End-to-end TCP enhancements performance on satellite links

C. Caini and R. Firrincieli

DEIS/ARCES, University of Bologna, Viale Risorgimento 2, 40136, Bologna, Italy
Email: ccaini@deis.unibo.it, rfirrincieli@arces.unibo.it, tel: +390512093062, fax: +390512095410

***Abstract*—Although TCP has proved very effective and robust for many years, its performance on emerging heterogeneous networks is challenged by the impairments originated by the presence of radio links. In particular, satellite communications are affected by long RTTs and possibly also by random segment losses, which can severely affect end-to-end performance. To cope with these problems, several TCP enhancements have been presented in the literature. The paper aim is to investigate the effectiveness of such modifications, when applied to TCP satellite connections. In particular, the analysis focuses on some emerging proposals, namely TCP Hybla, developed by the authors, and TCP Westwood, examined here in three variants. They are compared with three well established TCP variants, such as NewReno, SACK and Vegas, taking into account both a pure satellite environment and more challenging, but also perhaps closer to reality, heterogeneous network. Performance is assessed by means of ns-2 simulations considering goodput, fairness and friendliness as performance metrics. Results show that large performance improvements may be achieved by some of the considered TCP enhancements, without infringing the end-to-end semantics of this protocol.**

***Index Terms*— Transport Protocols, Satellite Communications, Heterogeneous Networks, SACK, TCP Westwood, TCP Hybla.**

I. INTRODUCTION

A transport protocol, such as TCP, has the goal of providing the upper layers with a reliable and ordered end-to-end flow of data, by hiding the imperfection of lower layers. To realize this demanding task, TCP has been improved several times, by updating some basic functions, but leaving the essential architecture unaltered. However, the recent spreading out of heterogeneous networks poses new problems, in particular when a satellite radio link is present in end-to-end TCP connections. First, the Round Trip Time (RTT) is greatly increased by the long propagation time on the satellite radio channel (for GEO, RTT ≅ 600 ms if both the forward and the return links are via satellite). Second, the presence of random losses originated by radio propagation cannot be considered negligible. Both these aspects severely penalize the transport layer performance, as shown in [1], [2], [3]. In particular, it is well recognized that the performance of standard TCP protocols (Tahoe, Reno) but also NewReno [4], SACK [5] and Vegas [6] largely depends on network delays, with significant throughput degradation for long RTT connections. This is due to the standard congestion control mechanism that increases the congestion window, cwnd, (and consequently the segment transmission rate) on an RTT time base, penalizing long RTT connections. Regarding radio channel losses, they mainly affect performance in an indirect way, because TCP ascribes all the losses to congestion (as this was the most sensible choice for wired networks, where errors were rare). As a result, in presence of channel losses the cwnd may be repeatedly halved, with dramatic consequences on satellite connections, where the long RTTs prevent a fast re-opening of the cwnd.

This work was supported in part by the IST-507052 SatNEx Network of Excellence.

The possible effective solutions to the problems posed by satellite, and in general by wireless links can be classified in two categories. In the first, the radio links are somewhat isolated from the wired network by introducing "protocol accelerators" (or PEPs, Performance Enhancing Proxies) [7]. The second category consists of end-to-end TCP enhancements that introduce some important modifications in the basic TCP congestion control algorithm or in the recovery procedures [8-13]. Although they do not fully comply with the NewReno de facto standard, by contrast to the former category they still retain both the end-to-end semantics of TCP and the layer separation principle. Among them, this paper focuses on TCP Hybla, a TCP enhancement presented by the authors in [8], and on TCP Westwood (TCPW) [9], considered here in three variants, Westwood, Westwood+ [10], [11], and E-TCPW (Enhanced TCPW) [12]. All of them have been implemented in the network simulator ns-2 [14], which has been adopted here for performance evaluation. The paper aim is to provide an extensive performance comparison, of the four cited end-to-end TCP enhancements with three well established TCP variants, such as NewReno, SACK and Vegas, considering various satellite environments. Being interested in the development of transport protocols for satellite channels, the authors have already addressed this topic [8], [12]. However, the present paper differs from previous works because it is not focused on the presentation of a new TCP variant, but aims to present a selection of results related to performance comparison. As that, new results and new simulation scenarios are presented here.

II. KNOWN TCP VARIANTS

A. TCP NewReno and TCP SACK

TCP NewReno [4] is the most widely adopted TCP version and is considered here as a reference. When a new TCP connection is established, the sender probes for bandwidth

availability by gradually increasing the congestion window (cwnd). In the Slow Start (SS) phase, starting from an initial value typically equal to one or two maximum segment size (MSS), the cwnd is increased by *MSS* bytes per every non-duplicate received ACK. When the congestion window reaches the slow start threshold (ssthresh), the source switches to the Congestion Avoidance (CA) phase, during which the cwnd increment is reduced to $MSS / cwnd$ bytes per every new received ACK. In short, by denoting with W the value of the congestion window in MSS units, we have

$$W_{i+1} = \begin{cases} W_i + 1 & SS \\ W_i + 1/W_i & CA \end{cases} \quad (1)$$

where the index i denotes the reception of the i^{th} ACK.

In a real channel the cwnd rise continues until either the size of receiver buffer (advertised window) is reached, or a segment is lost, triggering the Fast Retransmit and the Fast Recovery procedures. It is worth noting that in presence of multiple losses, the last procedure is inefficient, as it allows recovering only one packet per RTT. To overcome this problem, the selective acknowledgment (SACK) option was proposed [5]. By means of it, the sender can recover more than one packet per RTT. This ability is extremely useful when dealing with large cwnds (where multiple losses are frequent) and long RTTs, i.e. in satellite channel.

B. TCP Vegas

TCP Vegas introduces an alternative congestion control policy, with the aim of preventing congestion losses by means of a dynamic estimate of the available bandwidth [6]. This TCP variant tries to detect changes in the sending rate by comparing the measured throughput rate with the expected one. First, it estimates *BaseRTT* as the RTT of a segment when the connection is not congested, in practice, the minimum of all measured round trip times. As a result, the *expected* throughput is given by *cwnd* / *BaseRTT*. Second, TCP Vegas calculates the current *Actual* sending rate by recording how many bytes are transmitted between the time that a segment is sent and its acknowledgment is received, computing the RTT and dividing the number of bytes transmitted by the sampled RTT. This calculation is done once per round-trip time. Finally, TCP Vegas compares *Actual* to *Expected*, and adjusts the window accordingly. TCP Vegas also presents a number of minor modifications with respect to TCP NewReno, included the use of timestamps [15] to avoid spurious timeouts.

III. TCP HYBLA

TCP Hybla [8] was conceived with the primary aim of counteracting the performance deterioration caused by the long RTTs typical of satellite connections. It consists of a set of procedures, which includes an enhancement of the standard congestion control laws, the mandatory adoption of the SACK policy, the use of timestamps, the adoption of Hoe's channel bandwidth estimate [16], and the implementation of packet spacing techniques [17].

The modification of the standard congestion control rules is dictated by the TCP Hybla ideal aim of obtaining for long RTT connections the same instantaneous segment transmission rate as a comparatively fast reference TCP connection (e.g. a wired one). To this end, it is necessary to speed up the cwnd increase of the long RTT connection because the longer the RTT, the larger the cwnd required (the segment transmission rate being given by cwnd/RTT). To this end, a normalized round trip time, ρ, was introduced as the ratio between the actual RTT and the round trip time of the reference connection to which TCP Hybla aims to equalize the performance, denoted by RTT_0, (see [8] for a detailed discussion on the criteria for the RTT_0 choice),

$$\rho = RTT / RTT_0 \quad (2)$$

In the references it is shown, through a theoretical proof, that the same segment transmission rate of the reference connection can be achieved by longer RTT connections by replacing the standard congestion control rules with the following,

$$W_{i+1} = \begin{cases} W_i + 2^{\rho} - 1 & SS \\ W_i + \rho^2 / W_i & CA \end{cases} \quad (3)$$

It is worth noting that the CA rule reported in (3) is actually applied until the cwnd value reaches the estimated bandwidth delay product (BDP); whenever the cwnd is equal or greater than the estimated BDP, the standard CA rule described in (1) is followed. This feature has been recently introduced in order to prevent the algorithm from oscillating around the channel capacity.

As already mentioned, apart from the modification of the congestion control rules, TCP Hybla includes other several important enhancements. The rationale for their introduction is the following. SACK policy is mandatory because thanks to the improved congestion control algorithm of TCP Hybla, a larger average cwnd has to be expected for long RTT connections. As a result, multiple losses in the same window will occur more frequently. Large cwnds also frequently cause severe inefficiencies of the "exponential back-off" RTO policy. These can be avoided by resorting to timestamps [15]. Bandwidth estimate [16] is used in order to appropriately set the initial ssthresh. If the initial ssthresh is too low, compared to BDP, TCP prematurely switches to the CA phase, leading to inefficient bandwidth utilization. Vice versa, if the initial ssthresh is too high, the exponential cwnd increase of the SS phase may cause huge overflows when the capacity of the channel is reached. Packet spacing [17] has been adopted to counteract the burstiness due to large cwnds. Finally, the use of the recovery fix option [18] implies that at the end of the recovery phase the cwnd is set to the minimum of ssthresh and the actual number of packets in fly. This prevents the transmission of burst of data caused by the sliding of the cwnd at the end of the Fast Recovery.

IV. TCP WESTWOOD

A. TCP Westwood

TCPW [9] aims to limit the consequences of the losses

introduced by a wireless channel, by introducing a modification of the Fast Recovery algorithm called Faster Recovery. Instead of halving the congestion window after three duplicate ACKs, and fixing the slow start threshold to this value, TCPW sets the ssthresh as a function of the actual available bandwidth, so avoiding the dramatic slow-down in the transmission rate of the TCP standard versions. The bandwidth is estimated by averaging the rate of returning ACKs. In particular, the reception of an ACK at the time t_k implies that an amount of data d_k has been received. Therefore, the k-th sample of bandwidth used by a given connection is measured as:

$$b_k = \frac{d_k}{t_k - t_{k-1}} \tag{4}$$

where t_{k-1} is the arrival time of the previous ACK. These values are then filtered to obtain the available bandwidth estimate (BWE), $\hat{b}$. After loss detection, TCP Westwood sets the ssthresh and cwnd as follows:

$$sstresh = \hat{b} * RTT_{\min}$$
$$cwnd = \begin{cases} sstresh & \text{if } cwnd > sstresh \\ 1 & \text{after a time out} \end{cases} \tag{5}$$

Several variants of the basic algorithm have been presented in the literature [19], [20], to improve performance. In the numerical results section, data labeled TCPW are obtained by making use of the ns-2 release downloadable from the official TCPW web site [21].

B. TCP Westwood+

By replacing the filter used for the bandwidth estimate with an improved one, some of the authors of TCPW developed a new version of the algorithm, called TCP Westwood+ (TCPW+) [10], [11]. Numerical results for TCPW+ are obtained by exploiting the ns-2 release downloadable from the TCPW+ web site [22].

C. TCP Enhanced Westwood

With the exception of the congestion window rules, the other features included in TCP Hybla are potentially compatible with TCP Westwood. In practice, however, it is likely that packet spacing hinders the TCPW bandwidth estimation mechanism. Therefore, the authors identified in the remaining three features, namely, the SACK option, the initial ssthresh estimation and the recovery fix, three possible enhancements that could be straightforwardly and advantageously applied to TCPW, obtaining the Enhanced TCPW (ETCPW) [12]. To this end they developed a new ns-2 module by merging the TCP Sack1 and the TCP WestwoodNR agents, both already present in ns-2.

V. Topology and Simulation Setup

The network topology considered is shown in Fig. 1. The satellite TCP connections are composed of both wired legs and a satellite link, while the background traffic is represented by entirely wired paths. All the connections share the R1-R2 bottleneck link, whose bandwidth has been deliberately limited to 10 Mbit/s in order to study the congestion effects. The router R1, where all the congestion events are confined, follows either a DT (Drop Tail) or an Adaptive RED (Random Early Detection) policy, both with default parameters; all the other hosts follow a DT policy. The two-way propagation delay of the satellite links varies in such a way that the RTT of the satellite connections ranges from 25 ms (considered only for comparison purposes with the wired connections) to 600 ms (GEO satellite link). The wired links are supposed to be error free, while for the satellite link a uniformly distributed error model and a 2-state Gilbert-Elliot channel model [23] have been considered. The MSS is always 1500 bytes and the senders access the 10Mbit/s bottleneck through a 100 Mbit/s access link. The performance is evaluated either in terms of received packets time sequence or goodput (i.e. the amount of packets correctly received divided by the transfer process time). In order to prevent the transmission bit rate from being limited by the advertised window instead of the cwnd, the advertised window of the satellite receivers has been appropriately increased. This is necessary to grant fair conditions to all TCP flavors. Finally, the following parameter set-up has been used for tested TCP variants: RTT_0 = 25 ms for Hybla (default); ns-2 default parameters have been retained for Vegas ($\alpha = 1$, $\beta = 3$ and $\gamma = 1$) and all Westwood variants [21], [22].

VI. Numerical Results

We start our comparative performance evaluation by considering first an ideal GEO satellite channel. In this case, the analysis can be carried out in a deterministic way (to this end a deterministic DT policy in R1 has been adopted, instead of the random RED) by simply evaluating the performance of a single connection. Then, the study is extended to include the effects of a residual packet error rate and/or congestion on heterogeneous environments. In this case, performance is more conveniently evaluated in terms of goodput averaged over a series of independent persistent file transfers and RED is adopted on R1 to avoid the synchronization effect induced by DT.

A. Ideal channel

1) Start-up performance

We consider the presence of a single TCP connection on a 10 Mbit/s GEO satellite channel, in absence of congestion and link losses. This is the most suitable scenario to evaluate the ability of the investigated TCP flavors to take full advantage of a large satellite bandwidth under the most favorable conditions. To this end, in Fig.2 is plotted the number of received TCP segments vs. the time elapsed from the connection establishment for all the TCP flavors previously considered. For a better comprehension of results, on the same figure is plotted the “max-speed” straight line, which can be considered as the upper bound corresponding to a full utilization of the satellite bandwidth. The slope of performance curves should be compared with this straight line. The higher the slope the faster the transmission rate (when a line appear roughly parallel to the upper bound, the corresponding TCP

connection speed has reached its maximum). It can be observed that Hybla is the fastest in reaching full speed, followed by E-TCPW, TCPW+, Vegas, SACK and NewReno. The good performance of Hybla is due to its inherent capacity to fast open the cwnd even in the presence of a long RTT (600 ms in this case). This leads to a full utilization of the channel capacity from the very beginning of the connection. As other TCP variants have not specific countermeasures against long RTT, they need much more time to open the cwnd and reach the channel capacity. Moreover, they suffer from an initial buffer overflow, which takes some time to be recovered. The exception is E-TCPW, which thanks to the initial ssthresh estimation, the recovery fix and the SACK option, is quite fast to enter the steady-state. TCPW comes last, but a further investigation showed that its performance is severely affected, in this particular case, by the bandwidth reduction on the bottleneck (from 100Mbit/s to 10 Mbit/s), in absence of which it would have obtained much better performance. The problem should be ascribed to frequent overestimates of BWE, which causes repeated R1 buffer overflows. It is highly likely that the most recent version of Westwood has this problem fixed.

2) UDP spike

To assess the "agility" of different TCP versions in reacting to abrupt changes of the traffic load, an on-off concurrent 50s UDP traffic spike is introduced in the previous environment. Although the 5Mbit/s UDP flow occupies only a half of the available bandwidth, its abrupt rise induces a significant perturbation on the active TCP connections for all its length. The time to recover from the perturbation is however largely variable depending on the TCP flavor. Even in this case, Hybla and E-TCPW outperforms the other protocols (Fig.3). Here, again, the reason is that Hybla is very fast to re-open the cwnd after the UDP spike, while E-TCPW benefits largely from its improvements, although it needs more time than Hybla to react to the UDP interference. It is worth noting that Vegas seems to stall after the UDP spike. The reasons for this unexpected behavior may deserve further future investigation.

B. Performance in presence of residual packet error rate

As TCP does not distinguish the origin of packet losses, link errors cause spurious interference on the congestion control mechanism, causing detrimental cwnd reductions. The consequences may become severe in presence of long RTTs, because of the longer time required to re-open the cwnd. This is highlighted in Fig.4, where the goodput of a satellite connection with a uniformly distributed 1% PER, and no congestion, is reported. In particular, we can observe that for short RTTs both TCP Hybla and TCPW provide basically the same quite satisfactory performance (although through different TCP enhancements), while TCP Vegas, TCP SACK and NewReno show their limits. Increasing the RTT, the performance further (and faster) degrades except for TCP Hybla and E-TCPW.

The analysis continues by evaluating the impact of PER on performance, considering a GEO satellite connection, i.e. the worst case, as far as the RTT is concerned. Results, reported in Fig.5 show a dramatic performance reduction even for limited PER values. It is worth noting that because of the long RTT of GEO satellite connections, TCPW and TCPW+ countermeasures against random losses, usually really effective on shorter RTT channels, show only a limited impact. This is largely due to the absence of the SACK policy, which is of paramount importance for a fast recover of multiple losses, more frequent when dealing with large cwnds, as it is the case in long RTT connections. By contrast, both Hybla and E-TCPW appear more effective in counteracting the presence of high PER values. As well as to the presence of the SACK policy, this is also due to the adoption of the recovery fix, which prevents losses caused by the transmission of segment bursts at the end of the recovery phase.

Finally, the dependence on random error distribution is analyzed in Fig.6, by comparing results achieved adopting the uniform distribution model with those obtained by means of the aforementioned two state model. In a RTT=600 ms and PER=1% channel, goodput results generally improve with the average burst length (which is roughly 1 for the uniform distribution). This can be easily explained by considering that a burst of losses usually determines only a single cwnd reduction, being multiple reductions inhibited during the recovery phase. Therefore, for a fixed PER value, the impact of losses is greater if they are distributed (at least as long as they do not determine time outs). As a result, we can conclude that the widely adopted uniform distributed model, although not very close to reality, has the advantage of being conservative (for a fixed PER), as well as very simple.

C. Performance in heterogeneous networks

Previous scenarios do not take into account the severe penalization that long RTT satellite connections encounter whenever they have to compete, on the wired Internet, with short RTT fully wired connections, a situation destined to become always more frequent with the increasing pervasiveness of heterogeneous networks. Therefore, to study the effects of congestion in such a challenging environment, we added 5 background wired connections (RTT=25 ms, PER=0%, SACK) to a single foreground satellite connection (PER=0%). The goodput of the satellite connection is then reported vs. its RTT in Fig.7. The goodput of background connections is not reported for clarity, being also very close to the maximum fair share (i.e. the capacity of the bottleneck link divided by the number of sharing connections). The impact of competing wired connection on performance is apparent for all TCP variants, but TCP Hybla and E-TCPW, which achieve a remarkable independence of the RTT. While this limited dependence on the RTT is coherent with TCP Hybla primary aim, it can be explained for E-TCPW by supposing that the quite aggressive bandwidth estimate leads the TCP to often operate in slow start, compensating in this way the RTT penalization. In absence of it, i.e. for low RTT values, this BWE overestimation causes E-TCPW to overcome the maximum fair share.

Finally, it is worth noting that for RTT=25 ms (the RTT of

terrestrial connections) all tested TCP variants achieve a goodput very close to the maximum fair share, that is to say that they do not behave too aggressively. This aspect, further analyzed in the next subsection, is an important aspect for all TCP variants and especially for Hybla whose aim is to behave as the TCP standard when considering the same RTT conditions.

D. Fairness and friendliness

Fairness and friendliness properties deserve a careful evaluation. Fairness refers to the capacity to assure a fair band subdivision among competing connections that use the same version of the protocol, while friendliness indicates the same ability with reference to different protocol variants. We decided to study them by evaluating the share of R1-R2 bottleneck available bandwidth when three GEO satellite connections (variable TCP flavor, RTT=600 ms, PER=0%) have to compete with three wired connections (SACK, RTT=25 ms, PER=0%). Results are reported in Table I, where each column label denotes the TCP flavor adopted on the three satellite connections. From the first column, we can observe that SACK fairness is limited to connections that have the same RTT; otherwise it becomes extremely unfair against longer RTT connections (e.g. satellite). This severe penalization is maintained basically unresolved by adopting on satellite connections Vegas, TCPW and TCPW+, while both E-TCPW and Hybla are effectively able to reduce it at the expense of a limited increase in the amount of unused bandwidth. Note that E-TCPW seems a bit too aggressive, while the contrary holds for Hybla. Again, this behavior may be ascribed to the perfectible BWE of E-TCPW, which tends to overestimate the available bandwidth.

VII. Conclusions

The paper presents an extensive comparative evaluation of different end-to-end TCP enhancements, carried out through ns-2 simulations. From the many results presented we can conclude that in satellite networks, only TCP enhancements specifically designed to cope with long delays and/or not negligible losses are actually able to provide satisfactory performance.

References

[1] Y. Hu and V. Li, "Satellite-based internet: a tutorial", IEEE Commun. Mag., pp. 154-62, March. 2001.

[2] C. Barakat, E. Altman, and W. Dabbous, "On TCP performance in a heterogeneous network: a survey", IEEE Commun. Mag., pp. 40-46. Jan. 2000.

[3] M.Allman et alii, "Ongoing TCP Research Related to Satellites", Request for Comment RFC 2760, IETF, Feb. 2000.

[4] M. Allman, W. Stevens, "TCP congestion control", IETF RFC 2581, Apr. 1999.

[5] M. Mathis and J. Mahdavi, "TCP Selective Acknowledgment Options", Request for Comment 2018, IETF, Oct. 1996.

[6] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet", IEEE J. Select. Areas Commun., vol. 13, pp. 1465-1480, Oct. 1995.

[7] J. Border M. Kojo J. Griner Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", Request for Comment RFC 3135, IETF, June 2001.

[8] C. Caini, R. Firrincieli, "TCP Hybla: a TCP Enhancement for Heterogeneous Networks", Int. J. Satell. Commun. Network, vol. 22, pp.547-566, Sep.-Oct. 2004.

[9] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: end-to-end congestion control for wired/wireless networks", Wireless Networks, Kluwer Academic Publisher, vol.8, pp. 467-479, 2002.

[10] L.A. Grieco and S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control", ACM SIGCOMM Computer Commun. Review vol. 34, pp.25-38, April 2004,

[11] L.A. Grieco and S. Mascolo, "Mathematical analysis of Westwood+ TCP congestion control", IEE Proc.-Control Theory and Applications, - vol. 152, pp.35-42, Jan. 2005

[12] C.Caini and R.Firrincieli, "Further Improving TCP Performance on Satellite Channel", in Proc. ISWCS05, pp.2906-2910, Sept.2005

[13] I.F., Akyildiz, G. Morabito, S. Palazzo, "TCP Peach: a new congestion control scheme for satellite IP networks", IEEE/ACM Transactions on Networking, Vol. 9, No. 3, pp. 307-321, Jun. 2001.

[14] http://www.isi.edu/nsnam/ns/

[15] V. Paxon and M. Allman, "Computing TCP's Retransmission Timer", IETF RFC 2988, Nov 2000.

[16] J. C. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", in Proc.of ACM SIGCOMM '96, pp 270-280, 1996.

[17] C.Caini and R.Firrincieli, "Packet spreading techniques to avoid bursty traffic in satellite TCP connections", in Proc. of IEEE 59th Veh. Technol. Conf., VTC2004-Spring, Milan, Italy, May 2004, pp.2906-2910

[18] S. Floyd and T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, IETF, Apr. 2004.

[19] M. Gerla, B. K. F. Ng, M. Y. Sanadidi, M. Valla, R. Wang "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs", J. of Computer Commun., Vol. 27, Jan. 2004.

[20] R. Wang, G Pau, K. Yamada, M. Y. Sanadidi, M. Gerla, "TCP Startup Performance in Large Bandwidth Delay Networks ", in Proc. Of INFOCOM 2004, Hong Kong, March 2004, pp.796-805.

[21] http://www.cs.ucla.edu/NRL/hpi/tcpw/.

[22] http://193.204.59.68/mascolo/tcp%20westwood/tcpwestwood.htm

[23] E. N. Gilbert, "Capacity of a Burst-Noise Channel", Bell Sys. Tech. J., vol. 39, Sept. 1960, pp. 1253-1266

TABLE I
FAIRNESS AND FRIENDLINESS (6 COMPETING CONNECTIONS ON R1-R2 LINK)

	SACK	VEGAS	TCPW/ TCPW+	E-TCPW	Hybla
Sat. RTT=600ms	1.9%	2.8%	3.6%	16%	13.3%
Sat. RTT=600ms	1.9%	2.8%	3.6%	15.9%	13.5%
Sat. RTT=600ms	1.9%	2.9	3.6%	16.2%	13.6%
SACK RTT=25ms	31.3%	30.4%	29.4%	14.5%	17.5%
SACK RTT=25ms	31.5%	30.4%	29.4%	14.7%	17.4%
SACK RTT=25ms	31.0%	30.3%	29.4%	14.8%	17.5%
Unused bwidth	0.3%	0.4%	1.0%	7.9%	7.2%

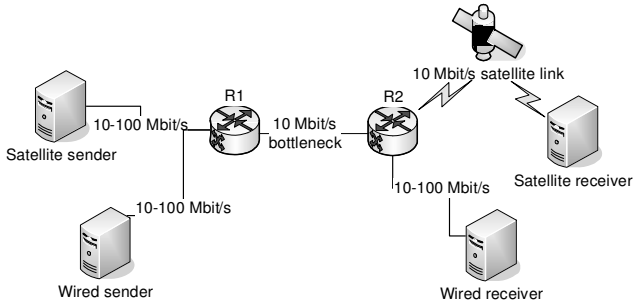

Fig. 1. Simulation topology.

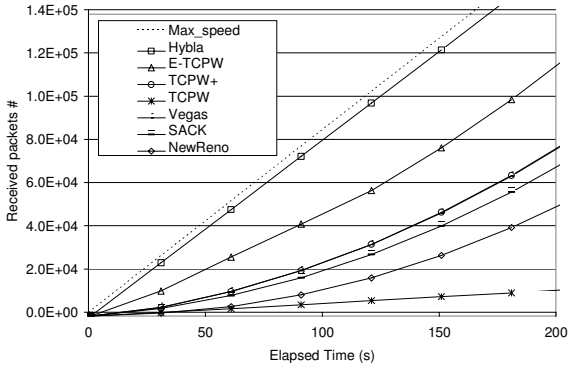

Fig. 2. Performance of various TCP flavors at start-up. Ideal 10 Mbit/s GEO satellite channel (no congestion, PER=0%, RTT=600ms).

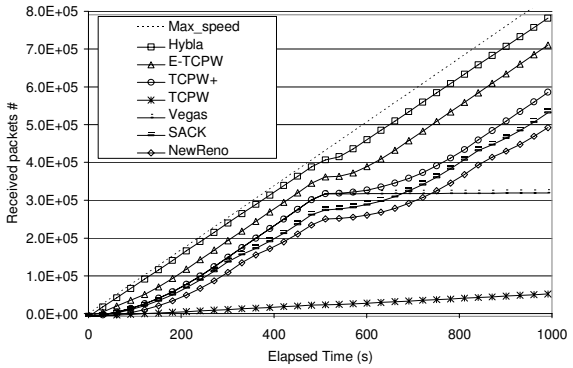

Fig. 3. Performance of various TCP flavors at start-up, in presence of a UDP spike (5 Mbit/s, active for 50s, starting at 500s). Ideal 10 Mbit/s GEO satellite channel (no congestion, PER=0%, RTT=600ms).

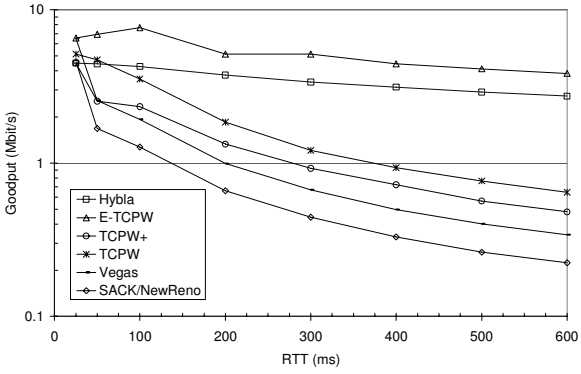

Fig. 4. Performance of various TCP flavors vs. RTT, in presence of a residual packet error rate on the satellite channel (PER=1%); no congestion.

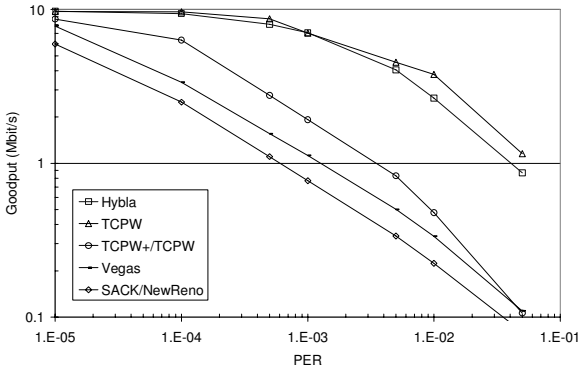

Fig. 5. Performance of various TCP flavors vs. PER on a GEO satellite channel (no congestion, RTT=600ms).

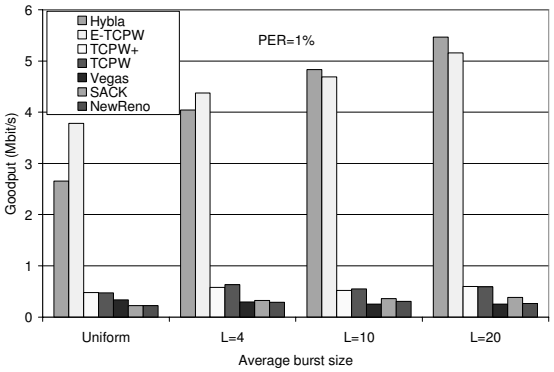

Fig. 6. Impact of random losses distribution on various TCP flavors performance; GEO satellite channel with PER=1% (no congestion, RTT=600ms).

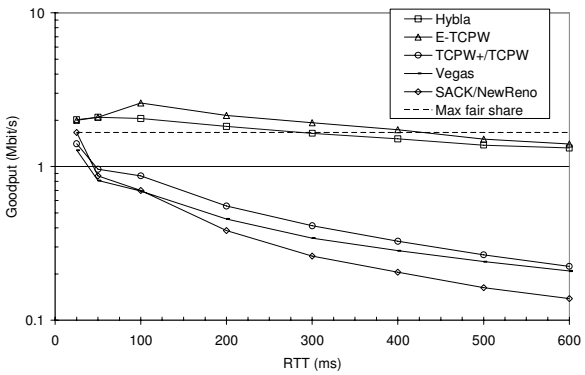

Fig. 7. Heterogeneous networks: performance of the foreground satellite connection in presence of R1-R2 congestion due to 5 background competing terrestrial connections (SACK, RTT=25 ms); PER=0%.