S(t)dt \quad (4)$$

It is important to notice that, differently from the Advertised window, the queue length x_n is not available as explicit feedback in the Internet. This means that the inner feedback loop in Fig. 2 is not present. However, we are able to show that the Smith predictor branch is able to provide the queue length x_n: in fact, queuing increases the actual round trip time that becomes:

$$\textit{round trip time} = T + \Delta T$$

where ΔT is the queuing time and T is the propagation time. Since the round trip time is continuously monitored by the TCP sender using packet time stamping, the branch that implements the Smith predictor *automatically* employs the correct value for the *round trip time*, that is, it implements the branch $\left(1 - e^{-s(T+\Delta T)}\right)/s$. Therefore the block diagram of the actual closed-loop system is the one shown in Fig. 3.

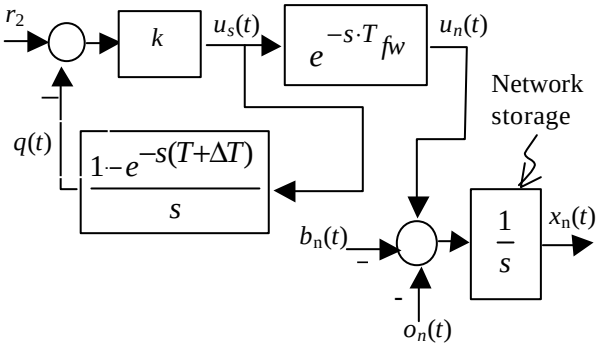

Fig. 3: Functional block diagram of the TCP congestion control without the x_n feedback.

The Smith predictor branch computes the quantity:

$$Q(s) = U_S(s)\left(1 - e^{-s(T+\Delta T)}\right)/s$$

that transformed back to time domain turns out:

$$q(t) = \int_{t-T-\Delta T}^{t} u_S(t)dt = x_n(t - T_{fb}) + \int_{t-T}^{t} u_S(t)dt$$

which represent the packets stored in the network plus the in flight packets. In other words, the Smith predictor branch in Fig. 3 automatically takes into account the network storage without requiring the explicit feedback of x_n and implements the equation:

$$\frac{u_S(t)}{k} = r_2(t) - \textit{outstanding packets} \quad (5)$$

where now $1/k = T + \Delta T$. Eq. (5) is identical to the control equation (4) that has been derived using the explicit feedback of x_n. The classic TCP congestion and flow control window-based equation (2) is then obtained by setting $r_2(t) = cwnd$ and by taking the minimum of ($AdWnd$,$Cwnd$).

4. THE SELF-CLOCKING PRINCIPLE AND THE STABILITY

In this Section we show two results: (1) the Smith predictor enforces the *self-clocking* principle; (2) the TCP congestion control algorithm is stable. We start by the *self-clocking* principle. It is known that the *self-clocking* principle is a key feature of the TCP congestion and flow control (Jacobson, 1988). This has been recently recognized also in the context of Transport Friendly Rate Control (TFRC) algorithms (Bansal *et al*. 2001), where it has been shown that algorithms that do not employ the self-clocking principle may exhibit a huge settling time, that is, they may require many RTTs to adapt the input rate to the bandwidth available in the network. In the extended version of this paper we will show that this

behaviour also happens when PID controllers are used. As a consequence, to overcome the disastrous effects due to the violation of the self-clocking principle, the original TFRC has been enhanced with the self-clocking mechanism. We show that the *self-clocking* can be mathematically interpreted as the effect of the Smith predictor branch.

Proposition 1. The Smith predictor branch $\left(1-e^{-sT}\right)/s$ enforces the *self-clocking* principle.

***Proof*:** By transforming back to time domain the quantity $q(s)=U_s(s)\left(1-e^{-sT}\right)/s$ it results

$$q(t)=\int_0^t u_S(t)dt-\int_0^{t-T}u_S(t)dt=\int_{t-T}^{t}u_S(t)dt\ ,$$

$q(t)$ represents the data that have been sent since the last round trip time T up to now, i.e. the outstanding packets. When the time t advances of Δ, the data $q_{acked}=\int_{t-T}^{t-T+\Delta}u_S(t)dt$ are acknowledged so that the same amount of data can be sent in the time interval [t, $t+\Delta$] by the control equation (3) or (5), that is, the self-clocking principle is enforced.♦

It is well know that feedback control of time-delay systems may produce an unstable system. It is also known that the Smith predictor is an effective technique to overcome delays and build an efficient and stable controller for systems with known dynamics. This is the case of the TCP that exactly knows the round trip time T via packet time stamping (Jacobson, 1988).

Proposition 2. The Smith predictor stabilizes the feedback controlled TCP connection shown in Fig. 2.

***Proof*:** From Fig. 2, it is possible to derive the input-output transfer functions of the flow and congestion control that are, respectively:

$$\frac{X_r(r)}{R_1(s)}=\frac{k}{k+s}e^{-sT}\ \text{ and }\ \frac{X_n(r)}{R_2(s)}=\frac{k}{k+s}e^{-sT_{fw}}$$

Both these transfer functions have the unique pole $p=-k$, which is asymptotically stable for any positive gain $k>0$. The time-constant is $t=1/k$.
In both cases, the disturbance-output transfer functions are:

$$\frac{X_r(r)}{O_r(s)+B_r(s)}=\frac{X_n(r)}{O_n(s)+B_n(s)}=-\frac{1}{s}+\frac{e^{-sT}}{s(k+s)}$$

Again the pole $p=-k$ is asymptotically stable for any positive gain $k>0$, whereas the pole $p=0$ is marginally stable, which concludes the proof.♦

5. ROBUST STABILITY ANALYSIS

In this section the stability of the TCP congestion control in the presence of uncertainty on the round trip time T is analysed. A bound for the error on T, which still guarantees the stability of the control algorithm, is provided.

Proposition 3: Let T_1 be the exact round trip time and let assume that the TCP sender knows the perturbed round trip time $T_2=T_1+DT$, where DT is a bounded uncertainty. A sufficient condition for the TCP congestion control to be stable is $k\cdot\Delta T<1$.

***Proof*:** Assuming that the exact round trip time is T_1 and that the TCP knows the round trip time T_2, the loop transfer function is $G(s)\frac{e^{-sT_1}}{s}$, where $G(s)$ is the controller $\frac{k}{1+k(1-e^{-sT_2})/s}$. The characteristic equation of the closed-loop system is given by:

$1+\frac{ke^{-sT_1}}{1+\frac{k}{s}(1-e^{-sT_2})}\frac{1}{s}=0$, which can be rewritten as $1+\frac{k}{s+k}\left(e^{-sT_1}-e^{-sT_2}\right)=0$. For the Nyquist criterion, a sufficient condition for the stability is

$$\left|\frac{k}{jw+k}\left(e^{-jwT_1}-e^{-jwT_2}\right)\right|<1 \text{ for any } w$$

which turns out the condition

$\cos w\Delta T>0.5-w^2/2k^2$ for any w.

A sufficient condition that guarantees that $f_1(w)=\cos w\Delta T>f_2(w)=0.5-w^2/2k^2$ is:

$$\left|\frac{df_2(w)}{dw}\right|>\left|\frac{df_1(w)}{dw}\right|$$

which turns out the condition: $w/k^2>\Delta T\sin w\Delta T$. The latter condition is satisfied if

$$\left|\frac{df_2^{\,2}(0)}{dw^2}\right|>\left|\frac{df_1^{\,2}(0)}{dw^2}\right|,$$

i.e., if $k\cdot\Delta T<1$, which concludes the proof.

6. MODELING RENO OR WESTWOOD TCP BY INPUT SHAPING

In this section we show that the dynamic model depicted in Fig. 2 is able to model successful variants of TCP congestion control, such as Tahoe/Reno (Jacobson, 1988) or the recent Westwood TCP (Mascolo et al. 2001). We have seen that the congestion control algorithm must estimate the available bandwidth using a probing mechanism. The classic TCP probing mechanism, which is currently used in all successful variants of the TCP such as Tahoe/Reno, New Reno or Westwood, comprises two mechanisms: the *slow-start* phase, which

exponentially increase the congestion window up to the *ssthresh*, and the *congestion avoidance* phase which linearly increase the *cwnd* when *cwnd*≥ *ssthresh*. Now we show that both these mechanisms can be modelled in the control theoretical framework reported in Fig. 2 by properly shaping the controller input $r_2(t)$.

6.1 The Reno algorithm

The TCP Reno *slow-start* phase can be modelled by setting the reference input $r_2(t)$ as follows:

$$r_2(t) = r_0 + 2^{\frac{t}{T}} \qquad while \qquad r_2(t) < ssthresh$$

where the initial window r_0 is generally equal to 1 or 2. TCP Reno enters the *congestion avoidance* phase when $r_2(t)$=*ssthresh* at $t_1 = T \log_2(ssthreh - r_0)$. This phase can be modelled by setting the reference input $r_2(t)$ as follows:

$$r_2(t) = ssthresh + \frac{t - t_1}{T} \qquad when \qquad r_2(t) \geq ssthresh$$

The TCP probing phase ends when 3 DUPACKSs are received or a timeouts happens, which indicate that the network capacity has been hit. In these cases the *cwnd* behavior can be modelled using the following settings for $r_2(t)$:

After a timeout at t_k

$$ssthresh = \frac{r_2(t)}{2}$$

$$r_2(t) = r_0$$

$$r_2(t) = r_0 + 2^{\frac{t - t_k}{T}} \qquad if \qquad r_2(t) < ssthresh$$

$$r_2(t) = ssthresh + \frac{t - t_k}{T} \qquad if \qquad r_2(t) \geq ssthresh$$

After 3 DUPACKs at t_k

$$ssthresh = \frac{r_2(t)}{2}$$

$$r_2(t) = ssthresh + \frac{t - t_k}{T}$$

6.2 The Westwood algorithm

TCP Westwood employs the same probing mechanism of Reno. It differs from Reno because of the behaviour after congestion. In fact, Westwood sets the *cwnd* and *ssthresh* using an end-to-end estimate of the network available bandwidth *B*. In particular, the Westwood TCP behaviour after congestion can be modelled as follows:

After a timeout at t_k

$$ssthresh = B \times \min(RTT)$$

$$r_2(t) = r_0$$

$$r_2(t) = r_0 + 2^{\frac{t - t_k}{T}} \qquad if \qquad r_2(t) < ssthresh$$

$$r_2(t) = ssthresh + \frac{t - t_k}{T} \qquad if \qquad r_2(t) \geq ssthresh$$

After 3 DUPACKs at t_k

$$ssthresh = B \times \min(RTT)$$

$$r_2(t) = ssthresh + \frac{t - t_k}{T}$$

7. CONCLUSIONS

In this paper it has been shown that (1) a proportional controller plus a Smith predictor provides an exact model of the Internet flow and congestion control; (2) a model of successful TCP congestion control algorithms, such as classic Reno or recent Westwood TCP, can be derived by proper shaping of the controller reference signal. Finally, the robust stability of the Internet congestion control has been analysed in the face of uncertain measurement of the round trip time.

REFERENCES

Bansal D., Balakrishnan H., Floyd S., Shenker S. (2001). Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", *Proc. of Sigcomm.*

Floyd, S., (2003). HighSpeed TCP for Large Congestion Windows. *IETF Draft in progress.*

Floyd, S., Fall, K., (1999). Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, **vol.7**(4), 458-72.

Gerla M., Locigno R., Mascolo S., Weng R. (2002). Generalized Window Advertising for TCP Congestion Control. *European Transactions on Telecommunications*, **6**, pp. 549-562.

Grieco, L. A., Mascolo, S. (2002). TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks. *Proc. of the VII Int. Workshop on Protocols for High-Speed Networks.* Lecture Notes on Computer Science. Springer Verlag.

Jacobson, V., Braden, R., Borman D. (1992), TCP Extensions for High Performance, *RFC 1323.*

Jacobson V. (1988) Congestion Avoidance and Control. ACM Computer Communications Review, **18** (4): 314 – 329.

Kelly, F. P., (2001). Mathematical modelling of the Internet. In: *Mathematics unlimited - 2001 and beyond*. Editors B. Engquist and W. Schmid. Springer-Verlag, 685-702, Berlin, 2001.

Mascolo S. (1999). Congestion control in high-speed communication networks using the Smith principle. *Automatica*, **vol. 35**(12), 1921-1935.

Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., Wang, R., (2001). TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. Proc. of *ACM Mobicom.*

Misra, V., Gong, W., Towsley, D., (2000). A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. *Proc. of Sigcomm*.