Expertise
and insight
for the future

Antti-Juhani Takamaa

Hardware-based WAN emulator replaced with software-based one

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communications Technology

Bachelor's Thesis

15 November 2020

Author Title Number of Pages Date	Antti-Juhani Takamaa Hardware-based WAN emulator replaced with software-based one 29 pages + 4 appendices 15 November 2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	IoT and Cloud Computing
Instructors	Marko Uusitalo, Senior Lecturer and Head of Degree Programme

The aim of this thesis was to find a more cost-efficient solution for Secure Land Communications, subsidiary of Airbus Defence and Space, to emulate a wide area network in their testing processes.

The old solution is a hardware-based network device that emulates network impairments in a wide area network. Scaling the emulation capability up with the old solution is expensive. The new researched solution is a software-based emulator, an application that emulates network impairments in a wide area network.

The new emulation solution needed to be tested against the old one. The phases of the testing process were planning, execution and examination of results. Planning started with discussion with stakeholders, the project requirements and project timing was agreed upon. Next, the test environment and the test case that would be ran with all the emulators, was created. Software-based WAN emulator options were researched and selected. The created testing plan was executed. Results following from the tests were examined. Conclusion were drawn from the results.

The aim of this thesis was reached. SoftPerfect Connection Emulator is a WAN emulation application, which's performance closely matches the performance of Apposite Technologies' Netropy N91 WAN emulation device. The cost of the new solution is approximately 7,5% of that of the old solution.

Keywords	Network, WAN, Emulation, SoftPerfect Connection Emulator, Apposite Technologies Netropy N91

Tekijä Otsikko Sivumäärä Aika	Antti-Juhani Takamaa WAN-emulaatio laitteen korvaaminen ohjelmalla 29 sivua + 4 liitettä 15.11.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	verkot ja pilvipalvelut
Ohjaajat	Lehtori Marko Uusitalo

Tämän insinöörityön tavoite oli löytää kustannustehokkaampi ratkaisu Secure Land Communications:lle, Airbus Defence and Spacen tytäryhtiölle, emuloida laaja-alaista verkkoa heidän testausprosesseissaan.

Vanha ratkaisu oli laitteistopohjainen verkkolaite, joka emuloi verkon heikentymistä laaja-alaisessa verkossa. Emulaatiokyvyn skaalaaminen suuremmaksi vanhalla ratkaisulla on kallista. Uusi tutkittu ratkaisu on ohjelmistopohjainen emulaattorisovellus, sovellus, joka emuloi verkon heikentymistä laaja-alaisessa verkossa.

Uutta emulation ratkaisua piti testata vanhaa vastaan. Testausprosessin vaiheet olivat suunnittelu, toteutus ja tulosten tarkastelu. Suunnittelu alkoi keskustelulla sidosryhmien kanssa ja projektin vaatimuksista sekä ajoituksesta sovittiin. Seuraavaksi testiympäristö ja testiskenaario, joka suoritettaisiin jokaisella emulaattorilla, luotiin. Ohjelmistopohjaiset laajan alan verkko emulaattorivaihtoehdot kartoitettiin ja soveltuvimmat valittiin jatkoon. Luotu testaussuunnitelma toteutettiin. Testeistä saadut tulokset tutkittiin. Päätelmät johdettiin tuloksista.

Tämän insinöörityön tavoite saavutettiin. SoftPerfect Connection Emulator on laajan alan verkkoemulaatiosovellus, jonka suorituskyky läheltä vastaa Apposite Technologies Netropy N91 -laajan alan verkkoemulaatiolaitetta. Uusi ratkaisu maksaa noin 7,5 % vanhan ratkaisun hinnasta.

Avainsanat	Verkko, laaja-alainen verkko, Emulaatio, SoftPerfect Connection Emulator, Apposite Technologies Netropy N91

Contents

List of Abbreviations

CAT6a	Category 6A. Standardized twisted pair cable for Ethernet and other physical layers.
CLI	Command-Line Interface. Processes commands to a computer program in the form of text.
CMD	Command Prompt. Default command-line interface in Microsoft Windows operating systems.
CPU	Central Processing Unit. Electronic circuitry within a computer that executes instructions to make up a computer program.
GNSS	Global Navigation Satellite System. System that uses satellites to provide autonomous geo-spatial positioning.
GPS	Global Positioning System. Satellite-based radionavigation system owned by the United States government.
GUI	Graphical User Interface. Form of user interface that allows user to interact with electronic devices through graphical icons.
HPE	Hewlett Packard Enterprise. American multinational enterprise information technology company.
IP	Internet Protocol. Principal communications protocol in the Internet protocol suite.
IPv4	Internet Protocol version 4. Fourth version of the internet protocol.
IPv6	Internet Protocol version 6. Sixth version of the internet protocol.

MAC	Media Access Control. Unique identifier assigned to a network interface controller.
MPLS	Multiprotocol Label Switching. Routing technique in telecommunications networks.
NIC	Network Interface Controller. Computer hardware component that connects a computer to a computer network.
NTP	Network Time Protocol. Networking protocol for clock synchronization between computer systems.
OS	Operating System. System software that manages computer hardware, software resources, and provides common services for computer programs.
PC	Personal Computer. Multi-purpose computer whose size, capabilities, and price make it feasible for individual use.
PCIe	Peripheral Component Interconnect Express. High-speed serial computer expansion bus standard.
RAM	Random Access Memory. Form of computer memory that can be read and changed in any order.
RFC	Request for Comments. Publication from the Internet Society and its associated bodies, most prominently the Internet Engineering Task Force.
RTCP	RTP Control Protocol. Sister protocol of the Real-time Transport Protocol.
RTP	Real-time Transport Protocol. Network protocol for delivering audio and video over Internet Protocol networks.

SCE	SoftPerfect Connection Emulator. Wide area network environment emulator application.
SLC	Secure Land Communications. European public safety communication systems company.
TCP	Transmission Control Protocol. One of the main protocols of the internet protocol suite.
ToS	Type of Service. A field located in the second byte of the internet protocol version 4 header.
UDP	User Datagram Protocol. One of the main protocols of the internet protocol suite.
UPS	Uninterruptible Power Supply. Electrical apparatus that provides emergency power to a load
USB	Universal Serial Bus. Industry standard that establishes specifications for cables and connectors and protocols for connection.
VLAN	Virtual Local Area Network. Any broadcast domain that is partitioned and isolated in a computer network at the data link layer.
WAN	Wide Area Network. Telecommunications network that extends over a large geographic area.
WANem	Wide Area Network emulator. Wide area network environment emulator software.

1 Introduction

Testing is a necessary part of software development. With network-enabled applications, especially time-critical ones, it is crucial to test the product in suboptimal environments. Secure Land Communications develops mission-critical communication products, where the reliable operation of the whole communications system is key.

Secure Land Communications (SLC), subsidiary of Airbus Defence and Space, needs to test their products' reliability and performance in order to guarantee them to the customers. There are many tests for hardware and software. One of the tests for software is resilience in a network with poor connectivity.

When two machines are connected via Wide Area Network (WAN), some errors in the network traffic are expected. For SLC to test their products performance in a WAN environment, a WAN emulator is needed. Currently SLC is using mainly hardware-based WAN emulators.

One of the emulator models in use at SLC is Apposite Technologies Netropy N91 Network Emulator, later referred to as N91. The N91 supports the emulation of four links. Problem arises when testing needs to be scaled up and multiples of tens of WAN links need to be emulated.

The cost of a N91 unit and a license is high, so simply buying more of them is not the most cost-efficient solution. That is why it was decided to look for a more cost-efficient alternative, a software-based WAN emulator.

The goal of this project is to find a WAN emulator which is more cost-efficient than the N91.

2 Netropy N91

Apposite Technologies Netropy N91 is a high-precision appliance used for network emulation [1]. N91 appliance pictured below.

Figure 1. Netropy N91

The N91 can be used to emulate network impairments and view the impairment results in real time. It is configurable via browser-based Graphical User Interface (GUI), which is convenient as it is a rack mounted appliance. The N91 also offers a comprehensive Command Line Interface (CLI) for automated testing purposes. [1]

As stated earlier, the N91 has four separate emulation engines, each capable of 1 Gbps throughput. Each of the four links can be set to operate at a certain bandwidth from 100 bps all the way to the maximum 1 Gbps. The supported network impairments are delay, packet loss, packet corruption, packet reordering and packet duplication. [1]

3 Network Impairments

Different network error types, or so-called network impairments, will be addressed frequently later in this study. It is important for one to understand what the different errors mean exactly.

Packet delay is the most common effect witnessed in network traffic. It is the amount of time that elapses between the time a packet is transmitted and the time it is received [2]. Delay can never be eliminated, as the speed of light in a vacuum is the fastest a packet could theoretically propagate [2]. Packet delay only becomes a problem when it is constantly very high, but even then, it can not technically be classified as network error. If the delay of a connection is always high, but constant, delay is just seen as a character of the connection, not an error in the connection. For example, a satellite link connection has high delay, but it is just a natural character of a satellite connection, not an error.

Packet jitter is the measure of packet delay variation [2]. In other words, when the delay constantly changes from low to high, it is called jitter. Jitter is an important factor in a network connection as it affects how long software can expect to wait for data to arrive [2]. Significant changes in delay most likely cause errors in software functionality.

Packet loss is the disappearance of a packet that was transmitted [2]. There are ways to combat packet loss, but it always includes re-transmitting the packet that was originally lost, a lost packet can not be recovered.

Packet corruption occurs when the contents of a packet are damaged, but the packet continues to flow towards its destination [2]. Packet corruption mitigation works the same way as packet loss mitigation. From the receiver's point-of-view, a corrupted packet is the same as a lost packet, it needs to be received again.

Packet duplication occurs when one packet becomes two or more identical packets [2]. Packet duplication does not happen by itself, rather it is usually caused by the transmitter or a network device along the way to the receiver.

Packet reordering occurs when packets are received in different order than they were transmitted in [2]. Packet reordering is usually caused by the same factors as packet duplication.

4 Planning

4.1 Timing

The timing of the project was planned with the stakeholders interested in the project. This included the Laboratory team, the radio connectivity server team and the artificial intelligence laboratory team.

It was decided that the timeframe for the whole project is ready as soon as possible, because it possibly removes a major bottleneck from the testing process. The only hard deadline given was regarding the usage of the N91. The device would be available for limited testing until the end of August 2020.

4.2 Test Planning

First a benchmark needs to be taken with the N91 and after that the software test results are compared to the benchmark. The test for the software needs to reflect the current setup with the N91. This consists of physical devices and their connections, as well as the load moving in the system.

4.2.1 Environment Planning

The N91 is connected between two devices. From the two devices perspective, they have a point-to-point connection. There can be no network devices such as switches or routers in the system, as they could affect the emulation results by creating inconsistencies.

A movable testbench would need to be created in order to get two end-devices near the N91. This is because the N91 can not be moved, and there can be no network devices in the test system as previously mentioned. The end devices will be connected straight to the WAN emulation device with Category 6A (CAT6a) Ethernet cables.

The end devices will need to be able to run traffic generation as well as traffic capturing software. Two Linux Personal Computers (PC) running CentOS 7 will suffice for this purpose, as they can be also installed to the movable testbench.

Clock synchronization between the two end-devices will have to be considered, when running one-way jitter scenarios later discussed in chapter 4.2.2. One-way delay and jitter are measured by transmitting a precisely timestamped packet to the receiver and comparing the timestamp to the receiver's reference clock - the difference is the one-way delay and the variation in delay is jitter [3]. Differences in the reference clock times significantly affect the result of this comparison. The end-devices' reference clocks will be synchronized with a Network Time Protocol (NTP) time server.

The NTP time server model in use is Meinberg IMS – LANTIME M1000. It provides a stratum 1 level time base for a network. The device has a Global Navigation Satellite System (GNSS) receiver using Global Positioning System (GPS), that provides the device itself with a stratum 0 level time base. [4]. The offset of the stratum 0 level time was -2 µs.

4.2.2 Test Case Planning

In the test case traffic flows from one end-device to the other via a WAN emulator. The WAN emulator causes errors to the traffic. These errors will be detected and captured at the receiving end-device.

Traffic generation and receiving will be done with iPerf3 running on both end-devices. Packet capturing will be done on the receiving end-device with Wireshark. Error detection will be done on the receiving end-device with iPerf3 and Wireshark.

Traffic errors to be generated were discussed with the parties using the N91. Currently the WAN emulator is used to mainly emulate periodic packet loss. More accurately four packets lost on a period of 100 packets, resulting in a packet loss of 4%.

Other error generation scenarios were discussed to be used in the future and used for the comparison of the WAN emulators. The chosen error generation scenarios can be found in the table below:

Scenario	**Parameters**
Packet Loss	4% packets lost
Packet Reordering	5% packets reordered, 10 packets spacing
Packet Duplication	5% packets duplicated
Packet Jitter	Delay of 250ms, fluctuation of 250ms (0-500ms delay)

Table 1. Error generation types and parameters

The test case duration will be 30 seconds.

4.3 Software

Software to emulate WAN traffic was searched from the internet, using keyword *WAN emulation software*. Five different software were chosen for closer review.

4.3.1 WANem

Wide Area Network emulator, or WANem for short, is built on top of Knoppix Linux [5]. It does not need to be installed, it boots from a flash drive and runs as Live Linux. Once running, it offers a web GUI with a full control panel. WANem is an open-source software [5].

WANem supports packet filtering based on IPv4 source & destination address. The following network impairments are supported by WANem: delay, loss, duplication, reordering and corruption. [6]

4.3.2 SoftPerfect Connection Emulator

SoftPerfect Connection Emulator (SCE) is a WAN environment emulator [7]. It offers a free trial version of the software, limited to 30 seconds duration per emulation [7]. This is enough for as long as we just compare its performance with other software and the N91.

SCE allows to apply actions to specific streams of packets using filters. Packets can be filtered using Internet Protocol (IP) source & destination address range (IPv4 or IPv6), Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port number or Media Access Control (MAC) address. SCE can emulate the following network impairments: delay, loss, duplication, reordering and corruption. SCE runs on any PC with Windows 7 or higher version operating system. [8]

4.3.3 Gambit Communications Mimic NetFlow Simulator

Gambit Communications Mimic NetFlow Simulator was available for downloading after contacting Gambit Communications representative for download link.

Upon installation it was however discovered that the software was indeed just a network traffic simulator, it did not have any emulation capabilities as marketed which made it unfit for our use case.

4.3.4 WAN-Bridge

WAN-Bridge was installed to a test machine. It did have the network traffic emulation features we were looking for.

WAN-Bridge was also discontinued in 2010, so there is no support for it anymore. For these reasons, we chose to not investigate the software any further.

4.3.5 iTrinegy NE-ONE Flex

After more in-depth review of iTrinegy NE-ONE Flex network emulator, it was discovered that it only supports network traffic emulation between virtual machines. This made it unfit for our use case.

4.4 Hardware

Hardware specifications were needed for two different type of machines, the end-devices and the emulator. The end-devices would not need much performance, they only need to be able to run Linux and have ethernet port. Already available machines were used as end-devices.

Neither of the chosen WAN emulation software have any minimum hardware requirements, but the emulator machine was still configured to have enough performance to not cause a bottleneck in the WAN traffic emulation. It will also need more than one ethernet port, so an additional Peripheral Component Interconnect Express (PCIe) Network Interface Controller (NIC) was used. Such a machine was not available, so one was built.

The machines had the following specifications:

	Emulator	**End-devices**
CPU	Intel Xeon E5-1620 @ 3,6GHz	Intel Core i5-8250U @ 1,6GHz
RAM	16GB	16GB
NIC	Broadcom NetXtreme Gigabit Ethernet BCM5719-4P, 4-slot	Intel Ethernet Connection I219-V

Table 2. Test machine specifications

The before mentioned emulator machine will only be used for the testing phase. A different configuration would be chosen in the future, if tests succeed.

5 Execution

5.1 Software Installation

WANem .iso file was acquired and burned to a Universal Serial Bus (USB) flash drive. Functionality of the created media was tested by booting into the WANem Operating System (OS). SCE installer was acquired and installed on the emulator machine running Windows 10 LTS C 64-bit Build 17763. Drivers for the PCIe NIC were updated.

Wireshark and iPerf3 were installed to the two Linux PC end devices, both running CentOS 7 3.10.0-1127.10.1.el7.x86_64. The PCs had a point-to-point connection to each other, so they were configured with IP-addresses using the /31 mask.

5.2 Test Setup

Test setup was built according to the plan in chapter 4.2.

5.2.1 Environment Setup

Reference clocks on the two end-devices were synchronized with an NTP time server. The devices were connected to a network where they had connection to the NTP time server, then synchronization with the NTP stratum 1 source was invoked and finally the devices were removed from the network.

The two end device PCs were installed to a movable test bench. This way the devices could be easily moved into a cramped server room close to the N91 and moved out of the way if needed. The PCs were connected to specific power sources in the server room, not to disrupt load on the Uninterruptible Power Supply (UPS).

N91 emulation link usage was discussed and confirmed with relevant parties, and the link available for testing was identified. Engine 1 was used for testing, but engines 2-4 were not to be touched. The end devices were connected to the N91.

5.2.2 Test Case Setup

iPerf3 server was started at the receiving end-device, using the command:

```
iperf3 -s
```

In the above command the option -s starts an iPerf3 server on the machine with default settings. By default, the server is listening on TCP port 5201, waiting for connections from iPerf3 clients [9].

iPerf3 client-side command was prepared to be executed on the transmitting end-device:

```
iperf3 -c 10.10.10.2 -u -t 30 --get-server-output --logfile <Em-
ulator_Scenario>
```

The above command starts an iPerf3 client executable. The options are explained in the table below:

Option	**Explanation**
-c 10.10.10.2	Run iPerf3 in client mode connecting to a specific iPerf3 server
-u	Transmit using UDP protocol
-t 30	Time in seconds to transmit for
--get-server-output	Get the output from the server
--logfile <Emulator_Scenario>	Send output to a log file

Table 3. iPerf3 client-side options and explanations [9]

TCP protocol performs error checking and correction, whereas UDP does not [10]. It was critical to distinguish the errors generated by the WAN emulator, that is why UDP was used for transmission rather than TCP. In order to get the maximum amount of available data for review, both data from the client side, and data from the server side was needed. The results were finally recorded to a logfile, named as per the WAN emulator used and the test scenario performed, for example '*N91_Loss'*. Because of our requirements, in the tests the throughput report interval of iPerf3 was kept at the default value of one second. The throughput report is simply an interim result of the traffic generation, that in our case, is displayed on the iPerf3 server.

5.3 Test Execution

First a baseline was taken with the N91, then benchmarks run with WANem and SCE.

5.3.1 Baseline

Engine 1 needed to be configured on the N91 according to the planned emulation scenarios before each baseline test was taken. Shown in the figure below is the main view on the N91 web interface.

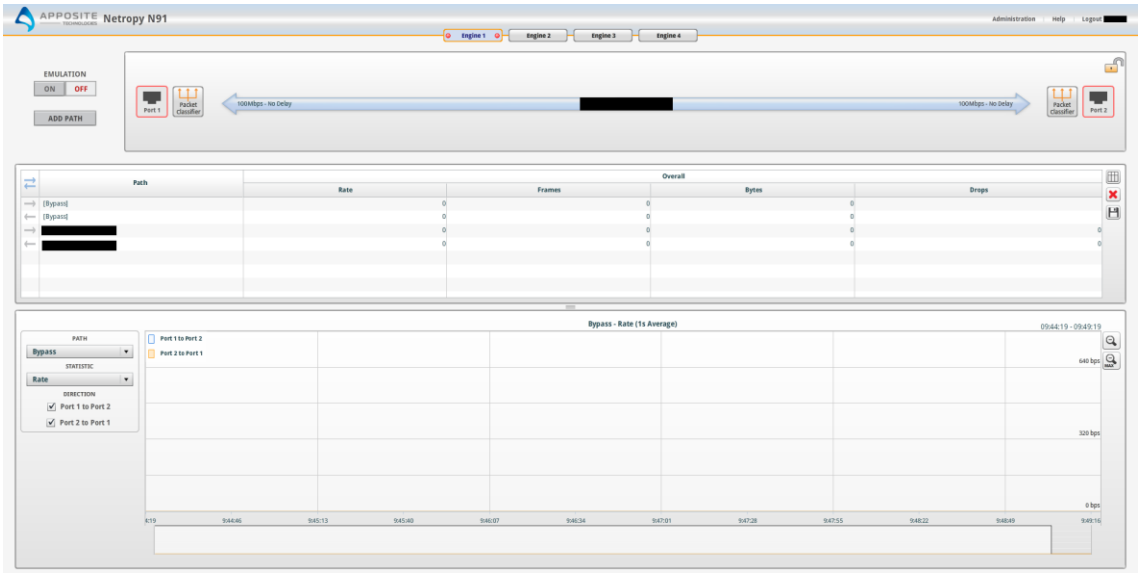

Figure 2. Netropy N91 web interface main view.

Engine 1 was selected from the main view, and the packet classification was first configured. With packet classification, the user can choose which packets go through the emulation engine and which packets are just forwarded straight through. Shown in the figure below is the packet classification view on the N91 web interface.

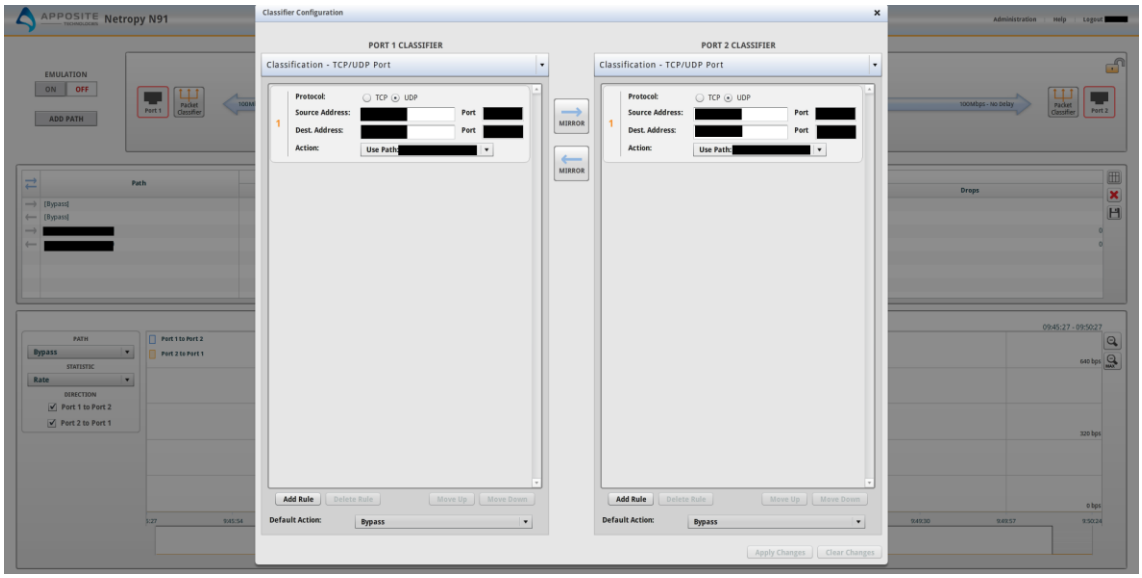

Figure 3. Netropy N91 web interface packet classification view.

Packets can be classified by IP source & destination address range (IPv4 or IPv6), Virtual Local Area Network (VLAN), TCP or UDP port number, IP ToS, MAC address, MPLS label, or any other packet contents [1]. For our use case none of the traffic needed to bypass the emulation, so the packet classification was turned off and all the traffic configured to use our test case path.

Before each test was ran, the emulation parameters were configured according to the plan. In the N91, the emulation is called a path, through which the traffic is directed. Shown in the figure below is the path configuration view on the N91 web interface.

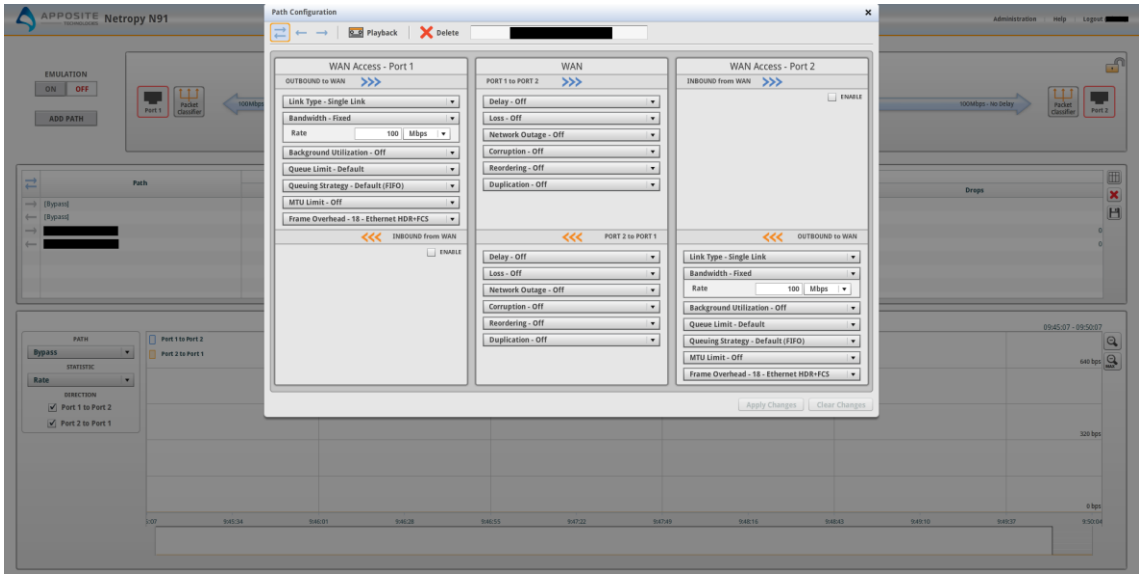

Figure 4. Netropy N91 web interface path configuration view.

The configuration window is divided in to three different columns: Port 1, WAN and Port 2. Each of these columns are split into two sections, inbound and outbound traffic. In our use case, we were only interested in the WAN column, the Port 1 & 2 columns were left untouched. In the WAN column both sections were always mirrored to match each other, so that the WAN emulation of the traffic would occur both ways. The parameters needed were Delay, Loss, Reordering and Duplication.

If the operator of the device wants to set a certain bandwidth limit, it can be accomplished in the Port 1 & 2 columns, using the Bandwidth parameter. The minimum bandwidth is 100 bps and this value can be increased in 1 bps increments up to the maximum bandwidth supported by the ports of the device or by the license key.

5.3.2 Benchmarks

WANem was chosen to be tested first. The emulator PC was booted to the WANem Live Linux OS with the media created earlier. However, before configuring the emulation parameters, bridging needed to be setup between the two end-devices. From the WANem web interface, a remote console session was established, and the bridge was configured. Shown in the figure below is the remote console view on the WANem web interface. See figure 5 below.

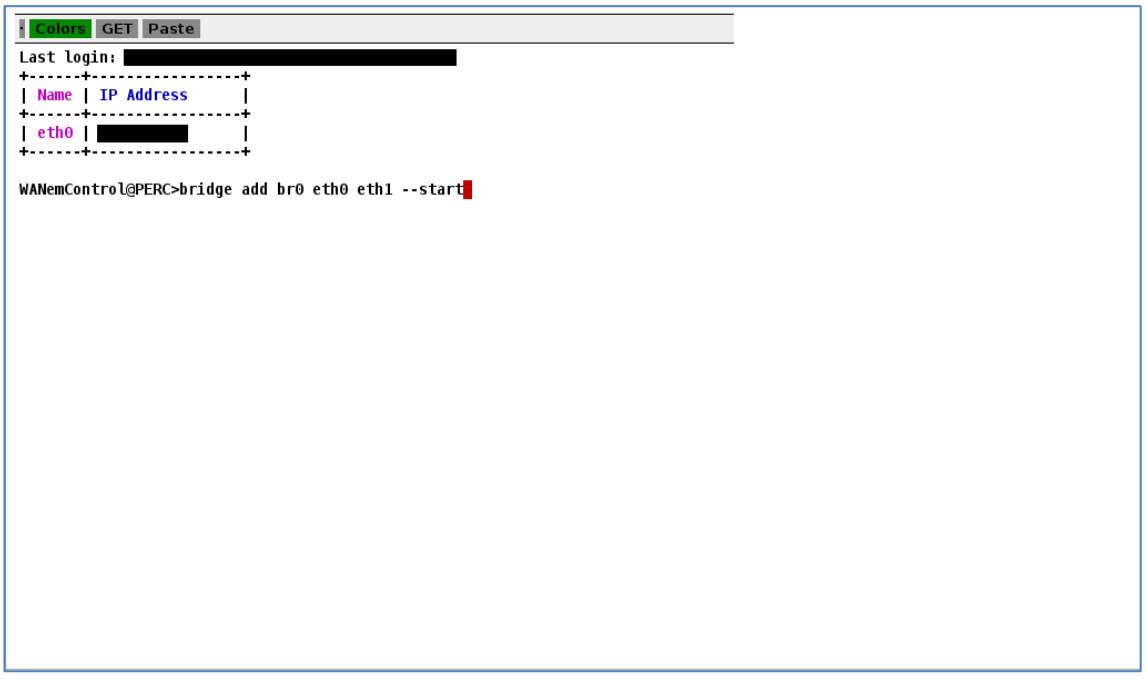

Figure 5. WANem web interface remote console view

The command given is not WANem specific, but rather an old universal Linux command:

```
bridge add br0 eth0 eth1 --start
```

Four different things were configured with the previous command:

1. Bridge with name br0 is created
2. Interface eth0 is added to the bridge br0
3. Interface eth1 is added to the bridge br0
4. Bridge interface br0 is started

Next emulation parameters were configured according to the planned test scenarios. From the WANem web interface advanced mode was chosen, and parameters set for both interfaces in configured in the bridge previously. Shown in the figure below is the advanced mode view on the WANem web interface.

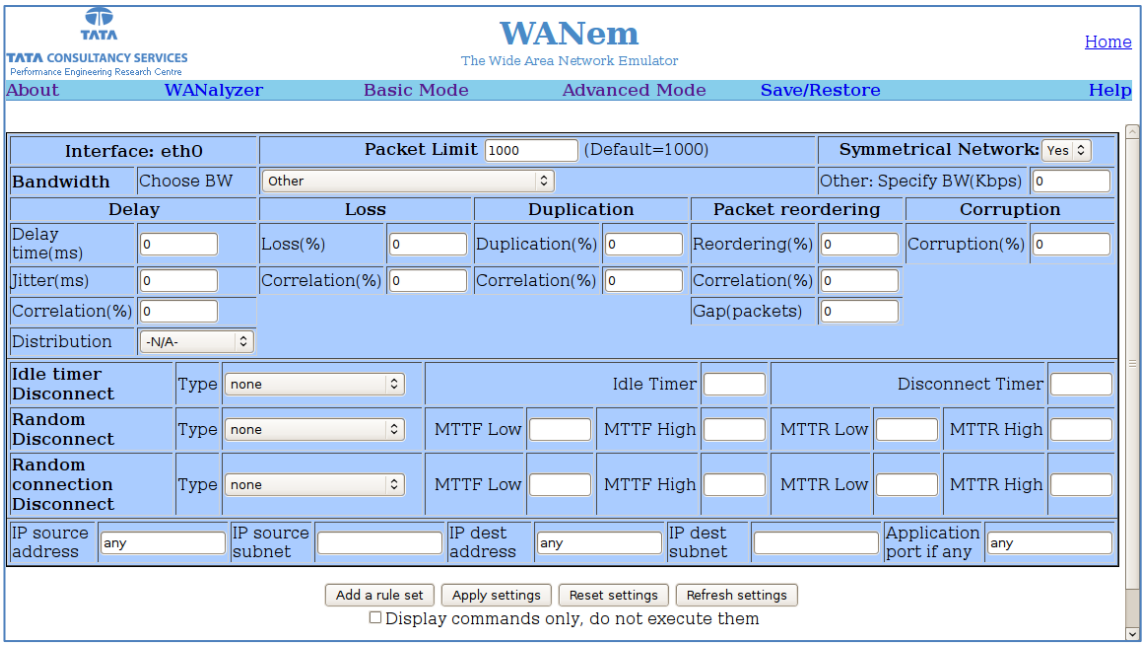

Figure 6. WANem web interface advanced mode view

In the configuration window all the parameters are arranged into a table. For our testing the only relevant sections of the table were Delay, Loss, Duplication and Packet reordering.

If a certain bandwidth would need to be emulated, it can be done by setting a value in the Bandwidth section of the table. The bandwidth drop-down menu offers multiple ready pre-sets, that reflect certain standard connection speeds. Many of these pre-sets however are old, so often a custom bandwidth limit needs to be set. The custom bandwidth limit that can be emulated accurately, can range from 120 kbps up to half of the speed supported by the ports on the device.

SCE was the final WAN emulator to be tested. The emulator PC was booted to the installed Windows 10 OS and SCE application was started. Shown in the figure below is the main view of SCE.

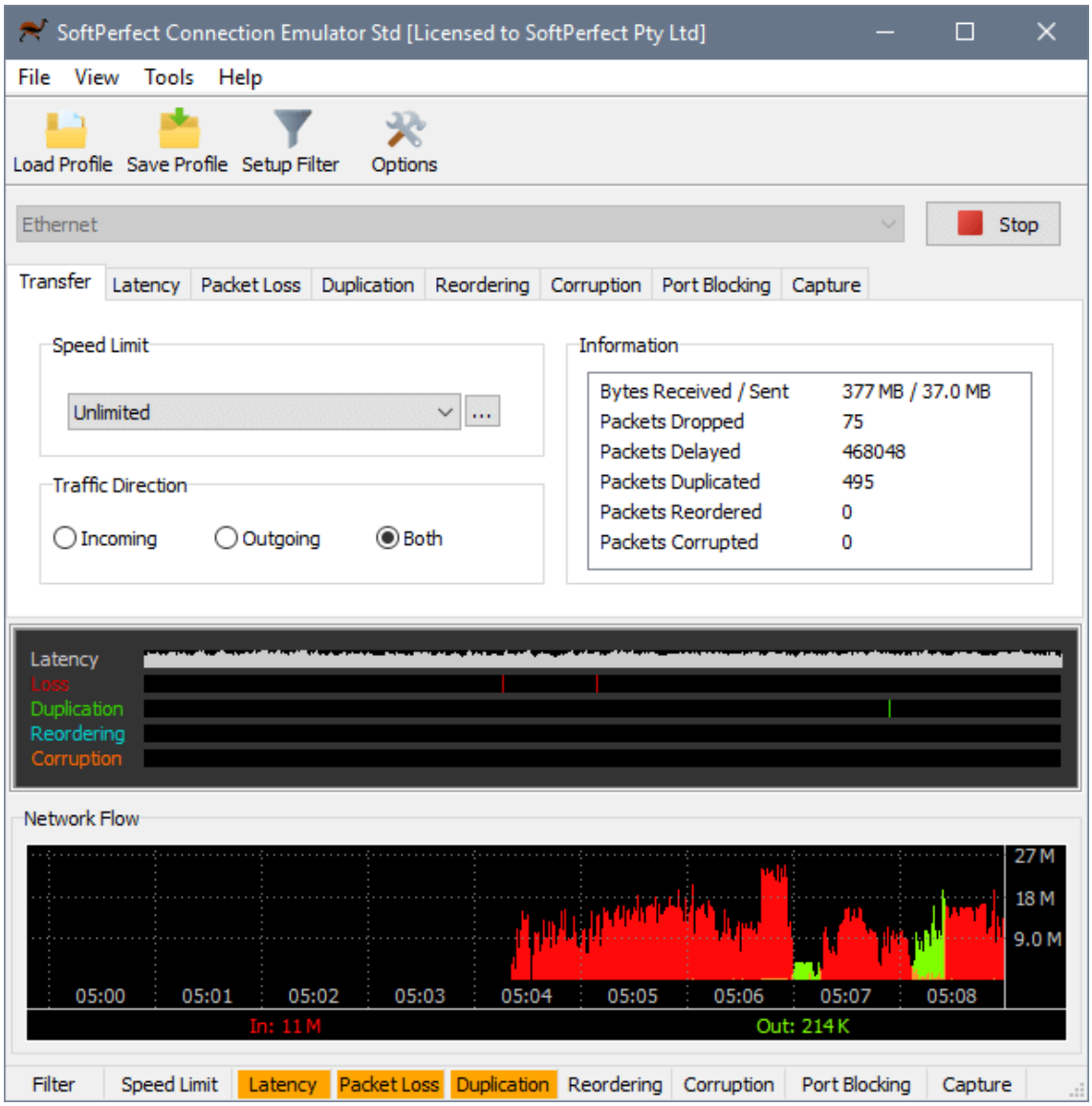

Figure 7. SCE application main view

In SCE as well, bridging needed to be configured before the emulation parameters could be set and testing started. From the main view **Tools - Bridging** was selected. Shown in the picture below is the bridge tool of SCE.

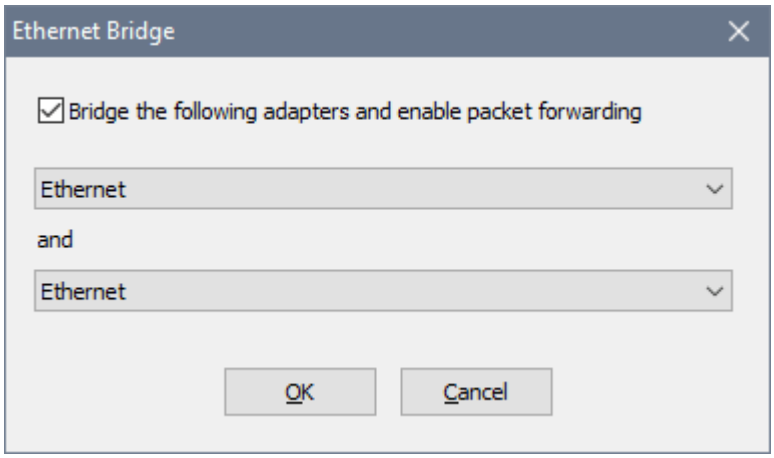

Figure 8. SCE application bridge tool

SCEs bridge tool was easy to use. Bridging could be set either on or off and the two network interfaces to be bridged were chosen from a drop-down menu of available interfaces.

Emulation parameters were set from the SCE main view (figure 7). In the main view below the toolbar there is a drop-down menu to choose a specific network interface from. After the interface is selected, the emulation parameters can be set from the different tabs. For our testing we were interested in the Latency, Packet Loss, Duplication and Reordering tabs. Illustrated in the pictures below are the Packet Loss and Latency tabs.

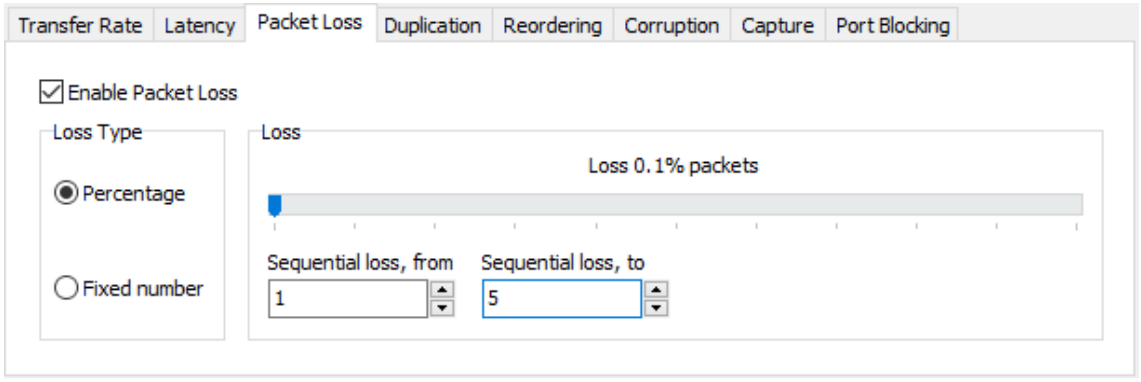

Figure 9. SCE packet loss tab

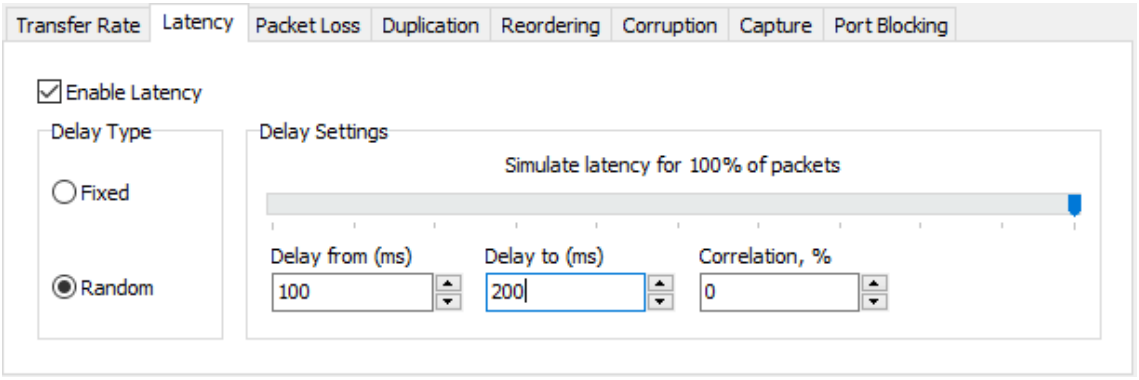

Figure 10. SCE latency tab

If needed, a certain bandwidth could be emulated with SCE. From the Transfer tab, the Speed Limit menu offers a wide variety of bandwidth limit pre-sets based on commonly used bandwidths. A custom bandwidth limit can also be set from the Speed Limit menu. Note that in case bandwidth limit is used, the Both-option should be chosen from the Traffic Direction menu. This way the bandwidth limit resembles common real-world situations.

6 Results

Main point in the comparison of results was the emulators ability to emulate packet loss. Secondary comparison points were jitter, duplication and reordering abilities.

6.1 Packet Loss

The baseline results from the N91 were as expected, it could emulate a steady packet loss according to the given parameters. As one can see in Appendix 1, the average packet loss emulated at the end of the test was 3,9%. The bottom half of the appendix is the iPerf3 server output. The server output has a list of lines, one for every one second interval of the test. At the end of each of these lines, is the lost percentage for that specific interval. Below the list is one final line, displaying a summary of all the intervals. The average lost percentage for the whole test can be found at the end of this summary line. The given packet loss target was 4,0%, making the result a very close match. More importantly, the N91 could keep the packet loss rate steady during the whole test. Every ten second interval started with packet loss of 0%, but for the rest of the interval the loss rate was 4,4%, averaging out to the previously mentioned 3,9%.

Emulation results with WANem were not satisfying. In Appendix 2, one can see that the average packet loss emulated at the end of the test was 3,8%. The bottom half of the appendix is the iPerf3 server output. The server output has a list of lines, one for every one second interval of the test. At the end of each of these lines, is the lost percentage for that specific interval. Below the list is one final line, displaying a summary of all the intervals. The average lost percentage for the whole test can be found at the end of this summary line. The given packet loss target was 4,0%. Even though this was a close match, the fluctuation of the packet loss was unacceptably high. In this test the loss percentage ranged from 0% to 7,8%. In some other tests a packet loss of 10.0% was recorded, with the packet loss parameter being set to 4,0%. Because of our requirements, in our testing the emulated packet loss needs to be steady. Deviation of two and a half times from the given parameter value is too high, be it even momentarily.

SCE emulation results were satisfying. Appendix 3 shows, that the average packet loss emulated was 4,2%. The bottom half of the appendix is the iPerf3 server output. The server output has a list of lines, one for every one second interval of the test. At the end of each of these lines, is the lost percentage for that specific interval. Below the list is one final line, displaying a summary of all the intervals. The average lost percentage for the whole test can be found at the end of this summary line. The given packet loss target was 4,0%. This means that the packet loss emulation was close enough to the target. Important thing to notice is that the packet loss percentage mostly remained within 0,5% of the target 4,0%. Even though this is less steady than the N91, it still steady enough emulation for our purposes.

In conclusion the N91 gave us a reliable packet loss emulation baseline. SCE reached results good enough compared to the baseline, WANem did not. The remaining tests were still performed with WANem to gain reference, even though WANem would not be selected for our final use.

6.2 Jitter, Duplication and Reordering

Jitter emulation was measured by inspecting the total packets received per interval section in the server section of iPerf3 output file. The target packets sent per interval depends on bandwidth of the connection, in our case iPer3 sent 90-91 packets per interval. When the connection has a fluctuating delay, the packets received per interval start to deviate from the default 90-91 packets.

When the tests were run with the N91, iPerf3 output showed a clear deviation in the packets received per interval. We wanted SCE to show similar deviation in the packets received per interval. SCE did also show a clear deviation in the packets received per interval section. The deviation was not as large in the case of SCE when compared to the N91. In the future the delay correlation parameter would need to be increased for SCE to gain similar results compared to the N91. The chart below illustrates the effect of latency correlation on random delay between 100ms and 200ms in SCE.

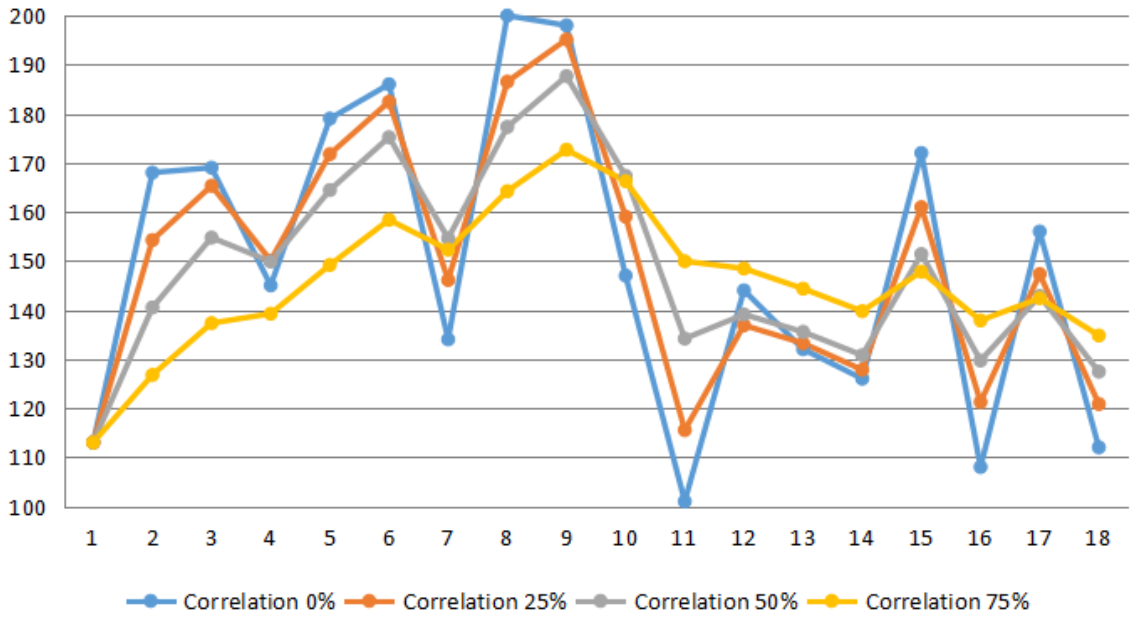

Figure 11. Latency correlation in SCE [8]

Duplication and reordering emulation were measured by inspecting the summary line in the server section of the iPerf3 output files. See Appendix 4. In the summary line, reordered packets are logically seen as *datagrams received out-of-order*. Duplicated packets however are also seen as datagrams received out of order. This is because whenever there is a packet duplication, for example when packet number 66 is duplicated, it causes iPerf3 to perceive that the packet number 67 is out of order.

N91 and SCE performed similarly when emulating packet reordering and duplication. In these tests the performance of the two emulators was a close match. The emulation results also were an extremely close match to the target 5% parameter in both scenarios for both emulators – The target result was 112 packets received out-of-order.

In conclusion SCE performed very similarly compared to the N91, when emulating jitter, duplication and reordering.

6.3 WANem Instability

Explaining the WANem packet loss inaccuracies and trying to resolve the cause(s) is not a target of this thesis. However, doing so has been deemed to provide valuable information for parties interested in this project.

The packet loss emulated by WANem was discovered to be irregular. When running the tests, packet loss target was set to 4%. The loss percentage ranged from 0% to 7,8%. In the charts below, is the I/O graph derived from the Wireshark captures taken during the packet loss emulation tests for the N91 and WANem respectively.

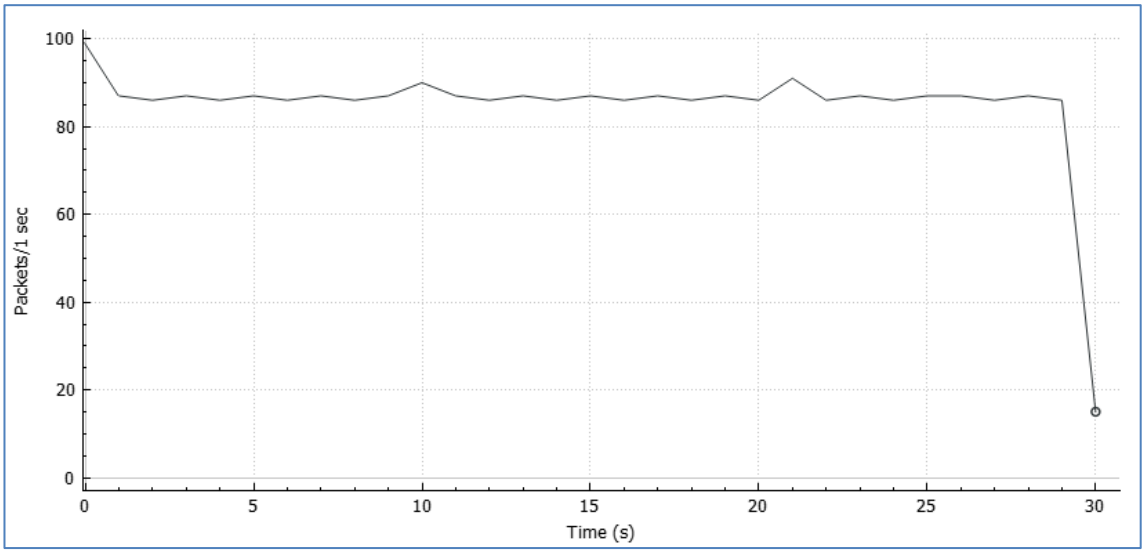

Figure 12. N91 packet loss emulation test Wireshark I/O graph

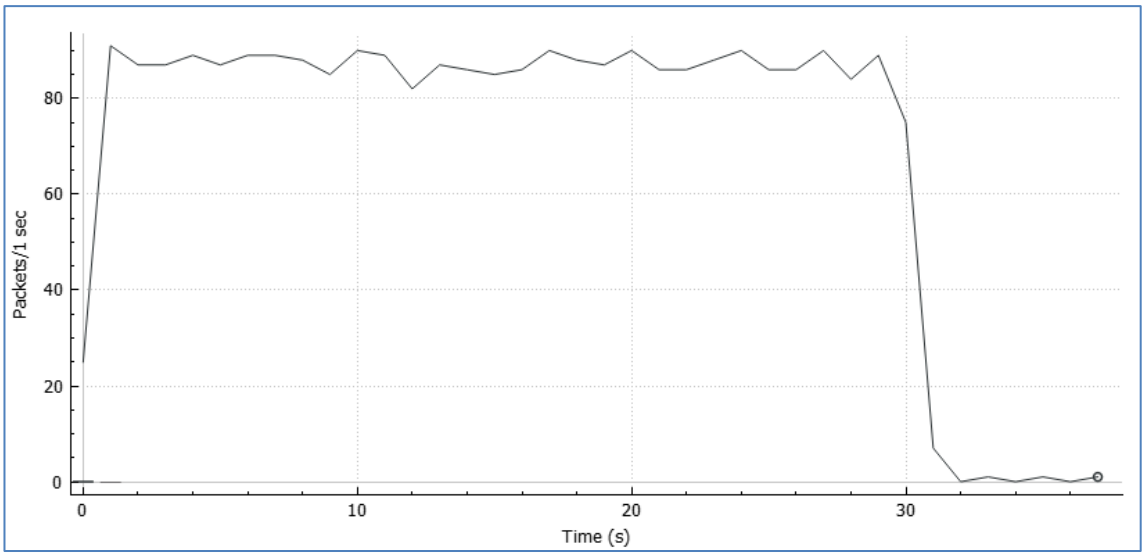

Figure 13. WANem packet loss emulation test Wireshark I/O graph

The previous charts further illustrate the irregularities in the packet loss emulation when WANem was used. Time in seconds is shown in the x-axis and packets sent per second is shown in the y-axis. The contour of the line is much more aggressive in the WANem chart than it is in the N91 chart.

The effect of increasing the duration of the emulation period was tested with WANem. The packet loss emulation test duration was increased from 30 seconds to 300 seconds. The test was started and the output on the iPerf3 server machine was observed. It seemed that now that the test period was longer, there was a higher change to witness anomalies in the packet loss emulation percentage. Three of these 300 second tests were performed, and the highest packet loss percentage per interval observed was 10.0%, as stated earlier in chapter 6.1. It is possible, that with even higher emulation periods, an even higher deviation could be witnessed.

The high fluctuation in the emulated packet loss percentage was tried to be mitigated by setting a correlation percentage value in the WANem advanced mode view (figure 6). Setting the correlation value to anything else than 0% however, caused the emulated packet loss percentage to drop to 0%. With some investigation, this behaviour was found out to be a known bug in the Netem-libraries, that WANem uses for packet loss emulation [11].

The reason what causes WANem packet loss emulation to be so erratic was not found. One speculated cause is that WANem, and the OS that it is built on, use 32-bit architecture. It might be that due to the limit of maximum 4 Gb RAM utilization, WANem simply can not emulate steady network impairments on high speed connections.

7 iPerf3 Operation

The operation methods of iPerf3 determine the results witnessed in the tests that were ran. For this reason, iPerf3 method to calculate packet loss is explained. iPerf3 methods to calculate jitter, duplication and reordering are also explained.

7.1 Packet Loss

iPerf3 transmitter inserts a running sequence number to all the payload packets it sends during a traffic generation session. iPerf3 receiver extracts these sequence numbers from the received packets and determines if packets have been lost. Below is the part of the iPerf3 source code performing packet loss detection [12].

```
if (pcount >= sp->packet_count + 1) {

          if (pcount > sp->packet_count + 1) {
           sp->cnt_error += (pcount - 1) - sp->packet_count;
          }
          sp->packet_count = pcount;
}
```

First iPerf3 checks if the sequence number of the packet received is going forward, by comparing the sequence number of the packet received `pcount` to the highest seen sequence number `sp->packet_count`. Then it checks if there is a gap in the two sequence numbers, if a gap is detected, it is count as a loss and the `cnt_error` variable is incremented. Lastly the highest seen sequence number is updated. [12]

When the iPerf3 traffic generation session ends, the transmitter and the receiver exchange control information with a TCP stream [9]. With this exchange the iPerf3 receiver gets the total number of payload packets sent from the transmitter. Now the iPerf3 receiver can compare the number of lost packets to the number of sent packets and confirm the total packet loss percentage. In chapter 5.2.2, it was mentioned that the iPerf3

throughput report interval was kept at the default value of one second. The length of this report interval has a fundamental effect on the value of packet loss perceived per interval, discussed in chapter 6.1 and Appendices 1-3.

The total packet loss calculated at the end of the iPerf3 traffic generation session is unaffected by the report interval, but the value of packet loss per interval can fluctuate a lot depending on the report interval. If the report interval was longer, in our testing all the emulators would have gotten per interval packet loss emulation results closer to the target 4%, simply because the average packet loss would have been calculated from a bigger sample of packets. A shorter report interval would have caused the emulators to get result that deviate more from the target 4% packet loss. This effect is illustrated in the table below.

A number sequence, N=24		Sample size	Average(s)	
4		24	4	
3		12	3,75|4,25	
3		8	3,63|4|4,38	
3		6	3,5|4|4,17|4,33	
4				
4				
4				
4				
4				
4				
4				
4				
4				
4				
4				
4				
5				
4				
4				
5				
5				
4				
4				
4				

Table 4. Relation of sample size and average

Average value is calculated from the same sequence of numbers using different sample sizes. The smaller the sample size gets, the more the average starts to deviate from the overall average. The same effect applies when iPerf3 calculates packet loss per interval.

7.2 Jitter

iPerf3 transmitter insert a timestamp to all the packets it sends during a traffic generation session. iPerf3 receiver then extracts these timestamps from the received packets and compares them to the current time to calculate the delay of the packet. These

computations are based on the Request for Comments (RFC) 1889 standard, sections 6.3.1 and A.8 [12]. Section 6.3.1 describes a sender report Real-time Transport Protocol (RTP) Control Protocol (RTCP) packet, as shown below [13].

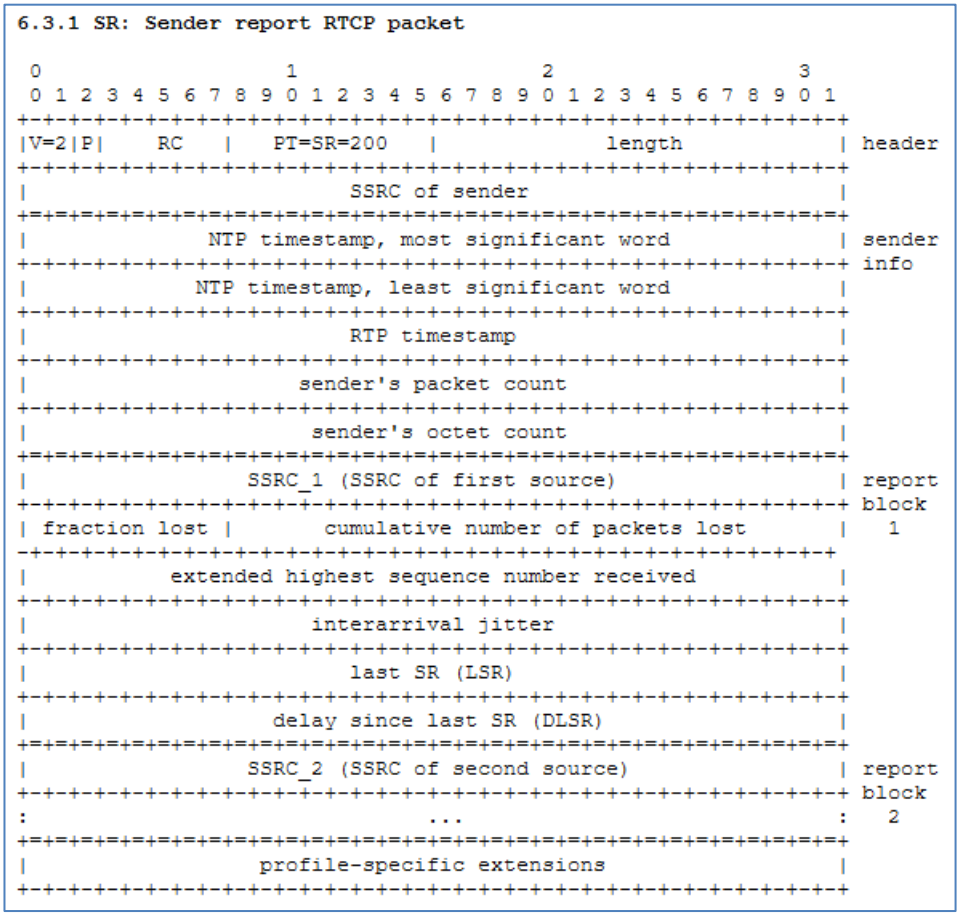

Figure 14. Sender report RTCP packet

The fields in this packet that we are interested in are the NTP timestamp fields. The NTP field contains the wallclock time when the packet was sent. If the wallclock time is not available to the transmitter, the value of this field would be set to zero. [13]. In our test case the transmitter had an NTP time server connection, so this field contains a non-zero time value. This is the timestamp that the iPerf3 receiver extracts from the packets in our test case.

Section A.8 describes the algorithm that iPerf3 uses to estimate jitter. The delta times of the received packets are unknown to iPerf3, so the jitter measurement is in fact a close estimate, not an exact measurement. [12]. Delta time is the time between the transmission of two consecutive packets. Below is the algorithm described in Section A.8 [13].

```
int transit = arrival - r->ts;
int d = transit - s->transit;
s->transit = transit;
if (d < 0) d = -d;
s->jitter += (1./16.) * ((double)d - s->jitter);
```

The above code snippet requires two inputs: `r->ts`, which is the timestamp from the incoming packet and `arrival`, which is the current time on the system. The variable `transit`, indicating the transit time for the received packet, is calculated by reducing the timestamp time from the current time. The floating variable `s->transit` holds the transit time of the previously received packet. A temporary variable `d` is calculated for jitter measurement and it is forced to be a non-negative number. The variable `s->jitter` holds a floating value of the estimated jitter. The integer 16 in the calculations is derived from the transmission window size. [13]

Below is the part of the iPerf3 source code, that performs jitter calculations [12].

```
iperf_time_now(&arrival_time);

iperf_time_diff(&arrival_time, &sent_time, &temp_time);
transit = iperf_time_in_secs(&temp_time);

d = transit - sp->prev_transit;
if (d < 0)
     d = -d;
sp->prev_transit = transit;
sp->jitter += (d - sp->jitter) / 16.0;
```

The first method `iperf_time_now` sets the variable `arrival_time` to be the current time prompted from the system. The second method `iperf_time_diff` sets the value for the variable `temp_time` by calculating the difference between the variable's `arrival_time` and `sent_time` according to the algorithm described in RFC 1889 section A.8 [14]. In the third line, the value of the variable `transit` is set to be the value of the variable `temp_time` in seconds. In the last five lines of the code snippet, a temporary variable `d` is created from the current and previous transit value, the variable `d` is forced to be non-negative and the variable `sp->jitter` is calculated from the value of variable `d` and the value of the previous jitter estimate. This calculation too is described in the RFC 1889 section A.8. [12]

7.3 Packet Reordering & Duplication

iPerf3 detects both reordered and duplicated packets as packets received out-of-order, as previously mentioned in chapter 6.2. In the iPerf3 source code, packet loss detection first checks if the packet sequence numbers are going forward as mentioned in chapter 7.1. If the packet sequence number is not going forward, meaning the first if-clause is not true, the logic moves on to the else-clause shown below. [12]

```
else {
      sp->outoforder_packets++;

      if (sp->cnt_error > 0)
            sp->cnt_error--;
}
```

If the sequence number moves backwards or stays the same, the value of the variable `outoforder_packets` is incremented by one. Also, if the value of the variable `cnt_error` is not zero, its value is decremented by one. The decrementing is done because an out-of-order packet offsets a prior sequence number gap counted as a loss. [12]

8 Follow-up

SCE was found to be a fit candidate to be an alternative to the N91. Next, SCE is to be tried out by the parties currently using N91 for WAN emulation in their testing. The SCE will replace the N91 in some of the current tests for some period. The purpose is to test SCE operation in the real environment.

Currently the tests in which the N91s are used are automated. Ability to operate the N91 with CLI is crucial because of this. SCE also offers the ability to operate with command-line parameters used from the Windows PowerShell or Command Prompt (CMD) [8]. The current automated tests will have to be adjusted to use the WAN emulator this way.

A Professional Edition license of SCE for a single device will be purchased, as it comes with the necessary features needed [15]. SCE will be installed on a specified Hewlett Packard Enterprise (HPE) Workstation. The machine has the following specifications:

CPU	Intel Xeon E -2236 3.4GHz 6C
RAM	16GB DDR4 2666 DIMM ECC
NIC	Intel X550-T2 10GbE Dual Port NIC
SSD	HP Z Turbo Drive M.2 512GB TLC SED
HDD	2TB 7200RPM SATA

Table 5. HPE Workstation specifications

The testing parties have had complains about the graphs produced by the N91 regarding the unclarity and usability of them. By early remarks, SCE does produce clear and easy-to-use graphs, that the testing parties are satisfied with. Graphs produced by the emulators was not a comparison point in our testing, but it is, nevertheless, a good addition with the SCE according to the users of the N91.

Multi-link emulation with SCE is something that will be tested in the future, if the single-link emulation in the real environment is found to be successful. This means that the NIC presented in Table 5, would be switched to one with four ports. The goal is to be able to simulate four links with the SCE workstation in such a way that it matches the link capacity of the N91.

If the testing parties find the SCE to be a satisfying alternative to the N91, more SCE-licenses and workstations will be acquired depending on the future.

9 Conclusion

The current way to emulate a WAN is not cost-efficiently scalable, and a new way needed to be found. The requirements for the new WAN emulator were carefully set based on the currently used WAN emulator. When the plan was ready, it was executed closely, and the following results were reviewed.

The results of this project indicate that a hardware-based WAN emulator can be replaced by a software-based WAN emulator. There is, however, a lot of variation in the reliability and quality of the emulator applications offered by different companies. The applications should always be tested against the solution currently in place before implementation.

One thing to keep in mind is that even though the software-based emulators performance comes close to that of a hardware-based one, it can not currently truly match it.

SoftPerfect Connection Emulator was selected for further comparison against the Netropy N91. The next step in the comparison process is to implement SCE into the production testing environments, to see how it performs in our real use scenarios. The thorough pilot testing phase performed by this project suggests that SCE can replace the N91 in some of our use cases. Netropy N91 devices will how ever remain in use, as some of the use cases require quite accurate performance from the WAN emulator. As stated earlier, software can not yet, if ever, fully match hardware performance.

The goal of this project was to find a more cost-effective WAN emulator. This goal was met, as the cost of the new solution is approximately 7,5% of that of the old solution. The final testing of the new solution, however, is not yet complete. The duration of the transfer period from the old solution to the new one depends on many different things, most of which can not be influenced by the implementer of this project.

References

1 Apposite Technologies. 2017. Online. Netropy Datasheet - N91. https://www.epsglobal.com/Media-Library/EPSGlobal/Products/files/apposite/N91-1G.pdf?ext=.pdf. Accessed October 25, 2020.

2 IWL. 2020. Online. Causes and Correlation of Network Impairments. https://iwl.com/idocs/causes-and-correlation-of-network-impairments. Accessed October 25, 2020.

3 Accedian Networks. 2012. Online. White Paper – One-Way Delay Measurement Techniques. https://accedian.com/wp-content/uploads/2015/05/One-WayDelay-MeasurementTechniques-AccedianWhitePaper.pdf. Accessed November 5, 2020.

4 Meinberg. 2020. Online. Modular Synchronization Platform LANTIME M1000. https://www.meinbergglobal.com/english/products/modular-1u-sync-system.htm. Accessed November 5, 2020.

5 WANem. Online. WANem: The Wide Are Network Emulator. http://wanem.sourceforge.net/. Accessed October 25, 2020.

6 WANem. 2007. Online. Wide Area Network Emulator User Guide. https://netix.dl.sourceforge.net/project/wanem/Documents/WANemv11-User-Guide.pdf. Accessed October 25, 2020.

7 SoftPerfect Network Management Solutions. 2020. Online. SoftPerfect Connection Emulator. https://www.softperfect.com/products/connectionemulator/. Accessed October 25, 2020.

8 SoftPerfect Network Management Solutions. 2020. Online. SoftPerfect Connection Emulator Online User Manual. https://www.softperfect.com/products/connectionemulator/manual/. Accessed October 25, 2020.

9 ESnet. 2020. Online. Invoking iPerf3. https://software.es.net/iperf/invoking.html. Accessed October 25, 2020.

10 Doyle, Sean. 2018. Online. TCP vs. UDP: Understanding the Difference. Private Internet Access. https://www.privateinternetaccess.com/blog/tcp-vs-udp-understanding-the-difference/. Accessed October 25, 2020.

11 Networking Group. 2014. Online. NetemCLG (Correlated Loss Generator): fixing loss model of Netem. http://netgroup.uniroma2.it/twiki/bin/view.cgi/Main/NetemCLG. Accessed November 5, 2020.

12 GitHub/ESnet/iperf. 2020. Online. iperf/iperf_udp.c. https://github.com/esnet/iperf/blob/master/src/iperf_udp.c. Accessed November 14, 2020.

13 Schulzrinne H., Casner S., Frederick R., Jacobson V. 1996. Lawrence Berkeley National Laboratory. Online. RTP: A Transport Protocol for Real-Time Applications. https://tools.ietf.org/html/rfc1889#section-6.3.1. Accessed November 14, 2020.

14 GitHub/ESnet/iperf. 2019. Online. iperf/iperf_time.c. https://github.com/esnet/iperf/blob/master/src/iperf_time.c. Accessed November 14, 2020.

15 SoftPerfect Network Management Solutions. 2020. Online. Purchasing Information. https://www.softperfect.com/order/?product=sce. Accessed October 25, 2020.

iPerf3 output - N91 / 4% packet loss

```
Connecting to host xxx.xxx.xxx.xxx, port 5201
[  5] local xxx.xxx.xxx.xxx port 35471 connected to xxx.xxx.xxx.xxx port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  5]   0.00-1.00   sec   116 KBytes   950 Kbits/sec  82
[  5]   1.00-2.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   2.00-3.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   3.00-4.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   5.00-6.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   7.00-8.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   9.00-10.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  10.00-11.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  11.00-12.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  12.00-13.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  13.00-14.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  14.00-15.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  15.00-16.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  16.00-17.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  17.00-18.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  18.00-19.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  19.00-20.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  20.00-21.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  21.00-22.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  22.00-23.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  23.00-24.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  24.00-25.00  sec   127 KBytes  1.04 Mbits/sec  90
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-25.00  sec  3.11 MBytes  1.04 Mbits/sec  0.164 ms  88/2254 (3.9%)
[  5] Sent 2254 datagrams
```

Server output:

```
Accepted connection from xxx.xxx.xxx.xxx, port 36936
[  5] local xxx.xxx.xxx.xxx port 5201 connected to xxx.xxx.xxx.xxx port 35471
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   116 KBytes   950 Kbits/sec  9.232 ms  0/82 (0%)
[  5]   1.00-2.00   sec   123 KBytes  1.01 Mbits/sec  0.185 ms  4/91 (4.4%)
[  5]   2.00-3.00   sec   122 KBytes   996 Kbits/sec  0.160 ms  4/90 (4.4%)
[  5]   3.00-4.00   sec   123 KBytes  1.01 Mbits/sec  0.164 ms  4/91 (4.4%)
[  5]   4.00-5.00   sec   122 KBytes   996 Kbits/sec  0.161 ms  4/90 (4.4%)
[  5]   5.00-6.00   sec   123 KBytes  1.01 Mbits/sec  0.170 ms  4/91 (4.4%)
[  5]   6.00-7.00   sec   122 KBytes   996 Kbits/sec  0.103 ms  4/90 (4.4%)
[  5]   7.00-8.00   sec   123 KBytes  1.01 Mbits/sec  0.164 ms  4/91 (4.4%)
[  5]   8.00-9.00   sec   122 KBytes   996 Kbits/sec  0.150 ms  4/90 (4.4%)
[  5]   9.00-10.00  sec   123 KBytes  1.01 Mbits/sec  0.166 ms  4/91 (4.4%)
[  5]  10.00-11.00  sec   127 KBytes  1.04 Mbits/sec  0.157 ms  0/90 (0%)
```

```
[  5]  11.00-12.00  sec   123 KBytes  1.01 Mbits/sec  0.161 ms  4/91 (4.4%)
[  5]  12.00-13.00  sec   122 KBytes   996 Kbits/sec  0.162 ms  4/90 (4.4%)
[  5]  13.00-14.00  sec   123 KBytes  1.01 Mbits/sec  0.167 ms  4/91 (4.4%)
[  5]  14.00-15.00  sec   122 KBytes   996 Kbits/sec  0.134 ms  4/90 (4.4%)
[  5]  15.00-16.00  sec   123 KBytes  1.01 Mbits/sec  0.168 ms  4/91 (4.4%)
[  5]  16.00-17.00  sec   122 KBytes   996 Kbits/sec  0.138 ms  4/90 (4.4%)
[  5]  17.00-18.00  sec   123 KBytes  1.01 Mbits/sec  0.141 ms  4/91 (4.4%)
[  5]  18.00-19.00  sec   122 KBytes   997 Kbits/sec  0.172 ms  4/90 (4.4%)
[  5]  19.00-20.00  sec   123 KBytes  1.01 Mbits/sec  0.191 ms  4/91 (4.4%)
[  5]  20.00-21.00  sec   122 KBytes   997 Kbits/sec  0.211 ms  4/90 (4.4%)
[  5]  21.00-22.00  sec   129 KBytes  1.05 Mbits/sec  0.164 ms  0/91 (0%)
[  5]  22.00-23.00  sec   122 KBytes   996 Kbits/sec  0.159 ms  4/90 (4.4%)
[  5]  23.00-24.00  sec   123 KBytes  1.01 Mbits/sec  0.166 ms  4/91 (4.4%)
[  5]  24.00-25.00  sec   122 KBytes   996 Kbits/sec  0.164 ms  4/90 (4.4%)
[  5]  25.00-25.04  sec  0.00 Bytes  0.00 bits/sec  0.164 ms  0/0 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  5]   0.00-25.04  sec  0.00 Bytes  0.00 bits/sec  0.164 ms  88/2254 (3.9%)

iperf Done.
```

iPerf3 output - WANem / 4% packet loss

Connecting to host xxx.xxx.xxx.xxx, port 5201
[5] local xxx.xxx.xxx.xxx port 50619 connected to xxx.xxx.xxx.xxx port 5201
[ID] Interval Transfer Bandwidth Total Datagrams
[5] 0.00-1.00 sec 116 KBytes 950 Kbits/sec 82
[5] 1.00-2.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 2.00-3.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 3.00-4.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 4.00-5.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 5.00-6.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 6.00-7.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 7.00-8.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 8.00-9.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 9.00-10.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 10.00-11.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 11.00-12.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 12.00-13.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 13.00-14.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 14.00-15.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 15.00-16.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 16.00-17.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 17.00-18.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 18.00-19.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 19.00-20.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 20.00-21.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 21.00-22.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 22.00-23.00 sec 127 KBytes 1.04 Mbits/sec 90
[5] 23.00-24.00 sec 129 KBytes 1.05 Mbits/sec 91
[5] 24.00-25.00 sec 127 KBytes 1.04 Mbits/sec 90
- -
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[5] 0.00-25.00 sec 3.11 MBytes 1.04 Mbits/sec 0.009 ms 86/2254 (3.8%)
[5] Sent 2254 datagrams

Server output:
Accepted connection from xxx.xxx.xxx.xxx, port 36928
[6] local xxx.xxx.xxx.xxx port 5201 connected to xxx.xxx.xxx.xxx port 50619
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[6] 0.00-1.00 sec 113 KBytes 926 Kbits/sec 6.892 ms 2/82 (2.4%)
[6] 1.00-2.00 sec 119 KBytes 973 Kbits/sec 0.066 ms 7/91 (7.7%)
[6] 2.00-3.00 sec 122 KBytes 996 Kbits/sec 0.006 ms 4/90 (4.4%)
[6] 3.00-4.00 sec 122 KBytes 996 Kbits/sec 0.010 ms 5/91 (5.5%)
[6] 4.00-5.00 sec 126 KBytes 1.03 Mbits/sec 0.014 ms 1/90 (1.1%)
[6] 5.00-6.00 sec 123 KBytes 1.01 Mbits/sec 0.007 ms 4/91 (4.4%)
[6] 6.00-7.00 sec 122 KBytes 996 Kbits/sec 0.005 ms 4/90 (4.4%)
[6] 7.00-8.00 sec 126 KBytes 1.03 Mbits/sec 0.005 ms 2/91 (2.2%)
[6] 8.00-9.00 sec 122 KBytes 996 Kbits/sec 0.004 ms 4/90 (4.4%)
[6] 9.00-10.00 sec 123 KBytes 1.01 Mbits/sec 0.006 ms 4/91 (4.4%)
[6] 10.00-11.00 sec 122 KBytes 996 Kbits/sec 0.006 ms 4/90 (4.4%)
[6] 11.00-12.00 sec 124 KBytes 1.02 Mbits/sec 0.010 ms 3/91 (3.3%)

```
[  6]  12.00-13.00  sec   120 KBytes   985 Kbits/sec  0.007 ms  5/90 (5.6%)
[  6]  13.00-14.00  sec   119 KBytes   973 Kbits/sec  0.008 ms  7/91 (7.7%)
[  6]  14.00-15.00  sec   123 KBytes  1.01 Mbits/sec  0.010 ms  3/90 (3.3%)
[  6]  15.00-16.00  sec   126 KBytes  1.03 Mbits/sec  0.005 ms  2/91 (2.2%)
[  6]  16.00-17.00  sec   122 KBytes   996 Kbits/sec  0.007 ms  4/90 (4.4%)
[  6]  17.00-18.00  sec   124 KBytes  1.02 Mbits/sec  0.008 ms  3/91 (3.3%)
[  6]  18.00-19.00  sec   124 KBytes  1.02 Mbits/sec  0.007 ms  2/90 (2.2%)
[  6]  19.00-20.00  sec   129 KBytes  1.05 Mbits/sec  0.007 ms  0/91 (0%)
[  6]  20.00-21.00  sec   124 KBytes  1.02 Mbits/sec  0.009 ms  2/90 (2.2%)
[  6]  21.00-22.00  sec   126 KBytes  1.03 Mbits/sec  0.010 ms  2/91 (2.2%)
[  6]  22.00-23.00  sec   117 KBytes   961 Kbits/sec  0.008 ms  7/90 (7.8%)
[  6]  23.00-24.00  sec   127 KBytes  1.04 Mbits/sec  0.007 ms  1/91 (1.1%)
[  6]  24.00-25.00  sec   122 KBytes   996 Kbits/sec  0.009 ms  4/90 (4.4%)
[  6]  25.00-25.04  sec  0.00 Bytes  0.00 bits/sec  0.009 ms  0/0 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  6]   0.00-25.04  sec  0.00 Bytes  0.00 bits/sec  0.009 ms  86/2254 (3.8%)

iperf Done.
```

iPerf3 output - SCE / 4% packet loss

```
Connecting to host xxx.xxx.xxx.xxx, port 5201
[  5] local xxx.xxx.xxx.xxx port 58430 connected to xxx.xxx.xxx.xxx port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  5]   0.00-1.00   sec   116 KBytes   950 Kbits/sec  82
[  5]   1.00-2.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   2.00-3.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   3.00-4.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   5.00-6.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   7.00-8.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   9.00-10.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  10.00-11.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  11.00-12.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  12.00-13.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  13.00-14.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  14.00-15.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  15.00-16.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  16.00-17.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  17.00-18.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  18.00-19.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  19.00-20.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  20.00-21.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  21.00-22.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  22.00-23.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  23.00-24.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  24.00-25.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  25.00-26.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  26.00-27.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  27.00-28.00  sec   127 KBytes  1.04 Mbits/sec  90
[  5]  28.00-29.00  sec   129 KBytes  1.05 Mbits/sec  91
[  5]  29.00-30.00  sec   127 KBytes  1.04 Mbits/sec  90
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-30.00  sec  3.74 MBytes  1.05 Mbits/sec  0.032 ms  114/2707 (4.2%)
[  5] Sent 2707 datagrams


Server output:
Accepted connection from xxx.xxx.xxx.xxx, port 36898
[  5] local xxx.xxx.xxx.xxx port 5201 connected to xxx.xxx.xxx.xxx port 58430
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   110 KBytes   903 Kbits/sec  7.827 ms  4/82 (4.2%)
[  5]   1.00-2.00   sec   117 KBytes   961 Kbits/sec  0.044 ms  8/91 (4.4%)
[  5]   2.00-3.00   sec   126 KBytes  1.03 Mbits/sec  0.009 ms  1/90 (3.9%)
[  5]   3.00-4.00   sec   124 KBytes  1.02 Mbits/sec  0.012 ms  3/91 (3.7%)
[  5]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec  0.004 ms  0/90 (0%)
[  5]   5.00-6.00   sec   123 KBytes  1.01 Mbits/sec  0.033 ms  4/91 (4.4%)
[  5]   6.00-7.00   sec   117 KBytes   961 Kbits/sec  0.010 ms  7/90 (4.4%)
```

```
[  5]   7.00-8.00   sec   123 KBytes  1.01 Mbits/sec  0.010 ms  4/91 (4.4%)
[  5]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  0.008 ms  4/90 (4.0%)
[  5]   9.00-10.00  sec   117 KBytes   961 Kbits/sec  0.005 ms  8/91 (3.9%)
[  5]  10.00-11.00  sec   124 KBytes  1.02 Mbits/sec  0.007 ms  2/90 (4.1%)
[  5]  11.00-12.00  sec   124 KBytes  1.02 Mbits/sec  0.009 ms  2/90 (4.2%)
[  5]  12.00-13.00  sec   116 KBytes   950 Kbits/sec  0.007 ms  9/91 (4.2%)
[  5]  13.00-14.00  sec   120 KBytes   985 Kbits/sec  0.013 ms  6/91 (2.2%)
[  5]  14.00-15.00  sec   126 KBytes  1.03 Mbits/sec  0.019 ms  1/90 (4.3%)
[  5]  15.00-16.00  sec   126 KBytes  1.03 Mbits/sec  0.033 ms  2/91 (4.2%)
[  5]  16.00-17.00  sec   122 KBytes   996 Kbits/sec  0.026 ms  4/90 (3.8%)
[  5]  17.00-18.00  sec   119 KBytes   973 Kbits/sec  0.028 ms  7/91 (3.9%)
[  5]  18.00-19.00  sec   122 KBytes   996 Kbits/sec  0.028 ms  4/90 (3.8%)
[  5]  19.00-20.00  sec   119 KBytes   973 Kbits/sec  0.038 ms  7/91 (4.3%)
[  5]  20.00-21.00  sec   127 KBytes  1.04 Mbits/sec  0.032 ms  5/90 (4.4%)
[  5]  21.00-22.00  sec   126 KBytes  1.03 Mbits/sec  0.033 ms  2/91 (3.9%)
[  5]  22.00-23.00  sec   112 KBytes   915 Kbits/sec  0.030 ms  9/90 (4.0%)
[  5]  23.00-24.00  sec   124 KBytes  1.02 Mbits/sec  0.026 ms  3/91 (4.3%)
[  5]  24.00-25.00  sec   127 KBytes  1.04 Mbits/sec  0.026 ms  1/91 (4.0%)
[  5]  25.00-26.00  sec   124 KBytes  1.02 Mbits/sec  0.028 ms  2/90 (3.6%)
[  5]  26.00-27.00  sec   127 KBytes  1.04 Mbits/sec  0.032 ms  1/91 (4.0%)
[  5]  27.00-28.00  sec   120 KBytes   985 Kbits/sec  0.032 ms  5/90 (4.4%)
[  5]  28.00-29.00  sec   126 KBytes  1.03 Mbits/sec  0.032 ms  2/91 (3.6%)
[  5]  29.00-30.00  sec   122 KBytes   996 Kbits/sec  0.032 ms  4/90 (4.3%)
[  5]  30.00-30.04  sec  0.00 Bytes  0.00 bits/sec  0.032 ms  0/0 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  5]   0.00-30.04  sec  0.00 Bytes  0.00 bits/sec  0.032 ms  114/2707 (4.2%)

iperf Done.
```

iPerf3 output – N91 & SCE / Duplication & Reordering

The following extracts show the iPerf3 output server section summary lines for each test accordingly.

N91, Duplication 5%:
[SUM] 0.0-25.0 sec 116 datagrams received out-of-order

SCE, Duplication 5%:
[SUM] 0.0-25.0 sec 120 datagrams received out-of-order

N91, Reordering 5%:
[SUM] 0.0-25.0 sec 101 datagrams received out-of-order

SCE, Reordering 5%:
[SUM] 0.0-25.0 sec 115 datagrams received out-of-order