Satellite link emulation platform for aeronautical application validation

Fabien Garcia, Alain Pirovano, Mathieu Magnaudet

Satellite Link Emulation Platform for Aeronautical Application Validation

F. Garcia* , A. Pirovano† , M. Magnaudet‡

ENAC : Ecole Nationale de l'Aviation Civile, Toulouse, France

In this paper we present our approach toward the validation of new aeronautical applications in satellite communication environment through the use of link emulation and exogenous traffic generation. We describe the hardware and software components of the satellite emulation platform we have implemented for a project named CAPITOLE that has been initiated through the Aerospace Valley world competitiveness cluster. After a short presentation of the overall project, a list of hardware and software components used is given and detailed. Lastly, we explain how the platform itself is validated through network metrology techniques.

I. Introduction

As the air traffic is forecasted to grow, the European Union has launched the SESAR (Single European Sky ATM Research) program. The aim of this program is to encourage innovation in Air Traffic Control through the introduction of state-of-the-art technologies. In this context, the European Space Agency (ESA) is defining a new satellite standard named IRIS dedicated to digital air-ground communications for Air Traffic Management often known as datalink. By 2020 IRIS should contribute to the modernization of air traffic management by providing digital datalinks in continental and oceanic airspace.

In order to take full advantage of this new satellite-based communication solution, several projects have been initiated through the Aerospace Valley world competitiveness cluster. One of these projects, in which we are involved, is the CAPITOLE project co-funded by the Aerospace Valley pole (www.aerospace-valley.com) and the French government (Direction générale de la compétitivité, de l'industrie et des services). This project aims at defining new aeronautical services based on satellite communications, such as real-time weather reports for aircraft pilots. CAPITOLE is mainly concerned with ATC (Air Traffic Control) and AOC (Airline Operational Communication) services, covering aircontroller/pilot communications and airline communications respectively.

In this context, the ENAC/Leopart laboratory has been involved in defining, proposing and implementing a means to estimate and study the behaviour of the new communicating applications that will be developed in the CAPITOLE project in a satellite environment.

There are several ways to validate application performances for a given type of network link, among them simulation, emulation and tests on real-systems.

- Simulation relies on models for both the application and the network and runs in simulated time (as opposed to real-time), its strengths are the re-usability of the tests and the ease of automation, but the results will only be as accurate as the models and will only reflect the aspects which have actually been coded in the models. Simulation is often used for protocol development and testing.

- Emulation relies on deploying real applications on a network that reproduce the characteristics of the real network, allowing deployment of real applications (as opposed to models of applications) with end-users. Again the results will only be as accurate as the representativity of the network emulation, though qualitative results can be produced through the intervention of real users.

*Associate professor, ENAC/Leopart, fabien.garcia@enac.fr.
†Associate professor, Head of Leopart, alain.pirovano@enac.fr.
‡Research assistant, ENAC/Leopart, mathieu.magnaudet@recherche.enac.fr.

- Real-system tests are the most accurate that can be thought of, but in the case of satellite systems, they imply prohibitive costs that prevent their use in the definition phase of new services.

On the one hand because there is a need to test real applications, considering CAPITOLE project requirements, and on the other hand, because the IRIS project is still in the definition phase, the emulation approach has been chosen. The aim of the satellite emulation platform developed by the ENAC/Leopart laboratory is to estimate and measure the behaviour of new applications that will be developed in the CAPITOLE project. It has to be noted that this platform emulates the satellite link at packet level.

II. System and service characterization

The emulation of the system is based on the characterization of the link between the ground and the aircraft. As we decided to develop an exogenous traffic generator, there is also a need to characterize network traffic flows in these communications.

II.A. ESA's telecommunication program IRIS

The main objective of the ESA (European Space Agency)/IRIS program is to propose a new air-ground communication system for ATM (Air Traffic Management). By 2020, this system should be the satellite-based solution for the Single European Sky ATM Research (SESAR) program. IRIS will contribute to the modernization of air traffic management by providing digital datalinks to cockpit crews in continental and oceanic airspace. The ESA/IRIS program is composed of three phases:

1. The first is the definition phase. The main objective is to provide inputs on satellite capabilities in order to make important choices for the subsequent phases.

2. The second aims to develop and validate the new satellite-based communication system dedicated to the future European ATM system.

3. The third and final phase will mainly focus on verification and certification of the pre-operational system. According to the ESA, it will also provide technical support to the deployment of the satellite-based communication system.

The IRIS Program completed its definition phase (Phase 1) in january 2009 and is now in the first part of phase two[1]. Considering the actual IRIS program outcomes, some key points still remaining undefined, accurately characterizing the satellite link which will be proposed as the future system is difficult and even infeasible.

For the moment, according to Ref. 1 several solutions have been considered during Phase 1 activities such as the use of specific techniques for a new dedicated standard, the adaptation of the AMSS standard or the adaptation of DVB-S2/RCS. Nevertheless, some elements may allow to list, estimate, or bound the parameters which have to be considered to characterize the future IRIS satellite link above the link level.

The first is that the solution is based on geostationary satellites. Considering line-of-sight communication, the one-way delay introduced by the system will be at least 250 milliseconds. As the satellite link should be shared by several aircraft, adequate access control has to be defined in order to prevent or avoid packet collision. This access control technique could introduce jitter on the delay, considering existing satellite systems and specific equipment constraints. The capacity should be between 10 kbit/s and 10 Mbit/s. For instance current Inmarsat AeroH satellite aeronautical systems offer a bit rate of around 10 kbit/s.

Finally, as the bit error rate is generally known, a close approximation of the PER (packet error rate) can be deduced. The bit error rate is the ratio of received bits corrupted by noise or interference divided by the total number of transferred bits. As several corrupted bits may be found in a single packet, PER will be less than or equal to BER. Generally, we can consider that bit error rates on satellite links are at least in the order of 10^{-12} on more recent systems and at worst in the order of 10^{-5} in AMSS (Aeronautical Mobile Satellite Service), proposed by Inmarsat, for instance. For a data packet length of N bits the expected PER P_p can be expressed relative to the bit error probability P_e as :

$$P_p = 1 - (1 - P_e)^N \tag{1}$$

In our case, we made the assumption that all packets containing at least one corrupted bit will be discarded at link level. Hence, from now on we will consider the PER as the loss ratio parameter of the link.

It has to be noted that one-way delay, jitter, bit rate, and PER parameters may have different values or statistical properties considering the forward or return channel of the satellite communication system.

II.B. Datalink Traffic characterization

The EUROCONTROL/FAA document "Communications Operating Concept and Requirements for the Future Radio System" (COCR)[2] defines the Future Radio System (FRS) sub-system that encompasses the Physical and Data Link Layers of the satellite system. Our satellite link emulator aims at reproducing the characteristics of the network as seen by packets arriving at the interface between the FRS and Network layer in terms of delay, loss and throughput. The delay and loss parameters can be set at fixed values or be defined in terms of a probability function that they should obey. The throughput on the other hand can be set as a pair of values, the maximum data rate and the Maximum Transmission Unit in the FRS. The reason we do not stick with fixed delay and loss on a satellite link (where jitter is mostly negligible and losses are mainly due to interferences) is the traffic generated by other aircraft and applications sharing the same link. To further improve the way this traffic is taken into account, we include an exogenous traffic generator in our platform. This generator allows the user to emulate the competition for resources on the satellite link with realistic traffic. To allow for easy and accurate definition of this traffic, both packet and application flow level generation will be supported. That is to say the user can define exogenous traffic as stochastic laws on inter departure time and size of packets or simply specify the type and number of application flows he wishes to emulate.

To do so, the first task consists in determining properties of the network traffic that will be emulated. Studies underlining interesting information related to aeronautical datalink traffic properties and specificities have been conducted.

As explained in Ref. 2, datalink traffic may have different properties considering flight phase and services provided. Ref. 3 describes five phases : airport, departure, cruise, arrival, airport. Moreover, three classes, with different properties and requirements, are generally considered in the aeronautical datalink context :

- ATC (Air Traffic Control), mostly dedicated to controllers and pilots.
- AOC/AAC (Airline Operational Communications/Airline Administrative Communications), mainly for airline companies
- APC (Airline Passenger Communications), for passengers applications.

In order to estimate the necessary capacity of a satellite communication system dedicated to aeronautical datalink, Ref. 3 gives the communication pattern of one aircraft conforming with COCR. Some properties of the expected traffic are given :

- the expected message size is from 77 to 21077 bytes long
- the peak rate for one aircraft , considering some short messages with stringent latency requirements, should be around 15 Kbps
- the average throughput per aircraft should be a few bps

Based on these conclusions and the aircraft traffic density given in Ref. 4, an estimation of the information throughput for all aircraft flying simultaneously over a given area during the busiest day of the year is possible.

III. Satellite link emulation

Our satellite emulation platform consists of two software components : the link emulator which emulates the link characteristics (delay, throughput and losses); and the traffic generator which allows for tests on a loaded network. These two components run on linux servers with dedicated networks. The following sections present first the hardware part of the platform, and then the two software components.

III.A. Hardware components

Our satellite emulation platform consists of three Linux servers and two 1000 Base TX Ethernet switches. One Linux computer hosts the link emulation application, the other two host the traffic generator. The over-provisioning of the switch allows us to enforce the fact that competition for resources should only happen on the satellite link. This platform uses IPv6 as its network protocol, which is consistent with ICAO recommendations for ATN/IP systems[5]. The use of these standards allows a user to simply plug an Ethernet cable into a station in order to test an application.

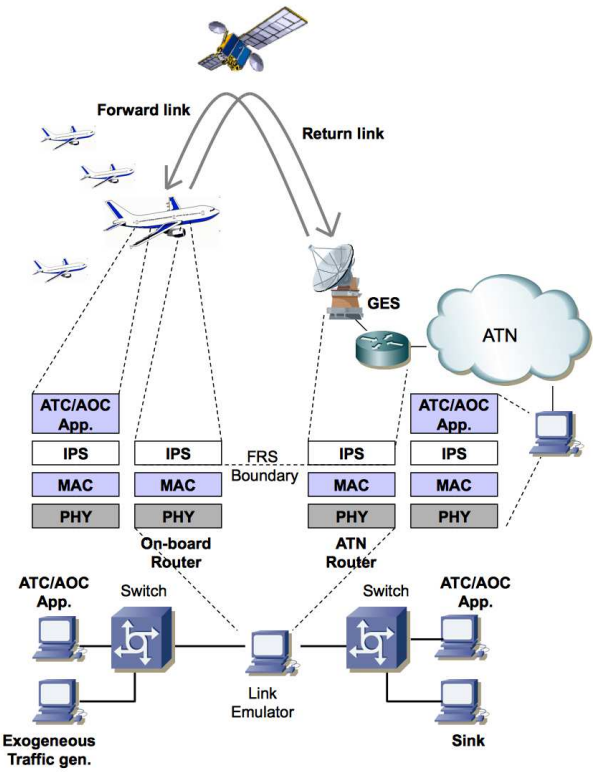


Figure 1. Emulation platform overview

III.B. Software component

III.B.1. Satellite link emulation

The philosophy of the software conception has been to give priority to the user experience and to make the emulator usable for an average user, unaware of the subtleties of the Linux kernel parameters for traffic control. So the GUI of the software allows the user to specify networks features and to keep ignoring the internal Linux machinery.

As shown in figure 2, three parameters can be specified :

1. Delay: propagation delay in millisecond. It has associated jitter and probability law, as well as a correlation percentage.
2. Loss: percentage and correlation percentage.
3. Throughput: in bits per second.

To emulate a new link, the user has to associate a set of parameters to two of the available network interfaces, one for each traffic flow direction. The parameters can be different for each selected interface. Once selected, an emulated link can be launched, stopped, restarted and deleted.

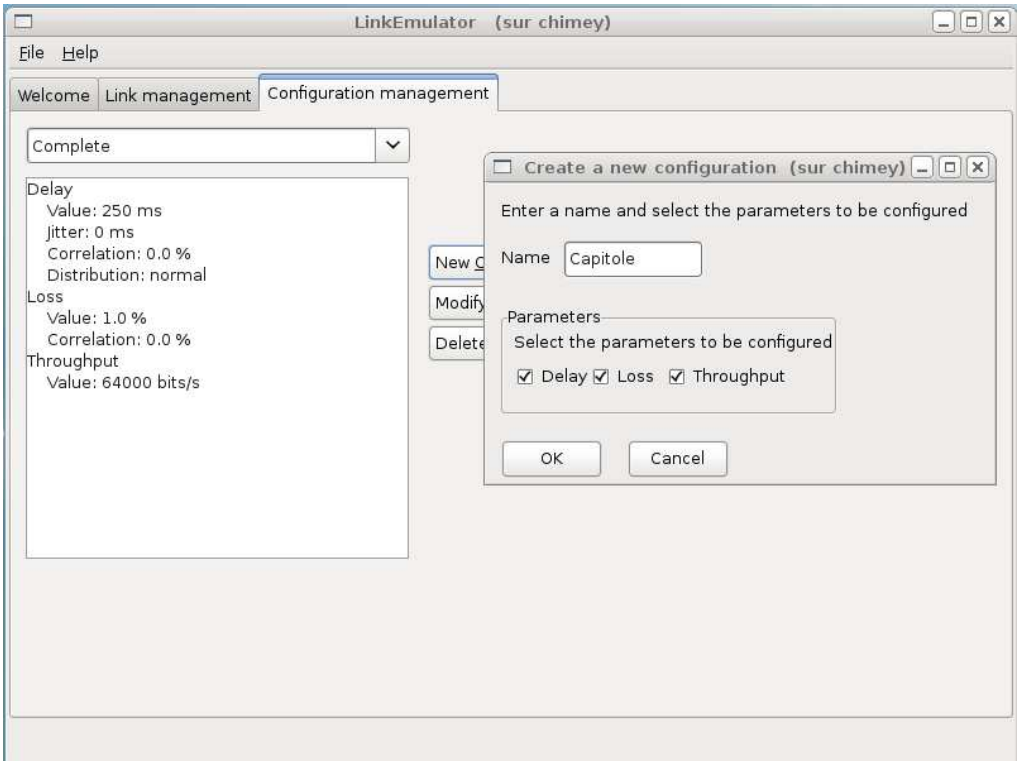


Figure 2. Link emulator GUI

In order to gain independence from any external tool, we chose not to build the network emulator on top of the command line tool *tc* which is the standard tool for Linux traffic control . We chose instead to directly address kernel modules via the C open source library libnl. Besides libnl, the various components of the software were written in JAVA and interfaced with the C component by using the Java Native Interface framework. Thus, the data coming from the user are stored in two XML files via a persistence layer which is based on the Java Architecture for XML Binding (JAXB). The graphical user interface is based on the SWT/JFACE libraries, chosen for performance and aesthetic reasons. Attention has been paid to make a clean separation between the various components such that it should be possible to substitute another implementation to one of them without changing the others. The final link emulation architecture is shown in figure 3.

The two modules shown in the kernel space in figure 3, netem and tbf, each emulate some parameters of the link. Netem[6] is used to emulate delays and losses, whereas tbf[7] is used to emulate the delay of the link. The Token Bucket Filter (TBF) is in charge of slowing down packets so they respect a given maximum rate. In the terminology introduced in Ref. 8, TBF is a traffic shaper. Other scheduling methods beside TBF could have been used but several sources[7,9] and the netem author himself point to TBF as the preferred bandwidth limitation tool. We will show later in the validation section that our platform would benefit from a new approach.

III.B.2. Exogenous traffic generator

The exogenous traffic generator is built around two pieces of software : a sink, to be launched on the receiver side, and the traffic generator, to be launched on the sender side. The sink is a simple graphical tool that allows the user to open a list of ports, it also provides some information about the incoming packets such as their number and the inter-arrival time.

On the sender side, the traffic generator allows the user to define and control a list of UDP flows. In the current version, a UDP flow is characterized by :

1. a destination address and port number,

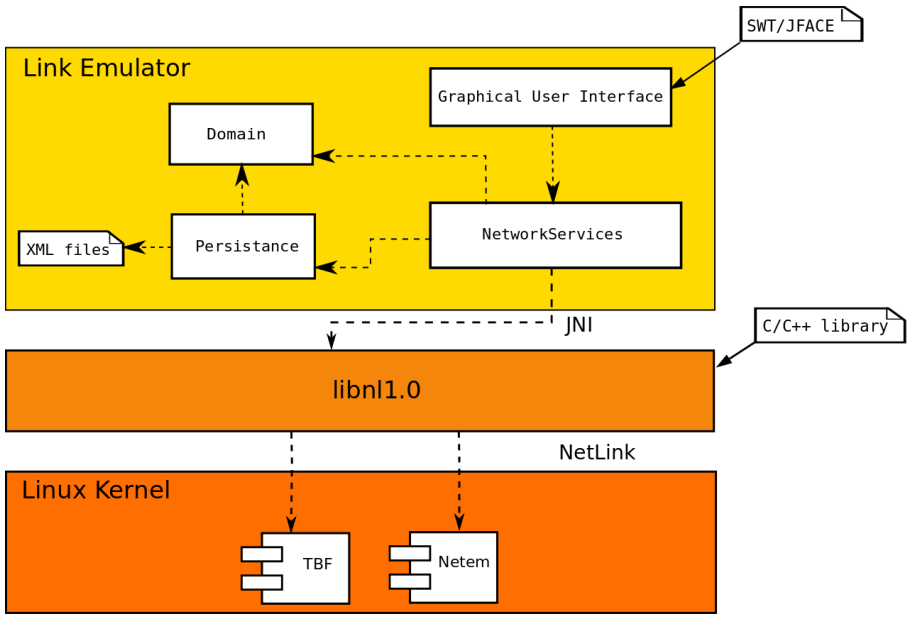


Figure 3. Software Architecture of the link emulator

2. a statistical law governing the packet size (constant, uniform, normal or exponential),

3. a statistical law governing the inter-arrival time (constant, uniform, normal or exponential).

The last two parameters allow us to easily implement the communications as described in the COCR document[2].

In the next version of the generator, each UDP flow will be attached to a source type (e.g. : aircraft with AOC and ATC services) and we will provide, as a new functionality, a source duplication tool. This way we will be able to, for example, emulate the traffic generated by a fleet of aircraft each having the same traffic profile.

IV. Platform Validation and First results

In this section, we first present results concerning the validation of our emulation platform, and then some user feedback on perceived application quality in the context of satellite link communication.

IV.A. Platform validation

As we said in previous sections, our link emulation software allows us to emulate throughput, losses and delays. In order to validate our platform, we will therefore test each parameter in turn and measure its values in several configurations. It should be noted that these tests are not sufficient to fully validate our platform. They will, however, give us confidence in the behaviour of our emulation but cannot test all possible values for each parameter. It will therefore be necessary to repeat these tests each time new links are to be emulated.

In the following sections, we first present the validation tool we developed, and then the methodology and the results of the validation for each parameter.

IV.A.1. Network measurement tool

In order to validate our platform, we have developed a measurement tool that is able to measure the one-way delay; throughput; and loss ratio of a given network link between two hosts. This tool, launched at the two ends of the link, generates UDP packets at a given rate and in one direction. Each packet is marked with its sequence, its size and the time at which it was sent. At the reception side, the packets are just stored along with their reception time.

In order to compute the one-way delay, both hosts need to synchronize their clocks, we used the Network Time Protocol[10] (NTP) for this task. Throughput calculations have been made both as an average and as an instantaneous value. The average value is based on the throughput of the measurement session. The instantaneous value is computed as the size of a packet divided by the inter-arrival time between the current packet and the next. Loss calculations were made on average using the sequence numbers of the packets received.

This tool will soon be made available as open-source software under the name *OWDMeter* (One-Way Delay Meter).

IV.A.2. Throughput validation

To validate throughput emulation, we used *OWDMeter*, which uses timestamped UDP packets in order to measure delays, throughput and losses on a link. Using UDP packets sent at a rate higher than the one configured in the emulation, we were able to measure the throughput at the receiver side. It is to be noted that our tool outputs the rate at the application level whereas our emulation takes place at the network level, it is therefore necessary to take into account the overhead of IPv6 and UDPv6 protocols. IPv6 adds a 40 byte header and UDP adds an 8 byte header so for a 1000 byte packet at application level, we send a 1048 byte IP datagram.

We tested two different rates :

- 10Mb/s which is typical of low speed Ethernet Local Area Networks;
- 64Kb/s which is the expected throughput available on one channel of the future IRIS satellite.

Our first batch of tests showed rather poor results for the emulation at 64 Kbits/s. This was traced to a problem with the Linux queuing system. We first sized the queue so as to reflect classical router queues[11] of around 100 packets (for a total of 150000 byte queues). Such a queue, if full, would take 18.6 s to empty at 64 Kbit/s, which is unreasonable. During these tests, we noticed that long queues tended to be discarded by Linux (all their packets being destroyed at once), generating bursts of losses. As this behaviour was not the intended one, where once full the queue would reject only new packets arriving, we decided to limit the queue size so as to have a maximum waiting time of 1s[a] in the queue.

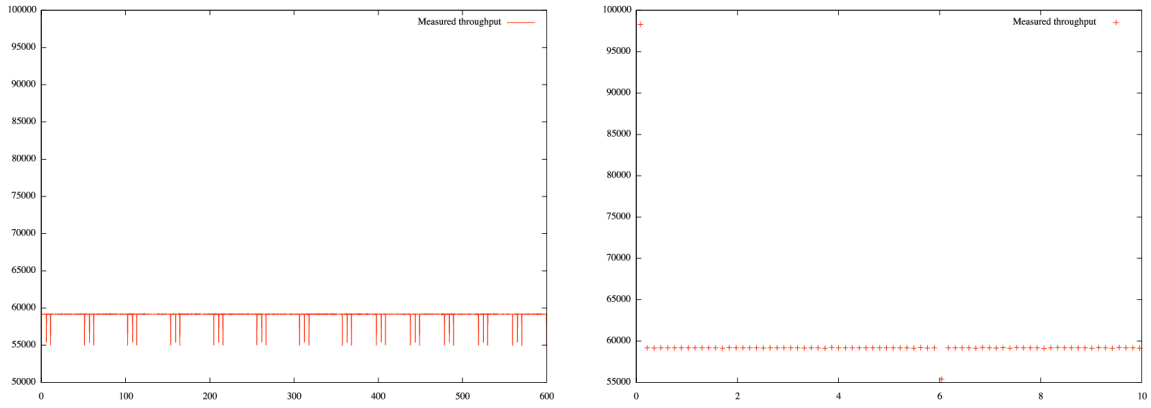
Table 1 sums up the configuration used for each scenario as well as the measured throughput at the receiver side. Figure 4(a) shows the instantaneous measured throughput (at user level) during the length of a test.

Configured emulation throughput (IP Level)	packet size (IP Level)	Test duration	Average Measured Throughput (User Level)	Average Measured Throughput (IP Level)
10Mb/s	1048 bytes	600 s	9.256 Mb/s	9.7 Mb/s
64Kb/s	1048 bytes	600 s	59.147 Kb/s	61.986 Kb/s

Table 1. Throughput validation results

These results show that the throughput of the link was limited by TBF. The accuracy of the limitation, though high enough for our purpose, could be enhanced by modifying the shaping method. Actually it has already been noted in Ref. 9 that the sensitivity of TBF to small irregularities in input traffic leads to packet losses above the expected ratio, which in turn causes a lower average throughput. Furthermore, as can be seen in figure 4(b), the token bucket technique allows small transient bursts to be sent at a high rate. These two problems lead us to conclude that a new throughput limitation method should be used. At the time of writing, such a technique is still to be implemented.

[a]The value of 1s was found empirically not to cause problem, further research in the subject will have to be done.



(a) Whole test (b) Detail of the ten first seconds

Figure 4. Measured instantaneous throughput as a function of receiving time

IV.A.3. Packet loss validation

Unlike throughput that can be thought of as an almost continuous value (using a sliding window to make measurements), loss is a stochastic process that can only be measured statistically.

The validation for the loss parameter has also been done with the same tool, using UDP packets with a 1000 byte payload. Three values of the loss parameter have been tested, 50 percent, 10 percent and 1 percent. The two first values, represent unrealistic values on common networks (apart from heavily congested networks), the last one being the value of packet error rates found on a satellite link with 10^{-5} binary error rate and packet of 1048 bytes.

Table 2 sums up the results of these tests.

Configured emulation loss (%)	packet size (IP Level)	Packet received	Measured loss (%)
50%	1048 bytes	1000000	49.98%
10%	1048 bytes	1000000	10.01%
1%	1048 bytes	1000000	0.99%

Table 2. Loss validation results

These results confirm the implementation of loss processes in our emulation platform.

IV.A.4. Delay validation

Delay emulation has been tested with a first version of our measurement tool that allowed only for delay measurement. It worked in the same way as the last version, sending timestamped UDP packets and noting their reception time. These tests were made with two NTP synchronized hosts.

We tested both fixed delay emulation and variable delay emulation with normal distribution. Each of these have been tested with three values of the delay :

- 30 ms, which is typical of the delay experienced on the french national research network from Toulouse to Paris;
- 100 ms, which is typical of the delay found on links from Europe to the USA east coast;
- 250 ms, which is typical of geostationary satellite links.

The measured delays have been averaged for 10000 packets, sent at a rate of 40 packets per second. The NTP error shown has been taken as the NTP error given at the beginning of each test and has been considered constant throughout the 250 seconds of each test. Monitoring of NTP errors between the two hosts during the tests has validated this assumption.

Table 3 shows the results of these tests. These results show that the emulation is correct within the limit of our measurement precision. A final step, done for each of the three normal law distribution tests, was to test the match between the measured distribution and a computed normal distribution. To this end, we used a scilab script to plot both the measured distribution (after normalization) and the computed normal law distribution. The results are shown in figure 5 and show an almost perfect match.

Random distribution	Configured delay	Configured jitter	*measured av.delay*	*measured jitter*	NTP reported error
Fixed	30 ms	0 ms	29.7ms	0.0612ms	2 ms
	100 ms	0 ms	99.7	0.0632ms	4 ms
	250 ms	0 ms	249.7ms	0.0609ms	4 ms
Normal	30 ms	10 ms	29.8ms	9.8s	1 ms
	100 ms	20 ms	100.7ms	19.8ms	2 ms
	250 ms	10 ms	249.9ms	10.1ms	2 ms

Table 3. Delay validation results

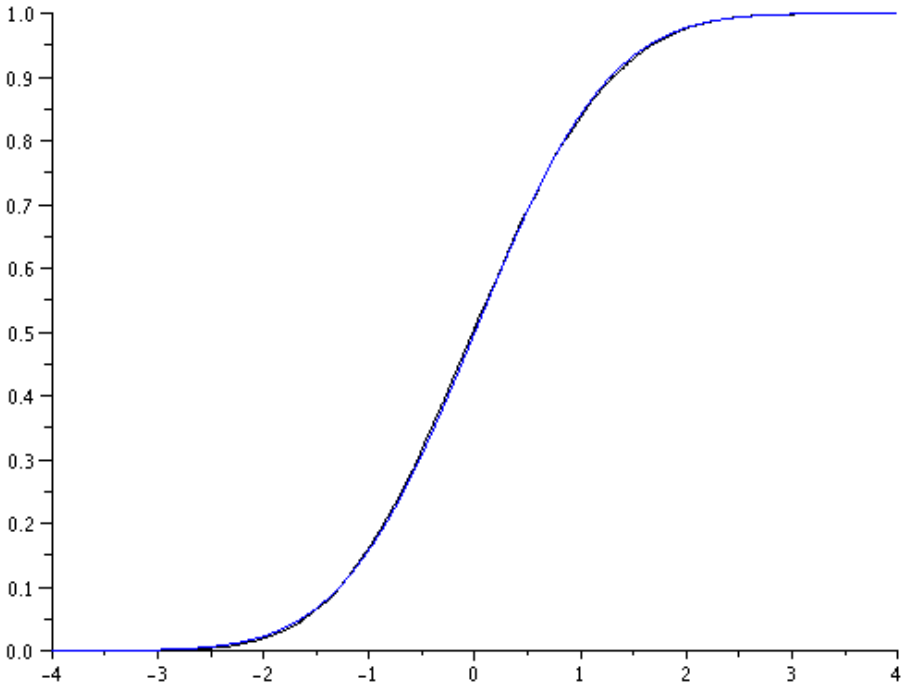


Figure 5. Comparison of measured Cumulative Distribution Function (black) and computed normal law CDF (blue)

IV.A.5. Traffic generator validation

For the traffic generation validation, passive measurements techniques will be used. The traffic generator will be successively configured with several traffic profiles and its output will be recorded in a file via the *tcpdump* tool. We will then use a script to extract size and time information for each packet. As the parameters for the traffic generator consist in a probability distribution on both the size and inter-arrival time of the packets, a scilab script will then be used to extract their characteristics (i.e. : mean value and standard deviation) and to verify that they fit to the configured probability distribution for each parameter[b].

At the time of writing, these tests are yet to be done.

[b] the same script that was used for delay can be reused here.

IV.B. User experience evaluation

After the validation of the link emulation part of our platform, we wished to test some common applications in order to evaluate the satellite channel from the application users perspective. We used four applications in these tests, a web application, an ftp application, an interactive remote login application (*ssh*) and a video streaming application (using *vlc* and a live webcam capture). These applications have been chosen so as to have both TCP and UDP protocols, and both interactive and streaming (one-way) applications.

The link we emulated was a satellite link with 64 Kbps throughput, 250 ms one-way delay and 10^{-2} loss ratio. For each application, the users have noted the perceived quality of the application.

WEB BROWSING Web browsing performance on a satellite link depends heavily on the site being browsed. Web pages containing mainly text load in an acceptable time, whereas pages with more images will take a longer time. So far, the conclusion are unsurprising as the reduced bandwidth lengthen the download time of each image. The delay induced by the satellite is not really noticeable for the user.

SSH REMOTE LOGIN Secure Shell (*ssh*) allows a user to remotely and, above all, securely connect to an host. We tested both command line interaction and display redirection[c] with different results. Although the delay induced by the satellite link is noticeable in command line, due to the display showing half a second later than the typing of the commands, working through this remote connexion is possible. In remote display mode though, the delay and the low throughput do not allow a easy interaction on a GUI.

FTP TRANSFER We tried FTP both to evaluate the interaction level and the download speed of a file. Although the interaction was at the same level as with ssh in command line, the download speed was, unsurprisingly, really low. It took almost an hour to download a 15 MB file. For this application, no TCP enhancing technique (such as described in Ref. 12, 13) can improve the download time, as the limiting factor resides in the 64 Kbps throughput rather than in the TCP mechanisms[d].

VIDEO STREAMING We used a webcam to send live video (using a UDP stream) to a *vlc* server on the other side of the emulated satellite link. The video was successively configured to send 64 Kbps, 128 Kbps and 512 Kbps toward the server. In each cases, the results were similar, the video shown was of low to average quality. The delay induced by the satellite is irrelevant in this case as it acts only as an time offset, not a limiting factor.

These tests, although their results are not new to the network community, emphasize our view of application validation through emulation : we now have a good understanding of the usability of each of these applications in a satellite communication environment. By letting users test new applications in a realistic environment, we can validate their use in several conditions without resorting to expensive, real-systems tests.

V. Conclusion

This paper presents our approach of aeronautical application validation in a satellite communication environment. This approach uses both link parameters emulation as well as traffic generation to reproduce the characteristics of a loaded satellite link, allowing us to test applications in a most realistic way. Our system being built as a generic platform for network emulation, it allows us to represent various satellite links by entering the right parameters in our GUI. Validation tests have been run on our platform using both active and passive measurements of the various parameters to extract relevant values with error margins. These have shown the platform reliability in terms of emulation of the satellite link. For example, the tests on delay emulation show no error above the precision advertised by the NTP synchronisation process. Lastly, the use of a simple GUI allows users with only basic network knowledge to emulate different links in an easy way.

Future works on this platform will consist of two phases : the first part, already in progress, consists in the enhancement of the exogenous traffic generator as described in section III.B.2 (and its subsequent validation); the second part will be to write a linux kernel module to shape packets so as to replace the TBF shaper. On the one hand, the addition of source types, and the possibility to replicate them will allow

[c] This consists in displaying a remote application on the local computer

[d] The ($bandwidth * round\ trip\ time$) product for this link is below the standard 64KB for a TCP receive window.

users to easily define the load of the emulated link as a function of the number of aircraft and their current state. On the other hand, the replacement of the TBF shaper will increase the accuracy of the throughput emulation.

Acknowledgments

The authors wish to thank Jean-Baptiste Laborde, Loïc Jurdy and Rémi Blondin, students at ENAC and Jane Jean Kiam, trainee at the Leopart laboratory, for their help in the validation tests. We also wish to thank Nadine Atwood and Graeme Riddick for their review and comments on this paper.

References

[1]ESA, "IRIS Project, http://telecom.esa.int/iris," .

[2]EURCONTROL/FAA, "Communications Operating Concept and Requirements for the Future Radio System, COCR version 2," 2007.

[3]Ricard, N., "ESA Iris Programme Overview," http://www.ffg.at/getdownload.php?id=3581, April 2009.

[4]EUROCONTROL, "Eurocontrol Long Term Forecast," http://www.eurocontrol.int/statfor/public/standard_page/forecast3_reports.html.

[5]ICAO, "Doc 9896, Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocols." 2010.

[6]Hemminger, S., "Network Emulation with NetEm," *Linux Conf Au*, April 2005.

[7]Hubert, B., Graf, T., Maxwell, G., van Mook, R., van Oosterhout, M., Schroeder, P. B., Spaans, J., and Larroy, P., "Linux advanced traffic control HOWTO rev1.43," 10 2003.

[8]Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W., "RFC 2475: An Architecture for Differenctiated Services," December 1998.

[9]Stattenberger, G., Braun, T., Scheidegger, M., Brunner, M., and Stüttgen, H. J., "Performance evaluation of a Linux DiffServ implementation," *Computer Communications*, Vol. 25, No. 13, 2002, pp. 1195 – 1213.

[10]Mills, D., "Network Time Protocol (Version 3) Specification, Implementation," 1992.

[11]Appenzeller, G., Keslassy, I., and McKeown, N., "Sizing Router Buffers," *In proceedings of ACM SIGCOMM*, 2004, pp. 281–292.

[12]Jacobson, V., Braden, R., and Borman, D., "TCP Extensions for High Performance," 1992.

[13]Border, J., Kojo, M., Griner, J., Montenegro, G., and Shelby, Z., "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135 (Informational), June 2001.