5718 Final Report

1st Abhishek Jha
1st grad student in Department of ECE
University of Florida
Gainesville, USA
abhishek.jha@ufl.edu

2nd Rui GUO
1st grad student in Department of ECE
University of Florida
Gainesville, USA
guor@ufl.edu

3rdWenxuan BAO
1st grad student in Department of ECE
University of Florida
Gainesville, USA
wenxuanbao@ufl.edu

***Abstract*—This report assesses Random Access usage in a Satellite network. Contention Resolution Diversity Slotted ALOHA (CRDSA) scheme, introduced in the Digital Video Broadcasting - Return Channel via Satellite standard (DVB-RCS2) is chosen as the baseline and TCP New Reno is the exchange protocol followed. Through a detailed simulation based on the Satellite NS3 simulator, this work evaluates the Transmission-Receiver characteristics and the time elapsed in sending packets for the aforementioned scenario.**

Keywords— Random access channel; TCP New Reno; DVB-RCS2; M2M

I. INTRODUCTION

This project aims at simulating the Contention Resolution Diversity Slotted ALOHA protocol over a satellite network. CRDSA is a simple yet effective improvement over slotted ALOHA. It relies on MAC bursts repetition and on interference cancellation to increase the throughput of a classic slotted ALOHA access scheme. This improvement permits to achieve a throughput up to T= 0.55, whereas slotted ALOHA is capable of providing T= 0.36. In this project we have used 3-CRDSA protocol i.e. 3 replicas of a single packet.

DVB-RSC2 is a mature open source satellite communication standard with highly efficient bandwidth management. This make it a cost-efficient alternative solution for many users. It also provides an established foundation for further satellite communications research. It uses a satellite linked connection for the return (up-link) channel in addition to the standard down-link channel.

TCP New Reno improves retransmission during the fast-recovery phase of TCP Reno. During fast recovery, to keep the transmit window full, for every duplicate ACK that is returned, a new unsent packet from the end of the congestion window is sent. For every ACK that makes partial progress in the sequence space, the sender assumes that the ACK points to a new hole, and the next packet beyond the ACKed sequence number is sent. Because New Reno can send new packets at the end of the congestion window during fast recovery, high throughput is maintained during the hole-filling process, even when there are multiple holes, of multiple packets each.

This projects aims at simulating the transmission of data packets from satellite to ground station receivers using the TCP New Reno protocol over a CRDSA random access channel. In order to realize the same, satellite network simulator (SNS3) is utilized. SNS3 possesses satellite configurations and data libraries which enable to simulate the satellite scenarios in a real world setting. As part of this project, the satellite is connected with 14 nodes which are ground receiver terminals over two channels.

Identify applicable funding agency here. If none, delete this.

II. REFERENCE PAPER

A. Information about reference paper

First, here is the citation of reference paper: Abdelsalam A, Roseti C, Zampognaro F. TCP performance for Satellite M2M applications over Random Access links[C]//2018 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2018: 1-5.

Second, the abstract of this paper will be introduced below. Contention Resolution Diversity Slotted ALOHA (CRDSA) random access scheme, introduced in the DVBRCS2/NG standard, is specifically designed to support the transfer of a variety of traffic profiles that Machine to Machine (M2M) and Internet of Things (IoT) applications may generate. Protocols for M2M and IoT has recently received substantial upgrades, but still mainly designed with an underlying terrestrial network in mind. If clusters of sensor nodes exchange data via satellite terminals toward a sink via short-lived TCP/IP connections, many shortcomings may incur. This work aims at investigating in details the behavior for such data transfer in presence of a shared random access channel, and the advantages when using a new TCP version specifically designed for satellite links, namely TCP Wave. In particular, through a detailed simulation campaign based on the NS3 simulator, this work assesses the completion time of data delivery for M2M elastic traffic via CRDSA satellite random access, when standard TCP (NewReno) or dedicated satellite TCP (Wave) is adopted[1].

B. Description of the analysis in the reference paper

In the paper authors created a system with a finite set of nodes transmitting MQTT-like messages over a Random Access (RA) geostationary satellite channel. They assumed that each transmission super frame contains one frame with k slots and one RA block. DVB-RCS2 is a satellite transmission standard allowing for Dedicated access and Random access mechanisms for the shared use of return link[5-7]. It provides two different options as Random Access scheme: Slotted Aloha (SA) and Contention Resolution Diversity Slotted Aloha (CRDSA)[3-4]. In this paper, authors used CRDSA as Random

Access scheme in their scenario. The traffic arrival process for each RCST is determined by a set of MQTT-like publishers, sending messages to the broker through the RCST (gray circles) to the satellite channel, as pictured in Figure 1[1]. TCP

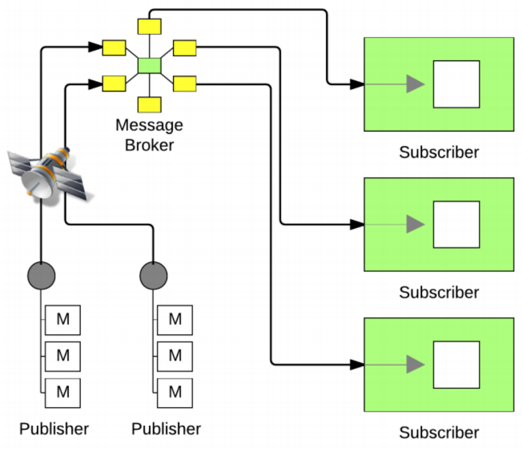


Fig. 1. Logical network topology

New Reno or TCP Wave is used in the paper at transport layer and can be alternately enabled[2]. Fig2[1] shows numerical values in use during the simulations. In this research,the length

Name	Value
TCP flavor	NewReno OR Wave
TCP MSS	486 bytes
TCP header options size	12 bytes
TCP/IP headers size	40 bytes
Neat bandwidth per slot	≈115 Kbit/s
MAC queue size	$\gg$ Bandwidth Delay Product
RA scheme	3-CRDSA
RA blocks per superframe	1
RA block duration	0.013 s
Slots per RA block	64
Gross slot size	188 bytes
Net slot size	182 bytes
Bandwidth	8012820 Hz
Roll off	0.2
Carrier spacing	0.3
One-way PHY delay	0.13 s

Fig. 2. System setup parameters

of the MQTT payloads, delivered through New Reno or Wave TCP connections, is randomly drawn from Pareto distributions having increasing average values, to cover most typical M2M payload lengths[1][8]. The results of this paper shows that how the use of TCP as transport protocol for M2M traffic through a satellite channel is feasible and offers in good performance.

III. Simulation Description

In this section, the simulation parameters and reasons are given, and finally the simulation results are listed. In our simulation, SNS3 is selected to be the simulator. The Satellite Network Simulator 3 (SNS3) is a satellite network module extension component of the Network Simulator 3 (NS-3) platform. SNS3 establishes a satellite network model of multi beam interaction. The satellite reference system consists of 72 European covered point beams, 5 gateways and Ka-band frequencies.

Because SNS3 is highly compatible with this simulation, so we choose SNS3 as the simulator. In this simulation, we used the CRDSA model and disabled CRA. The specific parameter settings will be described later. The parameter setting process is shown in Fig. 3.

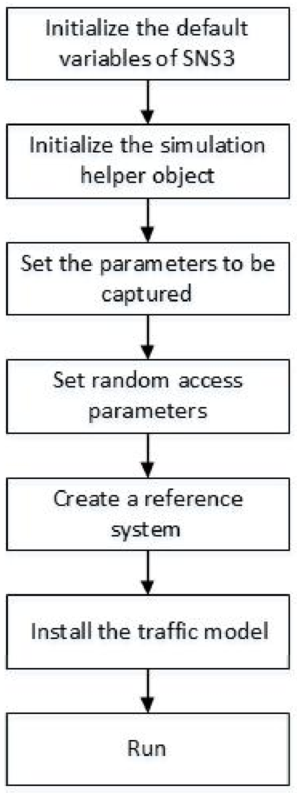

Initialize the default variables of SNS3
Initialize the simulation helper object
Set the parameters to be captured
Set random access parameters
Create a reference system
Install the traffic model
Run

Fig. 3. Parameter setting flowchart

A. Initialize the default variables of SNS3

In this part, variables describe the simulation topology. Beam ID is used to extract the configuration vector of a given satellite beam ID. In this simulation, beam is used as the carrier of satellite signal. We set beamID as 1. In order to reduce the simulation scale, we set the number of user terminals of each beam to 1 and the number of users of each user terminal to 1. The length of the sent packet is set to 20. The interval is set to 10ms, and the simulation duration is 1s. These are the parameters that need to be set for simulation. The specific parameters are shown in TABLE I.

B. Initialize the simulation helper object

NS3 is a system based on C++ classes. In NS3, every C++ rule can still be used, such as objects can be declared and

Name	Value
beamId	1
endUsersPerUt	1
utsPerBeam	1
packetSize	20
interval	0.01
simLength	1

TABLE I
DEFAULT VARIABLES

instantiated. The concepts of C++ abstraction, inheritance, encapsulation and polymorphism are often used in NS3. In this simulation, a helper object should be created before the simulation. Simulation helper will run through the whole simulation.

C. *Set the parameters to be captured*

In this part, packet capture is set to enabled, and then set TCP to new reno, enable random access with CRDSA , set random access collision model and at last set dynamic load control parameters. The specific parameters are shown in TABLE II.

Name	Value
PacketTraceEnabled	true
SocketType	TcpNewReno
RandomAccessModel	RA_MODEL _CRDSA
RaCollisionModel	RA_COLLISION _CHECK_AGAINST _SINR
EnableRandomAccess DynamicLoadControl	false
RandomAccessAverage NormalizedOfferedLoad MeasurementWindowSize	10

TABLE II
STRUCTURAL PARAMETERS

D. *Set random access parameters*

This part sets important parameters for this simulation. In this step, set maximum unique payload per block to 3, maximum conservative block accessed to 6, minimum Idle Block to 2, back off time in milliseconds to 250ms, back off probability to 10000, high load back off probability to 30000, number of instances to 3, average normalized offered load threshold to 0.5, default control randomization interval to 100ms, CRDSA signaling overhead in bytes to 5 and slotted Aloha signaling overhead in bytes to 3. The specific parameters are shown in TABLE III.

Name	Value
Maximum Unique Payload Per Block	3
Maximum Consecutive Block Accessed	6
Minimum Idle Block	2
Back Off Time In MilliSeconds	250
Back Off Probability	10000
High Load Back Off Probability	30000
Number Of Instances	3
Average Normalized Offered Load Threshold	0.5
Default Control Randomization Interval	100ms
CRDSA Signaling Overhead In Bytes	5
Slotted Aloha Signaling Overhead In Bytes	3

TABLE III
RANDOM ACCESS PARAMETERS

E. *Create a reference system*

Create a reference system by binding the parameters with the simulation helper object. It should be noted that at present, the satellite module only supports one reference system, called ”scenario 72”. This name is used to map the scene to the desired reference system profile. It can’t be named as other here, because any other name may cause fatal errors. Refer to TABLE IV for the function called and the parameters used.

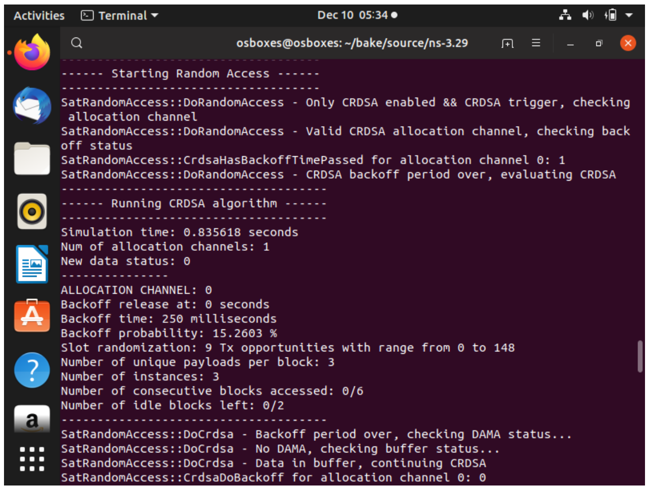
Function	Parameter
SetSimulationTime	simLength
SetUserCountPerUt	endUsersPerUt
SetUtCountPerBeam	utsPerBeam
SetBeamSet	beamId
CbrApplication::Interval	interval
CbrApplication::PacketSize	packetSize

TABLE IV
PARAMETERS ASSIGNMENT

F. *Install the traffic model and run simulation*

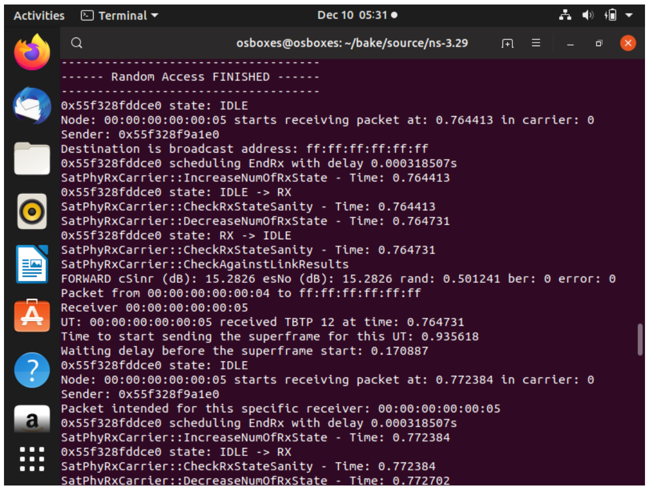
In this step, the simulation parameters are already set, call the InstallTrafficModel method in the SimulationHelper class to install a simple traffic model. In this simulation, set TCP as the transport layer protocol. After that, RunSimulation is used for simulation. And the simulation results are shown from Fig. 4 to Fig. 6.

It can be seen from the figure that we have realized the simulation of Random Access and CRDSA algorithm. Compared to the original paper, they also implemented the MQTT part, which is not included in our simulation. In addition, the original paper compared the duration in the case of different packet sizes, while we just set the packet size to a fixed value for simulation because we found that setting the packet size to a larger value will cause some errors.



```
------ Starting Random Access ------
------------------------------------
SatRandomAccess::DoRandomAccess - Only CRDSA enabled && CRDSA trigger, checking
 allocation channel
SatRandomAccess::DoRandomAccess - Valid CRDSA allocation channel, checking back
off status
SatRandomAccess::CrdsaHasBackoffTimePassed for allocation channel 0: 1
SatRandomAccess::DoRandomAccess - CRDSA backoff period over, evaluating CRDSA
-------------------------------------
------ Running CRDSA algorithm ------
-------------------------------------
Simulation time: 0.835618 seconds
Num of allocation channels: 1
New data status: 0
---------------
ALLOCATION CHANNEL: 0
Backoff release at: 0 seconds
Backoff time: 250 milliseconds
Backoff probability: 15.2603 %
Slot randomization: 9 Tx opportunities with range from 0 to 148
Number of unique payloads per block: 3
Number of instances: 3
Number of consecutive blocks accessed: 0/6
Number of idle blocks left: 0/2
-------------------------------------
SatRandomAccess::DoCrdsa - Backoff period over, checking DAMA status...
SatRandomAccess::DoCrdsa - No DAMA, checking buffer status...
SatRandomAccess::DoCrdsa - Data in buffer, continuing CRDSA
SatRandomAccess::CrdsaDoBackoff for allocation channel 0: 0
```

Fig. 4. Simulation Result-1



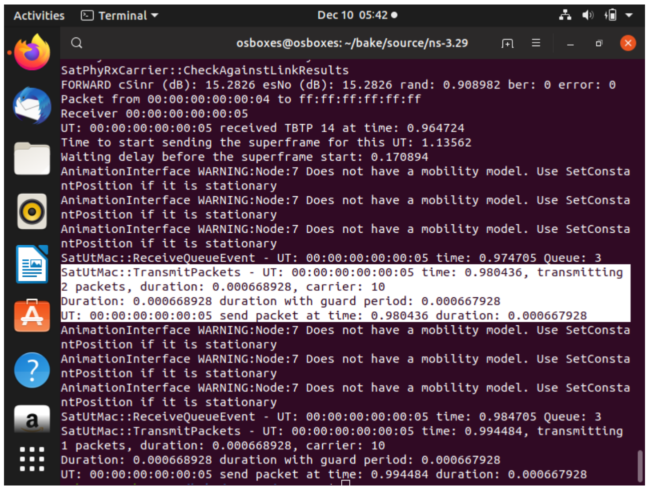
```
------------------------------------
------ Random Access FINISHED ------
------------------------------------
0x55f328fddce0 state: IDLE
Node: 00:00:00:00:00:05 starts receiving packet at: 0.764413 in carrier: 0
Sender: 0x55f328f9a1e0
Destination is broadcast address: ff:ff:ff:ff:ff:ff
0x55f328fddce0 scheduling EndRx with delay 0.000318507s
SatPhyRxCarrier::IncreaseNumOfRxState - Time: 0.764413
0x55f328fddce0 state: IDLE -> RX
SatPhyRxCarrier::CheckRxStateSanity - Time: 0.764413
SatPhyRxCarrier::DecreaseNumOfRxState - Time: 0.764731
0x55f328fddce0 state: RX -> IDLE
SatPhyRxCarrier::CheckRxStateSanity - Time: 0.764731
SatPhyRxCarrier::CheckAgainstLinkResults
FORWARD cSinr (dB): 15.2826 esNo (dB): 15.2826 rand: 0.501241 ber: 0 error: 0
Packet from 00:00:00:00:00:04 to ff:ff:ff:ff:ff:ff
Receiver 00:00:00:00:00:05
UT: 00:00:00:00:00:05 received TBTP 12 at time: 0.764731
Time to start sending the superframe for this UT: 0.935618
Waiting delay before the superframe start: 0.170887
0x55f328fddce0 state: IDLE
Node: 00:00:00:00:00:05 starts receiving packet at: 0.772384 in carrier: 0
Sender: 0x55f328f9a1e0
Packet intended for this specific receiver: 00:00:00:00:00:05
0x55f328fddce0 scheduling EndRx with delay 0.000318507s
SatPhyRxCarrier::IncreaseNumOfRxState - Time: 0.772384
0x55f328fddce0 state: IDLE -> RX
SatPhyRxCarrier::CheckRxStateSanity - Time: 0.772384
SatPhyRxCarrier::DecreaseNumOfRxState - Time: 0.772702
```

Fig. 5. Simulation Result-2

IV. CONCLUSIONS

A. Discussion on Elapsed Time and Simulation Tool

With the use of 3-CRDSA protocol, it was observed that for a packet size of 25 bytes, the average time elapsed from sending the packet over the CRDSA random access channel was about 0.000667 seconds. Signal to Interference Noise ratio was observed to be averaging 15.2 dB for the set values of parameters. Also, the average waiting delay for the User terminal to send to the satellite channel was 0.17 seconds.



```
SatPhyRxCarrier::CheckAgainstLinkResults
FORWARD cSinr (dB): 15.2826 esNo (dB): 15.2826 rand: 0.908982 ber: 0 error: 0
Packet from 00:00:00:00:00:04 to ff:ff:ff:ff:ff:ff
Receiver 00:00:00:00:00:05
UT: 00:00:00:00:00:05 received TBTP 14 at time: 0.964724
Time to start sending the superframe for this UT: 1.13562
Waiting delay before the superframe start: 0.170894
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
SatUtMac::ReceiveQueueEvent - UT: 00:00:00:00:00:05 time: 0.974705 Queue: 3
SatUtMac::TransmitPackets - UT: 00:00:00:00:00:05 time: 0.980436, transmitting
2 packets, duration: 0.000668928, carrier: 10
Duration: 0.000668928 duration with guard period: 0.000667928
UT: 00:00:00:00:00:05 send packet at time: 0.980436 duration: 0.000667928
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
SatUtMac::ReceiveQueueEvent - UT: 00:00:00:00:00:05 time: 0.984705 Queue: 3
SatUtMac::TransmitPackets - UT: 00:00:00:00:00:05 time: 0.994484, transmitting
1 packets, duration: 0.000668928, carrier: 10
Duration: 0.000668928 duration with guard period: 0.000667928
UT: 00:00:00:00:00:05 send packet at time: 0.994484 duration: 0.000667928
```

Fig. 6. Simulation Result-3

NS3 is a good simulator for discrete-event computer network. It can provide strong modules and easy to find tuition for how to use them. It is based on C++ classes so it is easy to read examples' codes. However, for our project, we need to build satellite network and modules in original NS3 are not suitable for this work.

So, we find SNS3 to build satellite network. SNS3 is built as an extension module to the NS3 network and it models a full interactive multi-spot beam satellite network with a geostationary satellite and transparent star (bent-pipe) payload. It implements a wide range of DVB-S2/RCS2 mechanisms which are required in our project. It is a dynamic system simulator, which works at a physical level burst resolution. Its use cases range from full network level performance to a single user Quality of Service verification. SNS3 is built to be modular and flexible to provide extensibility to different satellite communication networks and use cases. Due to its huge number of use cases, we find sat-random-access-crdsa-example.cc is suitable for our project and we follow this example to build our satellite network.

B. Difficulties

On further tuning of the model to observe the response for different packet sizes, such as altering the packet size to 100 bytes the code failed with signal error SIGIOT. Upon debugging it was observed that it failed due to an invalid attribute setting. This is part of a dependent data library which is used by satellite ns3. The data library has over 50 modules and it was unclear which one might have caused this error. Also being a relatively new framework, sns3 has limited catalog of debugging material available on the internet.

C. Final division of labor

First, we divided these project into 3 parts. Wenxuan BAo worked on Random access part, Abhishek Jha worked on TCP part and Rui Guo worked on DVB-S2/RCS2 part. But we found this way will not work during the integration testing phase. Random access part was supposed to be based on DVB-S2/RCS2 and the original NS3 is not suitable for our project. Abhishek Jha built a software environment for SNS3 and ran the simulation code. Rui Guo and Wenxuan BAO read the SNS3 API documents, looked up papers and finally completed the coding modification with Abhishek Jha. Wenxuan BAO wrote Demo presentation power point and summary. We worked on this report together.

D. *Future outlook*

In near future this model can be extended to implement MQTT based architecture which can have applications in simulating the scenarios based on IoT devices and M2M communications.

REFERENCES

[1] Abdelsalam A, Roseti C, Zampognaro F. TCP performance for Satellite M2M applications over Random Access links[C]//2018 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2018: 1-5.

[2] Abdelsalam A, Luglio M, Roseti C, et al. TCP Wave: A new reliable transport approach for future internet[J]. Computer Networks, 2017, 112: 122-143.

[3] R. De Gaudenzi and O. D. Herrero, “Advances in Random Access protocols for satellite networks,” in Satellite and Space Communications, 2009. IWSSC 2009. International Workshop on. IEEE, 2009, pp. 331–336.

[4] N. Celandroni, F. Davoli, E. Ferro, and A. Gotta, “On elastic traffic via Contention Resolution Diversity Slotted Aloha satellite access,” International Journal of Communication Systems, 2014.

[5] Salam A A, Luglio M, Roseti C, et al. A burst-approach for transmission of TCP traffic over DVB-RCS2 links[C]//2015 IEEE 20th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD). IEEE, 2015: 175-179.

[6] Meloni A, Murroni M. Random access in DVB-RCS2: Design and dynamic control for congestion avoidance[J]. IEEE Transactions on Broadcasting, 2013, 60(1): 16-28.

[7] Angelone M, Ginesi A, Re E, et al. Performance of a combined dynamic rate adaptation and adaptive coding modulation technique for a DVB-RCS2 system[C]//2012 6th Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC). IEEE, 2012: 124-131.

[8] Duan S, Shah-Mansouri V, Wang Z, et al. D-ACB: Adaptive congestion control algorithm for bursty M2M traffic in LTE networks[J]. IEEE Transactions on Vehicular Technology, 2016, 65(12): 9847-9861.

TCP performance for Satellite M2M applications over Random Access links

A. Abdelsalam, C. Roseti, and F. Zampognaro*
University of Rome “Tor Vergata”, Department of Electronics Engineering
00131, Rome, Italy
e-mail: zampognaro@ing.uniroma2.it

***Abstract*—Contention Resolution Diversity Slotted ALOHA (CRDSA) random access scheme, introduced in the DVB-RCS2/NG standard, is specifically designed to support the transfer of a variety of traffic profiles that Machine to Machine (M2M) and Internet of Things (IoT) applications may generate. Protocols for M2M and IoT has recently received substantial upgrades, but still mainly designed with an underlying terrestrial network in mind. If clusters of sensor nodes exchange data via satellite terminals toward a sink via short-lived TCP/IP connections, many shortcomings may incur. This work aims at investigating in details the behavior for such data transfer in presence of a shared random access channel, and the advantages when using a new TCP version specifically designed for satellite links, namely TCP Wave. In particular, through a detailed simulation campaign based on the NS3 simulator, this work assesses the completion time of data delivery for M2M elastic traffic via CRDSA satellite random access, when standard TCP (NewReno) or dedicated satellite TCP (Wave) is adopted.**

I. INTRODUCTION

A relevant standardization effort on various aspects concerning the Internet of Things (IoT) and Machine to Machine (M2M) communications is carried on by dedicated standardization bodies (e.g., IETF), such as the Constrained RESTful Environments (CoRE) working group[1]. When the number of sensors and receiver for IoT/M2M communication is high, a Publisher/Subscriber (pub-sub) model is typically adopted, based on a design pattern where the senders, namely *publishers*, send data messages to specific receivers, namely *subscribers*, relying optionally on intermediate entities (aggregators or *brokers*).

MQTT[2] (Message Queue Telemetry Transport) is a M2M/IoT designed as an extremely lightweight pub-sub message exchange protocol. MQTT is particularly suitable for large sensors networks and it is a widely adopted standard. MQTT uses a broker as intermediate entity, responsible of collecting the interest of subscribers on available data sources, and for delivering new messages coming from publishers on subscribed topics. Its use in this configuration is already consolidated in a wide number of cases, such as home automation or other scenarios with a small number of devices involved, up to healthcare scenarios, or wide areas sensor networks with a large number of devices involved [1]. MQTT represents a more and more relevant portion of the Internet traffic related to IoT/M2M, and relies on the TCP transport protocol.

The DVB-RCS2 [2] satellite transmission standard, is introduced as a flexible extension of DVB-RCS to support a large variety of applications in broadband satellite communications, allowing in particular Dedicated Access (DA) or Random Access (RA) mechanisms for the shared use of the return link. DA is a contention free access allocation, typically adopted for broadband communications and large transfers, and the only access mechanism available in previous DVB-RCS standard. RA was introduced as an access option specifically designed for short transfers, elastic traffic and M2M/IoT communications in DVB-RCS2. RA offers a constant access delay, although it introduces possible losses due to collisions which must be handled at transport or application layers. RA access is assumed in some cases as the only available mechanism due to a lower terminal complexity (so with a lower cost). Due to the use of TCP for the short transfers belonging to MQTT, using RA satellite access may introduce collisions (and, consequently, packets loss), leading to a poor resources utilization, far from optimal performance achieved in the Congestion Avoidance (CA) phase [3] of TCP.

DVB-RCS2 specification offers two different options as RA access scheme: Slotted Aloha (SA) or Contention Resolution Diversity Slotted Aloha (CRDSA). The bandwidth efficiency (and losses pattern) when using CRDSA is better with regard to that offered by SA; furthermore, the use in particular of 3 replicas of the same timeslot, allows for a larger throughput, as proved in [4] and [5]. Therefore, 3-CRDSA is here selected as the best candidate for DVB-RCS2 based system. In this context, the interaction of TCP short-lived connections associated to the MQTT protocol with the use of RA access schemes, as considered in the present scenario, requires specific investigations. The main interest in this work is the characterization of the *completion time* of TCP connections opened by a MQTT session or. To this aim, the time a publisher requires to successfully complete a data exchange with the broker, in the case 3-CRDSA is adopted, will be evaluated.

The main contributions of this work are assessing the performance of MQTT publishing sessions on DVB-RCS2 return satellite links using 3-CRDSA [6] and one of the following TCP version:

- TCP New Reno, assumed as baseline and already introduced in [7] and [8] (for long TCP transfers), and in [9] for mixed random/dedicated access;

*Authors in alphabetical order
[1]https://datatracker.ietf.org/wg/core/charter/
[2]www.mqtt.org

- TCP Wave [10], which makes use of a burst-based transmission approach, considered particularly suitable for the satellite scenario considered as already discussed in [11].

In [7] a similar scenario as the one adopted herein is addressed for the first time, focusing on the start-up phase of TCP New Reno connections and offering a comparison of simulations with theoretical models. The incremental original contribution of this work with regard to [7], is to focus on a large set of objects to transfer in the long-run, offering a better insight on protocol repose for M2M/IoT over TCP. Additionally, the use of an alternative innovative TCP, namely TCP Wave, is proposed in this context for the first time.

The rest of this work is organized as follow: section II describes the scenarios we deal with; section III gives details on the newer TCP Wave specification and section IV shows and provides comments on the simulation results. Finally, the conclusions are drawn in section V.

II. Scenario Description

In the scenario we deal with in this work, a system with a finite set of nodes (i.e. RCSTs, Return Channel Satellite Terminals) transmitting MQTT-like messages over a Random Access (RA) geostationary satellite channel is considered. We assume that each transmission superframe contains one frame and one RA block [2], spanning the whole frame, without DA access. Each frame is composed of k slots and each RCST can exploit any transmission opportunity, i.e., available timeslot, to send data over the RA channel.

The DVB-RCS2 standard offers several waveforms to deal with different channel conditions and traffic types: the redundancy added by channel coding schemes can be tuned according to channel statistics; the slot size can also be different to adapt to different traffic types, such as small payloads for M2M-like traffic, or large payloads. Physical layer parameters are therefore chosen accordingly to those defined for Waveform 14 in DVB-RCS2, leading to a gross slot size of 188 bytes; this waveform provides the largest layer-II burst size, which minimizes the fragmentation of IP datagrams that encapsulate TCP segments.

When using RA mechanisms for accessing to the satellite link, packet loss event can be triggered by collisions when multiple RCSTs decide to transmit on the same slot. Since no Automatic Repeat-reQuest (ARQ) algorithm is implemented at MAC layer, and optimal quasi-error-free link conditions can be assumed, all possible losses (handled by TCP) are due to the RA access: they are a function of the instantaneous offered load, and already studied in a satellite scenario in [8] for long TCP transfers.

The traffic arrival process for each RCST is determined by a set of MQTT-like publishers, sending messages to the broker through the RCST (gray circles) to the satellite channel, as pictured in Figure 1. Therefore, the outgoing traffic for each RCST terminal is representative of the aggregated M2M traffic, as proposed in [12], which mixes short, medium and long connections. The detailed traffic characterization is similar to the one presented in [7] with reference to Scenario #1. The main characteristics of such traffic random process can be summarized as:

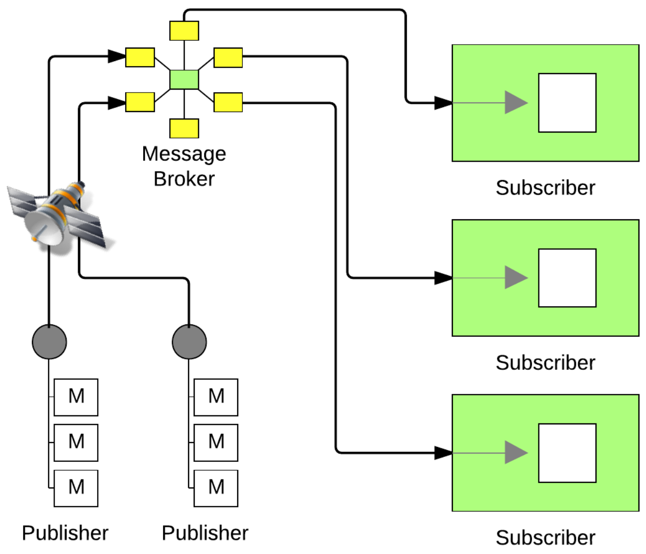


Fig. 1: Logical description of the proposed scenarios

- data payload length randomly drawn according to three different Pareto distributions of fixed shape $\alpha = 1.1$ and average values $x_{f1} = 524888$ bytes, $x_{f2} = 104858$ bytes, $x_{f3} = 10240$ bytes, representing three different types of aggregated sensor data;
- periodic data generation at application layer (MQTT) happens according to an exponential distribution with rate $\lambda = 1s$;

The sensors incoming traffic is forwarded by the RCST, through TCP connections, to reach the Broker at the other end of the satellite link. Each new TCP connections shall initiate three-way handshake (3WHS) procedure for connection establishment. If 3WHS fails in setting up the connection, it is repeated by allowing more time for the receiver to respond (most TCP implementations increase such time progressively until aborting connection establishment after 4-5 failures). In presence of possible collisions at the connection setup, the 3WHS connection establishment may fail: to simplify the analysis and focus on the effective data transfer and TCP performance, the simulations only consider connections successfully established at the first attempt. In this way, the results are not affected by 3WHS retries and failures, which are system dependent.

Once the connection is successfully set up, the data transmission starts. Each publishing application (one process per RCST) enqueues data in the TCP send buffer rapidly enough to allow for TCP to send data as fast as the CWND allows. Each TCP segment sent to lower layers, i.e., network layer first and MAC layer then, is queued in a finite buffer in RCST, here referred to as *MAC queue*, with a capacity of 128 kbytes, which is adequate to avoid losses for the application considered. Eventually, when the data transfer of a given aggregated block ends, the connection is closed.

TCP New Reno or TCP Wave is in use at transport layer and can be alternately enabled. The Maximum Segment Size (MSS) in use is 486 bytes, since this size fits exactly into an integer number f of slots, being $f = 3$, when considering

Name	Value
TCP flavor	NewReno OR Wave
TCP MSS	486 bytes
TCP header options size	12 bytes
TCP/IP headers size	40 bytes
Neat bandwidth per slot	$\approx$115 Kbit/s
MAC queue size	$\gg$ Bandwidth Delay Product
RA scheme	3-CRDSA
RA blocks per superframe	1
RA block duration	0.013 s
Slots per RA block	64
Gross slot size	188 bytes
Net slot size	182 bytes
Bandwidth	8012820 Hz
Roll off	0.2
Carrier spacing	0.3
One-way PHY delay	0.13 s

TABLE I: System setup parameters

waveform 14 [2] and suitable to contain MQTT-like messages. Table I recalls numerical values in use during the simulations for PHY, MAC, and TCP/IP layers.

In relation to the message length, due to the contention nature of the MAC access using 3-CRDSA, the data transfer may be short enough to suffer a few packet losses or zero losses, or be longer and suffer also severe loss conditions. To this aim, the error recovery mechanism implemented by each TCP version considered is of great importance.

III. TCP-Wave overview

TCP Wave is the follow-up of a protocol family started by TCP Noordwijk [13]. Basically, TCP Wave maintained the original "burst transmission" concept, while broadening the target application scenario: TCP Wave extends its algorithms and transmission/re-transmission methods with the aim to become a valid alternative to standard TCP flavors in every broadband communication environment (including terrestrial-only links, as discussed in [14]). In fact, although preserving the burst transmission concept, which has been largely proven as advantageous in satellite communications [11], TCP Wave was re-designed starting from the requirements of nowadays traffic characteristics and networks evolution, including sensors networks, lossy networks (not limited to satellite networks but also applicable to dynamic networks with variable delays, as described in and [15]) and cloud computing.

The new design principles allows, in particular, the guarantee of optimal performance under varying network and physical conditions, with adaptation to wide bandwidth and/or delay changes (as for instance assumed in [16]), acceleration of small transfers, quick achievement of high throughput and efficient handling of transmission errors, also in presence of link disconnections and narrowband/wideband links combination. All these goals are defined within a new TCP framework that guarantees the protocol interoperability in any current and future communication scenario, assuming no changes in the receiver nodes (sinks).

TCP Wave, with relation to the lossy nature of the communication environment resulting by the use of 3-CRDSA, implements a novel strategy for loss recovery, inspired by the classic TCP recovery mechanisms (referenced hereafter as "standard TCP" [17]. Similarly to standard TCP, TCP Wave performs a packet retransmission when receiving three DupAcks referred to the same packet (the packet with the next sequence number is retransmitted). Differently, upon this event, TCP Wave sender handles this call through a dedicated Fast Retransmission algorithm, to individually retransmit packet, considered lost, while the regular burst transmission is maintained.

Error recovery operations and timeouts of standard TCPs are widely covered in literature [17] [18] and well-known, so not detailed herein. On the other hand, being TCP Wave a relatively new and experimental protocol, some insights on its implementation of these mechanisms are provided in the remainder of this section.

TCP Wave fast retransmission algorithm is designed to recover single/multiple packet losses without incurring into re-transmission time-outs (RTO) events. In fact, the lost packet is immediately retransmitted, while burst transmission scheduling is kept unaltered. Most likely, the retransmitted packet successfully delivered will generate a cumulative ACK covering packets transmitted on multiple bursts. When this happens, TCP Wave sender performs only a single ACK-based statistic update accounting the cumulative ACK to complete computation concerning the ACK train partially received before the loss. As a result, every cumulative ACK will produce at most 1 new RTT sample. The Fast Retransmission algorithm does not limit in practice the maximum number of packet losses that can be recovered without triggering RTO, because the uninterrupted transmission of "fresh" packet bursts allows generation of all the DupAcks needed to report multiple consecutive losses. To opposite, standard TCP during fast retransmit is able to generate just a limited number of "new" packets according to the Congestion Window (*cwnd*) value, increasing the possibility to experience an RTO before the recovery of all the consecutive losses.

Both the new Fast Retransmission and RTO algorithms, following the principles of the TCP Wave burst-based transmission, rely on an internal timer instead of using ACK reception as clock, making the actual response to losses substantially different from the one of standard TCP, in terms of resulting recovery time and throughput during recovery.

IV. Performance evaluation

The performance evaluation shown herein is based on the NS3 [19] simulator, including its satellite extensions for DVB-S2/RCS2 satellite enabled networks, according to the modules developed by Magister Solution Ldt. and described in [20] (SNS3 project). NS3 already includes a very accurate TCP New Reno model, in its baseline configuration. Concerning TCP-Wave, it was specifically developed on purposed, based on the TCP Noordwijk implementation described in [21] and available online, with major adaptations and a complete algorithm re-write.

Please note that TCP New Reno is assumed as reference for the comparison to TCP Wave, without considering other major TCP flavours (e.g., TCP Cubic [22], TCP Vegas [23], etc.). This is due to the fact that the main characterization of the connection in the RA Satellite environment considered is the slow-start phase, the retransmission and the ACK-clocked window transmission mechanisms, which in all main TCP

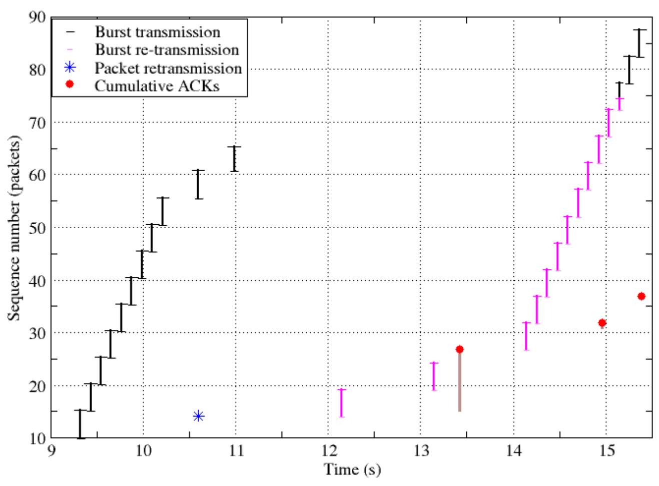


Fig. 2: TCP Wave detailed Sequence Number evolution in the target scenario for a reference connection

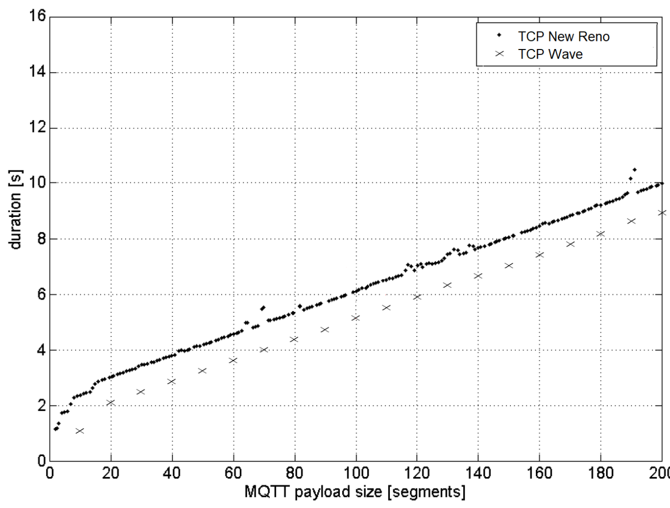


Fig. 3: TCP Wave completion time with buffer $\gg$ BDP

protocols behave similarly. Furthermore, NS3 still does not support many additional TCP flavors (at present only Tahoe and Westwood [24] are available): nevertheless, as soon as possible, an extensive comparison will be presented as future work, similarly to what presented in [25]. The main aim of the paper is to characterize the completion time for M2M-like connections delivered through a TCP based transport protocol. Before that, an initial validation of the TCP Wave suitability to the identified MQTT scenario is provided.

The collisions experienced by the active TCP connections are function of the load, as already discussed in [8]; for instance, in presence of a load of 70% when using 3-CRDSA, the resulting slot collision probability (i.e., impossibility of recovery of the transmitted data slot and, consequently, of the whole TCP packet involved) is about 10%. At the same time, the load is dependent on the reaction of the TCP protocol to the RA channel considered, thus realizing a mutual interactive effect, and the traffic model.

In Figure 2, a plot of sequence number of transmitted packets, as function of time, is presented. It is extracted in the middle of a MQTT TCP Wave transfer, and shows the bursty transmission of the protocol (black bars). It shows, as well, the presence of retransmitted packets (asterisks), due to the RA collisions, which do not interfere to a great extent to the normal transmission process. During the presented time range, a timeout event occurs (at time 11.8 s), leading to further re-transmissions (purple bars) starting from the last acknowledged packet. Nevertheless, the transmission is recovered promptly: even if many packets are transmitted twice, the reception of cumulative ACKs (dots) allows to recover transmission efficiently. These aspects can be extremely beneficial to the scenario considered. Full details on TCP Wave algorithms and error recovery are available in [26].

After this preliminary test, the *completion time* shall be evaluated, when using different TCP versions. Due to the dynamic behavior of TCP congestion control and error recovery, which is affected by the error rate and it is impacting the load of the RA channel, it is not easy to forecast in general the TCP performance with accuracy. In [27] a modelization is proposed, but limited to small payloads (1 to 9 IP packets). So, to address the analysis required, a multi-round simulation run adopting a Monte-Carlo like approach is proposed. In practice, a big number of MQTT-like messages is generated within the simulation, following the proposed traffic model, and delivered through one of the available RCST. Each RCST has at maximum one slot out of 64 available for its transmission, making use of two more slots for the replicas. Many runs are executed using different random seeds, allowing to generate independent traffic conditions and system state. Then, completion times are collected from all simulations outputs, sorted by payload size, and averaged. Simulations number and duration are increased until the confidence value of the average calculated is satisfactory.

In Figure 3 the average completion time (not taking into account the 3WHS time) for all possible number of segments between 1 and 200 is represented, when TCP New Reno or TCP Wave are used. The initial parameters used for TCP Wave are a maximum burst size of 10 packets, with an initial burst spacing of 1 s, using the same buffer values and application configuration with regard to TCP New Reno.

The curve related to TCP New Reno shows an initial ramp up of completion time, in the range of 1-10 segments, followed by an almost linear increase, with the exception of few glitches (for instance, at 64 segments). When the CWND in the Slow Start phase increases from 64 segments to 128 segments (7th RTT if no losses are experienced), a buffer overflow happens, forcing the connection to recover the lost segments by entering in TCP New Reno Fast Retransmit Fast Recovery or by experiencing a RTO; this is very common in these scenarios, and explains why the completion time suddenly grows in average when the MQTT payload size ranges between 64 and 128 segments. Finally, loss due to collisions in using the RA access of the proposed scenario have a limited impact, since the increase in object transfer is almost linear with regard to its size (then, no connection reset or multiple RTO events happen).

The curve associated to the use TCP-Wave shows that it is better handling the channel available capacity, especially in the initial stages, with a transfer duration time always lower than TCP-Reno. In fact, the combined presence of a narrow-band channel and a relatively high channel latency, makes the CWND growth for TCP Reno inefficient. On the contrary, the initial transmission of bursts for TCP Wave (at a distance of 1 s), before feedback from the link is collected, followed by an adjustment to the effective channel bitrate, are more suitable for the proposed system and offer a better performance. With messages bigger than 40 segments, the protocol completion time when using TCP Wave has the same linear increase in completion time with regard to New Reno, since in this cases the limiting factor is the channel bitrate (approximately to 115 Kbit/s). Please note that, contrary to previous releases such as TCP Noordwijk, it was not necessary to adjust any of the parameters of the protocol, which in other simulations with the same configuration exploit at best the full channel capacity for broad-band terrestrial/satellite mixed scenarios.

V. Conclusion

In this work, the use of the MQTT protocol for IoT/M2M traffic profile over DVB-RCS2 links has been investigated. The length of the MQTT payloads, delivered through New Reno or Wave TCP connections, is randomly drawn from Pareto distributions having increasing average values, to cover most typical M2M payload lengths. The focus of the performance analysis is on the averaged behavior of every M2M connection established, sorted by payload dimensions, for characterizing with accuracy the completion time and then the specific TCP protocol performance. The results demonstrated how, the use of TCP as transport protocol for M2M/IoT traffic through a lossy satellite channel is feasible and offers in average good performance. Furthermore, TCP Wave is confirmed slightly better performance compared to standard TCP, if considering the completion time, due to its bursty nature (thus accelerating the small objects involved, specifically in the startup phase).

References

[1] C. Pereira and A. Aguiar, “Towards Efficient Mobile M2M Communications: Survey and Open Challenges,” *Sensors*, vol. 14, no. 10, pp. 19 582–19 608, 2014.

[2] “Second generation, DVB-RCS2 Part2: Lower layers for satellite standard,” ETSI EN 301 545-2, 2012.

[3] C. Barakat and E. Altman, “Performance of short TCP transfers,” in *Networking 2000 Broadband Communications, High Performance Networking, and Performance of Communication Networks*. Springer, 2000, pp. 567–579.

[4] R. De Gaudenzi and O. D. Herrero, “Advances in Random Access protocols for satellite networks,” in *Satellite and Space Communications, 2009. IWSSC 2009. International Workshop on*. IEEE, 2009, pp. 331–336.

[5] N. Celandroni, F. Davoli, E. Ferro, and A. Gotta, “On elastic traffic via Contention Resolution Diversity Slotted Aloha satellite access,” *International Journal of Communication Systems*, 2014. [Online]. Available: http://dx.doi.org/10.1002/dac.2885

[6] E. Casini, R. De Gaudenzi, and O. R. Herrero, “Contention Resolution Diversity Slotted ALOHA (CRDSA): An enhanced random access scheme for satellite access packet networks,” *Wireless Communications, IEEE Transactions on*, vol. 6, no. 4, pp. 1408–1419, 2007.

[7] M. Bacco, T. De Cola, A. Giambene, and A. Gotta, “Advances on Elastic traffic via M2M Satellite User Terminals,” in *The Twelfth International Symposium on Wireless Communication Systems, ISWSC, Brussels, 2015*. IEEE, 2015.

[8] M. Bacco, A. Gotta, C. Roseti, and F. Zampognaro, “A study on TCP error recovery interaction with Random Access satellite schemes,” in *the 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC) 2014*. IEEE, 2014, pp. 405–410.

[9] F. Manlio Bacco, A. Gotta, M. Luglio, C. Roseti, and F. Zampognaro, “A new RA-DA hybrid MAC approach for DVB-RCS2,” in *Communications and Networking (BlackSeaCom), 2015 IEEE International Black Sea Conference on*. IEEE, 2015, pp. 215–219.

[10] M. Luglio, C. Roseti, A. A. Salam, and F. Zampognaro, “A burst-approach for transmission of TCP traffic over DVB-RCS2 links,” in *20th IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks, CAMAD 2015, Guildford, United Kingdom, September 7-9, 2015*. IEEE, 2015, pp. 175–179.

[11] C. Roseti, M. Luglio, and F. Zampognaro, “Analysis and performance evaluation of a burst-based TCP for satellite DVB RCS links,” *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 911–921, 2010.

[12] J. Kim, J. Lee, J. Kim, and J. Yun, “M2M service platforms: survey, issues, and enabling technologies,” *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 61–76, 2014.

[13] M. Luglio, C. Roseti, G. Savone, and F. Zampognaro, “TCP Noordwijk for high-speed trains,” in *Proceedings - 2009 1st International Conference on Advances in Satellite and Space Communications, SPACOMM 2009*, 2009.

[14] A. Abdelsalam, M. Luglio, C. Roseti, and F. Zampognaro, “Evaluation of TCP Wave performance applied to real HTTP traffic,” in *International Symposium on Networks, Computers and Communications (ISNCC), pp. 1-6.*, 2017.

[15] M. Luglio, C. Roseti, G. Savone, and F. Zampognaro, “TCP Wave resilience to link changes - A new transport layer approach towards dynamic communication environments,” in *13th International Joint Conference on e-Business and Telecommunications (DCNET 2016), Lisbon, Portugal*, 2016.

[16] A. Abdelsalam, M. Luglio, C. Roseti, and F. Zampognaro, “Cross-layer architecture for a satellite–wi-fi efficient handover,” in *IEEE Transactions on Vehicular Technology, 58(6), 2990-3001*, 2009.

[17] M. Allman, V. Paxson, and E. Blanton, “TCP Congestion Control,” RFC Editor, RFC, Sep. 2009. [Online]. Available: http://tools.ietf.org/html/rfc5681.txt

[18] M. Stevens, *TCP/IP illustrated, vol. 1*, 1st ed. Addison Wesley, 1994.

[19] “The Network Simulator NS-3,” http://www.nsnam.org, [Online].

[20] J. Puttonen, S. Rantanen, F. Laakso, J. Kurjenniemi, K. Aho, and G. Acar, “Satellite model for Network Simulator 3,” in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, Lisboa, Portugal, March 2014, pp. 86–91.

[21] N. Patriciello, M. Casoni, C. A. Grazia, and M. Klapez, “Implementation and validation of tcp options and congestion control algorithms for ns-3,” in *Workshop on ns-3 (WNS3) 2015, Barcelona, Spain*. IEEE, 2015.

[22] S. Ha, I. Rhee, and L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” *SIGOPS Operating System Review*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[23] L. Brakmo and L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[24] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, “TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links,” in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom)*. New York, NY, USA: ACM, 2001, pp. 287–297.

[25] A. Abdelsalam, C. Roseti, and F. Zampognaro, “Steady-state performance evaluation of Linux TCPs versus TCP Wave over leaky satellite links,” in *China Communications 2017/3, pp.17-30*, 2017.

[26] A. Abdelsalam, M. Luglio, C. Roseti, and F. Zampognaro, “TCP Wave: A new reliable transport approach for future Internet,” in *Computer Networks Volume 112, 15 January 2017, Pages 122-143*, 2017.

[27] M. Mellia and H. Zhang, “TCP model for short lived flows,” *Communications Letters, IEEE*, vol. 6, no. 2, pp. 85–87, Feb 2002.