5th Federated and Fractionated Satellite Systems Workshop
November 2–3, 2017, ISAE SUPAERO – Toulouse, France

Impact of Delayed Acknowledgment on TCP performance over LEO satellite constellations

Bastien Tauran*, Emmanuel Lochin*, Jérôme Lacan*, Fabrice Arnal†, Mathieu Gineste†, and Nicolas Kuhn‡

*Univ. de Toulouse ISAE-SUPAERO, TéSA, Toulouse, France. *firstname.name*@isae-supaero.fr
†Thales Alenia Space, Toulouse, France. *firstname.name*@thalesaleniaspace.com
‡Centre National d'Études Spatiales, Toulouse, France. *firstname.name*@cnes.fr

Abstract

This paper aims at quantifying the impact of a default TCP option, known as Delayed Acknowledgment (DelAck), in the context of LEO satellite constellations. Satellite transmissions can suffer from high channel impairments, especially on the link between a satellite and a ground gateway. To cope with these errors, physical and link layer reliability schemes have been introduced, at the price of an increase of the end-to-end delay seen by the transport layer (e.g. TCP). Although DelAck is used to decrease the feedback path load and for overall system performance, the use of this option conjointly with satellite link layer recovery schemes might increase the delay and might be counterproductive. To assess the impact of this option, we drive simulation measurements with two well-deployed TCP variants. The results show that the performance gain depends on the variant used and that this option should be carefully set or disabled as a function of the network characteristics. DelAck has a negative impact on TCP variants which are more aggressive such as TCP Hybla, and should be disabled for these versions. However, it shows benefits for TCP variants less aggressive such as NewReno.

Keywords

Satellite Constellations; TCP; Delayed Acknowledgment

1 Introduction

LEO satellite constellations aim to connect to the Internet areas which are impossible to connect via terrestrial networks, as rural areas or moving devices. On this kind of constellations, where the delays are still low compared to GEO (less than 100 ms), TCP can be used as on terrestrial networks. The amount of TCP traffic carried out by these constellations is expected to be significant [1], but the high channel constraints, mostly on Land Mobile Satellite (LMS) channels [2] make the use of satellite constellations more difficult than terrestrial networks. Thus, to counteract the high potential error rate on the LMS channel, reliability mechanisms have been introduced on this link [3, 4, 5]. One of the most efficient is Hybrid-ARQ (HARQ), which combines forward error-correcting coding and link layer retransmission. However, due to the link-layer retransmissions, HARQ increases the transmission delay and the jitter. In this paper we study type II HARQ where the main principle of this scheme is explained in the following. HARQ can retransmit data on the LMS channel, causing higher end-to-end delay and jitter that directly impact on TCP, resulting in a decrease of the TCP throughput.

To increase TCP performance, Delayed Acknowledgment [6] has been introduced for TCP. This mechanism is now activated by default over each TCP stack. DelAck reduces the number of acknowledgments sent by the TCP receiver making some packets acknowledged later. While this scheme decreases the number of acknowledgments and thus, the feedback path load, it increases the end-to-end performance. It also decreases the CPU load [7], explaining why it is activated by default. There are some studies on DelAck impact, in particular J. Chen *et al.* [8], studied the impact of DelAck on TCP over wireless links, and showed that the impact of DelAck on TCP performance depends on the topology used, and that path length has to be taken into account. Activating DelAck does not always improve TCP throughput.

The impact of DelAck has not been deeply studied in the context of satellite communications, where high and variant delays can influence the efficiency of this mechanism. L. Wood *et al.* [1] have already raised some concerns about the use of such mechanism. Furthermore, losses on satellite constellations might be due to transmission errors on the LMS channel and not to congestion as it happens on terrestrial networks. Such particular loss patterns might have an impact on the consistency of using DelAck. All these reasons motivate the present study which aims at investigating this default TCP option.

2 Scenario

This section presents how we simulate the satellite environment and the different schemes that are considered throughout this paper: the reliability mechanisms on the LMS channel to deal with the high error rate, and the transport protocol.

2.1 Satellite environment

We have chosen Network Simulator 2 (ns-2) to simulate the satellite environment, composed of 66 satellites on Low Earth Orbit, at an altitude of 800 km, ensuring a global coverage of any point on earth at any time. Due to the movement of the satellite and route changes, the transmission delay varies within the satellite constellation. Thus the delays in our constellation are varying from 70 ms to 90 ms. Moreover, except on the LMS forward link, we consider that there are no transmission errors.

The topology simulated is described in Figure 1.

2.2 Hybrid Automatic Repeat reQuest

We previously explained that HARQ schemes are used to mitigate link-layer impairments on the LMS channel. These schemes aim to compensate the high error rate on this channel. We present in this paragraph the basic principle of the version used in our simulations, which is type II HARQ. This scheme benefits from both ARQ and Forward Error-Correction coding, and optimizes the usage of the LMS channel. This version allows up to 3 retransmissions. The receiver side of the HARQ link stores the bits received while a packet has not been decoded. Each time useful or redundancy bits are received, the module computes if it has enough data to decode the packet. If not, it sends a negative ACK to the HARQ sender to notify it it has to send more redundancy bits. On the worst

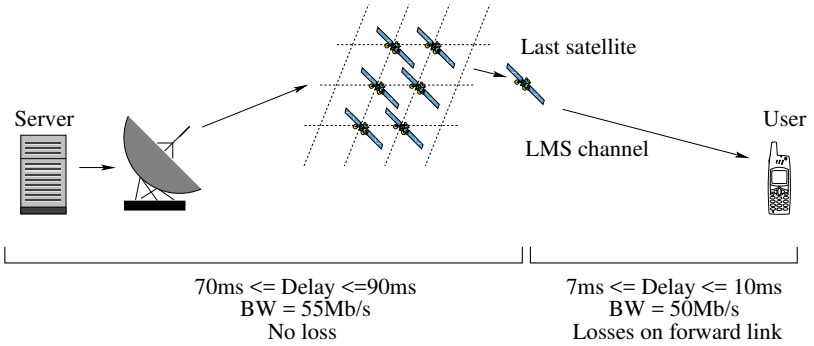


Figure 1: Model for a satellite constellation

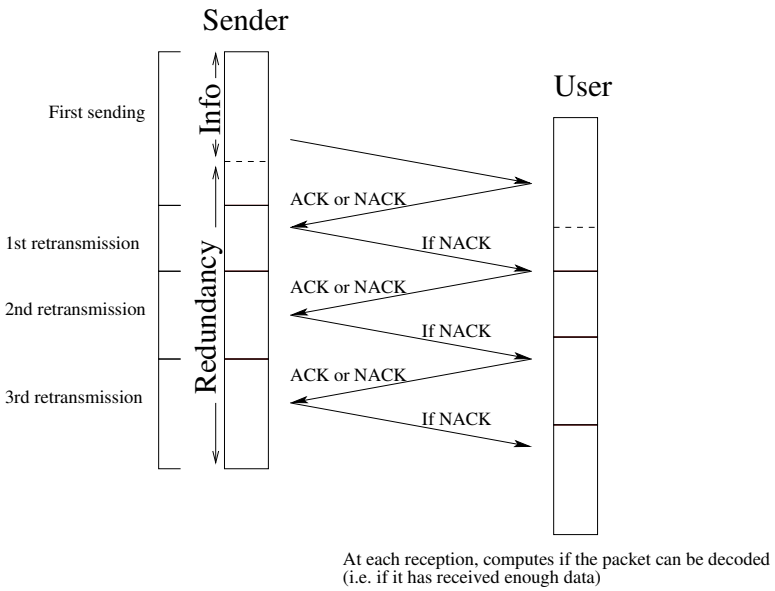


Figure 2: Description of type II HARQ

case, the packets are recovered at most $70ms$ after entering HARQ module, which corresponds to the time needed to get the first transmission and the 3 retransmissions. Figure 2 details the principle of this scheme.

2.3 Version of TCP and parameters

We experiment both TCP NewReno and CUBIC in our simulations. TCP is today the major Internet transport protocol, and has been shown to provide reasonable performances over LEO constellations [1, 9], without requiring specific PEP optimization mechanisms. TCP NewReno features basic reliability functionalities of most TCP variants and is a good variant to understand the mechanisms in our simulations. We also consider CUBIC as (1) its error recovery is more aggressive than TCP NewReno and (2) it is enabled by default in GNU/Linux and OSX systems (since `10.9`).

DelAck can combine two in-order packets if and only if they are received within a fixed time window [10]. A timer is started, when a first packet is received, and if another packet is received before the timer expires, a single cumulative acknowledgment is sent for both packets. If the timer expires, TCP acknowledges only the packet received, and reset the timer. If TCP receives an out-of-order packet, an ACK is instantly sent for this packet allowing TCP to adapt its congestion window.

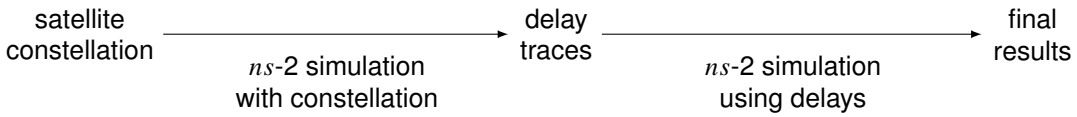


Figure 3: Illustration of the procedure to generate delay traces and their use in ns-2

2.4 Simulation scenario

To simplify the simulation, we only used three nodes in our ns-2 simulations, to represent the sender, the last satellite on the message route, and the receiver. We simulated the constellation by changing the delays on the links, using a temporal trace giving at any moment the value of the delay between the sender and the last satellite, and the last satellite and the receiver. This allows us to mimic the constellation in our simulations with only three nodes. These delays have been measured on a first ns-2 simulation using the LEO constellation described in Section 2.1. The mechanism used to generate delay traces is described in Figure 3.

The simulations are run with a Land Mobile Satellite (LMS) channel [11], [12] between the last satellite and the ground gateway in an ITS (Intermediate Tree Shadowed) environment. The LMS channel enables the HARQ module. We also vary the quality of this channel by setting an average SNR ranging from 7 dB to 13 dB. During the simulations, the link quality changes over time around this SNR average value. Each simulation lasts 600 s with one single TCP performing.

3 Study of the impact of DelAck

To analyze the impact of DelAck, we report values of RTO and DUPACK which are essential metrics used by TCP to recover lost packets. Basically, RTO is a timeout to trigger retransmissions when no packets have been received during a long time. DUPACK is a loss detection system allowing to detect a loss during a flow transmission, this mechanism allows the receiver to inform the sender that a packet is missing and trigger retransmission of this packet as soon as possible. These losses cause the congestion window of TCP to decrease, lowering the sending rate of TCP.

In this section and the following, the values of RTO, DUPACK and spurious retransmission have been normalized by the number of packets sent in order to have comparable results with the different graphs and tables.

Table 1 and Table 2 show the goodput (application throughput) achieved by TCP without DelAck for the first one and with DelAck for the second. The RTO and DUPACK columns represent the proportion of congestion events. We divided the number of times TCP received a request for retransmission by the total number of packets sent. For example, when $SNR = 7$ dB, during 600 s, CUBIC transmitted 21051 packets, the RTO timer expired 212 times and 1274 retransmissions have been triggered due to DUPACK. We have also measured 442 spurious transmissions. Thus, the proportion for RTO is 0.22 %, for duplicate acknowledgments is 2.33 % and for spurious is 2.00 %.

We observe that we have a low goodput, whatever the value of SNR, whereas we could expect goodput of 40 Mb/s if there was no errors in the network. On the other hand, the values of DUPACK and spurious retransmissions remain on high rate, and do not decrease when the channel quality improves as we could expect. This is mainly due to out-of-order packets interpreted by TCP as congestion losses which trigger spurious retransmissions and halve the congestion window.

3.1 Impact of DelAck

The default value of DelAck on GNU/Linux systems depends on the system architecture, but it is often around 40 ms. This is also the default value in ns-2 that we kept in our simulations. Results can be seen on Figures 4 and 5.

We observe that DelAck significantly improves TCP performance when using NewReno, the goodput has increased and the number of RTO, DUPACK and spurious retransmissions has decreased. However, DelAck has no impact on CUBIC goodput, even if we observe a diminution of the number of DUPACK and spurious retransmissions.

This shows that there is another mechanism, linked to DelAck and the topology studied in this paper, which counteracts the diminution of DUPACK, RTO and spurious retransmissions, especially when using CUBIC. An explanation for this bad performance is provided in Section 4.

Table 1: Without DelAck

SNR (dB)	TCP Goodput (kb/s)		HARQ Success (%)		RTO (%)		DUPACK (%)		Spurious (%)	
	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC
7	245	274	96.47	95.23	0.49	0.22	2.35	2.33	2.51	2.00
8	267	333	97.22	96.95	0.29	0.04	2.33	2.18	2.60	1.81
9	295	384	97.89	97.60	0.15	0.01	2.34	1.91	2.75	1.56
10	314	411	98.47	98.29	0.09	0.006	2.22	1.84	2.83	1.57
11	330	479	98.77	98.82	0.05	0.002	2.25	1.66	2.90	1.39
12	361	496	99.12	99.22	0.03	0.002	1.97	1.44	2.84	1.40
13	384	493	99.52	99.36	0.003	0	1.92	1.53	2.81	1.54

Table 2: With DelAck

SNR (dB)	TCP Goodput (kb/s)		HARQ Success (%)		RTO (%)		DUPACK (%)		Spurious (%)	
	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC
7	285	271	96.12	95.33	0.36	0.20	1.65	1.96	0.90	1.50
8	346	352	97.49	97.18	0.10	0.03	1.58	1.71	1.00	1.19
9	366	355	97.83	97.46	0.08	0.007	1.43	1.72	1.10	1.46
10	381	416	98.49	98.43	0.05	0.003	1.29	1.51	1.19	1.40
11	418	443	98.92	98.82	0.01	0	1.32	1.34	1.26	1.55
12	432	451	99.16	99.14	0	0	1.21	1.27	1.41	1.39
13	444	484	99.31	99.42	0.008	0	1.27	1.23	1.40	1.55

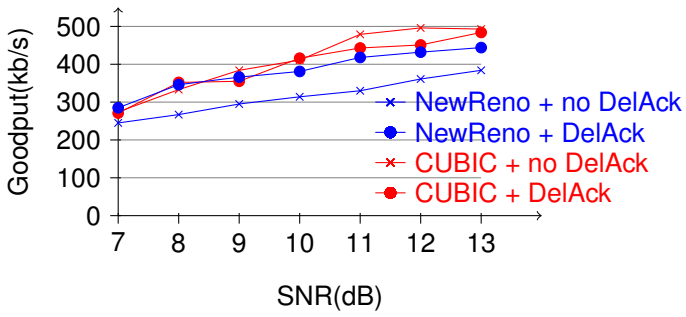
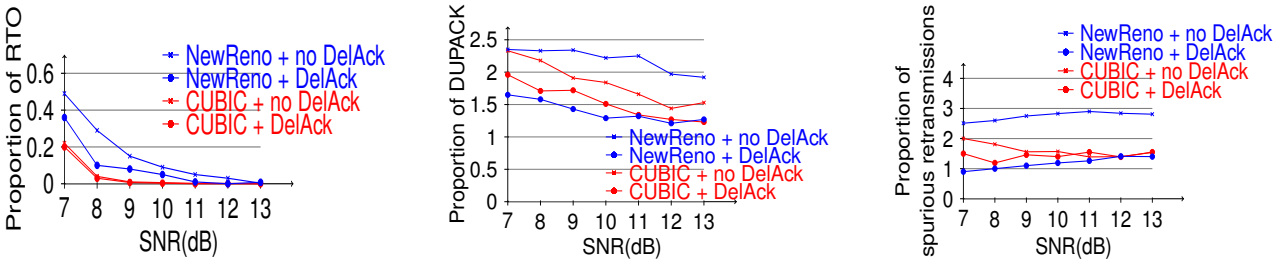


Figure 4: Impact of DelAck on end to end goodput

3.2 DelAck timer

To go on with DelAcks, we first crosscheck whether the default DelAck timer value was consistent in our scenario. It appears that in our case and as shown in Figure 6, a value around 40 ms is a fair compromise. DelAck timer mechanism is explained in Subsection 2.3. We computed the number of packets received by HARQ depending on the value of timeout, in a range between 10 ms and 500 ms, which is the maximum recommended value [10]. The results for different SNR values are shown in Figure 6. We can see a maximum between 30 ms and 40 ms, for all values of SNR, confirming the value of 40 ms chosen in our simulations. This value is linked to our topology studied, a LEO satellite constellation, to the capacity of the links and the delays. However, other topologies might



(a) Impact on RTO (b) Impact on DUPACK (c) Impact on spurious

Figure 5: Impact of DelAck on TCP performance

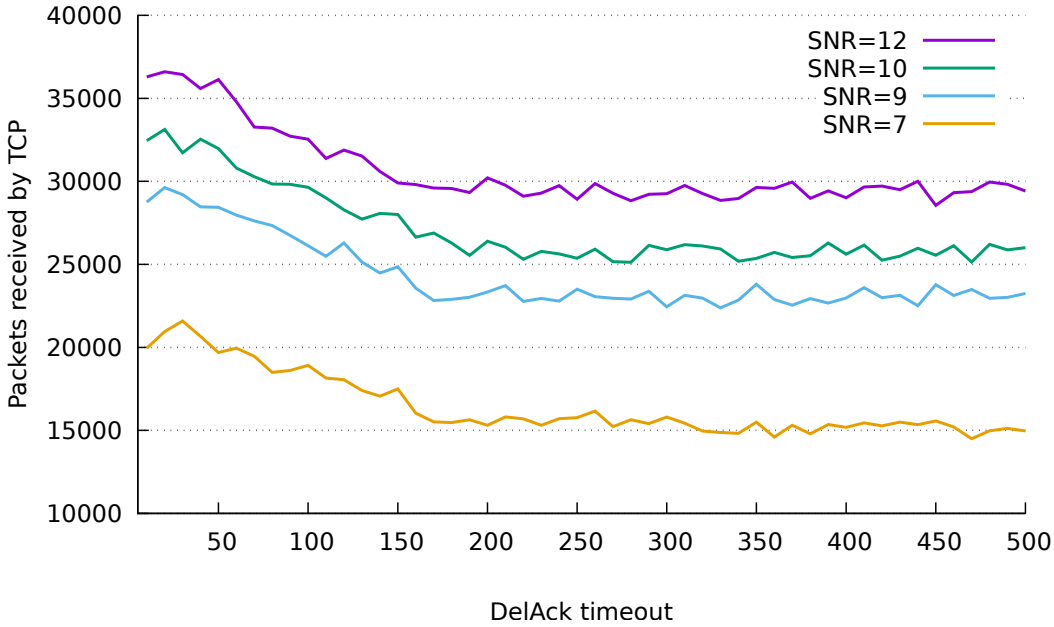


Figure 6: Number of packets received depending on the DelAck timeout value and SNR

lead to other optimal values. Other systems are using different timer values to have the best performance with their topology.

3.3 Impact with other TCP protocols

We recall that we only presented detailed results with NewReno and CUBIC. We also tested our results with other TCP protocols such as Hybla. Hybla [13] is a TCP version specifically designed for long delay links. The idea of Hybla is to fake the RTT of the link by using a normalized RTT, making the evolution of the congestion window independent of RTT. With TCP Hybla, we observe a strong decrease of the goodput when DelAck is activated. The goodput loss can be up to $40\,\%$ with this protocol, and DelAck should be disabled when using this TCP variant over a LEO satellite constellation.

4 Analysis

The evolution of TCP congestion window being generally based on the number of acknowledgments received on slowstart or in fast recovery, the use of DelAck makes this growth less aggressive in these phases because of the

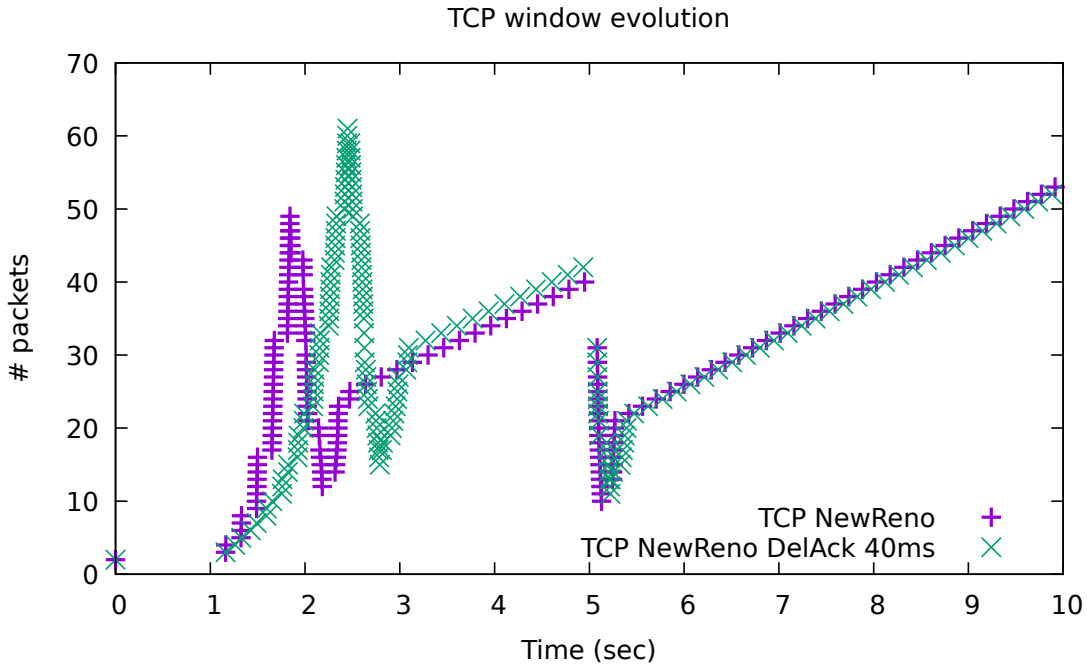


Figure 7: Evolution of TCP congestion window

Table 3: Number of packets retransmitted, when DelAck is activated

SNR (dB)	Number of retransmissions		
	NR	CUBIC	Hybla
7	1068	1333	6969
8	932	1100	16914
9	925	1105	23073
10	789	949	16782
11	748	923	22399
12	742	780	23852
13	706	789	20688

reduced number of acknowledgments received [1]. This evolution can be seen on Figure 7.

Such pacing might have benefits for congested networks by delaying the filling of the buffer at the bottleneck and then allowing the flow to have a larger congestion window [14]. However in our scenario, we are placed in an uncongested network, and pacing will only delays transmissions and loss detection signals [15]. For the less aggressive TCP variants, the low frequency of retransmissions allows to have better TCP performance, by taking advantage of DelAck. However, with the most aggressive variants, TCP performance will be badly impacted by the high number of retransmissions.

In our context of LEO satellite communications, where delays are varying due to satellite movements, route changes and reliability mechanisms such as HARQ, any TCP variant triggers more retransmissions than on terrestrial networks. Moreover, TCP needs to recover packets losses due to errors on the LMS channel, in addition to those lost due to congestion. Some solutions have been proposed to improve TCP performance, but this cannot totally mitigate the effects brought by the use of satellite constellations, and we still have more slowstart and fast recovery phases than on terrestrial networks due to these random losses.

We observe on Table 3 that CUBIC triggers more often retransmissions than NewReno. CUBIC is more often

in slowstart or fast recovery phase, where the congestion window grows slower. Thus the gain of performance for TCP is lower with CUBIC than NewReno. This trend is also observed with TCP Hybla, where the number of retransmissions is very higher than with NewReno or CUBIC due to the larger congestion window [13]. In this case, the goodput is badly impacted by DelAck, as our simulations showed. On the other hand, TCP variants such as NewReno are often in congestion avoidance phase, where the congestion window growth is the same with and without DelAck. For these kinds of variants, DelAck improves TCP performance.

On a second time, we showed that the default value of DelAck timer, 40 ms is the best value to maximize TCP performance. Thus transmissions using DelAck do not need to change the DelAck timer value and can keep the default one.

5 Conclusion

We argued in this paper on the importance of studying the impact of DelAck on TCP performance over LEO satellite constellations. We showed that adding DelAck can have different impact on TCP performance, depending on the variant of TCP used. It is recommended to activate it when using NewReno, with a default timer value of 40 ms. However, with CUBIC, it can decrease TCP performance in some cases, and the DelAck activation is not necessary.

More generally, TCP variants which are more aggressive and retransmit often such as Hybla are negatively impacted by DelAck, which should be disabled to improve performance.

Acknowledgment

The authors would like to thank CNES and Thales Alenia Space for funding support.

References

[1] L. Wood, G. Pavlou, and B. Evans. Effects on tcp of routing strategies in satellite constellations. *IEEE Communications Magazine*, 39(3):172–181, Mar 2001.

[2] N. Blaunstein, Y. Cohen, and M. Hayakawa. Prediction of fading phenomena in land-satellite communication links. *Radio Science*, 45(6), 2010.

[3] A. Sastry. Performance of hybrid error control schemes of satellite channels. *IEEE Transactions on Communications*, 23(7):689–694, Jul 1975.

[4] Tomaso De Cola, Harald Ernst, and Mario Marchese. *Application of Long Erasure Codes and ARQ Schemes for Achieving High Data Transfer Performance Over Long Delay Networks*, pages 643–656. Springer US, Boston, MA, 2008.

[5] Nicolas Kuhn, Emmanuel Lochin, Jérôme Lacan, Roksana Boreli, and Laurence Clarac. On the impact of link layer retransmission schemes on tcp over 4g satellite links. *International Journal of Satellite Communications and Networking*, vol. 33:pp. 19–42, 2015.

[6] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (INTERNET STANDARD), October 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864.

[7] Glenn Judd. Attaining the promise and avoiding the pitfalls of tcp in the datacenter. In *NSDI*, pages 145–157, 2015.

[8] Jiwei Chen, Mario Gerla, Yeng Zhong Lee, and M.Y. Sanadidi. Tcp with delayed ack for wireless networks. *Ad Hoc Networks*, 6(7):1098 – 1116, 2008.

[9] Y. Chotikapong, H. Cruickshank, and Zhili Sun. Evaluation of tcp and internet traffic via low earth orbit satellites. *IEEE Personal Communications*, 8(3):28–34, Jun 2001.

[10] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. RFC 5681 (Draft Standard), September 2009.

[11] F. Perez-Fontan, M. A. Vazquez-Castro, S. Buonomo, J. P. Poiares-Baptista, and B. Arbesser-Rastburg. S-band lms propagation channel behaviour for different environments, degrees of shadowing and elevation angles. *IEEE Transactions on Broadcasting*, 44(1):40–76, Mar 1998.

[12] F. P. Fontan, M. Vazquez-Castro, C. E. Cabado, J. P. Garcia, and E. Kubista. Statistical modeling of the lms channel. *IEEE Transactions on Vehicular Technology*, 50(6):1549–1567, Nov 2001.

[13] Carlo Caini and Rosario Firrincieli. Tcp hybla: a tcp enhancement for heterogeneous networks. *International journal of satellite communications and networking*, 22(5):547–566, 2004.

[14] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of tcp pacing. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1157–1165 vol.3.

[15] Renaud Sallantin, Cedric Baudoin, Emmanuel Chaput, Fabrice Arnal, Emmanuel Dubois, and Andre-Luc Beylot. Initial spreading: A fast start-up tcp mechanism. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 492–499. IEEE.