2001 European Control Conference (ECC)
Porto, Portugal, 4-7 September, 2001

INSIGHTS INTO TCP/IP CONGESTION CONTROL USING THE SMITH PRINCIPLE

Saverio Mascolo

Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari
Via Orabona 4, Bari, 70125 Italy
mascolo@poliba.it
Fax.: +39 080 5963410

Keywords: Congestion control, TCP/IP, Computer networks

Abstract

In this paper we propose classical control theory to look at the end-to-end congestion control algorithm used by the Transmission Control Protocol/Internet Protocol (TCP/IP). By using this approach we show that classical control modeling using Laplace transforms and Smith's principle give a simple and rigorous model of the role played by *congestion window* and *advertised window* variables in today's in TCP/IP. In particular, we clearly show that TCP performs flow and congestion control using a Smith's predictor. The proposed analysis gives clear guidelines to improve TCP in a way that is backward compatible and incrementally deployable. In particular, the model shows how it is possible to include in the control scheme, which has been obtained using a Smith predictor, any new type of explicit feedback from network nodes, any new algorithm executed at the source, as well as any new implicit feedback. Following this approach a modification to TCP called *faster recovery* is briefly sketched.

1. Introduction

After the Transfer Control Protocol/Internet protocol (TCP/IP) had become operational, the network was suffering from congestion collapse. TCP/IP congestion control was introduced into the Internet in the late 1980s and has been tremendously successful in preventing congestion collapse [1]. Many improvements have been introduced since that time and intense research activity is going on to improve the efficiency of this control mechanism [2], [4], [5], [6], [7], [8], [10], [11], [12].

Today's TCP has two feedback mechanisms to tackle congestion: the TCP *flow control* and the TCP *congestion control*. The TCP flow control avoids 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 avoids the flooding of the network and is based on implicit feedback such as time out, duplicate acknowledgments (DUPACKs), round trip time measurements. In this case the source infers the status of network congestion without using any explicit congestion notification from the network. In other words, TCP obeys the principle that the network is a "black box" that cannot supply any *explicit feedback* information to the source. Thus, the end-systems probe the network state by gradually increasing the window of packets that are outstanding in the network until the network becomes congested and drops packets. When a packet drop is detected, first the window is shrunk and then it starts to increase until a packet drop is again detected. The advantage is that network nodes are not in charge of performing congestion control. The drawback is that the congestion control mechanism is slow due to retransmissions and packet losses that are intentionally provoked in order to probe network capacity. This causes link under-utilization, when the congestion window is small, and packet loss when the congestion window is large. Moreover, increasing the congestion window until packets are lost fills the buffers and increases delays, which are harmful for delay sensitive applications.

In this paper we propose a rigorous control theoretic approach to derive a general model of the TCP/IP control algorithm. The derived framework is able to include, in a way that is incrementally deployable and backward compatible, any new explicit feedback that the network supplies to sources, any new implicit feedback measured by the source and any change to the algorithm executed at the TCP end nodes. Finally, a modification to TCP we called *faster recovery* is introduced.

2. Today TCP/IP flow and congestion control algorithm

A TCP connection is a virtual pipe between the send socket buffer and the receive socket buffer (see Fig. 1). Today's 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.

ISBN 978-3-9524173-6-2

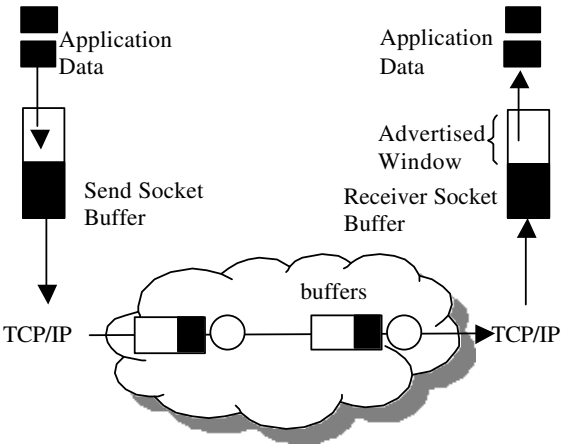

Fig. 1. Scheme of a TCP connection

2.1 The flow control mechanism

The TCP flow control avoids 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. 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 (*AdvWin*) of

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

which represents the amount of free space remaining in the receiver buffer. The TCP on the send side computes an *Effective Window W* that limits how much outstanding packets it can send

$$W = AdvWin - (LastByteSent - LastByteAcked) = AdvWin - OutstandingPackets \quad (1)$$

2.2 The congestion control mechanism

The TCP congestion control algorithm avoids the flooding of the network. Considering the fundamental end-to-end assumption of TCP/IP for which the network is a "black box" which does not supply any explicit feedback to the source, the fundamental issue here is to look for "some measurement" of the congestion status of the network which must be inferred at the end nodes using implicit feedback received from the networks such as timeout and acknowledgments. Today's TCP estimates the best effort capacity of the network using a variable called *Congestion Window* (*cwin*). The congestion window plays, in congestion control, the role of the advertised window in flow control.

Now, since TCP assumes no explicit feedback from the network, the problem is how TCP comes to learn an appropriate value of the *Congestion Window*. Van Jacobson's paper [1] defines an increase/decrease mechanism to throttle the size of the congestion window. The key idea is to probe the network capacity by increasing the load until packet loss is experienced. When a loss is experienced the window size is reduced. More precisely, first *cwin* is increased until packet losses are experienced and then is rapidly decreased. The increasing process goes through two phases: *slow start* and *congestion avoidance* that are shown in Fig. 2.

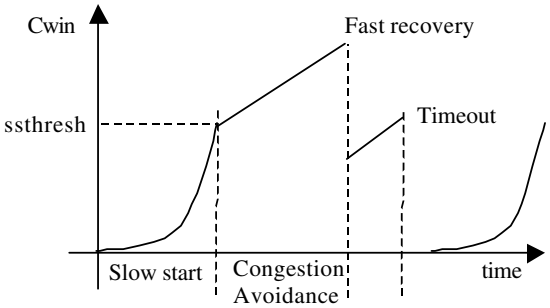

Fig. 2: TCP Reno Congestion Window dynamics

During the slow start phase *cwin* 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 *cwin* 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 is experience the window is drastically reduced to one and the slow start, congestion avoidance cycle repeats. This behavior is known as TCP Tahoe.

During the slow start phase the network can be strongly underutilized. To avoid this under utilization there were introduced two other mechanisms called *fast retransmit* and *fast recovery*. The implementation of these mechanisms forms what is currently called TCP Reno [13].

Fast retransmit and *fast recovery* are both triggered after that the sender receives three duplicate acknowledgments. The mechanism of *fast retransmission* sends again the packet that was acknowledged three times before it experiences timeout. The mechanism of *fast recovery* reduces the congestion window to half and enters into congestion avoidance. After a time out, the Congestion window is still reduced to one (see [13] for details).

The TCP sender computes the minimum of the congestion window and the advertised window and computes the *Effective Window W* as follows

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

Where the *effective window* corresponds to the amount of data that are injected into the network.

3. Modeling a TCP connection using transfer function and Smith's principle

A TCP connection is a virtual pipe between the send socket buffer and the receive socket buffer (see Fig. 1). The Sender Application can send arbitrary bytes into the Send Socket Buffer. TCP guarantees that the Receiver Application will receive all the data in the correct order. The objective of TCP control algorithm, which is executed at the sender side, is to avoid network congestion and overflow of buffer at the receiver side. In other words, bottleneck queue level must be less than queue capacity to avoid overflow and greater than zero to ensure full link utilization.

Today's 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 [2].

A block diagram of the system to be controlled is reported in Fig. 3. The system is a single-input/single-output (SISO) system. The input variable $u(t)$ is the TCP input rate (or equivalently the window[1]). The output variable $x(t)$, which must be controlled by throttling the input rate, is the bottleneck queue level. The buffer is modeled as an integrator, which is represented in the Laplace domain by 1/s. It integrates the difference between the input rate and the output rate $d(t)$. The output rate $d(t)$ models the best effort bandwidth that is available for the considered TCP connection. It is the bandwidth left over by the coexisting connections. The set point $r(t)$ represents a queue level threshold. The block diagram of the closed loop system is shown in Fig. 3.

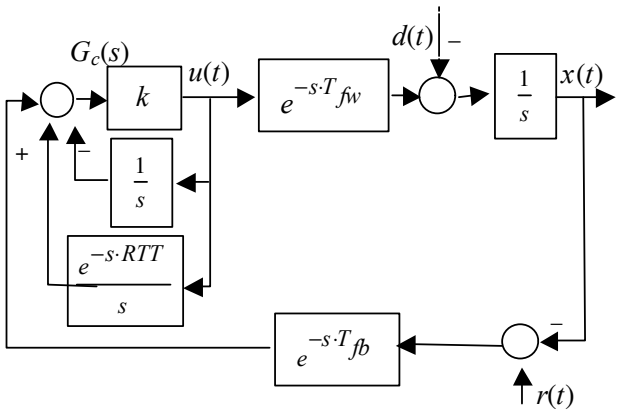

Fig. 3: Block diagram of the closed-loop controlled system

[1] The relation between the window W and the rate r is W= r*RTT, where RTT is the round trip time. This relation is quite intuitive. It could be derived using arguments from the theory of sampled control systems.

System in Fig. 3 contains a delay inside the feedback loop. This delay may affect the stability of the closed loop system. We propose to follow the Smith principle to design the control algorithm. Smith's principle is a classical control technique for time-delay systems. The principle suggests looking for a controller, which transforms the closed loop system into a delay free system. We decide to transform the system in Fig. 3 into the system in Fig. 4. The desired input-output dynamics reported in Fig. 4 has been carefully chosen so that:

a) The closed loop part of the desired system is delay free, that is, the delay is pushed out of the feedback loop and does not affect stability;
b) The desired system is a *simple first order system* with a delay in cascade, that is, the transfer function is $\frac{k}{s+k}e^{-T_{fw}\cdot s}$. The first order system is obtained by choosing a simple proportional controller k for the delay–free plant ($1/s$). Notice that this system is asymptotically stable for any $k>0$ and has no overshoots in response to a step like function threshold;
c) A controller $G(s)$ exists that renders the input-output dynamics of the system reported in Fig. 3 equal to the input-output dynamics of the target system reported in Fig. 4.

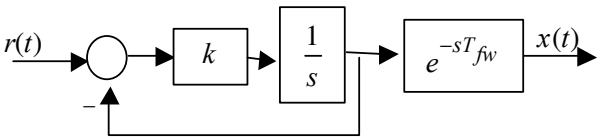

Fig. 4: Desired input-output dynamics

By equating the transfer functions of the systems reported in Fig. 3 and Fig. 4 the controller

$$G_c(s) = \frac{k}{1+\frac{k}{s}\left(1-e^{-RTT\cdot s}\right)} \tag{3}$$

is derived after a little algebra.

Looking at the controller (3) shown in Fig. 3, it is easy to write the rate control equation

$$u(t) = k\left(r(t-T_{fb}) - x(t-T_{fb}) - \int_{t-RTT}^{t} u(\tau)\cdot d\tau\right)$$

This equation can be intuitively interpreted as follows: the computed input rate is proportional, through the coefficient k, to the free space in the bottleneck queue, that is $r(t-T_{fb})-x(t-T_{fb})$, decreased by the number of cells released by the source during the last round trip time *RTT*. It is worth noting that the role of the Smith predictor is to take

into account the "in flight" cells given by the integral $\int_{t-RTT}^{t} u(\tau)\cdot d\tau$.

In [11] it has been shown, via mathematical analysis, that the proposed control law guarantees bottleneck queue stability and full link utilization during both transient and steady state condition in a store and forward packet switching network with general network topology and traffic scenario. It was assumed per-flow queuing and explicit feedback. In the next, we will remove this assumption to fit the end-to-end design principle of TCP.

3.1 TCP flow control

In this section, we show that Eq. 4, which has been obtained using a Smith predictor controller, can be transformed to the current TCP flow control equation. In fact, by setting the queue threshold $r(t)$ equal to the receiver buffer capacity it results that the available buffer is

$$r(t-T_{fb}) - x(t-T_{fb}) = AdvertisedWindow$$

Since by definition of *outstanding packets* it results that

$$\int_{t-RTT}^{t} u(\tau)\cdot d\tau = Outstanding\ Packets$$

Eq. 4 can be rewritten as

$$u(t)/k = u(t)\cdot\tau = W = AdvWin - OutstandingPackets \qquad (5)$$

where W is the amount of data sent in the interval time $\tau=1/k$, that is the *Effective window.* Eq. 5 is exactly the TCP flow control equation (4), that is, *current TCP flow control can be derived by controlling the receiver buffer using a Smith's predictor.* In this sense TCP flow control is optimal.

3.2 TCP congestion control

A TCP flow goes through a path of store and forward nodes. Letting $x_i(t)$ be the queue level of the generic *i-th* node along the path, the available buffer at the *i-th* node is $r(t) - x_i(t)$. In order to apply control equation (6) to TCP congestion control, the minimum available buffer (*AvBuf*) encountered along the path should be fed back to the TCP sender so that the congestion control equation becomes

$$W = \min_{i\in path}(AvBuf_i, AdvWin) - Outstanding Packets \qquad (6)$$

However, the fundamental end-to-end design principle of TCP makes impossible to implement Eq. 6 because network nodes cannot supply explicit feedback. Therefore, in order to apply control equation (4) it is necessary to use an end-to-end estimation of the minimum available buffer along the path. Thus, if we replace $\min_{i\in path}(AveBuf_i)$ in Eq. 6 with its end-to-end estimate and we call this estimate *congestion window*, we get:

$$W = \min(Cwin, AdvWin) - Outstanding Packets \qquad (7)$$

This is the current TCP congestion control equation (2).

Remark 1: Derivation of Eq. 7 using control theory and the Smith predictor shows that TCP already implements a Smith's predictor to avoid the overflow of the receiver's buffer and to avoid network congestion. This surprising result gives a theoretical insight into today's TCP congestion control algorithm and gives clear guidelines to the research activity aimed at improving TCP efficiency. In fact by looking at Eq. 7, it is clear that the research activity to improve TCP should focus on how to improve the end-to-end estimation of congestion window using implicit feedback and explicit feedback. Changes to Implicit Feedback imply only changes to the TCP stack protocol, whereas changes to the Explicit Feedback imply changes to network routers. It is also clear that the optimal solution is to perform per-flow buffering and to supply queue level as explicit feedback.

Remark 2: Any modification to Congestion Window and Advertised Window variable is completely backward compatible with current TCP because they fit into the general framework obtained using the Smith principle. As an example, we consider Differentiated Services (DS) that have been designed for scalability through handling aggregates of traffic instead of individual flows as in the Integrated Services. Stoica et al. [9] remark that Fair Queuing has many desirable properties for congestion control in the Internet. However, such mechanism needs to manage buffers and maintains state on a per flow basis and this complexity may prevent it from being cost-effectively implemented in routers. For these considerations, they propose an interesting architecture to approximately achieve fair bandwidth allocation. The approach distinguishes between edge routers, which maintain per flow state, and high-speed core routers, which do not maintain per flow state and use FIFO queuing. Core routers are in charge to compute a *Fair Share* of network capacity for each flow. It is easy to realize that this mechanism finds a perfect description in the general framework presented in this paper. In fact, the *Fair Share* is the fair network capacity assigned to each flow. Therefore, an edge router could set the Advertised Window as

$$AdvWin = \min(AdvWin, FairShare\times RTT)$$

In this way, mechanism described in [9] can be implemented in today TCP in a way that is incremental and backward compatible.

Remark 3: The interoperability of different congestion control mechanisms operated by different Internet Service Providers can be easily modeled using the framework described in this paper. In fact, the explicit feedback E_i supplied by a network N_i can be delivered to the TCP sender

by using the Advertised Window space in the TCP header, i. e. by setting

$$AdvWin = \min_{i \in N_i} (E_i, AdvWin)$$

where the *i-th* sub-network supplies the explicit feedback E_i and the minimum is over all the sub-networks the connection goes through.

Remark 4: Smith's predictor is very sensitive to system modeling errors. We have seen that the dynamic model of a TCP connection is an integrator with a time-delay in cascade. The only system parameter known with uncertainty is the time-delay. In fact, the round trip delay is the sum of two terms: the propagation delay, which is fixed and known, and the queuing delay that is stochastic and unknown. However, it should be noticed that the round trip time is continuously monitored and measured by the sender side of current TCP. In fact, round trip time is measured by time stamping packets when they are sent and by reading the time when the corresponding acknowledgments are received at the source. These measurements are continuously taken and averaged using an exponential filter. Measurements are used to fix the retransmission timeout in the Karn/Partridge algorithm or in the Jacobson/Karels algorithm [13]. For these considerations, we can conclude that, in the case of TCP, the Smith predictor does not suffer stability problems due to uncertain modeling of the plant since delays are continuously measured by the TCP source.

4 TCP with faster recovery

We have seen that congestion window dynamics should be designed in order to get a value that represents the best estimate of the network available bandwidth. Formally, we can describe the TCP Tahoe/Reno or Vegas Congestion Window dynamics using a discrete-event nonlinear equation:

$$Cwin(i+1) = f(ImplicitFeedback(i+1), Cwin(i)) \quad (8)$$

This is an estimator of the bandwidth available to the TCP connection.

This equation means that *cwin* at time t_{i+1} is a nonlinear function of the past value of *cwin* computed at time t_i, and of signals, representing implicit feedback from the network, received at t_{i+1}.

Implicit Feedback (IF_i) can represent any *variable*, such as round trip time measurements, which can be measured at the source, or any *event* that can be monitored at the source such as duplicate acknowledgments, timeouts, and duplicate acknowledgment. In other words, it results:

$$IF=g(timeouts, acknowledgments, DUPACKs, variables) \quad (9)$$

Thus, TCP Tahoe/Reno or Vegas can be modeled as an integrator plus a time-delay, which are controlled by a Smith's predictor, plus a nonlinear event driven estimator of the congestion window. Any new implicit or explicit feedback and any modification to TCP Congestion Window mechanism can be formally described by means of a new nonlinear function:

$$Cwin(i+1)=f_{new}(IF(i+1), EF(i+1), Cwin(i)) \quad (10)$$

Now we briefly sketch a new way to compute the Congestion Window. We propose to modify the way TCP Reno reduces the window after a timeout or three duplicate acknowledgments by using a new implicit feedback. This implicit feedback is an estimate of the available bandwidth obtained by measuring the rate of acknowledgments.

Notice that, when a timeout happens or three duplicate acknowledgments are received, the source is pumping an amount of data greater than available bandwidth. We also observe that in this condition the rate of acknowledgments corresponds just to the amount of data that have been delivered to the destination. Thus, under this condition, the rate of acknowledgments can be used to estimate the available bandwidth. The available bandwidth is used to set the congestion window equal to the *available bandwidth×RTT* product and enter the congestion avoidance phase, that is, the window threshold is set equal to the available *available bandwidth×RTT*. We call this modification *faster recovery.* A crucial feature is how to filter the noise contained in the measurement of the available bandwidth. This task can be accomplished by low-pass filtering the rate of acknowledgments. Low-pass filtering is also necessary to estimate the available bandwidth since congestion occurs whenever the low-frequency input traffic rate exceeds the link capacity [3].

Faster recovery is only one of the modifications that can fit in the general framework obtained by applying the Smith principle to TCP. Any new function used to calculate the Congestion Window fits in Eq. (7) and corresponds to a new version of TCP. Performance evaluation of faster recovery and its comparison with current TCP Reno is an ongoing research.

5. Conclusions

In this paper, we exploit classical control theory and Smith's principle to look at the flow and congestion control algorithms in the TCP/IP. By using this approach we derive a general framework which models the basic mechanisms of today's TCP/IP congestion control, illustrates what is optimal in current TCP and shows the guideline to improve it in a way that is backward compatible and incrementally deployable. In particular he framework shows how it is possible to include any new type of explicit feedback coming from network nodes, any new algorithm executed at the source as well as any new implicit feedback. On going research is evaluating the performance of a proposed TCP modification we called *TCP Westwood.*

Acknowledgments

This research was partially supported by "Progetto Giovani Ricercatori" del Politecnico di Bari.

References

[1] Jacobson, V.,"Congestion Avoidance and Control", *Proc. of Sigcomm88* in *ACM Computer Communication Review*, vol. 18, no. 4, pp. 314-329, 1988.

[2] 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.

[3] S. Q. Li and C. Hwang, "Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks", *IEEE/ACM Trans. Networking*, vol. 3, no. 1, Feb. 1995, pp. 10-25.

[4] Floyd, S., "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10-23, 1994.

[5] Floyd, S. and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp.397-413, 1993.

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

[7] Kalampoukas, L. and Varma, A., "Explicit Window Adapatation: A Method to Enhance TCP Performance", *IEEE Proc. Infocom 98,* vol. 1, pp. 242-251, 1998.

[8] Keshav S., "A Control-Theoretic Approach to Flow Control", *Proc. ACM Sigcomm91* in *ACM Computer Communication Review,* vol. 21, no. 4, pp. 3-15, 1991.

[9] Stoica I., Shenker S. and Zhang H., "Core-stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocation in High Speed Networks", *Proc. of Sigcomm98* in *ACM Computer Communication Review,* vol. 28, no. 4, pp. 118-130, 1998.

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

[11] S. Mascolo, "Congestion Control in high-speed communication networks using the Smith principle", Automatica, Special Issue on Control Methods for Communication Networks, Dec. 1999.

[12] Allman, M., Paxson, V., and Stevens, W., "TCP Congestion Control", RFC 2581, April 1999.

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