Kathará: A Lightweight Network Emulation System

Mariano Scazzariello
Roma Tre University
Rome, Italy
mariano.scazzariello@uniroma3.it

Lorenzo Ariemma
Roma Tre University
Rome, Italy
lorenzo.ariemma@uniroma3.it

Tommaso Caiazzi
Roma Tre University
Rome, Italy
tommasocaiazzi@gmail.com

***Abstract*—In computer networks, tests to ensure the correct behaviour of network equipment or protocols are often required. Because of the high cost of physical hardware, these tests are always performed in a virtual environment. Kathará is a network emulation system which accurately reproduces the behaviour of a real system. It can exploit several virtualization technologies leveraging on its modularity. Lately, Kathará has been rewritten to overcome some implementation limitations and performance issues. This paper presents the Kathará model and its new architecture, demonstrating its value, comparing its scalability and performance with Netkit (another state-of-the-art tool for network emulation) and with the previous version of Kathará.**

***Index Terms*—emulation, network virtualization**

I. Introduction

In computer networks, it is often required to perform experiments to ensure that network equipment are working properly or to test new features and protocols.

Performing such experiments is often a difficult task. On one hand, experiments cannot be performed in the production network, since it hosts services for customers that cannot be interrupted or even perturbed. On the other hand, the possibility of setting up test-beds that are large enough to perform meaningful experiments is often too much expensive.

For these reasons, plenty of virtual test environments have been developed. They put at user's disposal a virtual environment that can be exploited for tests, experiments, and measures. Virtual test environments can be classified in two broad categories: systems that aim at *reproducing the performance* and systems that aim at *reproducing the functionalities* (configurations, architectures, protocols), with limited attention to performances.

In the latter context, several tools have been developed. Among them there are Netkit [1] and Kathará [2]. Such tools share the same configuration language but differ in terms of the virtualization technology that is used to emulate a network scenario. Recently, Kathará as been rewritten to overcome some implementation limitations and performance issues.

This work aims to compare the performance between Netkit and the two versions of Kathará. First, the Kathará model and the architecture of the new version are described. Then, the main differences between the aforementioned tools are presented. Finally, a practical comparison between the tools is illustrated.

978-1-7281-4973-8/20/$31.00

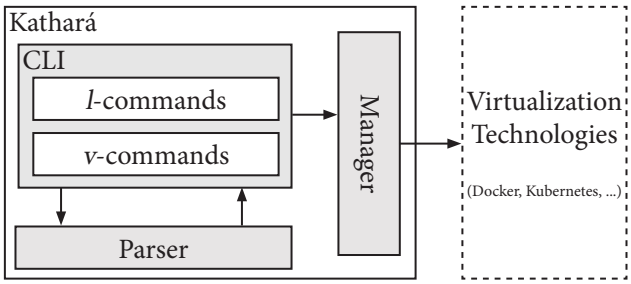

Fig. 1: The new Kathará Architecture.

II. Kathará Framework

This section presents the Kathará Framework, giving a glance at its model and at its new architecture. Kathará is released under GPL-v3 licence. Source code is available at [3].

A. Model

Kathará is based on the concepts of *device*, *collision domain*, and *network scenario*.

Device: a virtual entity which acts like a real network device. It is mainly composed of one or more virtual network interfaces, a virtual CPU with RAM, and a virtual disk. For example, a Kathará device can be used to emulate a fully-fledged router, or a DNS server, or a webserver, etc.

Collision Domain: a virtual Layer 2 LAN that interconnects devices. It acts like a physical link between device's interfaces, forwarding all packets received from an interface to all the other interfaces on that collision domain, without modifying them. Simply, it behaves like a network hub.

Network Scenario: a set of devices connected by means of collision domains. It provides a simple representation of complex networks consisting of several devices and collision domains. In Kathará, a network scenario is represented as a directory containing a file with the network topology, and, for each device, files and folders containing the configuration of that device.

B. Architecture

The new Kathará architecture, illustrated in Fig. 1, consists of three main components: CLI, Parser, and Manager.

CLI – Command Line Interface: Kathará exposes a set of commands accessible through a shell which allows to interact with Kathará modules. Such commands are divided into three categories.

1) *v*-prefixed commands: commands used to configure, start up, and shut down a single network device;
2) *l*-prefixed commands: commands used to configure, start up and shut down a whole network scenario;
3) General commands: general purpose commands for system settings, for getting running devices information, and for performing global actions on devices and scenarios.

Parser: The responsibility of this component is to convert the network scenario into Kathará domain objects which are agnostic with respect to the adopted virtualization technology. The network scenario can be described in any format, as long as the Parser logic is changed accordingly. The Parser acts as an indirection between the network scenario and the managers, so its goal is to convert the chosen format into domain objects.

Manager: A Manager has the responsibility to interact with a specific virtualization engine. It converts a network scenario, represented with domain objects obtained from the Parser, into the virtualization engine's language. So, the engine has the role to deploy and manage a network scenario. Managers are components that implement the same interface. This solution easily allows to support several virtualization technologies, increasing system modifiability.

III. Tools Comparison

This section compares the features of Netkit and the first Kathará release (referred as *Old Kathará*, until version 0.36.1). Then, the evolution from Old Kathará to the latest Kathará release (*New Kathará*, from version 2.0) is discussed.

A. Differences between Netkit and Old Kathará

The main difference is that Netkit leverages on UML (User-Mode Linux) virtual machines to implement devices, while Old Kathará uses Docker [4] containers. Containers represent a lightweight alternative to standard virtualization solutions, avoiding the overhead introduced by full virtualization that needs to maintain multiple kernels (one per virtual machine).

With Docker containers, the management of different filesystem images is simplified by Docker images, allowing devices to easily use different images in the same network scenario.

Also, since Kathará is based on Docker, it runs on all main OSes (Windows, MacOS and Linux), while Netkit can only be installed on Linux platforms.

On the other hand, Old Kathará is single threaded while Netkit leverages on parallel execution for starting up and shutting down network scenarios.

Also, Netkit runs its virtual machines in user space, so it is resilient to any privilege escalation attack. Docker is known to allow privilege escalation attacks [5], so it requires root privileges to run containers. Moreover, since Old Kathará starts its containers with the `privileged` flag, it is even easier to perform those attacks.

In Old Kathará, collision domains are implemented with Docker `bridge` networks, so they must be patched in order to behave like hubs and not like switches. Docker networks require an IP subnet assigned to them. For this reason, they do not satisfy the definition of collision domain.

B. Differences between Old Kathará and New Kathará

The New Kathará introduces multithreading for starting up and shutting down network scenarios. The whole code has been rewritten to support multiple managers, overcoming the limitations of Old Kathará which only supports Docker as virtualization technology. Currently, New Kathará implements a Docker Manager and a Kubernetes [6] Manager (called Megalos [7]).

Old Kathará leverages on Docker `bridge` networks for collision domains, that require to be patched in order to behave like hubs. This results in a performance loss, since the patch is applied for each collision domain in the scenario. To overcome this limitation, a new Docker Network Plugin has been developed and used by New Kathará. The plugin, written in Go, is specifically designed to implement Kathará collision domains. Hence, bridges are already defined with the proper configuration (to behave like hubs) and no further patches are required. Moreover, unlike Old Kathará, networks do not require an IP subnet since they are created as pure Layer 2 LANs.

Unlike Old Kathará, New Kathará does not need the `privileged` flag to start containers. This reduces the chances to perform privilege attacks. However, it is still possible to run containers with the aforementioned flag, if required.

IV. Demonstration

We are going to present a performance comparison between Netkit, Old Kathará and New Kathará (using the Docker Manager). First, we describe some network scenarios, consisting of several devices, which will be used for the comparison. Then we will use the presented tools to run the scenarios and measure the time needed to start up, showing some real time statistics such as CPU and RAM usage. Finally, we show the performance boost offered by the New Kathará compared with the older tools.

V. Conclusions

This paper describes Kathará, a network emulation tool. It focuses on its new architecture and main features. Then, it compares Netkit and Old Kathará, and Old Kathará with the latest Kathará release. Finally, it describes the demonstration that will be performed.

References

[1] M. Pizzonia and M. Rimondini, "Netkit: Easy Emulation of Complex Networks on Inexpensive Hardware," 01 2008, p. 7.
[2] G. Bonofiglio, V. Iovinella, G. Lospoto, and G. Battista, "Kathará: A container-based framework for implementing network function virtualization and software defined networks," 04 2018, pp. 1–9.
[3] Kathará Framework. [Online]. Available: https://github.com/KatharaFramework
[4] Docker Inc. Enterprise Container Platform. [Online]. Available: https://www.docker.com/
[5] Docker. Docker security. [Online]. Available: https://docs.docker.com/engine/security/security/
[6] Kubernetes. Kubernetes. [Online]. Available: https://kubernetes.io/
[7] M. Scazzariello, L. Ariemma, G. Di Battista, and M. Patrignani, "Megalos: A Scalable Architecture for the Virtualization of Network Scenarios," to appear in NOMS 2020.