

# Sender-Side TCP Modifications: An Analytical Study<sup>\*</sup>

R. Lo Cigno<sup>1</sup>, G. Procissi<sup>2</sup>, M. Gerla<sup>3</sup>

<sup>1</sup> Politecnico di Torino, Dipartimento di Elettronica,  
Corso Duca degli Abruzzi, 24, 10129 Torino, Italy  
locigno@polito.it

<sup>2</sup> Università di Pisa, Dipartimento di Ingegneria della Informazione,  
Via Diotallevi 2, 56126 Pisa, Italy  
g.procissi@iet.unipi.it

<sup>3</sup> Computer Science Department – Boelter Hall – UCLA  
405 Hilgard Ave., Los Angeles, CA, 90024 – USA  
gerla@cs.ucla.edu

**Abstract.** This paper consider a number of modifications that can be applied to the congestion control algorithm of a TCP sender without requiring the cooperation either of the network or of the receiver, analyzing their impact on the performance of the protocol. We use a theoretical approach based on the use of queueing networks for the description of the protocol dynamics and a fixed point approximation to derive the working point of the IP network. Our results show that in presence of short lived connections the impact of the transient behavior of TCP on the network performance is dominant, and major performance improvements can be obtained only if the transient behavior is improved.

## 1 Introduction and Motivations

Recent years have seen a large research effort focused on Internet congestion control, the TCP protocol being the pivot issue of the effort. Several papers [1, 2] and RFCs [3] (just to mention some of them), were devoted to propose improved TCP versions. Other works focused on modeling either the TCP behavior [4, 5, 7] or the whole Internet “transfer function” [6], with the aim of gaining insight in the network dynamics, thus enhancing capabilities for designing and deploying improved protocol versions.

While several recent proposals [8–10] advocate the use of explicit feedback from the network, or, at least, the cooperation of the sender and the receiver (like TCP-SACK), in order to improve the congestion control, other authors [11] claim that modifications in the congestion control algorithms should involve the TCP sender only in order to avoid any necessity for co-ordination in the deployment of new TCP versions. Clearly, new TCP versions must be backward compatible and “friendly” to existent TCP implementations (TCP-NewReno in particular).

---

<sup>\*</sup> This work was supported in part by the MAYA (Next Generation Modeling & Simulation Tools for Global Networks) project and in part by the Italian Ministry for University and Research (MIUR) through the PLANET-IP project. The work was carried out when both Renato Lo Cigno and Gregorio Procissi were visiting U.C.L.A. during summer 2001.

The focus and contribution of this paper lie in the exploration of several possible TCP modifications on the sender side that change the way TCP tries to avoid congestion and react to it. We do not claim that these are the only possible modifications, but they surely have a major impact on the main parameters that govern the closed-loop behavior of the TCP-sources/transport-network system. As discussed in [12, 6, 15, 16, 13, 14] the reasons of TCP performance cannot be searched for in the protocol alone, but have to be analyzed in a context where TCP is just one piece of the overall transfer function of the closed-loop system. All the papers cited above give a deep insight in the system behavior, but the modeling technique adopted there does not include enough details to account for “apparently minor” protocol modifications and, most of all, for the transient behavior of TCP during the first slow start after opening the connection, which instead dominates the performance when short lived connections are involved.

The modeling technique we adopt, shortly described in Sect. 2, allows taking into account both the TCP transient and any kind of protocol modification. The closed-loop nature of the system is taken into account with a FPA (Fixed Point Approximation) method, that, in the case of the system under analysis, ensure the unicity and existence of the solution as demonstrated in [17].

The aim of our work is not the proposal of a specific TCP modification, but the exploration of the possible modifications and the benefits that derives from them. Moreover, this paper shows that the use of a powerful modeling technique can help in designing protocol modifications (new protocols?) without incurring the risks inherent to empirical/heuristic design.

## 2 The Modeling Technique

Following the approach first adopted in [7, 19, 20], in this paper we use an open multi-class queuing network (OMQN)-based description of the TCP protocol to investigate the benefits and drawbacks of possible modifications to TCP. An OMQN is a queueing network in which all queues are  $M/G/\infty$ . In this model, each customer (namely a TCP connection) is uniquely identified by a pair  $(q, c)$ , where  $q$  represents a specific state of the protocol and  $c$  identifies the number of remaining packets to be sent before the completion of the flow. An OMQN model is capable of describing any protocol whose dynamics can be described with a Finite State Machine (FSM); the use of classes to describe the backlog of the connection allows to model short lived connections. Given the average RTT of connections and the average packet loss probability, the OMQN model defines the load offered to the IP network, as well as the throughput and the duration of connections. The model is complemented with a single- or multi-bottleneck description of the IP network loaded by the TCP connections, that, given the load, computes the average loss probability. The overall solution is obtained with a fixed point iterative method.

The OMQN modeling technique has proven to be extremely accurate [7, 20], and this is the main reason that led us to choose this analytic approach for our study, instead of a more common simulation approach. Its powerful modeling paradigm and computational efficiency, associated with its proven accuracy, enables to gain the best possible insight in the protocol dynamics and in the consequences of protocol modifications;

while simulation studies are always limited in their scope by computational costs and difficulties in the interpretations of the results.

Any further detail about the OMQN modeling technique is superfluous in this context and we refer the interested reader to the available literature [18, 7, 19–21].

Protocol modifications are modeled in two possible ways. The first one imply modifying the protocol FSM, hence adding o deleting queues in the OMQN. The second one does not affect the protocol states, but only its dynamics, and corresponds to the modification of queues service rates and transition probabilities, rather than modifying the OMQN structure.

### 3 TCP Protocol Modification

The basic congestion avoidance algorithm of TCP NewReno (presently the most diffused TCP version) is AIMD. The basic properties of AIMD algorithms, generalizing the TCP protocol, were studied in [22, 23]. Several approximations are introduced for analytical tractability, the main of which is the incorrelation of the loss process with the TCP protocol, which does not allow to fully appreciate any property of the protocol that leads to a smaller loss probability.

Recently, studies like [12, 15] hint to possible performance limitations of TCP, when coupled with AQM (Active Queue Management) schemes, due to oscillatory behaviors rooted in an excessive loop gain of the congestion avoidance mechanism. On the other hand, there is no doubt that explicit bandwidth feedback from the network can optimize the performance of TCP.

These observations, together with the desire to keep the congestion control algorithm in end-hosts only, without requiring the cooperation of the network, lead to propose new versions of TCP, like Vegas [2] or Westwood [11], that try to estimate the available bandwidth (or fair bandwidth share) within the network. In both schemes the major problem arises from the difficulty of correctly estimating the available bandwidth.

Finally, measures and simulations highlighted how the first slow start is often dominating the performance of the whole flow when its length is limited. This is obvious if the flow is composed of just a few packets, but, if the maximum window size is large, its effect is dominating also connections of hundreds of packets, that indeed dominate the performance of the whole Internet. To appreciate the importance of the first slow start, it must be recalled that TCP transmits  $2 \times W_{fp}$  packets in slow start, where  $W_{fp}$  is the dimension of the congestion window when the first packet loss of the flow is detected. For instance, if there are no losses and the maximum window size is 50, then 100 packets are transmitted during the first slow start. With the standard Ethernet's MSS, this roughly corresponds to 1.2 Mbits. Indeed what may happen, and really happens more often that one would believe, so that most of the packets of a flow are transmitted during the first slow start without losses, then when the window is inflated enough to load the network, a burst of packets is lost, leading to a timeout.

In light of the above considerations we analyze three possible modifications of the TCP protocols, whose separate impact on performance will be discussed in Section 4:

- RFS, that reduces the window oscillations;
- $W_pP$ , that modifies the steady state loop gain;

- ESSE, that improves the TCP transient behavior.

In the following, we formally define these modifications and describe how they can be modeled, assuming that the reader is familiar with the description of TCP through an OMQN.

### 3.1 RFS: Packet Drop Reaction and Bandwidth Estimation

In lack of explicit bandwidth feedback from the network, the congestion control algorithm cannot avoid probing the network capacity and including some form of window oscillations around the steady state operating point. Indeed, the fast recovery option of TCP implicitly assumes that the available bandwidth within the network is somewhere between  $W_l/2 \cdot \overline{\text{RTT}}$  and  $W_l \cdot \overline{\text{RTT}}$ , where  $W_l$  is the dimension of the window when a packet loss is detected. Without discussing here the correctness of this assumption, it is clear that, if some better estimation of the available bandwidth can be obtained (and we are not concerned here on *how* it is obtained), then, after a loss event, the TCP protocol can restart the transmission from the window size corresponding to this estimation, say  $W_{fs}$  (the subscript “fs,” standing for fair-share). This would reduce the amplitude and frequency of the window oscillations, and help increasing the network stability and performance. Obviously the estimation of the fair share window  $\hat{W}_{fs}$  is a real value, while the window size is an integer: the rounded or truncated value can be used instead. We point out that this modification is not easily compared with the *responsiveness* of an AIMD protocol as defined in [23], since the relationship between the current window size and  $W_{fs}$  is not known, and for this reason we will avoid the use of this term.

From the protocol point of view, this modification is trivial, since it only implies to assign a different value to *cwnd* after a fast recovery. The OMQN model can take this modification into account exactly in the same way it accounts for the window thresholds distribution (see [7]), assigning to  $W_{fs}$  the value of the average window size computed in the previous iteration, i.e., by evaluating an ensemble average taken over the active protocol states and iterating the solution until convergence.

Since the available bandwidth computed with the OMQN is not subject to evaluation errors, in order to estimate the impact of bandwidth estimation errors on the performance of the resulting protocol, we have included in the model an error function. The error function implementation is trivial: the transition exiting a fast recovery state is not deterministic but follows a probability distribution centered around the value of  $W_{fs}$  computed by the OMQN. Obviously the distribution can assume only integer numbers between 2 and  $W_{\max}$ , where  $W_{\max}$  is the maximum window size negotiated between the sender and the receiver.

To exemplify the modeling process, Fig. 1 reports the OMQN portion representing the transitions from a generic fast recovery state with window size  $n$  (queue  $LF_n$ ) to the congestion avoidance states with (queues  $L_i$ ). The broken line transition corresponds to the event of losing the re-transmitted packet and leads to a timeout (not shown in the figure for the sake of simplicity). The vector  $\mathcal{E}(w)$  is a probability vector that describes the estimation errors.

In this study we consider only three simple cases of bandwidth estimation errors.

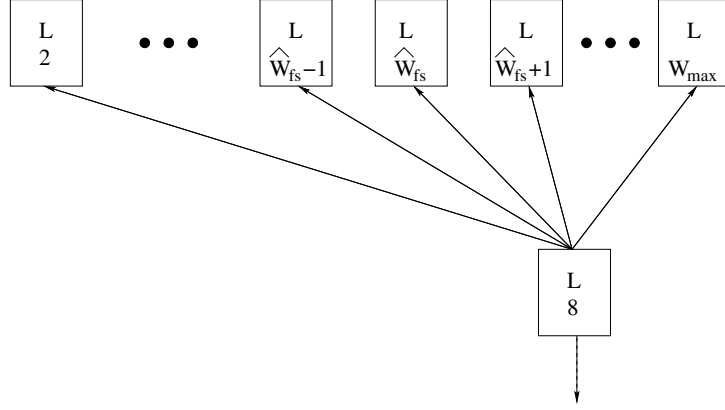


Fig. 1. OMQN description of RFS exiting a fast recovery with window size 8

1. *Best Bandwidth Estimator (BBE)*. The information on the connection's fair share is supposed to be exact. The corresponding distribution function  $\mathcal{E}(w)$  is then:

$$\mathcal{E}(w) = \begin{cases} 1 - p & w = \lfloor \hat{W}_{fs} \rfloor \\ p & \lfloor \hat{W}_{fs} \rfloor + 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where  $p = \hat{W}_{fs} - \lfloor \hat{W}_{fs} \rfloor$ .

2. *Worst Bandwidth Estimator (WBE)*. The information on the connection's fair share is supposed to be the worst possible. The RFS algorithm has no 'real' clue when setting the new value of the *cwnd*. The corresponding distribution function  $E(w)$  is then:

$$\mathcal{E}(w) = p \quad \forall w = 2, 3, \dots, W_{\max} \quad (2)$$

where  $p = \frac{1}{(W_{\max} - 1)}$ .

3. *'Triangular' Bandwidth Estimator (TBE)*. The information on the connection's fair share is supposed to be affected by an error function whose probability distribution is linearly decreasing, eventually reaching zero. The resulting distribution of the  $\mathcal{E}$  vector is triangular, centered on  $\hat{W}_{fs}$  as shown in Fig 2. The distribution can be asymmetrical, emulating skewed errors:

$$\mathcal{E}(w) = \begin{cases} c_u(w - \hat{W}_{fs} + e_u) & \hat{W}_{fs} - e_u \leq w < \hat{W}_{fs} \\ c_o(w - \hat{W}_{fs} - e_o) & \hat{W}_{fs} \leq w \leq \hat{W}_{fs} + e_o \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where,  $e_u$  is the maximum admissible error underestimating the fair share,  $e_o$  is the maximum admissible error overestimating the fair share, and  $\frac{p_{fs}}{2}(e_u + e_o) = 1$ ;  $c_u$  and  $c_o$  are the slopes of the under- and over-estimation, respectively, and are given by:  $c_u = p_{fs}/e_u$  and  $c_o = -p_{fs}/e_o$ .

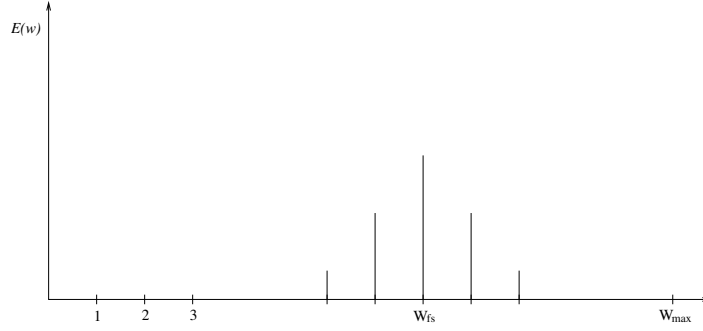


Fig. 2. Error distribution function for the ‘triangular’ Bandwidth Estimator

### 3.2 $W_pP$ : Window Increase and Aggressiveness

So far we have described the modifications of the congestion control algorithm as far as its reaction to packet drops concerns. However, if the aim is the reduction of the window oscillations, the window increase during the congestion avoidance phase must be also modified. Following the theoretical results in [13], to enhance network stability, the loop gain, and hence the protocol aggressiveness, should be reduced. Moreover, we argue that, if there is a reasonable estimation of the fair share, only a very gentle probing for extra bandwidth has to be performed in order to refrain from over-congesting the network and to assure the necessary level of friendliness with older versions of TCP.

We call this modification  $W_pP$  (*Window  $p$ -Persistent*), from the modeling solution in the OMQN, represented in Fig. 3, which is exactly the implementation of the  $p$ -persistent algorithm in MAC protocols: at any RTT, increase the value of  $cwnd$  of 1 segment with probability  $p$  and keep the window size unchanged with probability  $1 - p$ .

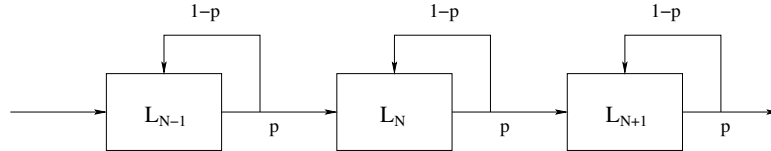


Fig. 3. OMQN representation of the  $W_pP$  protocol modification

The implementation in TCP can be deterministic, increasing the  $cwnd$  by 1 segment every  $1/p\overline{RTT}$ , or, closer to the present CP implementation, increasing  $cwnd$  by  $\frac{p}{cwnd}$  MSS every valid, non duplicated ACK.

### 3.3 ESSE: Reducing the Transient Overshoot

So far we have discussed only modifications to the steady state behavior of the congestion control algorithm, disregarding the role of the slow start phase. Internet traffic,

however, is dominated by short to medium connections, so that the steady state of the network is indeed the superposition of the transients of many flows. In such a scenario, disregarding the slow start, and specially the first one, when the TCP threshold is not yet set, is extremely dangerous.

Since during slow start TCP doubles the window size at each RTT, and TCP reaction time cannot be any smaller than the RTT itself, when the threshold is not set there is an enormous potential for burst losses. In line of principle, the modification is trivial: estimate the available resources and exit the slow start as soon as the window size reach this estimation. We are perfectly aware that the estimation of the resources during the initial transient is extremely critical. For this reason, we assume that there is no reliable estimation until the window reach the dimension of five segments after three RTTs. Errors on the bandwidth estimation are modeled as for the RFS case; however, the error distributions during the first slow start and during congestion avoidance are independent, so that we can model a larger error during the first slow start.

#### 4 Assessment of the Modifications Performance

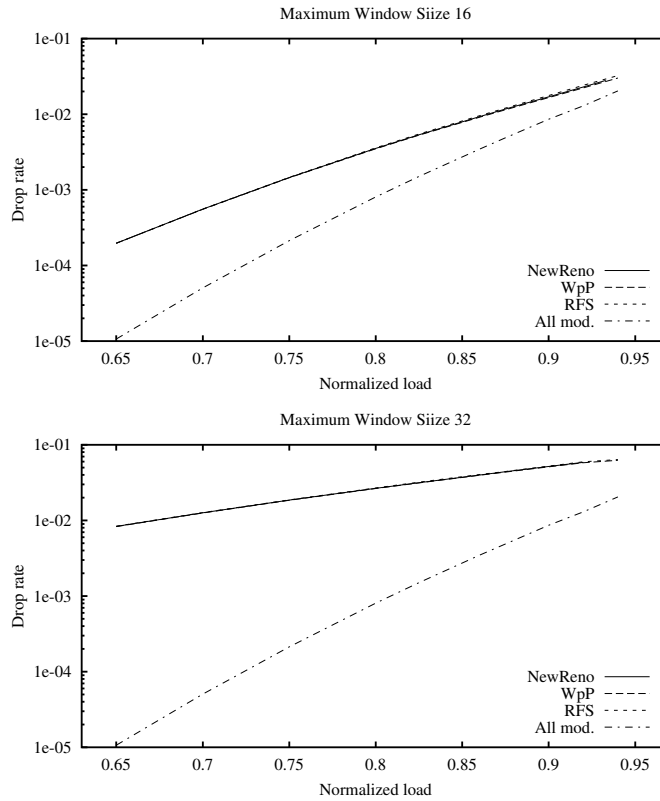
We consider two different scenarios, which are frequently encountered in the Internet, to discuss the impact of the modifications we analyze. Both scenarios refer to wide area networks (WAN), since in local (LAN) environment, the congestion control algorithms of TCP very rarely play a major role. We only consider standard, FIFO queueing with drop tail policies, which are by far the most diffused queueing schemes in deployed routers. Drop tail buffers introduce correlation in losses, which are accounted for in our models with the techniques described in [7, 20]. The presence of active queue management (AQM) policies of random losses due to link errors are left for future study, though they do not represent a major modeling problem.

The first scenario corresponds to the case where the bottleneck is represented by the peering point between to ISPs. This is often the case in web-browsing USA sites from Europe, since the Internet traffic is highly asymmetrical (more downloads *from* USA *to* Europe than vice-versa). The peering contracts between ISPs result in a bottleneck whose available bandwidth is nowhere near the installed capacity on transoceanic links. We assume, in this case, a bottleneck of 10 Mbit/s, with an average length of the connections of roughly 10,000 km, resulting in an average RTT of 100 ms plus the queueing at the bottleneck, which is directly computed by the model and depends on the buffer size and bottleneck load. We will refer to this scenario as the PEERING scenario.

The second scenario correspond to the case where the the bottleneck of the connections is the access link of the institution where the connections originate/terminate. In this case, the bandwidth of the bottleneck can be extremely variable, depending upon the institution dimension and similar factors. We deem that this scenario is most interesting when the access link is fairly fast and, on average, there are many flows competing for the resources. We assume that the bottleneck is a 100 Mbit/s link. In this case, the average connection length is smaller, since there are both transoceanic connections and “European” connections<sup>1</sup>. We arbitrarily set the average length of the connections to

<sup>1</sup> Notice that we set this hypothetical institution in Europe, but reversing the situation and having it in the USA, would not change the basic layout of the scenario

3,000 km, obtaining an average RTT of 30 ms. We call this scenario ACCESS. In both cases, the bottleneck buffer is set to 64 packets.

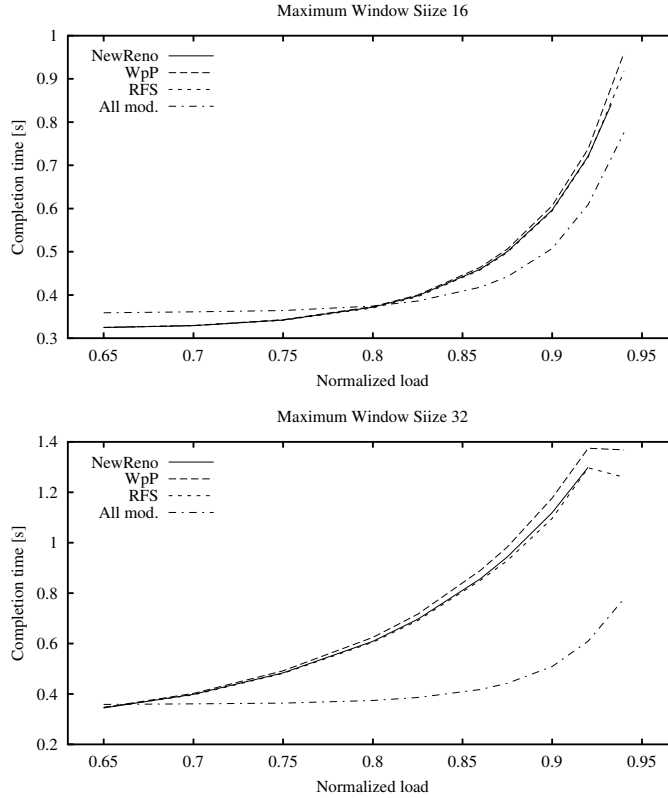


**Fig. 4.** Packet drop rate in the ACCESS scenario considering the  $W_pP$ , RFS modifications separately, or joint with the ESSE modification

First of all we examine the packet drop rate in the ACCESS scenario, considering separately the  $W_pP$  and the RFS modifications, or both of them joint with ESSE modifying also the transient behavior. Flows are all 100 packets long, but we consider two different maximum window sizes (MWS), namely 16 and 32 packets. Fig. 4 reports the packet drop rate as a function of the nominal<sup>2</sup> normalized traffic load. It is quite clear that the drop rate is dominated by the transient behavior: unless the ESSE modification is enabled, the loss rate is almost indistinguishable (at least in logarithmic scale) from the loss rate of NewReno connections. It is also interesting to notice that the MWS has

<sup>2</sup> This is the load that the bottleneck would have if there were no retransmissions, hence the actual load of the network is higher

a major impact on the loss rate, and the gain induced by the ESSE modification grows with the window size.



**Fig. 5.** Average completion times in the ACCESS scenario considering the W<sub>p</sub>P, RFS modifications separately, or joint with the ESSE modification

The performance gain obtained in terms of packet drop rate are striking; however, end users are more interested in the time they need to transfer the information. The two plots in Fig. 5 report the average flow completion time corresponding to the same situation of Fig. 4. It is clear that with large MWS the gain with the three joint modifications is determinant, specially at high loads (the solution of the model show some numerical instability at very high loads, which cause the connections' duration of NewReno, W<sub>p</sub>P and RFS to decrease slightly at load 0.94).

The situation with small MWS is instead more complicate. Though not quantitatively large, the advantage at high loads is still clear when all modifications are considered together; however, at light loads, the ESSE modification seems to penalize, though only marginally, the connections. The reason is a too early exit from the slow start, that avoids connections to reach the MWS. Indeed, with small drop rates, most

of the connections terminate without losses, hence the larger the reached window in slow start, the faster is the transfer. Results would probably be different considering the 90<sup>0</sup> percentile of the completion time. Unfortunately, our model is not yet capable of computing completion time variances or distributions.

#### 4.1 The Influence of Bandwidth Estimation Errors

The performance of a protocol that has a reliable estimation of the available bandwidth and makes an intelligent use of this knowledge shouldn't be a surprise; however, we are interested in analyzing the resilience of such protocols to estimation errors. The estimation errors are modeled as described in Sect. 3.1. When the errors have a triangular distribution, the maximum relative error is 25% of the average available bandwidth. We only consider all the modifications implemented together and the error estimation function is the same during slow start and congestion avoidance. The completion times of NewReno are reported for comparison.

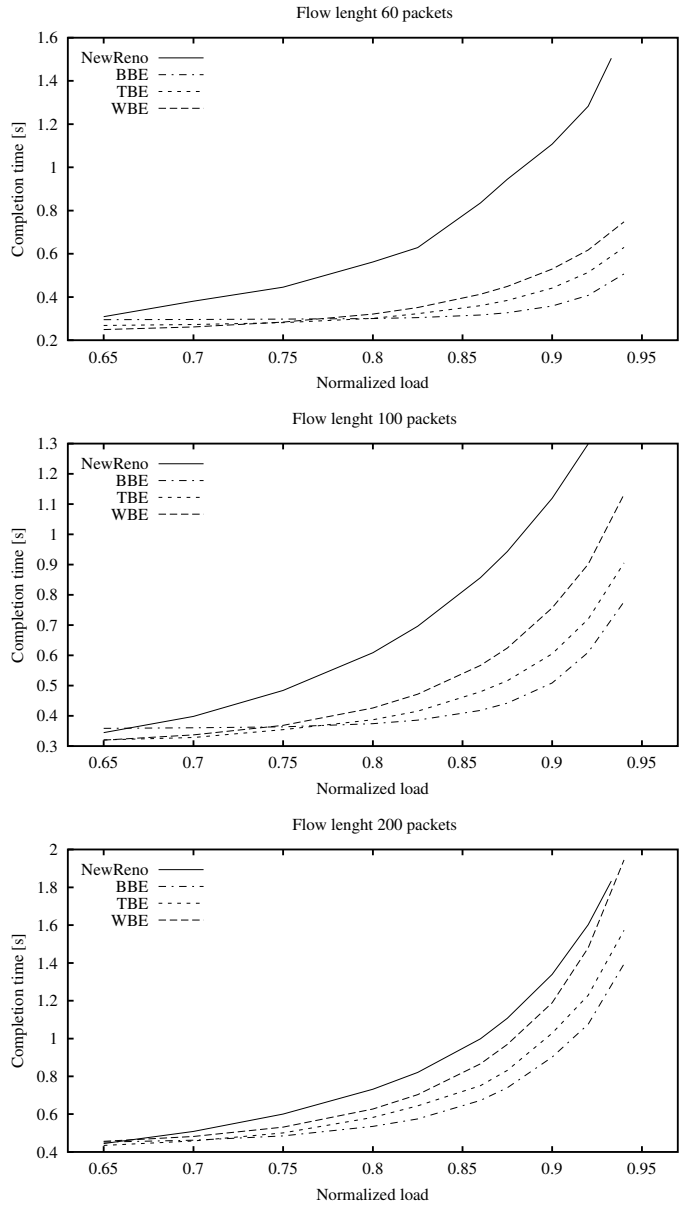
We analyze here only the completion times of connections, since these also reflect the underlying loss rate.

Figs. 6 and 7 report the average completion times for connections that are 60 (top plot), 100 (middle plot) and 200 (bottom plot) segments long, in the ACCESS and PEERING scenarios respectively. 'BBE' curves refer to the ideal bandwidth estimation, 'TBE' curves to the triangular error function and 'WBE' curves to the case when the estimation is uniformly distributed, i.e., drawn completely at random. The qualitative behavior is the same in all cases, regardless of the bottleneck position or capacity, or of the average connections' length. The striking result is that NewReno, at high loads, always performs so poorly as to be worse than choosing the window size at random. The explanation is rooted once more in the transient behavior of the protocol, and indeed, when connections are longer, the performance gap is partially filled. NewReno, as all TCP implementations, during the first slow start grows the window until the network is congested and packets are dropped. In other words, during the initial transient, NewReno does nothing to estimate the available resources, but assumes they are "infinite," deterministically overloading the network; this behavior ends up in poor performance because of the higher packet loss rate, even higher than a protocol that, instead of correctly evaluating the available resources draws a uniform random variable to set the window size.

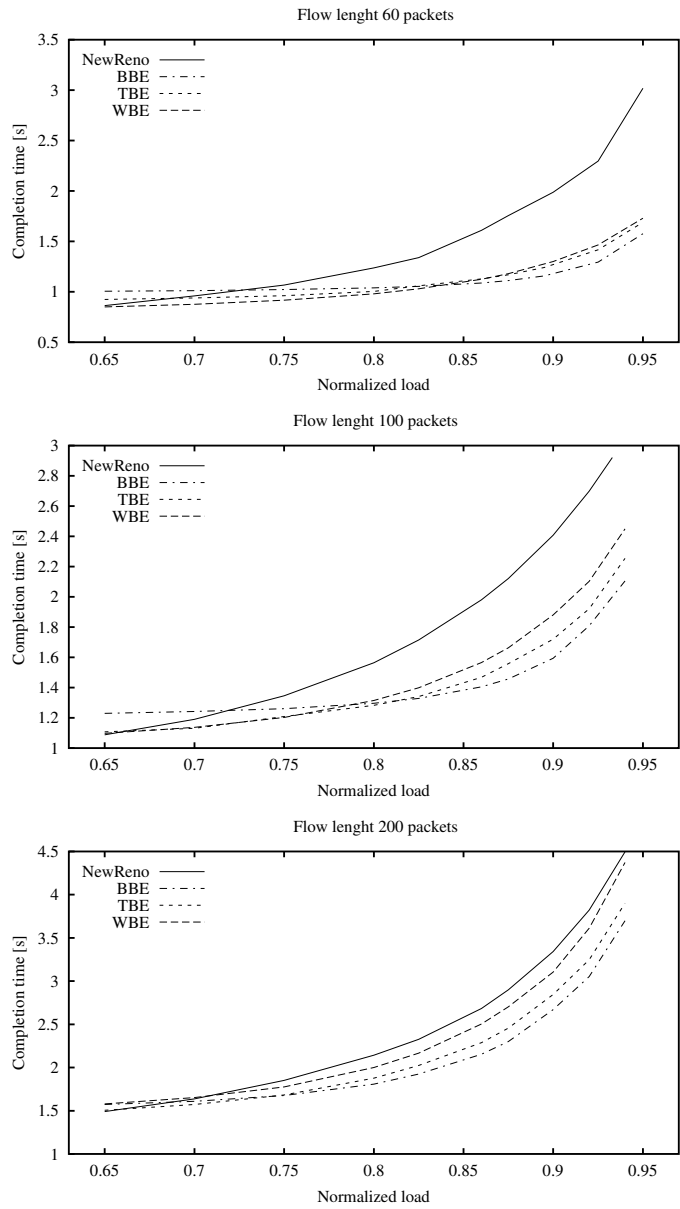
Going a little bit more into detail, we can observe that, for shorter connections and low loads, the early exit from the slow start implemented by the ESSE modification may be a little penalizing, though always marginally. Comparing the PEERING and ACCESS scenario it is clear that the larger the available resources, the higher the gain we can achieve from a correct estimation of the available resources from its use to reduce the initial transient overshoot.

## 5 Conclusions and Future Work

TCP is a reliable, robust and reasonably well performing protocol; however, it is far from perfect, and scores of modifications have been proposed to improve its performance. Some of them were successfully implemented and are now available in standard



**Fig. 6.** Average completion times in the ACCESS scenario when all modifications are implemented, but errors impair the bandwidth estimation



**Fig. 7.** Average completion times in the PEERING scenario when all modifications are implemented, but errors impair the bandwidth estimation

implementations, others did not have the same success. The best of the effort in TCP study was always devoted to heuristic modifications, and their implementation either in a simulation environment or on test beds. The work we present in this paper is a first attempt to tackle the problem from a theoretical point of view, using models that predicts the impact of the modifications before implementing them and allowing a much easier evaluation of the results, since the computing effort to obtain the results is orders of magnitude smaller than that of simulations, and testbeds can only offer results for very limited settings.

We have considered three different modifications that require changes of the sender side only, thus having potentially a minor impact if deployed in the Internet. One of the major result coming out of this work is that, in many situations, the performance of competing TCP connections is dominated by the initial transient when the threshold is not yet set. Hence, any attempt to improve TCP performance should take into account the initial transient too, while, browsing the literature, it is clear that the best part of the effort in TCP research was devoted to its steady state.

We admit that the results we have are not definitive, since we have to refine our model in at least three different directions. First of all we have to verify the behavior of the modified TCP version when competing with NewReno implementations, in order to verify their friendliness to the existing Internet. Though not trivial, this modeling step is feasible, hence we hope to show also this aspect in short time. As a “side effect” of this additional modeling effort, the model will also be able to predict the performance of any mix of different length flows. Second, since the considered modifications imply the end-to-end estimation of the available bandwidth, additional studies are required on how the estimation can be carried out and what kind of errors would presumably affect this estimation. Finally, it would be extremely interesting, though it is still not clear if it is possible, to extend the modeling technique to compute not only averages over the whole flows, but also some additional information about the actual distributions of, for instance, the completion time of the connections.

### Acknowledgments

The authors express their deepest appreciation to Michele Garetto, who provided the code for solving the TCP NewReno model, assistance in the code exegesis and a lot of useful discussions

### References

1. S. Floyd, “A Report on Recent Developments in TCP Congestion Control,” *IEEE Communications Magazine*, April 2001
2. L. Brakmo, L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” *IEEE Journal on Selected Areas in Communications*, 13(8), Oct. 1995.
3. M. Mathis, J. Madhavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgment Options”, RFC 2018, IETF, Oct. 1996.
4. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation,” *IEEE/ACM ToN*, Apr. 2000.

5. N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency," *Infocom 2000*, Tel Aviv, Israel, March 2000.
6. V. Mishra, W. B. Gong, D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with and application to RED", in *Proc. SIGCOMM'2000*, Aug. 28–Sept. 1 2000, Stockholm, Sweden.
7. M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, "A Detailed and Accurate Closed Queueing Network Model of Many Interacting TCP Flows," *IEEE Infocom 2001*, Anchorage, Alaska, USA, April 22–26, 2001.
8. S. Floyd, "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, V. 24 N. 5, pp. 10-23, Oct. 1994.
9. K. K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, IETF, Jan. 1999.
10. M. Gerla, W. Weng, R. Lo Cigno, "Bandwidth feedback control of TCP and real time sources in the Internet," *IEEE Globecom 2000*, San Francisco, CA, USA, Nov. 27 – Dec. 1 2000.
11. M. Gerla, M. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, "TCP Westwood: Window Control Using Bandwidth Estimation," *Proc. IEEE Globecom 2001*, San Antonio, Texas, USA, Nov. 25-29, 2001.
12. S.H. Low, D.E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, 7(6):861-75, Dec. 1999.
13. S.H. Low, F. Paganini, J.C. Doyle, "Internet Congestion Control", to appear on *IEEE Control Systems Magazine*, Feb. 2002.
14. S.H. Low, "A Duality Model of TCP and Queue Management Algorithms," *Proc. of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 18-20, 2000, Monterey, CA (USA).
15. C. V. Hollot, V. Mishra, W. B. Gong, D. Towsley, "A Control Theoretic Analysis of RED," *IEEE Infocom 2001*, Anchorage, Alaska, USA, April 22–26, 2001.
16. C. V. Hollot, V. Mishra, W. B. Gong, D. Towsley, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," *IEEE Infocom 2001*, Anchorage, Alaska, USA, April 22–26, 2001.
17. T. Bu, D. Towsley, "Fixed Point Approximation for TCP Behavior in an AQM Network," *ACM SIGMETRICS 2001*, June 16-20, Cambridge, MA, USA.
18. R. Lo Cigno, M. Gerla, "Modeling Window Based Congestion Control Protocols with Many Flows", *Performance Evaluation*, No. 36–37, pp. 289–306, Elsevier, Aug. 1999.
19. M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, "On the Use of Queueing Network Models to Predict the Performance of TCP Connections," *Proc. 2001 Tyrrhenian International Workshop on Digital Communications*, Taormina (CT), Italy, Sept. 17–20, 2001.
20. M. Garetto, R. Lo Cigno, M. Meo, E. Alessio, M. Ajmone Marsan, "Modeling Short-Lived TCP Connections with Open Multiclass Queueing Networks," submitted to *IEEE Infocom 2002*.
21. M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, "Queueing Network Models for the Performance Analysis of Multibottleneck IP Networks Loaded by TCP Short Lived Connections," submitted to *IEEE Infocom 2002*.
22. Y. Yang, M. Kim, S. Lam, "Transient Behaviors of TCP-friendly Congestion Control Protocols." *IEEE Infocom'01*, Anchorage, AK, USA April 2001.
23. Y. Yang, S. Lam, "General AIMD Congestion Control," Tech. Rep. TR-2000-09, University of Texas at Austin, May 2000. Available from <http://www.cs.utexas.edu/users/lam/NRL/TechReports/>.