Grant Agreement No.: 610764
Instrument: Collaborative Project Call
Identifier: FP7-ICT-2013-10

PANACEA

Proactive Autonomic Management of Cloud Resources

D4.2: Overlay Simulation Environment

Version: v.1.0

Work package	WP 4
Task	Task 4.2
Due date	31/03/2015
Submission date	09/04/2015
Deliverable lead	QoS Design
Version	1.0
Authors	Anouar Rachdi
Reviewers	David Garcia, Gokce Gorbil

Abstract	This document presents the overlay simulation environment of the PANACEA cloud and service management solution. In contrast with the other modules in the PANACEA solution, the simulation module is not used during the operational phase of cloud management activities, but rather during the design and evaluation of the solution, specifically the overlay network component. Due to practical reasons, it is generally infeasible to conduct realistic large scale networking experiments in the Internet, in which we can inject different types of anomalies to evaluate and validate the self-healing and self-optimization properties of the overlay network. The overlay simulation module enables us to conduct such experiments for the validation of the overlay network at scale.

Keywords	Overlay, Simulation, Adverse events, MPLS, Emulation, Web application, DAaaS

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	15.02. 2015	First version of deliverable	Anouar Rachdi
V1.0	09.04.2015	Final version	Anouar Rachdi

Disclaimer

Copyright notice

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Nature of the deliverable:	R	
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to bodies determined by the PANACEA project	
CO	Confidential to PANACEA project and Commission Services	

EXECUTIVE SUMMARY

Overlay networks are seen as a solution for flexible traffic engineering over IP networks. In fact, IP networks are suffering from significant congestion and slow recovery from failures. This can have a major impact on distributed applications (over clouds).

The overlay system developed by CNRS within PANACEA project remediates to some of the main deficiencies of the Internet. It's a self-healing, self-optimizing and highly scalable overlay system that accepts customized routing policies formulated by distributed applications according to their own needs.

The PANACEA overlay network (PON) is formed by software routers and is able to monitor the quality of Internet paths (latency, bandwidth, loss rate) between overlay nodes and optimize routes for packets when the given primary path becomes unavailable or suffers from congestion.

QoS Design's contribution within PANACEA project is a generic and scalable overlay simulation module. This module is intended for designing and optimizing overlay networks. This simulation module will be used also for validation purposes and afterwards integrated in the QoS Design´s NEST Simulation Environment.

This document describes the methodology that will be used to validate the PON. Two approaches will be investigated, the first one is purely simulation (relies on an analytical model). This approach has the benefit of being able to experiment any kind of scenarios. The drawback is that all targeted applications cannot be easily modelled within the simulation process. This is why we will only consider the web application, from the TPC framework, use case to experiment this approach.

The second approach proposes a validation lab environment that combine emulation and simulation (hybrid simulation), not only for the PON, but also for any critical distributed application (over the cloud) that has to be stress-tested in near real conditions before deployment.

In this approach both overlay and underlay are included in the simulation and the emulation processes. Scalabity can be an issue for large scale experimentation. But there is a workaround by spanning the emulation process over multiple instances. For this approach we will consider a first step validation by using the web service use case. Our target is to validate the PON on the DAaaS use case.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
BGP	Border Gateway Protocol
BSD	Berkley Software Distribution
CAPEX	Capital Expenditure
CDN	Content Delivery Network
CORE	Common Open Research Emulator
CPU	Central Processing Unit
DAaaS	Data Analytics as a Service
DNS	Domain Name System
DWDM	Dense Wavelength Division Multiplexing
FTTx	Fiber to the x (Curb/Building/Home)
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
I/O	Input/Output
IP	Internet Protocol
ISIS	Intermediate system to Intermediate system
LAN	Local Area Network
LSP	Label Switched Path
MPLS	Multiprotocol Label Switching
NVO	Network Virtualization Overlays
OPEX	Operational Expenditure
OSPF	Open Shortest Path First
QoS	Quality of Service
SDH	Synchrounous Digital Hierarchy
SLA	Service Level Agreement
TPC-W	Web e-Commerce Transaction Processing Benchmark
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
VXLAN	Virtual Extensible Local Area Network

WAN	Wide Area Network
WLAN	Wireless Local Area Network
WP	Work Package

1 INTRODUCTION

1.1 State of the art

First of all, we distinguish between network simulators and overlay simulators. The formers provide packet-level simulation of network protocols (TCP, UDP, IP, etc) over realistic Internet topologies. They include, congestion-aware simulation including packet-loss and queuing delays. On the other side, overlay simulators are usually more interested in evaluating overlay algorithms and its routing behaviour without even taking into account the underlying network layer.

For example, the NS [16] network simulator provides a standard framework for accurate simulation of network protocols. NS is appropriate to simulate networks in the link, switching and transport layer but it is not aimed for application level overlays. Besides, for smaller scale scenarios NS performs gracefully, but for overlays over a hundred nodes in size suffers considerable scaling problems. Another example is the J-Sim [17] network simulation framework that follows a component oriented approach. Being easier to use than Ns-2, J-Sim also lacks enough scalability and performance for big overlays.

Other network simulators like SSFNET[18] and OMNET++[19] have also been successfully used for peer to peer applications. Particularly, OMNET++ provides a rich environment that enables both packet-level simulations and high-level overlay protocols. Nevertheless, all these network simulators are mainly aimed for packet-level protocols, and impose additional complexity to the user learning curve.

In the end, many research groups have created their own overlay simulators, sacrificing accuracy for scale. Examples of these include p-sim, FreePasty, SimPastry, 3LS, PLP2P, and SimPˆ2. In the field of structured overlays, one of the pioneers is MIT's pspsim. This simulator currently supports many protocols, including Chord, Koorde, Kelips, Tapestry, and Kademlia. Besides, from the software engineering perspective, this simulator is poorly documented and difficult to extend for different purposes.

FreePastry [20], the Java open-source implementation of the Pastry structured P2P protocol includes as well, the possibility to simulate applications on top of this overlay network. As in PlanetSim[21], FreePastry provides a Common API to the applications built on top of it, thus making it very easy for developers to create and simulate complex distributed applications. Protocol specific details remain hidden from the application-level point of view. However, FreePastry is highly tied to the Pastry protocol, and it does not permit simulation of its applications on top of other structured P2P protocols.

Another interesting approach is the one followed by MACEDON [22]. Macedon provides an infrastructure to ease development, evaluation, and iterative design of overlay algorithms. Macedon is not limited to structured P2P networks, and it includes an impressive variety of protocols and applications such as AMMO, Bullet, Chord, NICE, Overcast, Pastry, Scribe, and SplitStream. Furthermore, MACEDON simplifies development of new overlays using a finite state machine (FSM) model for defining overlay protocols. MACEDON is a very nice tool for overlay simulation but it follows a completely different approach than PlanetSim. MACEDON is mainly related to Domain-specific languages (DSLs) that generate functional code from domain specific representations.

1.2 Goal of the simulation in the PANACEA project

The overlay simulation module of the PANACEA solution operates closely with the overlay network system as it enables the design and optimization of **large-scale** overlay networks. The simulator enables accurate modelling of the underlying physical network, including the routers, links, data centres, and network protocols. Thus, it can evaluate the performance of the PON under large-scale realistic Internet conditions. It can perform what-if analysis for several adverse events, such as link, router and server failures and congestion.

These evaluations allow the validation of the self-healing and self-optimizing features of the overlay network system. The overlay simulator will be used in the design phase of the overlay network in order to evaluate different designs for efficient network optimization, and to evaluate the robustness of the overlay to network disruptions.

The overlay simulator will also be used for validating the final PON according to the selected uses cases from Deliverable D4.1 (Web service [23] and DAaaS [24]).

The performances of each distributed service will be compared while using the overlay system and without.

1.3 Targeted groups

Several user categories can benefit from this simulation module. Among others, we first mention telecom operators performing virtualization of some carrier functionalities. Using this module will help achieve this in a controlled fashion. These operators can validate latency and reliability goals before deploying in the real system.

On the other hand, WAN operators can perform the “slicing” of the network and share network resources efficiently among all deployed services, and thus ensuring better QoS for the customers. The simulation module can be also beneficial for CDN operators. In this case, performance metrics and bandwidth consumption can be validated before any deployment.

In general, any operator building an overlay network for mission critical services and aiming at guaranteeing some performances can benefit from the simulation approach, lowering operational expenditures (OPEX) as the implemented solutions will be more robust and will not need a lot of manual configuration over the time. They can also reduce network consumption, deploy more services and delay investments for network equipment (CAPEX reduction).

1.4 Document structure

The rest of this document is organized as follows. Section 2 will and 3 describes the background that will be used to create the simulation module.

The second section is dedicated to QoS Design’s NEST environment. It will briefly illustrate the existing functionalities of NEST and their application domain. It will also introduce the required evolutions to be applied to conduct the aimed experiments.

The third section is dedicated to the CORE Emulation platform. It will explain the functioning of the emulator and the possibilities to be combined with a simulation environment.

The Fourth section describes the first approach based on analytical simulation within NEST Environment. We will explain in this section how NEST simulator will implement overlay networks how they will be simulated and optimized. We will also explain the experimentation with a web application and discuss the main metrics and impacts.

The Fifth section describe a hybrid approach that combine simulation and emulation. We will first describe how we will combine the simulation and the emulation processes. We will describe then the two experiments and discuss the metrics and impacts in both cases.

The last chapter is dedicated to a conclusion of this work.

2 DESCRIPTION OF NEST ENVIRONMENT

2.1 Introduction

In all countries, telecommunication networks have become a strategic issue for the economic development. Telecom operators responsible of such systems are currently waiting for rich and powerful tools for the engineering, the overall mastery of the quality of service at a large-scale, the planning, the control and the supervision of next-generation, fixed, mobile and optical networks. The market at a worldwide scale is enormous.

QoS Design is a spin-off that emerged from LAAS-CNRS (www.laas.fr) in 2004. The founders, recognized researchers, have developed a set of innovative software solutions under the name of NEST (Network Engineering and Simulation Tool). The NEST environment consists of a set of tools for mastering the performance of next-generation, fixed, mobile and optical telecommunications networks (IP networks, MPLS networks, ADSL, Metro Ethernet, service platforms, Mobile (3G/4G), VPN, SDH, DWDM, FTTx).

The solutions proposed by QoS Design are aimed at telecom operators to optimize the infrastructure planning, guarantee the quality of service, and supervise all fixed and mobile networks. The market is huge as it concerns not only telecom operators, but also international WAN network administrators (banks, big companies, etc.) and security/defense operators (e.g. the DIRISI in France).

Today, QoS Design's environment NEST is fully operational. After several years of research, innovation and software development, the solutions proposed by QoS Design have reached industrial maturity and are already used by many large telecom operators. In addition, several international consultations were successfully won.

2.2 NEST Overview

NEST suite consists of a set of software that allows the design, the engineering and the planning of the complex network infrastructures of an operator (fixed / mobile).

The software suite consists of the following products:

- **NEST Designer** provides a unified framework for designing wired network architectures. It acts as a "design and plane" tool for IP-MPLS networks. NEST Designer provides a global optimization approach to conceive new network architectures, design its topology, select and dimension equipment while optimizing the overall project cost under technical, financial (CAPEX) and QoS constraints. NEST Designer also provides algorithms for capacity planning and the brownfield design [9,10]. Starting from an existing architecture, NEST Designer computes optimal solutions for the network extension, taking into account technical, financial and QoS constraints.

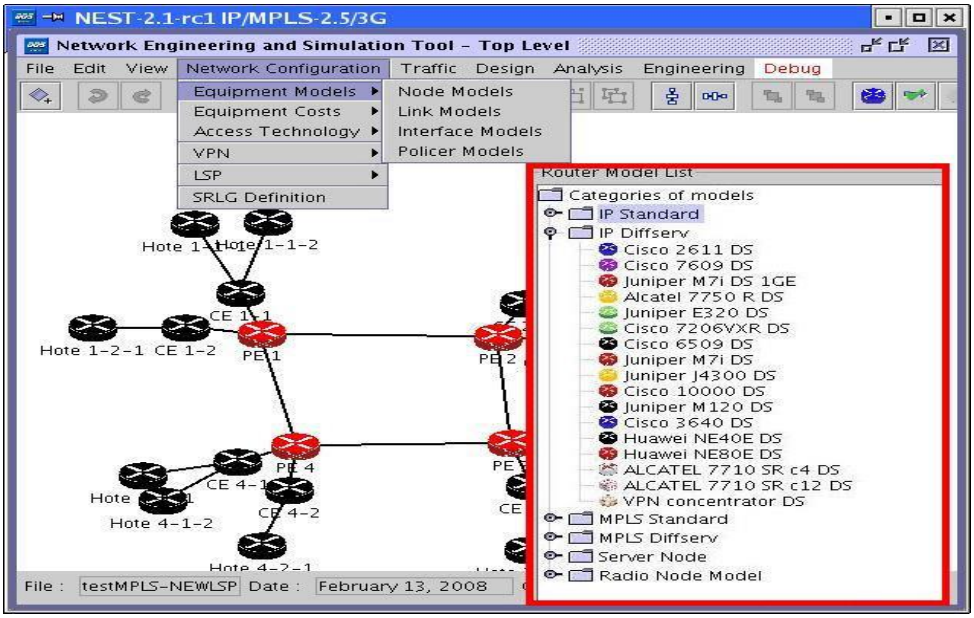

Figure 1: *NEST Designer Module*

- **NEST SDH/DWDM** provides a simulation environment for SDH/DWDM optical transport Networks. NEST SDH/DWDM permits users to reproduce a real operator network in its operational state, powered by automated circuit routing algorithms, rerouting algorithms, failure healing processes and cost analysis.

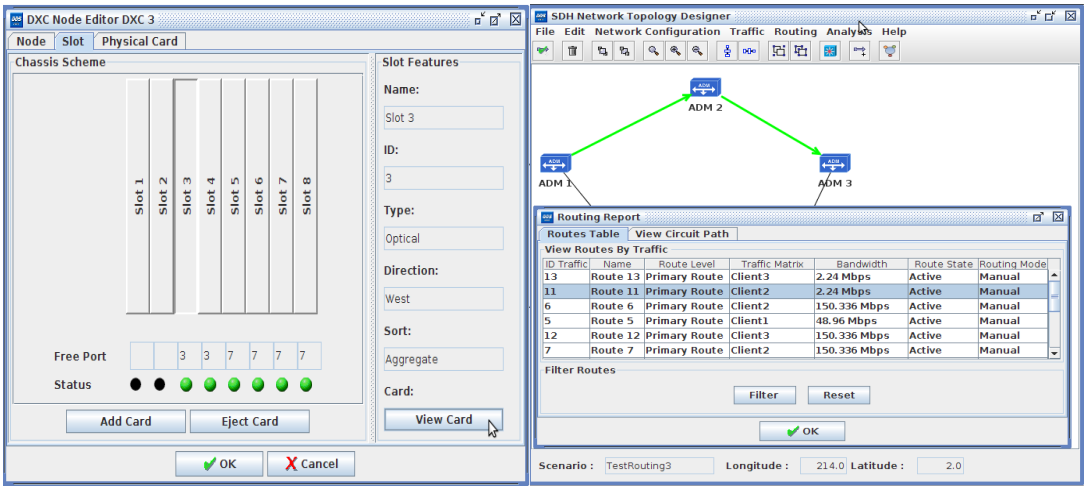

Figure 2: *NEST SDH/DWDM Module*

- **NEST FTTx** provides a simulation and planning environment for FTTx optical access Networks. NEST FTTx enables users to reproduce a real operator network architecture powered by GIS maps, interconnected equipment, end-to-end fiber connections from the customer to the central office, existing duct installation, new ducts planning, concentrator optimal placement, civil work optimization and global cost analysis.

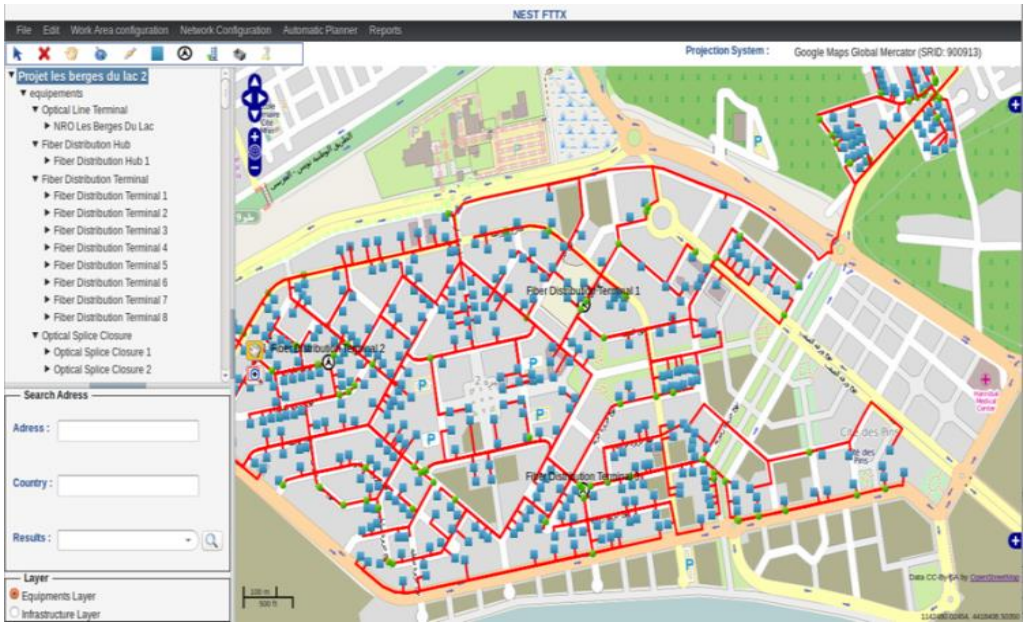

Figure 3: *NEST FTTx Module*

- **NEST NGN** provides a unified framework for wired (e.g. IP, MPLS, ADSL, Metro Ethernet) and the core network (SGSN, GGSN, mobile DNS,…) of wireless networks (e.g. 2G, 3G). It offers to network planning teams an unbalanced, design and traffic engineering capabilities. It supports several protocols (OSPF, ISIS, BGP, MPLS, etc.), inter-AS routing policies, multicast, VPNs and many other features. It supplies a global simulation kernel able to handle several million user multimedia traffic flows and various decision support tools such as resilience and bottleneck analyses, IP routing optimizations, MPLS traffic engineering and LSP optimal placement and protection[2,3,4,5,6,7].

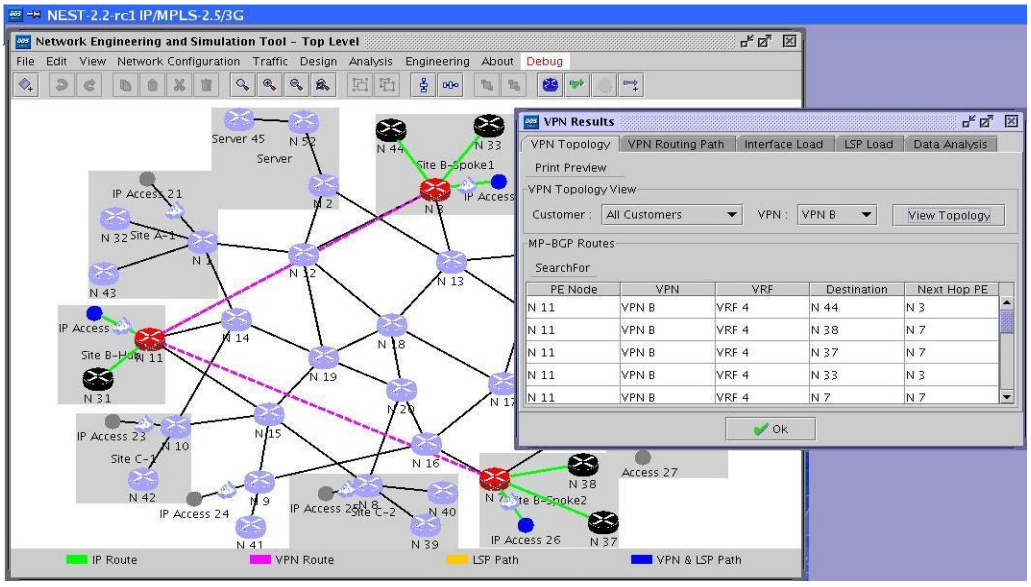

Figure 4: *NEST NGN Module*

- **NEST Entreprise** is a workbench that allows autonomic control of IP/MPLS networks. It provides the user with powerful tools for supervision, dynamic analysis (real-time) and intelligent control of networks.

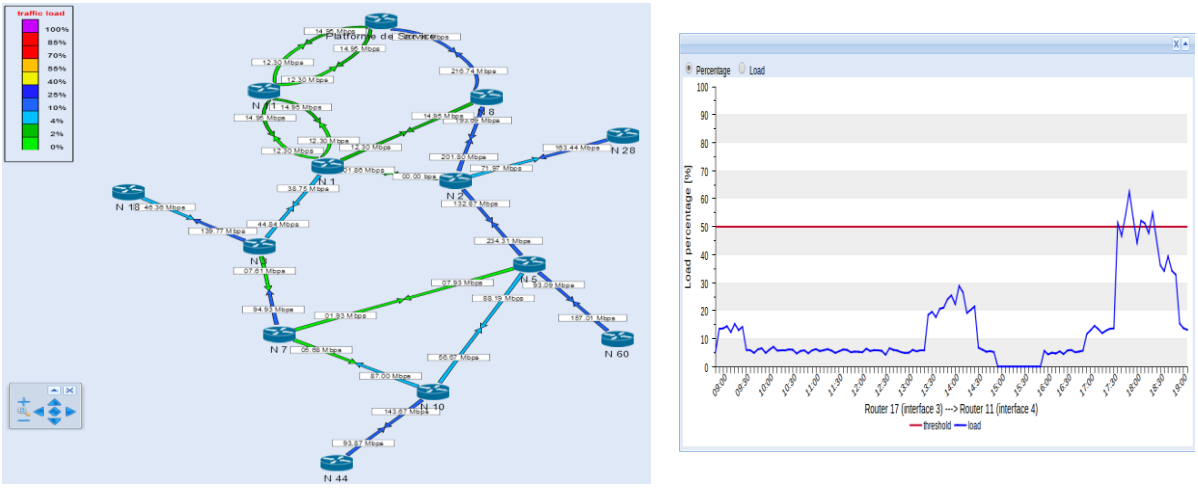

Figure 5: *NEST Entreprise*

NEST Entreprise coupled with NEST NGN suite provides a powerful integrated workbench for supervision & intelligent control of IP/MPLS Networks. NEST Entreprise relies on simulations in the loop and autonomic control algorithms powered by real-time measurements.

NEST Entreprise is a workbench that allows autonomic control of IP/MPLS networks. It provides the user with powerful tools for supervision, dynamic analysis (real-time) and intelligent control of networks.

- NEST Entreprise provides network managers with powerful dashboards for a complete control over carrier grade infrastructures.
- NEST Entreprise provides more than traditional supervision functionalities by enabling the detection of QoS degradation causes.
- NEST Entreprise integrates a real-time simulation kernel that provides network performance analysis and QoS estimations.

NEST Entreprise embodies several functionalities:

- **Discovery Engine**:
  - Discovers IP devices (routers, switches, servers),
  - Discovers MPLS networks (LER, LSR, LSP, Tunnel, …),
  - Discovers VPNs from Layer 2 and Layer 3.
- **Real Time Measurements:**
  - Provides monitoring of mission-critical components,
  - End-To-End traffic measurements (SNMP, NetFlow,...),
  - Bandwidth threshold alarms on links/interfaces,
  - Device failure alarms & reporting.

- **Traffic Matrix Estimation and trending**
  - IP and MPLS Traffic matrix estimation,
  - Trending of both IP and MPLS traffic.
- **Simulation in the loop Kernel**
  - Simulation in real time for the entire network,
  - Real-time QoS estimation and analysis.
- **Global Network Controler**
  - Dynamic optimization and decision support algorithms,
  - Coupled with the simulation kernel, the algorithms allow intelligent control decisions for the global network.

3 DESCRIPTION OF CORE EMULATOR

3.1 Overview

The Common Open Research Emulator (CORE) 0 is a tool for emulating networks on one or more machines. You can connect these emulated networks to live networks. CORE consists of a GUI for drawing topologies of lightweight virtual machines, and Python modules for scripting network emulation.

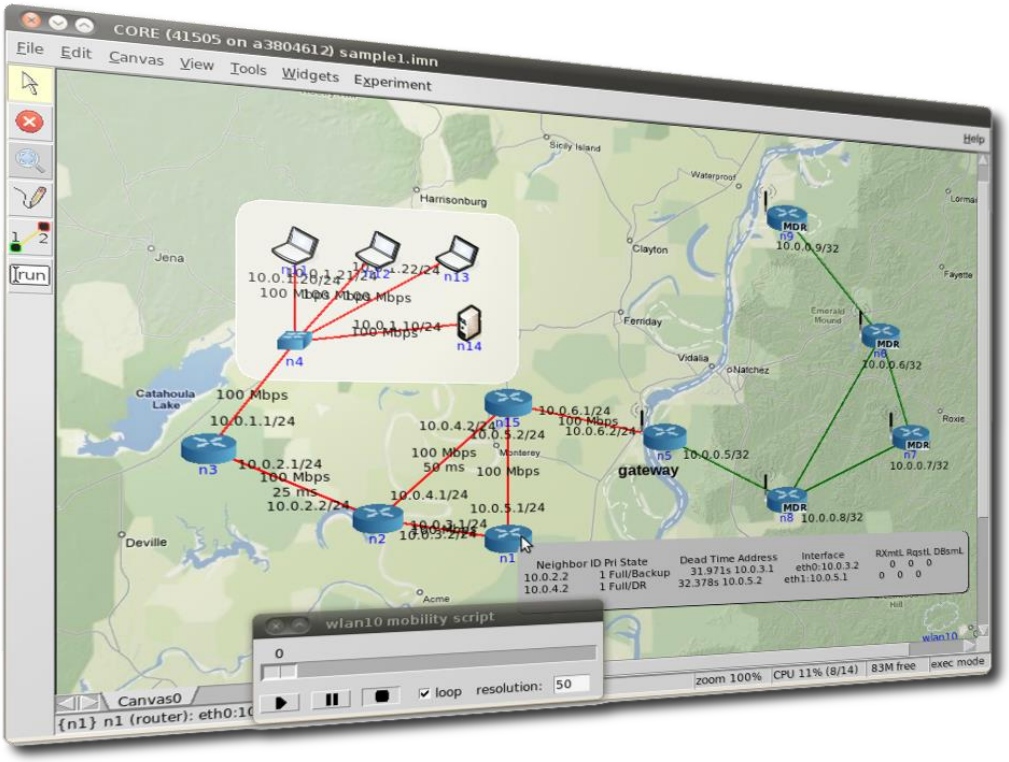

Figure 6: *Core screenshot sample*

CORE has been developed by a Network Technology research group that is part of the Boeing Research and Technology division. The Naval Research Laboratory is supporting further development of this open source project.

CORE is typically used for network and protocol research, demonstrations, application and platform testing, evaluating networking scenarios, security studies, and increasing the size of physical test networks.

3.2 How does it work?

The main components of CORE are shown in CORE Architecture. A CORE daemon (backend) manages emulation sessions. It builds emulated networks using kernel virtualization for virtual nodes and some form of bridging and packet manipulation for virtual networks. The nodes and networks come together via interfaces installed on nodes.

CORE emulator virtual nodes are:

- Router: by default runs Quagga OSPFV2 and OSPFV3 routing to forward packets,
- Host: emulated server machine having a default route, runs SSH server,
- PC: basic emulated machine having a default route, runs no processes by default,
- PRouter: Physical router connected to the emulated environment. Currently, when any emulated node is linked to a physical node, a GRE tunnelling interface will be created on the physical node and used to tunnel traffic to and from the emulated world,
- Hub: Ethernet hub forwards incoming packet to every connected node
- Switch: the Ethernet switch intelligently forwards incoming packets to attached host using an Ethernet address hash table,
- Wireless LAN: when routers are connected to this WLAN node, they join a wireless network; the WLAN node typically controls connectivity between attached wireless nodes based on the distance between them,
- RJ45: with the RJ45 Physical Interface Tool, emulated nodes can be linked to real physical interfaces on the Linux or FreeBSD machine; using this tool, real networks and devices can be physically connected to the live-running emulation,
- Tunnel: the Tunnel Tool allows connecting together more than one CORE emulation using GRE tunnels (Tunnel Tool).

Remark: It is possible to configure additional services that runs on a particular Node. For example on a host we can configure an apache Tomcat server that host a web application. We can run also other protocols on routers if needed.

The CORE daemon is controlled via the graphical user interface, the CORE GUI (frontend). The daemon uses Python modules that can be imported directly by Python scripts. The GUI and the daemon communicate using a custom, asynchronous, sockets-based API, known as the CORE API. The dashed line in the figure notionally depicts the user-space and kernel-space separation. The components the user interacts with are coloured blue: GUI, scripts, or command-line tools.

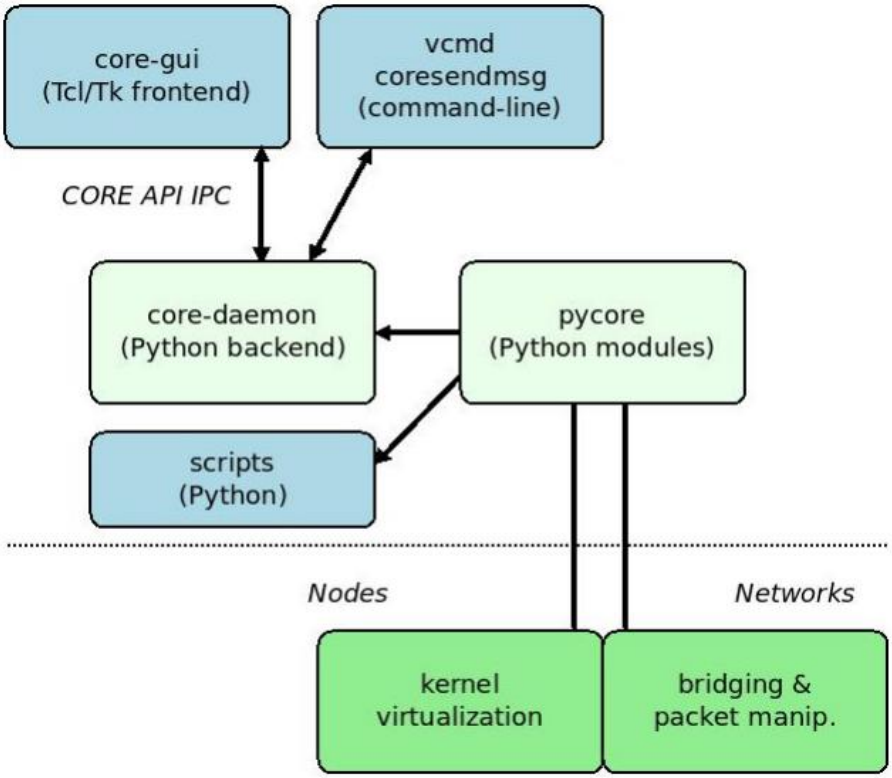

Figure 7: *Core architecture*

The system is modular to allow mixing different components. The virtual networks component, for example, can be realized with other network simulators and emulators, such as ns-3 and EMANE. Different types of kernel virtualization are supported.

Another example is how a session can be designed and started using the GUI, and continue to run in "headless" operation with the GUI closed. The CORE API is sockets based, to allow the possibility of running different components on different physical machines.

A CORE node is a lightweight virtual machine. The CORE framework runs on Linux and FreeBSD systems. The primary platform used for development is Linux.

- Linux CORE uses Linux network namespace virtualization to build virtual nodes, and ties them together with virtual networks using Linux Ethernet bridging.
- FreeBSD CORE uses jails [14] with a network stack virtualization kernel option to build virtual nodes, and ties them together with virtual networks using BSD's Netgraph system[15].

3.3 Key features

Core Emulator provides a network Lab in a box. This lab is efficient and scalable. The user have access to easy-to-use and highly customizable environment. For better performances, Multiple Core instances can be run over multiple servers.

Core Emulator, within its architecture, proposes centralized configuration and control. A key feature of Core emulator is the ability to run protocols and applications without modifying them.

CORE's emulated networks run in real time, so they can be connected to live physical ones. The RJ45 tool and the Tunnel tool help in connecting to the real world by creating virtual links. Using these links packets can be forwarded from a real router to an emulated one.

This feature can be used to extend the size of a real network within an experiment. We can imagine for the purpose of an experimentation combining a real 10 nodes network with a 40 nodes network to achieve a 50 nodes network.

4 FIRST APPROACH: ANALYTICAL SIMULATION

4.1 Introduction

NEST allows an exact modelling of the underlying network (routers, links, data centres, protocols, etc.). Thus what-if analysis can be performed for several adverse events (link failures, router failures, server failures, congestion, etc.).

As stated before, the user could, in a simulation environment, compare and validate several traffic engineering strategies and adopt the best suiting one.

In this approach we will create an environment that can run the simulation of the overlay network. NEST NGN will allow us to create the underlying topology infrastructure (Routers, Links, Servers, etc.). A distributed application (from the use case Web Service on the cloud from [23]) will be mapped onto the overlay network by adding its own proxies.

The simulation process of NEST Entreprise, will represent the evolution of the global network over a period of time. It will focus on overlay network performances by highlighting the metrics related to the distributed application. This evolution will be based on stochastic events generation within the simulation process.

The simulation process will take into account the evolution of the metrics of each link of the overlay network to compute, at each step, the best paths. It will also update the used paths and bandwidth consumption in the underlying network.

This approach has two main benefits. It will allow us to perform large scale experimentation that cannot be performed in real life due to feasibility and economic cost issues. It will also allow us to stress the overlay network under certain conditions to validate the self-optimizing and self-healing properties of the overlay network. This cannot be done in the real Internet.

The DAaaS is a distributed system comprised of several complex components. The level of difficulty to model this system won't be possible within the project deadlines. Thus we will restrict the experimentation of this approach to only use case 1 [23].

At a high level, the PANACEA overlay system is composed of three components. The first one is responsible of monitoring the network performances, the second one is responsible of computing the shortest paths for each communication the third one is the proxy system responsible of capturing the packets and adding the PANACEA header to change packet routing.

In this approach we will simulate the second one (computing shortest path according to some metrics) and the third one (changing the header of packets and modifying their routing).

4.2 Required modifications

For the purposes of this experiment we will use two different modules: NEST NGN and NEST Entreprise. NEST NGN will be used for modelling and designing overlay networks. NEST Entreprise will be used for the simulation of the evolution of overlay network performances during a time frame.

4.2.1 Required modifications for NEST NGN

To simulate overlay networks we have to introduce the overlay network concept into NEST NGN. To achieve this goal the following features will be added:

- **Modelling of Overlay Networks**: Overlay networks can be modelled using a variety of mechanisms. For example, each Overlay could be implemented as a virtual LAN (VLAN), Virtual Private Networks (VPNs), etc. Overlays can also be implemented using two networks, a physical underlay network and a virtual overlay one. This overlay networking technique has been widely deployed in the wireless LAN (WLAN) industry for more than a decade, but its application to data centres networks is relatively new. It is being standardized in various forums such as the Internet Engineering Task Force (IETF) through the Network Virtualization Overlays (NVO3) working group. In that case, the role of the physical underlay network is to provide an "IP fabric". This fabric is responsible to provide unicast IP connectivity from any physical device (server, storage device, router, or switch) to any other physical device. We shall use the later configuration (physical + virtual) for modelling overlay networks in NEST Environment.
- **Add mapping procedures** between overlay and underlay network: The vRouters running in the hypervisors of the virtualized servers create a virtual overlay network on top of the physical underlay network using a mesh of dynamic "tunnels" among themselves. These overlay tunnels can be MPLS over GRE/UDP tunnels or VXLAN tunnels. For NEST we shall use MPLS tunnel based communications between overlay nodes. This choice is also sustained by the ability to use "explicit routing" to take control of the used paths for each communication between overlay nodes.
- **Adapt the Routing Kernel**: To take into account overlay networks. The admitted traffics will be propagated alongside the specific routing paths (tunnels) used by the overlay. The overhead for the admitted traffics will also be taken into account to reflect accurate bandwidth consumption.

4.2.2 Required modifications for NEST Entreprise

In the NEST Entreprise module, we will develop the following features:

- **Create an adverse event editor**: this editor will allow the scheduling of adverse events that include failures (Routers, Links, Servers, etc.) and congestion. The idea is twofold. From one side the user can choose an equipment a starting date and a duration for a failure. On the other side the user can programme a congestion by choosing a link a date and an overhead describing how much extra traffic have to be generated according to the mean behaviour.
- **Create a network simulator**: that also simulates the behaviour of an overlay network, computes network performances and takes into account adverse events. This simulator will compute performances for the underlying network and reflect those performances metrics on the overlay layer.
- **Create a graphical representation of the overlay network**: this view allows the visualization of the paths used by the overlay network. It can be used to follow the evolution of an overlay network during a time frame.

4.3 Experimentation Use case 1 : Web Service

4.3.1 Experiment procedure

For this use case, the scenario to be used shall include a WAN to which several clouds, hosting a web application, are connected. We will implement the relevant (according to the simulation goals) components of the TPC framework web application.

Two traffic matrices will be considered, one representing background traffic, and another representing the connections (in thousands) between cloud platforms generated by client requests (distributed aspect of the application). The overlay network will be created between the cloud platforms. The overlay is aimed at optimizing the network performances of inter-cloud communications.

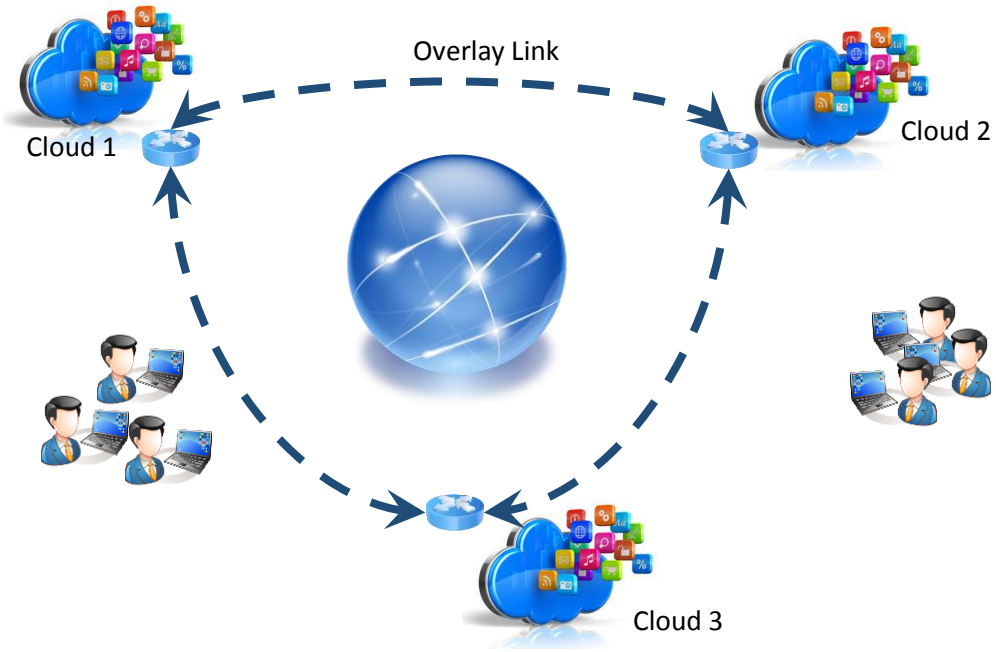

Figure 8: *Web Application deployed over several cloud platform*

During the simulation, the traffic of the aforementioned matrices will vary in time based on a given distribution. For the background traffic variation, we will set a standard distribution for all the experiments. For the traffic matrix representing the requests generated by the web application we will set up three profiles (Low, Medium and Heavy). Those levels will guide the process of validation and performance evaluation with and without Overlay.

At some point of the simulation, some links will experience congestion due to the volume of traffic traversing them. If this leads to a negative impact on the overlay links, re-optimization will be carried out by the simulation process to enhance the QoS performances of these overlay links.

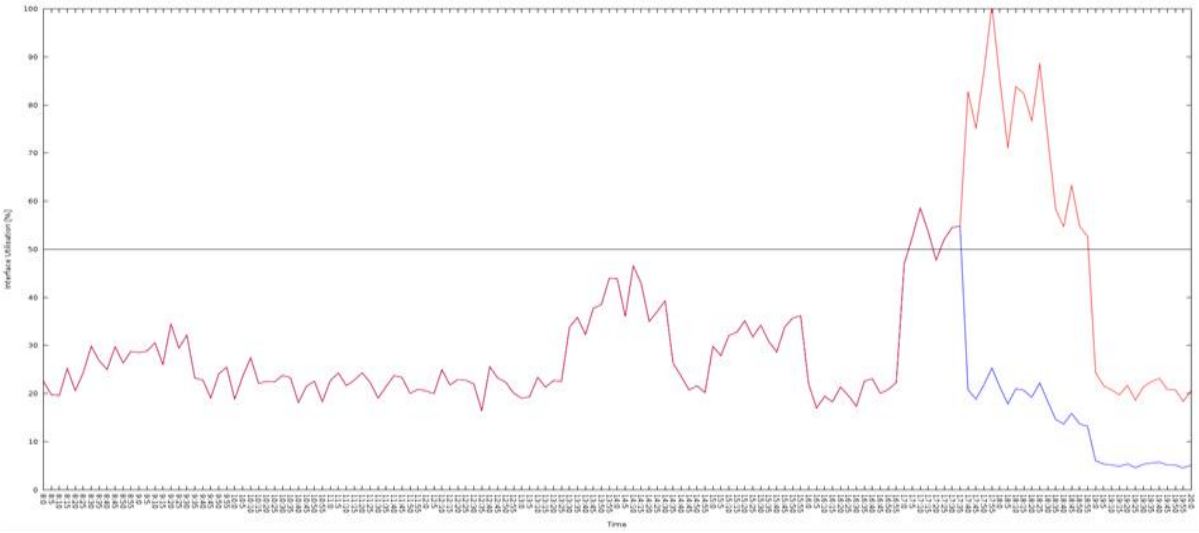

Figure 9: *Sample traffic Pattern used for background traffic*

Furthermore, various failures will be programmed. Whenever a failure occurs, the simulation process will compute backup paths for implicated overlay links.

4.3.2 Impact and metrics

At each step, the simulation kernel will compute the latency and the loss rate metrics for each overlay link and for each connection towards the web application.

By defining some SLA for the Web application (max latency, max loss rate), we can infer the percentage of satisfied clients (within SLA).

This statistic will also be used to compare the scenario with the overlay and without the overlay.

To increase the accuracy of the experiments, we will use for each one three traffic profiles (low / normal / heavy).

5 SECOND APPROACH: HYBRID SIMULATION

5.1 Introduction

In this approach we combine simulation and emulation in the same experimentation. The idea behind this is to use the real **PON** along with a **real application**. The underlying network and the adverse events are still simulated by NEST. The simulation process **drives** the emulation **by injecting** traffic and events in the emulation platform.

This concept offers the possibility to stress-test a real application under chosen conditions using the real overlay system. It will help assert the resilience of the network and the impact on the application performance.

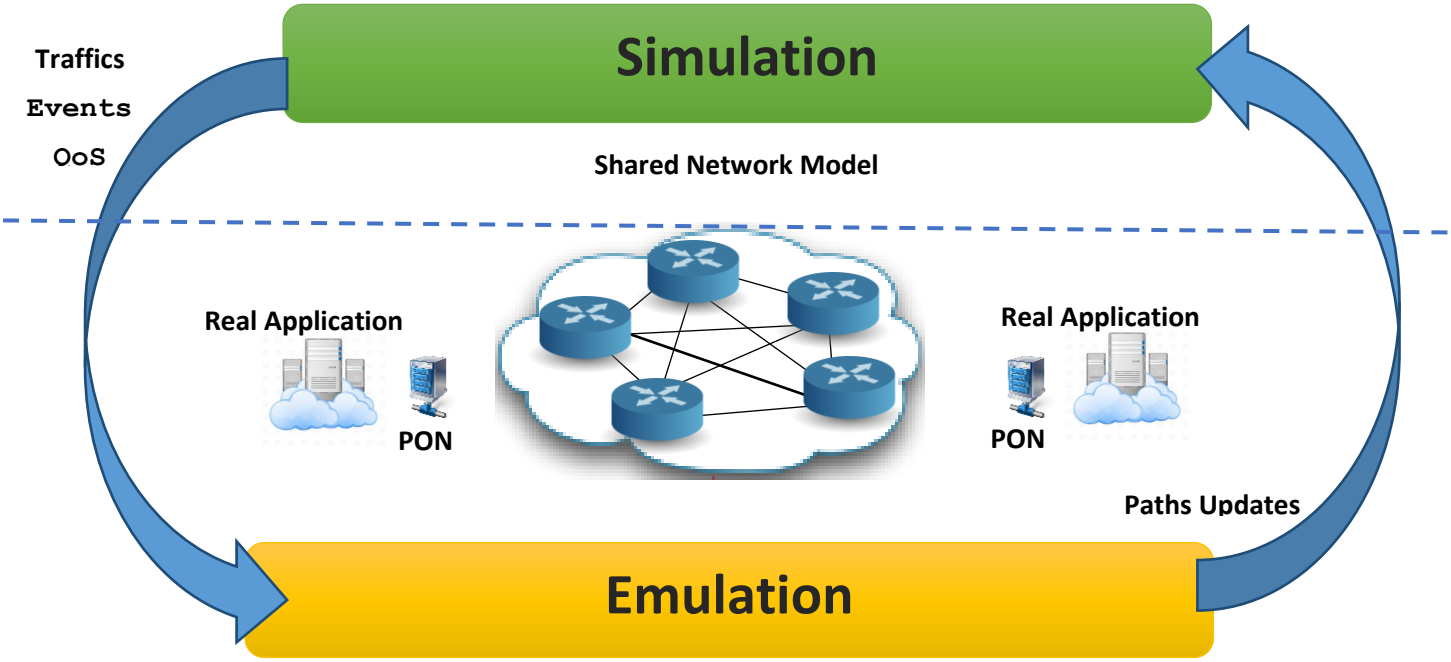

Figure 10: *Controlling the emulation process from NEST simulator*

Both overlay Layer and underlying network will be modelled in emulation and simulation process.

We will consider in this chapter two experimentations. One with the web service from the TPC framework. The second one will be with the DAaaS framework. In both cases our main concern is the overlay performances. In this context we consider the first experiment, due to its inherent simplicity, as a **first step validation** and the second one as the **real benchmark** for the PANACEA Overlay system validation.

5.2 Experimentation with Use case 1 : Web Service

5.2.1 Experiment procedure

In this use case, we consider a typical tiered e-commerce web application. The web application consists of the web servers that host the application logic and also provides a user interface to the clients. These application servers are connected to the database that stores all the information on the inventory, users, transactions, etc. We use the specific technologies of the Apache Tomcat web server[1], the MySQL database server[2] implemented in Java, and the TPC benchmark[3] in order to emulate web clients and requests.

Several virtual machines will be configured. The web application will be spread over several VMs dispatched on multiple hosts. Each host will have its own overlay agent (on a particular VM).

NEST simulation process will inject connections toward the web application according to a pattern (distribution of inter-arrival connection over the time). Different configurations can be tested (low, normal and high load).

NEST will evaluate the QoS of all equipment's in the underlying layer and set the level of performance (delay, loss rate, bandwidth,…) accordingly in the emulator equipment's. The PON can after that perform the path search as in the real internet. The result will be to choose the best path for the overlay link.

When an overlay path is changed due to re-optimization by the PON, NEST synchronize accordingly the path followed by packets in the underlying network. In fact we will use the explicit routing feature that allows us to fix directly the path of an MPLS tunnel. So when the overlay change the path of a communication we trigger an action that change accordingly the MPLS tunnel path.

The same process is applied for failures. The simulation process will change the status of a router, link, or an interface and will trigger the same action within the Core Emulator. The overlay system after detecting the failure will reassign traffic to another path. This change will also be applied within the simulation process.

5.2.2 Impact and metrics

So as to improve inter-cloud communication in a multi-cloud setting, we can take advantage of the PANACEA overlay network system to monitor internet paths between cloud sites. For this purpose, implementing routing agents in each of the sites will allow integrating the overlay network within the experiment.

The expected outcome would be not only a successful operation of the web services when the overlay network is active, but also an acknowledged validation of improvements brought by this network as compared to a non-overlay situation. This is done based on relevant evaluation metrics which are typical network metrics such as packet latency, loss rate and available bandwidth.

[1] http://tomcat.apache.org/

[2] http://www.mysql.com/

[3] http://www.tpc.org/

5.3 Experimentation with Use case 2 : DAaaS Service

5.3.1 Experiment procedure

In this case, we will be considering the same DAaaS environment as described in D4.1 [24]. We will apply it over a multi-cloud setting. The DAaaS logic will be implemented over multiple VMs. Each VM can be located in a different cloud.

For this experiment NEST don't have to inject traffics onto the emulation platform. Overlay traffics are only creatd by the DAaaS components. So for this case NEST simulator will be responsible of setting the QoS Levels and update the routing paths in the underlying network. All other processes will be held at application and overlay levels.

We will apply the same perturbation scheme (congestion, failures) over a period of time to stress-test the DAaaS logic. The goal will be again to see if the PON is capable of restoring acceptable levels of performance. To prove that, we will compute the performances of the communications with and without the overlay system.

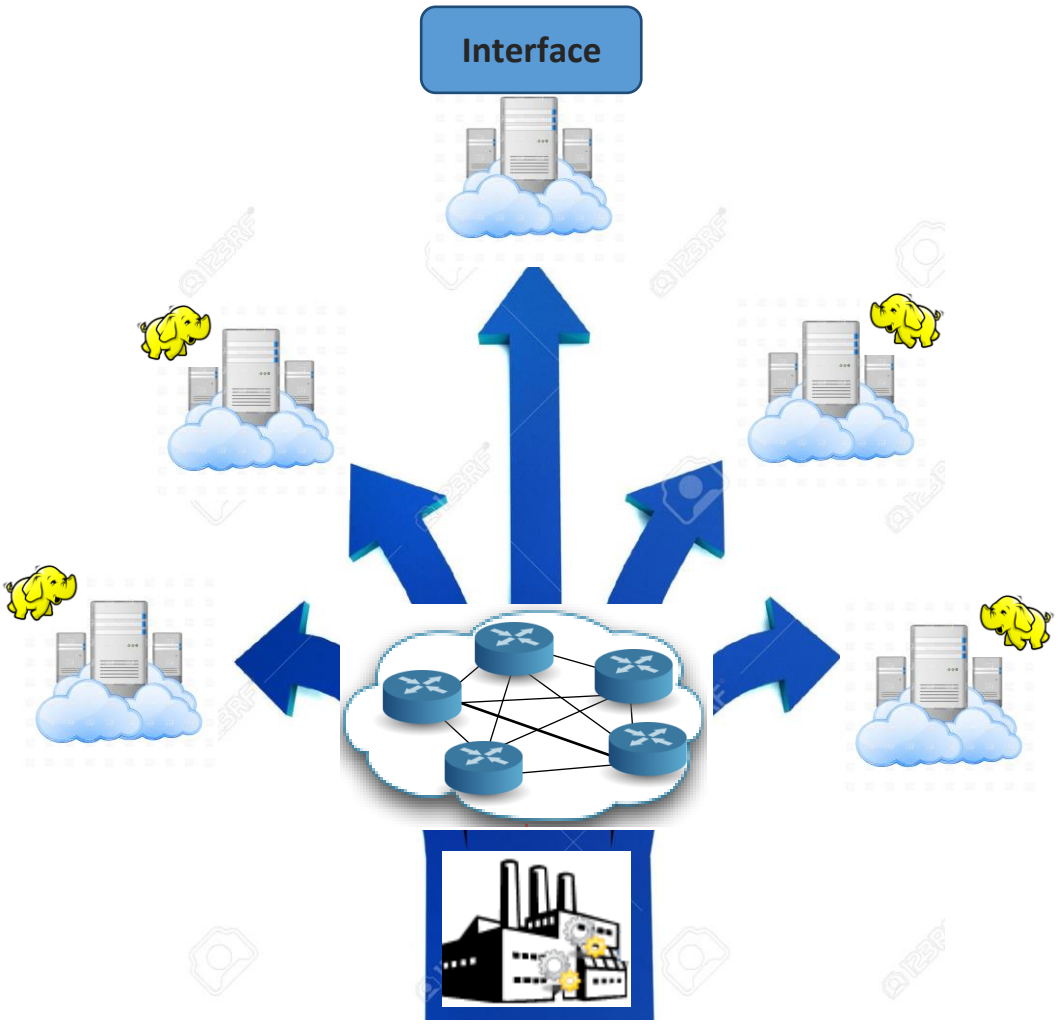

Figure 11: *Sensor's Data sent to worker nodes through overlay links*

5.3.2 Impact and metrics

In this scenario the main target will be to assess that the overlay network improves the communication between the geographically separated components of the DAaaS platform.

If possible the experiment will use real data sets collected from some factories managed by Atos.

All inter-cloud communications over the Internet are vulnerable to congestion caused by background traffic and failures. The experiment will showcase the improvement of such multi-cloud communication under realistic conditions. The same metrics, used for web application use case, will be reused packet latency, loss rate and available bandwidth.

An additional metric that could be evaluated is the total duration of the DAaaS process when spanned over multiple sites.

In all cases the improvements will be acknowledged by comparing the metrics with and without the overlay optimization.

6 CONCLUSIONS

We covered in this document the methodology that will be used for demonstrating the capabilities of the overlay simulation module. This module will help validate the performances of the PANACEA overlay network system.

To achieve that, two approaches are proposed. The first one rely on pure simulation that can be considered as a first step validation. The second one combine simulation and emulation (Hybrid simulation). This second approach will allow us to use the PON and a real application to achieve the experiments.

We are considering for our experiment the two selected uses cases from deliverable D4.1. We shall therefore use the Multi-tier E-Commerce application described in [23] for a first stage validation, and then use the Hadoop Map/Reduce scenario described in [24] for the validation of the PON.

REFERENCES

[1] Comparison of CORE Network Emulation Platforms, Proceedings of IEEE MILCOM Conference, 2010, pp.864-869.

[2] JM. GARCIA HA-DUC Le, D. GAUCHARD, O. BRUN, Modélisation Analytique Du Protocole Tcp New Reno Et Etude Comparative Avec Le Simulateur Ns (Network Simulator), Congrès RIF'03, Hanoi, Vietnam, 10, 13 February 2003

[3] JM. GARCIA, A. RACHDI et O. BRUN, Optimal LSP Placement with QoS Constraints in DiffServ/MPLS Networks, 18th International Teletraffic Congress (ITC'18), Haus am Köllnischen Park, Berlin, Allemagne, 31 august - 5 September 2003.

[4] F. CHAUVET & JM. GARCIA, Optimisation de Coeur de Réseau IP/DiffServ/MPLS, Conférence invitée, Algotel 2003, Banyuls sur mer, 12-14 may 2003.

[5] P. BACQUET, O. BRUN, J.M. GARCIA, T. MONTEIL, P. PASCAL & S. RICHARD, Telecommunication Network Modeling and Planning Tool on ASP clusters, International Conference on Computational Science 2003, (ICCS'2003), Saint Petersbourg, Russie et Melbourne, Australie, 2 - 4 June 2003.

[6] O. BRUN, JM. GARCIA, Resource Allocation in Communication Networks 5th International Conference on High-Speed Networks and Multimedia Communications, Jeju Island, Corée, 3 - 5 Jully 2002.

[7] JM. GARCIA, P. BACQUET Technique Avancées de Simulation ECOTEL 2002, Golf Juan, December 2002

[8] O. BRUN, JM. GARCIA, Modélisation Dynamique du Trafic dans les Réseaux MultiServices (MSR'2001), Toulouse, September 2001, Juanole G. et Valette R. Eds, Hermes.

[9] M.Z. BEN HAMOUDA, O. BRUN et JM GARCIA, Integration of Equipment Constraints in the Network Topology Design Process, 9th Int. Symposium on Communications and Information Technologies, Incheon, Korea, 2009.

[10] O. BRUN, A. RACHDI et JM GARCIA, Access network design with capacity-dependent costs, ValueTools'2008, Athenes, Grèce, 20-24 October 2008.

[14] https://www.freebsd.org/doc/handbook/jails.html

[15] https://www.freebsd.org/cgi/man.cgi?netgraph%284%29

[16] http://www.isi.edu/nsnam/ns/

[17] http://www.j-sim.org/

[18] http://www.ssfnet.org/

[19] http://www.omnetpp.org/

[20] http://www.freepastry.org/FreePastry/

[21] http://ants.etse.urv.es/planet/planetsim/

[22] http://www.macesystems.org/macedon/index.html

[23] Deliverable D4.1 (Use Cases), Section 3.1, page 17

[24] Deliverable D4.1 (Use Cases), Section 3.2, page 20