DVB-RCS2/S2 Testbed: A Distributed Testbed for Next-Generation Satellite System Design and Validation

Stefan Erl and Tomaso de Cola
German Aerospace Center (DLR)
Institute of Communication and Navigation
Oberpfaffenhofen, Germany
{Stefan.Erl, Tomaso.deCola}@dlr.de

***Abstract*—DVB-RCS2 is the second generation of DVB standards for the return link in interactive satellite systems, which is combined with the well consolidated DVB-S2 standard, applicable to forward links. This paper presents a DVB-S2/RCS2 testbed, able to reproduce all functionalities specified for higher and lower layers (HL and LL) of the protocol stack, down to the physical layer characteristics, which are instead accounted by means of a channel emulator for both return and forward links. The presented testbed turns out to be a formidable tool for research investigation and overall system design thanks to the implemented features and the flexible architecture, allowing easy reconfiguration and extensions of functionalities according to specific purposes. Some merit figures shown at the end of the paper also prove the possibility to plug the testbed to external systems and carry out performance monitoring/tracking tasks through regular Simple Network Management Protocol (SNMP) agents.**

***Index Terms*—DVB, emulation, RCS2, satellite communication, testbed.**

I. Introduction

The recently standardized DVB-RCS2 [1] is the second generation of digital video broadcasting (DVB) standard for interactive systems operating on the return channel of satellite networks and the natural completion of the well-consolidated DVB-S2 [2] for forward link, operating already since a few years. To this regard, the new specification introduces several new features, the most prominent is certainly the standardisation of the overall satellite system from the higher down to the lower layers of the protocol stack. This peculiarity is certainly in favour of a more complete interoperability between satellite terminals and gateways coming from different manufacturers. Further to this, new technical functions have been added, such as the Return Link Encapsulation (RLE) [3] protocol and the random access support, so at to extend the range of services possibly supported by next-generation satellite systems and also to enhance the quality of service (QoS) management jointly performed by higher and lower layers.

In light of the release of the DVB-RCS2 standard and the necessary update of DVB-S2 for what concerns the specification of protocol stack higher layers, the research community needs new or updated tools to assess the efficacy of the new standard on the one hand, and satellite manufacturers require new DVB-S2/RCS2 validation systems in order to guide the design and implementation of future satellite systems on the other hand. To this regard, simulation environments are certainly supporting the assessment of the new standard but are not able to capture the complexity of the overall system and, more importantly, do not allow shedding light on the issues arising in real systems. Alternatively, emulation systems are regarded as more suitable candidates as they offer the possibility to implement the overall system in order to perform very close to a real satellite deployment. Further to this, emulation systems can also interwork with real devices, so that it would be possible to test an emulated DVB-S2/RCS2 system with real modems transmitting data over a physical interface. Amongst some of the emulation system developed for DVB-S2/RCS evaluation, OpenSAND (formerly known as Platine [4]) is one of the most complete, especially for what regards QoS management performed at the Internet Protocol (IP) layer and resource allocation/request tasks performed by DVB-S2 and DVB-RCS layers, respectively. It also provides a relevant simulation of main physical layer functionalities, such as adaptive coding and modulation (ACM) for both forward and return link and channel characteristics are properly emulated. The overall testbed is composed of dedicated machines for each satellite component (e.g., satellite terminals and gateway), thus closely representing a real satellite system. The SNEP (Satellite Network Emulation Platform) [5] builds on a similar distributed architecture and implements DVB-S2/RCS especially with respect to resource allocation functionalities and higher layer protocols (not really specified in DVB-S2/RCS standards though). Hence it allows a better understanding of the dynamics ruling the interaction of transport protocols and Demand Assignment Multiple Access (DAMA) schemes, also offering the possibility to use enhanced versions of Transmission Control Protocol (TCP) [6] (e.g., TCP Noordwijk [7]) in a performance enhancing proxy (PEP) architecture based on the SCPS-TP gateway design [8].

To the best of authors' knowledge, no system emulation has been yet developed for the new generation of DVB-S2/RCS2 satellite systems. In particular, the aforementioned OpenSAND and SNEP, which indeed implement functionalities which are

978-1-4799-5893-1/14/$31.00 ©2014 IEEE

also part of the lately released standards, do not support new functionalities, such as random access, satellite virtual networks (SVNs) or negotiation protocols for PEP architectures, just to cite a few. Hence, this paper presents a novel DVB-S2/RCS2 testbed implementing all functionalities specified in the DVB-S2/RCS2 standard from the higher layer down to the physical layer, which is actually reproduced by channel emulators using realistic satellite profiles. Nevertheless, the architecture of the testbed allows to integrate it with real devices such as Tx/Rx modem for real system assessment. Similarly to OpenSAND and SNEP, it is implemented in a distributed way in order to precisely separate and reproduce the functions performed by satellite terminals, gateway, and network control center (NCC) (not implemented in OpenSAND). Further to this, the architecture of the proposed emulation system has been designed in a flexible way so as to allow easy extension of the current functionalities and configuration of some of them to meet the needs of scientific investigations. The testbed described in this paper is not intended as a system simulator capable of simulating large terminal populations, but rather to provide a reference implementation of a real RCS2 satellite terminal. It can be obtained for commercial, academical and research purposes from the European Space Agency (ESA) under [9].

The rest of this paper is structured as follows. More details about the recently released DVB-RCS2 are given in Section II. The description of the testbed architecture and its validation through example applications are provided in Section III and IV, respectively. Finally, V draws conclusions about the presented testbed and gives an outlook about possible extensions of the current testbed functionalities.

II. RCS2 Standard Overview

The DVB-RCS2 standard is the natural answer to the evolution of nowadays satellite systems in order to meet, on the one hand, the new traffic and high performance demand and, on the other hand, the need for reducing the market fragmentation because of the lack of terminals' interoperability [10]. The ever-increasing need for higher capacity systems able to accommodate current Internet traffic demands has actually pushed towards the evolution of the first generation of DVB standards. From this standpoint, it can be observed that the second generation standard for the forward link is in place already since 2005, whereas the native DVB standard for return link was initially subject to some update to meet the requirements of mobile services and scenarios (DVB-RCS+M) [11]. Afterwards, the need for a new standard has been regarded as necessary in order to work out a set of new specifications that reflected the advance of satellite technology and could take out obsolete technologies, such as the Asynchronous Transfer Mode (ATM) frame format.

The new standard has been scoped in 2009 and then finalised in 2012, with recommendations firstly published as DVB blue books and then finally as European Telecommunications Standard Institute (ETSI) technical specifications. The main novelty of DVB-RCS2 consists in addressing both lower and higher layers unlike the first version of the standard. In this way, a complete specification of the protocol stack from the physical layer down to the transport layer has been provided in terms of functionalities pertaining to the user, control and management planes. The overall specification has been actually split into three independent specifications: system design [1], Lower Layer Specification (LLS) [12], and Higher Layer Specification (HLS) [13] to which corresponding implementation guidelines [14] [15] are accompanied .

Amongst the main novelties appeared in the LLS, it is worth mentioning ACM [16], RLE and Random Access. The first was one of the key elements that upheld the success of DVB-S2 standard and has been also introduced over return link to improve the reliability of data transmission and to optimise the use of available satellite capacity. RLE is the natural counterpart of Generic Stream Encapsulation (GSE) in DVB-S2 and has been designed to support flexible encapsulation of higher layer protocol data units into physical layer frames. In particular several requirements have been met in order for the lower layer to efficiently transport IP datagrams respecting QoS demands and enable the use of random access schemes. In addition, the protocol also offers extensions to make the encapsulation adaptable to the needs of future satellite systems. Support for random access [17] in next-generation satellite systems is one of the main novelty of LLS as it allowed to complement traditional DAMA schemes. The use of random access is considered particularly appealing for applications generating bursty traffic or asynchronous messages, where the employment of traditional capacity allocation schemes cannot scale well. Alternatively, the possibility of combining the two flavours (random access and DAMA) is also contemplated in the standard so as to allow the satellite system provider to configure terminals in the most appropriate way to meet traffic demands and performance expectations.

As far as HLS [18] is concerned, no new protocols are recommended; in fact, the TCP/UDP/IP protocol stack is mandated as reference. The main novelty consists in the QoS management schemes required to properly map the IP-based classes of service onto the request classes defined at the lower layer to perform resource allocation functions. Further to this the inclusion of Robust Header Compression (ROHC) to enable compression of TCP, UDP, and IP datagrams is certainly an add-in that proved to very important to optimise the system performance. A quite new element specified for the implementation in Higher Layer (HL) is a dedicated protocol to negotiate the configuration of the PEP agent, also involving the setup of the aforementioned ROHC functionalities. Finally, a special note has to be dedicated to the definition and implementation of a management plane, in terms of the Fault, Configuration, Accounting, Performance, Security (FCAPS) model. In particular, performance and configuration functions assume some importance here as they have been implemented by means of SNMP protocol, thus allowing the satellite operators and providers to efficiently adjust the system configuration without manually intervening on the terminals and gateways.

978-1-4799-5893-1/14/$31.00 ©2014 IEEE

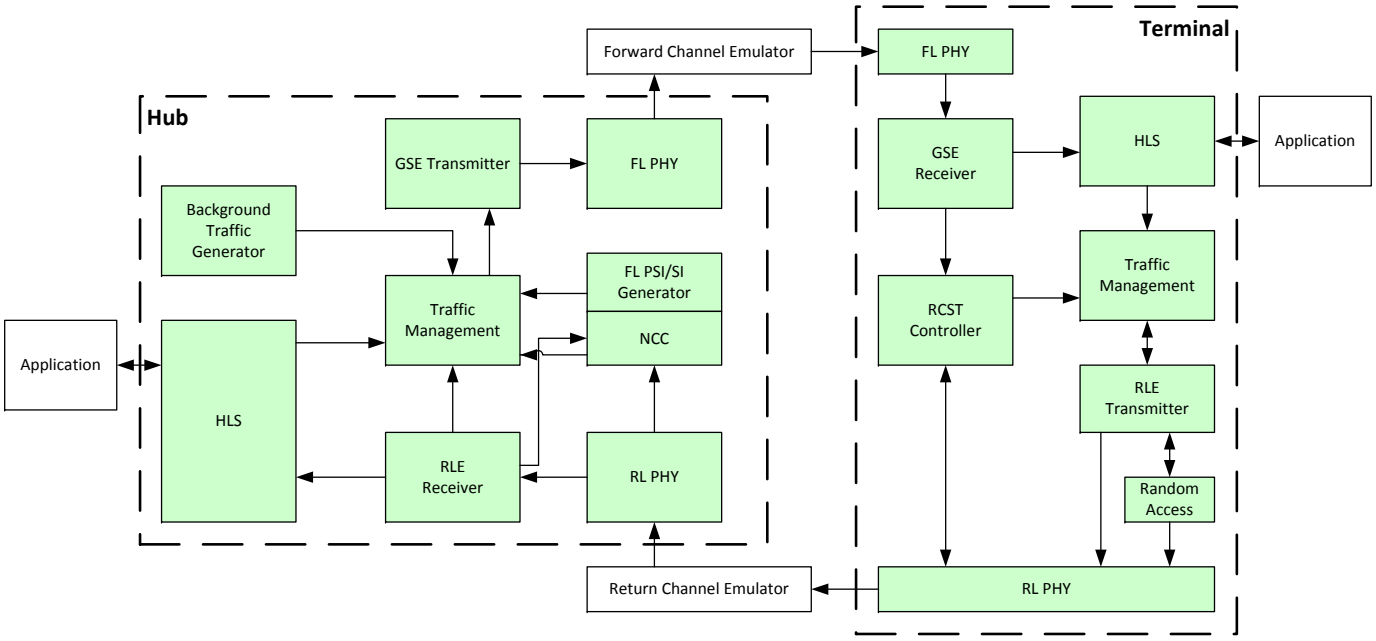


Fig. 1. Overall testbed architecture.

A last interesting building block of the RCS2 standard is the definition and specification of SVNs, which is a concept specific of Lower layer (LL) but also addressed at HL. SVNs essentially define the concept of virtual subnetworks within a reference satellite system, in order to allow a better use of available satellite resources and to make the network configuration and address planning an easier task, especially with respect to the assignment of IP addresses. The definition of SVNs also allows, for instance, to segregate traffic classes on specific portions of the satellite capacity in a very efficient way, also owing to the mapping functionalities implemented by HL and LL, and propagated by the RLE protocol.

III. RCS2 Testbed Architecture

The RCS2 testbed has a very modular software architecture. Each major building block is implemented as a separate module with well-defined interfaces. The modules are written in C++ and are designed for the *FreeBSD* operating system (OS). Figure 1 shows the overall architecture of the testbed with a hub module and one terminal module. In general, several terminals can be used with one hub module. The modules can be run as individual applications, also on different machines, or they can be packed together to bigger executables. This allows easy testing of each module and lowers the effort to change or extend a certain functionality.

The modules communicate with the help of two protocols, namely Next Generation testbed Protocol (NGP) and Higher Layer Protocol (HLP), both based on User Datagram Protocol (UDP) messages. The NGP protocol is a collection of messages with a type and length field. It is mainly used for information exchange between the modules implementing the lower layer functionality, e.g. below GSE and RLE, as well as for communication with the *NCC* and the *RCST Controller module*. The HLP protocol is designed for communication between the *HLS module* and the LLS counterparts. It contains the user traffic and the corresponding addressing and traffic classification information. To this regard, it is worth noting that the HLP protocol implements functionalities very similar to those provided by the SI-SAP [19] interface in the ETSI Broadband Satellite Multimedia (BSM) architecture. In particular, the QoS mapping functions offered by this interface in terms of resource reservation primitives (i.e. SI-C-QUEUE_OPEN) are actually the same as those available from the HLP protocol although the implementation in terms of protocol fields is slightly different.

In the following, a brief description of the LLS modules is given. Afterwards, the *HLS module* is presented in more detail, since the RCS2 standard introduces some new concepts and functionality in this layer.

A. Lower Layer Modules

On the hub side, the *NCC* is responsible for generating the network clock reference (NCR) and the signaling tables. Supported tables are, among several others, Superframe Composition Table (SCT), Frame Composition Table version 2 (FCT2) and Broadcast Configuration Table (BCT) for defining the superframe structure in the return link. The *NCC* also takes care of the login procedure for new terminals.

The *Hub Traffic Management module* has two main tasks. First, it receives classified user traffic from the *HLS module* or signaling tables from the *NCC* and assigns Modulation and Coding (ModCod) information to it, based on the current channel measurements reported by the terminals. The second task is the resource allocation in the return link. The allocation is done based on the capacity requests sent from the terminals and on an additionally configured continuous rate assignment

(CRA). The supported request types are rate-based dynamic capacity (RBDC) requests and volume-based dynamic capacity (VBDC) requests with different request classes. Unused capacity in the superframe can be distributed among the terminals using free capacity assignment (FCA) or marked as random access slots. The resource allocation is compiled into a terminal burst time plan 2 (TBTP2) signaling table. User traffic and signaling traffic is then forwarded to the *GSE Transmitter*.

A *Background Traffic Generator* can be plugged to the *Traffic Management module*. It can load the forward link (FL) by producing IP packets with random payload at certain rates. Regarding the return link (RL) load, no real traffic packets, but fake capacity requests are generated that can simulate several hundred terminals and influence the resource allocation algorithms.

In the *GSE Transmitter*, the traffic is fragmented and encapsulated into GSE packets and put into queues according to the assigned priority. When triggered by the *FL PHY module*, the traffic scheduler in the *GSE Transmitter* generates baseband frames (BBFRAMEs). The *FL PHY module* maintains a certain symbol rate taking into account the ModCod and size of the frames. A L.2 mode-adaption header [20] is added before the frame is sent to the *FL Channel Emulator*.

On the terminal side, the *FL PHY* module takes L.3 packets as input and passes it to the *GSE Receiver*. A media access control (MAC) filter is applied, and the accepted user and signaling traffic is reassembled, decapsulated and forwarded to either the *HLS module* or to the *RCST Controller*, depending on the protocol type of the packet. The *RCST Controller* implements the terminal part of the state-machine for the logon and synchronization procedures. It also extracts the allocated transmission resources from the signaling tables and forwards it to the *Terminal Traffic Management* and sends control bursts containing status information on a regular rate back to the hub.

The terminal's *Traffic Management module* generates capacity requests based on RLE queue statistics and sends them either in pure control bursts or, if available, in combined control and traffic bursts using RLE. It also forwards the information about the allocated resources for the current superframe to the *RLE Transmitter*. The *RLE Transmitter* consists, similar to the *GSE Transmitter*, of a priority-based scheduler with queues and several encapsulators, one for each multiple access transmission type. It outputs RLE packets based on the current channel resources available. RLE packets containing DAMA traffic are directly forwarded to the *RL PHY module*, while packets with random access (RA) traffic are fed into the *RA module*, together with the timeslots designated as random access slots. Potential time-overlapping slots with already utilized DAMA slots are removed before.

Depending on its configuration, the *RA module* selects one of the RA slots for each payload in the Slotted ALOHA (SALOHA) case, or it generates and distributes the appropriate number of replicas in the Contention Resolution Diversity Slotted ALOHA (CRDSA) case. The packet are also labeled accordingly. The terminal *RL PHY module* multiplexes the control and user traffic from the above modules and sends it to the *RL Channel Emulator*.

Back on the hub side, the *RL PHY module* demultiplexes the incoming traffic and forwards pure control bursts to the*NCC* and pure and combined traffic bursts to the *RLE Receiver*. Since the testbed has only a few physical terminal instances, RA traffic is dropped at this stage based on pre-configured load curves for different numbers of replica. In the *RLE Receiver module*, the source address of the packets is reconstructed from the RA tags or from the position of DAMA slots occupied. The reassembled and decapsulated traffic is sent to the next modules, signaling to the *NCC* and user traffic to the *HLS module*.

The channel emulators on both the FL and the RL apply delay to the transmitted packets, set the current signal-to-noise ratio (SNR) values in the packet headers according to some time-series and eventually erase packets with a pre-configured probability. The insertion of bit-errors is supported, but since the testbed does not implement forward error correction, all packets with flipped bits are dropped in the receivers due to cyclic redundancy check (CRC).

The use of the mode adaption interface in the FL allows the usage of real DVB-S2 equipment in the testbed. The utilization of real RCS2 hardware in the RL is expected to be accomplished without too much effort, since only the *RL PHY modules* might have to be adapted.

B. Higher Layer Modules

The *HLS module*, as the name may suggest, implements the new part of the standard dealing with the higher layers of the protocol stack. While the LLS functionalities are contained in the modules itself, the *HLS module* relies on several OS functionalities, such as routing tables, multicast forwarding caches and network stack virtualization. Also some third-party software is used for enabling PEP and ROHC functionality. The detailed architecture of the *HLS module* is shown in Figure 2. The *HLS modules* in the hub and the terminal are almost identical, except for example the software download, which is split into a server and a client side. IP traffic is supported in version 4 and 6, both unicast and multicast, where the latter relies on a static forwarding configuration.

The *HLS module* consists of the interfaces to the *LLS modules*, a switch for traffic coming from the *LLS modules*, and at least two SVNs, that have individual interfaces to the user applications. The exact number of SVNs is configurable and generally only limited by the available hardware resources. The overview in Figure 2 shows only the two mandatory SVNs, the structure of further SVNs would be identical to SVN 1. The interface to the user is a standard Ethernet interface, and allows the attachment of real hardware, like for example other computers, switches or wireless local area network (WLAN) access points. Also virtual machines running on the testbed machines can be used through virtual Ethernet interfaces. The *HLP Switch* distributes incoming traffic to the different SVNs according to the SVN-MAC address provided in the HLP header of the packets.

978-1-4799-5893-1/14/$31.00 ©2014 IEEE

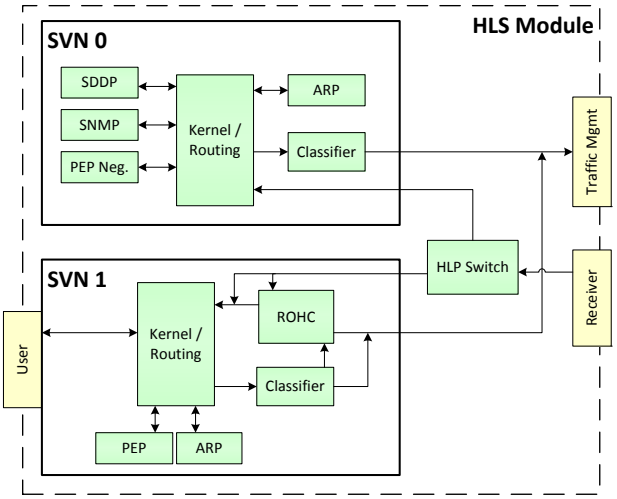


Fig. 2. The HLS module.

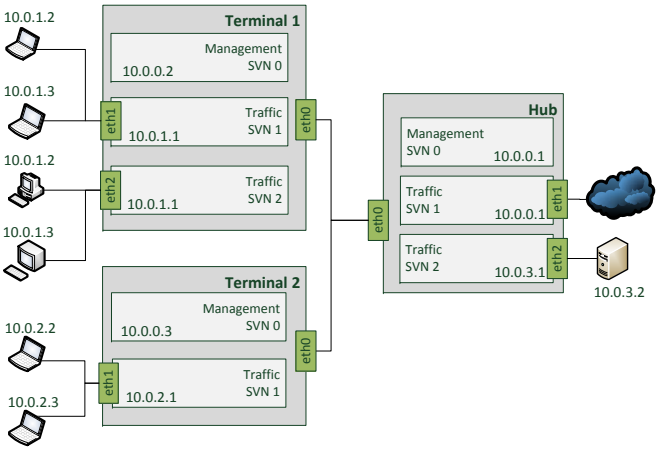


Fig. 3. An example SVN setup.

A SVN is realized with a *FreeBSD Jail*. That is a kind of a very light-weight virtual machine that provides an own memory and process space and a virtualized network stack, but shares the kernel and file-system with the host machine. The virtual network stack allows a SVN to have own independent IP addresses and routing tables. This completely separates the traffic of different SVNs. It is also possible to re-use the same IP addresses in different SVNs without interfering with each other. An example of such an addressing scheme is given in Section IV. Each SVN maintains its own Address Resolution Protocol (ARP) cache populated with the lower layer addresses of the hub resp. the logged-on terminals and responds to ARP requests from the user side. This enables easy routing of unicast traffic. Multicast traffic is processed separately, since the lower-layer multicast address can be computed algorithmically or based on the multicast mapping table (MMT), depending on the current configuration. The configurable classifier assigns the QoS priority and the preferred transmission method, DAMA or RA, to traffic going to the satellite side. It can filter the traffic based on typical fields like IP address, protocol type, port number or differentiated services code point (DSCP) value.

The management SVN, SVN 0, incorporates the management functionalities like software download, PEP negotiation and SNMP monitoring. The software download is done through Software Download Delivery Protocol (SDDP) [13]. The hub broadcasts *WRQ* and *DATA* packets on a known multicast group. The terminals listen to this group, and if the broadcasted software version is newer than the installed one, they start downloading the new version. This works without the need of being logged in to the RCS2 network, if the GSE MAC filter is set correctly. PEP negotiation is performed after a successful logon of a terminal. The message exchange is carried out on a known UDP port in the management SVN. The current implementation supports the negotiation of the usage of ROHC and PEP on a per-terminal basis, but not for each SVN individually. Several system parameters and counters, especially for the *LLS modules*, can be read via SNMP from the hub over the satellite network. The SNMP agent is composed of the agent included in FreeBSD, *bsnmpd*, and a plugin that adds the testbed-specific management information base (MIB) entries. The SNMP agent accepts connections also from the host network and provides a useful debug possibility in this way.

The traffic SVNs, SVN 1 and higher, handle the user traffic. In contrast to the management SVN, all traffic SVNs provide an interface to the user side on the terminal. If negotiated, ROHC is used for user traffic on the satellite link in both directions. Since the same IP addresses can be reused in every SVN, each SVN has its own instance of ROHC compressor and decompressor, to avoid confusion in the compression contexts. Packets are compressed after classification and are tagged with the designated protocol type for ROHC. For performing the actual (de-)compression, an open-source library called *rohc-lib* [21] is used, with IP/UDP profile in unidirectional mode.

The *PEP module* is based on the *SCSP-TP gateway* [8] with the *TCP Noordwijk* extension [7], a satellite-optimized TCP flavor, and works with TCP splitting. TCP connections originating at the user side are intercepted and terminated in the *PEP module*. An ACK message is sent back to the user, and a new TCP connection is established over the satellite link using *TCP Noordwijk*.

IV. Validation and Example Applications

In this section, we demonstrate several functionalities of the testbed with real applications. All numerical values except the TCP throughput are monitored with an SNMP tool, but the SNMP traffic itself is not part of the results. For traffic generation, *iperf* is used. Its output is also used to monitor the TCP throughput, since the SNMP data would also include retransmissions. The channel emulators apply a delay of 250 ms in each link direction. In order to get fair comparisons, we do not require the insertion of packet errors, since ACM is used and the timescale of the presented graphs is in the order of only a few minutes. The SNR of the channels is set to a fixed value of 16.5 dB, except in the ACM example.

A. Satellite Virtual Networks

In Figure 3, an example setup of the testbed consisting of three machines, one hub and two terminals, is shown. Each machine has several Ethernet interfaces, named *ethX*. There exist three SVNs in the system, SVN 0 for the management traffic, SVN 1 and SVN 2 for user traffic. SVN 1 provides access to the Internet and SVN 2 allows access only to a file server. The hub and terminal 1 belong to all three SVNs, while terminal 2 does not belong to SVN 2. The network between the *eth0* interfaces represents the satellite channel with channel emulators.

The users connected to terminal 1 on interface *eth1* are assigned the same IP addresses like the users connected on interface *eth2*. But this still works, since the SVNs are independent from each other. So the user with IP address 10.0.1.2, which is reachable from the Internet, is a different one than the user with the same IP address seen by the file server. The users connected to terminal 2 can only communicate with the Internet or the other users of SVN 1, but not with the file server, since terminal 2 does not belong to SVN 2.

B. Performance Enhancement Proxy

In this example, we demonstrate the effect of the testbed's PEP. In a first case, a sufficiently large file is downloaded to a user behind a terminal from a file server attached to the hub. In a second case, two files are uploaded from the user to the file server, where the upload of the second file starts after 100 s after the first file, while the upload of the first file is still in progress. The file transfer is done using File Transfer Protocol (FTP), that relies on TCP. The FL channel capacity is set to 10 Mbit/s, and the terminal has a constantly assigned rate of 16 kbit/s and an additional DAMA capacity up to 1.5 Mbit/s using rate-based capacity requests. The transfers are carried out twice, once with and once without PEP. In the case without PEP, an end-to-end *TCP NewReno* [22] connection, which is the *FreeBSD* standard, is used with an increased buffer-size of 2 MB. In the case with PEP, the aforementioned *SCPS-TP gateway* with *TCP Noordwijk* and TCP-splitting is activated.

Figure 4 shows the variation of the download speed over time for the first 180 s. The PEP reaches the channel limit within a few seconds and keeps the transmission rate quite stable. Without PEP, the transmission rate grows slowly within the first seconds, because of the slow-start mechanism of TCP and the high bandwidth-delay-product. The bursty arrival of the ACK messages at the server are interpreted as congestion by *TCP NewReno*, and so the transmission rate is lowered from time to time. Thus, the throughput barely reaches 4 Mbit/s. The upload case is shown in Figure 5. Here, both mechanisms reach the channel limit, but the PEP keeps the upload rate more stable and reaches the limit faster. The small drop in throughput at the beginning of the upload in the PEP case is a congestion due to DAMA capacity requests. When the second upload starts after 100 s, both TCP versions share the available bandwidth equally between the two transmissions.

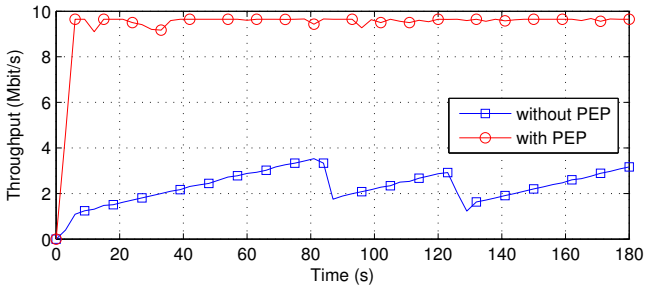


Fig. 4. FTP file download to a terminal.

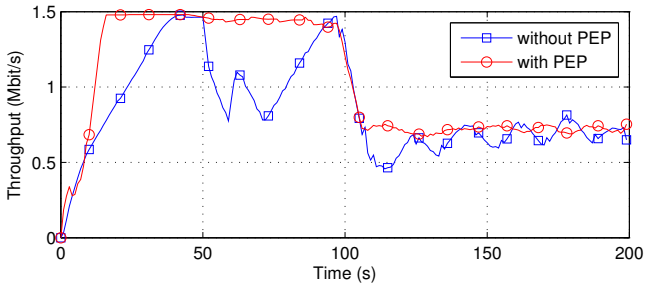


Fig. 5. FTP file upload from a terminal.

C. ACM in the Return Link

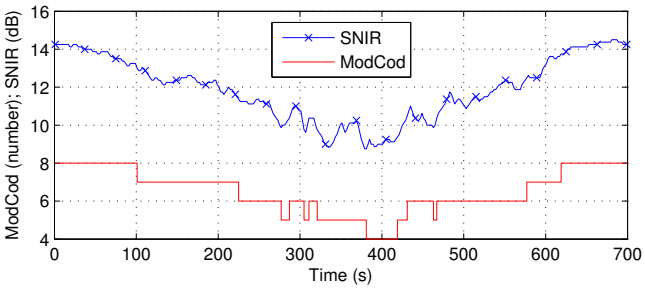
This experiment demonstrates the use of ACM in the RL during a rain-fade event. The hub measures the power of the received bursts from a terminal by reading the SNR value of the packet headers, which are set by the channel emulator. It chooses the highest possible ModCod that the terminal should use, so that the hub station can still receive the traffic without errors. The terminal has a fixed capacity of 300 kbit/s assigned using CRA. Figure 6(a) shows the channel measurements and the corresponding ModCod selection.

The resulting transmission rate at the terminal and the number of allocated time-slots are shown in Figure 6(b), where the time axis matches the one of Figure 6(a). The desired rate of 300 kbit/s is always guaranteed by adjusting the assigned transmission resources, here in terms of number of assigned time-slots. The small deviations in the assigned rate are caused by some over-allocation because of the capacity granularity of the time-slots using different ModCods.

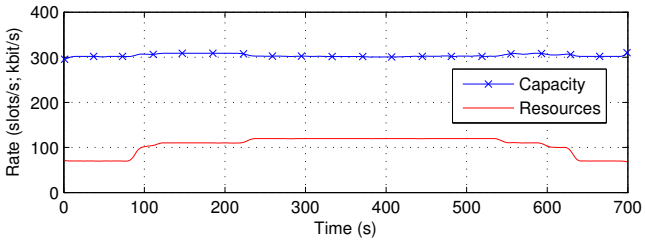
D. Capacity Requests

In this setup, a data stream of 100 kbit/s is sent from the terminal to the hub. The transmission resources are assigned only based on the capacity requests generated by the terminal. The difference between RBDC and VBDC requests is shown in Figure 7. The rate-based requests result in a smooth and constant rate, while the volume-based requests lead to a more on-off-like transmission behaviour. This is mainly because of the interaction of the traffic arrival rate and the sampling rate of internal queue statistics.

978-1-4799-5893-1/14/$31.00 ©2014 IEEE



(a) Channel quality and ModCod.



(b) Assigned rate and allocated time-slots.

Fig. 6. ACM in the return link.

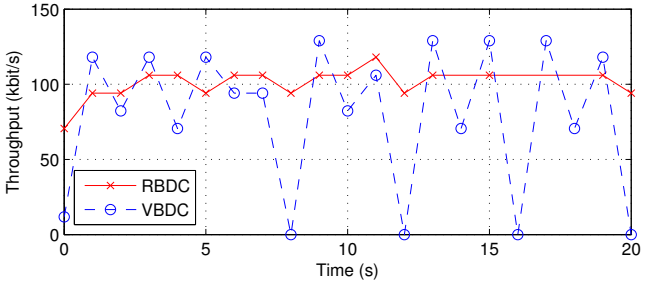
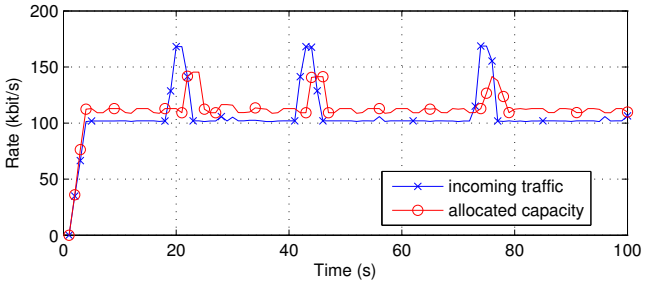
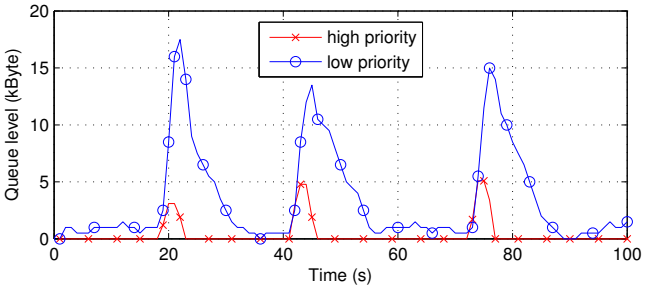


Fig. 7. Rate-based vs. volume-based file transfer.



(a) Incoming traffic and assigned capacity.



(b) Fill level of the queues.

Fig. 8. Transmission with two types of QoS.

E. Quality-of-Service

Different applications can produce traffic with different QoS requirements. On the one hand, a streaming application can produce a constant video stream, but it does not matter if there is some jitter in the frames, since buffering-techniques can be used. On the other hand, a Voice over IP (VoIP) application produces data only when someone speaks, and is very sensitive to jitter. Such a traffic pattern is also shown in Figure 8(a). There is a constant, low-priority traffic of 100 kbit/s sent from the terminal to the hub. On top of this, there are short bursts from time to time with high priority. The terminal has a CRA of 120 kbit/s, so slightly above the low-priority traffic. When the bursts arrive, additional capacity is requested via VBDC. The resulting capacity allocation is also shown in Figure 8(a). There one can also see the delay between the traffic burst and the corresponding assignment, that is caused by the internal rate of request generation, the round-trip delay time (RTT), and the allocation offset for the TBTP2.

Figure 8(b) shows the fill level of the queues for the two traffic classes. The time basis of the two plots in Figure 8 is identical. When the high-priority bursts arrive, the serving of the low-priority queue stops and the available capacity is used for the high-priority traffic. As soon as that queue is empty, the low-priority queue is served again. First, the queue level goes down quite fast, because the extra capacity caused by the burst gets assigned. Then, the queue is emptied at a lower rate that is the difference between the rate of the incoming traffic and the CRA.

F. DAMA and Random Access

Here, we investigate the difference between DAMA and RA transmissions. A terminal sends a small message consisting of 3 RLE packets to the gateway every 3.005 s, in the first case using only capacity assigned with VBDC requests, in the second case only RA slots. The time between generating the message and receiving a response from the hub is measured. CRDSA with 2 replica is used in the RA case, and channel has a simulated traffic load of 0.5, which results in a packet loss probability of 0.05. The low traffic rate ensures that capacity allocations caused by two consecutive messages do not interfere with each other. The results of this experiment are shown in Table I.

The RTTs of both methods are higher than the pure RTT of the satellite channel, which is around 500 ms, since also the superframe duration of 100 ms and several other internal delays have to be taken into account. In the DAMA case, an additional round-trip is necessary for the capacity assignment process, and there is also an offset in the superframe counter in the TBTP2 that tells when the assigned capacity becomes valid. This example is a kind of worst-case for the DAMA method, since there is no CRA and no other traffic that would cause more frequent capacity requests. On the other side,

978-1-4799-5893-1/14/$31.00 ©2014 IEEE

TABLE I
COMPARISON OF DAMA AND CRDSA.

	DAMA	CRDSA
RTT min	1603 ms	518 ms
RTT avg	2098 ms	600 ms
RTT max	2356 ms	675 ms
Loss	0%	17.1%

there are no losses in the DAMA case, which can reach a significant level in the RA case, especially for messages that are fragmented over several bursts.

V. CONCLUSION

The paper has introduced a DVB-S2/RCS2 testbed able to reproduce most functionalities of a real satellite system from the physical up to the transport layer. Though limited to only three satellite terminals, the architecture is flexible to extend the population and implement new functions (e.g., network coding). In addition, the validation campaign has demonstrated the value of the proposed testbed not only from a scientific point of view but also for industrial exploitation with respect to satellite system design. In particular, the implementation of performance monitoring and configuration functions provided by SNMP interfaces proved to be an added value of this testbed with respect to existing ones, as it allows to verify the behaviour of the system in real-time. Finally, the modular design of the system also allows to transparently integrate real satellite modems in order to more precisely take into account the limitations imposed by waveforms and synchronisation issues.

Future plans for the presented testbed include the implementation of network coding as extension of the GSE protocol, by defining a specific network coding protocol implemented between IP and the encapsulation layer.

ACKNOWLEDGMENT

The DVB-S2/RCS2 testbed has been developed within the framework of the "DVB-RCS Next Generation Support and Verification Testbed" project [9] coordinated by VeriSat and funded by the European Space Agency (ESA) under the contract 4000102764/11/NL/NR.

In particular, the authors would like to thank Dr. Nader Alagha and Stephane Combes (ESA) for the fruitful feedbacks during the testbed design and development phases.

REFERENCES

[1] *Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (RCS2) Part 1: Overview and System Level specification*, ETSI European Telecommunications Standards Institutes Std. TS 101 545-1, May 2012.

[2] *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, ETSI European Telecommunications Standards Institutes Std. EN 302 307, Mar. 2013.

[3] *Satellite Earth Stations and Systems SES; Return Link Encapsulation (RLE) protocol*, ETSI European Telecommunications Standards Institutes Std. TS 103 179, Aug. 2013.

[4] C. Baudoin and F. Arnal, "Overview of Platine emulation testbed and its utilization to support DVB-RCS/S2 evolutions," in *5th Advanced Satellite Multimedia Systems Conference (ASMS) and the 11th Signal Processing for Space Communications Workshop (SPSC)*, Sept 2010.

[5] M. Luglio, C. Roseti, and F. Zampgnaro, "A Satellite Network Emulation Platform for Implementation and Testing of TCP/IP Applications," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Korakis, M. Zink, and M. Ott, Eds. Springer Berlin Heidelberg, 2012, vol. 44, pp. 1–2. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35576-9_1

[6] C. Caini, R. Firrincieli, M. Marchese, T. d. Cola, M. Luglio, C. Roseti, N. Celandroni, and F. Potort, "Transport layer protocols and architectures for satellite networks," *International Journal of Satellite Communications and Networking*, vol. 25, no. 1, pp. 1–26, 2007. [Online]. Available: http://dx.doi.org/10.1002/sat.855

[7] C. Roseti, M. Luglio, and F. Zampognaro, "Analysis and Performance Evaluation of a Burst-Based TCP for Satellite DVB RCS Links," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 911–921, June 2010.

[8] R. Wang and S. Horan, "Protocol Testing of SCPS-TP over NASA's ACTS Asymmetric Links," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 2, pp. 790–798, April 2009.

[9] DVB-RCS Next Generation Support and Verification Testbed. [Online]. Available: http://telecom.esa.int/telecom/www/object/index.cfm?fobjectid=31731

[10] H. Skinnemoen, "Introduction special issue on DVB-RCS2," *International Journal of Satellite Communications and Networking*, vol. 31, no. 5, pp. 199–200, 2013. [Online]. Available: http://dx.doi.org/10.1002/sat.1058

[11] *Digital Video Broadcasting (DVB); Digital Video Broadcasting (DVB); Interaction channel for Satellite Distribution Systems*, ETSI European Telecommunications Standards Institutes Std. EN 301 790, May 2013.

[12] *Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 2: Lower Layers for Satellite standard*, ETSI European Telecommunications Standards Institutes Std. EN 301 545-2, Jan. 2012.

[13] *Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 3: Higher Layers Satellite Specification*, ETSI European Telecommunications Standards Institutes Std. TS 101 545-3, May 2012.

[14] "Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Guidelines for Implementation and Use of TS 101 545-3," ETSI European Telecommunications Standards Institutes, Tech. Rep. TR 101 545-5, May 2012.

[15] "Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Guidelines for Implementation and Use of LLS: EN 301 545-2," ETSI European Telecommunications Standards Institutes, Tech. Rep. TR 101 545-4, Dec. 2012.

[16] H. Brandt, C. Abdel Nour, N. Alagha, and H. P. Lexow, "Digital video broadcasting-return channel via satellite linear modulation with turbo coding," *International Journal of Satellite Communications and Networking*, vol. 32, no. 1, pp. 49–59, 2014. [Online]. Available: http://dx.doi.org/10.1002/sat.1027

[17] A. Munari, G. Acar, C. Kissling, M. Berioli, and H. P. Lexow, "Multiple access in DVB-RCS2 user uplinks," *International Journal of Satellite Communications and Networking*, 2013. [Online]. Available: http://dx.doi.org/10.1002/sat.1039

[18] G. Fairhurst and A. Yun, "Design of the DVB-RCS2 higher layer satellite architecture," *International Journal of Satellite Communications and Networking*, vol. 31, no. 5, pp. 233–248, 2013. [Online]. Available: http://dx.doi.org/10.1002/sat.1037

[19] *Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Common Air interface specification; Satellite Independent Service Access Point SI-SAP*, ETSI European Telecommunications Standards Institutes Std. TS 102 357, May 2005.

[20] "Mode Adaption Input and Output Interfaces for DVB-S2 equipment," SatLabs, Tech. Rep. sl_561, Feb. 2008.

[21] ROHC library. [Online]. Available: http://www.rohc-lib.org

[22] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," Internet Requests for Comments, RFC Editor, RFC 6582, April 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6582.txt