# TCP Performance Optimization for Handover Management for LTE Satellite/Terrestrial Hybrid Networks

Michael Crosnier, Riadh Dhaou, Fabrice Planchou, André-Luc Beylot

# TCP performance optimization for handover Management for LTE satellite/terrestrial hybrid networks

Michael Crosnier*†, Riadh Dhaou†, Fabrice Planchou* and Andre-Luc Beylot†
*Astrium, 31 avenue des cosmonautes, 31402 Toulouse Cedex 04, France
†University of Toulouse, IRIT-ENSEEIHT, 2 rue Charles Camichel,BP7122, 31071 Toulouse Cedex 7, France

***Abstract*—Long Term Evolution tends to become the next generation of civil security networks. In order to meet the safety user requirements, preference is given to an integrated satellite and terrestrial architecture. It provides a widespread connectivity built on the large satellite coverage. In this paper, we focus our works on the real behavior of TCP protocols, implemented in on-the-shelf devices, during a handover within the hybrid architecture. We aim to improve an optimized handover procedure by taking into account the real implementations that differ from simulation tools and RFCs. In order to highlight and solve the problematic behaviors of TCP protocol due to a S1 handover with an SGW and MME relocation, a test is set up with Linux virtual machines for TCP application linked to a NS3 LTE network simulatior.**

***Index Terms*—LTE; handover; Linux TCP; hybrid satellite/terrestrial network.**

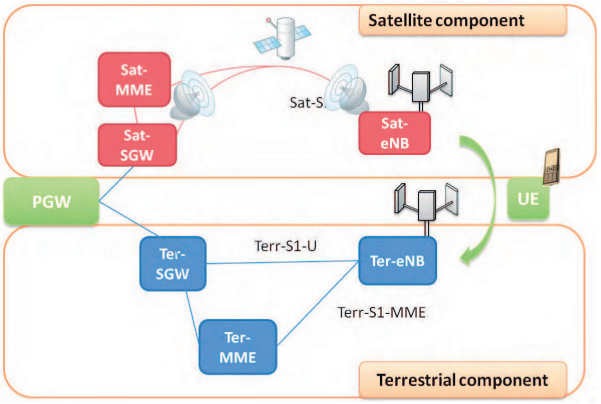


Fig. 1. Hybrid LTE architecture

## I. Introduction

Long Term Evolution (LTE) has eclipsed other competitors for the race for the next generation of mobile communication. We foresee that a LTE-like network will be deployed for the next generation of civil security networks. This paper carries on the attempt in [1] to optimize the handover procedure in a hybrid satellite and terrestrial LTE architecture. A LTE network solution that meets the civil security requirements has been designed by adding a satellite component to the common terrestrial architecture as in [2]. The terrestrial network will be deployed in densely populated area and this type of deployment will be too costly for an extensive coverage. Therefore, we propose to deploy autonomous LTE base stations (eNBs) that are backhauled with a satellite link. These satellite eNBs may temporarily be set-up in destroyed or isolated areas. However, the LTE specifications are not adapted to satellite systems since they consider the core network as a high speed and low delay network. They narrow the QoS challenge to the radio interface. As depicted in the Fig.1, the architecture is split into two components. The first one is a common terrestrial network and the second one is a LTE network with a satellite backhaul. A mobility management entity (MME) -in charge of the control plane-, a serving gateway (SGW) -in charge of the user plane- and the eNBs are dedicated to the satellite component. This distinction between core entities from the satellite and terrestrial components provides the opportunity to optimize procedures and protocols on the satellite interface between eNBs and core network entities (S1 interface). Tracking area and location management may be adapted to the satellite link as well as mobility procedures which involve the satellite component modifying the terrestrial LTE specifications to the minimum extent.

In [1], we have decided to tailor one specific procedure that suffers from the satellite integration. The transport and application implementations will not be modified by the optimization, instead of [3]. A handover from a satellite eNB to a terrestrial eNB particularly affects the TCP application performance because of the satellite system. The large difference of delay between the satellite and the terrestrial S1-interface is detrimental to TCP during the handover. This handover is depicted in [4]. During this procedure several problems are raised and there solutions are described in [1]. The main problem, depicted in [1], is the creation of forwarding tunnels in order to ensure a lossless handover. During a S1 handover between a satellite and a terrestrial eNb, a relocation of SGW and MME must be performed since core entities are different between the two components. As described in the Fig.2, indirect forwarding tunnels are set up during the handover preparation. After the handover command reception by the satellite eNb, all packets received by the satellite eNb will be sent to the satellite SGW (Fig.2, A) and transferred to the terrestrial eNb (Fig.2, C) thanks to the intermediate tunnel between the satellite and terrestrial SGW (Fig.2, B). The data path resulting from this forwarding mechanism entails a detrimental back and forth on the satellite interface whereas satellite resources are limited

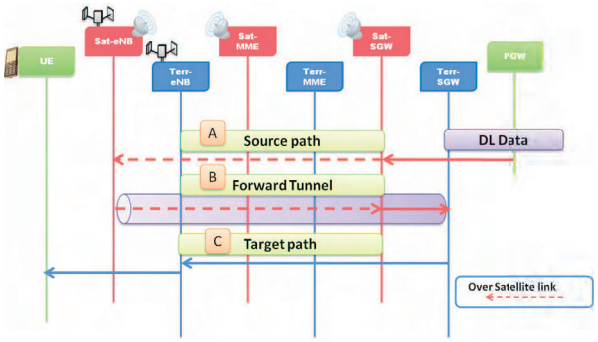


Fig. 2. Forward Problem

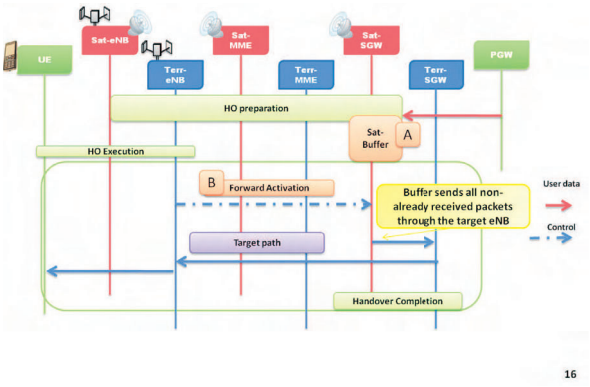


Fig. 3. LTE forward optimization

and expensive. In addition, this leads to double the delay during the handover duration whereas the new eNb is located in the common terrestrial LTE component.

The proposed solution in [1] is to avoid the back and forth on the satellite interface using the fact that the satellite SGW will unnecessarily forward twice the same packet. Consequently, after the handover completion, the packets that are not already received by the user equipment (UE) will be retransmitted to the terrestrial eNB from the satellite SGW thanks to tunnels with the intermediate terrestrial SGW. This solution leads to the loss of all the packets which are being transferred over the satellite link (S1-User Interface). To prevent this, the GTP packets, that are not received by the UE, are resent. Since GTP-U does not perform any control ([7]), the Sat-SGW needs to allocate a buffer during the handover preparation and to store all the packets which are received in a time period equal to the satellite delay plus the handover execution duration and the transmission time over the radio interface (Fig.3, A). In [1], this delay needs to be an accurate estimation, since an improper estimation of this delay will trigger detrimental TCP congestion mechanisms. Indeed, an undersized or oversized estimation will respectively lead to packet losses or packet duplications. Therefore [1] prescribes a slightly oversized buffer combined with a duplication avoidance mechanism. The Sat-eNB will inform the Sat-SGW of the last received GTP sequence number for each forwarded bearer thanks to the handover command/confirm message ([5]). As explained in [1], the handover preparation does not systematically lead to the immediate handover command transmission to the UE (Execution Handover Decision). Consequently, the forward mechanism cannot start during the handover preparation. A new GTP-C message is created between the Sat-MME and the Sat-SGW, named Data Forward Activation (Fig.3, B). When the Sat-SGW receives this message, it will begin sending the data stored in the forwarding satellite buffer. This message consists of GTP sequence numbers of the different bearers that have been already received by the UE. The Sat-SGW will discard all packets within the satellite buffer with a lower sequence number in order to avoid packet duplication. The Sat-SGW will inform the Sat-eNB of the last GTP-U packet for each forwarded bearer sending an end-marker GTP message at the satellite buffer creation.

In [1], we have performed simulation with NS3 and unfortunately the TCP implementation is really basic and does not reflect the real behavior of TCP in current implementations. As a consequence, the optimization proposed in [1] need to be refined and adapted to the real behavior of TCP. The remaining of this paper is organized as follows. In section II, the differences between the RFC, NS3 and real implementation will be explained. In section III, we will depict the modification of the optimization in order to be tailored to real TCP applications and, in section IV, we will describe the emulation architecture and highlight the benefits of the new optimized procedure.

## II. TCP IMPLEMENTATION ISSUES

The NS3 TCP implementation is very basic. Firstly, only old TCP versions exist in NS3, RFC 793, Tahoe, Reno and NewReno whereas the default TCP in Linux is TCP Cubic. Secondly, several options or optimizations have not been implemented in NS3 whereas they are used by default in Linux TCP and they are necessary for satellite communications ([8]).

### A. RFC and NS3 simulation

The main problem in NS3 simulation is the impossibility to use widespread TCP options and particularly these three options: window scaling, selective Acknowledgment (SACK) and timestamps.

*1) Window scaling:* This option becomes necessary and vital for TCP performance in case of satellite communication. The congestion window reaches its full size before the first ACK arrives on long delay network. This TCP option provides flexibility for the window size. Instead of the fix 16-bits size, the window can reach a 32-bits. The value of the option corresponds to a number of right-shifts operated on the awnd (with a maximum of 14 shifts). For instance, without this option the maximum bit rate is limited to around 110kB/s with a RTT equal to 600ms.

*2) Selective Acknowledgement:* As the common cumulative acknowledgment accepts only on retransmission during a RTT, the recovery from packet losses is inefficient. The SACK option notifies the sender up to 4 blocks of received segments. This option increases the recovery speed in long delay network.

*3) Timestamps:* This option adds time information in each packet in order to have a more accurate measure of the RTT. This is beneficial to the calculated RTO. This functionality can also be used to detect reordering. This measure accuracy is particularly necessary with high bandwidth delay product path.

### B. RFC and real implementation

In our study, we focus on the Linux implementations. [6] explains the specificities of Linux implementation. The Linux implementation is fully compliant with the three options previously depicted. Yet, some change has been decided in congestion control mechanisms from the RFCs.

*1) Congestion window management:* The congestion window is different from common congestion window depicted in RFCs. It is based on packets. The congestion window is compared with the currently outstanding segments ($P_{in-flight}$ in eq.1). For instance, packets that have been acknowledged by SACK do not count in the congestion control management.

$$P_{in-flight} = P_{sent} - P_{SACK} - P_{lost} + P_{retransmitted} \quad (1)$$

In RFCs, we only consider packets that are not already acknowledged. In addition, the first time TCP detects a duplicate or a selective acknowledgement, it enters in the disordering mode. In this mode, the TC congestion a window is kept constant i.e. for each new acknowledgment a new packet is sent. The disordering mode lasts a number of duplicate acknowledgments calculated by the reordering estimator in function of unnecessary retransmission previously observed. The default value is three duplicate acknowledgments. After the disordering state, the Linux TCP enters in the Recovery state. Instead of the Fast Retransmit, the Linux implementation reduces the congestion window by one segment each time it receives two acknowledgments until the congestion window reaches the slow start threshold.

*2) RTO:* The Linux implementation is little bit different from RFCs. It takes into account a RTO calculation problem. In case of a sudden decrease of RTT, which happens during the studied handover, the RTT variance significantly increases and the RTO will be overestimated. In addition, the minimum of the RTO is set to 200ms in Linux whereas its value is 1s for RFCs ([6]).

*3) Burst avoidance:* Linux implements a burst avoidance mechanism. When a large part of the user window is acknowledged, a large burst of data may be sent and create congestion. In order to prevent this burst, the congestion window will be reduced in order to only allow three segments to be transmitted. If the new congestion window is greater than the slow start threshold the Cubic congestion avoidance is used, otherwise it will restart the slow start algorithm from its new congestion window.

## III. Handover optimization issues with real TCP implementation

Because of the significant differences between NS3, RFCs and Linux implementations, the relevance of the results in [1] has to be questioned. As a consequence the real TCP behavior has to be studied with a real Linux implementation. There are two main mechanisms that have deep impact in our understanding of the TCP behavior. The window scaling that will increase a lot the congestion in satellite interface and misbehavior of TCP due to the forward mechanism.

### A. Window scaling

The window scaling is an essential mechanism to TCP over high speed and long delay network. Thanks to the window scaling, the maximum data rate are significantly increased. From this perspective, congestion in the satellite interface is limited and therefore, the buffer management turns out to be really simple. Firstly, the data rate, increased due to window scaling, will create congestion in the satellite device queue. Consequently, the duration estimation during which the buffer has to store the packets becomes more complex to find out. This duration needs to be greater in order to avoid packet losses and offset the time that the packets spent on the satellite queue. If we need a more accurate estimation in order to preserve resource in the satellite device, we can measure this delay in the satellite queue and calculate the maximum time during which packets will be stored ($T_{stored}$). The $T_{stored}$ is calculated as detailed in eq.2 where $T_{queue}$ is the elapsed time in queue, $T_{trans}$ is the transmission time and $\delta$ is an additional offset.

$$T_{stored} = T_{queue} + T_{trans} + \delta \quad (2)$$

Yet, this time limitation will be used there in order to restrict the buffer size in case of handover problems or failures. Indeed, the handover command will be sent just after the buffer creation. As it is a handover control message, it has the priority on the satellite interface. If the handover decision algorithms are well configured, the latency will be almost the same as the transmission delay and no packets will be outdated in the buffer. If a handover failure occurs or the handover execution happens a long time after the handover preparation, due to the new decision algorithms explained in [1], some packets may be outdated. Yet, this behavior will not be frequent and will not need a complex mechanism. Therefore, the buffer store duration will be set to a high value in order to limit this (to above 1s). A second problem is raised because of the congestion in the satellite queue. During the handover preparation, the Sat-SGW creates a buffer in order to store packets that will be not received by the UE after the handover execution. Yet, as explained previously, the handover will be executed just after the handover command and so after the satellite delay plus the handover execution delay. As a consequence, a major part of packets in the satellite device queue during the creation of the buffer will be lost. They will be transmitted on the satellite interface after the handover command message and there will not be stored in the buffer. To avoid this important loss, just after the creation of the buffer, the GTP packets stored in the satellite device queue with a TEID corresponding to the bearer, involved in the forward mechanism, will be copied in the buffer. Secondly, the

data rate increase implies a new congestion management. Two possibilities are proposed, the congestion is taken into account in the buffer or in the terrestrial eNB. In both solutions, after the path switch, two PDCP queues are set up in the terrestrial eNB during the handover (Fig.4 ). The first will handle packets that are forwarded, in our optimization these packets come from the satellite buffer (Fig.4, B) and the second queue will handle the non-forwarded packets originated from the normal terrestrial path after the path switch (Fig.4, C). The reordering of packets is ensured by the presence of this two queues. The forwarded queue is prioritized. The enqueued packets in the non-forwarded queue are sent only after the last forwarded packet has been transmitted. The knowledge of the last packet is transmitted from the satellite SGW to the Terrestrial eNB thanks to GTP-U packets with the End-Marker set to one in GTP option. Three of these packets will be send in order to maximize the probability to notify the eNB of the forwarding end. If the three End Marker messages are lost, a timer will trigger the transmission of non-forwarded packets. The difference between the two solutions is the time when the path switch is triggered; either just after the forward activation or after the buffer is almost empty. The first solution comes down to immediately modify the user path after the Forward Activation message. The GTP packets will no longer cross the satellite SGW (Fig.4, A) but the terrestrial SGW (Fig.4, C). Consequently the congestion will occur in the PDCP layer queues located in the Terrestrial eNB. The Terrestrial PDCP layer may not be well adapted to receive such an amount of forwarded data that may results in packet losses. Another issue in this solution is the data rate of the transmission of GTP packets by the buffer. We propose to send these packets to the maximum authorized data rate for the bearer (MBR). Therefore, losses in the two queues in the PDCP layer of the eNB are exacerbated. However this solution limits the duration of the handover completion that would be detrimental in case of frequent handover and increase congestion on the non-forwarded queue. The second solution tries to avoid the congestion problem in the eNB queues. Thus, we shift the congestion problem into the buffer management. The idea is to perform the path switch only when the size of the buffer will be similar to the amount of forwarded packets during a common terrestrial handover. After the buffer size reaches this threshold, a new control message will be sent in order to trigger the path switch. This solution needs an additional control message. However, it will be complex to ensure the buffer size decrease and the time to reach the threshold may be long and detrimental in case of frequent handovers. As a consequence the first solution seems to be more adapted to our architecture.

### *B. Large window acknowledgement*

After the handover execution, a large window of the TCP congestion will be acknowledged. Indeed, the first acknowledgement will be received by the correspondent node long before the last sent packets through the satellite link. As a consequence, in the NS3 simulation in [1], there is a burst of data that create congestion problems. In the linux implementation, a maximum of three packets is sent for each received acknowledgement and the congestion windows is reduced to the number of in-flight packets plus three. The TCP protocol may restart a slow start if the new congestion window value is below the threshold in [6]. During the slow start phase, the congestion will increase whereas we acknowledge only the forwarded packets, consequently the non-forwarded queue size will grow rapidly and will create a burst of losses. In order to avoid this we may link the management of the two eNB queues and packets in the forwarded queue will be discarded instead of packets in the non-forwarded queue. However this mechanism is complex and may be not necessary because the packets from the buffer will be sent at the maximum bit rate and as a consequence congestion in the forwarded queue are likely to happen. This congestion will reduce the congestion in the non forwarded queue.

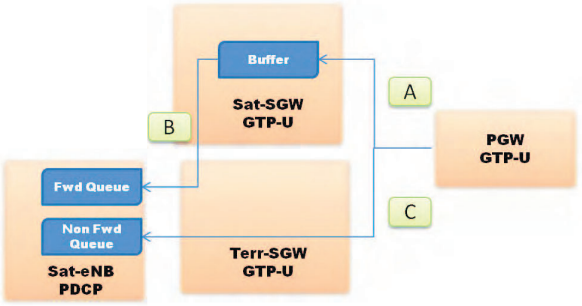


Fig. 4. Congestion management

## IV. TCP EMULATION AND LTE SIMULATION

In order to observe the efficiency of the new optimization mechanism, we have performed emulation. The main problem lies in the fact that the NS3 TCP implementation is too simple and does not reflect the behavior of real TCP implementations with a satellite component ([9]). Consequently, we have used the real Linux TCP stack instead of NS3 TCP. The default implementation is the TCP Cubic. This TCP is emulated in two Linux virtual machines that represent the PGW and UE application parts (Fig.5). The TAP device links the Linux device to the NS3 simulator in real time simulation mode. All specific LTE protocols, used in the handover, are implemented NS3 as in [1]. The control messages involved in the handover procedure has been implemented as well as the GTP-U protocol. In order to simplify the implementation and allow a real time simulation, the air interface protocols has been simplified.

The satellite interface is characterized by a latency of 300 ms and a bit rate of 512 kb/s. The LTE air interface has a latency of 4 ms and a bit rate of 1 Mb/s. In order to observe the congestion windows, TCP probe is set up during the simulation. The TCP application is an extensive file transfer with the FTP protocol thanks to wget. In order to have comparable results, we trigger the handover at the expiration of a timer. This timer starts at the reception of the first FTP packet and ends after a fixed duration. We

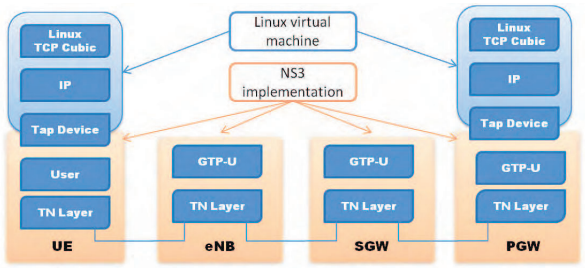


Fig. 5. Emulation architecture

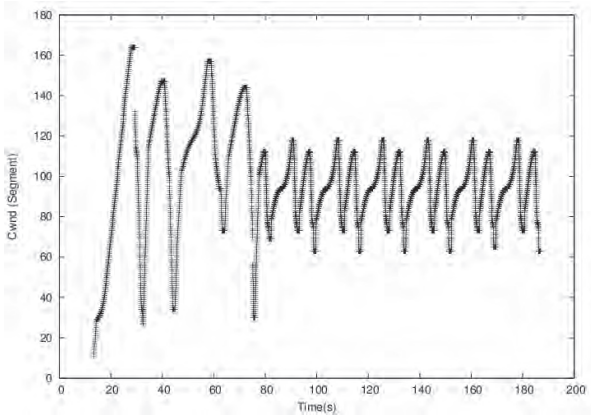


Fig. 6. Cwnd with the forward optimization during Fast retransmit

performed several simulations in order to observe the TCP behavior when the handover occurs at different TCP states (Congestion Avoidance, Fast Retransmit). In order to assess the performance of our optimization, we compare the simulation results of our optimization with those of a handover without forwarding mechanism. In Fig.6, the handover is triggered during a Fast Retransmit phase that correponds to the CWND state ([6]) in the Linux implementation whereas the Fig.7 and 8 respectively show the TCP congestion window for an handover triggered during the Congestion Avoidance phase with the new optimization procedure and without any forwarding mechanism.

Linux TCP cubic has a good behavior with the new optimization procedure during the congestion avoidance phase (Fig.7) as well as during the Fast Retransmit phase (Fig.6) compared to the behavior of the TCP congestion window without any forwarding mechanism. Indeed, in Fig.8, TCP has to retransmit a large part of the TCP congestion window after the handover execution as these packets have been lost. Yet the difference between the two mechanisms is not really substantial. For instance, the FTP download lasts 188 s in the Fig.7 and 189 s Fig.8. Yet this difference is exarcebated in case of frequent handover and may jeopardize TCP flows and in this case the forwrd mechanism is necessary.

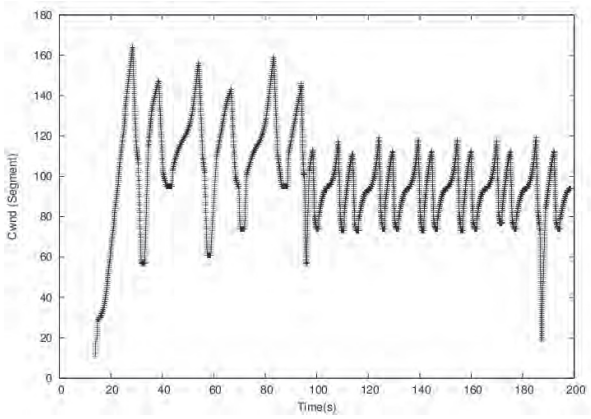


Fig. 7. Cwnd with the forward optimization during congestion avoidance

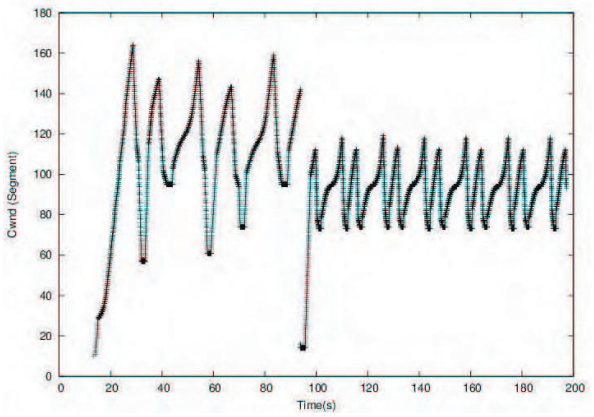


Fig. 8. Cwnd without the forward optimization during congestion avoidance

## V. CONCLUSION

The benefits of new forward mechanism turn out to be substantial during frequent handovers. The new optimization takes into account the real behavior of TCP linux implementation with a more accurate buffer management. In order to assess the enhancement, other emulations need to be performed, modifying several parameters such as the satellite delay, the satellite and air interface data rate.

## REFERENCES

[1] Crosnier, M.; Planchou, F.; Dhaou, R.; Beylot, A.; "Handover Management Optimization for LTE Terrestrial Network with Satellite Backhaul"; Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd , pp.1-5, 15-18 May 2011

[2] Corbel, E.; Buret, I.; Gayrard, J.-D.; Corazza, G.E.; Bolea-Alamanac, A.; "Hybrid Satellite & Terrestrial Mobile Network for 4G : Candidate Architecture and Space Segment Dimensioning"; Advanced Satellite Mobile Systems, 2008. ASMS 2008. 4th , pp.162-166, 26-28 Aug. 2008

[3] Amadeo, M.; Araniti, G.; Iera, A.; Molinaro, A.; "A Satellite-LTE Network with Delay-Tolerant Capabilities: Design and Performance Evaluation"; Vehicular Technology Conference (VTC Fall), 2011 IEEE , pp.1-5, 5-8 Sept. 2011

[4] 3GPP TS 23.401 GPRS enhancements for Evolved Universal Terrestrial Radio Access Network (E- UTRAN) access, version 11.1.0, Mars 2012.

[5] 3GPP TS 36.331 "Radio Resource Control (RRC); Protocol specification (Release 10)", version 10.5.0, Mars 2012.

[6] Sarolahti, P.; Kuznetsov, A.;"Congestion Control in Linux TCP"; Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, p.49-62, June 10-15, 2002

[7] 3GPP TS 29.281 GPRS Tunnelling Protocol User plane (GTPv1-U), version 11.2.0, Mars. 2012.

[8] Borman, D.; Braden, R.; Jacobson, V.; "TCP Extensions for High Performance"; RFC 1323, May 1982.

[9] Network Simulator 3, http://www.nsnam.org/