Excel@FIT 2016
http://excel.fit.vutbr.cz

RINA Simulator: Creating Distributed Applications in RINA Architecture

Bc. Kamil Jeřábek*

Abstract
This paper is focused on application approach in RINA Architecture, on its design and implementation in RINA Simulator in OMNeT++. The work is carried out within research project PRISTINE within which the RINA Simulator is developed. Contribution of this work is to extend the functionality of the simulator by a programming interface and structure for creating distributed applications. There is also presented simple design of Application Programming Interface.

Keywords: OMNeT++ — RINA Architecture — Distributed Application — RINA Simulator

Supplementary Material: Downloadable Code

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

1. Introduction

Recursive InterNetwork Architecture (RINA) is new network architecture and its author is John Day. This architecture solves inherently some drawbacks of current networks such as mobility, multihoming and more.

RINA also presupposes that every application is distributed application. This brings easier approach for creating distributed applications. It is also possible to create distributed applications over current computer networks based on TCP/IP for example by using Open MPI [1]. But it is more complicated than it should be in RINA.

The paper is focused on design and creation of model of Application Layer. The main topics are design of communication Application Programming Interface (API), creation and management of application connections, isolation of resources.

The aim of the work is to extend current model of application layer of RINA architecture in RINA Simulator (RINASim) in OMNeT++. The RINASim is developed on Faculty of Information Technology on Brno University of Technology within PRISTINE project.

The RINASim contained in Application Processes of application layer only one communication entity, the entity was only used to generate traffic for testing bottom layers. This work designs and implements core modules that provides functionality for creation of distributed applications. The core modules are RIB Daemon, Application Entities, Application Process, Enrollment module. The meaning and functionality is described in more details in following chapters. A part of this work is also design and implementation of API between modules as well as API at top a top level to program applications easier way. This includes also dynamic creation of modules. It should bring less configuration.

The following chapter 2 contains only brief description of RINA Architecture. Also it is more focused on Application Layer called Distributed Application Facility (DAF), its parts and functionality. Chapter 3 describes design and implementation of each new module described in Chapter 2. Chapter 4 includes brief description of validation of new implemented parts.

The following chapters came out from RINA Architecture Specifications [2],[3],[4],[5],[6], the book [7] by John Day and discussions with John Day and Steve Bunch.

2. State of the Art

The Recursive InterNetwork Architecture is distributed network architecture. This architecture is based on the presumption that computer networking is just Inter Process Communication (IPC). The main building block of the architecture is a single repeating layer called Distributed IPC Facility (DIF). Each occurence of the layer has the same mechanisms, but from layer to layer it can have different policies. The DIF includes IPC Processes running on different machines that works together to provide flow services to Application Processes (AP). The RINA supports without need of creating any extra mobility mechanisms, multihoming and Quality of Service. There is only one protocol, called Common Distributed Application Protocol (CDAP) which provides communication between two Application Processes (AP).

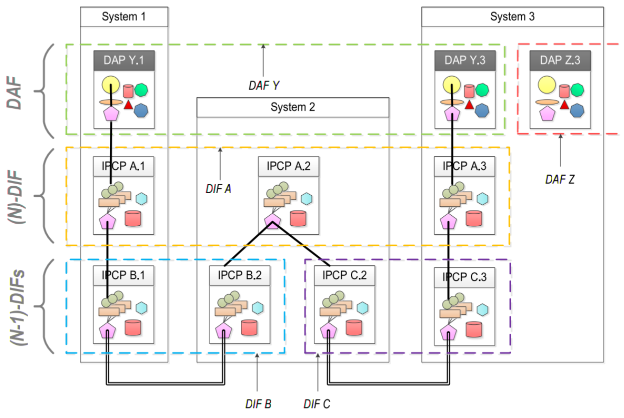


Figure 1. DIF, DAF, DAP with IPCP processes [8]

2.1 Distributed Application Facility

A top layer called Distributed Application Facility (DAF) is an application layer. This layer consist of AP that are instantiations of a program intended for some purpose, which is executed in a processing system. An AP contains one or more Application Entities (AE).

2.2 Application Entity

The Application Entity is task within an application process that is concerned with the interactions with IPC thus providing communication with other AP.[2] From the point of OSI Model the AE implements an application protocol. In RINA model, it is more likely shared understanding of objects on which they can operate.

2.3 RIB Daemon

The Resource Information Base (RIB) is distributed database that is maintained by every single member of a DIF and a DAF. The RIB should be compared to SNMPs Management Information Base that is used for the similar purpose in current TCP/IP based networks, to store objects. The RIB contains information about neighbors, connections, application and user specified data.

The RIB Daemon is responsible for memory management within a DAF. It is common for all subtasks or threads of application process participating in distributed application. Because of every single subtask or thread may require data from the others. It could be asked for an information once, more time (it depends on events) or periodically. The RIB Daemon according to the requests may optimize the information. It maintains database synchronization rules (e.g., commits on update).

The RIB Daemon should be able to provide data within a DAP to programming applications as it would be on a single system available with possibly no delay. Application tasks of DAF members could register subscription to RIB Daemon to take information or to distribute data to one or more members also with defining whether on request or periodically. According to this the RIB Daemon make measurements to reduce the amount of data to be send.

The RIB Daemon works as a gate to access RIB database. The RIB Daemon should provide only relevant information to tasks.

Each AE have to authenticate wheather it can create connection, this is described in more detail in 2.5. In addition each different AE can access the objects that are bound to or intended to be used by this AE. Also different instances of the same AE may have access to the same objects. The AE instances also may not have access to objects of different AE instances of the same AE. This also work the same way for AP instances.

2.4 Common Distributed Application Protocol

The Common Distributed Application Protocol (CDAP) is the only required protocol for the communication between application processes.

From the point of view of the application, there are only six fundamental functions that application can perform on objects. These operations are create/delete, read/write and start/stop. More operations are not needed for communication. All other manipulations does not depend on communication, but it depends on manipulation with data in applications.

2.5 Communication

AE provides communication between two APs. Every communication between two AE consists of three phases: Common Application Connection Establishment (CACE), Authentication, Application Data Transfer Phase.

2.5.1 Connection

The CACE phase is showed on figure 2. It is the first communication. Application connection between two AEs is established during this phase. It consist of

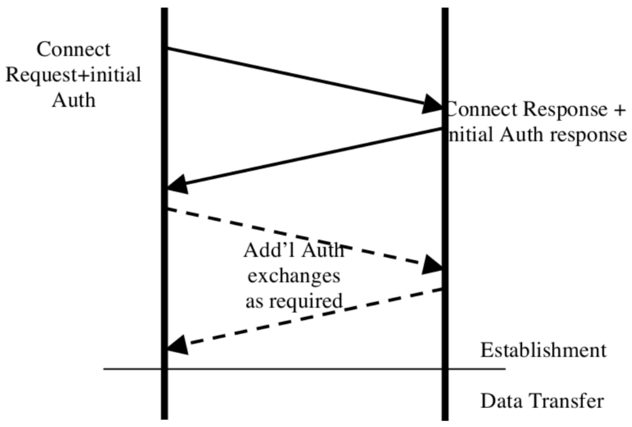


Figure 2. Connection establishment[3]

CDAP connect message exchanges (depicted in Table 1), that create application connection between two AEs. CDAP Connect messages contains authentication. It is also possible to extend this authentication mechanisms and use more authentication exchanges if needed.

Table 1. CDAP Messages used while CACE phase

Opcode	Description
M_CONNECT	Initiate message to create connection from requesting AP, carries authentication
M_CONNECT_R	Response message as an reaction on the previous message, carries response

2.5.2 Enrollment

There is also Enrollment Phase that takes place when Application Process tries to connect to DIF for the first time or it is reconnecting due to connection lost. Enrollment phase occurs right after CACE Phase and authentication. The purpose of Enrollment is to create sufficient shared state within DAF. This means to get information from AP that is member of DAF and to which the AP is trying connect to. Then the member Application Process have to distribute important data to connecting AP. Also it is assigned one or more synonyms (i.e., adresses) to connecting AP to use within DAF. And there may be created one or more additional connections with members of the DAF to support distributed RIB operations.

2.5.3 Data transfer

There are different data transfers that can occur. The first one is interactions with RIB Daemon that provides local access to shared objects stored in RIB database. This should be the vast majority of traffic. The APs may use the local RIB as a local memory. There should be more approaches of actualization of objects in the local RIB. As was said in section 2.3, RIB Daemon can reduce the amount of data to be send. Some applications also do not need actual state of objects. In that case, we use CDAP Messages to create/delete, start/stop, write/read objects from RIB.

Another data transfer correspond to more traditional view of networked applications sending requests and responds. It is direct CDAP exchanges between APs. Or it can use encapsulating data directly to Protocol Data Units (PDU).

3. Design and Implementation

3.1 Design

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++ [9], the main concept is in modules that are hierarchicaly structured and communicating with each other. Therefore, Application Process Instance should be a module.

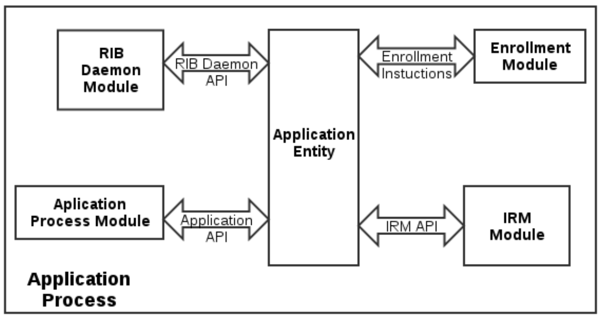


Figure 3. Design of Application Process

The figure contains all modules described earlier in chapter 2. Each module must have communication API with different purpose because the modules are communicating with each other to provide desired functionality. There is also 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.

3.2 Implementation

The modules in OMNeT++ are communicating through messages that may come from a module to another module or could be sent from a module to a module itself. In simulator, we use another approach, the communication between modules are also done by emiting signals just like in Qt [10].

All early mentioned tasks of Application Process should be seen as a modules in Figure 4.

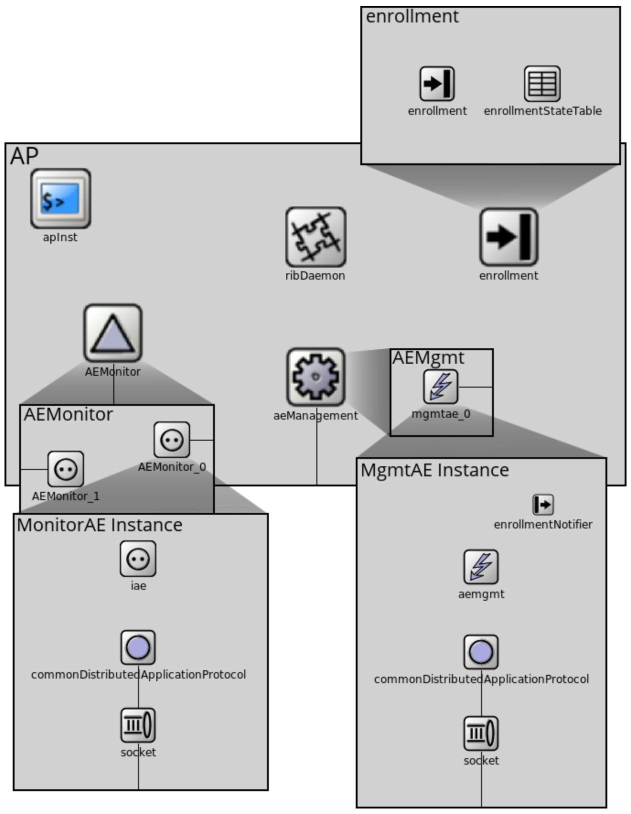


Figure 4. Structure of implemented modules in RINA Simulator

3.2.1 Application Programming Interface

At the top of the application is Application Process instance module, this module is implemented to provide an usable application API. This API is inspired by API described in document [4]. The OMNeT++ is discrete event based simulator. Hence, there is a difference between implemented and specified API. All API calls are implemented in AP Instance and AP Instance emits signals to AE Instance. While creating application connection API call also dynamicaly create requested AE instance that manages all sent and received messages.

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

3.2.2 Application Entity

Application Entity reacts on signals received from AP Instance API. Reaction on signals should be programmed as callback function. This gives an opportunity to create user specified AEs. The AE have also API to communicate with RIB Daemon. And also with Socket module that is responsible for buffering sent and received messages. More detailed Socket description is out of the scope of this paper.

3.2.3 Enrollment

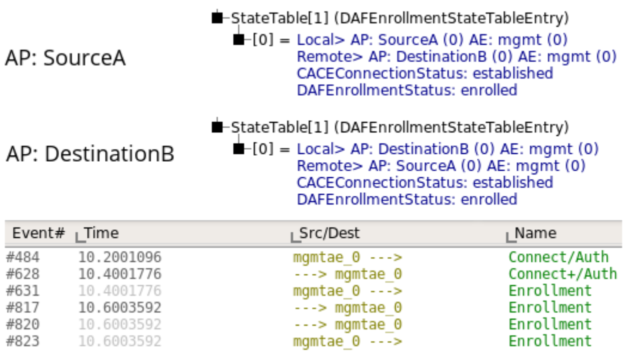
Enrollment module dynamically create Management Application Entity Instances and manages creating of Application Connection, CACE phase and Enrollment Phase. Enrollment module include finite state machines for both phases as well as for both roles (requestor and responder).

Management Application Entity is special AE only to use as communicator controlled by DAF Management tasks such as Enrollment.

4. Testing and Validation

The behavior of implemented model is deterministic. The Enrollment and the CACE phase part of model was tested on five different simulation examples repeatedly with the same result. The main factors on which the emphasis was that each side react on a received message by the transition to the appropriate state. And wheather each side answers with the expected message. Also if API calls dynamically create all required modules and submodules.

In the Figure 5, there should be seen final states of



```
AP: SourceA          StateTable[1] (DAFEnrollmentStateTableEntry)
                       [0] = Local> AP: SourceA (0) AE: mgmt (0)
                             Remote> AP: DestinationB (0) AE: mgmt (0)
                             CACEConnectionStatus: established
                             DAFEnrollmentStatus: enrolled

AP: DestinationB     StateTable[1] (DAFEnrollmentStateTableEntry)
                       [0] = Local> AP: DestinationB (0) AE: mgmt (0)
                             Remote> AP: SourceA (0) AE: mgmt (0)
                             CACEConnectionStatus: established
                             DAFEnrollmentStatus: enrolled

Event#  Time          Src/Dest            Name
#484    10.2001096    mgmtae_0 --->       Connect/Auth
#628    10.4001776    ---> mgmtae_0       Connect+/Auth
#631    10.4001776    mgmtae_0 --->       Enrollment
#817    10.6003592    ---> mgmtae_0       Enrollment
#820    10.6003592    ---> mgmtae_0       Enrollment
#823    10.6003592    mgmtae_0 --->       Enrollment
```

Figure 5. Enrollment communication and final states

both APs (AP: SourceA and AP:DestinationB) righ after CACE and enrollment proceeded. The successfully created application connection is indicated by state `CACEConnectionStatus:established` and the enrollment by state `DAFEnrollmentStatus:enrolled`. The figure also shows send and received messages. This graph of communication is taken from first AP: SourceA that participate in communication as requestor. Simple scenario with only two hosts and communicatio between them was used here.

The other parts was tested in the similar way as the Enrollment and the CACE. All requested modules was successfully dynamically created and reacted as it was designed.

5. Conclusions

This paper is focused on design and implementation of Application Processes of Distributed Application Facility (application layer) in RINASim. Application Process contains couple of modules such as RIB Daemon, Enrollment, Application Process itself and Application Entities. These modules was not provided 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 significant part of the work is also design and implementation of API that provides easier and uniform creation of applications in RINASim. It also provides better understanding of how to create distributed applications in RINA Architecture.

The model was validated agains specifications and communications with author of RINA Architecture. The implementation of the model was successfully validated as deterministic.

The implementation should be part of upcoming official version of RINASim. RINASim is used world wide by many researchers to help understand how RINA works and to experiment with its model. Simulator is still in development.

The future work should be focused on design of RIB Daemon policies to distribute and manage objects in RIB. Also to create more dynamic creation of every layer.

Acknowledgements

This work is done as part of PRISTINE EU-7FP-ICT project. I would like to thank Ing. Vladimír Veselý and Ing. Marcel Marek for their support and suggestions while this work.

References

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

[2] John Day. Patterns in network architecture - recursive ipc network architecture - the interina reference model - part 1: Basic concepts of distributed systems, 2013.

[3] John Day. Recursive ipc network architecture - the interina reference model - part 2: Distributed applications - chapter 1: Basic concepts of distributed applications, 2013.

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

[5] John Day. Patterns in network architecture - recursive ipc network architecture - basic enrollment specification, 2012.

[6] Eleni Trouva Steve Bunch, John Day. Patterns in network architecture - recursive ipc network architecture - common application connection establishment phase (cacep). http://www.stud.fit.vutbr.cz/~xjerab18/CDAPSpec.pdf, 2012.

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

[8] Vladimír Veselý. A new dawn of naming, addressing and routing on the internet, 2016.

[9] Omnet++ community site. http://www.omnetpp.org. Accessed: 2015-1-9.

[10] Qt site. http://www.qt.io/. Accessed: 2016-4-10.