# SAPIENT-Simulator Modelling and Architecture

Antonio Virdis, Giovanni Stea
University of Pisa, Largo Lucio Lazzarino 1, Pisa, Italy, {a.virdis, g.stea}@iet.unipi.it
Stefano La Barbera, Roberto Winkler
Thales Alenia Space Italia S.p.A. - Via Saccomuro, 24 - 00131 Roma - Italy
{stefano.labarbera, roberto.winkler}@thalesaleniaspace.com

## Abstract

Future aeronautical communications will be based on the TCP/IP protocol stack, and will occur through a number of different data-link channels (e.g., satellite, terrestrial), with multipath capabilities – the so-called *multilink*. Seamless *vertical* handover between different data-links is a requirement and it will improve the safety and reliability of AEROCOM systems, possibly enabling remote-piloted aircrafts (RPAs) for civil operations. This paper describes the modelling, design and implementation of an AEROCOM system simulator based on OMNeT++, developed in the framework of the SAPIENT EU project. The simulator includes models of the aircrafts, including their mobility, terrestrial and satellite data links and core network. Moreover, it includes a solution to simulate the effect of multilink capabilities, which enables one to test multilink decision policies.

## 1 INTRODUCTION

The Future Communication Infrastructure (FCI) for AEROCOM systems will need to manage data-link (DL) bandwidth in an efficient way, in order to make the system scalable, safe and secure. This will necessarily entail being able to use *different* DLs, e.g., terrestrial and satellite, at different times (or, possibly, simultaneously). This ability, called *multilink*, will enable aircrafts (A/Cs) to dynamically select – depending on the scenario – the DL with the highest signal strength, or the less-congested DL, so that the communication quality of service remains optimal. This should be done *automatically and dynamically,* i.e. not requiring human intervention or static, pre-loaded flight plans. Doing so would also enable Remote-piloted Aircrafts (RPAs) to operate on civil airspace with the necessary safety.
The SAPIENT project [1], funded by SESAR Joint Undertaking under European Union's Horizon 2020 research and innovation programme, proposes to enable future A/Cs (including RPAs) to report to a central entity time- and geo-referenced measures of the quality of the DL they are using. This, in turn, enables the central entity (the so-called "solution owner", SO) to build and update a 4D map of the DLs over the European sky, and to feed back to the A/C themselves *global* information regarding the state of all the available DLs in their neighbourhood. This allows several benefits, e.g. giving the A/Cs all the relevant information to choose the best DL, thus improving continuity of service, reducing delays, and increasing the number of A/Cs that can operate in nominal conditions in a certain area.
Assessing the viability of the SAPIENT system concept (coherently with Technology Readiness Level 2) requires studying two complementary aspects. First, assessing the overhead of the exchange of information on the AEROCOM system: it might be the case that the relevant benefits (such as those mentioned above) require A/Cs to report too much information to the SO, thus making the idea unfeasible. Second, to assess the performance of communications (both SAPIENT-generated and standard AOC) in an end-to-end fashion, traversing all the relevant network elements and protocol layers, at the relevant scale in terms of number of A/Cs, terrestrial/satellite ground stations, etc.. In this paper, we argue that the appropriate tool for such an assessment is a *system-level simulator*, i.e. one that models interacting modules (network elements, protocol layers, etc.), focusing on reproducing their observable behaviour rather than its internal structure. We describe the SAPIENT system simulator, showing how we model the underlying AEROCOM network, including the multi-link function. We then discuss results obtained with the SAPIENT simulator, showing how SAPIENT enables proactive vertical and horizontal handover at a negligible communication overhead. The SAPIENT simulator is based on the OMNET++ simulation framework [2], which boasts a very large worldwide community of users in both academia and corporate worlds, and has been designed to be *modular* and *extensible*. This has several benefits. First of all, each module can be further specialized to the



SUPPORTED BY SESAR JOINT UNDERTAKING
Founding Members
EUROPEAN UNION
EUROCONTROL

required level of detail: for instance, the channel model can be specialized to incorporate the effect of different atmospheric impairments on communications. Second, and no less important, other components of an AEROCOM network (e.g., different DLs) can be added seamlessly, without modifying the rest of the simulator. This makes the SAPIENT simulator a useful tool, beyond the project scope, for rapid prototyping (still at low TRLs), large-scale testing and comparison of FCI software solutions, e.g. multilink decision policies, or algorithms run within the AEROCOM network.
Over the last decade, system-level simulators for aeronautical systems have been developed either for commercial purposes or within EU-funded projects. The available system-level simulators mainly focus on simulating satellite or aircraft mobility, with few to none details on the underlying communication protocols. The Elsa Air traffic Simulator [1] is an agent-based simulator to test high-level concepts of air traffic management in a European scenario. It covers generation of flight plans and interactions between air traffic controller and pilots, and does not model communication protocols and links. The Airborne network Simulator [8] simulates aircraft trajectories over the US Sky. It provides models for directional antennas and wireless point-to-point channels. No details on the communication protocol are provided. $OS^3$ [6] and Satellite Network Simulation Platform [7] are OMNeT++-based simulators of satellite constellations. The latter also provides a basic protocol stack and channel models for microwave and laser links. Work [9] evaluates the performance of VDL Mode 2 via a simulator toolchain consisting of separate tools, whose output represents the input for the next tool in the chain. However, a comprehensive study of the end-to-end performance requires models of other network components, such as the core network, multilink facilities, and beyond-layer-3 network protocols (e.g., TCP, UDP, and application-layer protocols), which are not covered in [9].
The rest of the paper is organized as follows: Section 2 describes the OMNET++ framework, while Section 3 describes the modelling of the AEROCOM network and the SAPIENT system. Section 4 reports performance evaluation results, and Section 5 concludes the paper.

## 2 OVERVIEW OF THE OMNET++ SIMULATION FRAMEWORK

As mentioned in the Introduction, the SAPIENT simulator is based on the OMNET++ framework. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, implemented following the *discrete-event dynamic simulation* (DEDS) paradigm, i.e. one where simulated time is advanced in discrete steps, as a consequence of *events* occurring (e.g., the arrival of a packet at a node). It can be used to model several things – i.e., not only networks. The basic OMNeT++ building blocks are *modules*, which can be either *simple* or *compound*. Modules communicate through *messages*, which are usually sent and received through a *connection* linking the modules' *gates*, which act as interfaces. Message reception at a module is the *event* that updates the simulated time. Modules can be grouped to form a *network,* a compound module with no gates to the outside world, which sits at the top of the hierarchy. Connections have a bitrate, delay and loss rate, and cannot bypass module hierarchy: with reference to Figure 1, simple module *n* cannot connect to 2, but must instead pass through the compound module gate.

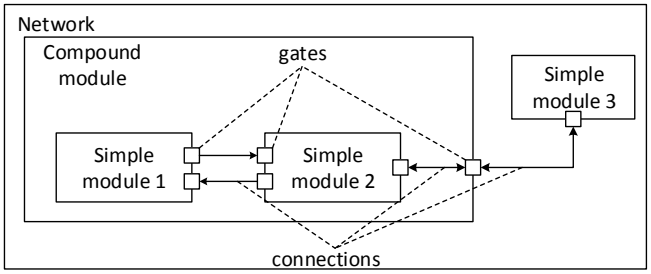

Figure 1 - Main elements of the OMNeT++ architecture

Simple modules implement model *behaviour*. They are provided with event handlers, which are called by the simulation kernel when a module receive a message, i.e. when an event is triggered on such module. Besides handlers, simple modules have *initialization* and *finalization* functions, that manage the initial configuration and write results to a file at the end of the simulation. The OMNeT++ kernel includes an event queue, whose events correspond to messages (including those a module sends to itself). All it takes to create the model behaviour within a module is to implement the above functions to

manage message reception and to send new ones. Model's *implementation, description and parameter values* are kept separate. The implementation (or *behaviour*) is coded in C++ and is generally written to .h and .cc files. The description (i.e. the definition of gates, connections and parameters) is expressed in files written in the Network Description (NED) language. Parameter values instead are written in initialization (INI) files. NED is a declarative language, which exploits inheritance and interfaces, and it is fully convertible (back and forth) into XML. Using NED, the user can write *parametric* topologies, e.g. a ring or tree network composed of a variable number of nodes. INI files are used to define simulation scenarios, and contain the initial values for the model parameters, as well as information on the simulation itself, e.g. simulation duration, etc..
A design choice of the SAPIENT project has been to focus on the TCP/IP protocol stack for communications, following the ICAO Doc 9869 [4]. Models of components and protocols of the TCP/IP stack come pre-packaged in the INET library [3], a freely available model library for OMNeT++. For instance, the Internet stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), wired and wireless link layer protocols (Ethernet, PPP, IEEE 802.11, etc.), are included, making the developing considerably faster.
The INET framework includes a model of a general-purpose computer, called Application Node (see Figure 2). The latter is a compound module containing all the layer of the TCP/IP stack, themselves implemented as (simple or compound) modules. An application node can have multiple applications running simultaneously on top the communication stack. Applications generate and/or receive user data. An application node connects to the rest of the network via interfaces placed at the bottom of the stack. The INET framework also provides an IP-router model. IP routers are compound modules having an IP layer, which performs routing, and two or more interfaces with the rest of the network. Each interface can be configured to implement QoS policies (e.g., prioritization of packets).

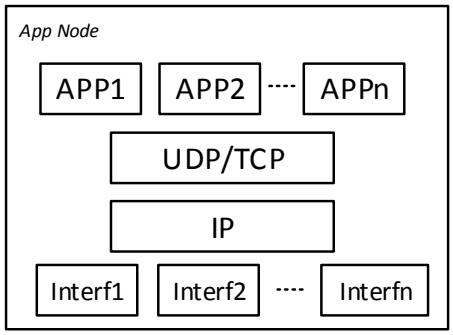

Figure 2 - Structure of the application node

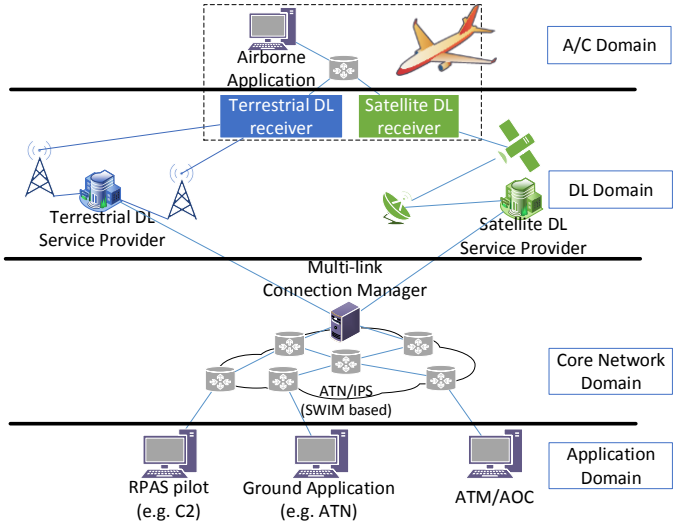

Figure 3: Domains of the SAPIENT communication network.

## 3 OVERVIEW OF THE SAPIENT SIMULATOR

The SAPIENT simulator includes more than 19k lines of code, not including the models *imported* from the INET library. It models a communication network which includes several components: the aircrafts themselves, the DL infrastructure, and a core network.
The main actors involved in the communication network are represented in Figure 3. The whole communication architecture can be divided into several domains. Each domain is further composed of a number of nodes, which are modelled and implemented with the required level of detail. The SAPIENT simulator is developed with modularity and flexibility in mind, first of all providing clear interfaces between domains. The *Application domain* includes the applications that are used for the purpose of the ATM operations, e.g. ATS and AOC. *Foreground* applications (i.e., those whose state at a given time is meaningful and must be known as part of the simulation process) are modeled with state machines. *Background* traffic can be added to model additional load (e.g., induce queuing delay, bandwidth contention at the endpoint, losses). The latter is instead modelled with *traffic generators*, i.e. – at a first level of approximation – distributions of packet length and inter-arrival times.

The *Core Network* (CN) *domain* encompasses the core elements of the ATN/IPS, e.g. IP routers, which are used to provide connectivity for the ground network. The model of the CN domain includes information on the network topology, link bandwidth and delay, routing etc.
The *DL Domain* includes the DLs that can be used to transport ATM/AOC data. Two data links are considered within the SAPIENT project (hence, in the simulator as well):

- A *terrestrial* DL, modelling the LDACS data link [10], consisting of several antennas located on a floorplan (at given 3D coordinates), between which seamless layer-2 handover may occur.
- A *satellite* DL (SATCOM), modelling the ANTARES satellite DL, where an orbiting satellite relays communication between the A/C and a ground station.

The *A/C Domain* models the A/Cs as communication end-points, e.g. running ATM/AOC applications, mobility models, etc. A/Cs are mobile nodes, whose trajectory is simulated as a waypoint model, i.e. a sequence of {3Dcoordinates, speed}, meaning that the aircraft moves at the specified constant speed towards the next waypoint. The A/Cs communicate with the system using either or both the two available Datalink networks, i.e. SATCOM or Terrestrial. Ground Nodes are attached to the CN.

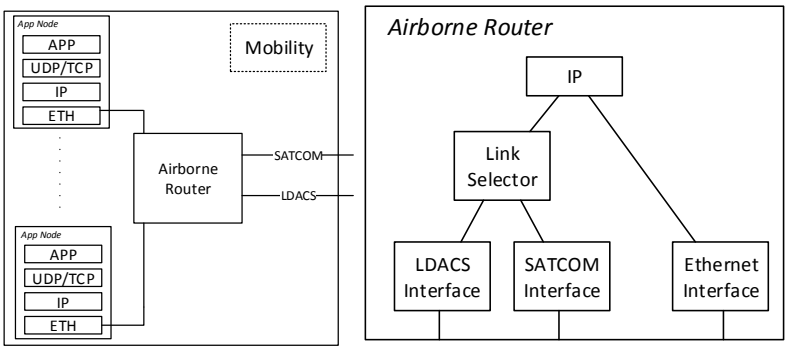


Figure 4: Internal structure of the A/C (left) and of the airborne router (right).

## 3.1 The A/C model

The A/C is modelled as a compound module, containing one or more application nodes, one airborne router and one mobility model, as we show in Figure 4. The application nodes are used to generate traffic, and can reach the rest of the communication network using the Airborne Router (ABR). The latter is connected with the application nodes via an Ethernet model, which provides the internal connectivity within the A/C. The number of application nodes and the data-application model they are running (e.g. VoIP, C2, etc.), can be configured for each A/C. Finally, the mobility model manages the movement of the A/C in a 3D space over time, according to a certain mobility model.

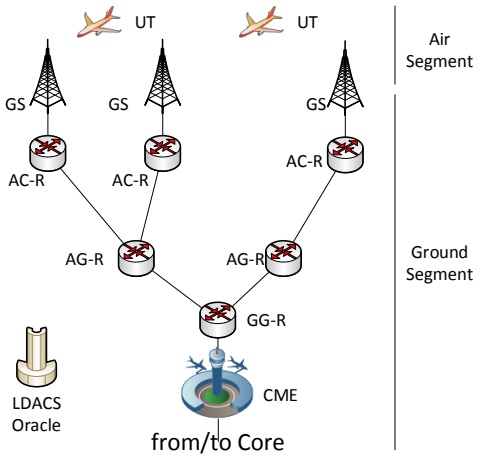


Figure 5: Terrestrial network model.

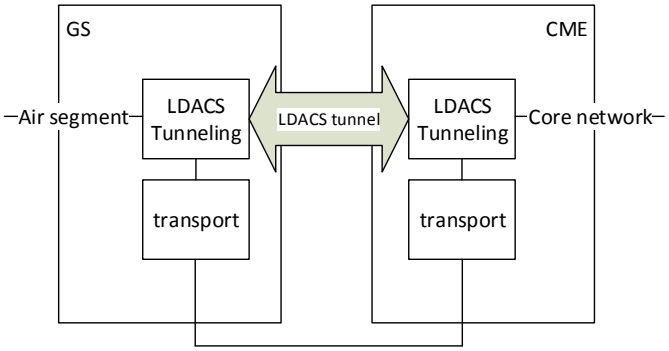


Figure 6: Representation of the tunnel betwern GS and CME

The airborne router is itself modelled as compound module. It provides connectivity among application nodes within the aircraft and towards the rest of the network. It has an IP layer that implements routing operations, i.e. receives IP packets from one incoming interface and forwards it to the proper outgoing

one, depending on the configured route. It has an Ethernet interface towards the A/C internal network, and two additional interfaces with the rest of the AEROCOM network, one for each available DL, i.e. a LDACS and a SATCOM interface. The two DL interfaces are accessible via a link selector that receives packets from the IP layer and forwards them to the currently active DL. Note that the Link selector does not decide which is the active DL, it does only applies a decision that has been made elsewhere. In other words it performs only forwarding operations.
Each simulated A/C within the SAPIENT simulator contains a module that implements the mobility mode. Each instance of mobility model can be configured by means of a flight mode and a set of flights to be performed. The mobility model within each A/C will maintain the 3D position of the A/C itself at any given time. The former will also provide means to other modules within the system to read such position and to use that during computations. E.g. a module modelling the behaviour of the transmission channel experienced by an A/C might use its position to compute the distance from a ground station, thus computing the path loss and/or the packet error probability, etc.

## 3.2 Terrestrial DL

The two DL models implemented in the SAPIENT simulator share a number of common features. Each DL network is composed of two main segments, namely a Ground and Air segment. The former provides connectivity between the entry point of the DL network on the ground side, namely the Central Management Entity (CME), and the Ground Stations (GSs), using a network of IP routers. We first describe the Terrestrial DL, and then explain the SATCOM by highlighting the major differences.
The terrestrial DL models LDACS DL network. With reference to Figure 6, the latter is connected to the CN, on one side, and to A/Cs, on the other. Each UT is associated to a single serving GS at a time, which provides it connectivity over the air segment. Whenever a packet coming from the CN reaches the CME, the latter identifies the destination User Terminal (UT, any A/C which has LDACS communication capabilities, i.e. an LDACS interface in its ABR) using the IP address of the given packet, and accordingly its serving GS. It then forwards the packet towards said GS via IP tunnelling operations: the IP packet is encapsulated within a new IP packet using the GS's IP address as destination address. This way the packet traverses the ground segment using only *standard IP routing*, up to the serving GS. The latter then decapsulates the original packet and forwards it to the destination UT via the air segment. The endpoints of the tunnels are implemented using two LDACS tunnelling modules, i.e. two applications lying on top of a transport protocol, as shown in Figure 7.
The GS is the interface between the Ground and Air segments of the LDACS network. As shown in Figure 8, left, it has two interfaces, one towards each segment. The first one terminates the tunnelled connection with the CME and is connected to the rest of the ground segment through an IP router. The LDACS tunnelling module decapsulates packets coming from the tunnel. The second is instead an LDACS interface, which lets the GS communicate through the air segment. The GS also has an IP layer, which forwards IP packets between interfaces. Multiple GS can be positioned on the floorplan, to cover different areas and serve multiple UTs. Their number and position is configurable.

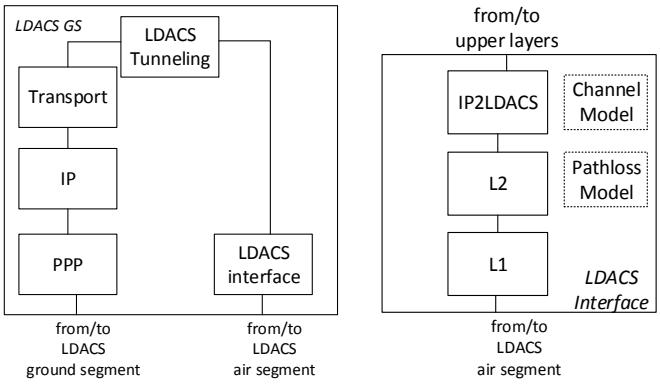


Figure 7: Internal structure of the LDACS GS model (left) and the LDACS interface model (right).

The DL interface provides connectivity through the air segment. The containing node can be either a GS or the ABR of an A/C. Each interface is implemented as a compound module realizing a multilayered model of the LDACS protocol stack with three main layers, as shown in Figure 7, right. It

also contains two modules, namely the *Channel Model* and *Pathloss Model*, that model the LDACS channel as perceived by the containing node. They are used to model the propagation of LDACS transmissions over the air channel, possibly factoring in environmental aspects such as rain, etc. Both modules are implemented so as to be easily customized and extended. The default implementation uses a Free Space Pathloss model as Path Loss Model. More advanced models may include elements such as Nakagami Fading, Rayleigh Fading, etc.

The IP2LDACS module is the interface between lower layer functions of the LDACS interface and the upper layers of the containing node, namely IP. Its main functions are the registration to the network of the LDACS interface during the initialization of the node itself and packet forwarding during communications. The L2 module implements layer-2 functionalities, such as resource management etc. The L1 module in turn, module implements all the layer-1 functionalities, such as frame decoding, signal strength measurement, etc. Both L2 and L1 modules are implemented so to be easily extended. Any implementation of L2 and L1 will share the same interface between layers, hence they can be extended without affecting the rest of the system.

Ground Routers provide IP connectivity in the LDACS ground segments. Three types of routers are envisaged, namely Access Router (AC-R), Air/Ground Router (AG-R), Ground/Ground Router (GG-R), and they all perform standard IP routing operations. In our simulations, we assume that IP routing is stable, and pre-compute all the IP routes at the beginning of each simulation Note that, since horizontal handovers are implemented using IP tunnelling, no modifications to IP routing tables is required when a UT changes its serving GS.

The CME is placed at the edge of the LDACS network and connects the latter with the CN. Its structure is similar to that of the GS (see again Figure 7), the only difference being that the rightmost interface connects to the CN via the Point-to-point Protocol (PPP).

An *LDACS oracle* is used to abstract all the functions and communications that are not relevant to the scope of the SAPIENT simulator, notably control plane protocols. The LDACS Oracle maintains information regarding every node within the LDACS system, and can be queried by the other modules to obtain that information. For instance, an *association table* within the oracle keeps track of the association between an A/C and its serving GS, and is updated after each horizontal handover. The CME then queries the oracle to get the exit endpoint of the tunnel for ground-to-air packets.

As far as resource allocation is concerned, the L2 module includes priority queues. Packets coming from upper layers are buffered depending on their priority. On the A/C side there is a buffer for each priority class, e.g. high, medium and low priority, as we show in Figure 8. The priority class is stamped on each packet by the IP2LDACS module, as it is used at L2 to identify the relevant buffer. At the GS, instead, a separated buffer for each priority class of each destination A/C is maintained.

The L2 module manages resource scheduling for both forward and return directions. Each direction is handled independently: each A/C may receive a periodic slot of configurable size for reception and one for transmission respectively. The scheduling algorithm within the L2 layer decides the size of such slots, i.e. the amount of data that can fit each transmission. The sum of the slot sizes for each direction cannot exceed the system capacity, which has a configurable value.

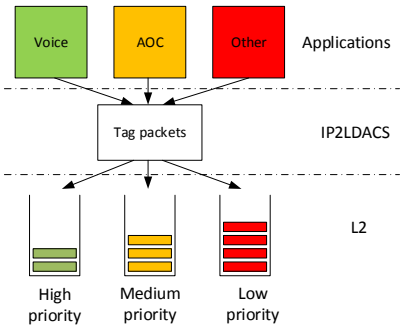


Figure 8: Priority-based queueing

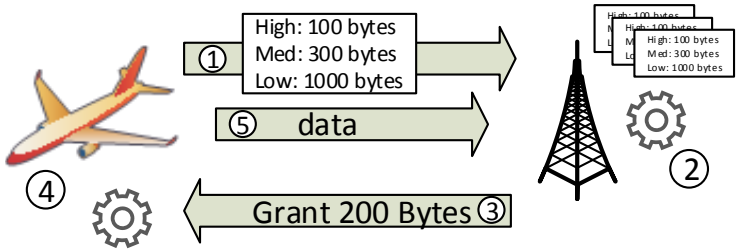


Figure 9: LDACS scheduling in the return direction.

On the forward direction, the algorithm *knows* the actual size of the transmission buffer for each A/C and priority, and uses it to make scheduling decisions. On the return direction, instead, a five-step process is required, as shown in Figure 9: the A/C sends *scheduling request*, reporting to the serving GS the size of each one of its buffers (1). The scheduling algorithm then selects the slot size for each A/C (2) and notifies it to them (3). Finally, each A/C runs its own application-level scheduling algorithm (4), to decide which buffer to transmit data from. The simulator is designed so that multiple algorithms

can be easily plugged in and tested. The current implementation provides a priority-based algorithm. The L2 module also manages horizontal handover. While connected to a serving GS, an UT can also *sense,* i.e. measure the signal coming from other (non-serving) GSs. In a legacy environment, when the SINR of a non-serving GS exceeds the one of the serving one, a handover is triggered. The SAPIENT system can override this decision according to its policies (which also use global information, e.g., the cell load). During handover (whose duration is configurable), new packets arriving at the interface are buffered at the IP2LDACS module. At the end of the handover, the interface clears all the transmission buffers and resumes.

## 3.3 SATCOM DL

The SATCOM DL provides *satellite* coverage for communications between A/Cs and ground nodes. The structure of the SATCOM network mirrors the Terrestrial one's, composed of a Ground and Air segment. The former provides connectivity between the entry point of the SATCOM network on the ground side, namely the Network Control Center (NCC), and the Ground Earth Stations (GESs), through a network of IP routers. The air segment instead includes the connections between GESs and Satellites, and between the latter and A/Cs, called User Terminals.
Satellites relay SATCOM communication between GES and UTs. They are modelled as layer-1 repeaters, comprising a satellite-specific implementation of SATCOM interface and a Mobility module to model satellite mobility. Obviously, the channel model is different from the one of the terrestrial link.
Resource allocation in SATCOM is managed at layer 2. Each connected A/C is given a periodic slot of fixed (and configurable) size for either transmitting direction. Each GES has a maximum number of available slots, thus it can serve only a given number of A/Cs simultaneously. Similarly to LDACS, each A/C runs a local packet-scheduling algorithm, to decide which buffer to transmit data from.
The SATCOM interface of each A/C also manages horizontal handover among satellites. It provides two operating modes: a first one tailored to *future* SATCOM DLs, which allows GS sensing of multiple satellites simultaneously, and is implemented similarly to LDACS. A second one, instead, models the behavior of legacy/current SATCOM DLs, which can only monitor one frequency at a time.

## 3.4 Core Network

The CN sits between the ground segments of the DLs and the ground nodes, as shown in Figure 11. It provide QoS to traffic belonging to applications having diverse priorities. It also implements the routing support to multilink, allowing the ground nodes to reach an A/C using either DLs. This means the CN provides means to establish and use an IP path from ground nodes towards the network of a specific DL. To do so, besides standard IP routers with QoS support, the CN includes one or more *Multilink Routers* (MRs). The number and connection of IP routers and MRs can be configured.

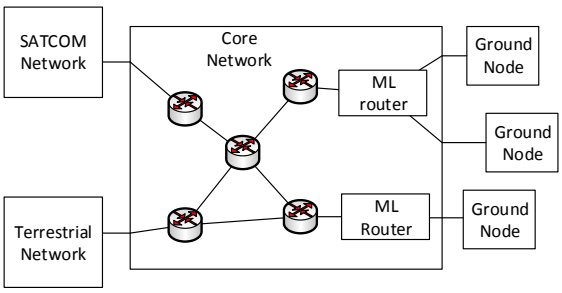


Figure 11: Core-network model.

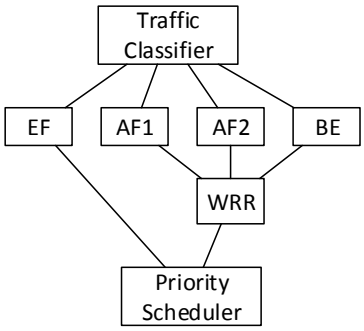


Figure 12: QoS manager within IP routers.

QoS management is implemented in a compound module called QoS manager, which embodies the Differentiated Service (DiffServ) architecture, as specified by ICAO Doc 9869 [4] (see Figure 12). DiffServ relies on traffic classification and priority scheduling. Each packet is associated to a traffic class based on information contained in its IP header, such as destination address, source address, destination port, source port or others. The available traffic classes, listed from higher to lower priority, are Expedited Forwarding (EF), Assured Forwarding (AF) and Best Effort (BE). Multiple priorities for AF traffic can also be defined, e.g. AF1, AF2, etc.. Packets are classified when they enter an output

interface by a Traffic Classifier, which reads the IP header and queues the packet into a per-class queue. AF and BE queues are managed according to a Weighted Round Robin (WRR) policy, and served at a lower priority than EF. The mapping from IP headers to traffic classes is configurable.

## 3.5 Multilink Management

Multilink refers to an A/C communicate seamlessly using two DLs. This results in two possible paths, one for each available DL, which fork at two points within the network. Such points are placed respectively at the A/C and in the CN. A *Link Selector* within the A/C's ABR receives packets from the A/C's internal network and forwards them to the appropriate DL interface. Once the packet reaches the ground, it is forwarded according to the (standard) routing of the ground segments of the used DL, and then according to the IP routing of the CN. Given the above two policies, there is only one routing path between the Link Selector and the ground node.
In the ground-to-air direction, instead, the forking point is within the CN, and the path between this point and the DL consists of multiple IP hops. A fork point in the CN will thus *tunnel* packets towards the ground segment of the appropriate DL, i.e. establish an IP tunnel with either SATCOM's NCC, or the LDACS's CME, as shown in Figure 13. The above operations are implemented using a compound module called Multilink Router, whose structure is shown in Figure 14. The *ML application* within the Multilink Router instructs the router to use one tunnel or the other. A multilink decision is thus implemented by sending to the Link Selector (on the A/C) and the ML application (on the Multilink Router) coherent information regarding the path to be traversed. The above tunnel-based implementation has a number of benefits. First, it allows IP routing to work unmodified, without being affected by multilink decisions: in fact, the only routing tables that have to be changed are those at the Multilink Router(s), while *all* the standard IP routers need not know about the very existence of multilink. Therefore, it is *scalable*. Second, it is oblivious to *who* actually makes the Multilink decision, and to the *decision policy*. Within SAPIENT, multilink decisions are made by the A/C, using local and global information, but any network component could do that, using whatever policy is appropriate, as long as it informs the two above entities. Third, it allows *concurrent* multipath, and even different paths on the forward and return links (although these options have not been considered within SAPIENT).

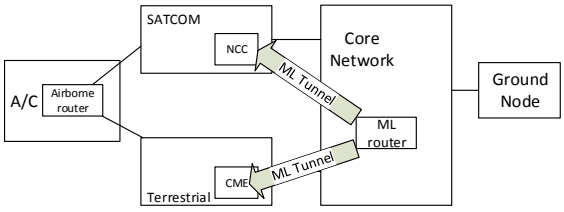


Figure 13: Tunnelling in a multilink communication.

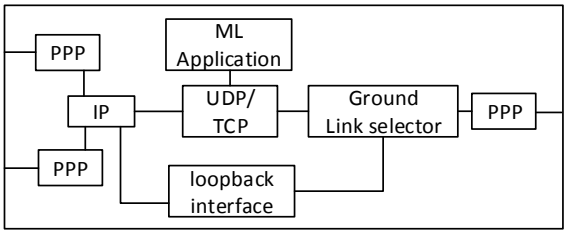


Figure 14: Internal structure of the Multilink Router.

## 3.6 Modelling of the SAPIENT system

SAPIENT is a distributed system, composed of nodes whose purpose is - on one side - to report, collect and distribute Key Performance Indicators (KPIs) regarding the DL state, and - on the other - to make/support KPI-based decisions.
Each node of the SAPIENT system is an Application Node, capable of running a SAPIENT Application, performing measurements on the communication interfaces, computing KPIs, communicating with other SAPIENT nodes, and enforcing decisions on the communication elements. We will refer to such nodes as *SAPIENT Application Nodes* (SAN). SANs are located within A/Cs, within each DL network, and within the CN. The A/C SAN performs periodic measurements, computes and reports KPIs towards the ground, and receives KPI summaries from the ground. They also run algorithms to provide support to horizontal and vertical handovers, instructing ABRs to perform a handover. The model of a SAN and its interfaces within an A/C is shown in Figure 15. The *DL Monitor* module obtains measurements from the DL interfaces within the ABR. The former periodically queries the DL interfaces for measurements such as SINR, buffer occupancy etc., then stores them internally. The SAPIENT application will then request such measurements when needed, and convert them into KPIs. The SAPIENT application will also issue commands for the ABR, and report/receive KPIs.

Modelling measurement and KPI computation within separate entities allows us to decouple the two operations, thus possibly running them at different paces. This also allows us to apply statistical operations on measured values, e.g. taking means, maxima or minima, computing distributions etc.
With DL domains, the DL monitor and the SAN are located at two distinct entities. Each ground station - either a GSs or GESs, depending on the considered DL - is equipped with a DL monitor, which performs periodic measurements on the portion of the communication system it controls, namely its coverage. Such measurements can be the number of connected A/Cs, the overall throughput in the forward and return directions, etc. The SAN is located within either a CME (in case of LDACS) or NCC (in case of SATCOM) depending on the considered DL. It periodically manages KPI reporting and reception. SAPIENT applications at the DL domain can run algorithms and enforce decisions on the considered DL. However, this use case has not been investigated in the frame of the project. Finally, SANs within the CN collect KPIs from both A/Cs and DLs, store and process them, and disseminate KPI summaries. They do not include DL monitors, as they do not perform direct measurements.

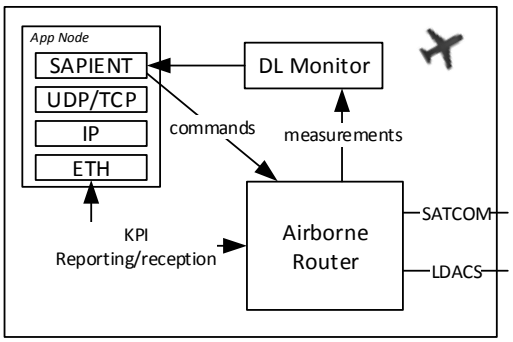


Figure 15: SAPIENT Application Node and DL monitoring at the A/C

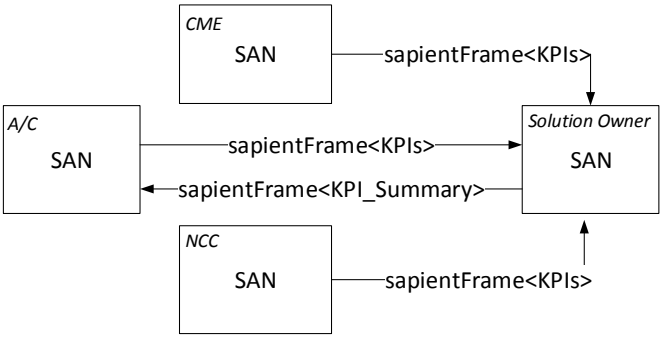


Figure 16: Message exchange among SANs.

Each SAN receives KPIs coming from other SAPIENT nodes, processes measurements from DL monitor and received KPIs, transmits KPIs or KPI summaries, and runs decision algorithms. Each of these operations is performed periodically, and has a configurable period. Figure 16 shows a high-level representation of the message exchange among SANs. Information is shared via SAPIENT frames, which are basically vectors of <type, value, 4D> tuples. SAPIENT frames can contain either KPIs or KPI summaries. The A/C and DL SANs periodically transmit SAPIENT frames containing KPIs to the SO. The SO in turn transmits a SAPIENT frame containing KPI summaries towards each A/C. Decision algorithms can be run at any SAN, and can leverage locally measured KPIs as well as KPI summaries. We describe here two examples of decision algorithms, one for horizontal and one for vertical handover.
As for SAPIENT-assisted Horizontal Handover, the A/C SAN can provide support in various scenarios:

- *Legacy DLs* (e.g., a SATCOM system) that are able to measure KPIs for one frequency only. In this case, we assume that the A/C SAN possesses an *association map*, i.e. a 4D map associating points in space with the best satellite, and exploits it to perform horizontal handover. Such association map can be created by the SAN at the SO using the received KPIs, and be transmitted to the A/C using KPI summaries, shared before take-off. During the flight, the A/C SAN will locate the position of the A/C on the *association map* to identify the best serving satellite over time. This information will be used to trigger a horizontal handover when needed.
- *Future DLs* that can measure and collect KPIs for multiple frequencies. For instance, an LDACS system where A/Cs can monitor the KPIs of multiple LDACS GSs simultaneously. Such information will be used by the A/C SAN to trigger a horizontal handover when needed. Note that, depending on the actual implementation of the considered DL, this operation might be done directly using internal operations of the DL itself. However, the availability of KPIs enables more complex horizontal handover strategies, as we show in the next section.
- DLs that can report (per-cell) load information. Large-scale aeronautical communications system will observe a varying communication load over time. To prevent congestion, DL SANs periodically report the load of their ground stations to the SAN within the CN. The latter, in turn, generates and report to each A/C a KPI summary, containing the most updated load information. Finally, the A/C SAN collects KPI summaries, and uses them in conjunction with the locally measured ones to select the best available GS or satellite.

As for SAPIENT-assisted Vertical Handover, we consider here the case where each A/C can use either SATCOM of LDACS. Two applications are envisioned:

- Each DL works in nominal conditions. In this case the A/C SAN will compare the locally measured KPIs of both DLs, and trigger vertical handovers to stay on the best DL at any time.
- One or more DLs are suffering from localized performance drops due to external factors. We assume the A/C SAN possesses a *performance-drop map*, with information on possible performance drops. Such map can be built by the SAN in the CN domain, using KPIs coming from both A/Cs and DLs, and transmitted to the A/Cs via KPI summaries. The A/C SAN will use the locally measured KPIs for the available DLs, and the (global) *performance-drop map,* and trigger a vertical handover to avoid connection losses due to performance drops.

# 4 CASE STUDIES

In this section, we report the results of two case studies, related to SAPIENT-assisted horizontal and vertical handover.

## 4.1 SAPIENT-assisted Vertical Handover Scenario

An A/C entering the continental airspace from the ocean one will stick to the SATCOM DL regardless of the availability of the LDACS one, resulting in potentially inefficient communications. On the other hand, an A/C leaving the continental airspace will remain connected to the LDACS DL as long as it has connectivity with it, and will try to switch to SATCOM *after* the LDACS connection drops. Both cases present inefficiencies: in the first one, the A/C will communicate through the SATCOM DL even if better performance can be provided by the LDACS one. In the second case instead, the A/C will react to a connection drop, resulting in possible packet drops and/or increased delays at the application level.To overcome such problems and enforce a make-before-break approach, a SAPIENT-enabled A/C will monitor continuously the KPIs of the available DLs, and will perform vertical handover when needed. This way the A/C will be always able to select the best DL, efficiently selecting the vertical handover point/time.

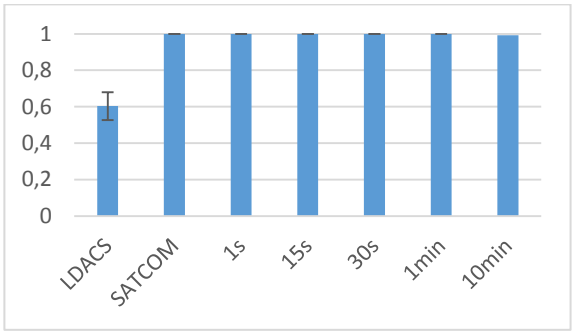


Figure 17; Average connection availability for various periods of SAPIENT reporting.

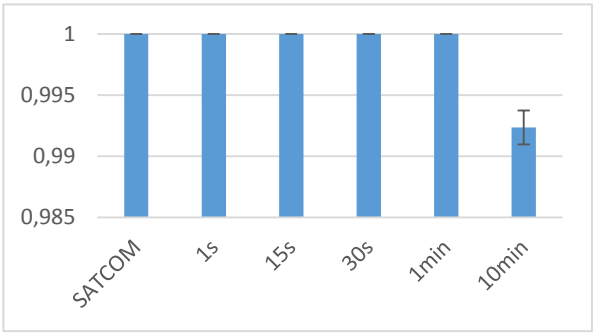


Figure 18: Zoom-in of the previous graph

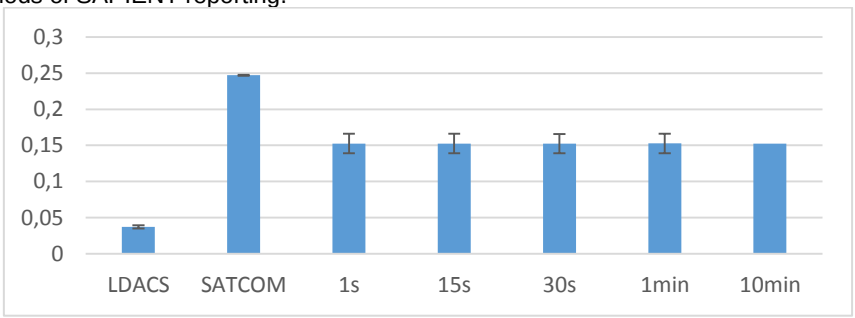


Figure 19: Average application delay for various periods of SAPIENT reporting.

The simulated network includes two ground node running multiple applications, a model of a core network router, a model of the LDACS ground and air segments having 8 GSs, a model of SATCOM

ground and air segments having 4 satellites, a configurable number of multilink-enabled ACs, having traffic generation capabilities and configurable with different traffic priorities. The network is deployed so to reproduce the border between continental and oceanic airspaces, the first one being covered by both LDACS and SATCOM and the second one only by SATCOM. The A/C moves alternating between *flight* phases at a constant speed of 900 km/h, and *parking* phases on the ground. Ground applications transmit packets of fixed size (500 bytes for lo-pri, 100 for hi-pri) with a fixed inter-packet time of 20 ms. When enabled, the A/C SAN will periodically evaluate vertical handover, by comparing the measured SINRs.
The performance of the SAPIENT system is compared against two baseline scenarios, where the selected DL is always the LDACS or SATCOM one respectively. Figure 17 shows the connection availability for the oceanic-continental transitions (assuming that no other unavailability causes are present) over a 24-hour simulated time. The system that always stays on LDACS (leftmost bar) loses connectivity whenever the A/C is deep in an oceanic zone, whereas the one that always uses the SATCOM (second bar from the left) can communicate in both oceanic and continental zones. The other bars show the availability seen by an A/C using the SAPIENT system, at various periods. Figure 18 zooms into the previous graph, showing that the SAPIENT reporting period affects the availability. In fact, a *very high* reporting period (10 mins) allows for losing connectivity and cannot support effectively a make-before-break approach during the transition from continental to oceanic. One may misinterpret the above graphs to construe that only the SATCOM DL, which encompasses both zones, could be used without the need for vertical handover. However, Figure 19 shows that the average delay of SATCOM is considerably higher than an LDACS's, hence it might be preferable to switch to the latter for performance reasons, whenever possible. On the other hand, the average delay with SAPIENT is smaller, being a weighted average of LDACS and SATCOM delays.

## 4.2 SAPIENT-assisted Horizontal Handover Scenario

In a large-scale AEROCOM system, the communication load varies over time. The number of served A/Cs and/or the per-A/C load might reach a point where some cells can become overloaded. Thus, packet delays at the application level will get higher due to queueing. Load increase and overload conditions can be modelled by increasing either or both the number of A/Cs and their offered load.
In a baseline scenario, A/Cs will maintain the same DL regardless of the perceived load level. On the other hand, a system running SAPIENT will continuously monitor the load level for each DL and perform a handover when needed. Moreover, load information of each GS can be used also to guide horizontal handovers, possibly balancing the load over different portions of the network.
We simulate two ground nodes running multiple applications, a multilink router in the core network, an LDACS network with 16 GSs, a SATCOM network having 4 satellites, a configurable number of multilink-enabled ACs, having traffic generation capabilities and configurable with different traffic priorities. Two SAPIENT nodes are deployed for the SSP and CSP, located respectively at the edge of the SACTOM and LDACS ground segment. The number of active A/Cs is varied from 10 to 40. The SAPIENT applications within the CME periodically report the load of their GSs to the SAPIENT ground node. The latter periodically generates and reports to each A/C a KPI summary, containing the load information. Finally, the A/C SAN collects KPI summaries, and uses them in conjunction with the measured SINR values to select the best available GS, at periods varying from 1 to 60 seconds.
Figure 20 reports the loss rate for high-priority traffic on the forward link. Frames are considered lost if they are either dropped (e.g., due to decoding errors) or they arrive at their destination later than a predefined (end-to-end) delay threshold, set to 485ms. The figure shows that the baseline has high loss rates with more than a few A/Cs. On the other hand, SAPIENT reduces the loss rate. While benefits are there regardless of the period, a trend is clearly visible: too long a period implies that the reaction to an overload condition is delayed, and frames may get lost in the meantime. A shorter period, instead, guarantees faster reaction, but increases the load on the forward link as well (due to the reporting traffic). This last effect shows up at very short periods (1s in the simulation). This implies that an optimum reporting period can be configured based on the DL load. Moreover, this demonstrates that the overhead due to the SAPIENT signalling can be considered negligible, unless too short periods are used. Figure 21 shows the end-to-end application-level delay distribution for four reporting periods, with 10 to 40 A/Cs. Each bar represents the interval between the 90th (upper edge) and 10th (lower edge) percentiles. The inner lines are the 25th, 50th and 75th percentiles. This graph allows one to estimate the frame loss rate due to missed deadlines, assuming a varying deadline. For instance, a deadline of 1s would see more than 75% frame loss rate in a baseline system with 30

A/Cs, and less than 10% loss rate in a SAPIENT-enabled system with the same number of A/Cs, configured with a 15s reporting period.

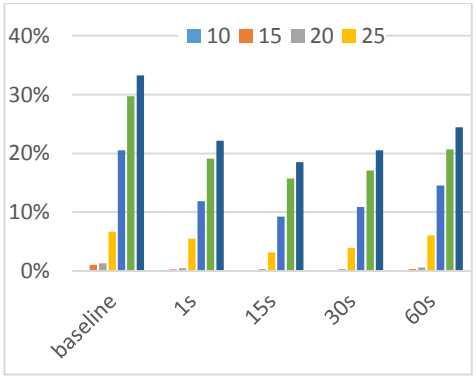


Figure 20: Average Frame Loss for an increasing number of A/Cs and various SAPIENT reporting periods.

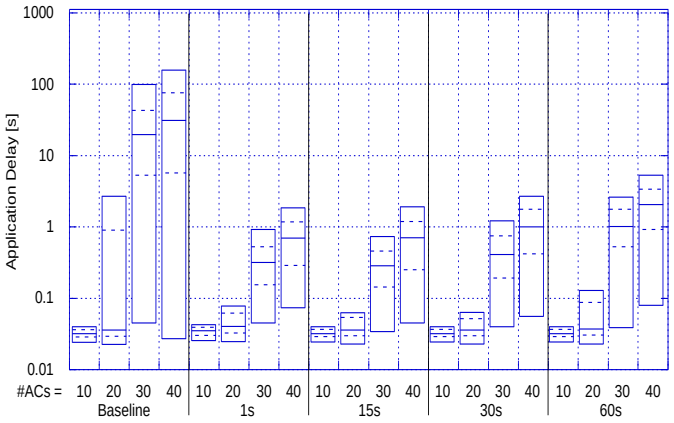


Figure 21: Application delay per A/C, with an increasing number of A/Cs and for various SAPIENT reporting periods.

## 5 CONCLUSIONS AND FUTURE WORK

In this work, we presented a dynamic discrete-event simulator that models an AEROCOM system. This simulator, developed in the framework of the SAPIENT EU project, allows one to simulate the end-to-end performance of applications running over an AEROCOM network including several data links. We have described the modelling of the various network components, including the multilink function, and shown results of the evaluation of the SAPIENT system obtained using the simulator.
The work described in this paper can be extended in several directions. First, refining the models of the components described in this paper, incorporating more details. Second, extending the SAPIENT simulator to incorporate other DLs (namely, AeroMACS and VDL mode 2). Third, evaluating multilink functions and policies, in terms of scalability, effectiveness, and efficiency. Fourth, supporting an advanced evaluation of the SAPIENT system, with a more extensive parameter tuning.

### Acknowledgements

This work was supported by the SESAR Joint Undertaking under grant agreement No 699328 under European Union's Horizon 2020 research and innovation programme. The authors wish to thank all the SAPIENT consortium partners for their work.

### References

[1] SAPIENT-project website; http://sapient-project.eu, accessed Sept. 2017.
[2] OMNeT++ website, https://omnetpp.org, accessed Sept. 2017.
[3] INET-Framework website, https://inet.omnetpp.org, accessed Sept. 2017.
[4] ICAO "Manual on Required Communication Performance (RPC)", Doc 9869.
[5] C. Bongiorno, S. Miccichè, M. Ducci, G. Gurtner, "ELSA Air Traffic Simulator: an Empirically grounded Agent Based Model for the SESAR scenario", 5th SESAR Innovation days, 1-3 Dec, 2015, Bologna, Italy.
[6] B. Niehoefer, S. Šubik, C. Wietfeld, "The CNI Open Source Satellite Simulator based on OMNeT++", 6th International OMNeT++ Workshop, 2013, Cannes, France.
[7] Li Xiangqun, *et al.*, "OMNeT++ and Mixim-based protocol simulator for satellite network," Aerospace Conference, 2011 IEEE, Big Sky, MT, 2011, pp. 1-9.
[8] B. Newton, J. Aikat, K. Jeffay, "Simulating large-scale airborne networks with ns-3", in Proceedings of the 2015 Workshop on ns-3 (WNS3 '15), New York, NY, USA.
[9] VDL mode 2 Capacity and Performance Analysis, PDF available at http://tinyurl.com/y97rtd6k
[10] M. Sajatovic, et al., "Updated LDACS1 System Specification", SESAR JU, Brussels, Belgium.