Grant Agreement No.: 687871

ARCFIRE

Large-scale RINA Experimentation on FIRE+

Instrument: **Research and Innovation Action**
Thematic Priority: **H2020-ICT-2015**

D4.5 How to experiment with the IRATI RINA implementation on FIRE+

Due date of Deliverable: Month 28
Submission date: April 2018
Final version: April 30th 2018
Start date of the project: January 1st, 2016. Duration: 30 months
version: V1.0

Project funded by the European Commission in the H2020 Programme (2014-2020)		
Dissemination level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

H2020 Grant Agreement No.	687871
Project Name	Large-scale RINA Experimentation on FIRE+
Document Name	Deliverable 4.5
Document Title	How to experiment with the IRATI RINA implementation on FIRE+
Workpackage	WP4
Authors	Dimitri Staessens (imec) Sander Vrijders (imec) Eduard Grasa (i2CAT) Leonardo Bergesio (i2CAT) Miquel Tarzan (i2CAT) Bernat Gaston (i2CAT) Sven van der Meer (LMI) John Keeney (LMI) Liam Fallon (LMI) Vincenzo Maffione (NXW) Gino Carrozzo (NXW) Diego Lopez (TID) John Day (BU)
Editor	Dimitri Staessens
Delivery Date	April 30th 2018
Version	v1.0

Abstract

This deliverable D4.5 consists of a set of links to webpages and wikis that assist experimenters in deploying and experimenting with the RINA prototypes on FIRE+ using Rumba. These webpages provide comprehensive descriptions of RINA examples and how to deploy them on the FIRE+ platform.

This brief document serves as a companion report to these wiki pages.

Table of Contents

1 Introduction 6

2 Rumba: A framework to bootstrap a RINA network on a testbed 7

3 IRATI 9

4 rlite 10

5 Ouroboros 11

List of Figures

1 A picture of the Rumba wiki 7
2 IRATI wiki 9
3 rlite project on GitHub 10
4 Ouroboros tutorials 11

1 Introduction

This document briefly introduces the online documentation that will assist experimenters in deploying the software to conduct RINA experiments on FIRE+ and GENI testbeds. The online documentation consists of wikis, one for the RUMBA prototype [1], which was developed as part of ARCFIRE, and one for each of the three supported prototypes.

All development on testbed tools in ARCFIRE were done as part of Rumba and is detailed in D3.1 [2] and D3.2 [3].

This deliverable contains 4 sections, each briefly introducing the wiki, its current status and planned future additions. These wikis are living documents and are expected to be updated considerably in the next few months with detailed descriptions of how to reproduce ARCFIRE experiments. Some details may only be provided when the outcomes are accepted for publication.

2 Rumba: A framework to bootstrap a RINA network on a testbed

Rumba is a Python framework that allows users to write Python scripts to define recursive internet networks and run scripted experiments locally and on FIRE+ and GENI. First, Rumba creates a physical network on one of the selected testbed plugins. If needed, Rumba can do an installation of the selected prototype plugin on the testbed machines. The network is then bootstrapped on the available nodes. Users can then run a Storyboard which emulates real network traffic. Finally, the experiment can be swapped out of the testbed. The logs of both the applications which were ran with the Storyboard as the prototype logs are retrieved if desired. The main goals of Rumba are ease of use, automated experimentation and reproducibility of results.

The Rumba wiki can be found at https://arcfire.gitlab.io/rumba/. The purpose of the wiki is to allow network researchers to easily start experimenting with Rumba.

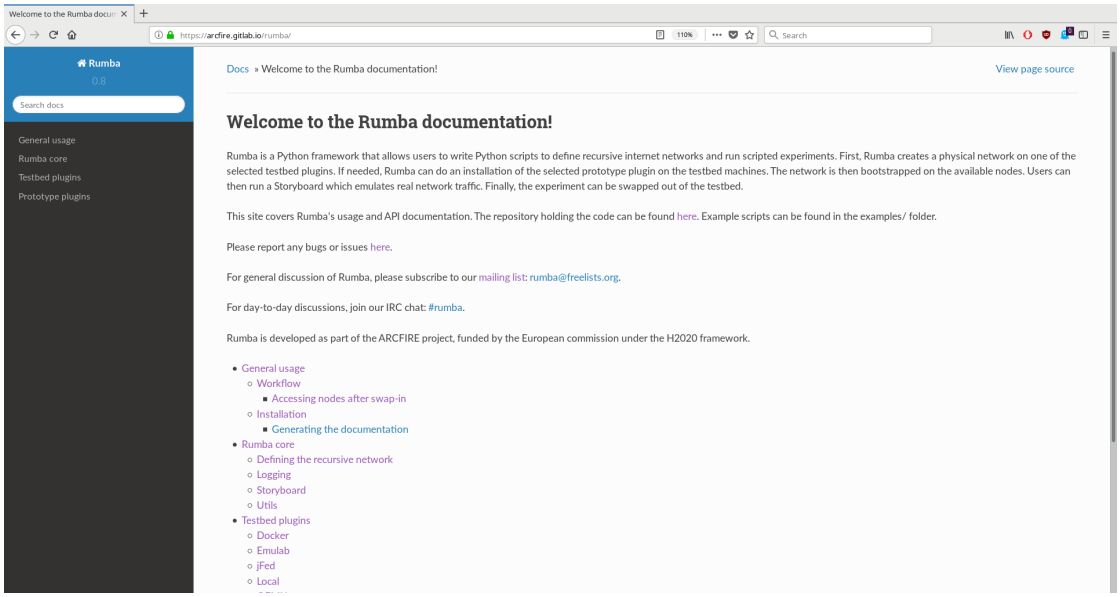

Figure 1: A picture of the Rumba wiki

The current wiki consists of 4 main sections. The first section is a section on general usage. It explains how to install Rumba; either through pip, or through cloning the git repository and running setup.py. It also explains how to generate the documentation, if a user wants to view it locally, or as a pdf instead of the HTML pages. The general workflow of Rumba is also detailed.

The second section describes the API of the Rumba core, which is used to define the recursive network. The storyboard's API is also described, which allows a user to emulate a real network environment.

In the third section, the user can read about the API of a specific testbed. For instance, the jFed

testbed will describe that the user needs to download their certificate file before running the script, and pointing Rumba to this file.

In the last section, the specific API of the prototypes is described. The different prototypes all have their own wiki as well, as detailed in the other sections of this document.

In the future, we intend to extend this wiki with short tutorials which should make it even easier for new experimenters to define their recursive network. Furthermore, we intend to detail how the different ARCFIRE experiments can be reproduced by providing a section that shows these Rumba scripts and explains them.

3 IRATI

IRATI is an open source implementation of the RINA architecture targeted to the OS/Linux system, initially developed by the FP7-IRATI project, and enhanced within the FP7 PRISTINE and H2020 ARCFIRE projects. It supports RINA over Ethernet, TCP, UDP and shared memory, as well as IP over RINA scenarios.

IRATI features an SDK that allows the programmability of layer management and data transfer policies for the different IPC Process components. IRATI implements the POSIX-like RINA API contributed by ARCFIRE (described in D3.1). This API is the same offered by rlite (Sec. 4), allowing applications to seamlessly run on top of both prototypes.

IRATI is supported by Rumba and can be deployed on the jfed and emulab experimental infrastructures, as well as locally via the Rumba qemu testbed.

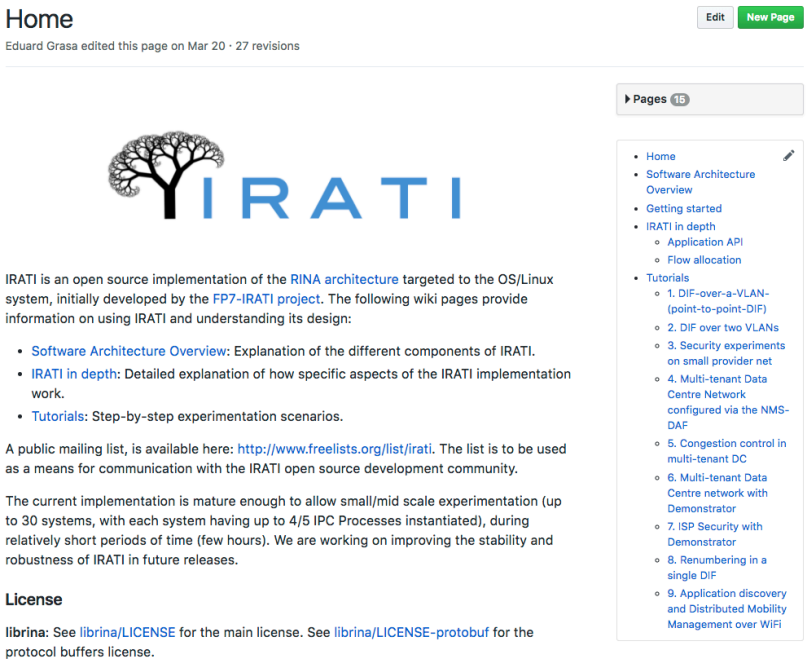

Figure 2: IRATI wiki

The IRATI wiki is accessible via the IRATI github site at `https://github.com/IRATI/stack/wiki`. It contains documentation on the operating systems supported by IRATI, installation instructions and how to use and configure the basic implementation and the different plugins. It also documents the use of the example applications and utilities shipped with IRATI, and features a comprehensible set of tutorials that illustrate a number of RINA configurations.

4 rlite

The *rlite* project provides a lightweight Free and Open Source implementation of RINA for Linux-based operating systems. The main goal of rlite is to become a baseline implementation for RINA systems to be used in production. To achieve this goal, rlite focuses on robustness and performance (up to 10Gbps or more) by leveraging on a clean keep-it-simple design. The current implementation includes about 38 Klocs of C/C++ code, split between kernel-space and user-space. Support for programmability is provided for the DIF constants (i.e., EFCP header layout) and policies for the DIF management layer components. Applications can use the POSIX-like RINA API defined by ARCFIRE (described in D3.1 and D3.2). The rlite RINA stack can work over legacy transports like like Ethernet and WiFi, or interoperate with IP-free networks. The shim DIF over UDP enables RINA over IP; the iporinad deamon allows IP over RINA with an MPLS-like architecture; finally, the rina-gw daemon allows to deploy RINA next to TCP/IP networks.

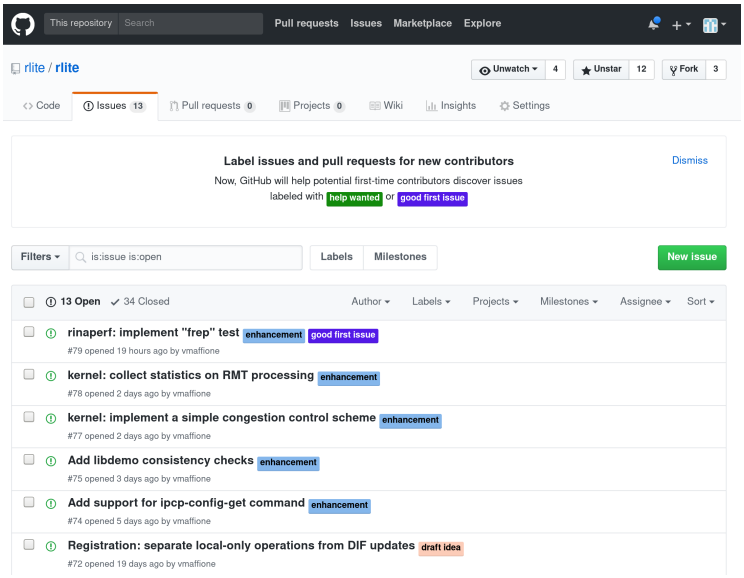

Figure 3: rlite project on GitHub

The rlite prototype is supported by Rumba and can be deployed on the jfed and emulab experimentation infrastructures, as well on the local Rumba qemu testbed. Documentation is available at `https://github.com/rlite/rlite` in the `README.md` file, which includes installation instructions, an overview the software architecture, tutorials on how to setup RINA networks over Ethernet, WiFi or UDP, a description of all the items that can be tuned and the policies available for the IPC processes, documentation for the RINA API, and an overview of various RINA applications and tools.

5 Ouroboros

Ouroboros is an Inter-Process Communications (IPC) subsystem for POSIX operating systems. It incorporates a fully decentralised packet switched transport network based on concepts that were developed as part of the Recursive Internet Architecture (RINA). It provides a simple and minimal API for synchronous and asynchronous IPC over this network. Ouroboros can use IP, Ethernet LLC, Ethernet DIX or Ethernet PHY at the lowest layer.

Ouroboros is supported by Rumba and can be deployed on the jfed, emulab and qemu testbeds. In addition, it can be deployed locally using docker containers and an Ouroboros-specific local testbed.

The Ouroboros website is maintained at imec at https://ouroboros.ilabt.imec.be. It's designed to be a one-stop portal for everything about the Ouroboros prototype and contains tutorial sections targeted at users and application developers.

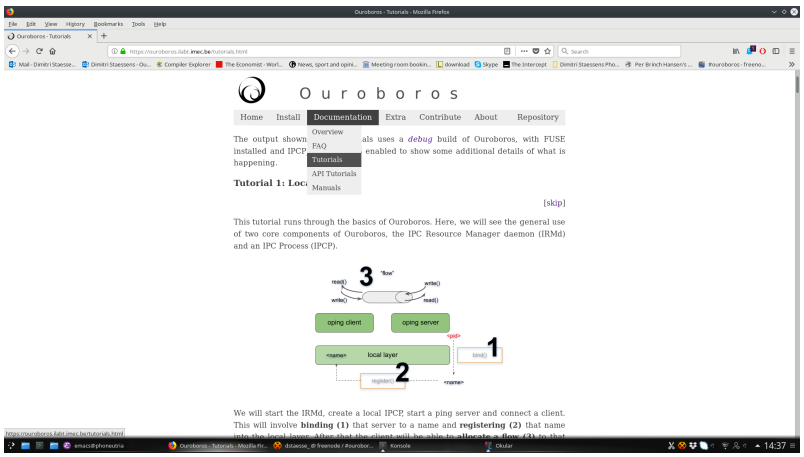

Figure 4: Ouroboros tutorials

The current tutorials section contains basic tutorials for manual set up of Ouroboros networks locally and over and Ethernet network.

The web page will be updated frequently, tracking the ongoing development of Ouroboros.

References

[1] Rumba measurement framework. [Online]. Available: https://gitlab.com/arcfire/rumba

[2] A. consortium. (2016, December) H2020 ARCFIRE deliverable D3.1: Integrated software ready for experiments: RINA stack, Management System and measurement framework. [Online]. Available: http://ict-arcfire.eu

[3] ——. (2018, April) H2020 ARCFIRE deliverable D3.2: Final integrated software: RINA stack, Management System and measurement frameworkk. [Online]. Available: http://ict-arcfire.eu