Impact of Delayed Acknowledgment on TCP performance over LEO satellite constellations

Bastien Tauran* and Emmanuel Lochin† and Jérôme Lacan‡
Université de Toulouse ISAE-SUPAERO, TéSA, Toulouse, 31400, France. firstname.name@*isae-supaero.fr*

Fabrice Arnal§ and Mathieu Gineste¶
Thales Alenia Space, Toulouse, 31100, France. firstname.name@*thalesaleniaspace.com*

Nicolas Kuhn‖
Centre National d'Etudes Spatiales, Toulouse, 31400, France. firstname.name@*cnes.fr*

Satellite transmissions can suffer from high channel impairments, especially on the link between a satellite and a mobile end user. To cope with these errors, physical and link layer reliability schemes have been introduced at the price of an increased end-to-end delay seen by the transport layer (*e.g.* TCP). By default, TCP enables Delayed Acknowledgment (DelAck), that might increase the end-to-end delay when performing over satellite link-layer recovery schemes. As a matter of fact, even if this option enables to decrease the feedback path load and the stack processing overhead, it might be counterproductive in a satellite context. This motivates the present paper that aims at quantifying the impact of such TCP option in the context of Low Earth Orbit (LEO) satellite constellations. We perform several simulation measurements with two well-deployed TCP variants and show that DelAck should be disabled when used over link-layer HARQ schemes particularly when these schemes enable reordering buffer.

Introduction

LEO satellite constellations allow to cope with the digital fracture by providing internet access to isolated or rural areas. The delays of LEO constellations are in the same order of magnitude than terrestrial links which authorize the use of TCP as transport protocol without deploying Proxy-Enhanced-Proxy (PEP) systems [1]. Moreover, the new satellite constellations [2, 3] have a broadband goal, in which the part of TCP is expected to be significant [4], but the high channel constraints, mostly on Land Mobile Satellite (LMS) channels [5], may induce challenges for end-to-end protocols.

*PhD student, ISAE-SUPAERO, TéSA, 7 Boulevard de la Gare, 31500 Toulouse, France
†Professor, ISAE-SUPAERO, 10 Avenue Edouard Belin, 31400 Toulouse, France
‡Professor, ISAE-SUPAERO, 10 Avenue Edouard Belin, 31400 Toulouse, France
§Research Engineer, Telecom Business Unit, Thales Alenia Space, 26 Avenue Jean François Champollion, 31100 Toulouse, France
¶Research Engineer, Telecom Business Unit, Thales Alenia Space, 26 Avenue Jean François Champollion, 31100 Toulouse, France
‖SATCOM Research Engineer, CNES - Telecommunication System Department, 18 Avenue Edouard Belin, 31400 Toulouse, France

Reliability mechanisms have been introduced on the LMS channel [6–8] to counteract the high potential error rate on this link. One of the most efficient is Hybrid Automatic Repeat reQuest (HARQ) which combines forward error-correcting coding and link layer retransmission. In our evaluations, such mechanism is deployed between the last satellite on the route and the ground receiver. We select in this paper type II HARQ, which is commonly deployed over physical layer. HARQ can retransmit data on the LMS channel, causing higher end-to-end delay and jitter that directly impact on TCP. This may result in a decrease of the throughput of TCP. To improve TCP performance, a low layer reordering mechanism has been proposed [9] to mitigate the jitter caused by HARQ. However, this buffer might increase the end-to-end delay, unfortunately high due to the path length and the possible HARQ retransmissions. At the transport layer, Delayed Acknowledgments (DelAck) [10] have also been introduced to improve TCP performance but mostly considering terrestrial communication links. 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 also increases end-to-end performance. In particular, DelAck decreases the CPU load [11] as the number of acknowledgments processed decrease, explaining why this option is activated by default. In short, these reliability mechanisms exhibit a trade-off between reliability and delay. However, when deployed at different layers, their conjoint use can lead to counterproductive effects that need to be investigated and is tackled throughout this paper.

This paper assesses whether the TCP DelAck option improves TCP performance when used conjointly with layer 2 reliability mechanisms, in the context of mobile users of LEO constellations. We found only few studies investigating the impact of DelAck. J. Chen *et al.* [12], studied the impact of DelAck on TCP over wireless links and showed that their impact on TCP performance depends 1) on the topology used; and 2) the path length. They conclude that activating DelAck does not always improve TCP throughput. In the context of satellite systems, L. Wood *et al.* [13] have also raised some concerns about the use of such mechanism. As losses on satellite constellations might be due to transmission errors on the LMS channel and not to congestion, such particular loss patterns and their corresponding recovery mechanisms 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 and to fill the gap by explaining the rationale in using or not this mechanism over LEO systems.

We present in the following the scenario used. We first describe the satellite environment, then introduce the low layer reliability mechanisms used to deal with the high channel impairments on this environment, then the TCP versions chosen and the DelAck option. We study the impact of DelAck on TCP performance in our scenario in Section III, then explain these results obtained in Section IV.

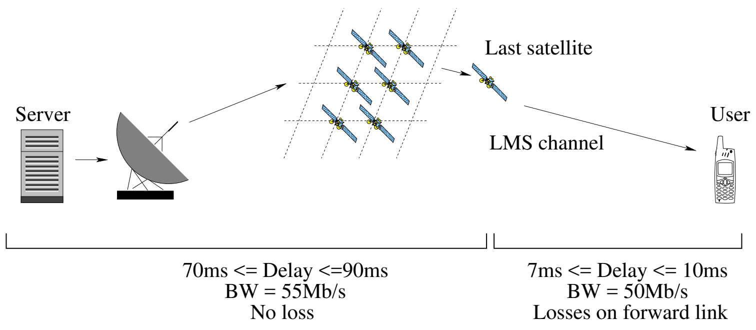


Fig. 1 Model for a satellite constellation

Scenario

This section presents the satellite scenario, 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.

Satellite environment

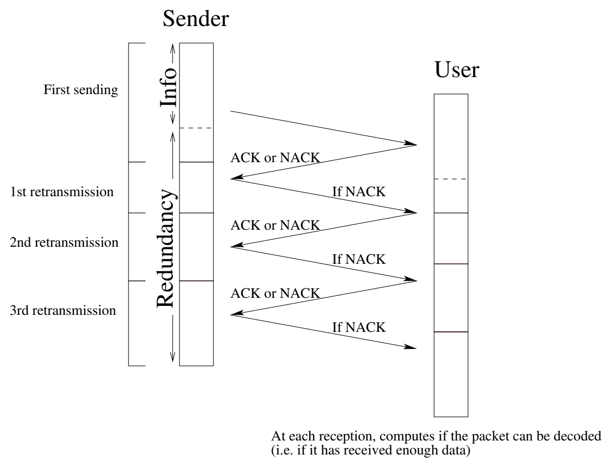
We have chosen Network Simulator 2 (*ns*-2) to simulate the satellite environment, composed of 66 LEO satellites, at an altitude of 800 km, ensuring a global coverage of any point on earth at any time. Due to the movement of the satellite and route changes, the transmission delay varies from 70 ms to 90 ms within the satellite constellation. Moreover, except on the LMS forward link, we consider a path error free, as shown in Figure 1, between the sender and the last satellite. Finally, messages are sent from the server to the mobile receiver through a satellite constellation representing the existing satellite constellations already deployed.

Hybrid Automatic Repeat reQuest

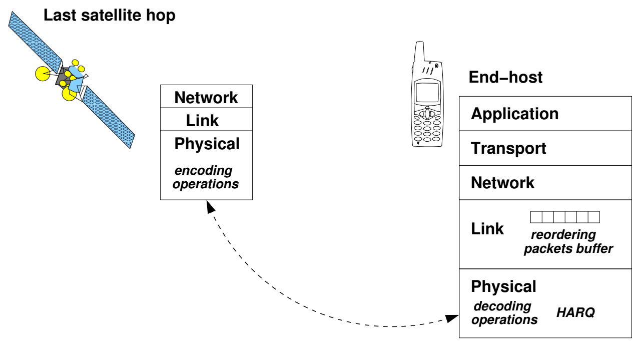
We previously explained that HARQ schemes are used to mitigate link-layer impairments on the LMS channel. These schemes aim to compensate the high error rate characterizing such channel, by benefiting from both Automatic Repeat reQuest (ARQ) and Forward Error-Correction (FEC) coding, and optimizing the usage of the LMS channel. There are two main types of HARQ:

- Type I HARQ, also named Chased Combining (CC): when a retransmission is needed, the sending node sends again the same message than in the first transmission, composed of the message and redundancy bits;
- Type II HARQ, also named Incremental Redundancy (IR): the information bits are only sent at the first sending, with some redundancy bits. When a retransmission is asked, the sender sends more redundancy bits, that are different from the bits sent in previous transmissions. Thus each reception of additional data will add more redundancy bits to help decode the message.

We present in this paragraph the basic principle of the version used in our simulations, which is Adaptive-HARQ [14].



(a) Description of type II HARQ



(b) Description of HARQ with the reordering mechanism

Fig. 2 Description of the low layer mechanisms

This version is an improvement of type II HARQ, and uses the mutual information to compute at each reception if a packet can be decoded, and if not, how many additional redundancy bit are needed. This optimizes the use of the LMS channel capacity by sending only the desired number of bits. This version allows up to 3 retransmissions in case of erased packets. The principle detailed in Figure 2a is as follows: 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 whether enough data has been received to decode the packet. If not, a negative ACK is sent to the HARQ sender, asking for more redundancy bits. Transmissions and decoding times take between 10 ms (no retransmission of redundancy bits) and 70 ms (3 retransmissions), depending on the channel quality. If the packet cannot be decoded after 3 retransmissions, it is dropped.

As mentioned in Section I, solutions have been proposed to improve the performance of the communication by mitigating the jitter caused by HARQ. Due to the varying decoding time by HARQ, packets are delivered out-of-order by this module to the upper layers. TCP stores these out-of-order packets, but this generates a lot of retransmissions that lower TCP performance and should be avoided. The reordering mechanism proposed by [9] allows the packets to be reordered at the link layer and then delivered in-order to the upper layers. This solution shows better results in terms of TCP performance. Thus upper layers, particularly transport layer, receive ordered packets that lead to better performance. The cost of this mechanism is an additional delay that needs to be below the TCP timeout value. An overview of this mechanism including both HARQ and the link layer reordering mechanism is given in Figure 2b.

Concerning the buffer sizes, the buffers used to store packets being decoded by HARQ (waiting for additional redundancy bits) and to reorder them have been designed to always handle all the packets being processed. Thus we never have overflow in these buffers.

Finally, we reused the LMS channel details used in [14] to characterize the LMS channel:

- Modulation QPSK
- Mother FEC code, CCSDS Turbo Codes 1/6
- Codeword length: 53520 bits (Data bits: 8920 bits)

In the following, we study the impact of DelAck with and without this link layer reordering mechanism, and assess whether enabling DelAck is mandatory in both cases.

Version of TCP and parameters

We experiment both TCP NewReno and CUBIC in our simulations, presented in Section 5 of [15] as recommended TCP variants. TCP is today the major Internet transport protocol, and has been shown to provide reasonable performance over LEO constellations [13, 16] without requiring specific PEP optimization mechanisms. TCP NewReno features basic reliability functionalities of most TCP variants and is used here as a reference protocol in our simulations. We also consider CUBIC [17] as (1) it is enabled by default in GNU/Linux and OSX systems (since `10.9`) and (2) performs better than TCP NewReno over long delay links [8]. CUBIC congestion window is growing faster than TCP NewReno window, in order to reach faster the optimal value. It has been designed to improve the scalability of TCP over fast and long distance networks.

DelAck can combine two in-order packets if and only if they are received within a fixed time window [18]. 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.

Simulation scenario

To ease the simulation and make it faster, we performed simulations with only three nodes to represent the sender, the last satellite on the message route, and the receiver. The satellite constellation is simulated by varying the delays between the nodes. We used `SaVi` [19] to get the parameters of the previously described LEO satellite constellation, then we played a first simulation to get the evolution of the delays between the sender and the last satellite, and between the last satellite and the ground receiver. Then all *ns*-2 simulations are played using the three nodes and the temporal traces generated.

The simulations are run with a LMS channel [20], [21] between the last satellite and the ground gateway in an Intermediate Tree Shadowed (ITS) environment. HARQ is always activated in our simulations. We also vary the quality of this channel by setting a reference Signal Noise Ratio (SNR) ranging from 7 dB to 13 dB. During the simulations, the link quality changes over time around this SNR reference value. Each simulation lasts 600 s with one single TCP flow performing.

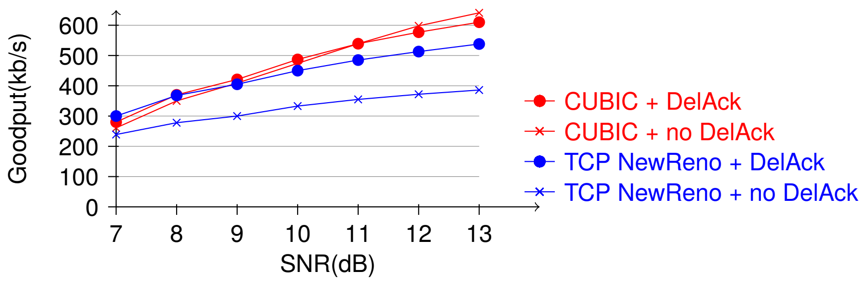


Fig. 3 Impact of DelAck on end to end goodput

Study of the impact of DelAck

To analyze the impact of DelAck, we report values of TCP Retransmission Timeout (RTO) and TCP Duplicate Acknowledgment (DUPACK) which are essential metrics used by TCP to recover lost packets. Basically, RTO is a timeout to trigger retransmissions when no acknowledgments have been received after a given period, while DUPACK detects a loss in the middle of a flow transmission. These mechanisms allow the receiver to inform the sender that a packet is missing and trigger retransmission of this packet as soon as possible. These congestion events lower the sending rate of TCP.

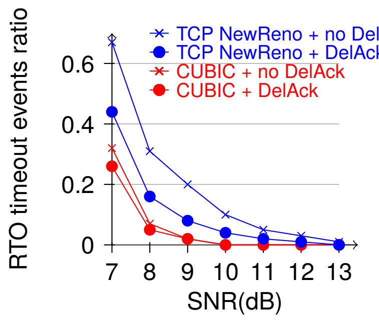
In this section and the following, the values of RTO timeout events, DUPACK and spurious retransmission (unnecessary retransmission of TCP segments that are delayed and not lost) have been normalized by the number of packets sent in order to have comparable results in the different graphs and tables.

We observe, as shown in Figures 3 and 4, that we have a low goodput, whatever the value of SNR, whereas we could expect a goodput of 40 Mb/s in case of error free network. On the other hand, the values of DUPACK and spurious retransmissions remain high, and do not decrease when the channel quality improves as we could expect. This is mainly due to out-of-order packets arriving too late and as a consequence, interpreted by TCP as a congestion event which results in spurious retransmissions and congestion window halving.

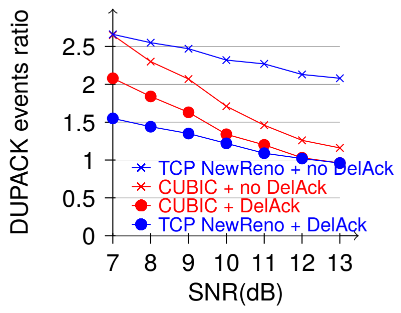
Some packets cannot be decoded after 3 retransmissions by HARQ and are dropped. This proportion of packets ranges from 4 % with SNR=7 to less than 0.5 % with SNR=13. As we could expect, the proportion of packets not recovered by HARQ decreases when the channel quality improves.

Impact of DelAck without reordering mechanism

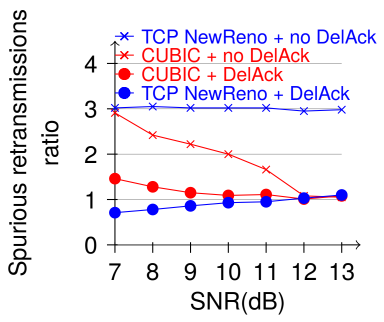
The default value of DelAck on GNU/Linux systems depends on the system architecture, but is often around 40 ms. This is also the default value in *ns*-2 that we kept in our simulations. As shown in Figures 3 and 4, we observe that DelAck significantly improves TCP performance when using TCP NewReno, the goodput has increased and the number of RTO timeout events, 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



(a) Impact on RTO timeout events

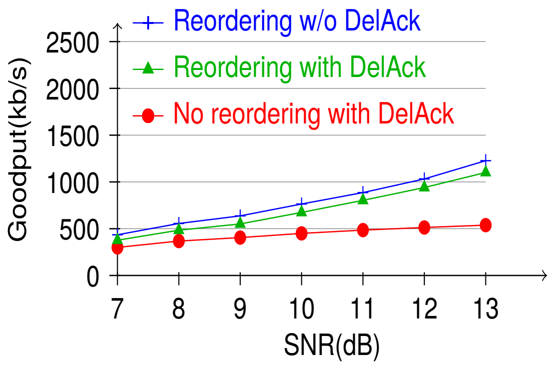


(b) Impact on DUPACK events

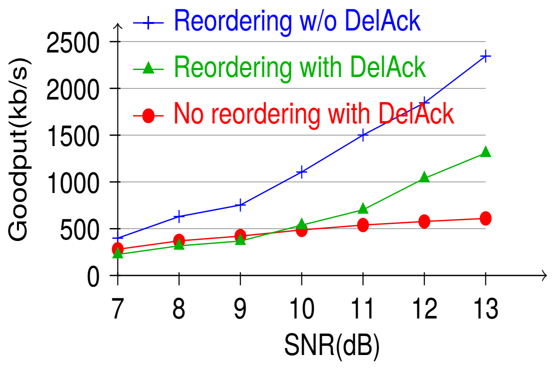


(c) Impact on spurious retransmissions

Fig. 4 Impact of DelAck on TCP performance



(a) With TCP NewReno



(b) With CUBIC

Fig. 5 Impact of DelAck on TCP performance with a link layer reordering mechanism

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

Impact of DelAck with the link layer reordering mechanism

As seen in Figure 5, when the link layer reordering mechanism is enabled, activating DelAck greatly decreases TCP performance, more particularly when using CUBIC, where the goodput fall is significant for high values of SNR. This behavior can be observed for both TCP NewReno and CUBIC.

This points out the need to understand why the activation of DelAck decreases TCP goodput in this scenario, explications are given on Section IV.

Impact with other TCP protocols

Although we only presented detailed results with TCP NewReno and CUBIC, we also experimented without reordering mechanisms using other TCP protocols such as TCP Hybla, a TCP version designed for GEO satellite links [22]. Its performance on long delay links makes TCP Hybla a good candidate to study and it can bring additional results in our scenario. We observe a strong decrease of the goodput when DelAck is activated. The fall can be up to 40 % with this protocol, leading to the conclusion that DelAck should be disabled when using TCP Hybla over a LEO satellite constellation.

However, despite this performance drop when enabling DelAck, TCP Hybla still performs better goodput than TCP NewReno or CUBIC. With DelAck, TCP Hybla goodput is always higher than 850 kb/s, when TCP NewReno and CUBIC cannot achieve goodputs higher than 650 kb/s. Without DelAck, the goodput with TCP Hybla ranges from 1141 kb/s ($SNR = 7$) to 1764 kb/s ($SNR = 13$).

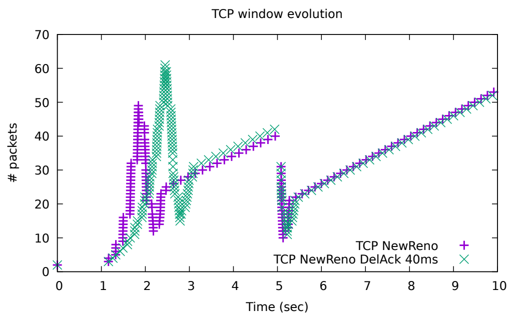
Analysis

TCP congestion window being generally updated when an acknowledgment is received, during the slowstart or fast recovery phase, the use of DelAck slows down this growth because of the reduced number of acknowledgments, as observed in [13]. The evolution of the congestion window with and without DelAck is shown in Figure 6. Each curve has been generated independently with one single TCP flow between two nodes, without any loss and without HARQ. This is just a simple case illustrating the impact of DelAck on the congestion window evolution. 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 [23]. However in our scenario, we are placed in a mostly uncongested network, meaning that losses are mainly due to transmission errors. As a results, pacing only delays transmissions and loss detection signals [24], resulting in a decrease of the goodput.

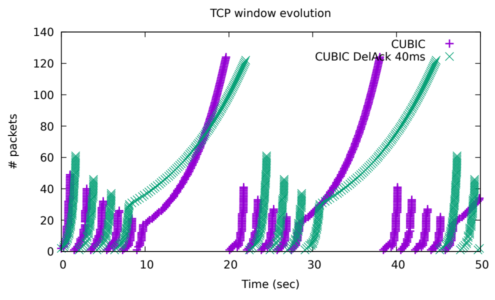
In our context of LEO satellite communications, where delays are varying due to satellite movements, route changes and reliability mechanisms such as HARQ, any TCP variant triggers more retransmissions than on terrestrial networks. Moreover, TCP needs to recover packets lost due to errors on the LMS channel, in addition to those lost due to congestion. All the solutions previously listed cannot totally mitigate the effects brought by the use of satellite constellations, and we still observe more slowstart and fast recovery phases than on terrestrial networks due to these random losses [25].

We observe on Table 1 that CUBIC triggers more retransmissions than TCP NewReno. CUBIC is more often in slowstart or fast recovery phase, where DelAck delays the progressions of the congestion window. Thus the gain of performance for TCP when DelAck is activated is lower with CUBIC than TCP NewReno. This trend is also observed with TCP Hybla, where the number of retransmissions is higher than with TCP NewReno or CUBIC due to the larger congestion window [22]. Thus, in our scenario, when a transport layer protocol is aggressive (e.g. TCP Hybla or CUBIC), enabling DelAck results in slower progression of the congestion window and a reduced goodput, whereas enabling DelAck for transport protocols with lower retransmission levels (e.g. TCP NewReno) improves the goodput of the connection, by taking advantage of DelAck.

Concerning TCP performance when the link layer reordering mechanism is present, activating DelAck increases the



(a) With TCP NewReno



(b) With CUBIC

Fig. 6 Evolution of TCP congestion window

SNR (dB)	Number of retransmissions		
	TCP NewReno	CUBIC	TCP 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

Table 1 Number of packets retransmitted during all the simulation, when DelAck is activated

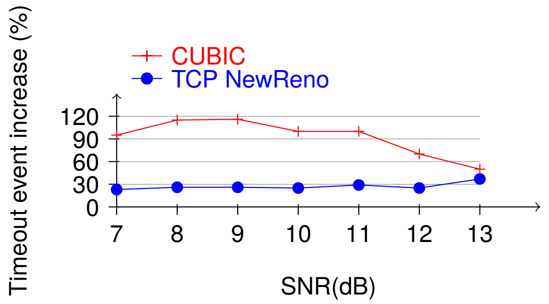


Fig. 7 Impact of the activation of DelAck on the retransmission ratio due to timeout when the reordering mechanism is present

transmission delay, increasing also the probability of TCP timeout. We measured that the waiting time between two packets forming a DelAck is higher than 10 ms in 10 % of the cases. This implies that the acknowledgment of the first packet of DelAck is delayed by the same value. This waiting time is significant due to the reordering mechanism, and we observe that this additional delay is often proportional to the time needed to get an additional HARQ transmission to decode a packet.

Furthermore, we observe a decrease of the value of RTO for both TCP NewReno and CUBIC when DelAck is enabled, giving less time to the packets to reach their destination, increasing the probability to trigger retransmissions due to timeout. We recall that the RTO timer is regularly updated during the transmission, and is the sum of $SRTT$ and four times $RTTVAR$ [26]. $SRTT$ is the exponential moving average of RTT, and $RTTVAR$ is the exponential moving average of the absolute value of the difference between last RTT measured and $SRTT$. This algorithm is made to adapt the RTO to the RTT changes. Thus if the RTT has a high variation, the computed RTO is higher to handle packets with a longer transmission delay and avoid spurious retransmissions. On the other hand, if there is no RTT variation, the RTO is smaller to detect as soon as possible a packet loss. We observe in the simulations that the RTT variation is lower when DelAck is activated. Thus both $RTTVAR$ and RTO value are lower.

This lower RTO value and the additional delay caused by DelAck, added to the constellation transmission delay, HARQ and the reordering mechanism, cause a higher number of retransmissions by RTO as presented in Figure 7. This high number of timeout badly impacts the performance of TCP, decreasing its performance: it lowers the congestion window and switches to the slowstart mode, where the growing of the congestion window is slower. The goodput is finally highly impacted by the changes brought by DelAck. However, thanks to the reordering mechanism, we have a lower proportion of DUPACK, and TCP performance remains higher than without reordering mechanism, even if DelAck has a counterproductive impact.

Conclusion

This paper assesses the impact of activating DelAck on the behavior of different TCP variants over LEO satellite constellations where link-layer reliability schemes are considered. 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 TCP NewReno. However, with CUBIC, it can decrease TCP performance in some cases, and the DelAck activation is not mandatory. More generally, TCP variants which are more aggressive and retransmit often such as TCP Hybla are negatively impacted by DelAck, which should be disabled to improve performance. There is also a need to carefully take into account the presence of other link layer mechanisms aiming to improve TCP performance with HARQ, the use of DelAck conjointly with them can be counterproductive, by adding delay to a network where transmission delays are already high.

References

[1] Border, J., Kojo, M., Griner, J., Montenegro, G., and Shelby, Z., "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135 (Informational), Jun. 2001. URL `http://www.ietf.org/rfc/rfc3135.txt`.

[2] "Iridium NEXT," , 2017. URL `https://www.iridium.com/network/iridium-next/`.

[3] "OneWeb," , 2018. URL `http://www.oneweb.world/`.

[4] Lee, D., Carpenter, B. E., and Brownlee, N., "Observations of UDP to TCP Ratio and Port Numbers," *2010 Fifth International Conference on Internet Monitoring and Protection*, 2010, pp. 99–104.

[5] Blaunstein, N., Cohen, Y., and Hayakawa, M., "Prediction of fading phenomena in land-satellite communication links," *Radio Science*, Vol. 45, No. 6, 2010.

[6] Sastry, A., "Performance of Hybrid Error Control Schemes of Satellite Channels," *Institute of Electrical and Electronics Engineers Transactions on Communications*, Vol. 23, No. 7, 1975, pp. 689–694.

[7] De Cola, T., Ernst, H., and Marchese, M., *Application of Long Erasure Codes and ARQ Schemes for Achieving High Data Transfer Performance Over Long Delay Networks*, Springer US, Boston, MA, 2008, pp. 643–656.

[8] Kuhn, N., Lochin, E., Lacan, J., Boreli, R., and Clarac, L., "On the impact of link layer retransmission schemes on TCP over 4G satellite links," *International Journal of Satellite Communications and Networking*, Vol. vol. 33, No. 1, 2015, pp. pp. 19–42.

[9] Tauran, B., Lochin, E., Lacan, J., Arnal, F., Gineste, M., Clarac, L., and Kuhn, N., "Making H-ARQ suitable for a mobile TCP receiver over LEO satellite constellations," , 2017.

[10] Braden, R., "Requirements for Internet Hosts - Communication Layers," RFC 1122 (INTERNET STANDARD), Oct. 1989. URL `http://www.ietf.org/rfc/rfc1122.txt`, updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864.

[11] Judd, G., "Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter," *Networked Systems Design and Implementation*, 2015, pp. 145–157.

[12] Chen, J., Gerla, M., Lee, Y. Z., and Sanadidi, M., "TCP with delayed ack for wireless networks," *Ad Hoc Networks*, Vol. 6, No. 7, 2008, pp. 1098 – 1116.

[13] Wood, L., Pavlou, G., and Evans, B., "Effects on TCP of routing strategies in satellite constellations," *Institute of Electrical and Electronics Engineers Communications Magazine*, Vol. 39, No. 3, 2001, pp. 172–181.

[14] Ahmad, R. A., Lacan, J., Arnal, F., Gineste, M., and Clarac, L., "Enhanced HARQ for Delay Tolerant Services in Mobile Satellite Communications," *The Seventh International Conference on Advances in Satellite and Space Communications 2015*, Barcelona, ES, 2015, pp. pp. 1–6.

[15] Kuhn, N., Natarajan, P., Khademi, N., and Ros, D., "Characterization Guidelines for Active Queue Management (AQM)," RFC 7928 (Informational), Jul. 2016. URL `http://www.ietf.org/rfc/rfc7928.txt`.

[16] Chotikapong, Y., Cruickshank, H., and Sun, Z., "Evaluation of TCP and Internet traffic via low Earth orbit satellites," *Institute of Electrical and Electronics Engineers Personal Communications*, Vol. 8, No. 3, 2001, pp. 28–34.

[17] Ha, S., Rhee, I., and Xu, L., "CUBIC: A New TCP-friendly High-speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, Vol. 42, No. 5, 2008, pp. 64–74.

[18] Allman, M., Paxson, V., and Blanton, E., "TCP Congestion Control," RFC 5681 (Draft Standard), Sep. 2009. URL `http://www.ietf.org/rfc/rfc5681.txt`.

[19] Wood, L., "SaVi: satellite constellation visualization," *First Annual Centre for Communication Systems Research Research Symposium*, 2011.

[20] Perez-Fontan, F., Vazquez-Castro, M. A., Buonomo, S., Poiares-Baptista, J. P., and Arbesser-Rastburg, B., "S-band LMS propagation channel behaviour for different environments, degrees of shadowing and elevation angles," *Institute of Electrical and Electronics Engineers Transactions on Broadcasting*, Vol. 44, No. 1, 1998, pp. 40–76.

[21] Fontan, F. P., Vazquez-Castro, M., Cabado, C. E., Garcia, J. P., and Kubista, E., "Statistical modeling of the LMS channel," *Institute of Electrical and Electronics Engineers Transactions on Vehicular Technology*, Vol. 50, No. 6, 2001, pp. 1549–1567.

[22] Caini, C., and Firrincieli, R., “TCP Hybla: a TCP enhancement for heterogeneous networks,” *International journal of satellite communications and networking*, Vol. 22, No. 5, 2004, pp. 547–566.

[23] Aggarwal, A., Savage, S., and Anderson, T., “Understanding the performance of TCP pacing,” *Conference on Computer Communications. Nineteenth Annual Joint Conference of the Institute of Electrical and Electronics Engineers Computer and Communications Societies*, Vol. 3, 2000, pp. 1157–1165.

[24] Sallantin, R., Baudoin, C., Chaput, E., Arnal, F., Dubois, E., and Beylot, A.-L., “Initial spreading: A fast start-up tcp mechanism,” *Local Computer Networks (LCN), 2013 Institute of Electrical and Electronics Engineers 38th Conference on*, Institute of Electrical and Electronics Engineers, 2013, pp. 492–499.

[25] Altman, E., Avrachenkov, K., and Barakat, C., “TCP in presence of bursty losses,” *Performance Evaluation*, Vol. 42, No. 2, 2000, pp. 129 – 147.

[26] Paxson, V., Allman, M., Chu, J., and Sargent, M., “Computing TCP’s Retransmission Timer,” RFC 6298 (Proposed Standard), Jun. 2011. URL `http://www.ietf.org/rfc/rfc6298.txt`.