PEPsal: a Performance Enhancing Proxy for TCP satellite connections

Carlo Caini, Rosario Firrincieli, Daniele Lacamera
DEIS/ARCES, University of Bologna, Italy
ccaini@deis.unibo.it, rfirrincieli@deis.unibo.it, root@danielinux.net

***Abstract*[1] — Internet communications with paths that include satellite link face some peculiar challenges, due to the presence of a long propagation wireless channel. In this paper, we propose a Performance Enhancing Proxy (PEP) solution, called PEPsal, which is, to the best of the authors' knowledge, the first open source TCP splitting solution for the GNU/Linux operating systems. PEPsal improves the performance of a TCP connection over a satellite channel making use of the TCP Hybla, a TCP enhancement for satellite networks developed by the authors. The objective of the paper is to present and evaluate the PEPsal architecture, by comparing it with end to end TCP variants (NewReno, SACK, Hybla), considering both performance and reliability issues. Performance is evaluated by making use of a testbed set up at the University of Bologna, to study advanced transport protocols and architectures for internet satellite communications. At present PEPsal is adopted, with success, by a satellite Internet provider.**

I. INTRODUCTION

Internet communications with paths that include satellite link face some peculiar challenges, due to the presence of long propagation wireless channel. First, the presence of losses not originated by congestion cannot be considered negligible. Second, the Round Trip Time (RTT) is greatly increased by the long propagation time on the satellite radio channel (a GEO bi-directional RTT can exceed 600 ms). Both these aspects pose severe challenges to the transport layer performance when an application requires reliable service [1], [2]. To cope with these issues, several TCP enhancements have been proposed in the past years. They could be generally classified in two main categories: TCP enhancements that involve only the end hosts [3], [4], and TCP enhancements that envisage intermediate agents on the path usually called Performance Enhancing Proxies (PEPs) [5]. Among the PEP proposals, the most promising solution is represented by the splitting approach. The rationale of the splitting concept is to separate the satellite portion from the rest of the network. To this end, two alternative architectures can be adopted, namely integrated and distributed PEP [5]. In the first case, the TCP connection established among the end hosts is split in two separated connections, with a sole PEP agent in the middle. The first connection makes use of the TCP standard and is terminated on the PEP. The second connection,

[1] A preliminary version of this paper will be included in the Proc. of IEEE VTC'06 spring, Melbourne, May 2006. This work was supported in part by the IST-507052 SatNEx II Network of Excellence.

between PEP and the final user, can exploit an enhanced TCP version compatible with a standard TCP receiver. By contrast, the distributed architecture splits the connection in three sections, by isolating the satellite link between two PEP agents. In this case, it is possible to adopt a proper transport protocol on the satellite link, such as SCPS [6], while the end hosts continue to adopt the TCP standard protocol. In a backbone application the distributed approach can be preferable. However, when the satellite link represents the last hop for Internet access, the integrated approach presents the advantage of not requiring at the end user any non standard (i.e. provider dependent), additional hardware or software modification to the receiver box. For this reason, in this paper the attention is focused on the integrated architectures. As it will be shown in the paper, integrated PEPs based on TCP splitting offer higher performance because they are able to confine the satellite impairments (long RTTs and random losses) on the second component of the split connection, where they can be counteracted by specific optimized TCPs.

Despite the good performance provided, there are some important issues related to the splitting approach. First, splitting violates the basic end to end TCP semantics. The sender receives acknowledgments from an intermediate agent instead of the final receiver. Second, the intermediate agent needs to access the TCP header of the packets in order to send ACKs back to the sender and perform all its optimization procedures. This prevents the use of IPSEC technique which encrypts the IP payload making the TCP header not available. Thus, an IPSEC flow can not be managed by splitting PEP and performance can be improved only by a non splitting PEP or an end to end TCP enhancement.

Although many proposals based on the splitting approach (either integrated or distributed) have been presented in the literature, some of them lack any OS implementation and consequently it is not possible to evaluate their performance on a testbed [7], [8]. Other proposals are not available for the GNU/Linux operating systems [9], [10], [11], or, being commercial implementations [12], [13], [14], are proprietary solutions and neither implementation details, nor the source software, are available. An open source distributed PEP based on the SCPS suite has been released by JPL [6]. This will not be considered here, having focused our attention on the integrated approach.

In this paper, we propose a new PEP solution, called PEPsal, which is, to the best of the authors' knowledge, the first integrated splitting solution available for Linux OS under the GNU GPL license [15]. PEPsal improves the performance of a TCP connection over a satellite channel making use of the TCP Hybla, a TCP enhancement for satellite network developed by the authors [4]. After describing the PEPsal architecture, the paper focuses on the performance comparison either with end to end TCP variants (namely, NewReno, SACK and Hybla), and a "pure" splitting architecture, which

does not exploit any advanced TCP protocol on the satellite segment. Performance is evaluated by making use of a testbed set up at the University of Bologna, in the framework of the European SatNEx II project [16].

II. THE PEPSAL ARCHITECTURE

A. PEPsal classification

The term Performance Enhancing Proxy, is used in the literature to indicate a wide set of quite different solutions. Before examining in details the PEPsal architecture is therefore useful to anticipate its classification according to [5], which takes into account the following characteristics: layering, distribution, symmetry and transparency. As far as layering is concerned, PEPsal can be considered as a multi layer proxy, because in order to implement the TCP splitting, which is of course a transport mechanism, it must also operate at IP and Application layers. Considering distribution, PEPsal can be classified as an integrated PEP, since it runs only on a single box on the forward link satellite gateway. PEPsal can be either asymmetric or symmetric, depending on its network layer configuration: i.e. it can act in the forward direction (usual configuration), but also in the return one (in this case, with proper modifications on the receiver side). Finally, PEPsal is transparent in the customary asymmetric configuration. Since modifications are not required in both the connection endpoints, TCP users are unaware of the connection splitting performed at the satellite gateway.

B. PEPsal description

The TCP integrated splitting architecture that derives from the application of the PEPsal software on the satellite gateway is reported in Figure 1. PEPsal operates at three different layers (IP, TCP, and Application); hence it can be well described by analyzing the working scheme of each layer. At the network layer, PEPsal uses “netfilter” [17] to intercept the connections that involve the satellite link (it “steals” the TCP SYN packet in the three-way handshake phase of a TCP connection). Then, it works at transport layer, pretending to be the opposite side of the TCP connection for each of the two endpoints involved. It acts as the TCP receiver with the source, by acknowledging the incoming packets, while at the same time it sets up a new TCP connection towards the real endpoint receiver. It is important to stress that on this second connection an enhanced TCP variant can be used. Finally, to exchange data between the two connections, it is necessary to make use of an application that directly copies data between the two sockets.

After describing the basic PEPsal architecture, it is convenient to move the attention on the mechanisms that allow a real performance improvement. First of all, we would like to highlight the fact, not well recognized in the literature, that an important improvement is achieved by simply splitting the connection. In this way, the impairments introduced by the

satellite leg (long RTTs and random losses) are removed from the first connection, which becomes short-RTT and basically error free. A path with short RTT and no errors makes the connection speed grow fast, for two reasons [4]. First, because the TCP congestion control algorithm is fundamentally driven by the ACK flow, a short RTT is a requisite for a fast opening of the congestion window (cwnd), i.e. for a fast increase of the transmission rate. A fast opening of the cwnd is also a requisite in order not to be penalized in the share of bandwidth resources, whenever different connections have to compete for a bottleneck bandwidth. Second, the spurious reductions of the transmission rates, caused by random losses, are avoided. To understand this point, it may be useful, for the unfamiliar reader, remember that as TCP standard ascribes any loss to congestion, it reacts to any loss with a halving of the cwnd, i.e. with a halving of the transmission rate. In presence of random losses (i.e. losses due to a bad channel) this results in a severe interference of the recovery mechanisms on the congestion control, which TCP standard is unable to cope with.

Although useful, the splitting technique itself is not able to counteract random losses and long delay on the satellite segment. To cope with them, it is necessary to make use a transport protocol designed to deal with these problems. While distributed architectures can make use of transport protocols different form TCP (at the expenses of the introduction of an additional PEP element on the receiver side), integrated architectures, as PEPsal, must still rely on TCP. Instead of making use of TCP standard (NewReno, SACK), they can benefit the advantages provided by advanced TCP versions, provided that they are fully compatible with a standard TCP receiver. This is the case, for instance, of TCP Westwood, TCP Hybla and others. In particular, TCP Hybla, which is the TCP variant adopted by the PEPsal architecture, was conceived by the authors 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 algorithms, the mandatory adoption of the SACK policy, the use of timestamps, the adoption of Hoe's channel bandwidth estimate and the implementation of packet spacing techniques. For a complete description of this TCP variant the interested reader is referred to [4].

C. PEPsal implementation

All the software components used to implement the PEPsal architecture are reported in Figure 2, which also outlines its functioning. The bottom area ("Linux") represents the Linux kernel, which is accessed to set up the netfilter targets and to make use of a modified protocol for all TCP connections. Incoming TCP segments are mangled, so the TCP SYN segment is passed to the PEPsal user application via the ipqueue library, while the rest of the packets containing segments for that connection are redirected to local TCP port 5000. The small area on the right hand side ("Libs") shows which

system libraries have been used to implement PEPsal. Shared memory is a fast and powerful IPC method, used by the PEPsal processes to share information about incoming and outgoing connections, their state and their original endpoints. A bitmap array index is used by the application to give faster access to this memory zone. Ipqueue library is commonly used to pass a whole IP packet, including network and transport headers, to user space applications. PEPsal uses it to read information from incoming TCP SYN packets, which contain no useful data except for the headers themselves, and would be normally forwarded by the gateway to their destinations to initiate new connections.

The top area ("PEPsal") represents the user space application itself, which is completely written in C using the two libraries described above. One process, the "queuer", constantly waits for data coming from netfilter, by blocking on the ipqueue read routine. When Linux netfilter reads incoming TCP SYN segments and copies them into a queue, the queuer annotates the information (IP addresses and TCP ports) on the two endpoints in a known zone of the shared memory. Then, the SYN packet is released and continues its path through the netfilter chain. Just after that, the SYN packet, as well as all every subsequent packet containing a segment of that connection, is redirected by netfilter to TCP port 5000, where a TCP daemon, the "connection manager" is listening for it. Another process, the "proxy server", accepts the connection and searches in the shared memory for the instance matching the source address and the TCP port of the host that has started the connection. Once the destination IP address and port have been found in the connection array, a new TCP connection is attempted towards the real destination. After establishing the two connections, the proxy starts reading from one TCP socket and writing all the data in the other one. When one of the two connections ends, its twin socket is closed and its memory zone is released.

III. TESTBED DESCRIPTION

The topology of testbed used for performance evaluation is shown in Figure 3. It consists of several PCs running the Linux operating system fully controlled through a web interface specifically designed by the authors in order to speed up the execution of the tests and to enable a fast collection of execution logs [18]. Sources and sinks, as well as the router R2, where the PEPsal software is mounted, have been patched with the Multi-TCP package [19], which allows the user to easily select at run time the TCP variant to be used, as well as a fine tuning of a wide variety of TCP parameters. As the testbed aims at reproducing a heterogeneous network, both satellite and terrestrial TCP connections are present. Satellite connections are composed of both wired legs and a satellite link (emulated by dedicated PC running NistNet software [20]), while TCP background traffic is present only in the entirely wired paths. All the connections share the R1-R2 bottleneck link, whose bandwidth has been deliberately limited to 10 Mbps in order to study the congestion effects. All

other links are set to 100 Mbps, but the satellite channel that is set to 10 Mbps. The router R1, where all the congestion events are confined, follows a RED (Random Early Detection) policy (qlen=50 seg., maxth=15 seg., and minth=5 seg.); all the other hosts follow a DT policy. The RTT is set up to 25 ms on the wired network segment, while the two-way propagation delay of the satellite link varies in such a way that the RTT of the satellite connections ranges from 50 ms (mainly considered for comparison purposes) to 600 ms (corresponding to the case of a forward and return GEO satellite link). The wired links are supposed to be error free, while uniformly distributed random errors can be introduced on the satellite link, with a variable Packet Error Rate (PER). The Maximum Segment Size (MSS) is 1448 bytes and the senders access the 10 Mbps bottleneck through 100 Mbps access links. For every TCP connection, a persistent FTP file transfer process is considered. The performance is evaluated in terms of 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 receiver has been appropriately increased. This is necessary to grant fair conditions to the different TCP flavors. Finally, in TCP Hybla, the parameter RTT_0 [4] is chosen equal to the RTT of the wired connections for comparison purposes.

IV. PERFORMANCE EVALUATION

This section presents a numerical evaluation of the proposed architecture considering a variety of different environments. PEPsal performance is compared with end to end NewReno, SACK and Hybla. Moreover, in addition to the preliminary results presented in [21], we expanded this numerical evaluation to a “pure” TCP splitting architecture, by adopting TCP SACK, instead of TCP Hybla, on the satellite segment. In this way, it is possible to independently evaluate the different contribution of TCP splitting and of TCP Hybla.

A. Performance in presence of congestion

Standard TCP congestion control is fair in the allocation of bandwidth resources as long as competing connections have comparable RTTs. This is not the case of heterogeneous networks, where satellite connections are severely penalized by the presence of competing wired connections. Figure 4 shows the performance of a satellite connection, supposed error free (PER=0%), in presence of 5 wired connections on the R1-R2 bottleneck. As expected, standard TCP performance (e.g. NewReno, SACK) shows a fast a severe degradation with increasing RTT, with totally unsatisfactory performance for the RTTs typical of GEO satellite (600 ms). For clarity, the goodput of background connections is not reported but it is always close to the maximum fair share (i.e. the bottleneck bandwidth divided by the number of

competing connections). PEPsal, by splitting the end to end satellite connection, removes the causes of this penalization achieving a goodput always very close to the maximum fair share. This is because the first connection (from the satellite sender to R2), being fully wired, has the same RTTs of competing background traffic. As a result, any penalization in the bottleneck bandwidth share is removed. As the satellite segment is supposed error free (PER=0%), the adoption of TCP Hybla on the second connection (from R2 which acts as a satellite gateway to the satellite receiver) does not provide any additional advantage with respect to the use of SACK, when dealing with persistent file transfers (PEPsal and splitting SACK show the same performance). However, note that if we have dealt with small data transfers, TCP Hybla would have further improved performance thanks to its fast opening of the cwnd at start-up. Being TCP Hybla compatible with a TCP standard receiver (it is only required to enlarge the advertised window) the TCP end-points can continue to adopt standard TCP versions. For comparison, in the figure is reported also the performance achievable by an end to end TCP Hybla connection. Although results are very close, it is worth pointing out that TCP Hybla, used as end to end protocol, would require to be installed on all the Internet Service Providers (ISP) servers which the satellite users want to reach (virtually all the ISPs). On the contrary, PEPsal requests the modification of just the satellite gateway which is usual under the satellite operator control.

B. Performance in presence of link losses

As the standard TCP does not distinguish the origin of packet losses, link errors cause spurious interference on the congestion control mechanism, causing even in this case a fast degradation of performance with increasing RTTs. This behavior is apparent by examining the performance of NewReno and SACK reported in Figure 5, which refers to the case of a single satellite connection without background traffic. Having eliminated any form of congestion, all the losses are due to the non ideal satellite channel (PER=1%). By contrast to the previous case, here the benefit provided by PEPsal is entirely to be ascribed to the adoption of TCP Hybla on the satellite segment, which is much more efficient of TCP standard in reopening the cwnd after the spurious reductions caused by random losses. This is proved by the unsatisfactory performance of the splitting SACK, which provides roughly the same result of NewReno and SACK. Performance advantage of both PEPsal and TCP Hybla, whose performance is equivalent, is impressive. In Figure 6 the same scenario is considered with a PER=5%; comments related to the previous case still hold although the higher PER induces a general performance worsening for all the protocols.

C. *Performance in presence of congestion and link losses*

The simultaneous presence of both congestion on a link shared with wired connections and random losses on the satellite channel represents the most challenging environment. Related results, obtained by setting PER=1% in the congestion only scenario previously considered, are reported in Figure 7. A limited performance worsening with respect to congestion only data reported in Figure 4 can be observed, however, the qualitative terms of comparison between the different possible solutions are left substantially unchanged by the addition of random losses. The only exception is represented by the splitting SACK, whose performance is basically ruled by the amount of PER, being totally unable to counteract random losses.

D. *Fairness and friendliness*

Fairness and friendliness are two important features for any version of TCP protocol. 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. To study fairness and friendliness in a heterogeneous environment, in Figure 8 is reported the R1-R2 bandwidth share of three satellite connections (RTT=600 ms) and three wired connections (RTT=25 ms), all simultaneously active end error free. TCP SACK confirms its full inability at providing a fair share in presence of different RTTs. By contrast PEPsal removes any penalization against satellite connections, presenting very good properties of both fairness and friendliness.

V. CONCLUSIONS

To the best of authors' knowledge, PEPsal represents the first free software implementation of an integrated TCP splitting available for Linux OS under the GNU GPL license. Results presented in the paper indicate that the PEPsal architecture is able to remove the penalization suffered by satellite connections in heterogeneous environments. Of course, as other PEP solutions at transport level, its use should be carefully planned by satellite providers in order to minimize the possible disadvantages of these architectures. At present PEPsal is adopted by WIALAN network devices manufacturer [22], which supplies some Internet satellite providers.

REFERENCES

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

[2] G. Xylomenos, G.C. Polyzos, P., Mahonen, and M. Saaranen, "TCP performance issues over wireless links" IEEE Commun. Mag., vol. 39, pp 52-58, April 2001.

[3] M. Marchese, “TCP/IP-Based Protocols Over Satellite Systems: A Telecommunication Issue,” in Reliability, Survivability and Quality of Large Scale Telecommunication Systems, P. Stavroulakis, Ed. Chichester: Wiley, 2003, pp. 167-198.

[4] C. Caini and R. Firrincieli, “TCP Hybla: a TCP Enhancement for Heterogeneous Networks”, International Journal of Satellite Communications and Networking, vol. 22, n. 5, Sep. 2004, pp. 547-566.

[5] Border J, Kojo M, Griner J, Montenegro G, Shelby Z. “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations”, IETF RFC 3135, June 2001.

[6] K. Scott, P. Feighery, B. Crow and M. Jurik, “TCP congestion control in shared satellite environments”, MILCOM 2002 - IEEE Military Communications Conference, no. 1, October 2002, pp. 46 – 50.

[7] C. E. Palazzi, C. Roseti, M. Luglio, M. Gerla, M. Y. Sanadidi, and J. Stepanek, “Enhancing Transport Layer Capability in HAPS-Satellite Integrated Architecture”, Wireless Personal Communications, Springer Science+Business Media B.V. (formerly Kluwer Academic Publishers B.V.), vol. 32, no. 3-4, Feb 2005.

[8] D. Velenis, D. Kalogeras, and B. Maglaris, “SaTPEP: a TCP Performance Enhancing Proxy for Satellite Links,” 2nd International IFIPTC6 Networking Conference, May 2002.

[9] A. Bakre, B. R. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, Proc. 15th IEEE International Conference on Distributed Computing Systems, Vancouver, British Columbia, pp. 136-143, May 1995.

[10] R. Yavatkar and N. Bhagawat, “Improving end-to-end performance of TCP over mobile internetworks”. IEEE Workshop on Mobile Computing Systems and Applications, pages 146--152, December 1994.

[11] T.R. Henderson, and R.H. Katz, “Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks”, 2nd International Workshop on Satellite-based Information Services (WOSBIS `97), Budapest, Hungary, Oct. 1, 1997.

[12] XipLink, web site: http://xiplink.com.

[13] Mentat, “SkyX XH45,” web site: http://www.mentat.com.

[14] ViaSat, “Intelligent Performance Enhancing Proxy (iPEP)”, web site: http://www.viasat.com.

[15] PEPsal source code available at http://www.sourceforge.net/projects/pepsal/.

[16] Satellite Network of Excellence, phase II, web site: http://www.satnex.de/.

[17] The Netfilter/iptables project, firewalling, NAT and packet mangling for Linux. http://www.netfilter.org.

[18] A. Bon, C. Caini, T. De Cola, R. Firrincieli, D. Lacamera, M. Marchese “An Integrated Testbed for Wireless Advanced Transport Protocols and Architectures”, in 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom06), March 1-3 2006, Barcelona, Spain.

[19] C. Caini, R. Firrincieli and D. Lacamera, “A Linux Based Multi TCP Implementation for Experimental Evaluation of TCP Enhancements”, SPECTS 2005, Philadelphia, July 2005.

[20] NistNet, web site: http://snad.ncsl.nist.gov/itg/nistnet/.

[21] C. Caini, R. Firrincieli, D. Lacamera, “PEPsal: a Performance Enhancing Proxy designed for TCP satellite connections”, in 2006 IEEE 63rd Vehicular Technology Conference (VTCSpring06), 7-10 May, Melbourne, Australia.

[22] WIALAN, web site: http://www.wialan.net/index.htm.

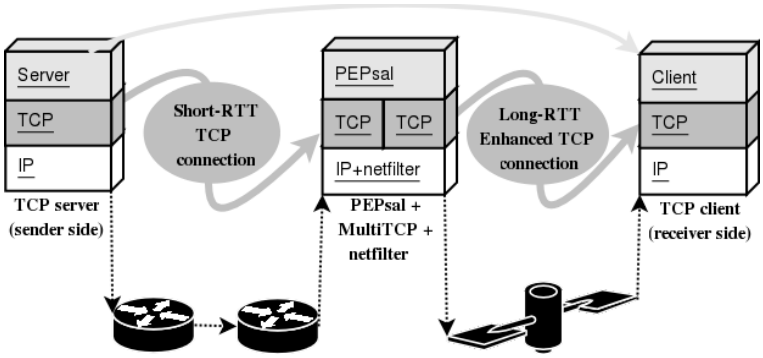

Figure 1: PEPsal architecture (based on integrated TCP splitting approach).

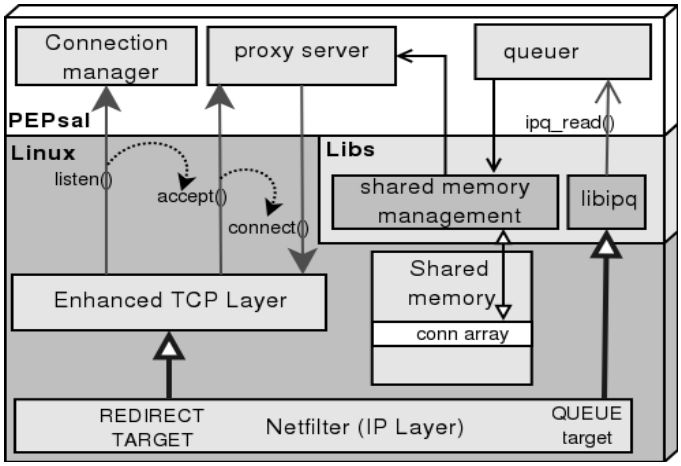

Figure 2: The PEPsal software modules.

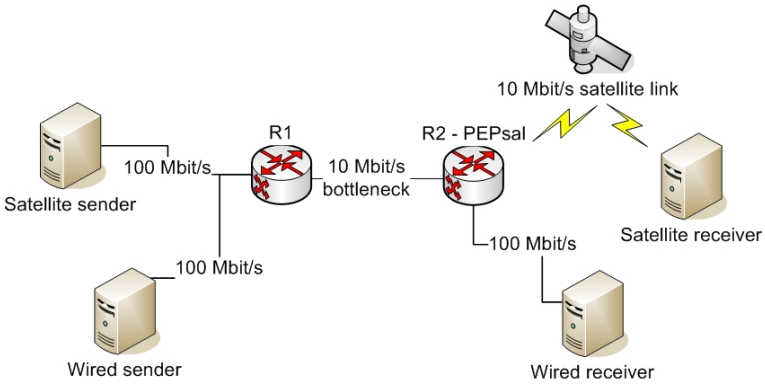

Figure 3: Layout of the testbed used for performance evaluation.

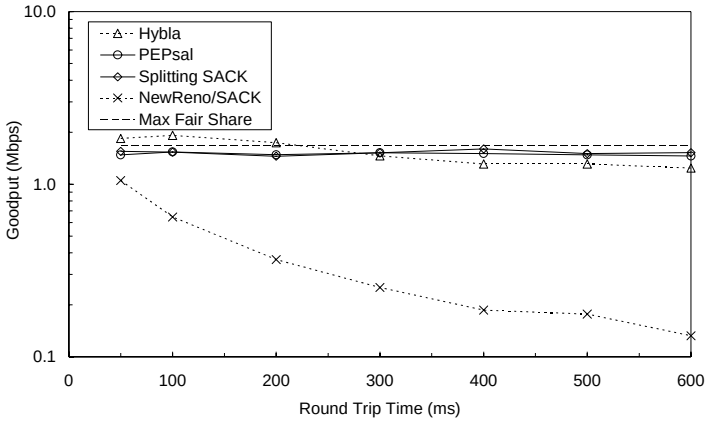

Figure 4: Performance comparison in the presence of congestion on the R1-R2 bottleneck: goodput of a satellite connection vs. its RTT; 5 background wired connection (RTT=25 ms) active. PER=0% on the satellite channel.

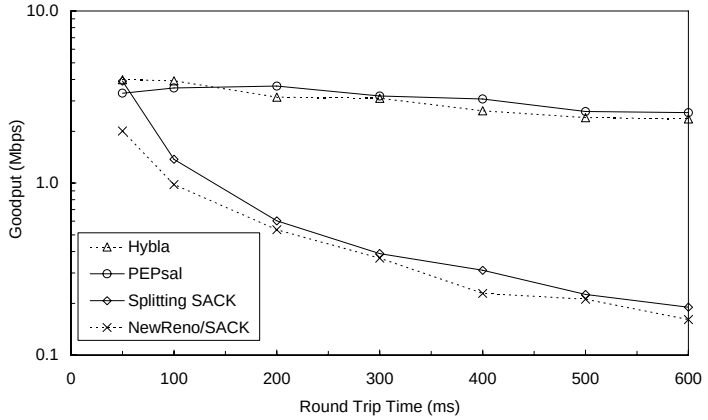

Figure 5: Performance comparison in the presence of random losses on the satellite channel (PER=1%): goodput of a satellite connection vs. its RTT; no background wired connections.

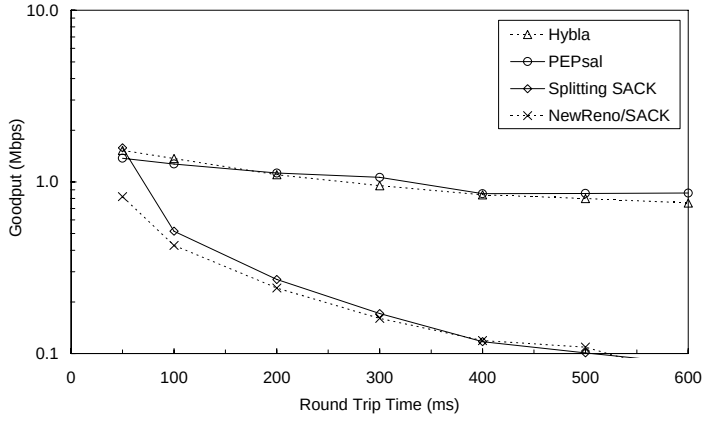

Figure 6: Performance comparison in the presence of random losses on the satellite channel (PER=5%): goodput of a satellite connection vs. its RTT; no background wired connections.

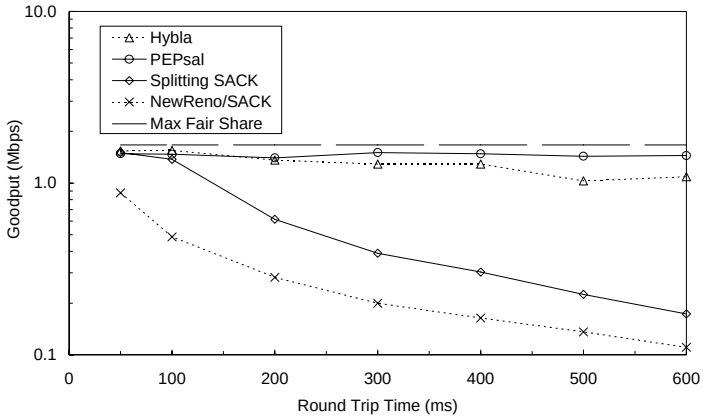

Figure 7: Performance comparison in the presence of both congestion on the R1-R2 bottleneck and random losses (PER=1%) on the satellite channel: goodput of a satellite connection vs. its RTT; 5 background wired connections active (RTT=25 ms).

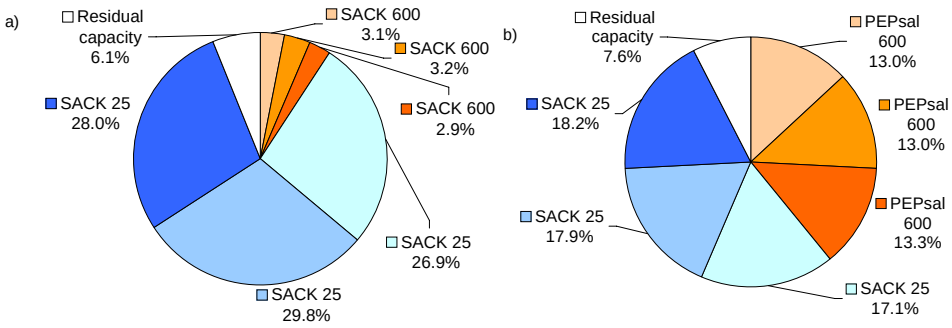

Figure 8: TCP fairness and friendliness: R1-R2 bottleneck share in a heterogeneous environment. SACK on the terrestrial connections (PER=0%, RTT=25 ms) and: a) SACK, b) PEPsal, on the satellite ones (PER=0%, RTT=600 ms).