Modeling the TCP/IP Flow and Congestion Control Using a Smith Controller with Input Shaping

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

***Abstract*— This paper proposes a classical control theoretic approach to model the TCP flow and congestion control. In particular, it shows that classic sliding window control, employed by both the TCP flow and congestion control algorithms, can me modeled using a proportional controller (P) plus a Smith predictor (SP), which compensates feedback delays due to propagation times. Moreover, it shows the stability properties of the TCP and models different variants of TCP congestion control algorithms, such as classic TCP Reno or the recent Westwood TCP, in the same unified control framework by properly shaping the controller reference input. The proposed analysis gives a simple and rigorous insight into TCP flow and congestion control and provides an effective framework to design new control algorithms that are TCP friendly. As an example, an application to the design of a TCP friendly rate control algorithm is given**.

Keywords: TCP, TCP Modeling, Congestion Control Design, Rate-based Control

I. 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 [1], [5], [6]. Since its introduction in the late eighties [1], the Transmission Control Protocol (TCP) congestion control has been quite successful in preventing congestion collapse.

The (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. 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 increase/decrease probing paradigm [8]. 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 [1], [4]. It is interesting to notice that these mechanisms continue to be at the core of all enhanced TCP congestion control algorithms.

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 [9], [12], [25] or the high-speed Internet [20], [26]. We briefly summarize the most significant modifications that have been proposed up to now.

The New Reno feature is an enhancement of Reno that has been proposed to avoid multiple window reductions in a window of data [10]. TCP Vegas estimates the expected connection rate as $cwnd/RTT_m$ and the actual connection rate as $cwnd/RTT$; when the *difference* between the expected and the actual rate is less than a threshold $\alpha>0$, the *cwnd* is additively increased. When the *difference* is greater than a threshold $\beta>\alpha$ then the *cwnd* is additively decreased. When the difference is between α and β, *cwnd* is maintained constant [11]. TCP Westwood uses an end-to-end estimation of the available bandwidth to adaptively set the control windows after congestion [12], [13]. Both Vegas and Westwood preserve the standard multiplicative decrease behavior after a timeout. TCP Santa Cruz proposes to use estimate of delay along the forward path rather than round trip delay and to reach a target operating point for the number of packets in the bottleneck of the connection [14].

TCP Westwood [12] uses an end-to-end estimation of the available bandwidth to adaptively set the control windows after congestion. In [3] 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 dynamics of the TCP congestion

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

window (*cwnd*) [13], [15]-[17]. In these models, the dynamics of the expected value of the *cwnd* is mainly expressed as a function of the packet drop probability through a non-linear differential equation. These models, and their linearized ones, have been used to predict the long-term TCP throughput and to design control laws for throttling the packet drop probability of routers implementing Active Queue Management [27]. In particular in [27], the mentioned nonlinear stochastic differential model of the TCP window has been linearized around the equilibrium to derive a transfer function from the packet drop probability to the bottleneck queue length. The linearized model had been employed to design a control law for the packet drop rate aiming at stabilize the queue average length. It is not clear how effective is the model to deal with real-time dynamics of TCP and in presence of multi-bottleneck topologies.

This paper proposes a classical control theoretic approach to model the TCP flow and congestion control, along with its variants such as for example Reno and Westwood, in a unified framework. The model is general and captures multi-bottlenecks as well as moving bottleneck. The work is organized as follows: Section 2 outlines the TCP flow and congestion control algorithm; Section 3 models a generic TCP going over a store-and-forward shared networks using buffers and integrators; Section 4 models the TCP flow and congestion control using a Smith predictor and a proportional controller; Section 5 models different TCP algorithms, such as Reno and Westwood TCP, by properly setting the controller input; Section 6 proposes an application of the developed model to design a TCP friendly rate-based control algorithm; finally, Section 7 draws the conclusions.

II. TCP FLOW AND CONGESTION CONTROL

A TCP connection establishes a virtual pipe between the send socket buffer and the receive socket buffer as shown in Fig. 1.

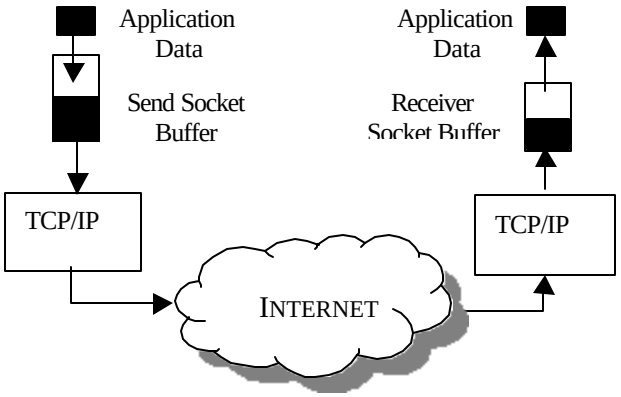

Fig. 1. Schematic of a TCP connection

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.

A The TCP 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

$$LastByteRcvd - NextByteRead \le MaxRcvBuffer$$

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 [7].

B The TCP Congestion Control Algorithm

The TCP congestion control employs a *trial and error* probing mechanism aiming at learning the network capacity using only implicit feedback such as timeouts and acknowledgments. In particular, the TCP estimates the best effort capacity of the network by increasing and decreasing the *congestion window* variable. There are two increasing 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 [1].

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

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

where

$$OutstandingPackets = LastByteSent - LastByteAcked$$

are the in flight packets [7].

III. MODELLING A GENERIC TCP FLOW

In his milestone paper, Van Jacobson (1988) clearly states that: "A packet network is to a very good approximation a linear system made of *gains*, *delays* and *integrators*" [1]. This paper proposes a detailed model of a TCP/IP connection using (a) integrators to model network and receiver buffers and (b) delays to model propagation times.

A data network is a set of store-and-forward nodes connected by communication links. A generic TCP flow goes through a communication path made of a series of buffers and communication links.

The number of packets of the considered TCP flow that are stored at the generic *i-th* buffer along the communication path is given by the following dynamic equation:

$$x_i(t) = \int_{-\infty}^{t} [u_i(\tau) - b_i(\tau) - o_i(\tau)]d\tau \qquad (3)$$

where $u_i(t) \geq 0$ models the data arrival rate, $b_i(t) \geq 0$ models the data depletion rate, i.e. the used bandwidth, and $o_i(t) \geq 0$ models 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 dynamic equation of the generic communication link (*i*-1) connecting the (*i*-1)-*th* buffer to the next (*i*)-*th* buffer is a pure delay. In particular, letting $b_{i-1}(t)$ be the link input rate at the (*i*-1)-*th* buffer and $u_i(t)$ be the link output rate at the next (*i*)-*th* buffer, it results:

$$u_i(t) = b_{i-1}(t - T_{i-1}) \qquad (4)$$

where T_{i-1} is the link propagation time.

Starting from the basic equations (3) and (4), we propose to model a generic TCP flow over an IP network as it is shown in Fig. 2. In particular, Fig. 2 shows a functional block diagram made of:

1) The TCP connection receiver buffer of length $x_r(t)$, which is modeled using an integrator 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;

2) The *n-th* buffer that the TCP connection goes through before reaching the receiver buffer, which is modeled using an integrator with output $x_n(t)$. The *n-th* buffer receives the inputs $u_n(t)$, $b_n(t)$, $o_n(t)$, which, again, represent the input rate, the depletion rate and the overflow data rate, respectively. It is important to notice that the depletion rate $b_n(t)$ reaches the next buffer (n+1), which is the receiver buffer, after the propagation time T_n, *i.e.* $u_r(t) = b_n(t-T_n)$. Moreover, it should be noted that the input rate $u_n(t)$ is equal to the depletion rate $b_{n-1}(t)$ at the previous (n-1)-th buffer, i.e. $b_{n-1}(t-T_{n-1}) = u_n(t)$, where T_{n-1} is the propagation time from the (n-1)-th buffer to the n-th buffer. Depletion rates are unpredictable because they model the best effort bandwidth available for a TCP connection when going over statistically multiplexed IP network.

The series of buffers shown in Fig. 2 can be recursively augmented both in the left direction, to model up to the first buffer node encountered by the TCP connection, and in the right direction to model buffers *n+j*, with *j=2,p* encountered by ACK packets when going back from the receiver to the sender.

By considering a closed surface that contains the TCP path going from the first to the last buffer modeled by a set of integrators indexed from 1 to n+p=m, where the m-th integrator models the last buffer encountered by the TCP along the connection round trip, we can invoke the flow conservation principle for the unique input rate, which is the TCP input rate $u_1(t)$, and the output rates that are: (a) $b_m(t)$, which models the bandwidth used by the TCP connection, i.e. the best-effort bandwidth as viewed by the considered TCP flow through the ACK stream; and (b) the overflow rates $o_i(t)$, for i=1,m, which represent packets that are lost at each buffer along the path connection.

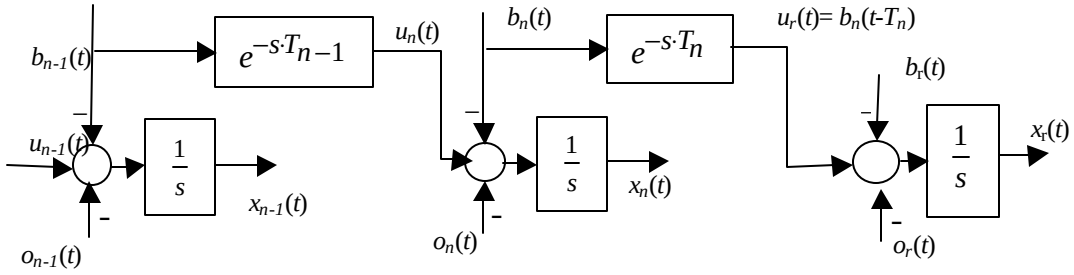

Fig. 2 Dynamic block diagram of a generic TCP/IP flow

In equations, we can write the number $x(t)$ of packets belonging to the considered TCP flow and stored into the network by adding packets stored at each buffer along the path:

$$x(t) = \sum_{i=1}^{m} x_i(t) \tag{5}$$

Substituting (3) in (5) and considering the (4) it turns out

$$x(t) = \int_{-\infty}^{t} [u_1(t) - b_m(t) - \sum_{i=1}^{m} o_i(t) - \sum_{i=1}^{m-1} (b_i(t) - b_i(t - T_i))]dt$$

that can be rewritten as

$$x(t) = \int_{-\infty}^{t} [u_1(\boldsymbol{t}) - b_m(\boldsymbol{t}) - \sum_{i=1}^{m} o_i(\boldsymbol{t})]d\boldsymbol{t} - \sum_{i=1}^{m-1} \int_{t-T_i}^{t} b_i(\boldsymbol{t})d\boldsymbol{t} \tag{6}$$

Eq. (6) states that the network storage is equal to the integral of the TCP input rate $u_1(t)$ minus the output rate $b_m(t)$ leaving the last buffer of the path, minus the sum of the overflow rates $o_i(t)$, minus the sum of packets that are in flight over each link *i*.

Since the TCP implements an end-to-end congestion control that does not receive any explicit feedback from the network, it is not possible for the controller to know terms in (6). Thus, we consider the sum of the in *flight packets* plus the *stored packets*, which we call the total network storage x_t :

$$x_t(t) = x(t) + \sum_{i=1}^{m-1} \int_{t-T_i}^{t} b_i(\boldsymbol{t})d\boldsymbol{t} = \int_{-\infty}^{t} [u_1(\boldsymbol{t}) - b_m(\boldsymbol{t}) - \sum_{i=1}^{m} o_i(\boldsymbol{t})]d\boldsymbol{t}$$

and the sum of overflow rates o_t :

$$o_t(t) = \sum_{i=1}^{m} o_i(t)$$

Thus, we can write

$$x_t(t) = \int_{-\infty}^{t} [u_1(\tau) - b_m(\tau) - o_t(\tau)]d\tau \tag{7}$$

By considering that the TCP establishes a "circular flow", i.e. that the data input rate comes back to the sender as an ACK rate, it can be said that $b_m(t)$ models the rate of ACK packets. Thus we can write:

$$b_m(t) = u_1(t - T) - o_t(t) \tag{8}$$

which says, in mathematical words, that the ACK rate is equal to the input rate, delayed by the round trip time, minus the loss rate. By substituting (8) in (7) it turns out:

$$x_t(t) = \int_{-\infty}^{t} [u_1(\tau) - u_1(\tau - T)]d\tau = \int_{t-T}^{t} u_1(\tau)d\tau \tag{9}$$

Equation (9) states that the network total storage is equal to the integral of the input during the last round trip time *T*.

IV. MODELING THE TCP FLOW AND CONGESTION CONTROL

This section aims at showing that the closed loop control system depicted in Fig. 3 implements both the TCP flow and congestion control. In details, the following variables and blocks are shown:

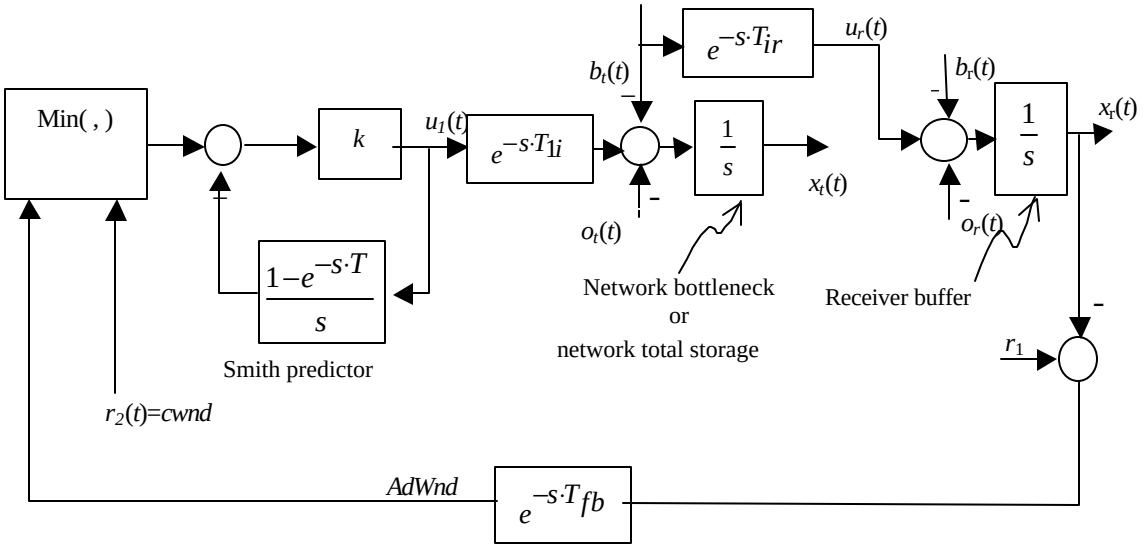

Fig. 3: Functional block diagram of the TCP flow and congestion control

(1) The receiver queue length x_r and the receiver capacity r_1 provide the term $r_1 - x_r$ (i.e. the *Advertised Window*), which reaches the sender after the propagation time T_{fb} that is modelled in the Laplace domain by the transfer function $e^{-sT_{fb}}$;

(2) The set point $r_2(t)$ represents a threshold for the total network storage, which is modeled by the queue $x_t(t)$;

(3) The minimum block takes the minimum between the *Advertised Window* and $r_2(t)$;

(4) Delays T_{1i} and T_{ir} model the time delay from the sender to the generic node i and from the node i to the receiver, respectively; the forward delay from the sender to the receiver is $T_{fw} = T_{1i} + T_{ir}$;

(5) The controller transfer function

$$G(s) = \frac{k}{1 + \frac{k}{s}(1 - e^{-sT})}, \qquad (10)$$

which contains the proportional gain k and the Smith predictor $(1 - e^{-sT})/s$, where T is the round trip time sum of the *forward* delay T_{fw} and the *backward* delay T_{fb}. Notice that 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).

Notice that the buffer x_t in Fig. 3 can model both the total network storage of packets but also it can model the generic buffer x_i that is the bottleneck of the TCP connection at time t; moreover, a moving bottleneck is easily captured by the model through delays T_{1i} and T_{ir} where i is the generic moving bottleneck.

In order to show that the block diagram in Fig. 3 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.

A. The TCP Flow Control

By assuming that the bottleneck is at the receiver, it results: $\min(Adwnd, r_2(t)) = Adwnd$, $u_r(t) = u_1(t - T_{fw})$ and $o_t(t) = 0$. In other words, the connection is constrained by the receiver, and the input rate reaches the receiver after the forward delay without network queuing, that is $b_t(t) = u_1(t - T_{1i})$. Under these conditions, Fig. 3 can be transformed into Fig. 4 that models the TCP flow control. The following propositions can be shown.

Proposition 1: The Smith controller (10) implements the TCP flow control equation (1).

Proof: To find the input rate $u_1(t)$ computed by the TCP sender we use standard Laplace techniques, that is, we compute the Laplace transform of the input rate:

$$U_1(s) = [R_1(s) - X_r(s)]e^{-sT_{fb}} \frac{k}{1 + k\left(\frac{1 - e^{-sT}}{s}\right)}$$

that can be written as

$$U_1 = -kU_1\left(\frac{1 - e^{-sT}}{s}\right) + k[R_1 - X_r]e^{-sT_{fb}}$$

By transforming back to time domain it results:

$$\frac{u_1(t)}{k} = r_1(t - T_{fb}) - x_r(t - T_{fb}) - \int_{t-T}^{t} u_1(t)dt \qquad (11)$$

By considering that

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

and that

$$\int_{t-T}^{T} u_1(t)dt = \text{Outstanding packets}$$

Equation (11) gives the classic window-based flow control equation (1), where $W = u_1(t)/k$. By considering that $u_1(t) = W/T$ relates the rate and the window of a window-based control, it results $1/k = T$.

Notice that the *outstanding packets* automatically take into account the round trip time T that in general can be time varying due queuing delays. In the case of flow control T is constant since it is assumed that there is no congestion inside the network which implies that network queuing delay is zero and round trip time is pure propagation delay.

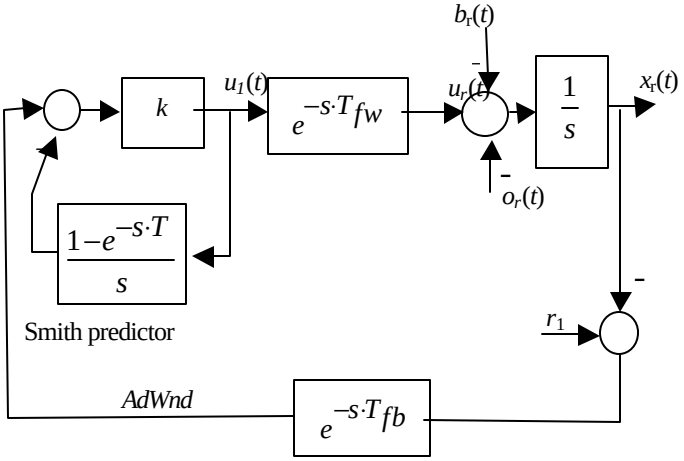

Fig. 4: Functional block diagram of the TCP flow control

Proposition 2: The TCP flow control equation (11) guarantees that the receiver queue is always bounded by the receiver capacity $\bar{r}_1$. *i.e.*

$$x_r(t) < \bar{r}_1 \text{ for any } t$$

Proof: The queue length can be computed by exploiting the superposition property of linear systems. In particular, it is easy to compute the input-output transfer function from $R_1(s)$ to the receiver queue length $X_r(s)$ that is:

$$\frac{X_r}{R_1} = \frac{k}{k+s} e^{-sT}$$

and the transfer function from $B_r(s)$ and $O_r(s)$ to $X_r(s)$ that is:

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

By assuming $r_1(t) = \bar{r}_1 \cdot 1(t)$, where $\bar{r}_1$ is the receiver buffer capacity and $1(t)$ is the step function[1] that models a connection starting at $t=0$, there results: $R_1(s) = \frac{\bar{r}_1}{s}$. By exploiting the superposition property of linear systems and by transforming back to time domain there results:

$$x_r(t) = L^{-1}\left\{\frac{\bar{r}_1}{s}\frac{k}{s+k}e^{-sT}\right\} - L^{-1}\left\{\frac{B_r + O_r}{s+k}e^{-sT}\right\} - \\ - \int_{t-T}^{t} [b_r(t) + o_r(t)]dt$$

which satisfies the condition

$$x_r(t) \le L^{-1}\left\{\frac{\bar{r}_1}{s}\frac{k}{s+k}e^{-sT}\right\} = \bar{r}_1 \cdot \left(1 - e^{-k(t-T)}\right) \cdot 1(t-T) < \bar{r}_1$$

since $o_r(t)$, $b_r(t)$ are always non negative. This concludes the proof.

Lemma 1: Proposition 2 guarantees $o_r(t)=0$ for any t.

Proof: Proposition 2 proves that the receiver queue length is always upper bounded by the receiver queue capacity, which implies that receiver overflow is always avoided, i.e. $o_r(t)=0$ for any t.

[1] The step function is defined as $1(t) = \begin{cases} 1 \; for \; t \ge 0 \\ 0 \; for \; t < 0 \end{cases}$.

B. The TCP Congestion Control

By assuming that bottleneck is localized inside the network, there results min(Adwnd,$r_2(t)$)= $r_2(t)$ and we can ignore the outer feedback loop. Therefore, Fig. 3 can be transformed into the equivalent one shown in Fig. 5, which models the TCP congestion control.

Proposition 3: The Smith controller (10) implements the TCP congestion control equation (2).

Proof*:* By assuming that the bottleneck is inside the network, there results: min(Adwnd,$r_2(t)$)= $r_2(t)$. From Fig. 5, the output of the Smith predictor in the Laplace domain is:

$$Q(s) = U_1(s)\frac{1 - e^{-sT}}{s}$$

By transforming back to time domain it results:

$$q(t) = \int_{t-T}^{T} u_1(t)dt \; = outstanding packets$$

Therefore the output of the controller is:

$$u_1(t) = k\left(r_2(t) - outstanding packets\right) \qquad (12)$$

that can be rewritten as

$$\frac{u_1(t)}{k} = r_2(t) - outstanding packet. \qquad (13)$$

Equation (13) gives the classic window-based congestion control equation (2), where $W = u_1(t)/k$, and $r_2(t)$=*cwnd*. This concludes the proof.

Remark 1: It should be noted that (12) and (13) are the rate-besed and window-based versions of the same control equation.

Proposition 4: The TCP congestion control equation (13) guarantees a total network storage x_t that is always bounded by the threshold $r_2(t) > 0$, i.e.:

$$x_t(t) \le r_2(t) \text{ for any } t$$

Proof: From (9), the total network storage is:

$$x_t(t) = \int_{t-T}^{t} u_1(t)dt = q(t)$$

Since $u_1(t)$ and $q(t)$ are always non negative, and $r_2(t)$ is strictly positive, from the control law:

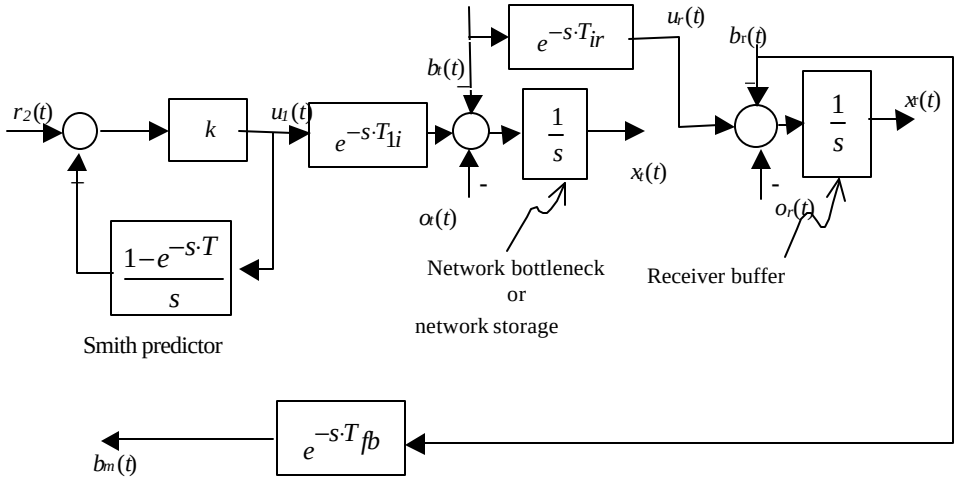

Fig. 5 Functional block diagram of the TCP congestion control

$$u_1(t) = k\left(r_2(t) - \int_{t-T}^{T} u_1(\tau)d\tau \right)$$

it turns out $r_2(t) \geq q(t) = x_t(t)$, which concludes the proof.

Lemma 2: If a TCP flow finds in each buffer it goes through a space of c_i packets, where $c_i > r_2(t)$ for any t and i, then the Proposition 4 guarantees $o_t(t)=0$ for any t.

Proof: From Proposition 4, which proves that $x_t(t) \leq r_2(t)$, and assumptions of Lemma 2, the proof follows directly.

V. MODELING RENO OR WESTWOOD TCP BY INPUT SHAPING

In this section we show that the dynamic model depicted in Fig. 5 is able to model successful variants of TCP congestion control, such as for example Tahoe/Reno [1] or the recent Westwood TCP [12]. Other TCP variants, such as Vegas or Santa Cruz, could also be modeled in the same unified framework.

We have seen that the congestion control algorithm aims at estimating 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 $cwnd \geq ssthresh$. Now we show that both these mechanisms can be modeled in the control theoretical framework reported in Fig. 5 by properly shaping the controller input $r_2(t)=cwnd$.

A. The Reno Algorithm

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

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

where the initial window r_0 is generally equal to 1 or 2 [19]. 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 modeled using the following settings for $r_2(t)$:

After a timeout at t_k

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

$$r_2(t) = r_0$$

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

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

After 3 DUPACKs at t_k

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

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

B The Westwood algorithm

TCP Westwood employs the same probing mechanism of Reno. It differs from Reno because of the behavior after congestion. In fact, Westwood sets the *cwnd* and *ssthresh* using an end-to-end estimate of the network bandwidth $b_m(t_k)$ available at time of congestion. In particular, the Westwood TCP window behavior after congestion can be modeled as follows:

After a timeout at t_k

$$ssthresh = b(t_k) \cdot RTT_{\min}$$

$$r_2(t) = r_0$$

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

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

After 3 DUPACKs at t_k

$$ssthresh = b(t_k) \cdot RTT_{\min}$$

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

VI. DESIGNING A TCP-FRIENDLY RATE CONTROL

The TCP congestion control is window-based. As a consequence, it sends packets in bursts. Burstiness degrades the performance of the control algorithm and makes it unsuitable for application such as audio and/or video where a relative smooth rate is of importance [18,28]. To overcome the mentioned problem, rate-based control algorithms have been proposed [24],[29]. A key requirement that a new rate-based congestion control algorithm must satisfy is friendliness towards TCP, i.e. it must share network bandwidth with TCP fairly. The idea of TFRC is to enforce and guarantee friendliness by using the TCP long-term throughput equation [30] to compute the input rate. This approach could reveal to be unfriendly since a TFRC sets instantaneously the rate that, in similar conditions, a TCP flow would reach only in long-term conditions [31,24].

This section sketches how the analysis developed in this paper can be used to design a TCP friendly rate-based congestion control.

For that purpose we start from the TCP congestion control equation in rate-based form, which is

$$u_1(t) = k(cwnd - outstandingpackets) \quad (14)$$

In order to propose a rate-based congestion control that is friendly to TCP, we propose the exact rate-based version of the TCP Reno congestion control by properly setting $cwnd=r_2$ in (14) to match the increasing/decreasing mechanism of Reno.

A Exponential probing corresponding to the slow-start phase

This phase aims at quick probing of network capacity and corresponds to Reno slow-start phase. It is obtained by setting the controller input $r_2(t)=cwnd$ in (14) as follows:

$$cwnd = r(t_0) \cdot 2^{\frac{t-t_0}{T}} \quad \text{while } cwnd < ssthresh$$

where t_0 is the last update time, T is the round trip time and $r(t_0)$ is equal to one or two [19].

B Linear probing corresponding to the congestion avoidance phase

This phase aims at gentle probing of network capacity and corresponds to Reno congestion avoidance phase. It is obtained by setting the controller input $r_2(t)=cwnd$ in (14) as follows:

$$cwnd = cwnd(t_0) + \frac{t-t_0}{T} \quad \text{when } cwnd > ssthresh$$

After a congestion episode (timeout or DUPACK) the controller input $r_2(t)=cwnd$ in (14) is set as follows

After a timeout a t_0

$$ssthresh = \frac{cwnd(t_k)}{2}$$

$$r_2(t) = r(t_0)$$

Enter the exponential probing

After 3 DUPACKs at t_k

$$ssthresh = \frac{cwnd(t_k)}{2}$$

$$cwnd = cwnd(t_k) + \frac{t-t_0}{T}$$

VII. CONCLUSIONS

This paper has developed a classical control theoretic approach to model the dynamic behavior of TCP congestion control. It has been shown that (1) a proportional controller plus a Smith predictor provides an exact model of the Internet sliding window 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. In order to show the utility of the proposed model, an application to the design of a TCP friendly rate control algorithm has been sketched.

VII. REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control," ACM Computer Communications Review, 18(4): 314 - 329, August 1988.

[2] S. Mascolo, "Congestion control in high-speed communication networks using the Smith principle", Automatica, vol. 35, no. 12, dec. 1999.

[3] M. Gerla and R. Locigno and S. Mascolo and R. Weng, "Generalized Window Advertising for TCP Congestion Control", European Transactions on Telecommunications, no. 6, Nov/Dec. 2002.

[4] M. Allman, V. Paxson, W. R. Stevens, "TCP congestion control," RFC 2581, April 1999.

[5] D. Clark, "The design philosophy of the DARPA Internet protocols," In Proceedings of Sigcomm' 88 in ACM Computer Communication Review, vol. 18, no. 4, pp. 106 - 114, 1988.

[6] Floyd, S.; Fall, K., "Promoting the use of end-to-end congestion control in the Internet", IEEE/ACM Transactions on Networking, Aug. 1999, vol.7, (no.4): 458-72.].

[7] L. L. Peterson, B. S. Davie, *Computer Networks*, Morgan Kaufmann, San Francisco, CA, 2000.

[8] Dah-Ming Chiu;Jain, R., "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN Systems, June 1989, vol.17, (no.1), p. 1-14.

[9] T.V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, 5(3), June 1997.

[10] S. Floyd, T. Henderson, " NewReno Modification to TCP's Fast Recovery", RFC 2582, April 1999.

[11] Brakmo L. S., O'Malley S. W., and Peterson L. L., "TCP Vegas: End-to-end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 13, no.8, pp. 1465-1480, 1995.

[12] 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 and Wireless Networks, vol. 8, no. 5, Sept. 2002.

[13] L.A. Grieco, S. Mascolo, "TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks", Proc. of the VII International Workshop on Protocols For High-Speed Networks (PfHSN'2002), April, 2002 Berlin, Germany. Lecture Notes on Computer Science (Lcns), Springer Verlag.

[14] C. Parsa, J .J. Garcia-Luna-Aceves, "Improving TCP Congestion Control over internets with heterogeneous Transmission media", Proc. IEEE Int. Conf. On Network protocols, Toronto, Oct. 31- Nov. 3,1999.

[15] F. P. Kelly, "Mathematical Modeling of the Internet," *Proc. 4th International Congress on Industrial and Applied Mathematics*, July 1999.

[16] V. Misra, W. Gong, D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED", *Proc. of Sigcomm2000* in *ACM Computer Communication Review*, vol. 30, no. 4, pp. 151-160, 2000.

[17] S. H. Low, "A duality model of TCP flow control", *Proc. of ITC Specialist Seminar on IP Traffic Measurements, Modeling and Management*, Sept. 2000.

[18] D. Bansal and H. Balakrishnan and S. Floyd and S. Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", *Proc. of Sigcomm 2001*.

[19] M. Allman, S. Floyd, C. Partridge, "Increasing initial TCP's initial window," RFC 2414, Sept. 1998.

[20] V. Jacobson, R. Braden, D. Borman, " TCP Extensions for High Performance ", RFC 1323, May 1992.

[21] Hoe, J., C., "Improving the Start-up Behavior of a Congestion Control Scheme for TCP," *Proc. of ACM Sigcomm'96*, pp. 270-280.

[22] Villamizar, C. and Song C. (1995), "High Performance TCP in ANSNET", *ACM Computer Communication Review*, vol. 24, no. 5, pp. 45-60.

[23] S. Mascolo, "Modeling and Stability Analysis of the Internet Congestion Control", Technical Report no. S17/03.

[24] L. A. Grieco, S. Mascolo, "Adaptive Rate Control for streaming flows over the Internet ", to appear on ACM Multimedia System Journal.

[25] Balakrishnan, H.; Padmanabhan, V.N.; Seshan, S.; Katz, R.H. A comparison of mechanisms for improving TCP performance over wireless links, IEEE/ACM Transactions on Networking, 5(6), (1997), 756-769.

[26] S. Floyd, "HighSpeed TCP for Large Congestion Windows", draft-ietf-tsvwg-highspeed-00.txt.

[27] C.V. Hollot, Vishal Misra, Don Towsley, Wei-Bo Gong, " On Designing Improved Controllers for AQM Routers Supporting TCP Flows", Proc. of Infocom 2001.

[28] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widme, “Equation-Based Congestion Control for Unicast Applications”, Proc. Sigcomm 2000.

[29] Handley, M., Floyd, S., Pahdye, J., and Widmer, J., “TCP Friendly Rate Control (TFRC): Protocol Specification.”, RFC 3448, Jan. 2003.

[30] J. Padhye, V. Firoiu, D. Towsley and J. Kurose", "Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proc. ACM Sigcomm 1998, pp. 303-314.