Computer Communications 80 (2016) 33–44

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

NEmu: A distributed testbed for the virtualization of dynamic, fixed and mobile networks

Vincent Autefage, Damien Magoni*

University of Bordeaux, LaBRI, 351, Cours de la Liberation 33400 Talence, France

ARTICLE INFO

Article history:
Received 3 March 2015
Revised 14 January 2016
Accepted 22 January 2016
Available online 1 February 2016

Keywords:
Emulation
Mobile
Network
Testbed
Virtualization

ABSTRACT

Experimentation is typically the last step before launching a network application on a large production scale. However, it is often difficult to gather enough hardware resources for experimenting with a reasonably sized distributed application inside a controlled environment. Virtualization is thus a handy technique for creating such an experimentation testbed. We propose a tool called *NEmu* designed to create virtual dynamic networks by using emulation for testing and evaluating prototypes of networked or distributed applications with a complete control over the network topology and link parameters. *NEmu* leverages system emulators such as QEMU for virtualizing the hosts and the routers. It uses *vnd* for virtualizing components such as links and switches. In addition, *NEmu* allows users to create such customized topologies with limited hardware resources and without any administrative rights. We validate *NEmu* by replicating two network experiments and by showing that *NEmu* gives results very similar to the original ones.

1. Introduction

Experimentation is important to realistically and accurately test and evaluate network applications. Experimentation on algorithms is usually made by *simulation*. This technique is available through well known software like *ns* [1] or *OMNeT++* [2] and enables to evaluate the efficiency and the scalability of algorithms or protocols. Experimentation on a real program, i.e. implementation, is different to the extent that it is more focused on execution time, processor usage, memory consumption, network properties, etc.

It can be a difficult task when trying to experiment with a network application involving dozens of machines or more. Moreover, the mobility or the dynamics of scenarios can drastically increase the difficulty of experimentation. Using the Internet as a test bed is impractical as no parameters can be controlled. Setting up a hardware test bed is expensive and cumbersome. Furthermore, network applications can have very different ways of connecting hosts to each others and changing the network topology and network parameters of a hardware test bed is time consuming and error prone. Virtualization techniques for creating such an experimentation test bed can save resources and ease manipulations. It is a proven method for reducing the equipment and space costs as well as the energy consumption of using physical hosts [3].

Our solution to overcome the above hardware constraints is thus to build a test bed able to set up virtualized networks. A virtual network uses virtual machines instead of physical hosts and connects them with virtual links in order to build a virtual network topology. The virtual machines of a virtual network can be hosted on one or several physical hosts depending on the number of virtual machines needed and the resources capacities of the physical ones.

We propose a tool designed to create virtual networks for testing and evaluating prototypes of applications on the top of static, dynamic or mobile networks with a complete control over the network topology and link properties (bandwidth, delay, bit error rate, etc.) and the mobility of nodes. The goal of our tool is to enable the creation of reasonably sized virtual networks while minimizing the number of necessary physical hosts and network equipment needed. It can build host-based overlay networks by using emulators such as QEMU [4]. We have called our tool *NEmu* which stands for *Network Emulator for mobile universes* because it is able to create both fixed and mobile emulated networks. It is also a tribute to the name of the QEMU software which is a powerful machine emulator heavily used by *NEmu*. The contributions of our work are as follows:

- A detailed description of our *NEmu* software which is able to manage a distributed set of virtual nodes and links for emulating any arbitrary static, dynamic or mobile network topology (Section 2).

* Corresponding author. Tel.: +33 5 4000 3540; fax: +33 5 4000 6669.
E-mail addresses: autefage@labri.fr (V. Autefage), magoni@labri.fr (D. Magoni).

http://dx.doi.org/10.1016/j.comcom.2016.01.005
0140-3664/

- A detailed description of our *nemo* tool which implements the mobility inside *NEmu* and enables to create a complete autonomous mobile network by following a predefined scenario (Section 4).
- Two validation experiments by replicating the Mosh experiment originally done with *Mininet* [5] (Section 5.1), and the AMiRALE experiment originally done with JBotSim [6] (Section 5.2).
- A state of the art on related and previous work targeted at networking emulation and a comparison of the features provided by *NEmu* with the ones offered by similar alternative virtual networking testbeds (Section 6).

This paper is an extended and revised version of our previous work published in [6].

2. Description of *NEmu*

2.1. Overall design

NEmu is a python program consisting of 6000 lines of code which allows to build a dynamic and distributed virtual network infrastructure.[1] It is based on the concept of *Network Virtualization Environment* (NVE) introduced by Chowdhury and Boutaba in [7]. The main characteristic of a NVE is that it hosts multiple *Virtual Networks* (VN) that are firstly not aware of one another, and that are secondly completely independent of each other. A VN is a set of *virtual nodes* connected by *virtual links* in order to form a virtual topology. *NEmu* provides the possibility of creating several virtual network topologies with the central property that a VN is strictly disjoint from another in order to ensure the integrity of each VN.

Thus, *NEmu* integrates characteristics that are fundamental to a NVE: First, the *flexibility and heterogeneity* allows the user to construct a customized topology, with custom virtual nodes and virtual links. The *scalability* allows different virtual nodes to be hosted by different physical hosts in order to avoid limitations of a unique physical machine. The *isolation* decouples the different virtual networks which run on the same infrastructure. It also guaranties a strict separation between the host and the virtual networks. The *stability* ensures that faults in a virtual network would not affect another one. The *manageability* ensures that the virtual network and the physical infrastructure are completely independent. Therefore, a VN created on an infrastructure *A* can be deployed on another infrastructure *B*. The *legacy support* ensures that the NVE can emulate former devices and architectures. Finally, the *programmability* provides some optional network services to simplify the use of the virtual network (such as DHCP, DNS, etc.). It also implies that the user can develop and integrate his own additional services.

In addition, *NEmu* includes four important extra properties:

- The *accessibility* which means that *NEmu* can be fully executed without any administrative rights on the physical infrastructure. Indeed, the major part of public infrastructures, like universities and laboratories, does not provide administrative access to their users in order to ensure the security and the integrity of the whole domain. Therefore, the user execution would allow most people to use *NEmu* freely.
- The *dynamicity* of the topology enables node hot-connections which means that a virtual node can join or leave the topology dynamically without perturbing the overall virtual network.
- The *mobility* of nodes provides a way to create a self defined topology evolution through time and space. In other words, it is possible to create an autonomous connectivity scenario.
- The *community aspect* of the virtual network provides the possibility for several people to supply virtual sub-networks in order to build a community network like the Internet is.

2.2. Network elements

NEmu is a distributed virtual network environment which allows users to create arbitrary and dynamic topologies. To this end *NEmu* is based on different building blocks. *NEmu* uses *virtual nodes* connected by *virtual links* in order to create a virtual network topology. A virtual topology can be hosted by one or several physical hosts. The part of the virtual topology laying on a given physical host represents a *NEmu session* which is configured by the *NEmu manager*.

2.2.1. Virtual node

A *virtual node* for *NEmu* is an emulated machine that requires a hard disk *image* to work. This image is typically provided as a regular file on the physical host machine. Two types of *virtual nodes* currently exist in *NEmu*:

- A `VHost` is a virtual *host* machine (i.e., end-user terminal) on which the hardware properties and the operating system can be fully configured by the user.
- A `VRouter` is a virtual *router* directly configured by *NEmu* and provides ready-to-use network services.

Each virtual node uses a *virtual storage* which can be either a real media (cdrom, hard drive, etc.), a *raw* file or a host directory. A raw file can be privately dedicated or shared by several other virtual nodes. A modification of a shared file by one virtual machine will affect the others which may be troublesome if the file contains the operating system. To solve the problem, *NEmu* uses *Copy-on-Write* (CoW) operations on the original file. A *CoW file* (also known as a *sparse file*) only stores the differences with its original file. The advantage, compared to a regular copy, is that the CoW file is much smaller. In addition, *NEmu* can use a regular directory on the physical host (without building a CoW), as a storage media in four different ways:

- by making a *Sparse* file which only stores the differences with its original file;
- by making a *Squash* file system which is a read-only raw image;
- by using a *FAT16* emulated interface which enables a direct access to a host's file system;
- by using a *Virtio* interface [8] which also enables a direct access to a host's file system;
- by using a *Network Block Device* which enables a virtual node to remotely access to a block device through the real IP network [9];
- by using a *SSH* tunnel which enables a virtual node to remotely access to a block device through a secured connection.

As said before, a `VHost` needs a disk image which must be supplied by the user. This image must be prepared prior to creating the virtual network. Furthermore, one image can be used by many `VHosts` by using *sparse files*. *NEmu* provides a network topology visualization option by processing its topology data file through *Graphviz* [10].

A `VRouter` is directly configured by *NEmu* and provides several services to simplify the virtual network management: DHCP, DNS, NFS, HTTP, SSH, NTP, Netfilter, dynamic routing protocols (RIP and OSPF), and QoS management with *Traffic Control* [11]. Moreover, it is easily possible to add some new services through a plug-in system available in *NEmu*. A router is running a customized image version of *TinyCore* which is a lightweight and highly configurable Linux distribution [12]. Such a system typically requires about ~30 MBytes on disk and ~100 MBytes in memory with all services

[1] http://nemu.valab.net