# Introducing ProtoRINA: A Prototype for Programming Recursive-Networking Policies

Yuefeng Wang
Boston University
wyf@bu.edu

Ibrahim Matta
Boston University
matta@bu.edu

Flavio Esposito*
Exegy, Inc.
fesposito@exegy.com

John Day
Boston University
day@bu.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

## Keywords

Future Network, Recursive Networking, RINA, Policy-based Programming, Experimental Evaluation.

## ABSTRACT

ProtoRINA is a user-space prototype of the Recursive InterNetwork Architecture. RINA is a new architecture that builds on the fundamental principle that *networking is interprocess communication*. As a consequence, RINA overcomes inherent weaknesses of the current Internet, *e.g.*, security, mobility support, and manageability. ProtoRINA serves not only as a prototype that demonstrates the advantages of RINA, but also as a network experimental tool that enables users to program different policies using its built-in mechanisms. In this note, we introduce ProtoRINA as a vehicle for making RINA concepts concrete and for encouraging researchers to use and benefit from the prototype.

## 1. INTRODUCTION

The current Internet has been facing many problems due to its inherently flawed architecture. Many proposals (new network protocols and architectures) have attempted to ameliorate these problems. Along with these proposals, many networking tools (*e.g.*, emulators, simulators, experimental testbeds) have been built to develop, test, and evaluate these new protocols or architectures before a real world deployment. Our approach has relied on the fundamental principle that *networking is Inter-Process Communication (IPC) and only IPC*. As a consequence, our Recursive InterNetwork Architecture (RINA) [1, 2] inherently supports capabilities such as security, mobility, and manageability. ProtoRINA is a user-space prototype of RINA.

ProtoRINA is designed based on two main principles: (i) divide and conquer (recursion), and (ii) separation of mechanisms and policies. ProtoRINA provides a framework with common mechanisms so researchers do not have to implement these from scratch, rather they can focus on programming different policies (supported by user applications or network management applications).

ProtoRINA offers several features: (i) ProtoRINA is not restricted to the Internet Protocol (IP), so it enables experimentation with new control and management applications; (ii) ProtoRINA supports research not only on user applications but also network management; (iii) ProtoRINA can be used as a teaching tool by educators in networking and distributed systems classes; and (iv) ProtoRINA can be used to run real experiments both on local-area networks and on wide-area network testbeds such as GENI [3]. Details of ProtoRINA, including code and user manual, can be found in [4].

The rest of this note is organized as follows. Our RINA architecture is briefly described in Section 2. Some details of ProtoRINA are presented in Section 3. Section 4 summarizes some experimental results obtained using ProtoRINA.

## 2. RECURSIVE ARCHITECTURE

RINA is based on the fundamental principle that *networking is IPC and only IPC*. It is part of a more general model of distributed applications, operating systems, and networks. A set of distributed application processes, called a Distributed Application Facility (DAF), cooperate to perform some function, *e.g.*, communication service, weather forecast, genomics, *etc.* A Distributed IPC Facility (DIF) is a specialization of a DAF which only provides communication service. DIFs are specialized to manage a given range of operation with respect to performance characteristics and scale. The greater the range of these in a network the more DIFs that are necessary. In some cases, the scope of these DIFs increases with increasing rank, so a higher-level DIF is supported by multiple lower-level DIFs.

Applications request the allocation of communication resources with another application. Resolving the "what" an application wants to talk to, to the "where it is" is done by a Flow Allocator. Addresses within each DIF are not exposed to applications, and there are no well-known ports. One implication of the IPC model is that the application name space can have greater scope than any one DIF and hence a global address space is not required.

RINA separates mechanisms and policies. For example, IPC processes all use the same mechanisms but may use different policies in different DIFs with different scopes. Also RINA simplifies the network system by only using two policy-configurable protocols. The Common Distributed Application Protocol (CDAP) is the only application protocol required, and is also used for network management. The Error and Flow Control Protocol (EFCP) is used for data transfer.

## 3. PROTORINA: A PROTOTYPE OF RINA

ProtoRINA Version 1.0 has been tested on our Boston University campus network and on the GENI testbed, and we have done some preliminary cross-debugging with other RINA prototypes [5, 6]. The current version consists of around 55,000 lines of Java code following the RINA specifications of January 2013. This version is not a complete implementation of RINA and we continue to modify and add elements.

*F. Esposito's work was done while at Boston University.

## 3.1 RINA Node

In ProtoRINA, a *RINA node* is a host (or machine) where application processes and IPC processes reside. As shown in Figure 1, application processes or high-level IPC processes communicate with their peers using the communication service provided by underlying low-level IPC processes, which act as points of attachment. The mapping from an application process (or high-level IPC process) to the lower level IPC process is resolved by the underlying DIF.

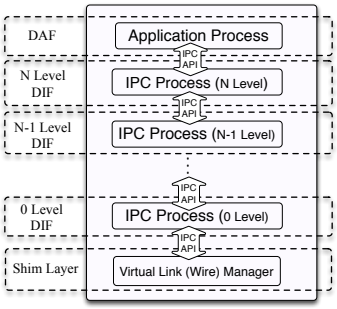


**Figure 1: RINA Node**

Physical connectivities between IPC processes in level-0 DIFs are emulated by TCP connections via a shim layer that exposes a RINA API. The shim layer includes functionalities such as resolving a user-defined level-0 IPC process name to an IP address and a port number. More generally, using the shim layer enables building RINA overlays on top of Ethernet, TCP, or UDP.

## 3.2 RINA Components and APIs

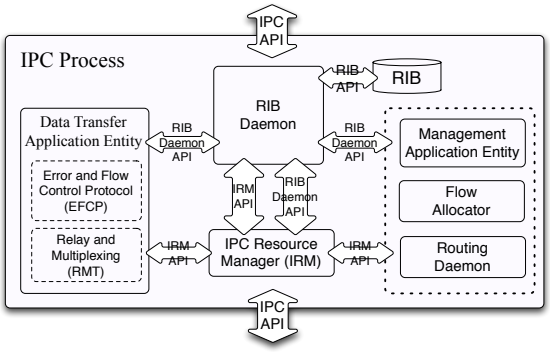


**Figure 2: IPC Process**

Figure 2 illustrates how different components of an IPC process interact with other components through RINA APIs. ProtoRINA provides users with two RINA APIs: *RIB Daemon API* and *IRM API*. Users can develop new applications or define new network management policies using these two RINA APIs.

Each application (or IPC) process has a Resource Information Base (RIB) that stores its view of the information related to the operation of the DAF (or DIF). The *RIB Daemon API* is used to access information stored in the local RIB or in a remote application (or IPC) process' RIB through CDAP messages. The RIB Daemon is based on a publish/subscribe model, and it provides timely information to the layer management (application) tasks of the DIF (DAF). The *IRM API* is used by an application (or IPC) process to allocate and maintain the connections to its peers. More details of ProtoRINA components and RINA APIs can be found in [4].

## 3.3 Configuration and Policies

In this prototype, each RINA node has a configuration file that includes the information of all processes (application processes and IPC processes) residing on it. This information includes process' naming information and location of its configuration file. When a RINA node is initialized, IPC processes and application processes on the node are bootstrapped based on their own configuration files. The configuration file includes information on the process' underlying DIF, routing policies, and so on. Users can define other properties in the configuration file for their own applications.

Users can specify different policies in ProtoRINA configuration files. The following is a portion of an IPC process' configuration file. In this example, the enrollment of a new IPC member into the DIF requires it to provide user and password information, and the IPC process is instantiated to use a link-state routing protocol where link-state updates are sent to neighbor processes every 10 seconds, and the path cost is calculated using hop count.

```
rina.enrollment.authenPolicy = AUTH_PASSWD
rina.routing.protocol = linkState
rina.routingEntrySubUpdatePeriod = 10
rina.linkCost.policy = hop
```

ProtoRINA has policy holders where users can also define their own policies using the given RINA APIs.

# 4. RESULTS OBTAINED USING PROTORINA

We have been using ProtoRINA to demonstrate the RINA architecture and its advantages, and also to experiment with different policies. In [7] we demonstrate RINA over the GENI testbed with two fundamental experiments: DIF enrollment and dynamic DIF formation. We also show how RINA naturally supports the provision of a virtual private cloud service in [8]. In [9] we use ProtoRINA on GENI to experiment with different routing policies configured over different DIF topologies.

# 5. ACKNOWLEDGEMENT

We would like to thank the National Science Foundation (NSF grant CNS-0963974), the GENI Project Office (CNS-1346688), and also Lou Chitkushev and other members of the RINA team for their support.

# 6. REFERENCES

[1] Boston University RINA Lab. http://csr.bu.edu/rina.

[2] J. Day, I. Matta, and K. Mattar. Networking is IPC: A Guiding Principle to a Better Internet. In *Proceedings of CoNEXT/ReArch*, New York, NY, 2008.

[3] GENI. www.geni.net.

[4] ProtoRINA. http://csr.bu.edu/rina/protorina.

[5] The IRATI Project. http://irati.eu.

[6] TRIA Network Systems. www.trianetworksystems.com.

[7] Y. Wang, F. Esposito, and I. Matta. Demonstrating RINA Using the GENI Testbed. In *Proceedings of the* $2^{nd}$ *GENI Research and Educational Experiment Workshop (GREE)*, Salt Lake City, UT, March 2013.

[8] F. Esposito, Y. Wang, I. Matta, and J. Day. Dynamic Layer Instantiation as a Service. In *Demo at the* $10^{th}$ *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Lombard, IL, April 2013.

[9] Y. Wang, I. Matta, and N. Akhtar. Experimenting with Routing Policies Using ProtoRINA over GENI. In *Proceedings of the* $3^{rd}$ *GENI Research and Educational Experiment Workshop (GREE)*, Atlanta, GA, March 2014.