# IP Simulator User Guide

Emmanuel Nataf

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *IP Simulator*
# *User guide*

Emmanuel Nataf

**N° 6416**

Janvier 2008

Thème COM


ISSN 0249-0803 ISRN INRIA/RT--6416--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

# IP Simulator
# User guide

Emmanuel Nataf*

Thème COM — Systèmes communicants
Équipe-Projet Madynes

Rapport technique n° 6416 — Janvier 2008 — 19 pages

**Abstract:** This simulator allows to construct IPv4 network topologies with routers and end hosts. The focus of the simulator is on routing table configuration. The simulator could also run user network programs written with a java.net UDP like API.

**Key-words:** network, protocol, IP, UDP, simulator, java

* Maître de conférence - Université Nancy2

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

# Simulateur IP

**Résumé :** Ce simulateur permet de construire des topologies de réseaux IPv4 avec des routeurs et des machines terminales. Le but de ce simulateur est la configuration des tables de routage. Le simulateur peut galement exécuter des programmes utilisateurs écrits avec une interface proche de java.net pour UDP.

**Mots-clés :** réseau, protocole, IP, UDP, simulateur, java

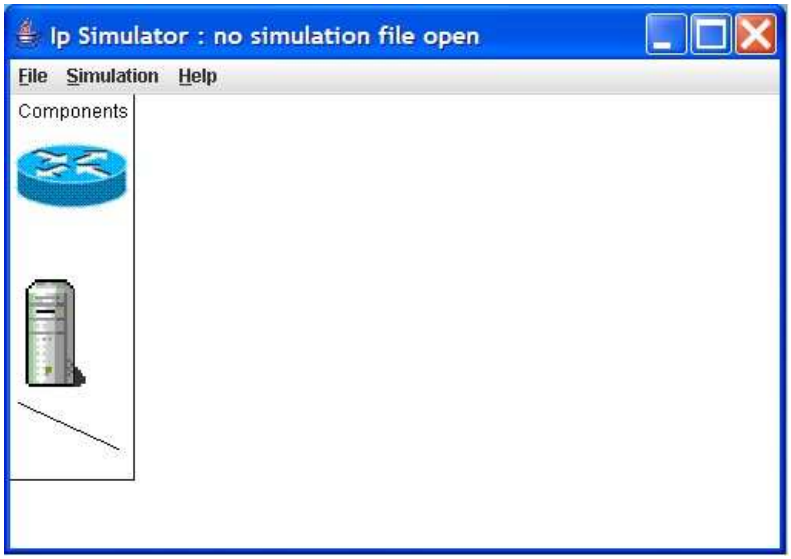



Figure 1: Graphical user interface

# 1 Introduction

The IP simulator allows to

- design a network topology from router and end hosts related by communication link;
- give IP addresses to network interfaces;
- define routing tables for each node;
- program network applications and load them in the simulator;
- save, load, rename topologies and applications.

The simulator is written in java language and could be launch with the `simulip.jar` file. Environment variable `CLASSPATH` must point on this file.

# 2 Topologies design

The figure 1 shows the simulator starting graphical user interface. One can see a *Component* part on the top left of the window. There are three component types from top to bottom as **router, PC** and **link** that are used to construct topologies.

A network topology is usually composed of severals **routers** related each other or to **PC** by **links**. Note that one should relate a **PC** with only one **link**. Mouse actions are "usuals" :

- Select a **routeur** or a **PC** with a simple left click (press and release) on the icon and do an other simple click to put the component on the map.

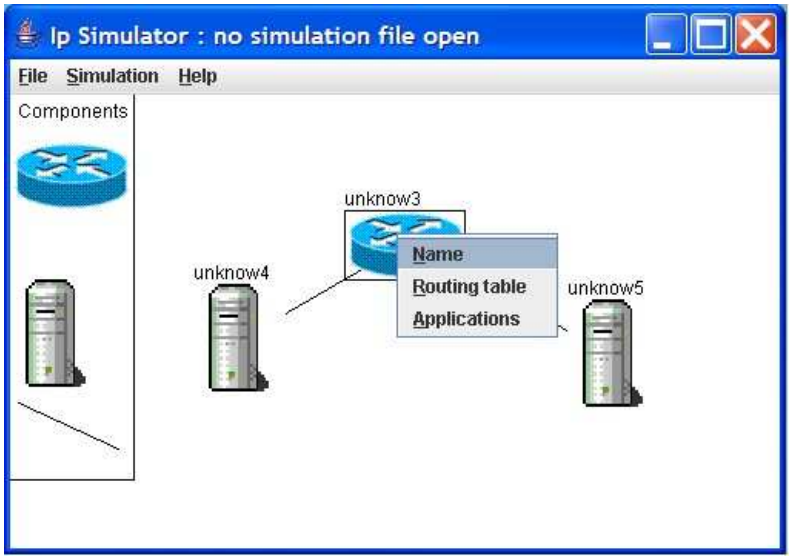



Figure 2: Node network menu

- To link two components do one left click on the link a the bottom of the component part and do an other left click on an existing component (not in the component part) and finally do a last click on an other component.
- Delete a component with first select it by a left click and use the back space or suppr key to delete it. Related **links** are also deleted if a node is selected.
- Move a component (a node) by a left press action on the component and drag it to the new position. Release the mouse to put it.

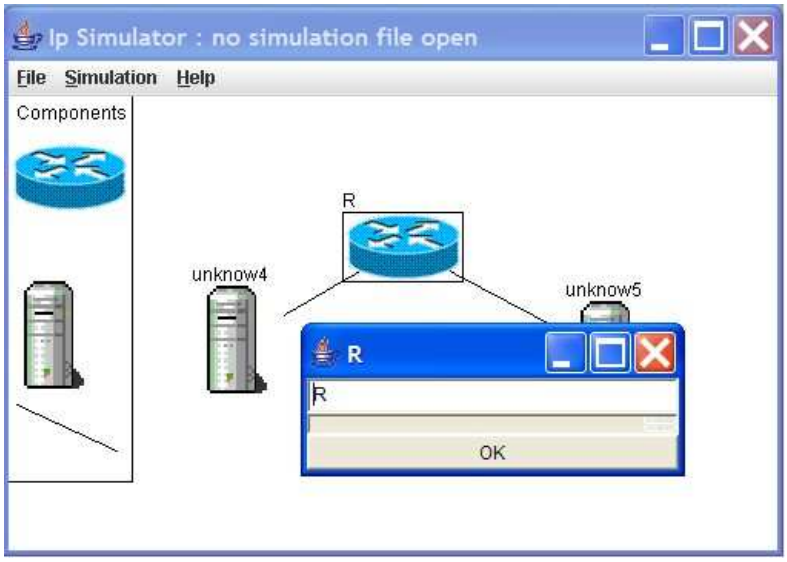
## 2.1 Internal parameters

Internal parameters are inside components and they are of several types and are accessible by a contextual menu (right mouse button, figure 2) on the component.

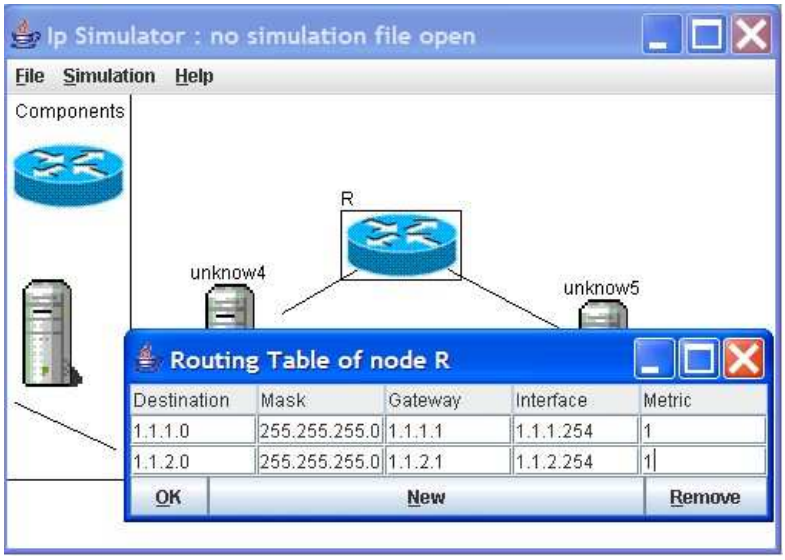
### 2.1.1 routeur and PC parameters

The menu has three parts :

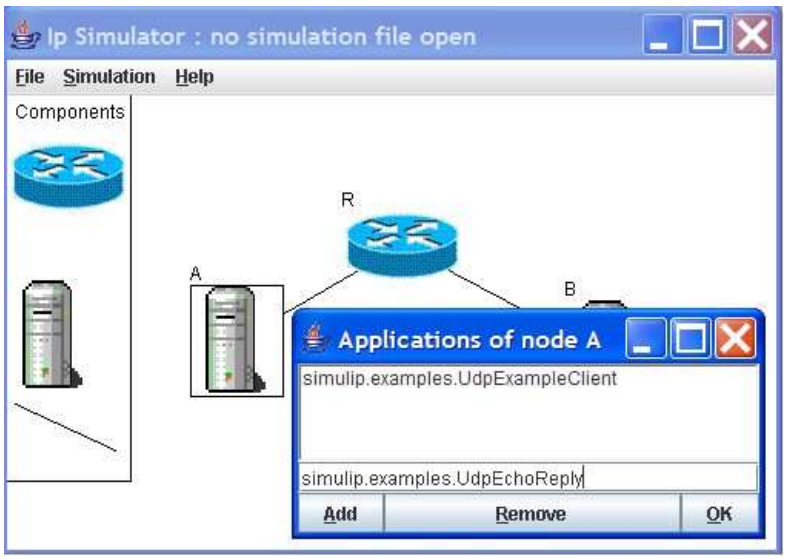
- `Name` : to change the name of the component (unique in the topology)



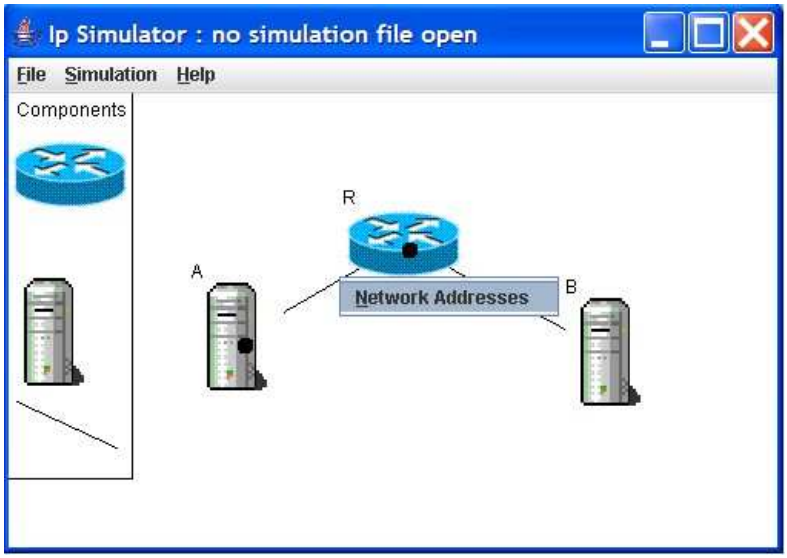

- `Routing table` : to create or update the component routing table. *New* and *Remove* button to create and remove routing table entries. Existing routes could be updated.



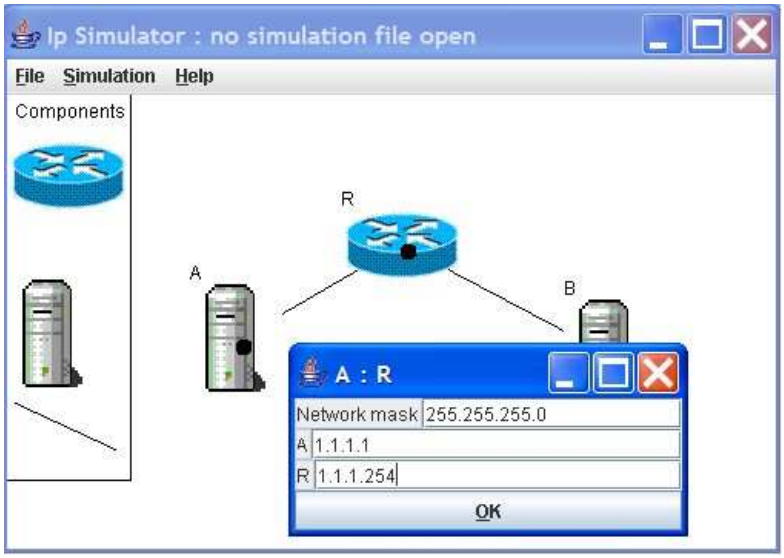

- `Application` : to add applications to the component. Be carefull that the name of the application is its java class name (for example : `simulip.examples.UdpEchoSend`).




### 2.1.2 link parameter




The menu has only one choice that is used to set network parameters of a **link**. Such parameters are network addresses of each end and the network mask.




## 2.2 Simulation management

The menu bar of the simulator has common functionalities on files (open, save, save as, new and quit). The file format is an XML document. It is not necessary to directly edit this file but it could be more convenient with large topologies.

Simulation are launched by the *Simulation* menu. A new window (figure 3) is created with the constructed topology.

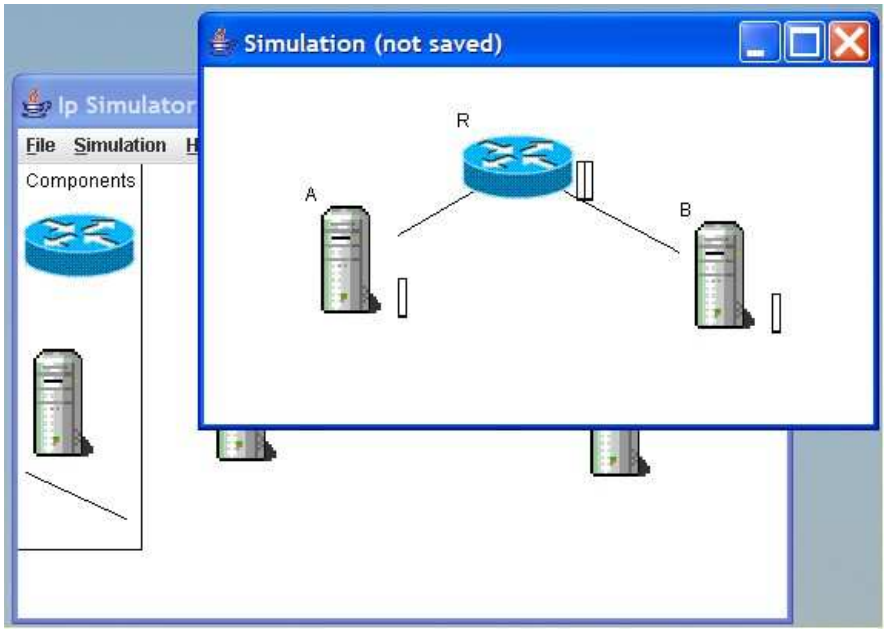
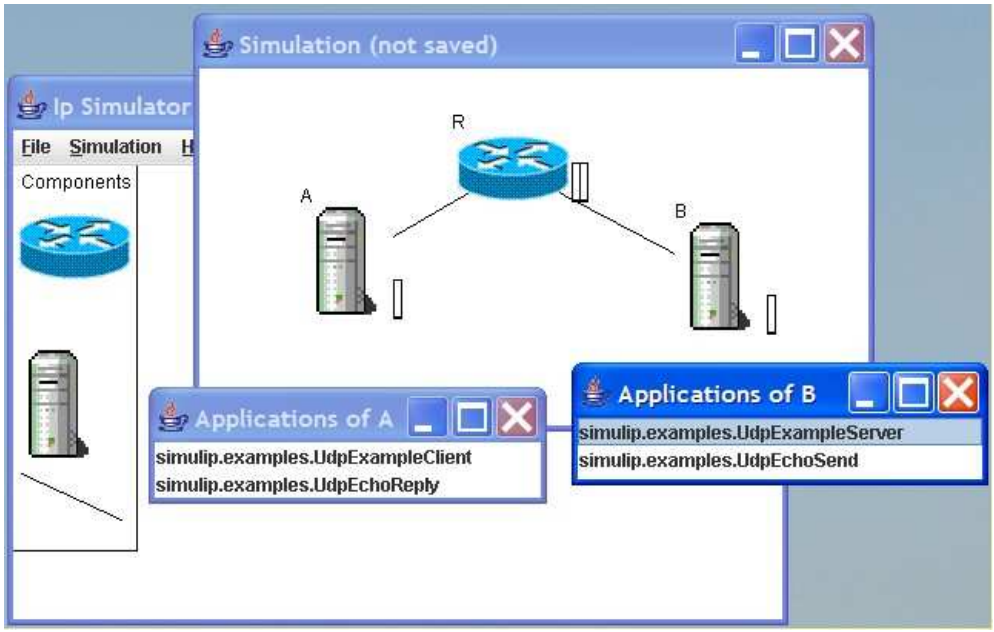



Figure 3: Simulation window

Contextual menus are differents inside the simulation window ;

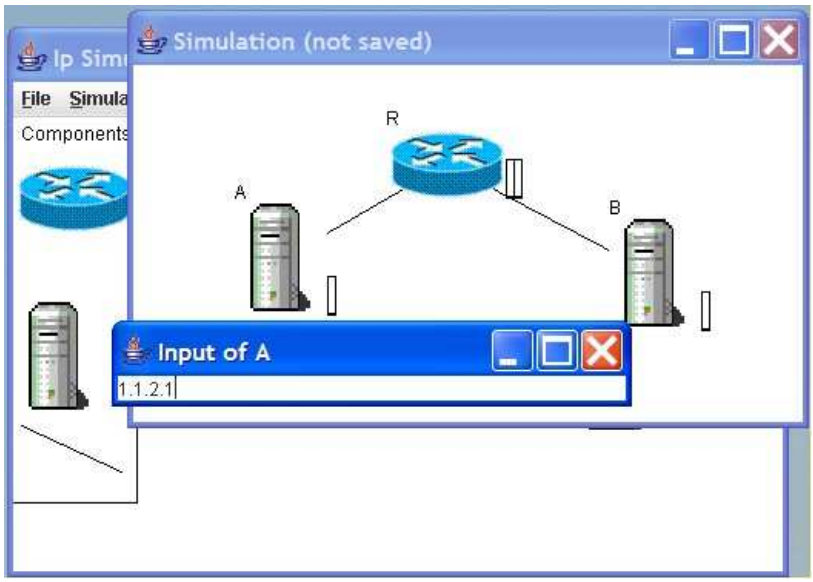
- node names are no more updatable as routing tables
- application selection launch it immediatly (and the application window choice disapear)



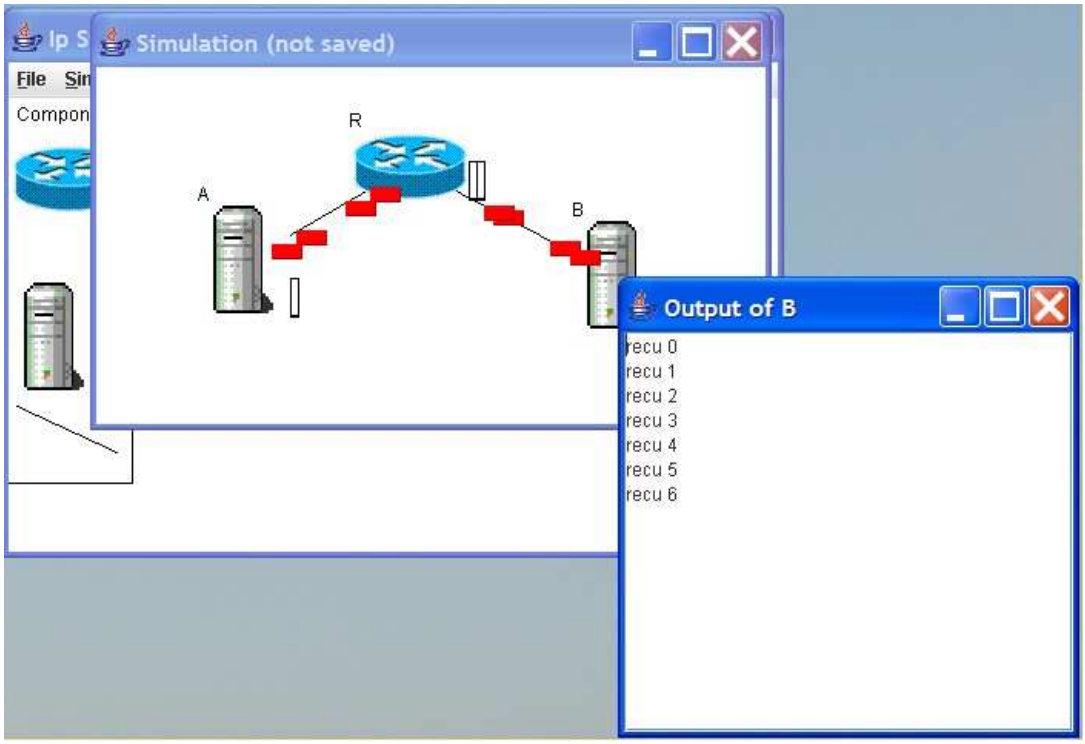

## 2.3 Application design

In order to be launch inside the simulator, an application must follow theses rules:

- import `java.net.*` and `simulip.net.*` packages
- the main class extends the `Application` class
- the starting point of application is the `run()` method
- input and output are provided by
  - `system.in.read()` that return a java `String` java. An input window is created and will disapear when Enter key is used.




– `system.out.println(String)` shows the string on a new window (one for each application).




**Warning** : it is `system` and not `System`

- Network input and output are made with UPD protocol. `simulip.net.DatagramSocket` and `simulip.net.DatagramPacket` are like these of `java.net`. It could be easy to change an application for the simulator to an application on real network.

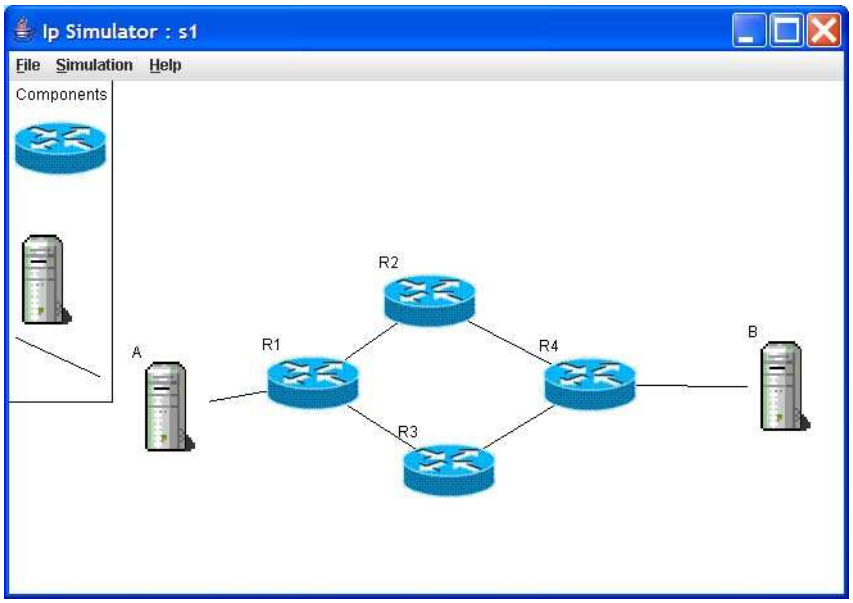
# 3 Routing exercices

Do the following topologies with addresses restrictions. Use couple of given applications (already done, see code in appendices) to test the routing.

- Client / Server where client continoulsy send packets and where the server return them to the client. With classes :
  - `simulip.examples.UdpExampleClient`
  - `simulip.examples.UdpExampleServer`

  Echo Send / reply where there is only on packet sended and returned. With classes :
  - `simulip.examples.UdpEchoSend`
  - `simulip.examples.UdpEchoReply`

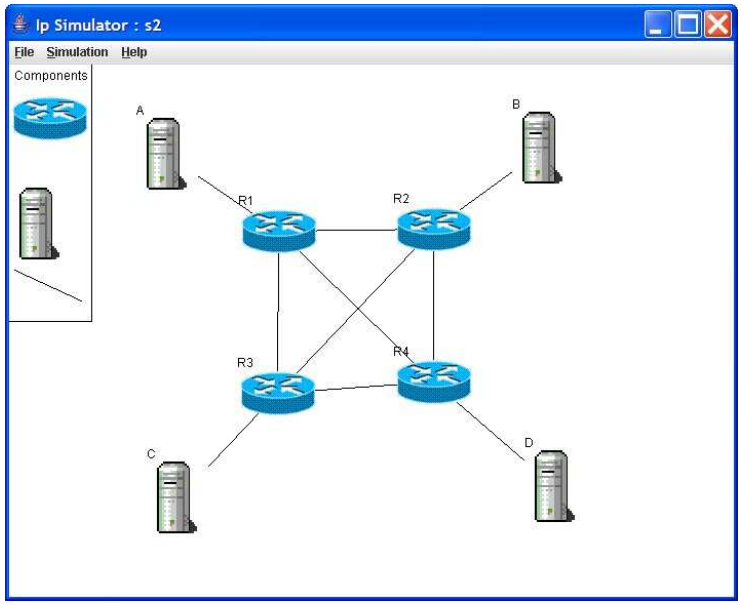
1. Network between A and R1 has the address 1.1.1/24, A has the number 1 and R1 the 254. For the network between B and R4, take 1.1.2/24; B with the number 1 and R4 with 254. Networks between routers are to be defined. The route must follow R2 from A to B and R3 from B to A.




2. Follow instructions :

| network | addresse |
|---|---|
| A and R1 | 1.1.1/24 |
| B and R2 | 1.1.2/24 |
| C and R3 | 1.1.3/24 |
| D and R4 | 1.1.4/24 |

Network between routers are to be defined. Try to do the best routes (the shortest).




# 4 Programmation exercice : a P2P chat

## 4.1 Outline

The chat application allows two users on separate **PC** to exange text messages without having any knowledge of IP network addresses and UDP port number they are used[1]. These informations are available in a server which network and tranport addresses are known by each user.

There are two types of users :

- users that register themself to the server by given it an user name and a port number;
- users (registered or not) that want communicate with a registered user.

The application works as following :

1. The server is launched on a **PC**.
2. An user application register itself with its name and its port number. When the server confirms this registration the user waits for communication on the given port.

[1] but at the begining for registered users

3. An other appication asks the server to return information for a given user name. The response sended by the server should allow communication with the waiting registered user.

4. Users could exchange messages.

## 4.2 Working

The two user applications are to be programmed as the server is already done (**P2PServer** in appendices). Informations exchanged between users and server are specified bellow:

- Server is waiting on the 530 port number and use UDP.
- Registration of an user must contain (in this order) :
  1. the char `r` on one byte,
  2. the port number used to communicate with other users, on two bytes with the most significant byte in the first position,
  3. the size of the user name on one byte,
  4. the user name (one byte by char).
- Server returns one byte with the char `y`. User name length should not exeed 25 bytes and the port number should be between 0 and 65535. If the server does not understand the query, it replies with a `n` char. Users application have to ensure correctness of such parameters.

  User application should wait on its port number after receiving the server reply, with UDP.
- The request from an user should contains (in this order) :
  1. the char `c` on one byte,
  2. the size of the user name (the one to which communicate) on one byte,
  3. the user name (te one to which communicate).
- The server returns with :
  1. a byte char `y`,
  2. the IP address of the asked user, on 4 bytes (most significant at the first position)
  3. the port number of the asket user, on two bytes beginning with the most significant one.
- If the server does not understand the request it returns the single char `n`.

## 4.3 Extending work

1. Change the server and client in order to allow only registered users
2. Change server and client in order to allow more than two users communicating each other.

# 5 Appendices

## 5.1 simulip.examples.UdpExampleClient

```
package simulip.examples;
import simulip.net.*;
public class UdpExampleClient extends  Application {
public UdpExampleClient(){};
 public void run(){
    try{
           int count = 0;
           byte[] data = String.valueOf(count).getBytes();
           String ip = system.in.read();
           simulip.net.DatagramSocket d = new simulip.net.DatagramSocket(this);
           simulip.net.DatagramPacket p =
                       new simulip.net.DatagramPacket(data,data.length,ip,53);
           while(true){
                      try{
                            Thread.sleep(400);
                      }
                     catch(Exception e){}
                     d.send(p);
                     count++;
                     byte[] nd = String.valueOf(count).getBytes();
                     p.setData(nd);
           }
       }
      catch(Exception e){
           system.out.println(e.getMessage());
      }
 }
}
```

## 5.2 simulip.examples.UdpExample.Server

```
package simulip.examples;
import java.net.*;
import simulip.net.*;
public class UdpExampleServer extends Application{
private byte[] data = new byte[5];
 public void run(){
     try{
          simulip.net.DatagramSocket d = new simulip.net.DatagramSocket(this,53);
          simulip.net.DatagramPacket p = new simulip.net.DatagramPacket(data,5);
          while(true){
                    d.receive(p);
                    p.setAddress(p.getAddress());
                    p.setPort(p.getPort());
                    system.out.println("recu " + new String(p.getData()));
                    d.send(p);
          }
     }
    catch(BindException b){
               system.out.println(b.getMessage());
    }
 }
}
```

### 5.3 simulip.examples.UdpEchoSend

```
package simulip.examples;
import simulip.net.*;
public class UdpEchoSend extends Application{
  public void run(){
       try{
        String ipdest;
        ipdest = system.in.read();
        int count = 0;
        byte[] data = String.valueOf(count).getBytes();
        simulip.net.DatagramSocket d = new simulip.net.DatagramSocket(this);
        simulip.net.DatagramPacket p = new simulip.net.DatagramPacket(data,5,ipdest,54);
        while(true){
           d.send(p);
           d.receive(p);
           system.out.println("ack de : " + new String(p.getData()));
           p.setAddress(p.getAddress());
           p.setPort(p.getPort());
           count++;
          data = String.valueOf(count).getBytes();
          p.setData(data);
        }
      }
      catch(Exception e){
         system.out.println(e.getMessage());
      }
  }
}
```

### 5.4 simulip.examples.UdpEchoReply

```
package simulip.examples;
import simulip.net.*;
import java.net.*;
public class UdpEchoReply extends Application{
   private byte[] data = new byte[5];
   public void run(){
      try{
         simulip.net.DatagramSocket d = new simulip.net.DatagramSocket(this,54);
         simulip.net.DatagramPacket p = new simulip.net.DatagramPacket(data,5);
         while(true){
            d.receive(p);
            p.setAddress(p.getAddress());
            p.setPort(p.getPort());
            system.out.println(new String(p.getData()));
            d.send(p);
         }
      }
      catch (BindException b){
         system.out.println(b.getMessage());
      }
   }
}
```

## 5.5 P2Pserver

```
import simulip.net.*;
import java.net.BindException;import java.util.*;
import java.math.*;

public class P2Pserver extends Application {

 private Vector<Record> records = new Vector<Record>();
 private byte[] data = new byte[29];

 public void run(){
  try{
    simulip.net.DatagramSocket d = new simulip.net.DatagramSocket(this,530);
    simulip.net.DatagramPacket p = new simulip.net.DatagramPacket(data,13);
    while(true){
       d.receive(p);
       data = p.getData();
       if(data[0] == 'r'){
          system.out.println("recording request");
          NetworkAddress add = p.getAddress();
          system.out.println("from " + add.getStrAddress());
          byte[] pnum = new byte[3];
          pnum[0] = 0;
          pnum[1] = data[1];
          pnum[2] = data[2];
          BigInteger pbi = new BigInteger(pnum);
          system.out.println("chat at port " + pbi.toString());

          byte[] blname = new byte[1];
          blname[0] = data[3];
          BigInteger it = new BigInteger(blname);
          byte[] bname = new byte[it.intValue()];
          for(int i = 0; i < bname.length; i++)
             bname[i] = data[i + 4];
          String name = new String(bname);
          system.out.println("name : " + name);
          byte[] twobyteport = new byte[2];
          twobyteport[0] = pnum[1];
          twobyteport[1] = pnum[2];
          Record rec = new Record(add.getStrAddress(), twobyteport,name);
          records.add(rec);
          byte[] ack = new byte[1];
          ack[0] = 'y';
          p.setData(ack);
       }
       else if (data[0] == 'c'){
          system.out.println("contacting request");
          byte[] blname = new byte[1];
          blname[0] = data[1];
```

```
        BigInteger it = new BigInteger(blname);
        byte[] bname = new byte[it.intValue()];
        for(int i = 0; i < bname.length; i++)
        bname[i] = data[i + 2];
        String name = new String(bname);
        system.out.println("ask for : " + name);
        Enumeration<Record> er = records.elements();
        Record rec = null;
        boolean found = false;
        while(!found && er.hasMoreElements()){
           rec = er.nextElement();
           found = rec.name.equals(name);
        }
        if(!found){
           byte[] nack = new byte[1];
           nack[0] = 'n';
           p.setData(nack);
        }
        else{
           byte[] resp = new byte[7];
           try{
              resp[0] = 'y';
              byte[] add =
                NetworkAddress.toBytes((new NetworkAddress(rec.address)).getBits());
              resp[1] = add[0];
              resp[2] = add[1];
              resp[3] = add[2];
              resp[4] = add[3];
              resp[5] = rec.port[0];
              resp[6] = rec.port[1];
              p.setData(resp);
           }
           catch(NetworkAddressFormatException nafe){
           }
        }
     }
     else {
        byte[] nack = new byte[1];
        nack[0] = 'n';
        p.setData(nack);
        system.out.println("unknow request ");
     }
     p.setAddress(p.getAddress());
     p.setPort(p.getPort());
     d.send(p);
   }
}
catch(BindException b){
     system.out.println(b.getMessage());
}
```

```
 }

 private class Record {
   public String address;
   public byte[] port = new byte[2];
   public String name;
   public Record(String a, byte[] p, String n){
      address = new String(a);
      port = p;
      name = new String(n);
   }
 }
}
```

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803