BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2017
http://excel.fit.vutbr.cz

A New Approach for Distributed Applications in RINASim

Kamil Jerabek*

Abstract
Recursive InterNetwork Architecture yields a completely different approach of communication and manipulation of objects in distributed applications. Instead of passing objects directly between nodes, it manipulates with them locally in RIB. This paper is focused on the meaning and description of the key components RIB Daemon and Resource Information Base that are responsible for handling of application objects within Distributed Application Facility.

Contribution of this work is to extend the functionality of the RINA Simulator, by all the components cooperating in Distributed Application Facility. Mainly the two mentioned modules, and an appropriate programming interface for creating distributed applications based on RIB Based Programming model.

Keywords: Recursive InterNetwork Architecture — RINA — RINA Simulator — OMNeT++

Supplementary Material: Downloadable Code

*xjerab18@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Distributed applications as well as cloud computing is taking advantage over standard communications principles. The communication goes through the network. The *Software Defined Networking* (SDN) moves to the forefront, mainly in this environment.

Todays network technologies are built on the current underlying architecture based on TCP/IP model. This model is facing lack of mobility mechanisms, multihoming. Despite, there is an effort to supplement such mechanisms to the model, it is hard to implement them.

Recursive InterNetwork Architecture (RINA) is a new network architecture created by John Day. The architecture is based on principles described in his book *Patterns in Network Architecture* [1]. RINA represents a clean slate architecture that solves all of the mentioned problems from scratch.

RINA is based on the observation that all communication is just interprocess communication between two application entities and every application is a distributed application. Hence, it defines an **Distributed Application Facility (DAF)** with an application layer, compared to the TCP/IP model.

TCP/IP needs explicit programming techniques (for example, using Open MPI [2]) to create distributed applications. RINA should provide an easier approach, which is native to architecture itself. It alludes to the fact that application programmers should not care about networking and they should just program the applications by manipulating application objects locally.

The paper is focused on design of model of Appli-

cation Layer and on distribution of data objects within this layer. Hence, emphasis is placed on two core components. **Resource Information Base (RIB)**, that is responsible for storing data objects within each Application Process of DAF, and **RIB Daemon**, that is intended to maintain and distribute objects stored in RIB between Application Processes.

The aim of the work is to extend current model of RINA architecture in RINASim [3] in OMNeT++ [4]. The RINASim is developed on Faculty of Information Technology on Brno University of Technology, as one of the outcomes of FP7 project Pristine [5].

Chapter 2 contains a brief description of RINA. There is only a description of the application layer DAF and its components. Chapter 3 describes design and implementation details. Chapter 4 shows validation of newly implemented modules.

The following chapters came out from the book [1] by John Day, discussions with John Day and Steve Bunch and from RINA Specifications and other resources: [6], [7], [8], [9], [10], [11], [12], [13], [14].

2. State of the Art

This section introduces basic RINA principles that are need to be mentioned to provide better understanding of the core of this work. However, more information is provided in resources mentioned in section 1.

The RINASim contained only one Application Entity that worked only as traffic generator. This work brings full-featured Application Process, containing all necessary components mentioned in specifications, such as Application Entity, Enrollment, RIB, RIB Daemon, CDAP, Socket, together with universal programming API. All modules are designed so that they can be extended by new specific implementations. The functionality of modules will be briefly described in the following section.

The Recursive InterNetwork Architecture is based on presumption that computer networking is just Inter Process Communication (IPC). In RINA, there exists only one layer called **Distributed IPC Facility (DIF)**, it is a set of cooperating **IPC processes**.[12] The layer is recursively repated and each DIF is completely independent. This concept of DIF could be generalized to **Distributed Application Facility (DAF)**. It is a set of cooperating **Application Processes (AP)**. Application Process is a program intended to accomplish some purpose and it can be instantiated on a processing system. An AP uses IPC processes for communication.

An AP contains one or more **Application Entities (AE)**. Application Entity is a component (task) of an AP, that is responsible for managing communication

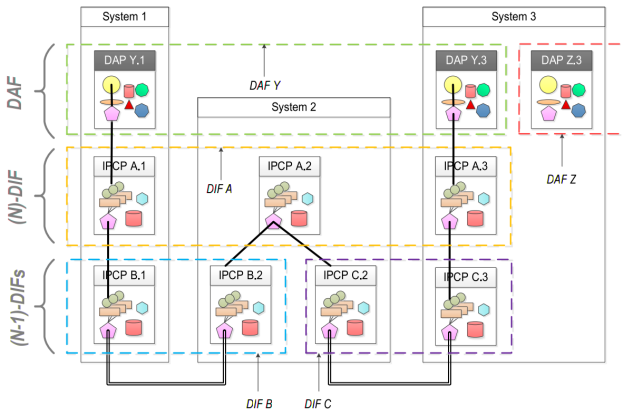

Figure 1. Recursive InterNetwork Architecture structure [11]

with other AEs (potentially multiple connections at the same time). AE implements an application protocol. Application protocol means shared understanding of the purpose of communication, protocol, set of objects and their meaning that the two AEs exchange.[6]

In RINA, there is only one application protocol called **Common Distributed Application Protocol (CDAP)**. CDAP enables distributed applications to deal with communication at the object level without the need to do an explicit serialization and other input/output operations. Only the create/delete, read/write and start/stop (execute/suspend) operations can be performed on objects from the application perspective.[7]

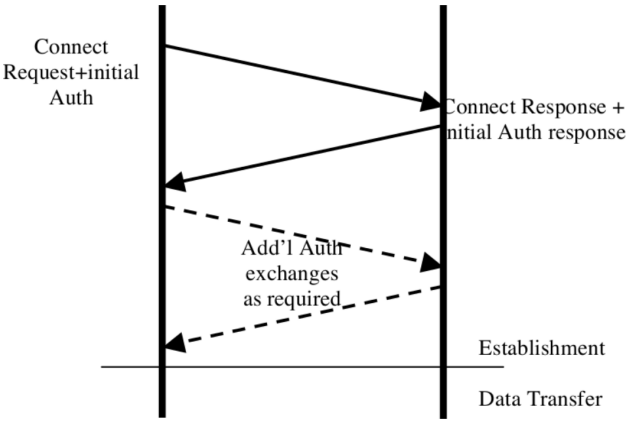

Figure 2. Connection establishment [7]

All communication between two Application Processes must go through **Common Application Connection Phase**. This is the first phase after positive allocation of a new connection, as it is depicted in figure 2. It consists of CDAP Connect messages. These messages also carry authentication. It is also possible to extend this authentication mechanisms and use more authentication exchanges if needed.

There is also another phase of communication called **Enrollment**. This phase starts right after AP establishes application connection with a member of a

DAF for the first time. Or, when AP is reconnecting due to lost connection.

The purpose of Enrollment is to create a sufficient shared state within DAF. This means to create necessary objects in database and get initial information from the member AP. Also, one or more synonyms (i.e., adresses) are assigned to the new member AP during this phase. After this phase, the connecting AP is full-featured and fully operational member of DAF.

2.1 Resource Information Base

Resource Information Base (RIB) is the logical representation of the local repository of objects in DAF. Each member of a DAF has own portion of the information stored in local RIB. All objects are accessible via RIB Daemon, that is responsible for managing and maintaining them.

The RIB is a storage, from the point of view of the operating system model. There are no restrictions for implementation of the RIB. In DAF, it should most likely be implemented as some kind of database of an application related objects.

In current TCP/IP based networks, the RIB should be compared to Management Information Base that is used for the same purpose (storing objects).

2.2 Objects

Object is the designation for a structured data that the CDAP is manipulating with. All modern programming languages have object concept, it is useful to deal with data on an object level from its perspective.

Objects are the main building blocks of the RIB. Two communicating AEs create a shared object space and they provide access to the portion of the application's RIB.

All objects belong under some access rights. There should be at least two groups of objects. The first one is accessible only by AEs that created it and using it to communicate. The other one should contain the rest of objects of an application. These objects can be accessed by any component, but more access rights models should be introduced.

The objects stored in RIB would be of two types. The passive objects that contain static information. And the active objects, in which information should have a side effect, e.g., setting it to true causes a reboot of an processing system.

2.3 RIB Based Programming Model

The RINA comes up with a different approach of programming distributed applications (or network applications) than it is in today's networks. As it was described earlier, the CDAP protocol and one or more AEs can be used to create a Distributed Application. All the data are stored in distributed RIB.

It introduces an idea that application programmers will be more likely manipulating with local objects in RIB rather than using explicit communication operations. The AEs and CDAP are completely hidden from the application programmer. The application will not use AEs and CDAP APIs directly. They should use higher level application programming interface.[6] The API is primarily designed for manipulation with objects.

2.4 RIB Daemon

RIB Daemon is one of the key component of the AP. It is responsible for managing and maintaining information objects in RIB within DAF. The RIB Daemon should monitor all events that are occurring within DAF because there could be a subscription that is triggered by the event. It can also keep a log of events if needed.

DAF's AP may have several sub-tasks or threads. The RIB Daemon is a common component for those sub-tasks.[9] If any sub-task has requirements for information from other participants in distributed application, it uses RIB Daemon to get it. The getting/setting of an information is done via subscriptions for objects.

All AP's tasks can communicate with the RIB Daemon via special RIB Daemon API. The RIB Daemon accepts subscriptions from tasks and it performs the subscriptions as efficiently as possible. It would contain mechanisms to avoid duplicate subscriptions. The subscription requests would be time driven, event driven, and direct. In DAF, it is expected that there will be a higher proportion of direct requests.

According to the RIB Based Programming model, the RIB Daemons role is also to make information available as efficiently as possible. It would keep the delay of the tasks that are using the data as little as possible. This needs an implementation of some kind of pre-paging mechanism with prediction of objects that will be needed next.

3. Contribution

RINASim should be full-fledged simulator of RINA architecture. The model has to correspond to specifications that are focused only on the resulting behavior. The simulator provided only one static AE that served as traffic generator and only one phase of communication within DAF.

The contribution of this work lies in design, description and implementation of DAF and all participating components. Moreover, special emphasis is on

managing objects in RIB between APs. The work also comes with API for creating more complex applications in RINASim.

The following subsection provides overview of design and implementation of the components and phases of communication.

3.1 Design and Implementation

Current state of RINASim lets us create one instantiation of AP on any host and use functionality of lower DIF IPC processes to provide connectivity. When modeling in the OMNeT++, the main concept is in modules that are hierarchicaly structured and communicating with each other. Therefore, Application Process Instance should be a module.

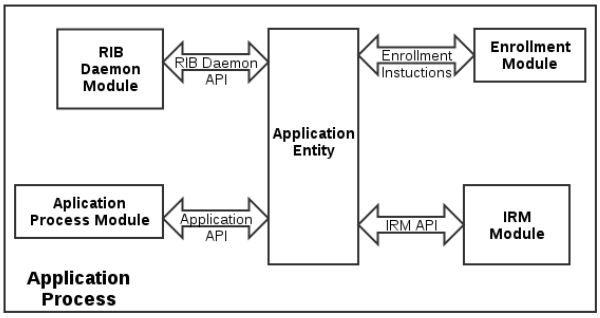

Figure 3. Design of Application Process

The figure 3 depicts all modules of Application Process described earlier. Each module has a communication API with different purpose. The modules are communicating with each other to provide desired functionality. There is also a showed IPC Resource Management (IRM) module and its API. The description of IRM is out of the scope of this thesis. The IRM module is already provided in simulator and it have to be used in this design.

Our implementation allows users to create one static instantiation of an AP with modules shown in figure 4.

There are AE that are communicating with other modules using signals. Reaction on received signal is implemented as a callback function. This enables the opportunity to create user specific AEs. The AEs are spawned dynamically by AP Instance.

The Enrollment module controls communication during enrollment phase and during connection establishment phase. It spawns special Management Application Entities, that are controlling management connections. These AEs are used for exchanging important information, and these AEs are spawned at first in AP.

Application Process instance module is at the top of the application, this module is implemented to provide an usable application API. This API is inspired by API described in document [6]. While creating an application connection, the API call also dynamicaly creates requested AE instance that manages all sent and received messages.

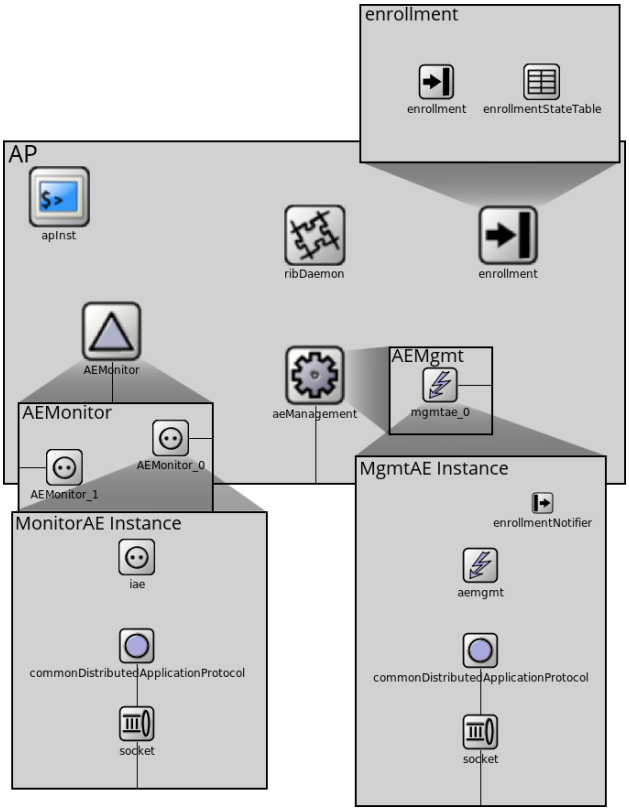

Figure 4. Structure of Application Process and its components

User may create their own Application Process using this API. There is a minimal need for configuration in DAF except for specification which AP instance has to use.

The RIB is implemented as a light weight database with access restrictions. RIB is designed so that more implementations are possible. Tree structure is used in this implementation.

RIB Daemon receives signals from other components of AP. It uses spawned AEs or Management AEs for communication. This depends on objects the RIB Daemon is manipulating with.

4. Testing

The testing was divided into two parts. The first part was focused on validation of implemented Enrollment and CACE phases of communication. Whereas, the second part was focused on API calls, dynamic creation of more modules and on RIB Daemon and appropriate manipulation with data in RIB.

The Enrollment and the CACE phase part of model was tested on five different simulation examples with

the same result. The main focus was on message exchange and appropriate reaction of implemented finite-state machines. And whether each side answers with the expected message.

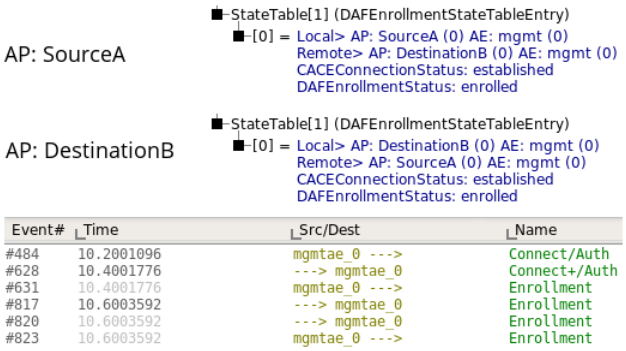

Figure 5. Enrollment communication and final states

Figure 5 depicts CACE and Enrollment message exchanges between two APs. Both APs transitioned to appropriate states in both phases.

The second part was tested on more complex simulation scenario.The scenario contains more nodes and each AP spawns more than two AEs. The main factor here was to focus on correct creation of multiple AE instances, different access rights of objects and distribution of requested objects between nodes.

The behavior of implemented model is deterministic. The simulation scenarios ran repeatedly with the same results.

5. Conclusion

The paper is focused on design and implementation of Application Process participating in DAF (application layer) in RINASim. The main benefit of this work is the understanding of behavior of all parts of Application Process, their design in OMNeT++ and their implementation.

The RINASim contained only single Application Entity that served just as a traffic generator between two static nodes. This work brings full-featured Application Process capable of creating multiple connections, dynamic creation of modules and primarily dealing with communication on data object level. The emphasis is on design a of new approach called RIB based programming model and distribution of data between nodes.

The significant part of the work is design and implementation of an API that provides easier and uniform creation of applications in RINASim. It also helps with better understanding of how to create distributed applications in RINA.

The model was validated against specifications. The model was tested on multiple scenarios always with same result. The implementation was successfully validated as deterministic.

The implementation is part of current official version of RINASim. RINASim is used world wide by many researchers to help understand how RINA works and to experiment with its model. The simulator is marked as one of the official OMNeT++ framework.

The future work should be focused on measuring efficiency of choosen RIB Daemon distribution policies an comparing the amount of communication with current network architecture.

6. Acknowledgement

This work was supported by the Brno University of Technology organization and by the research grant PRISTINE EU-7FP-ICT. I would like to thank Ing. Vladimír Veselý, Ph.D. for his suggestions and astonishing support while this work.

References

[1] John Day. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008. ISBN-13: 978-0-132-25242-3.

[2] Open mpi site. https://www.open-mpi.org/. Accessed: 2016-4-5.

[3] Brno University of Technology. kvetak/rina - github. https://github.com/kvetak/RINA, 2016.

[4] Omnet++ community site. http://www.omnetpp.org. Accessed: 2016-1-9.

[5] PRISTINE consortium. Pristine will take a major step forward in the integration of networking and distributed computing. http://ict-pristine.eu, 2016.

[6] Steve Bunch. Cdap - common distributed application protocol reference, December 2010.

[7] John Day. Recursive ipc network architecture - the interina reference model - part 2: Distributed applications - chapter 1: Basic concepts of distributed applications, 2016.

[8] John Day and Eleni Trouva. Recursive ipc network architecture - the interina reference model - part 2: Distributed applications - chapter 2: Introduction to distributed management systems, 2016.

[9] John Day. Recursive ipc network architecture - the interina reference model - part 3: Distributed interprocess communication - chapter 1: Fundamental structure, 2016.

[10] John Day. Recursive ipc network architecture - the interina reference model - part 3: Distributed interprocess communication - chapter 2: Dif operations, 2012.

[11] Vladimír Veselý. *A NEW DAWN OF NAMING, ADDRESSING AND ROUTING ON THE INTERNET*. PhD thesis, Brno University of Technology, Faculty of Information Technology, 2016.

[12] Vladimír Veselý, Marcel Marek, Kamil Jeřábek, et al. Deliverable 2.6: Rinasim - advanced functionality. http://ict-pristine.eu/wp-content/uploads/2013/12/pristine-d26-rina_sim-draft.pdf, 2015.

[13] Vladimír Veselý, Marcel Marek, Tomáš Hykel, and Ondřej Ryšavý. Rinasim: Your recursive internetwork architecture simulator, September 2015.

[14] Kamil Jeřábek. Recursive ipc network architecture: Analysis and modelling of enrollment. Bachelor thesis, Brno University of Technology, Faculty of Information Technology, 2014.