Extending NCTUns simulator to support Mobile Networks

Juliano V. Quaglio, Tetsu Gunji, Celso M. Hirata
Instituto Tecnológico de Aeronáutica
Praça Marechal Eduardo Gomes 50
12228-900 Sao Paulo, Brazil

juliano.quaglio@gmail.com, tgunji@uol.com.br, hirata@ita.br

ABSTRACT

NCTUns is a Linux based network simulator/emulator which has a great deal of features such as the possibility to execute real world applications without modifications, and provision to model a wide range of network devices using real TCP/IP network stack. However NCTUns only supports simulation/emulation of *mobile hosts* instead of *mobile networks*. Hence, using the conventional NCTUns, it is not possible to model more elaborated scenarios, including C4I2SR systems which have mobile networks, for example, aircrafts with internal embedded networks communicating with external networks (terrestrial control center). Motivated by this restriction, we propose and describe an extension of NCTUns in order to allow the modeling and emulation of systems which require two instances of NCTUns. The extension allows the emulation of C4I2SR systems scenarios with video stream and composed of *mobile networks* using distributed computers. The approach permits improving the confidence on the modeling.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modeling**]: Applications.

General Terms

Algorithms, Design, Experimentation, Verification.

Keywords

C4I2SR, NCTUns, emulation, mobile networks.

1. INTRODUCTION AND MOTIVATION

C4I2SR (acronym for Command, Control, Communications, Computers, Intelligence, Information, Surveillance and Reconnaissance) systems are defined by NATO (North Atlantic Treaty Organization) as integrated systems of projected doctrines, procedures, organizational structures, personal, equipments, installations and communications to support the commander in chief in the command and control of operations and military activities [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIMUTools 2009, Rome, Italy
Copyright 2009 ICST, ISBN 978-963-9799-45-5.

C4I2SR systems have functionalities that allow the operators to get effectiveness on their operations, reducing material and personal necessary to do the tasks, raising the probability of success in the missions under their coordination.

A C4I2SR system is composed of several types of processing nodes:

- Platforms – airborne (including ISR), terrestrial and maritime;
- Sensors – air transported radars, terrestrial radars, optical and multi-spectral sensors, electromagnetic sensors; and
- Terrestrial control centers.

One of the innovative technologies that needs to be developed in this context, is the integrated transmission of video/images (generated by video cameras, infrared and SAR radar – Synthetic Aperture Radar), voice (generated by the voice communication system of the aircraft) and messages (with geo-tagged data, status and commands), using tactical broadband data link such as TCDL [2](Tactical Common Data Link) between the ISR aircrafts (airborne platform) and the terrestrial control centers.

One of the main difficulties of building a C4I2SR system is its complexity and its wide range of equipments. Hence, the cost of developing such system is also very high.

In order to design and test, avoiding unnecessary resources, simulation and emulation studies of the networks involved are recommended. Such studies are an economical way to estimate the traffic, the system behavior and the feasibility of the topology proposed to overcome some type of constraints such as bandwidth, propagation delay or BER (bit error rate) among others.

Considering all the subsystems of C4I2SR systems, the most interesting one for a simulation/emulation study is the C4I2SR subsystem scenario consisting of a terrestrial control center and an aircraft embedded network. The study may help to test both the influence of the airborne segment in the communications and the feasibility of different protocols used by military data links.

Simulation and emulation differ in the following. Network simulation works by building a complete model of the network in software. The downside of simulation is that it must be possible to simulate all components in order to produce results. A complex simulation model is difficult to verify, and its validity may be called into question. On the other hand, network emulation is a technique whereby some functions of a network or

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

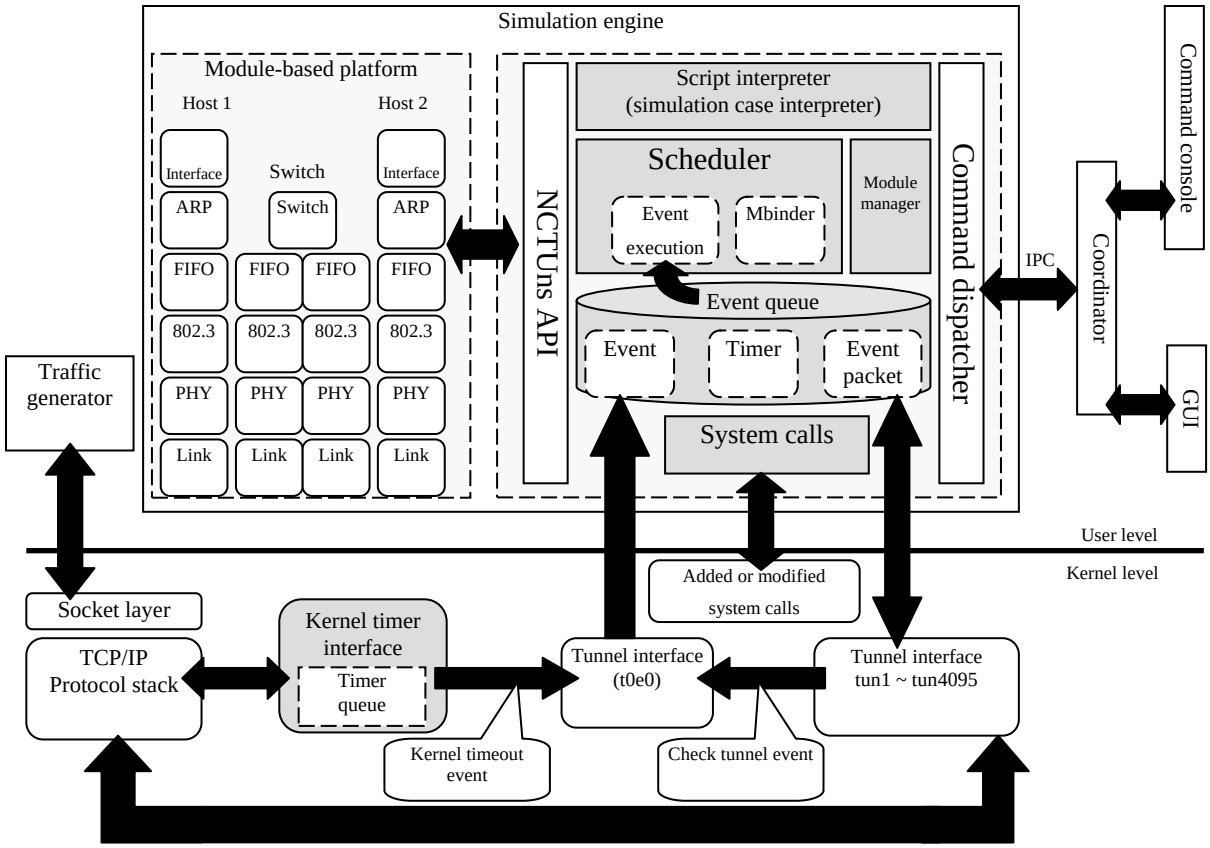

Figure 1 - NCTUns architecture [6].

application are reproduced in real time using software. The software presents the same interfaces to other system components that the emulated components themselves would present. Network emulation requires significant software and hardware resources in order to generate results, due to its real time nature [3].

Hence, to use real network components during the tests and to test real audio and video communications, the use of emulation instead of simulation would be more appropriate and trustable. Such emulation could be made by using a software network simulator/emulator, such as NCTUns (National Chiao Tung University network simulator) [4].

The emulation of the C4I2SR subsystem includes the internal (inside the aircraft) and the external networks (terrestrial and communication between the terrestrial hosts with the aircraft).

The main problem that arises using NCTuns is the simulation of two networks, since the aircraft network (mobile network [5]) moves with respect to the terrestrial network but the internal aircraft's network elements are static positioned with respect to each other.

In this paper, we present an implementation of an emulated solution to test video (preferably MPEG-4 [7]) and audio transmission between an aircraft embedded network (mobile network) and a terrestrial control center, using two instances of the NCTUns network simulator/emulator, running on two different computers. The modification also allows distributing the workload of emulation networks, making it possible to split a simulation case among several nodes, each one running an instance of NCTUns.

The rest of the paper is organized as follows. Section 2 describes the architecture used to achieve the proposed goal. Section 0 presents a case study to demonstrate the applicability and the feasibility of the architecture. Section 4 describes some performance and validation tests. Finally, Section 5 presents the conclusions, comments and future work.

2. ARCHITECTURE

The choice to use NCTUns has been made considering a comparison study of network simulators [3]. The main characteristics of the simulator that deserve to be mentioned are [6]:

- Open source code;
- Availability of a highly integrated GUI (Graphical User Interface);
- The possibility of execute real world applications without modification;
- The kernel re-entering method to use some real world protocol stack on the simulation/emulation, such as TCP/IP; and
- Its discrete-event based simulation engine.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

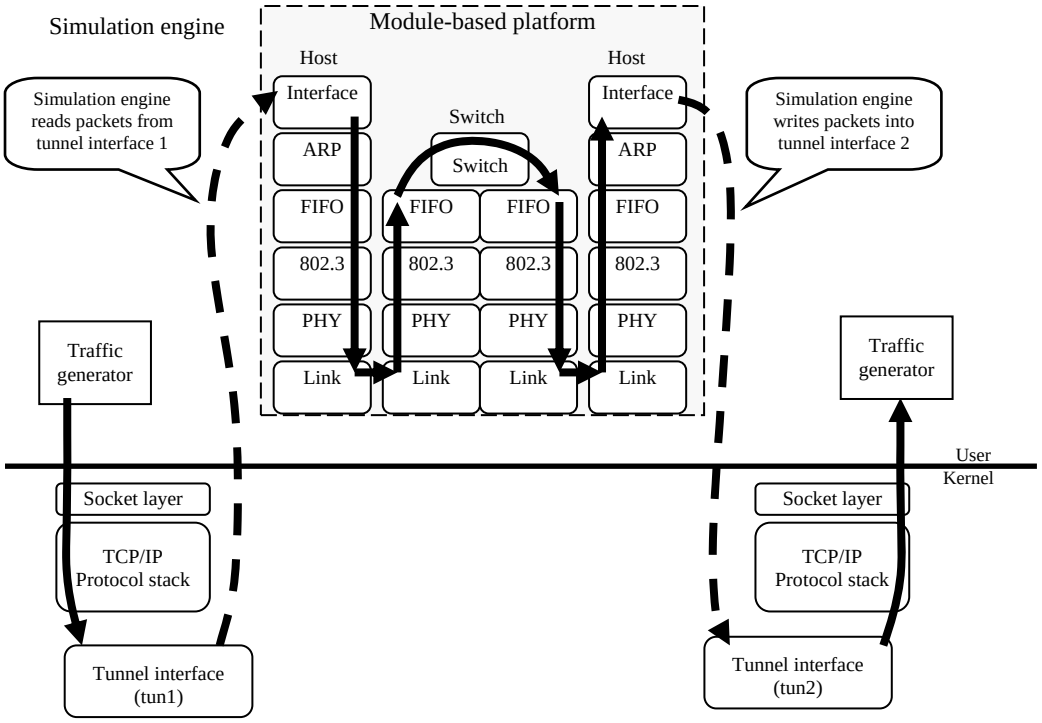

Figure 2 - Data exchange between two hosts [6].

The availability of the source code was a crucial point to make a smooth integration, since the version of the simulator/emulator without the proposed extension does not meet the requirements of emulating an external and an internal network subsystem.

The NCTUns network simulator is an open-source simulator/emulator implemented mainly using C++. It is designed to run on Fedora Linux. The version used in this study, NCTUns 5.0, runs on Fedora 9. The only proprietary part of the simulator is its GUI. Despite of the fact that the GUI is not open to change, it was possible to create a successful solution.

In the next subsections we present some considerations, the general NCTUns architecture, and the extensions necessary to achieve the proposed goal.

2.1 Considerations

The NCTUns network simulator is composed of several different modules as illustrated in Figure 1.

The simulation engine is a user-level program and has complex functions. It functions like a small operating system. Through a defined API (application programming interface), it provides useful and basic simulation services for protocol modules. The services include virtual clock maintenance, timer management, event scheduling, variable registration, script interpreter, and IPC (Inter-Process Communication) interface. The simulation engine also manages all of the tools and daemons that are used in a simulation case and decides when to start the programs, when to finish them, and when to run them. Figure 1 shows an architecture diagram of NCTUns.

For this paper, however, the most relevant parts of NCTUns are the virtual tunnel interfaces and the kernel modifications.

A detailed view of the mechanism of data exchange between two hosts is presented in Figure 2. It shows the flow path that a packet takes when it is exchanged between two traffic generators via the module-based platform. When the packet is read by the simulation engine from tunnel interface 1 (tun1), the packet follows the trace of Figure 2 and then the simulation engine inserts it into tunnel interface 2 (tun2). Finally, the kernel sends it to the traffic generator.

NCTUns can act in simulation or emulation mode. To change between the two modes, it is necessary to modify the speed setting. In the simulation settings menu, it is possible to choose between two speed modes: “As fast as possible” or “As fast as the real-world clock”. Emulation cases work using the speed set to “As fast as real-world clock”.

To add support for the simultaneous emulation of an internal and an external network, basically, we must tunnel the packet coming from one network to another using the interfaces on the two networks to act as interfaces of a router. In this way, only one interface of that new “router” is visible in each emulation case.

In the next subsection we present the modification of the architecture as well as the changes to the files on the simulation cases.

2.2 Detailed design

When executing a simulation/emulation case, the NCTUns software generates some files that are used to pass information to its modules/processes, such as the simulation engine and the modified kernel.

As far as this study is concerned, the most important files are the ones that describe the routes added to the routing table and the IP

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

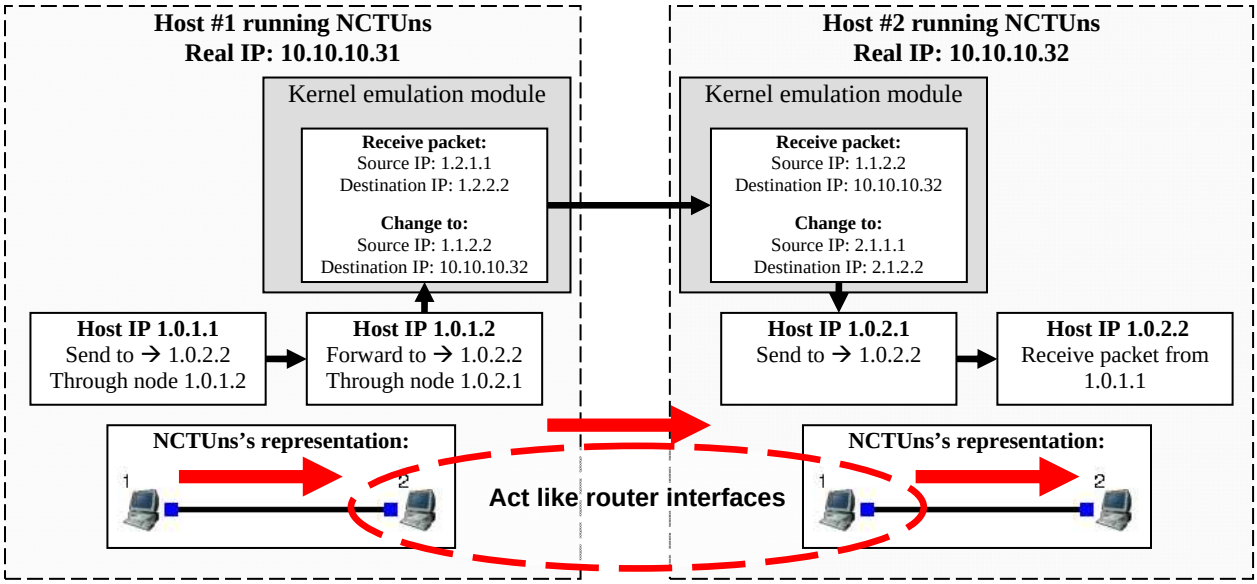

Figure 3 - Scheme of the translations needed to work with more than one instance of NCTUns.

addresses of real hosts and its corresponding simulated hosts in NCTUns.

The bases of the planned modifications are the already implemented features for using two classes of real equipments: hosts and routers.

These features uses the concept of divert sockets [8]. The divert sockets enable IP packet interception and injection on end systems as well as on routers. Packets are intercepted on an IP layer and can be made available for user processes outside the kernel via a modified version of raw sockets. The most used implementation of divert sockets [8][9][10] rely on the IP firewall mechanism for packet filtering, but NCTUns has its own implementation.

In the version 5.0 of NCTUns, the packets are intercepted by a module in the kernel and are not forwarded to user-space. The module itself handles the situation based on a configuration file present in each emulation case, doing the network address translations.

For example, suppose an emulated host with IP address 1.0.1.1 in the emulation and the real IP address 10.10.10.55. In this case, the kernel module changes the destination address of every packet sent to the emulated node from 1.0.1.1 to 10.10.10.55. It also changes the incoming IP address of every packet that the real host is sending to emulated hosts from 10.10.10.55 to 1.0.1.1. The changes allow the real host to communicate with virtual hosts/equipments of the emulation sending and receiving packets without any changes in the application running on that host. To simulate the router, some special conversions between real network interfaces IP and emulated ones are needed, but the setting remains the same.

Therefore, in order to make the desired emulation case work, it is necessary to add a new class of equipment to this kernel module, allowing the virtual interfaces in each NCTUns instance to act like a "virtual router". It means that we must change the source code from the kernel module that is responsible for the address translations in NCTUns.

At the same time, the modification of the routing information file of the emulation case is required. This kind of file is automatically generated by the GUI of NCTUns. Considering that the GUI source code is not available, we cannot change the behavior of system to reflect the needed modification in the routing file. Hence, we must directly modify that file. In the internal network we must add the routing path from the internal hosts towards "virtual router" interfaces. The routes from external network's hosts to internal network's ones, passing through the "virtual router" interface must be added as well.

It should be noted that the NCTUns uses an internal representation of IP address in the form of "S1.S2.D1.D2". In this scheme, S1 is the source subnet ID, S2 is the host number on source subnet, D1 is the destination subnet ID and D2 is the host number on destination subnet. For example: if the source IP address of the packet is 1.0.**1.1** and the destination is 1.0.**2.2**, the S.S.D.D address is 1.1.2.2. This scheme is used to avoid conflicts when adding routes to the routing table [6], since the routing tables for all nodes inside the emulation are added to a unique kernel routing table.

It is very important to understand this scheme to modify the routing file correctly.

This scheme was also used in the communications between the real hosts, to preserve the emulated source IP and emulated destination IP. Figure 3 shows an example of communication between two different hosts running NCTUns. In the example, the translation using the "S1.S2.D1.D2" IP scheme is made when the kernel module on host #1 changes the destination IP of the packet from 1.2.2.2 to 10.10.10.32. In order to preserve the original source and destination IP addresses, the source IP address of the packet is also modified from 1.2.1.1 to 1.1.2.2. Then, it is possible to know that emulated host 1.0.1.1 is sending the packet to emulated host 1.0.2.2.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

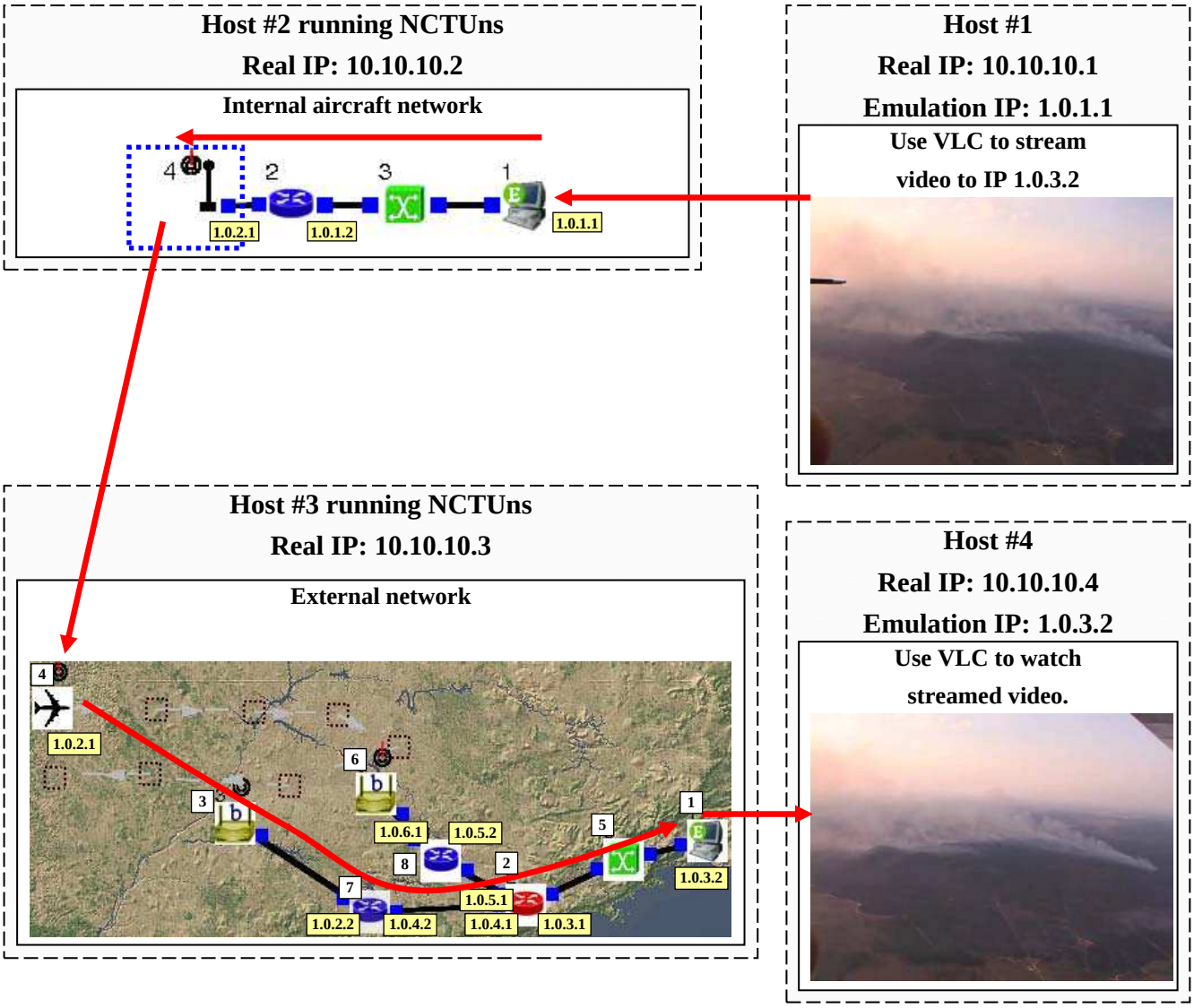

Figure 4 - Case study, showing the internal aircraft network, the external network and the real hosts.

Upon receiving the packet, the kernel module in host #2 uses the source IP address in the "S1.S2.D1.D2" format and the IP address from its "virtual router" interface to reconstruct the emulated source and emulated destination IP address.

In the next section we present a case study to test the modifications.

3. CASE STUDY

3.1 Considerations

The aim of the case study is to successfully stream a video from a real host passing through two real computers running NCTUns (emulating two different and connected networks) and reach another real host which receives the video stream with the proper delay and BER added by the instances of NCTUns.

For this purpose, we designed two different networks, pictured in Figure 4. In this figure, two modifications were made: in the internal aircraft network, the node #4 does not exist in the real emulation case on NCTUns and, in the external network, the node #4 is an "802.11(b) mobile node (infrastructure mode)" node, only its icon was changed in order to represent an aircraft. The two modifications were made to make it easier to understand.

To stream the video, we used the software VLC. VLC media player (initially VideoLAN Client) is a highly portable multimedia player (runs on top of Linux, Windows and Mac OS X among other operating systems) for various audio and video formats (like MPEG, DivX/Xvid, and Ogg) as well as DVDs, VCDs, and various streaming protocols. In recent years its use has also increased as a server to stream live and on demand video in several formats through various private networks and Internet [11].

3.2 Creating and running the case

The scenario created represents a surveillance aircraft with a video camera flying over the Brazilian states of *Mato Grosso do*

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

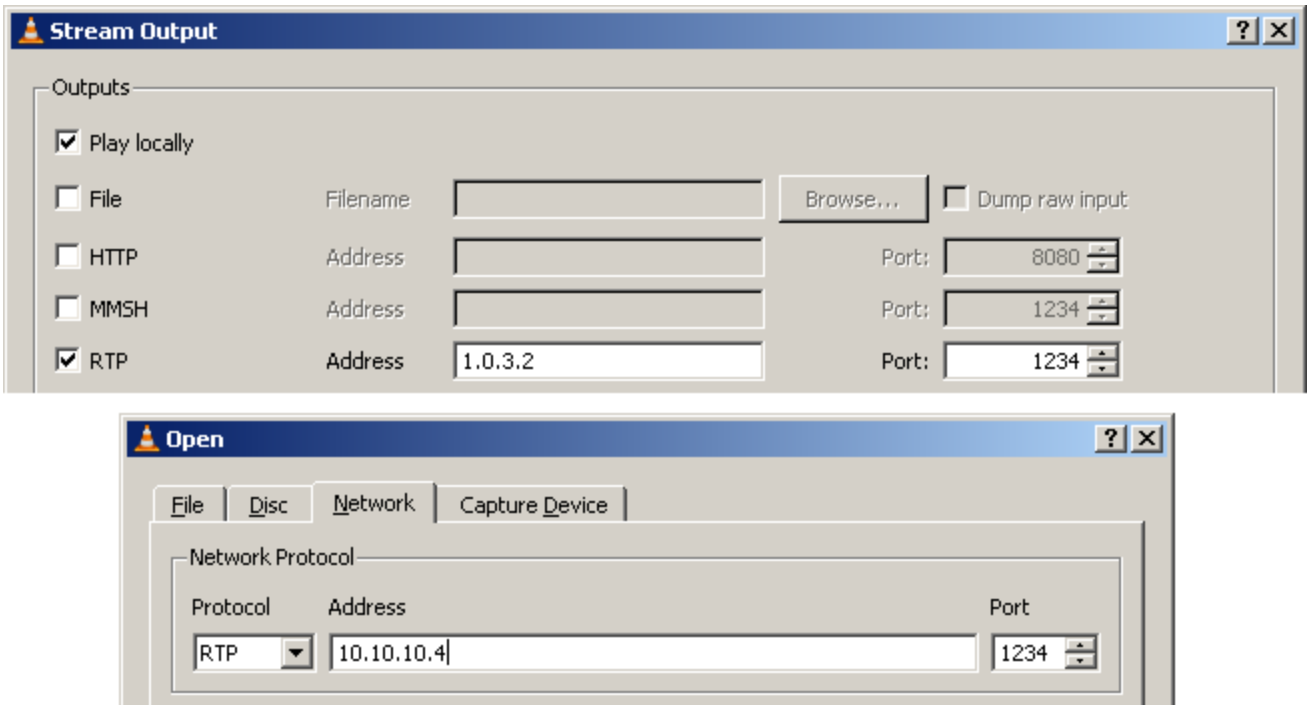

Figure 5 - Configuration of VLC server (top) and VLC client (bottom) to execute the case study.

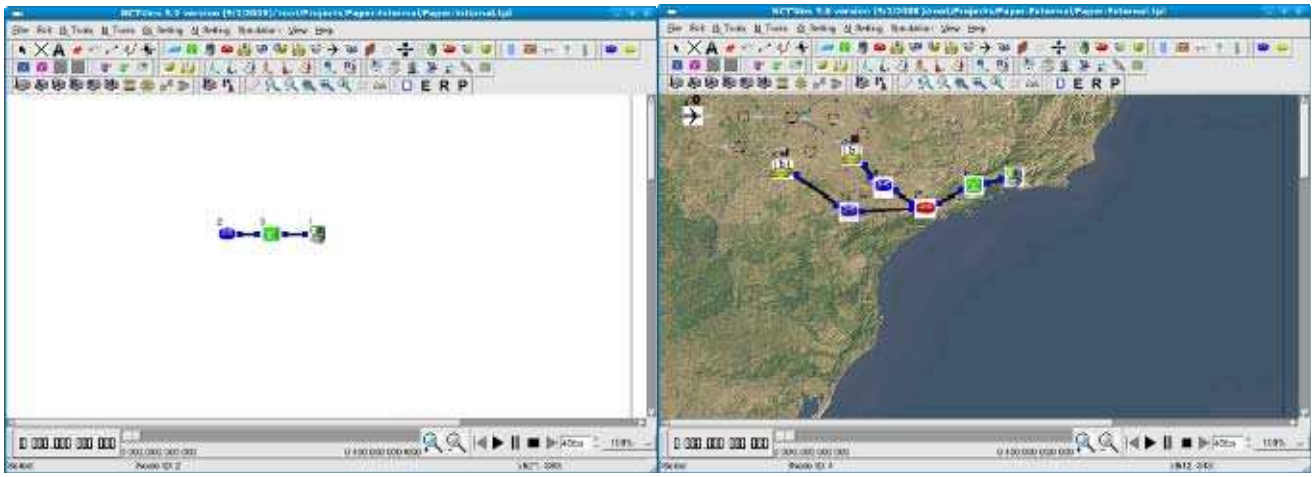

Figure 6 - Screenshots of both NCTUns instances. The left side shows the internal network representation and the right side shows the external network.

Sul and *Sao Paulo*. There is one terrestrial antenna in the southwest and another in the center of *Sao Paulo* state. The aircraft streams real-time data to the terrestrial antennas which are connected with a terrestrial control center in the *Rio de Janeiro* city through a terrestrial infrastructure.

In the specific case shown in Figure 4, the aircraft is flying over the state of *Mato Grosso do Sul* and detects a forest burn, transmitting its real-time video to the control center at first through the antenna in the left of the Figure 4. When the aircraft is out of the range of that antenna, the transmission occurs using the other one.

In the next paragraphs the modifications needed to execute this case are shown.

The VLC server runs on host #1 (real IP address is 10.10.10.1 and emulation IP address is 1.0.1.1) and the client runs on host #4 (real IP address is 10.10.10.4 and emulation IP address is 1.0.3.2). Both instances of VLC run on top of Windows XP.

One instance of NCTUns is running on host #2 (with real IP address of 10.10.10.2) and represents the internal aircraft network. This internal network is composed by an emulated host, a switch, a QoS DiffServ router [12] [13] and the representation of the communication antenna.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

The other NCTUns instance represents the external network, and is running on host #3 (with real IP address of 10.10.10.3). This network is composed by an emulated host, a switch, a router, two QoS DiffServ routers, two access points and the surveillance aircraft.

While changing the connection from one antenna to the other , in order to keep the same IP address, thus not loosing the connection, the aircraft network uses mobile IP [5] [14] concept.

To allow the traffic from host #1 to reach networks with IP addresses such as 1.0.X.X, we must add a route in this host, directing the traffic going to 1.0.X.X to go first to IP address 10.10.10.2, which is running an instance of NCTUns.

In the same way, we must add a route on host #4 to direct the traffic from this host to go to the emulation of the external network, which IP address is 10.10.10.3.

We also must change the routing files from the two emulation cases. In NCTUns, after we change the operating mode to "Run simulation", the emulation files are generated in folder "CASE_NAME.sim". There, we must pick the files named "CASE_NAME.emu" and "CASE_NAME.srt-l". The first file handles the emulation network address translations and the other file contains the routes to add to the kernel routing table.

In the ".emu" file, we must add the information for the kernel module to translate correctly the addresses in the communication of the two NCTUns instances. In host #2, initially, the file should contain the following lines:

```
#nctuns external routing table file
host    10.10.10.1     1.0.1.1
```

After those lines, we must add the lines which describe the "virtual router". For each pair of hosts in the external and internal network, there must be two lines, one of describing the translation of outgoing packets and the other about incoming packets.

Regarding the route file, we must direct the traffic going from one network to the other. An example, considering node #1 from internal aircraft network (1.0.1.1) sending packets to node #1 of the external network (1.0.3.2):

```
route add -net 1.1.3.0/24 gw 1.1.1.2
```

Note that the gateway address is used according to the "S1.S2.D1.D2" scheme, meaning that the host 1.0.**1.1** is sending packets to the gateway (router) address 1.0.**1.2**.

To run the emulation we must first start the two instances of NCTUns and then choose the appropriate settings to VLC server and client.

The configuration on the two instances of VLC is shown in Figure 5, while Figure 6 shows screenshots of the two NCTUns instances.

It is worth mentioning that the ".emu" and ".srt-l" files must be changed every time we change NCTUns's operating mode to "Run simulation".

In this work, the airborne communication used was the 802.11 (b) protocol already implemented in NCTUns. This had to be made because NCTUns does not provide tactical communication modules yet. In order to correctly emulate C4I2SR networks, a tactical data link protocol, such as TCDL, must be added to the NCTUns.

The case study results proved that the emulation is consistent with what we were expecting. The video streamed between the two hosts suffered from delay and BER added in the emulated networks, and the result was visible within the video stream on the receiving node.

It was also possible to see the influence of mobile IP in communications, observing the mobile node disconnection from one access point and reconnection to the other access point. This resulted in a few seconds without video in the receiving node.

4. VALIDATION AND PERFORMANCE TESTS

In order to perform additional verification of the accuracy and the performance of the modifications, we conducted other tests that are described in this section.

The equipment used in the tests is: one notebook with Intel Core 2 Duo T5550 (1,83 GHz) processor and 2 Gb of RAM (tests with one or two NCTUns instances) and a desktop with Intel Pentium Dual CPU E2180 (2,00 GHz) processor and 3 Gb of RAM (tests with two NCTUns instances). When the test involved two NCTUns instances, the computers were connected using an Ethernet cable (100 Mbps).

The first test was made to obtain the delay caused by the simulation due to the modifications in the NCTUns' kernel module. For this purpose, we created the case shown in Figure 7.

First, using one NCTUns instance only, we executed the command "`ping 1.0.1.1`" on host #2 to obtain the round trip time between the two virtual hosts. Considering that there are ten links between the simulated equipments, the round trip time is calculated as the sum of the ten individual link's delay two times, since the request and reply messages need to go through ten links each one. The expected delay for an individual link is calculated as the delay configured in NCTUns for that link plus the packet transmission time calculated using the link speed (10 Mbps) and the packet size (84 bytes). Since the configured delay in each of the ten links is 0,5ms, the total expected delay for the ping command is:

$$20 * [(84 \text{ bytes} / 10 \text{ Mbps}) + 0{,}5 \text{ ms}] = 11{,}3 \text{ ms}$$

Running the case, the obtained results are:

Average	11,8 ms
Standard deviation	1,3 ms

Considering the average time, the difference of approximately 0,5 ms can be explained by the clock synchronization precision within NCTUns (1ms) and the time consumed by other system-related running processes.

The next step was to run the same case divided into two NCTUns instances, as in Figure 7 (b) and (c), to check whether the delay can be negligible or not. The NCTUns configuration is the same as described above. Hence, the expected delays should

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

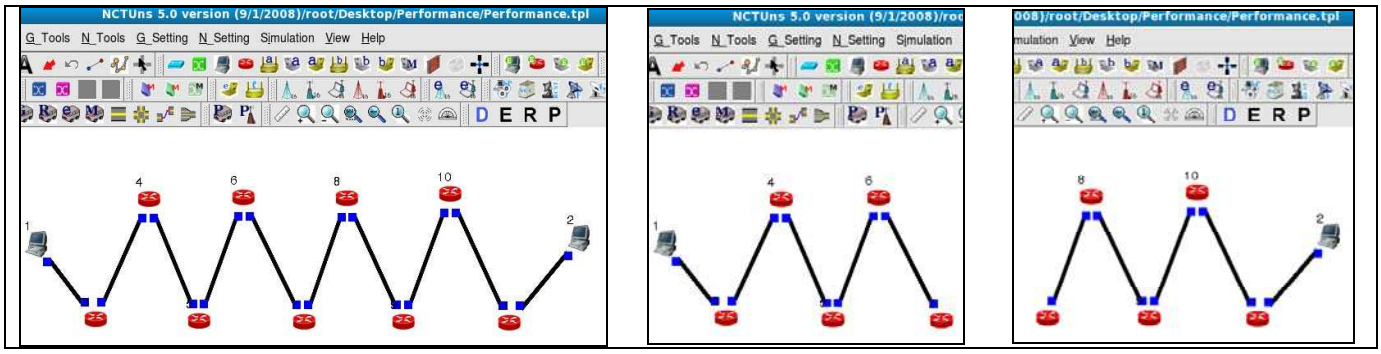

Figure 7 (a), (b) and (c) - Screenshots of the test cases: first, the test in a single NCTUns instance; the same case distributed into two instances.

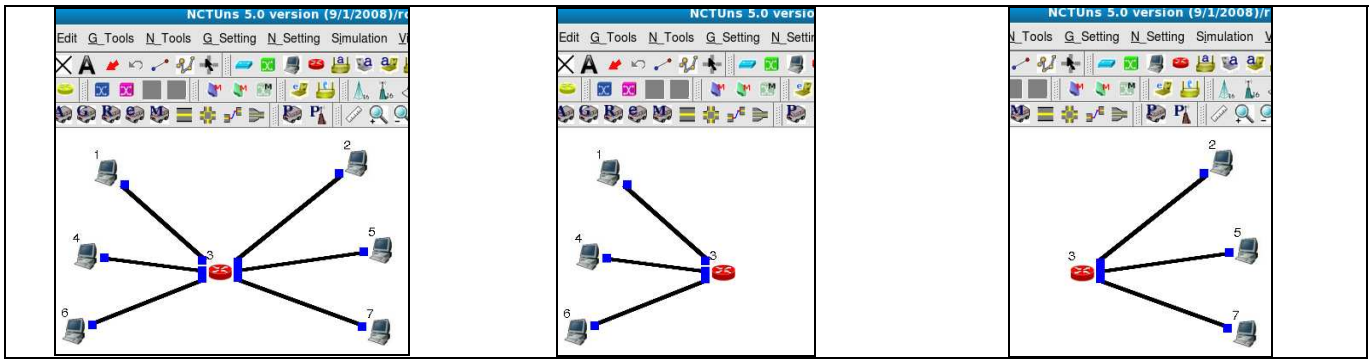

Figure 8 (a), (b) and (c) - Cases used to test the performance enhancements.

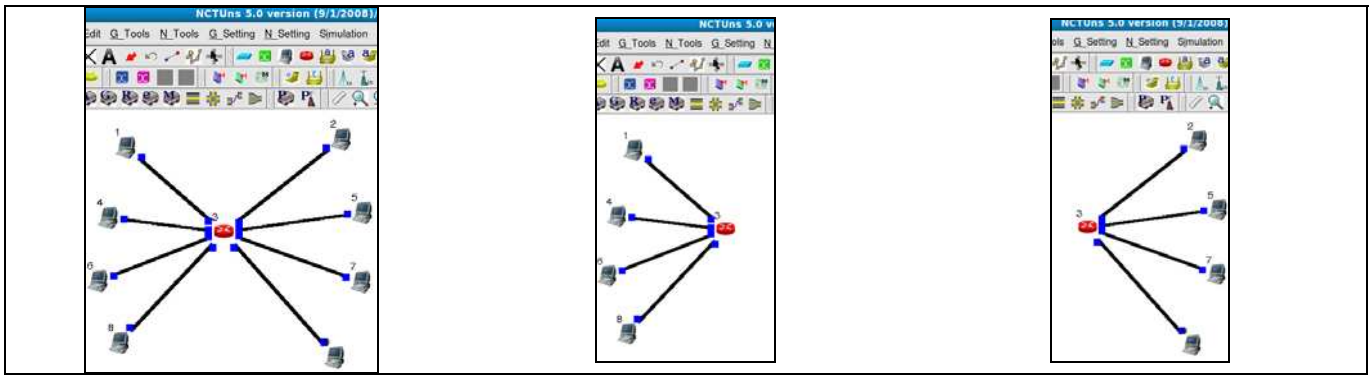

Figure 9 (a), (b) and (c) - Cases used to test the performance enhancements.

be as close as possible to the value obtained in the first test to indicate that the modifications are feasible enough.

Running the case, the values obtained are:

Average	12,8 ms
Standard deviation	4,2 ms

Analyzing the figures, we can observe that the average round trip time was only 1 ms greater than using a single NCTUns instance. We consider this as a good result, since 1 ms is also the synchronization precision time between the simulation and the real-time clocks in NCTUns. This difference allows the modifications to be used with other real emulation cases without loosing the correlation to the real world. We observe that the standard deviation is greater than the value from the first test. The difference can be caused by another system process using the network between the two computers running NCTUns. Disabling the processes should decrease the value, improving the results.

We also built a second case test to verify the performance boost when distributing a simulation case between two NCTUns instances. We created the case shown in Figure 8 (a). In this case, the nodes 1, 4 and 6 send data to nodes 2, 5 and 7 through TCP connections, at the same time.

To analyze the performance of a simulated case, we compared the total real-time used by NCTUns to the NCTUns' internal simulation time. To do this, we modified the clock synchronization code to print the times and the time difference

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

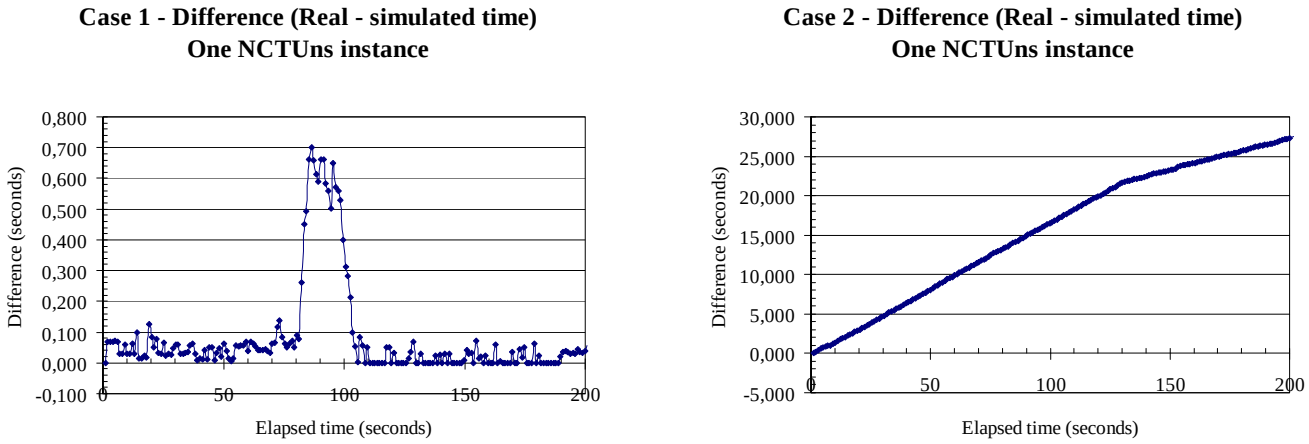

Figure 10 (a) and (b)– Performance analysis for one instance NCTUns emulation.

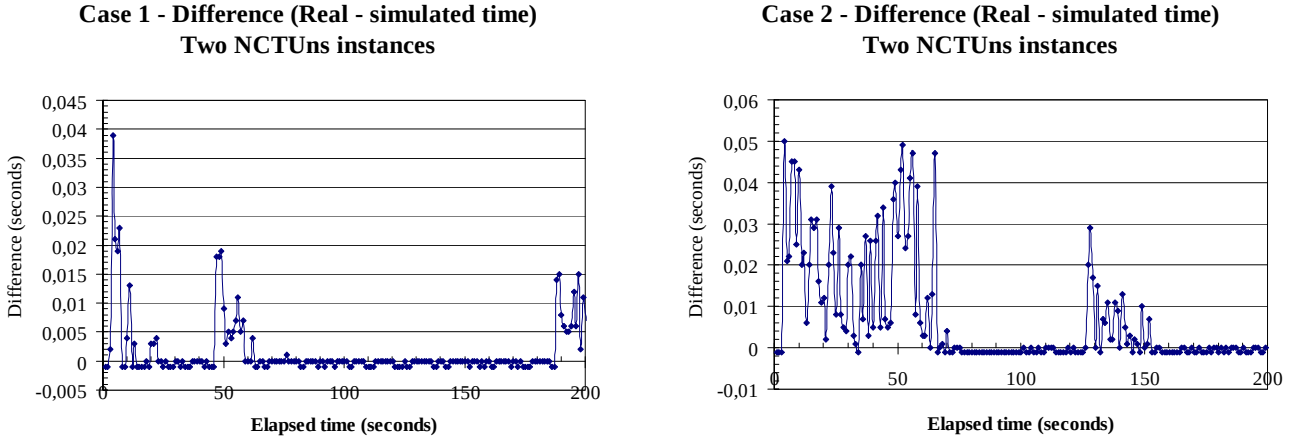

Figure 11 (a) and (b) – Performance analysis for two instance NCTUns emulation.

into a file, making it possible to further explore the results. Here we should explain that the synchronization code just tries to "slow down" the NCTUns simulator when the case is running faster than the real time. When the case is running slower, the simulator does not do anything. Considering this, we can conclude that, without our modification, it is difficult to use NCTUns as an emulator with a case having a large number of hosts or traffic generators, since the simulator would not be able to synchronize the time correctly.

In this evaluation, we first executed the cases in Figure 8 (a) and Figure 9 (a) using one NCTUns instance, obtaining the results in Figure 10 (a) and (b). From the plots we can observe that the simulator can handle the case 1 (a) synchronized with real time. However, in case 2 (b), the single NCTUns instance cannot simulate the network as fast as the real clock, increasing the time difference as the simulation runs.

Afterwards we divide the case into two different simulations to be executed within two NCTUns instances, as in Figure 8 (b) and (c) and Figure 9 (b) and (c). As one can see in Figure 11 (a) and (b), the new approach is able to execute the cases 1 (a) and 2 (b) synchronized with real-time. This enables NCTUns to execute more complex cases in emulation mode (thus using real network equipment). Also, the emulation mode with two NCTUns instances can be used to both execute large cases and meet performance requirements of emulation of the cases.

5. CONCLUSIONS, COMMENTS AND FUTURE WORK

In this paper we propose extensions to NCTUns network simulator to allow the consistent emulation of C4I2SR systems. The approach used to achieve such a goal can also benefit other emulation cases, allowing large emulations to be divided into smaller cases running on interconnected hosts, each one running an instance of NCTUns.

Regarding C4I2SR systems, future work includes the implementation of tactical data link protocols and aeronautical channel modeling [15] within NCTUns module's code to study the influence of the airborne segment in the communications.

Another important study is to test and discover the best traffic classifications and bandwidth allocations for different types of streams (video or audio or data) within the C4I2SR system scenario using the QoS DiffServ routers.

The availability of GUI source code also would be very important to avoid the problems with manually changing the files needed by NCTUns.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692

The solution presented in this work provides a new approach to network emulation, expanding the functionality and applicability of the open-source NCTUns network simulator/emulator, and creates the basis for emulate advanced systems.

6. ACKNOWLEDGMENTS

Our thanks to NCTUns development and support team who answered our questions and issues in the many forms.

7. REFERENCES

[1] DoD Dictionary of Military and Associated Terms/NATO Only Terms; http://www.dtic.mil/doctrine/jel/doddict/natoterm/c/00305.html.

[2] "Tactical Common Data Link [TCDL]"; http://www.globalsecurity.org/intell/systems/tcdl.htm

[3] Broady, A.; "Optimization and Extension of an Infrastructure Supporting Global Software Engineering Teams"; Thesis for MSc Software Engineering, August 2007.

[4] Wang, S. and Chou, C.; "Innovative Network Emulations Using The NCTUns Tool"; Computer Networking and Networks (Shannon, S., ed.), Nova Science Publishers, chapter 7, pp. 159–189 (2006). http://nsl10.csie.nctu.edu.tw

[5] Solomon, J.D.; "Mobile IP The Internet Unplugged"; Prentice Hall Series in Computer Networking and Distributed Systems, 1998.

[6] Wang, S.Y.; Chou, C.L.; Lin, C.C. and Huang, C.H.; "The Protocol Developer Manual for the NCTUns 5.0 Network Simulator and Emulator".

[7] Bennett, B., Dee, C. and Ngugen, M.H.; "Operational concepts of MPEG-4 H.264 for tactical DoD applications"; MILCOM 2005, October 17-20 Atlantic City, NJ, Unclassified Proceedings.

[8] Kellerer, W., Steinbach, E., Eisert, P. and Girod, B.; "A real-time Internet streaming media testbed"; Proc. Of International Conference on Multimedia and Expo, ICME 2002, August 2002.

[9] Ihara, A., Murase, S. and Goto, K.; "IPv4/v6 Network Emulator using Divert Socket"; Proc. of 18th International Conference on Systems Engineering (ICSE2006), Coventry, UK, pp. 159–166 (Sep. 2006).

[10] Bladine, I.; "Divert Sockets mini-HOWTO:" http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Divert-Sockets-mini-HOWTO.pdf

[11] VideoLAN Wiki; http://wiki.videolan.org/VLC

[12] Blake, S; Black, D. ; Carlson, M; Davies, E; Wang Z. and Weiss W.; "An Architecture for Differentiated Services"; IETF RFC 2475, December 1998, http://www.ietf.org/rfc/rfc2475.txt.

[13] Kumar, Vijay P.; Lakshman, T. V. and Stiliadis, Dimitrios; "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet"; IEEE Communications Magazine, May 1998.

[14] Perkins, C.; "IP Mobility Support for IPv4"; IETF RFC 3344, August 2002, http://www.ietf.org/rfc/rfc3344.txt.

[15] Haas, Erik; "Aeronautical Channel Modeling"; IEEE Transactions on Vehicular Technology, v. 51, n. 2, March 2002.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5692
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5692