

## **Title:** Running RINA on GENI

### **Overview:**

Recursive InterNetwork Architecture (RINA) is a new network architecture based on the fundamental principle that networking is Inter-Process Communication (IPC). RINA separates mechanisms and policies, and enables policy-based dynamic service composition. This tutorial introduces the basic elements of the Recursive InterNetwork Architecture (RINA) (<http://csr.bu.edu/rina/>) and describes how to setup RINA experiments on GENI nodes.

### **Prerequisites:**

- A basic understanding of GENI concepts such as those covered in the Introduction to GENI and Experimentation using GENI tutorial (<http://groups.geni.net/geni/wiki/GEC19Agenda/IntroToGENI>).
- Comfortable with running your own experiments on GENI resources.
- Verify that you are able to log into the GENI Portal (<https://portal.geni.net>).
- A basic familiarity with using a UNIX command line.
- A laptop with Mac OS or Linux OS. For Windows users, install Cygwin, or a recent version of Virtual Box (<https://www.virtualbox.org/>) with an Ubuntu image.
- A basic understanding of the Recursive InterNetwork Architecture (RINA: <http://csr.bu.edu/rina>).

### **Tools:**

- GENI Portal.
- UNIX command line.

### **Where to get help:**

- For support and general questions about ProtoRINA, please contact the ProtoRINA team at [protorina-developers@cs.bu.edu](mailto:protorina-developers@cs.bu.edu).
- You are welcome to subscribe to ProtoRINA users mailing list at <http://cs-mailman.bu.edu/mailman/listinfo/protorina-users>

## Experiment 3: Dynamic DIF formation

### Tutorial Instructions:



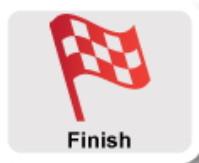
#### Part1: Design/Setup

- Design the Experiment
- Establish the Environment



#### Part2: Execute

- Login to VMs
- Import configuration files
- Start IDD and DNS
- Start Node2
- Start Node3
- Start Node1
- Analysis



#### Part3: Finish

## Experiment 3: Dynamic DIF formation

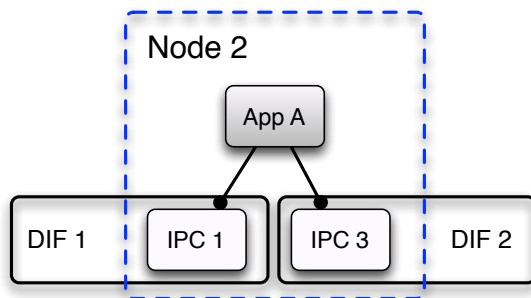
### Part1: Design/Setup:



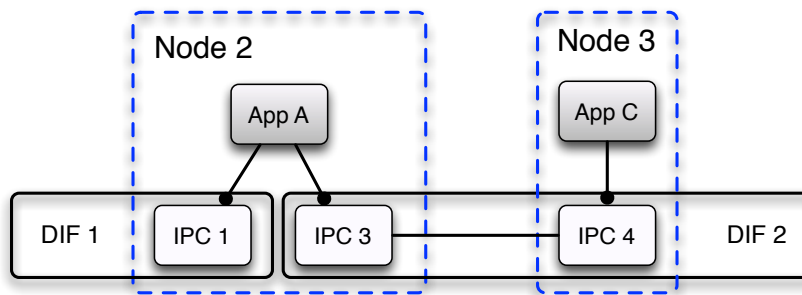
#### 1. Design the Experiment

In this experiment, we use three RINA nodes (Node 1, Node 2 and Node 3).

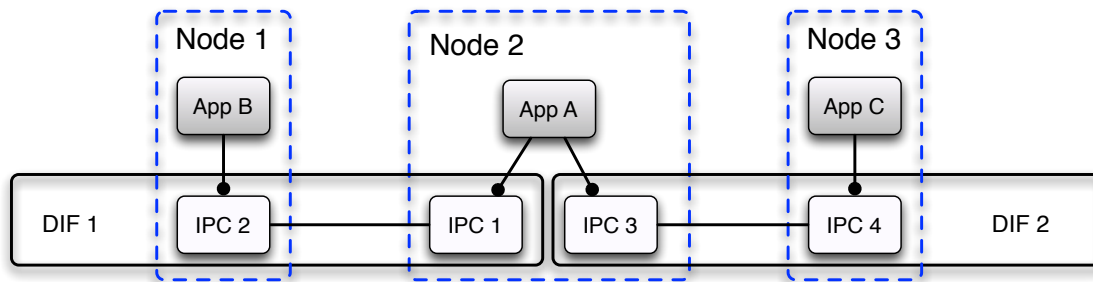
First we start Node2. It has two IPC processes (IPC1 and IPC3), and they belong to DIF 1 and DIF 2, respectively. AppA uses IPC1 and IPC3 as its underlying IPC processes.



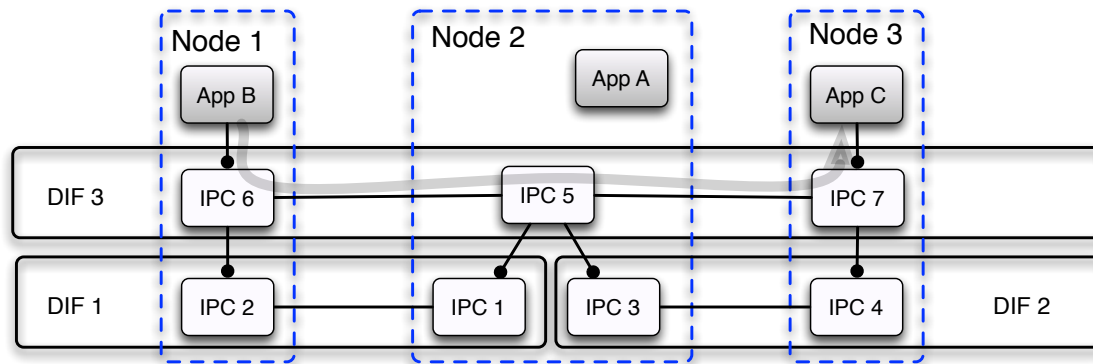
Second we start Node3. It has one IPC process (IPC4), and one application process AppC, which uses IPC4 as its underlying IPC process. IPC4 will join DIF2 through IPC3.



Third we start Node 1. It has one IPC process (IPC2), and one application process AppB, which uses IPC2 as its underlying IPC process. IPC2 will join DIF1 through IPC1. Then AppB wants to allocate a connection to AppC. However there is no common underlying DIF through which they can communicate. So AppB sends a request to AppA, which is registered as a relay service.



AppA creates DIF3 with IPC5 as the first member on Node2, then asks AppB and AppC to create a new IPC process (IPC6 and IPC7) to join DIF3 through IPC5. After that AppB is able to successfully create a connection to AppC via DIF3, and starts sending simple messages to AppC.



## 2. Establish the Environment

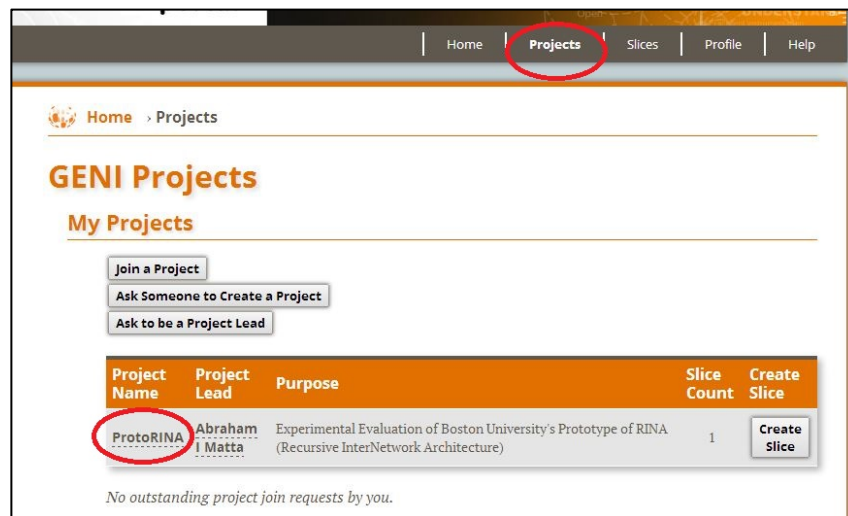


**Skip this section if you have already established your environment for RINA**

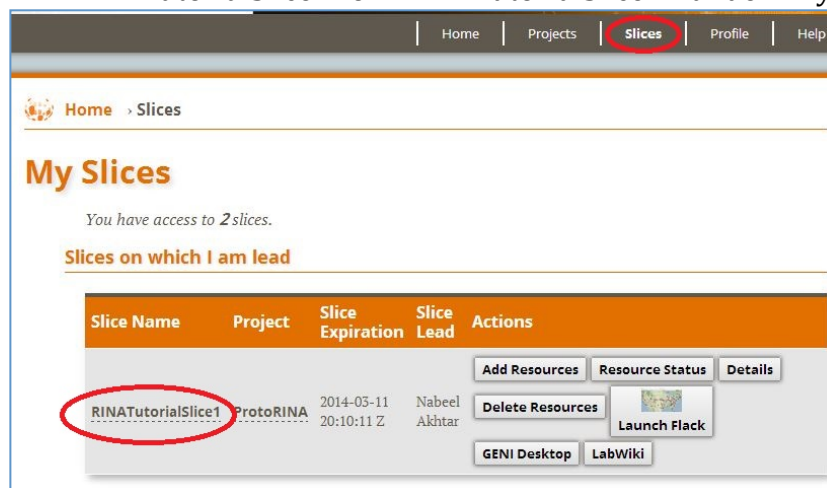
### 2.1. Pre-work:

- Ensure SSH keys are setup. If your SSH keys are not setup before, do the following steps:
  - Generate an SSH Private Key on the *Profile* page on the GENI portal
  - Download Private Key to `~/Downloads`
  - Open terminal and execute
 

```
$ mv ~/Downloads/id_geni_ssh_rsa ~/.ssh/
$ chmod 0600 ~/.ssh/id_geni_ssh_rsa
$ ssh-add ~/.ssh/id_geni_ssh_rsa
```
- Ensure you are part of the project “ProtoRINA”.
  - To check this, go to [portal.geni.net](https://portal.geni.net) -> Projects. You should be able to see project “ProtoRINA” under “My Projects”.



- Ensure that you are added to the slice by the lead.
  - To check this, go to portal.geni.net -> Slices. You should be able to see slice “RINATutorialSlice1” or “RINATutorialSlice2” under “My Slices”.



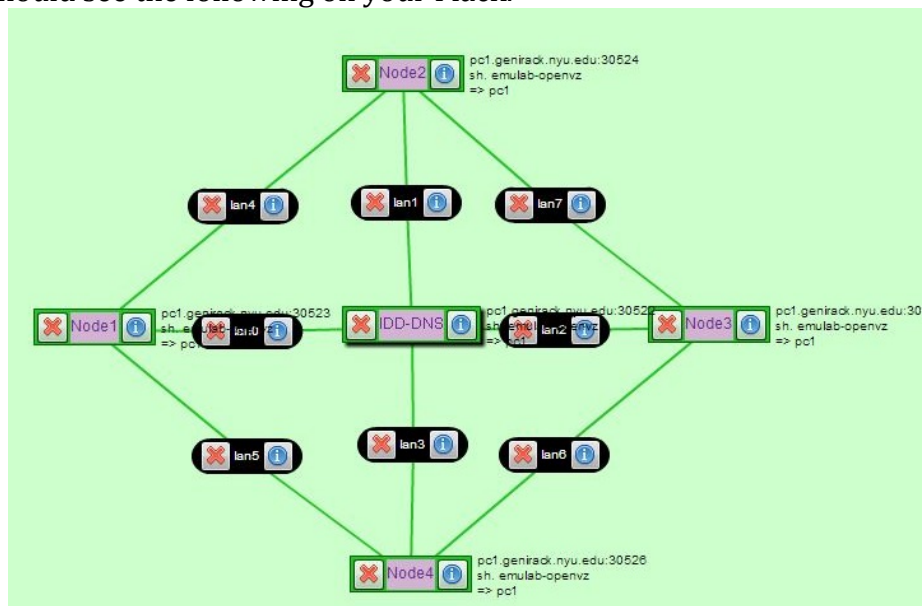
## 2.2. Check Resources:

- Click on “Launch Flack” on your slice. That might take a few minutes to display the whole GUI.



DO NOT DELETE RESOURCES!!!

- You should see the following on your Flack.



## Part2: Execute:




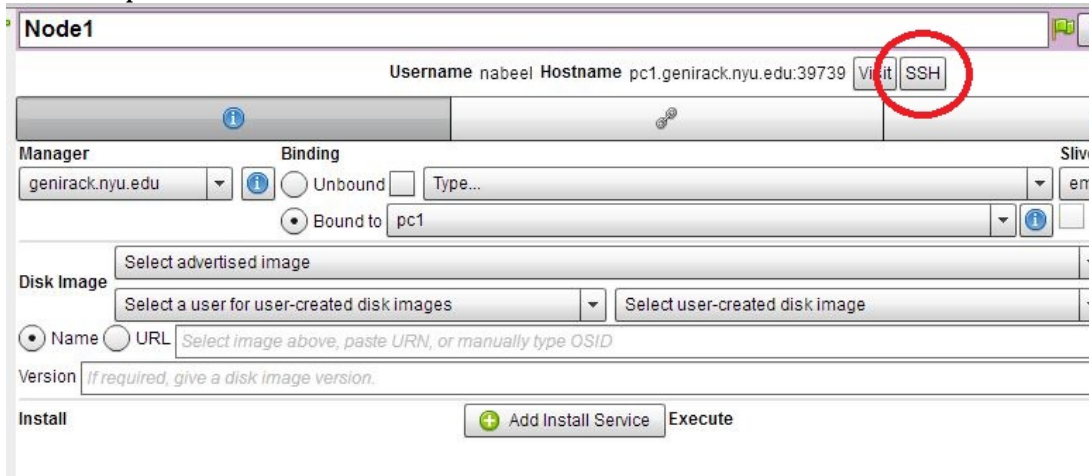
### 1. Login to VMs

In this experiment, we will use four VMs (named Node1, Node2, Node3 and IDD-DNS). You can see these four VMs on Flack.

- a) To login to a VM, click on  on “Node1” as shown below.



- b) Now, Click on  link as shown below. A new terminal window will pop up.



To ssh from the command line, do the following (substituting with values shown on the screen).

```
ssh USERNAME@HOSTNAME -p PORT
```

- c) Go back to Flack, and ssh to the other three VMs (Node2, Node3 and IDD-DNS) following the instructions given above.  
d) Make sure you open two terminal windows for “IDD-DNS”. We will run the IDD process and DNS process separately using two terminal windows.



You should have 5 terminal windows now: one for Node1, one for Node2, one for Node3 and two for IDD-DNS.



The IDD process provides RINA directory services

## 2. Import configuration files

Now you will import configuration files to your home directory for four VMs respectively.

For each VM, copy **your group** files to your local directory as shown below. You can find your group number in the worksheet provided to you.

```
cp -r /users/Tutorial/master/group1/ ~
```



Make sure you copy data for your **own group** provided in the handout.

## 3. Start IDD and DNS

We will start the IDD process and DNS process now.

- a) To start DNS, go to directory with the DNS properties file using one of the IDD-DNS terminal windows:

```
cd group1/IDD-DNS/DNS/
```

Now run following command to start DNS.

```
java -jar /users/Tutorial/master/IDD-DNS/DNS/DNS.jar dns.properties
```

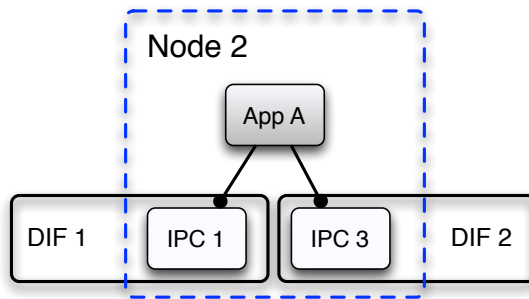
- b) To run IDD, go to directory with the IDD properties file using the second IDD-DNS terminal windows:

```
cd group1/IDD-DNS/IDD/
```

Now run the following command to start IDD:

```
java -jar /users/Tutorial/master/IDD-DNS/IDD/IDD.jar idd.properties
```

## 4. Start Node2



Now we will start Node2 process on the first VM (Node2).

- a) In the terminal for Node2, go to directory for Node2 as shown below:

```
cd group1/Configurations/exp3/Node2/
```

- b) Start Node2 process using the following command:

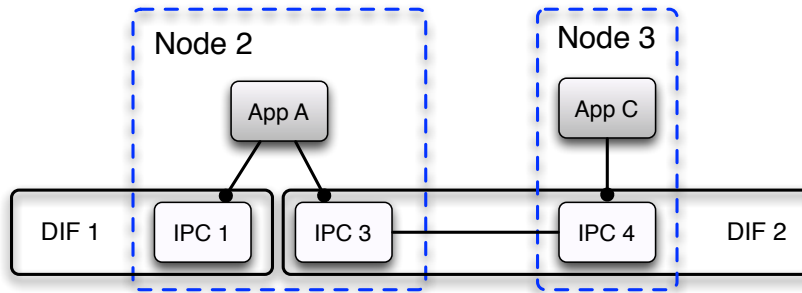
```
java -jar /users/Tutorial/master/RINANode.jar Node2.properties
```

At this point, your Node2 should be up and running.



## 5. Start Node3

Now we start Node3. It has one IPC process (IPC4) and one application process (App C). IPC4 joins DIF2 through IPC3.



a) In the terminal for Node3, go to directory for Node3 as shown below:

```
cd group1/Configurations/exp3/Node3
```

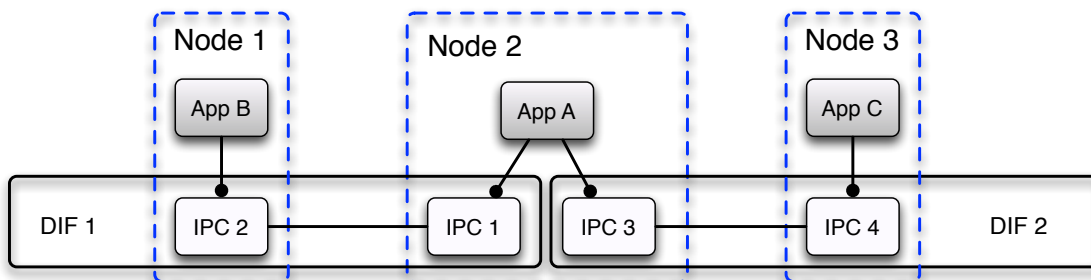
b) Start Node3 process using the following command:

```
java -jar /users/Tutorial/master/RINANode.jar Node3.properties
```

At this point, your Node3 should be up and running, and it will join DIF2 through IPC3 on Node2.

## 6. Start Node1

Now we start Node1. It has one IPC process (IPC2), and one application process AppB, which uses IPC2 as its underlying IPC process. IPC2 joins DIF1 through IPC1. Then AppB will allocate a connection to AppC. However there is no common underlying DIF through which they can communicate. So AppB sends a request to AppA, which is registered as a relay service for AppC.



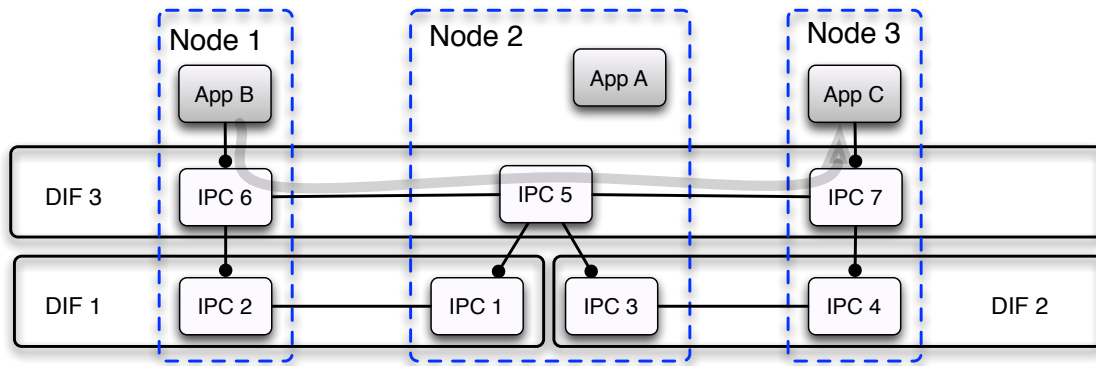
a) In the terminal for Node1, go to directory for Node1 as shown below:

```
cd group1/Configurations/exp3/Node1
```

b) Start Node1 process using the following command:

```
java -jar /users/Tutorial/master/RINANode.jar Node1.properties
```

AppA creates DIF3 with IPC5 as the first member on Node2, then asks AppB and AppC to create a new IPC process (IPC6 and IPC7) to join DIF3 through IPC5. After that AppB is able to successfully create a connection to App3 via DIF3, and starts sending simple messages to AppC.



## 7. Analysis

Now we will analyze the successful dynamic formation of DIF3 by parsing the log file at Node2.

- a) Stop Node1, Node2 and Node3.

You can terminate the node process by pressing “Ctrl+c” on the keyboard.



**DO NOT CLOSE TERMINAL WINDOWS**

- b) We will look at the simple messages sent from AppB to AppC. Type following in the terminal for Node3 to parse the log file.

```
/users/Tutorial/master/LogChecker.sh -i rinaBU.log -o SimpleApp
```

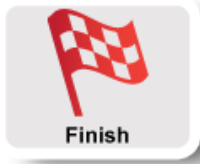
- c) This will generate a log file for all the simple application messages. To display this log file, type the following.

```
cat SimpleApp.log
```

Sample output is shown below:

```
[nabeel@Node3 Node3]$ cat SimpleApp.log
2014-03-03 16:00:43644 Regular handler started
2014-03-03 16:00:43820 Message received, and content is Hello, world: 0
2014-03-03 16:00:44825 Message received, and content is Hello, world: 1
2014-03-03 16:00:45779 Message received, and content is Hello, world: 2
2014-03-03 16:00:46806 Message received, and content is Hello, world: 3
2014-03-03 16:00:47807 Message received, and content is Hello, world: 4
2014-03-03 16:00:48832 Message received, and content is Hello, world: 5
2014-03-03 16:00:49848 Message received, and content is Hello, world: 6
2014-03-03 16:00:50814 Message received, and content is Hello, world: 7
2014-03-03 16:00:51844 Message received, and content is Hello, world: 8
2014-03-03 16:00:52845 Message received, and content is Hello, world: 9
```

### Part3: Finish:



You are done with the third experiment. Make sure you terminate all the processes on Node1, Node2, Node3 and IDD-DNS nodes.



**DO NOT CLOSE TERMINAL WINDOWS**

Let's proceed to the fourth experiment. We will use the same resources for the fourth experiment.