An Introduction to Modelling and Performance Evaluation for TCP Networks

Michele Pagano[1] and Raffaello Secchi[2]

[1] Dipartimento di Ingegneria dell'Informazione
Università di Pisa, Via Caruso, I-56122 Pisa, Italy
michele.pagano@iet.unipi.it
[2] Electronic Research Group
University of Aberdeen, Aberdeen AB24 3UE, United Kingdom
raffaello.secchi@erg.abdn.ac.uk

Abstract. The widespread diffusion of TCP over the Internet has motivated a significant number of analytical studies in TCP modelling, the most important of which are presented in this tutorial. The simplest approaches describe the dynamics of an individual source over a simplified network model (e.g. expressing the network behaviour in terms of average loss rate and latency). These models allow us to derive accurate estimations for the long-term TCP throughput under different network settings. More detailed techniques model the behaviour of a set of TCP connections over an arbitrary complex network. The latter are able to capture the network dynamics and effectively predict the closed-loop interaction between TCP and traffic management techniques. As an example the derivation of sufficient stability conditions for a network of RED queues is provided.

Keywords: TCP modelling, fluid models, stability conditions.

1 Introduction

The transmission control protocol (TCP) is used by the vast majority of Internet applications for reliable and ordered data transmission [1]. Since TCP is in charge of network congestion control and bandwidth fair sharing for these flows, it is in fact responsible for the stability of the entire Internet. Hence in the last decade, accurate modelling of its behaviour has attracted the interest of many researchers and a large amount of literature has been produced on this topic. This tutorial is a primer to the vast world of TCP modelling, which provides a first look to the most important contributions in this field. It is not intended to be comprehensive, but the interested reader may find here enough material to approach more advanced resources.

Accurate TCP models have been developed in the past with various purposes: lightweight simulation techniques, network dimensioning, systems requirement definition, design of new TCP-friendly protocols, etc. Simple mathematical models are today sometimes preferred to detailed network simulations to have a quick evaluation of capacity requirements and performance of TCP based services. As noted in [2], detailed computer simulations may be not able to scale to large network topologies or large number of concurrent flows. This rises concerns on what could be the consequences of the

D. Kouvatsos (Ed.): Next Generation Internet, LNCS 5233, pp. 571–593, 2011.

deployment of such a system over the Internet. By describing only the essential characteristics of congestion control, analytical models permit to produce accurate results for realistic scenarios in a reasonable short time.

TCP models proved useful also to design new standards for congestion control compatible with existing TCP implementations. As an example, the *throughput equation* of TCP has been used as the basis for the design of a congestion control algorithm for slowly variable flows [3], such as multimedia streaming. The analysis of other congestion control mechanisms at network layer, such as the active queue management (AQM) [4,5,6,7], has been successfully carried out by means of TCP models. As explained further in the tutorial, a closed control loop is formed between AQM and TCP, whose behaviour is well explained by analytical models.

TCP models find an application also in the solution of some network optimisation problems [8]. These techniques aim to assign flow rates to a set of flow sharing a common network in order to satisfy some optimisation criteria. These techniques are based on the definition of an *utility function* that describes the degree of satisfaction associated to a given flow rate. An optimisation algorithm (possibly distributed) is used to determine the solution that maximises the overall network utility. This idea, first proposed by Kelly in [9], has been elaborated upon by many researchers [8,10,11,12] and has been very useful for understanding the bandwidth sharing properties of TCP. For instance, Low identified [10] the utility function that well describe TCP Reno, TCP Reno with RED and TCP Vegas, while Massoulié et al. in [11] catalogued various utility functions that allow to achieve fairness, minimise delay and maximise throughput.

As far as the classification of TCP models is concerned, reference [13] distinguishes between TCP models for bulk file transfer and for short lived connections. In the first case, when the TCP steady state behaviour is analysed, accurate throughput formulas are obtained by simplifying some aspects of protocol dynamics, such as the connection start-up and the loss recovery phase, that have minor impact on the long term performance. On the other hand, models for short-lived connections, such as [14,15,16], focus on the start-up phase taking into account the connection establishment and the *Slow-Start* phase.

Here, we focus on TCP models for long lived connections that we divide into two families: *Single connection models* and *fluid models*. Although the TCP algorithms have evolved significantly over the years, the original mechanism based on additive increase multiplicative decrease (AIMD) paradigm has been preserved. Thus, capturing the essence of AIMD leads to a good approximation of TCP behaviour. Single connection models assume a network path with given characteristics, such as the mean round trip time (RTT) and the loss probability, and try to evaluate throughput as a function of these parameters [17,18,19,20,21,22,23,24,25]. On the other hand, fluid models [26,27,28,29] simplify the high complexity of the TCP algorithms using a fluid flow analogy. The discrete nature of packet transmission is neglected in favour of a representation of TCP dynamics using compact differential equations. In some case, fluid models are built introducing a sub-model of TCP protocol and a sub-model of TCP flow interaction over the network, and then iterating over the two sub-models by means of a fixed point procedure [26,27,30,31,32].

As a final consideration, we can say that TCP models have contributed significantly to the understanding of the nature of the Internet traffic. This approach is substantially different from the classical one, where traffic characterisation was achieved by fitting the corresponding time series to well-known statistical distributions [33,34]. Embedding a model of the source (TCP) into the mathematical framework allows us to explain the causes of the observed behaviour and potentially uncover better mechanisms.

The rest of the tutorial is organised as follows. In Section 2, we present models of individual connections and we derive throughput formulas for various operating conditions. In Section 3, the focus is on fluid models. The interaction between TCP and network layer for various congestion control settings is analysed. Finally, Section 4 ends the tutorial with the conclusions.

2 Single Connection Models

In this section we formulate a model for a single TCP connection with the intent of deriving a formula of TCP throughput. Later in the section, we will consider a model of wide-area traffic and analyse its implications. We start from a simple yet not accurate model and we proceed by a successive step of refinement.

Let us consider a TCP connection established over a lossy path [20] with large enough capacity and competing traffic so small that the queueing delay component of round trip time (RTT) can be assumed negligible. Let us also assume that the link introduces one drop every *1/p* successful packet deliveries. Under these fairly strong assumptions, it is easy to obtain a closed-form formula for the TCP throughput. Indeed, after an initial transient, the TCP congestion window (*cwnd*), which corresponds to the number of in flight packets over the link, takes the periodic evolution illustrated in Figure 1. According to the AIMD principle, when the *cwnd* reaches its maximum (W) and a packet loss is met, the *cwnd* is backed off to $W/2$.

The speed of growth of the *cwnd* (the slope of the curve) depends on the way the receiver is acknowledging packets. Standard implementations require TCP receivers to

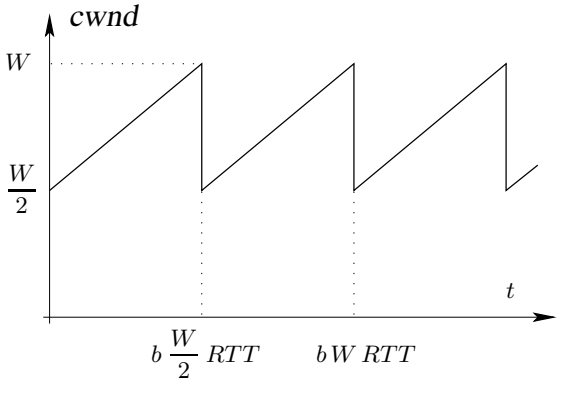

Fig. 1. Periodic evolution of TCP window

acknowledge the reception of every packet. However, some implementations adopt the delayed-ACK option [35], which consists of transmitting one cumulative ACK every two packets received. This roughly halves the number of ACKs transmitted, but it also slows down the growth of the *cwnd* during the Congestion Avoidance phase. Then, the duration of a cycle is $bW/2$ rounds, where b takes the value 2 or 1 depending on the delayed-ACK being enabled or disabled respectively. The duration of a cycle results

$$T_{\text{cycle}} = RTT \cdot b\frac{W}{2} \tag{1}$$

The total number of segments delivered within each cycle (A_{cycle}) is equivalent to the area under a period of the sawtooth, which is

$$A_{\text{cycle}} = b\frac{W}{4} \cdot \left(\frac{W}{2} + W\right) = b\frac{3W^2}{8}$$

packets per cycle. Since, by hypothesis, the number of packets in a cycle is $1/p$, we can solve for W obtaining

$$W = \sqrt{\frac{8}{3pb}} \tag{2}$$

The throughput achieved by the connection is the ratio between the total amount of data delivered in a cycle A_{cycle} and the duration of the cycle T_{cycle}. Substituting in the equation below, we get the mean throughput $\mathcal{B}$ in packet per second as:

$$\mathcal{B} = \frac{A_{\text{cycle}}}{T_{\text{cycle}}} = \frac{b\frac{3}{8}W^2}{RTT \cdot \frac{b}{2}W} = \sqrt{\frac{3}{2b}} \cdot \frac{1}{RTT\sqrt{p}} \tag{3}$$

This expression shows that the throughput is inversely proportional to RTT and the square root of loss probability. The constant of proportionality, which in this formula is $\sqrt{3/2b}$, is in general a function of TCP implementation and loss model. Slightly different values of this constant were derived in [19,21] or extrapolated from empirical data.

2.1 Stochastic Models of TCP Throughput

The TCP throughput formula presented above tends to overestimate the throughput for large loss rates, which is mainly due to not taking into account TCP retransmission timeouts. Indeed, timeouts occur when the number of packet drops in a window of data is such that the Fast Recovery algorithm [36] cannot detect the missing packet from duplicated ACKs. A refinement of the previous formulation which accounts for timeouts was given in [18], which is in part presented in the following.

Congestion Avoidance is described here in terms of rounds, each starting with the back-to-back transmission of a burst of packets equal to the congestion window and ending with the reception of the first ACK for the burst of delivered packets.

Since at each ACK reception the *cwnd* is incremented by $1/\lfloor cwnd \rfloor$ packets and an ACK acknowledges exactly b packets, at the beginning of the transmission of the next

packet batch the value of *cwnd* is *cwnd* $+ 1/b$. That is, during Congestion Avoidance and in absence of losses, the window size increases linearly with slope $1/b$. To simplify the analysis, the duration of each round is supposed independent of the congestion window.

The TCP algorithm stops increasing *cwnd* and slows-down transmission when it detects a packet loss, because TCP associates packet drops with congestion indications. In order to emulate the loss behaviour induced by the drop-tail queueing discipline, which is quite common for Internet routers, we assume that the losses of packets within the same round are correlated. On the other hand, we can assume that packet losses in different rounds are independent. This is justified by the fact that packets in different rounds are separated by one RTT or more, and thus these are likely to encounter independent buffer states. Simulative studies [37] confirm this claim highlighting that packet drops tend to be grouped in bursts when the (drop tail) buffer is congested. Hence, we suppose that each packet may be dropped with probability p if the previous packet is not lost, and that all the packets following the first loss in a round are also lost.

Loss detection at the sender side can occur through either the reception of three duplicate ACKs or when a timeout expires. In the first case, after lost segments are retransmitted, the sender resumes the transmission with the *cwnd* set to one half of its previous value. On the contrary, when the loss is detected via a timeout T_0 expiration, the *cwnd* is reset to one and a packet is sent in the first round following the timeout. In case another timeout expires, the length of timeout is doubled. Doubling the timeout is repeated for each unsuccessful retransmission up to $64T_0$, after which the timeout remains constant. If L_k indicate the duration of a sequence of k consecutive timeouts, that is

$$L_k = \begin{cases} (2^k - 1)T_0 & \text{for } k \leq 6 \\ (63 + 64(k-6))T_0 & \text{for } k \geq 7 \end{cases}$$

the mean length of timeouts can be easily calculated recalling that L_k has a geometric distribution due to the packet loss independence between consecutive rounds

$$\begin{aligned} T_{\text{timeout}} &= \sum_{k=1}^{\infty} L_k \cdot p^{k-1}(1-p) \\ &= T_0 \frac{1 + p + 2p^2 + 4p^3 + 16p^5 + 32p^6}{1-p} \end{aligned}$$

which means that the timeout period can be expressed in the form $T_{\text{timeout}} = T_0 \cdot f(p)$.

Figure 2 illustrates an example of the evolution of *cwnd*. The $i-$th TCP cycle consists of a sequence of n_i periods in which the *cwnd* has an AIMD behaviour (*cwnd-cycles*) and a sequence of one of more timeout periods. The first $n_i - 1$ *cwnd-cycles* end with a packet loss detection through three duplicate ACKs, while a timeout occurs in the last one. The throughput of a connection can be calculated as the ratio between the mean number of packets delivered within the cycle and the mean length of the cycle. If we assume that $\{n_i\}_i$ is a sequence of independent and identically distributed random variables with mean n, which are independent of the amount of data delivered in each *cwnd-cycle* and of the duration of the period, we can write the throughput as

$$\mathcal{B}' = \frac{n \cdot A_{\text{cycle}}}{n \cdot T_{\text{cycle}} + T_{\text{timeout}}} = \frac{A_{\text{cycle}}}{T_{\text{cycle}} + Q \cdot T_{\text{timeout}}} \tag{4}$$

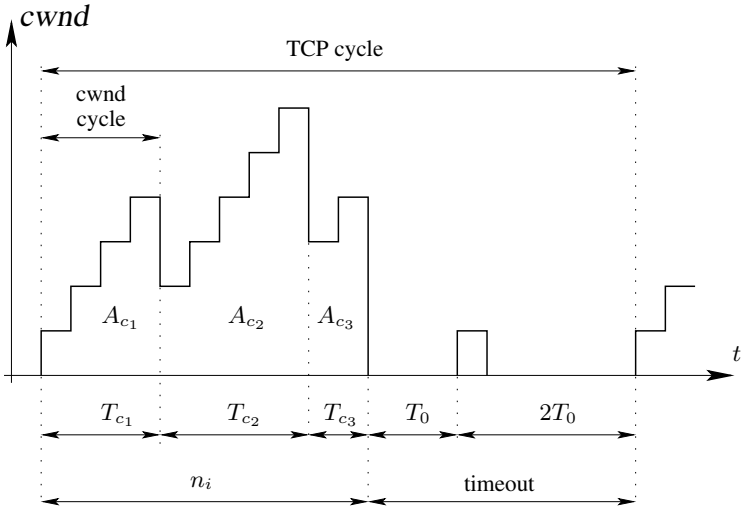

Fig. 2. TCP windowsize evolution

where $Q = 1/n$ denotes the probability that a loss indication is detected via a timeout. In order to complete the formula, we have to derive some closed-form expressions for A_{cycle}, T_{cycle} and Q. The T_{cycle} is given by (1), where W is substituted by the mean value of the steady-state window size, which is formally identical to equation (2) for small values of p:

$$E[W] = \sqrt{\frac{8}{3pb}} + o(1/\sqrt{p}) \tag{5}$$

A formal proof of this can be found in [18].

We concentrate now in deriving an expression for Q. Let us consider a *cwnd-cycle* where a loss indication occurs and let w and k represent respectively the *cwnd* and the number of packets successfully delivered in a round. According to our loss model, if a packet is lost, so are all the packets that follow up to the end of the burst. Then, the probability $A(w, k)$ to have $k < w$ successful transmissions in this round is given by

$$A(w, k) = \frac{(1-p)^k p}{1-(1-p)^w}.$$

Also, since the first k packets in the round are acknowledged, other k packets are delivered in the next round, which is the last round of this *cwnd-cycle*. Again, this round of transmission may have losses and the lost packets are the last transmitted in the burst. Indicating with m the number of packets successfully transmitted in the last round, the distribution of m can be expressed by

$$C(w, m) = \begin{cases} p(1-p)^m & \text{for } m \leq w-1 \\ (1-p)^w & \text{for } m = w \end{cases}$$

For the duplicated ACKs in the last round of transmission, the TCP receiver disables the Delayed-ACK mechanism [38] and every packet following the lost packets is acknowledged. If the sender receives less than three duplicate ACKs, the *cwnd-cycle* ends with a timeout and the probability of this event is given by

$$\hat{Q}(w) = \begin{cases} 1 & \text{if } w \leq 3 \\ \sum_{k=0}^{2} A(w,k) + \sum_{k=3}^{w} \sum_{m=0}^{2} A(w,k)C(w,k) & \text{otherwise} \end{cases} \tag{6}$$

since, when a timeout occurs, either the number of packets transmitted in the second last round or the number of packets successfully delivered in the last round is less than three. Simple algebraic manipulations allow us to rewrite this expression as

$$\hat{Q}(w) = \min\left(1,\ \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right).$$

Observing that $\hat{Q}(w) \approx 3/w$ when p tends to zero, we can get a good numerical approximation for $\hat{Q}$

$$\hat{Q}(w) \approx \min\left(1,\ \frac{3}{w}\right).$$

Finally, we can write the probability that a loss indication is given by a timeout and approximate it by

$$Q = \sum_{w=1}^{\infty} \hat{Q}(w) \cdot Pr\{W = w\} = E\{\hat{Q}\} \approx \hat{Q}(E[W]) \tag{7}$$

where $E[W]$ can be evaluated using expression (5). Now, let us concentrate on evaluating A_{cycle} by considering the $i-$th *cwnd-cycle*. We indicate with α_i the first packet lost during the $i-$th *cwnd-cycle*. As previously observed, after packet α_i, $W_i - 1$ more packets are sent in an additional round before the loss can be detected by the sender. Therefore, a total of $A_i = \alpha_i + W_i - 1$ packets are sent during the whole $i-$th cycle. It follows that

$$A_{\text{cycle}} = E[\alpha] + E[W] - 1$$

Based on our assumption on packet losses, the random process $\{\alpha_i\}_i$ is a sequence of independent and identically distributed random variables with distribution

$$Pr[\alpha = k] = p(1-p)^{k-1} \qquad k = 1, 2, \ldots$$

since the probability that $\alpha_i = k$ is equal to the probability that exactly $k-1$ packets are successfully acknowledged before a loss occurs. Thus, replacing the mean of α with $1/p$ we have

$$A_{\text{cycle}} = \frac{1-p}{p} + E[W] \tag{8}$$

where $E[W]$ is given by (5). Then, substituting (8), (1) and (7) into (4), we achieve an accurate estimate of the throughput of a TCP connection; after further simplifications, in [18] the following well known expression for TCP bandwidth (9) is given:

$$\mathcal{B}' = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1,\ 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)} \tag{9}$$

where the approximation $f(p) \simeq 1 + 32p^2$ in the expression of T_{timeout} was applied.

It is worth noting that (9) holds as long as the throughput is less than $W_{\max}/RTT$, where $W_{\max}$ is the maximum buffer advertised by receiver, since more than $W_{\max}$ packets cannot be transferred every round trip. We observe that this model does not capture all the aspects of Fast Retransmit algorithm and disregard the Slow Start phase. However, the impact of these omissions is quite low and the measurements collected to validate the model indirectly validate the assumptions as well. Live experiments also suggest that this model is able to predict the throughput for a wider range of loss rates than (3). Indeed, real experiments carried out over narrow-band links show that, when the congestion window size is small, many TCP timeouts occur. This indicates that in reality Fast Retransmit does not detect all the loss events and an accurate formula for TCP throughput has necessarily to take into account TCP timeouts.

3 Fluid Models

Fluid modelling is a widely used technique in the analysis of network systems where packet flows are approximated by continuous (i.e. fluid) streams of data. The main advantage is the formulation of a system of equations to describe the behaviour of congestion-controlled sources, network elements, and flows interaction over the network. Experimental results [26] show that relatively simple systems of ordinary differential equations (ODEs) provide satisfactory predictions of the real system dynamics. Since performance evaluation results can be achieved with relatively low cost, both in terms of computational time (ODEs can be fast solved using numerical methods) and development effort, these techniques are often used during the first stage of protocol design. In fact, fluid modelling techniques are rugged enough to establish the effectiveness of a protocol and resolve dimensioning issues. In this section, we introduce various techniques for fluid modelling of TCP networks that recur in many research studies.

3.1 Throughput Formulas for WANs

The analysis that we are about to present was carried out in [21] to express the TCP throughput as a function of the bandwidth-delay product (BDP). The BDP is a very important and widely used metric influencing TCP performance. It corresponds, under ideal conditions, to the amount of data that a bulk TCP transfer should hold in flight to achieve high utilisation of link bandwidth. Thus, studying the TCP throughput as a function of the BDP is a natural way of expressing TCP performance. As several authors pointed out [20,21], the TCP throughput is so correlated to the BDP, that a

direct relationship between the two metrics can be found. In the following we show an analytical way to achieve this goal.

It has been observed that the BDP is only relevant for wide area connections where the RTT may range from several tens to hundreds of milliseconds. In local area networks (LANs) the number of packets in flight is very small and the TCP throughput is limited by other factors, such as the loss rate induced by the medium access control scheme or the transmitter buffer size. Moreover, for long-haul connections the *cwnd* tends to open considerably and TCP performance are extremely sensitive to occasional packet losses due to transient congestion or packet corruptions. As it will be clarified in the following, TCP could not be able to *keep the pipe full*, i.e to achieve high capacity utilisation, in these cases.

Let us indicate with c the capacity of the link in packets per second, τ the round-trip propagation delay, and $T = \tau + 1/c$ the minimum observed RTT. For a wide area network (WAN) connection, the BDP, $c \cdot T$, is comparable in magnitude with the amount of packets queued at the bottleneck route.

Let us also assume that at a given time the bottleneck buffer is not empty. The packets are forwarded at rate c by the link server, ACKs are generated by the destination at rate c and, therefore, new packets can be released by the source every $1/c$ seconds[1]. Thus, the maximum possible number of unacknowledged packets is the sum of the packets in transit across the path, which is equal to cT, and of the packets in the buffer of size B. Therefore, if the size of the window exceeds $W_{\max} = cT + B$, a buffer overflow occurs. Actually, when the packet loss occurs, it is difficult to evaluate the exact *cwnd*, since it depends on the link capacity and on the RTT. However, as an approximation, we assume that TCP Reno, after retransmitting a lost packet, resumes Congestion Avoidance with the *cwnd* set to $(cT + B)/2$.

Using the fluid approximation, the *cwnd* dynamics can be easily written as a differential equation. Denoting with $a(t)$ the number of ACKs received by the source after t seconds spent in the Congestion Avoidance phase, the derivative of *cwnd* can be written as

$$\frac{dW}{dt} = \frac{dW}{da} \cdot \frac{da}{dt}.$$

If the *cwnd* is large enough, the bottleneck buffer is continuously backlogged, and the ACK rate is c. Otherwise, the ACK rate equals the sending rate W/T. In other words we have

$$\frac{da}{dt} = \min\{\frac{W}{T}, c\},$$

and recalling that during the Congestion Avoidance phase the *cwnd* is increased by $1/W$ for each ACK received, we have

$$\frac{dW}{da} = \frac{1}{W}$$

[1] For sake of simplicity we assume here and in the following that delayed-ACK is not implemented.

that is

$$\frac{dW}{dt} = \begin{cases} \frac{1}{T} & \text{if } W \leq cT \\ \frac{c}{W} & \text{if } W > cT \end{cases} \tag{10}$$

which means that the Congestion Avoidance phase consists of two sub-phases corresponding to $W \leq cT$ and to $W > cT$ respectively. We are now able to evaluate the duration of the first phase as

$$T_1 = T \cdot (cT - \tfrac{1}{2}W_{\max}) \tag{11}$$

and the number of packets successfully transmitted in this phase

$$\begin{aligned} N_1 &= \int_0^{T_1} \frac{W(t)}{T}\, dt = \frac{1}{T}\int_0^{T_1} \left(\frac{1}{2}W_{\max} + \frac{t}{T}\right) dt \\ &= \frac{1}{2T}\left(W_{\max}T_1 + \frac{T_1^2}{T}\right) \end{aligned} \tag{12}$$

When $W > cT$, the queue is increasing, the RTT as well increases and the *cwnd* opens more and more slowly. By integrating (10), in the second phase we get

$$W(t)^2 = 2c(t - T_1) + (cT)^2$$

and evaluating this expression for $t = T_1 + T_2$, we obtain the duration of the phase

$$T_2 = \frac{W_{\max}^2 - (cT)^2}{2c} \tag{13}$$

and the number of packets delivered

$$N_2 = c \cdot T_2, \tag{14}$$

since the link is fully utilised during this phase. Now, we can evaluate the throughput as a function of the ratio between the bottleneck buffer and the BDP

$$\mathcal{B}'' = \frac{N_1 + N_2}{T_1 + T_2} = \frac{3c}{4} \cdot \frac{(1 + \frac{B}{cT})^2}{1 + \frac{B}{cT} + (\frac{B}{cT})^2}. \tag{15}$$

This expression, which holds for $\frac{B}{cT} \leq 1$, suggests that the performance of a bulk transfer over a WAN might be negatively affected by small (with respect to the BDP) bottleneck buffers. In order to fully exploit the link capacity, the buffer size should be as close as possible to the BDP, which is a rule often used to configure router buffers [39].

3.2 Throughput Formulas for WANs and Random Losses

Let us now consider the case where, in addition to buffer overflows, packets can be randomly lost over the bottleneck link with probability q. In such a case, it is possible to exactly compute the throughput through Markov chain analysis. In the following we sketch this method.

In absence of random losses, the evolution of *cwnd* of TCP Reno is entirely determined by the window size at the beginning of the cycle w, which is half of *cwnd* at the end of the previous cycle. Then, the evolution of *cwnd* can be described introducing the following functions:

- $W(n, w)$ the window size after n successful packet transmissions.
- $T(n, w)$ the time required to complete the transmission of n packets.

If the cycle terminates with losses due to buffer overflow, then $N_{\max}(w)$ represents the number of packets transmitted in this period; otherwise, the cycle terminate with a random loss after successfully transmitting $N \leq N_{\max}(w)$ packets. According to the random loss model, the distribution of N is given by

$$Pr\{N = n\} = \begin{cases} q(1-q)^n, & \text{for } n < N_{\max} \\ (1-q)^{N_{\max}}, & \text{for } n = N_{\max} \end{cases} \tag{16}$$

For the $i-$th cycle, let w_i, N_i and T_i denote respectively the *cwnd* at the beginning of the Congestion Avoidance phase, the number of successful transmissions in the cycle, and the duration of the cycle. The TCP evolution can be expressed through the following recursive equations

$$\begin{cases} w_{i+1} = \frac{1}{2} W(N_i, w_i) \\ \\ T_{i+1} = T_i(N_i, w_i) \end{cases} \tag{17}$$

These equations, together with (16), define the transition probabilities for the continuous-time Markov chain $\{ w_i \}$, whose solution gives the stationary distribution of w_i. The steady-state throughput is then given by

$$\mathcal{B}''' = \frac{E[N_i]}{E[T_i]}$$

Since the exact solution of this Markov chain can be computationally expensive, an approximation of the previous method is also provided in [21].

An important result of the cited study is that the presence of random losses leads to significant throughput deterioration for large BDPs. In particular, it has been shown that the TCP throughput depends on the product of the loss probability and of the *square* of the BDP ($q \cdot (cT)^2$) and degrades rapidly when this parameter grows larger than one. This is due to the relatively earlier drops in the *cwnd-cycle* that lead to small initial values of the *cwnd*, thus requiring several transmission rounds to *fill the pipe* again. In other words, TCP over WAN networks suffers performance degradation in presence of non-congestion losses, such as the ones induced by competing real-time traffic or wireless links. Countermeasures, such as traffic differentiation or performance enhancing proxies (PEPs), are usually employed to eliminate this problem.

3.3 Interaction between TCP and AQM Mechanisms

So far, we presented various ways to relate TCP performance to network parameters. This was possible assuming a sort of independence between individual TCP flow state and network state. However, this assumption does not hold in many circumstances where the number of interacting flows is low or the network topology is not trivial.

We introduce now a different approach, still based on fluid modelling, where stochastic differential equations are uses to model TCP behaviour as well as network/flow interactions. This allows us to assess the benefits of the active queue management (AQM), and to select network parameters that enhance stability and performance.

As an example, we consider the random early detection (RED) technique [40], which is a highly popular AQM scheme within the Internet community. RED improves network performance reacting *earlier* (otherwise said pro-actively) to congestion. Similarly to other AQM schemes, RED randomly discards incoming packets with a probability that depends on the averaged queue size before reaching the full-buffer condition. This avoids the overstaying of congested situations in router buffers, which leads to a lower amount of congestion losses. Randomly selecting the packets to discard, RED also enables a flow management fairer than the drop-tail discipline. Indeed, the larger is the flow rate, the higher is the packet dropping probability, and the more frequent is the delivery of congestion signals. Eventually, this leads to a faster convergence to an equilibrium condition where fairness among flows is achieved.

Most of the IP routers today support the explicit congestion notification (ECN) option. This technique consists of marking a flag into the IP header instead of dropping the packet. The status of the flag is then copied into the returning ACK by the receiver to explicitly notify the sender the presence of congestion on the direct path. From a modelling point of view, the ECN marking scheme can be assimilated to the RED analysis, as the amount of packet losses is negligible with respect to the number of transmitted packets. Several enhancements of RED [5,41,42,43] have been proposed. These techniques consider more complicated packet dropping strategies to improve congestion control. Nevertheless, once the control law is known, fluid modelling techniques can be easily implemented to analyse the effects of these modifications. Figure 3 shows the classic dropping profile, which is used in the following analysis. It is a linear function between a lower $t_{\min}$ and an upper $t_{\max}$ threshold with a discontinuity at $t_{\max}$, which is analytically expressed by

$$p(x) = \begin{cases} 0 & 0 \le x < t_{\min} \\ \dfrac{x - t_{\min}}{t_{\max} - t_{\min}} p_{\max} & t_{\min} \le x \le t_{\max} \\ 1 & t_{\max} < x \end{cases} \tag{18}$$

3.4 The Model of the Network

The network is modelled as a set of L links with capacities $c_l,\ l \in \{1, 2, \ldots, L\}$. The links are shared by a set of S sources indexed by $s \in \{1, 2, \ldots, S\}$, each using a subset L_s of links. The sets L_s define a $L \times S$ routing matrix

$$A_{ls} = \begin{cases} 1 \text{ if } l \in L_s \\ 0 \text{ otherwise} \end{cases} \tag{19}$$

which is a binary matrix where the 1s on a $l-$th row indicate the sources that share the link l and the 1s on $s-$th column represent the link crossed by source s. Each source is associated with its congestion window $W_s(t)$ at time t and each link l is associated with both its packet loss probability $p_l(t)$ (a scalar congestion measure) and with the instantaneous queue size $q_l(t)$.

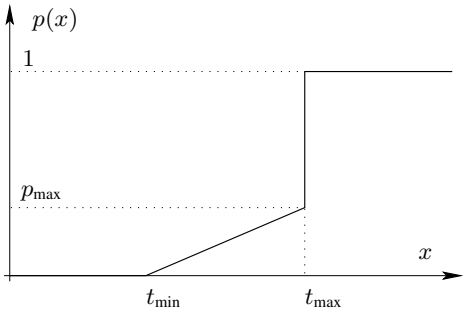

Fig. 3. RED drop function

The average RTT of the $s-$th TCP source at time t is approximated by

$$RTT_s(t) = \tau_s + \sum_{l \in L_s} \frac{q_l(t)}{c_l} \qquad s \in \{1, 2, \ldots, S\}, \tag{20}$$

which is the sum of the round trip delay τ_s associated with the connection and the total queueing delay of the path. Considering the losses at the different queues as independent of each other (a reasonable assumption when modelling RED queues), we can express the packet loss probability $\hat{p}_s(t)$ of source s as

$$\hat{p}_s(t) \;=\; 1 - \prod_{l=1}^{L} (1 - A_{ls} p_l(t)) \;\simeq\; \sum_{l \in L} A_{ls} p_l(t) \qquad s = 1, 2 \ldots S, \tag{21}$$

which corresponds to the end-to-end congestion measure for the $s-$th source.

The following is the differential version of the Lindley equation, describing the dynamic of $l-$th queue

$$\frac{dq_l(t)}{dt} = -1_{q_l(t)} c_l + \sum_{s=1}^{S} A_{ls} \frac{W_s(t)}{RTT_s(t)}.$$

Here, the derivative of the instantaneous queue length is the sum of two terms. The first one models the decrease of queue length, as long as it is greater than zero, due

to service of packets at a constant rate. The second term corresponds to the increase in queue length due to the arrival of packets from the TCP flows that share the $l-$th queue. Since we are interested in a mean value analysis, we take the expectation[2] of both sides of equation (3.4):

$$\begin{aligned}\frac{d\bar{q}_l(t)}{dt} &= E[-1_{q_l(t)}]\ c_l + \sum_{s=1}^{S} A_{ls} E\left[\frac{W_s(t)}{RTT_s(t)}\right] \\ &\approx -E[1_{q_l(t)>0}]\ c_l + \sum_{s=1}^{S} A_{ls} \frac{\overline{W}_s(t)}{\overline{RTT}_s(t)}\end{aligned}$$

In the derivation, we made the approximation $E[f(x)] \approx f(E[x])$, which is not strictly correct. However, simulation results suggest that at the steady-state the system reaches a quasi-periodic evolution where the random component of state variables, such as the instantaneous queue size, makes up a smaller and smaller contribution with respect to the deterministic one as the number of flows increases. Since the system is dominated by the deterministic evolution and the extent of fluctuations is small, the previous approximation is justified.

In order to approximate the term $E[1_q]$, we should consider that the bottleneck queues have $q_l(t) > 0$ with probability close to one, while the non-bottleneck queues are typically unloaded, which means $q_l(t) > 0$ with probability close to zero. On the basis of this observation, we approximate $E[1_{q(t)}] \simeq 1_{\bar{q}(t)}$, thus having

$$\frac{d\bar{q}_l(t)}{dt} = -1_{\bar{q}_l(t)} c_l + \sum_{s=1}^{S} A_{ls} \frac{\overline{W}_s(t)}{\overline{RTT}_s(t)} \tag{22}$$

The mean queue length is mapped by RED into a drop probability using the drop profile (18). We assume that RED estimates the average queue length using an exponential weighted moving average based on samples taken every T seconds. The smoothing filter (with α, $0 < \alpha < 1$, as weight) is described by

$$x_l[(k+1)T] = (1-\alpha)\, x_l[kT] + \alpha q_l[kT]$$

It is convenient to approximate the above equation with differential equation. Since this equation is a first order difference equation, the natural candidate is

$$\frac{dx_l(t)}{dt} = a x_l(t) + b q_l(t) \tag{23}$$

Recalling that in a sampled data system with $q(t) \equiv q[kT]$ in the interval [kT,(k+1)T], $x_l[(k+1)T]$ is given by

$$x_l[(k+1)T] = e^{aT} x_l[kT] + b \int_{kT}^{(k+1)T} e^{a(kT-\mu)} d\mu \cdot q_l[kT], \tag{24}$$

and comparing the coefficients of (23) and (3.4), we obtain

[2] Throughout this section we will indicate, when possible, the mean value with a bar sign to simplify the notation.

$$a = -b = \frac{\ln(1-\alpha)}{T}.$$

Then, we rewrite the expression (23), describing the behaviour of $x(t)$, by taking the expected value of both sides:

$$\frac{d\bar{x}_l(t)}{dt} = \frac{\ln(1-\alpha)}{T}(\bar{x}_l(t) - \bar{q}_l(t)) \tag{25}$$

3.5 The Model of the Source

The next step is to build a model of a TCP source. The model is based on the assumption that packet losses of a flow can be described by a Poisson counting process $\{N_s(t)\}$ with time varying rate $\lambda_s(t)$. The Poisson process is indeed suitable to represent the independent marking scheme used by AQM/RED. This process could be visualised imagining a *flow* of losses moving from network nodes towards TCP sources, whose rate is a function of the TCP flow rate.

If we denote with $N_s(t)$ the number of losses detected by the source s at time t, we can write the evolution of the congestion window as

$$dW_s(t) = \frac{dt}{RTT_s(t)} - \frac{W_s(t)}{2} dN_s$$

This equation only considers the AIMD behaviour of TCP. More specifically, the first term corresponds to the AI part, which increases the window size by one packet every RTT, and the second term corresponds to the MD part, which halves the congestion window immediately after the drop is detected by the sender ($dN_s(t) = 1$ in this case). Again, taking expectation, we obtain

$$\begin{aligned} dE[W_s(t)] &= E\Big[\frac{dt}{RTT_s(t)}\Big] - \frac{E[W_s(t)\, dN_s(t)]}{2} \\ d\overline{W}_s(t) &\simeq E\Big[\frac{dt}{RTT_s(t)}\Big] - \frac{\overline{W}_s(t)}{2}\lambda_s(t)dt \end{aligned} \tag{26}$$

where $\lambda_s(t)$ is the rate of loss indication at the sender. Note that in (26) in order to split the term $E[W_s(t)\, dN_s(t)]$ in a product of two factors $E[W_s(t)]E[dN_s(t)]$, we have assumed that the terms $W_s(t)$ and $dN_s(t)$ are independent. This is not exact, but it is still able to capture the dynamics of AIMD.

In proportional marking schemes (such as RED) the rate of marking/dropping indications is proportional to the share of bandwidth of the connection. That is, if the bandwidth achieved by source s is $W_s(t)/RTT_s(t)$, the expected value for drop rate at link $l \in L_s$ is

$$p_l(t) \cdot \frac{W_s(t)}{RTT_s(t)} \tag{27}$$

and equivalently the packet drop rate $\hat{p}_s(t)$ for a connection is calculated using (21).

However, we note that drops occur at the node about a RTT before they can be detected by the sender. This means that, in order to take into account the latency of feedbacks, we must shift the rate of congestion signals (27) forward in time of RTT_s seconds. Thus, from (26) the evolution of *cwnd* is given by

$$\frac{d\overline{W}_s(t)}{dt} = \frac{1}{\overline{RTT}_s(t)} - \frac{\overline{W}_s(t)}{2} \cdot \frac{\overline{W}_s(t - \overline{RTT}_s(t))}{\overline{RTT}_s(t - \overline{RTT}_s(t))} \hat{p}_s(t - \overline{RTT}_s(t)) \qquad (28)$$

In conclusion we have $2L + S$ coupled equations (22), (25) and (28) in the unknowns $(\bar{x}, \bar{q}, \bar{W})$, that can be solved numerically. The solution yields an estimate of the average transient behaviour of the system, providing directly the window size of each connection and the queue size at each node and, from them, the loss rate and average RTT. The time needed to get accurate results through the use of the model is several order of magnitude less than that needed by simulations and the gain increases as the topology becomes more complex. To have an idea of the advantage, let consider that the solution of a system consisting of a thousand connections takes a few seconds, while the corresponding detailed simulation can take several hours to complete.

3.6 Linear Analysis: The Single Link Case

In the following, we accomplish the task of linearising the previous set of equations in the case of a single link topology. The linearised system is suitable to be studied through the classic tools of linear control theory and gives us many suggestions on the way to modify the algorithm to fulfil requirements of stability and robustness [44,45].

Let us consider then N identical TCP Reno connections (i.e. with the same RTT) sharing a common link with capacity C. If we can assume all the connections synchronised (i.e. $W_s(t) = W(t)$, $\tau_s = \tau$ and $RTT_s(t) = R(t)$), we can rewrite, for a generic TCP flow, equation (22), defining the evolution of the mean value of *cwnd*, and equation (28), concerning the dynamic of the queue[3]

$$\begin{cases} \dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))} p(t - R(t)) \\ \dot{q}(t) = \frac{W(t)}{R(t)} N - C \end{cases} \qquad (29)$$

where the term $R(t) = \tau + \frac{q(t)}{C}$ represents the RTT for all the connections. When writing the equation for $\dot{q}(t)$, we assumed that the server is always transmitting packets, which is a reasonable assumption since we are studying the dynamics of the bottleneck. To complete the system of equations (29), we need also to specify the relationship between the queue size and the dropping probability, which depends on the employed AQM strategy. By borrowing the terminology from control-system language, we will refer to the AQM block as the *controller* and the rest of the system as the *plant*. A linear representation for the *plant* is derived in the following from the system (29), where the

[3] For ease of notation we will omit the sign of expectation and denote the temporal derivative of f as $\dot{f}$.

loss probability is considered as the input and the queue size as the output. Then, we will focus on the design of AQM controller using the tools of the linear control theory.

The main goal of AQM is to provide a stable closed-loop system. Beside the stability, there are other issues concerning control design. A feasible controller must possess an acceptable transient response and it must not be sensitive to the variations of model parameters and to disturbance factors. For instance, the presence of short lived connections in the queue has a noisy effect on long-lived TCP connections.

Small-signal analysis. The first step to linearise the system is to find the operating point (W_0, q_0, p_0), which is defined by $\dot{W} = 0$ and $\dot{q} = 0$. From (29) we have

$$\begin{cases} \frac{1}{R_0} - \frac{W_0^2}{2R_0} p_0 = 0 & \Rightarrow W_0 = \sqrt{\frac{2}{p_0}} \\ \\ \frac{W_0}{R_0} N - C = 0 & \Rightarrow W_0 = \frac{R_0 C}{N} \end{cases} \tag{30}$$

where $R_0 = \frac{q_0}{C} + \tau$. The operating point is the state-space point, to which the system would converge if it would be globally stable. For the case here considered, this is simply found accounting $1/N$ of the available bandwidth to each flow. Since equation (28) omits modelling of retransmission timeouts, the long term throughput W_0/R_0 comes out as the *one-on-square-root-p* law (3), which has been said inaccurate for high values of p. To refine the model, a term accounting for timeouts could be added to the right hand side of equation (28), as it was done in [26] introducing additional assumptions on the packet drop model. However, the small-signal analysis for this case would be quite complicated and would escape the introductory purposes of this paper.

Introducing difference variables $(\delta W, \delta p, \delta q)$, we can linearise (29) around the operating point

$$\begin{cases} \delta\dot{W}(t) = -\frac{N}{R_0^2 C}(\delta W(t) + \delta W(t - R_0)) \\ \qquad\qquad -\frac{1}{R_0^2 C}(\delta q(t) - \delta q(t - R_0)) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0) \\ \\ \delta\dot{q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \end{cases} \tag{31}$$

where equations (30) have been used to evaluate the derivatives. A simple expression for the eigenvalues of the linearised system can be estimated provided that

$$W_0 \gg 1 \qquad \Rightarrow \qquad \frac{N}{R_0^2 C} = \frac{1}{W_0 R_0} \ll \frac{1}{R_0}. \tag{32}$$

In this case indeed, the response-time of the aggregate of TCP flows is dominated by the time constant $R_0^2 C/N$, which is much larger than the round trip time R_0. This implies that in an interval R_0 the mean window size does not vary significantly with respect to its absolute value. Hence, we can approximate the system of equations by merging the terms $W(t)$ and $W(t - R_0)$, and neglecting the term $(\delta q(t) - \delta q(t - R_0))$

$$\begin{cases} \delta\dot{W}(t) = -\frac{2N}{R_0^2 C}\delta W(t) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0) \\ \\ \delta\dot{q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \end{cases} \tag{33}$$

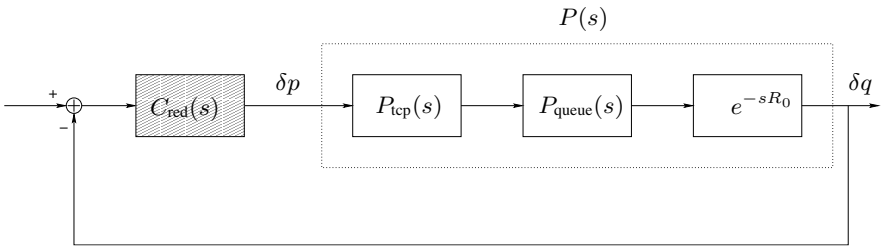

Fig. 4. Linearised block diagram

Figure 4 shows a block diagram for the system of equations (33), where $P_{\text{tcp}}(s)$ and $P_{\text{queue}}(s)$ are defined as

$$P_{\text{tcp}}(s) = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}}$$

$$P_{\text{queue}}(s) = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}}$$

The transfer function of the plant, relating the packet-marking probability to the queue length, is then given by:

$$P(s) = P_{\text{tcp}}(s)\, P_{\text{queue}}(s)\, e^{-sR_0} = \frac{\frac{R_0^3 C^3}{4N^2}}{(1 + s\frac{R_0^2 C}{2N})(1 + sR_0)}\, e^{-sR_0} \tag{34}$$

Some remarks on the $P(s)$ expression can be done. The static gain $R_0^3 C^3 / 4N^2$ is proportional to the RTT and the capacity of the link and inversely proportional to the number of active flows. The static gain is in turn inversely proportional to the *gain margin*, which is the maximum static gain of the control system that would make the system unstable. The gain margin is indeed defined as the amplitude response at the point where the phase response is $-\pi$.

If the number of flows is small, the static gain increases reducing stability and leading to a more oscillatory response. As we could expect, when we have few TCP flows, the extent of rate variation due to multiplicative decrease is larger than in the case of many flows and this impacts the stability of the system. Moreover, an increase of R_0, which corresponds to a larger delay in the control loop, reduces the controllability of the system, as confirmed by its negative influence on the *gain margin*.

From (34), it could be also easily deduced the phase margin $\phi_m = \omega_{pm} - \pi$, where ω_{pm} is the phase response when the amplitude response is 0 dB. The phase margin in this context could be interpreted as the amount of additional delay in the RTT that the system would tolerate without becoming unstable.

Setting RED parameters. The majority of Internet routers uses drop tail buffers. This could be interpreted, from a control-theory point of view, as an ON/OFF control

strategy, also known as relay-controller. Relay-controllers may cause instabilities in the system, such as an oscillating behaviour, and unpredictability due to non-linear effects. For instance, the queue size of a router can alternate between full and empty, which causes buffer overflows in one case and link underutilisation in the other. Mechanisms, such as RED or other AQM schemes, allow us to mitigate this condition.

Another objective of AQM is to reduce the mean queueing delays and delay variations. The smaller the queue-size, the less the time a packet spends in the queue and the less the end-to-end delay of a connection. However, forcing small queues may again lead to link underutilisation. A tradeoff between acceptable queueing delay and utilisation must be operated.

The queue control should be robust against variation of network parameters. This includes supporting a variable number of TCP sessions, variable RTTs and the presence of non congestion controlled traffic (that could be regarded as a sort of noise source on the TCP/AQM system). Here, we provide a condition that guarantees stability for a wide range of working conditions.

In order to find a Laplace representation of RED controller, we describe RED as the cascade of a smoothing filter (25) and a dropping function (18). Since the operating point falls between $t_{\min}$ and $t_{\max}$, it is easy to find a linear relation between small variations of dropping probability δp and small variations of the averaged queue size δx. This is then substituted into the transfer-function of AQM/RED yielding to

$$C_{\text{red}}(s) = K \cdot \frac{1}{1 + \frac{s}{\beta}} \tag{35}$$

where

$$K = -\frac{\ln(1-\alpha)}{T}, \qquad \beta = \frac{p_{\max}}{t_{\max} - t_{\min}} \tag{36}$$

$C_{\text{red}}(s)$ acts as a proportional controller with static gain K corresponding to the slope of the RED dropping profile. When we choose the $C_{\text{red}}(s)$, we need to take into account variations in the number of TCP sessions and RTT. In this case, the variations of the term R_0 are due to the propagation variable R_0.

We assume that the number of TCP sessions N is larger than a threshold $N_{\min}$ and the RTT R_0 is less than $R_{\max}$. Our goal is to select the RED parameters K and β that stabilise the system for all the values of N and R_0 included in their definition intervals. A closed-loop control system is stable if the response to any bounded input is a bounded output. In this case we have no inputs, so the system is stable if the response to whatever initial condition converges exponentially to zero.

Let us consider the frequency response of the open-loop system

$$L_{\text{red}}(j\omega) = C_{\text{red}}(j\omega)P(j\omega) = \frac{K\frac{(R_0C)^3}{(2N)^2}e^{-j\omega R_0}}{(\frac{j\omega}{\beta}+1)(\frac{j\omega}{\frac{2N}{R_0^2C}}+1)(\frac{j\omega}{\frac{1}{R_0}}+1)}$$

For the range of frequencies at least a decade lower than the minimum displacement of plant poles

$$\omega \leq \omega_g = \frac{1}{10}\min\left\{\frac{2N_{\min}}{(R_{\max})^2 C}, \frac{1}{R_{\max}}\right\} \tag{37}$$

the system frequency response can be approximated by

$$L_{\text{red}}(j\omega) \approx \frac{K\frac{(R_0C)^3}{(2N)^2}e^{-j\omega R_0}}{\frac{j\omega}{\beta}+1}$$

as the contribution of plant poles at the denominator is small. Then, we evaluate the amplitude of the system frequency response for $\omega = \omega_g$ and, under the hypothesis that $N \geq N_{\min}$, $R_0 \leq R_{\max}$, we find the following upper bound

$$\mid L_{\text{red}}(j\omega_g) \mid = \frac{K\frac{(R_0C)^3}{(2N)^2}}{\sqrt{\frac{\omega_g^2}{\beta^2}+1}} \leq \frac{K\frac{(R_{\max}C)^3}{(2N_{\min})^2}}{\sqrt{\frac{\omega_g^2}{\beta^2}+1}}$$

Now, enforcing the condition

$$\frac{K(R_{\max}C)^3}{(2N_{\min})^2} \leq \sqrt{\left(\frac{\omega_g}{\beta}\right)^2+1}, \tag{38}$$

we have $\mid L_{\text{red}}(j\omega_g) \mid \leq 1$. This means that the unit gain cross-over frequency, which is unique as the frequency response amplitude is a decreasing function of ω, is upper bounded by ω_g.

In order to establish the closed loop stability, we invoke the Nyquist criterion: if the open loop system has not unstable roots, the closed loop system is stable if the curve $L_{\text{red}}(j\omega), -\infty < \omega < \infty$ on the complex plane has not clockwise encirclement around $(-1+0j)$. Thus, recalling that $\omega_g R_0 \geq 0.1$ for (37), we have

$$\arg\{L_{\text{red}}(j\omega_g)\} \;\geq\; \arg\left\{\frac{K\frac{(R_{\max}C)^3}{(2N_{\min})^2}}{\frac{j\omega_g}{\beta}+1}\right\} - \omega_g R_0 \;\geq\; -\frac{\pi}{2} - 0.1 \;>\; -\pi$$

Hence, being $\mid L_{\text{red}}(j\omega) \mid \leq 1$ for $\omega \geq \omega_g$, the curve does not encircle the point $(-1+j0)$ and the system is stable. In conclusion, we found that, if K and β satisfy the condition (38), the linear system (34) with $C_{\text{red}}(s)$ as controller is stable for any $N \geq N_{\min}$ and $R_0 \leq R_{\max}$.

This example shows how the linear model can be used to build a robust design of RED, which accounts for the variation of the number of flows N and of the round trip time R. The rationale of this design is to force the controller to dominate the closed-loop behaviour. This is done by choosing a closed loop time-constant (close to ω_g) at least a decade higher than TCP time-constant or queue time-constant. The expression (37) leaves a degree of freedom in choosing the parameters (K, β) on the boundary of the set defined by (38). To determine the parameters, other constraints can be placed.

We could have chosen a multiplicative factor in (37) larger than 0.1. This would lead to a faster response time of the system, but would produce a controller with lower stability margins. In order to increase the bandwidth of the system, other strategies could be introduced as well, such as, for instance, the classical proportional-integral (PI) controller discussed in [6].

4 Conclusions

In this paper we introduced the most relevant works in the field of TCP analytical modelling. We considered models that describe the TCP behaviour both as an individual source (single connection models) or as a part of a network system described in terms of differential equations (fluid models).

The analysis of individual TCP connections in isolation led to expressions for the TCP throughput under given networking conditions. Since the analysis required the independence of the network state and the source state, these models are adequate when the network is loaded by a large number of flows. In fact, this condition is often verified. Asymptotic results [46,47] show indeed that the correlation between the state of two connections rapidly decreases as the number of connection increases.

In the second part, we faced the problem of analytically describing the interaction between TCP and AQM. In particular, we focused on networks with RED policy. We derived analytical results that allow a qualitative understanding of the transient behaviour of TCP over RED networks. Furthermore, we presented results on stability and robustness of the network system itself. As a final example of application of these techniques, we described a method proposed in literature to select the AQM parameters that lead to stable operations of the linear feedback control system.

This introduction did not address other important models available in literature. Some authors addressed for instance the modelling of short-lived connections [14,15,16] where other aspects of TCP, such as the Slow Start phase, dominate the TCP behaviour. These studies are well justified by the fact that many Internet flows are short and are completed before reaching a steady-state. As pointed out by Jacobson [48], the Slow Start phase should be seen as a mechanism for fast approaching of the equilibrium point, while the Congestion Avoidance as a mechanism for asymptotic stability. Thus, accurate modelling of this phase somehow complements the results provided by the analysis of long lived connections.

References

1. Postel, J.: Transmission Control Protocol. RFC 793 (Standard), Updated by RFCs 1122, 3168 (September 1981)
2. Paxson, V., Floyd, S.: Why We Don't Know How to Simulate the Internet. In: Proc. of the 29th Conference on Winter Simulation, Atlanta (US), pp. 1037–1044 (December 1997)
3. Floyd, S., Handley, M., Padhye, J., Widmer, J.: TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348 (Proposed Standard) (September 2008)
4. Floyd, S., Fall, K.: Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking 7(4), 458–472 (1999)
5. Ott, T.J., Lakshman, T.V., Wong, L.H.: SRED: Stabilized RED. In: Proc. of IEEE INFOCOM, vol. 3, pp. 1346–1355 (1999)
6. Hollot, C.V., Misra, V., Towsley, D., Gong, W.: Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. IEEE Transactions on Automatic Control 47(6), 945–959 (2002)
7. Chait, Y., Hollot, C.V., Misra, V., Towsley, D.F., Zhang, H., Lui, J.: Providing Throughput Differentiation for TCP Flows Using Adaptive Two Color Marking Multi-Level AQM. In: Proc. of IEEE INFOCOM, New York (US), pp. 23–27 (June 2002)

8. Kunniyur, S., Srikant, R.: End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks. In: Proc. of IEEE INFOCOM, Tel Aviv (Israel), vol. 3, pp. 1323–1332 (March 2000)
9. Kelly, F.P.: Charging and Rate Control for Elastic Traffic. European Transactions on Telecommunications 8, 33–37 (1997)
10. Low, S.H.: A Duality Model of TCP and Queue Management Algorithms. IEEE/ACM Transactions on Networking 11(4), 525–536 (2003)
11. Massoulié, L., Roberts, J.: Bandwidth Sharing: Objectives and Algorithms. IEEE/ACM Transaction on Networking 10(3), 320–328 (2002)
12. Mo, J., Walrand, J.: Fair End-to-End Window-based Congestion Control. IEEE/ACM Transactions on Networking 8(5), 556–567 (2000)
13. Khalifa, I., Trajkovic, L.: An Overview and Comparison of Analytical TCP models. In: Proc. of International Symposium on Circuits and Systems (ISCAS), Vancouver (Canada), vol. 5, pp. 469–472 (2004)
14. Heidemann, J., Obraczka, K., Touch, J.: Modeling the Performance of HTTP over Several Transport Protocols. IEEE/ACM Transactions on Networking 5(5), 616–630 (1997)
15. Cardwell, N., Savage, S., Anderson, T.: Modeling TCP Latency. In: Proc. of IEEE INFOCOM, vol. 3, pp. 1742–1751 (2000)
16. Mellia, M., Stoica, I., Zhang, H.: TCP Model for Short Lived Flows. IEEE Communications Letters 6(2), 85–87 (2002)
17. Paxson, V.: Empirically Derived Analytic Models of Wide-Area TCP Connections. IEEE/ACM Transactions on Networking 2(4), 316–336 (1994)
18. Padhye, J., Firoiu, V., Towsley, D.F., Kurose, J.F.: Modeling TCP Reno Performance: A Simple Model and its Empirical Validation. IEEE/ACM Transaction on Networking 8(2), 133–145 (2000)
19. Floyd, S.: Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic. ACM SIGCOMM Computer Communication Review 21(5), 30–47 (1991)
20. Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The Macroscopic Behaviour of the TCP Congestion Avoidance Algorithm. ACM SIGCOMM Computer Communication Review 27(3), 67–82 (1997)
21. Lakshman, T., Madhow, U.: The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. IEEE/ACM Transactions on Networking 5(3), 336–350 (1997)
22. Kumar, A.: Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. IEEE/ACM Transactions on Networking 6(4), 485–498 (1998)
23. Misra, A., Ott, T.J.: The Window Distribution of Idealized TCP Congestion Avoidance with Variable Packet Loss. In: Proc. of IEEE INFOCOM, New York (US), vol. 3, pp. 1564–1572 (1999)
24. Casetti, C., Meo, M.: A New Approach to Model the Stationary Behavior of TCP Connections. In: Proc. of IEEE INFOCOM, Tel Aviv (Israel), vol. 1, pp. 367–375 (2000)
25. Altman, E., Barakat, C., Ramos, V.M.R.: Analysis of AIMD Protocols over Paths with Variable Delay. Computer Networks 48(6), 960–971 (2005)
26. Misra, V., Gong, W.B., Towsley, D.F.: Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. ACM SIGCOMM Computer Communication Review 30(4), 151–160 (2000)
27. Hollot, C.V., Misra, V., Towsley, D.F., Gong, V.: A Control Theoretic Analysis of RED. In: Proc. of IEEE INFOCOM, Anchorage (Alaska), pp. 1510–1519 (April 2001)
28. Liu, Y., Presti, F.L., Misra, V., Towsley, D.F., Gu, Y.: Fluid Models and Solutions for Large-Scale IP Networks. ACM SIGMETRICS Performance Evaluation Review 31(1), 91–101 (2003)

29. Gu, Y., Liu, Y., Towsley, D.F.: On Integrating Fluid Models with Packet Simulation. In: Proc. of IEEE INFOCOM, Hong Kong (China), vol. 4, pp. 2856–2866 (March 2004)
30. Altman, E., Avrachenkov, K., Barakat, C.: A Stochastic Model of TCP/IP with Stationary Random Losses. In: ACM SIGCOMM, Stockholm (Sweden), pp. 231–242 (2000)
31. Bu, T., Towsley, D.: Fixed Point Approximations for TCP behavior in an AQM Network. ACM SIGMETRICS Performance Evaluation Review 29(1), 216–225 (2001)
32. Garetto, M., Lo Cigno, R., Meo, M., Marsan, M.A.: Closed Queueing Network Models of Interacting Long-lived TCP Flows. IEEE/ACM Transactions on Networking 12(2), 300–311 (2004)
33. Paxson, V., Floyd, S.: Wide-Area Traffic: The Failure of Poisson Modeling. IEEE/ACM Transactions on Networking 3(3), 226–244 (1995)
34. Park, K., Willinger, W.: Self-Similar Network Traffic and Performance Evaluation. Wiley-Interscience, Inc., Hoboken (2000)
35. Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. RFC 2581 (Proposed Standard) (April 1999)
36. Floyd, S., Henderson, T., Gurtov, A.: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (Proposed Standard) (April 2004)
37. Fall, K., Floyd, S.: Simulation-based Comparisons of Tahoe, Reno and SACK TCP. ACM SIGCOMM Computer Communication Review 26(3), 5–21 (1996)
38. Stevens, W.: TCP/IP Illustrated. The Protocols, vol. 1. Addison-Wesley, Reading (1994)
39. Villamizar, C., Song, C.: High Performance TCP in ANSNET. ACM SIGCOMM Computer Communications Review 24(5), 45–60 (1994)
40. Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking 1(4), 397–413 (1993)
41. Lin, D., Morris, R.: Dynamics of Random Early Detection. ACM SIGCOMM Computer Communications Review 9(4), 127–137 (1997)
42. Feng, W., Kandlur, D.D., Saha, D., Shin, K.G.: A Self-Configuring RED Gateway. In: Proc. of IEEE INFOCOM, New York (USA), vol. 3, pp. 1320–1328 (1999)
43. Liu, S., Basar, T., Srikant, R.: Exponential-RED: A Stabilizing AQM Scheme for Low- and High-speed TCP Protocols. IEEE/ACM Transactions on Networking 5(5), 1068–1081 (2005)
44. Christiansen, M., Jaffay, K., Ott, D., Smith, F.D.: Tuning RED for Web Traffic. IEEE/ACM Transactions on Networking 9(3), 249–264 (2001)
45. Bonald, T., May, M., Bolot, J.C.: Analytic Evaluation of RED Performance. In: Proc. of IEEE INFOCOM, Tel Aviv (Israel), vol. 3, pp. 1415–1424 (2000)
46. Tinnakornsrisuphap, P., La, R.J.: Asymptotic Behavior of Heterogeneous TCP Flow and RED Gateways. IEEE/ACM Transactions on Networking 14(1), 108–120 (2006)
47. Tinnakornsrisuphap, P., Makowski, A.M.: Limit Behavior of ECN/RED Gateways Under a Large Number of TCP Flows. In: Proc. IEEE INFOCOM, San Francisco (US), vol. 2, pp. 873–883 (2003)
48. Jacobson, V.: Congestion Avoidance and Control. ACM SIGCOMM Computer Communication Review 18(4), 314–329 (1988)