Omnet++

Olivier FLAUZAC
olivier.flauzac@univ-reims.fr
http://www.flauzac.eu

Généralités

Omnet++

- Simulateur à événement discret
- Simulation
  - algorithmes distribués
  - protocoles
- Mode d'exécution
  - mode graphique
  - mode console

Ressources

- Site WEB
  - http://www.omnetpp.org
  - version actuelle 4.6
- Extension de omnet++ par l'intermédiaire de frameworks
  - réseaux IP
  - réseaux ad hoc
  - réseaux pair-à-pair
  - ...

Projet Omnet++

- Fichiers de description de topologie / réseau
  - fichiers .ned
- Fichier d'initialisation
  - fichier .ini
- Fichiers de code
  - fichiers d'en tête : .h
  - fichiers de code : .cc

Fichier .ned

- Description du réseau
- Description de chacun des composants qui forment le réseau
- Langage spécifique
  - code source
  - graphique

Fichier .ini

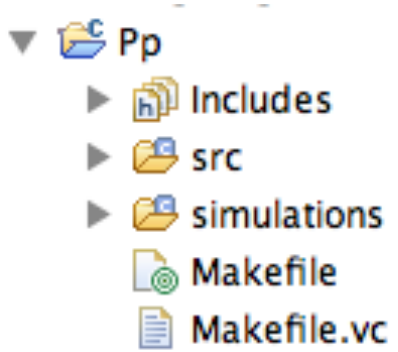
- Description du cas d'application
- Initialisation de paramètres de simulation
  - temps maximum d'exécution
  - nombre de noeuds
  - identifiants des noeuds
  - ...

Un premier projet :
Un Ping Pong

Ping Pong

- Principe :
  - Deux sites se renvoient un message à l'infini
- Réseau
  - deux sites
- Un site
  - un identifiant
  - un «port» en sortie
  - un «port» en entrée

Création d'un projet



- Omnet++ Project
  - name : Pp
  - Empty project with 'src' and 'simulations' folders

Création du réseau

- Définition générale du réseau
  - dossier simulation
  - fichier .ned
    - New Network Description File
    - Pp.ned
    - Empty NED file

simulations/Pp.ned

```
package pp.simulations;

import pp.*;
network Net
{
    submodules:
      Ping:Site{
          @display("p=50,50");
      }
      Pong:Site{
          @display("p=200,50");
      }
    connections:
        Ping.out  --> Pong.in;
        Pong.out --> Ping.in;
}
```

Définition du module site

- Module site :
  - noeud + comportement (fichier .h et .cc)
- dans le répertoire src
  - new Simple module
  - name Site.ned
  - création des trois fichiers
    - Site.ned , Site.h , site.cc

Définition du comportement

- Fichiers .h et .cc
- Définition des éléments liés (id)
- Définition de l'initialisation
  - lecture des paramètres id de omnetpp.ini
  - création d'un message
  - envoie d'un message

Exécution

- A la réception d'un message
  - afficher le contenu
  - créer un nouveau message
  - envoyer le message

src/Site.ned

```
package pp;

//
// TODO auto-generated module
//
simple Site
{
    parameters:
        int id;
    gates:
        input in;
        output out;
}
```

Fichier .h

```
#ifndef __PP_SITE_H_
#define __PP_SITE_H_

#include <omnetpp.h>

/**
 * TODO - Generated class
 */
class Site : public cSimpleModule
{
private:
  int id;  // rajouté
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

#endif
```

Fichier .cc

```
#include "Site.h"

Define_Module(Site);
using namespace std;

void Site::initialize()
{
    this->id = par("id").longValue();
    ev << "demarrage du site " << id << endl;
    if(id == 0){
       cMessage *m = new cMessage("depuis Ping");
       send(m,"out");
       ev << "Message envoye depuis " << this->id << endl;
    }
}
```

fichier .cc (fin)

```
void Site::handleMessage(cMessage *msg)
{
    ev << "Message recu au site : "<< this->id << endl;
    ev << "Contenu du message : " << msg->getName() << endl;
    delete msg;
    cMessage *m = NULL;
    if(id == 0){
        m = new cMessage("depuis Ping");
    }else{
      m = new cMessage("depuis Pong");
    }
    send(m,"out");
    ev << "Message envoye depuis " << this->id << endl;
}
```

Création de l'environnement (.ini)

- Création sur le projet
  - new initialization file
  - omnetpp.ini
  - Empty ini File
  - Select Network : pp.simulations.Net

Eléments

- Temps de simulation
- Identifiant des éléments (id)
- Création :
  - mode graphique
  - mode console

Fichier omnetpp.ini

```
[General]
network = pp.simulations.Net
sim-time-limit=100s
Net.Ping.id = 0
Net.Pong.id = 1
```

Compilation / exécution

- Compilation
  - Sur le projet
  - Build Project
- Exécution
  - Sur le projet
  - Run as - run Configurations
    - création d'une configuration au nom du projet
    - exécution

Extension du Ping Pong

Extension

- Deux sites différents
  - deux fichiers de description dans src
  - deux classes de comportement
- Changement des couleurs des sites

simulations/Pp2.ned

```
package pp2.simulations;
import pp2.*;

network Pp2
{
    submodules:
      S1:Ping{

      }
      S2:Pong{

      }
    connections:
        S1.out --> S2.in;
        S2.out --> S1.in;
}
```

src/ping.ned & src/pong.ned

```
package pp2;

simple Ping
{
    parameters:
        @display("i=misc/node_vs");
        int id;
    gates:
        input in;
        output out;
}
```

```
package pp2;

simple Pong
{
    parameters:
        @display("i=misc/node_vs");
        int id;
    gates:
        input in;
        output out;
}
```

src/pong.h

```
#ifndef __PP2_PONG_H_
#define __PP2_PONG_H_

#include <omnetpp.h>

/**
 * TODO - Generated class
 */
class Pong : public cSimpleModule
{
private:
    bool alternate;
    int id;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

#endif
```

src/ping.h

```
#ifndef __PP2_PING_H_
#define __PP2_PING_H_

#include <omnetpp.h>

/**
 * TODO - Generated class
 */
class Ping : public cSimpleModule
{
private:
    bool alternate;
    int id;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

#endif
```

src/ping.cc

```
#include "ping.h"
Define_Module(Ping);
using namespace std;
void Ping::initialize()
{
    this->id = par("id").longValue();
    ev << "Init in PING " << endl;
    getDisplayString().setTagArg("i",1,"red");
    this->alternate = false;
    cMessage *m = new cMessage();
    m->setKind(12);
    m->setName("from PING");
    send(m,"out");
}
void Ping::handleMessage(cMessage *msg)
{
    if(this->alternate == false){
        getDisplayString().setTagArg("i",1,"blue");
        this->alternate = true;
    }else{
        getDisplayString().setTagArg("i",1,"red");
        this->alternate = false;
    }
    ev << "Message name " << msg->getName() << " kind " << msg->getKind() << endl;
    delete msg;
    cMessage *m = new cMessage();
    m->setKind(12);
    m->setName("from PING");
    send(m,"out");
}
```

src/pong.cc

```
#include "pong.h"

Define_Module(Pong);
using namespace std;
void Pong::initialize()
{
     this->id = par("id").longValue();
     ev << "Init in PONG " << endl;
     getDisplayString().setTagArg("i",1,"yellow");
}

void Pong::handleMessage(cMessage *msg)
{
     if(this->alternate == false){
          getDisplayString().setTagArg("i",1,"purple");
          this->alternate = true;
     }else{
          getDisplayString().setTagArg("i",1,"yellow");
          this->alternate = false;
     }
     ev << "Message name " << msg->getName() << " kind " << msg->getKind() << endl;
     delete msg;
     cMessage *m = new cMessage();
     m->setKind(12);
     m->setName("from PONG");
     send(m,"out");
}
```

omnetpp.ini

```
[General]
network = pp2.simulations.Pp2
**.S1.id = 1
**.S2.id = 2
```

Circulation de jeton sur un anneau

Objectif

- Création de topologie depuis un wizard
- Création d'une topologie en anneau
  - taille fixe
- Attention aux définition de module et de simple node

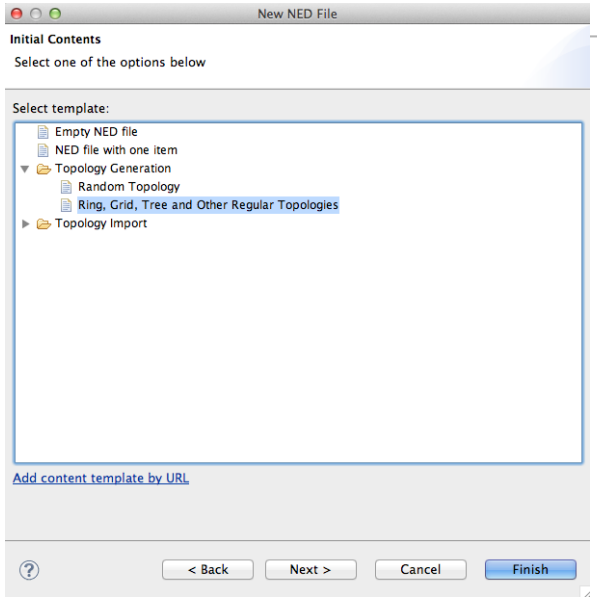
Création du projet

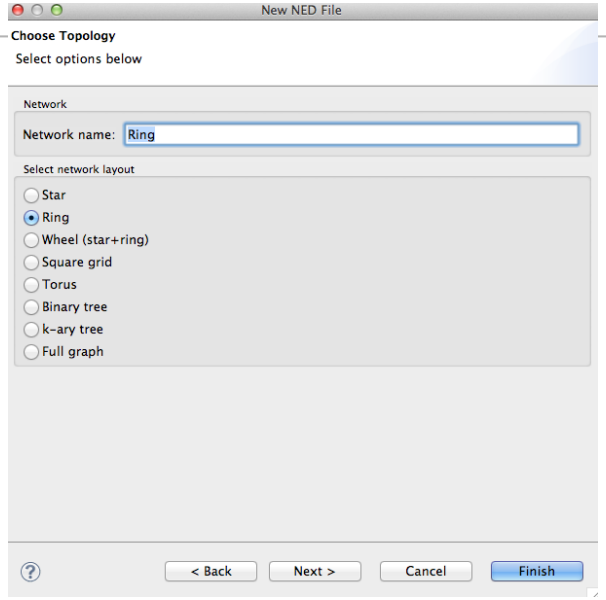
- Omnet++ Project
  - name : Ring
  - empty with src and simulations folder

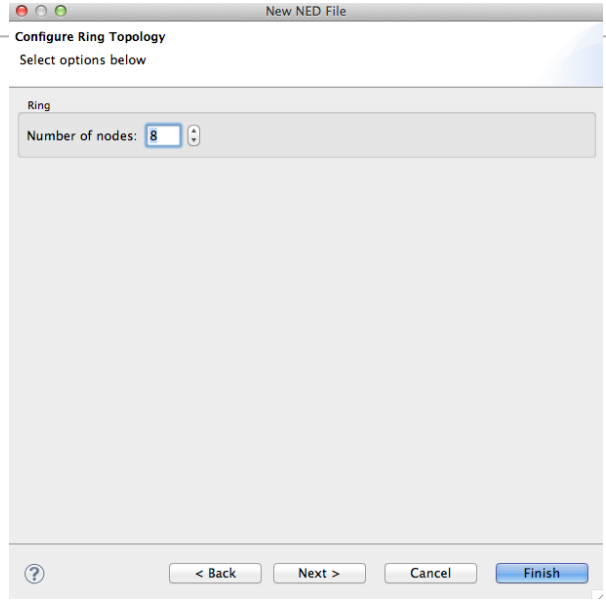
Création du réseau

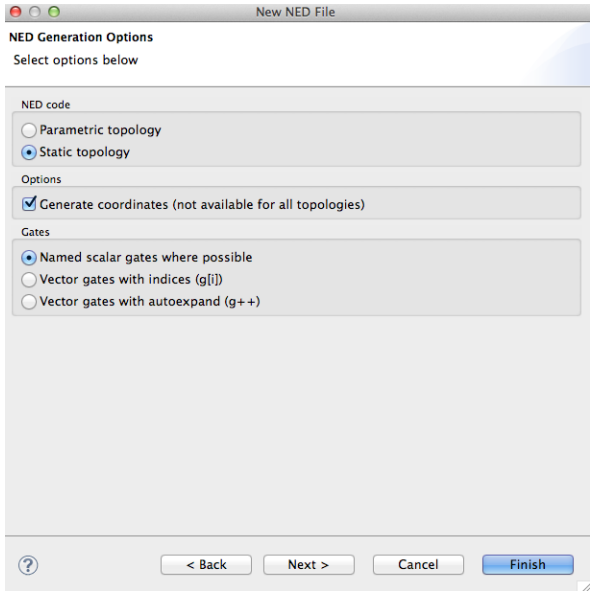
- Dans simulations création de Ring.ned
  - network description file
- Choix d'une topologie régulière

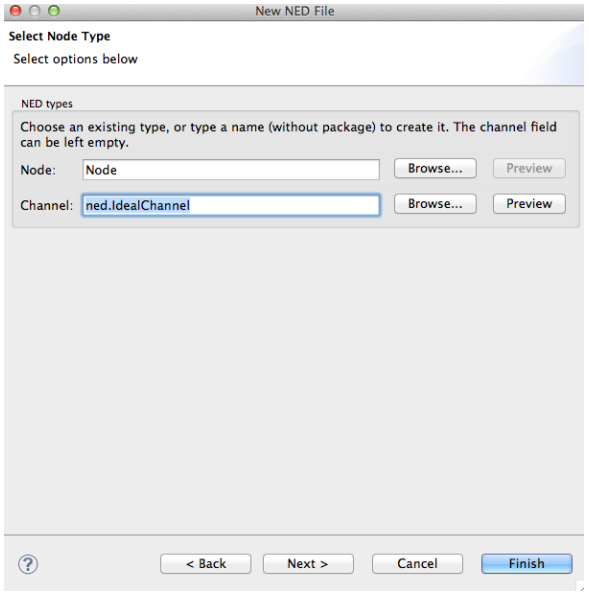
Wizard

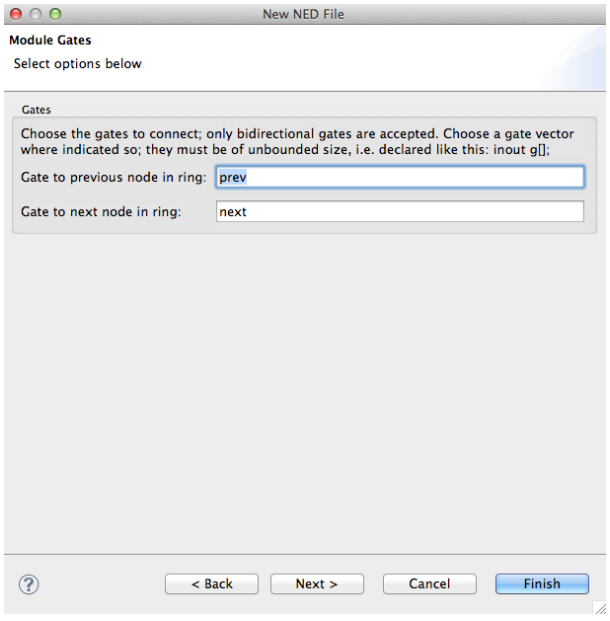

New NED File
Initial Contents
Select one of the options below
Select template:
Empty NED file
NED file with one item
Topology Generation
Random Topology
Ring, Grid, Tree and Other Regular Topologies
Topology Import
Add content template by URL
< Back
Next >
Cancel
Finish


New NED File
Choose Topology
Select options below
Network
Network name: Ring
Select network layout
Star
Ring
Wheel (star+ring)
Square grid
Torus
Binary tree
k-ary tree
Full graph
< Back
Next >
Cancel
Finish


New NED File
Configure Ring Topology
Select options below
Ring
Number of nodes: 8
< Back
Next >
Cancel
Finish


New NED File
NED Generation Options
Select options below
NED code
Parametric topology
Static topology
Options
Generate coordinates (not available for all topologies)
Gates
Named scalar gates where possible
Vector gates with indices (g[i])
Vector gates with autoexpand (g++)
< Back
Next >
Cancel
Finish


New NED File
Select Node Type
Select options below
NED types
Choose an existing type, or type a name (without package) to create it. The channel field can be left empty.
Node: Node
Browse...
Preview
Channel: ned.IdealChannel
Browse...
Preview
< Back
Next >
Cancel
Finish


New NED File
Module Gates
Select options below
Gates
Choose the gates to connect; only bidirectional gates are accepted. Choose a gate vector where indicated so; they must be of unbounded size, i.e. declared like this: inout g[];
Gate to previous node in ring: prev
Gate to next node in ring: next
< Back
Next >
Cancel
Finish

Ring.ned

```
package ring.simulations;
import ned.IdealChannel;
module Node {
    parameters:
        @display("i=misc/node_vs");
    gates:
        inout prev;
        inout next;
    connections allowunconnected:
}
network Ring
{
    submodules:
        node0: Node { @display("p=200,50");}
        node1: Node { @display("p=306,94");}
        node2: Node { @display("p=350,201");}
        node3: Node { @display(« p=306,307");}
        node4: Node { @display("p=199,350");}
        node5: Node { @display("p=93,307");}
        node6: Node { @display("p=50,200");}
        node7: Node { @display(« p=93,94");}
    connections:
        node0.next <--> IdealChannel <--> node1.prev;
        node1.next <--> IdealChannel <--> node2.prev;
        node2.next <--> IdealChannel <--> node3.prev;
        node3.next <--> IdealChannel <--> node4.prev;
        node4.next <--> IdealChannel <--> node5.prev;
        node5.next <--> IdealChannel <--> node6.prev;
        node6.next <--> IdealChannel <--> node7.prev;
        node7.next <--> IdealChannel <--> node0.prev;
}
```

Ring.ned modifié

```
package ring.simulations;
import ned.IdealChannel;
import ring.*;
network Ring
{
    submodules:
        node0: Node { @display("p=200,50");}
        node1: Node { @display(« p=306,94");}
        node2: Node { @display("p=350,201");}
        node3: Node { @display("p=306,307");}
        node4: Node { @display("p=199,350");}
        node5: Node { @display("p=93,307");}
        node6: Node { @display("p=50,200");}
        node7: Node { @display(« p=93,94 »);
    connections:
       node0.next <--> IdealChannel <--> node1.prev;
       node1.next <--> IdealChannel <--> node2.prev;
       node2.next <--> IdealChannel <--> node3.prev;
       node3.next <--> IdealChannel <--> node4.prev;
       node4.next <--> IdealChannel <--> node5.prev;
       node5.next <--> IdealChannel <--> node6.prev;
       node6.next <--> IdealChannel <--> node7.prev;
       node7.next <--> IdealChannel <--> node0.prev;
}
```

src/Node.ned

```
package ring;

//
// TODO auto-generated module
//
simple Node
{
    parameters:
        @display(« i=misc/node_vs");
         int id;
    gates:
         inout prev;
         inout next;
}
```

Fichier .ini

```
[General]
network = ring.simulations.Ring
cpu-time-limit = 300s
**.node0.id = 0
**.node1.id = 1
**.node2.id = 2
**.node3.id = 3
**.node4.id = 4
**.node5.id = 5
**.node6.id = 6
**.node7.id = 7
```

src/node.h

```
#ifndef __RING_NODE_H_
#define __RING_NODE_H_

#include <omnetpp.h>

/**
 * TODO - Generated class
 */
class Node : public cSimpleModule
{
private:
    int id;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

#endif
```

src/node.cc

```
#include "Node.h"

Define_Module(Node);
using namespace std;

void Node::initialize()
{
    this->id = par("id").longValue();
    ev << "Start in " << this->id << endl;
    if(this->id==0){
    ev << "Send first message" << endl;
    cMessage *m = new cMessage("Token");
    send(m,"next$o");
    }
}
void Node::handleMessage(cMessage *msg)
{
    ev << "Message received in " << this->id << endl;
    delete msg;
    ev << "Send message in " << this->id << endl;
    cMessage *m = new cMessage("Token");
    send(m,"next$o");
}
```

Circulation de jeton sur un anneau (2)

Objectif

- Création de topologie depuis un wizard
- Création d'une topologie en anneau
  - taille dynamique
  - définition d'un tableau de ports
- Attention aux définitions de module et de simple node

simulation/ring2.ned

```
package ring2.simulations;

import ned.IdealChannel;
import ring2.*;
//
// A generated network with ring topology.
//
network Ring2
{
    parameters:
        int n = default(8);
    submodules:
        node[n]: Node {
            gates: g[2];
        }
    connections:
        for i=0..n-1 {
            node[i].g[0] <--> IdealChannel <--> node[(i+1)%n].g[1];
        }
}
```

src/Node.ned

```
package ring2;

//
// TODO auto-generated module
//
simple Node
{
        parameters:
        @display("i=misc/node_vs");
      gates:
        inout g[];
}
```

omnetpp.ini

```
[General]
network = ring2.simulations.Ring2
cpu-time-limit = 300s
```

src/node.h

```
#ifndef __RING2_NODE_H_
#define __RING2_NODE_H_

#include <omnetpp.h>

/**
 * TODO - Generated class
 */
class Node : public cSimpleModule
{
  private:
   int id;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

#endif
```

src/node.cc

```
#include "Node.h"

Define_Module(Node);

void Node::initialize()
{
    this->id = this->getIndex();
    ev << "Start in " << this->id << endl;
    if(this->id ==0){
        ev << "send message in " << this->id << endl;
        cMessage *m = new cMessage("start");
        send(m,"g$o",0);
    }
}
void Node::handleMessage(cMessage *msg)
{
    ev << "Message in " << this->id << endl;
    ev << "Message : " << msg->getName() << endl;
    delete msg;
    ev << "send message in " << this->id << endl;
    cMessage *m = new cMessage("to next");
    send(m,"g$o",0);
}
```

Graphe aléatoire : première approche

Protocole

- Création d'un graphe aléatoire
  - attention au germe !
- Création d'un message spécifique
  - héritage de la classe cMessage
  - stockage dans le message des sites visités
  - utilisation d'un `vector`
- En chaque site, choix d'un voisin aléatoirement
  - mémorisation du site
  - émission du message
  - sauvegarde de l'id du site dans un fichier texte

simulations/Rnd.ned

```
package rnd.simulations;
import rnd.*;

//
// TODO auto-generated type
//
network Rnd
{
    parameters:
        int n = default(10);
        double densite=default(0.70);
    submodules:
        node[n]: Site {}

    connections:
        for i=0..n-1,for j=i+1..n-1, if(uniform(0,1) < densite) {
            node[i].neigh++ <--> node[j].neigh++;
        }
}
```

src/Site.ned

```
package rnd;

//
// TODO auto-generated module
//
simple Site
{
    parameters:
        @display("i=misc/node_vs");
        gates:
            inout neigh[];
}
```

src/myMessage.h

```
#ifndef MYMESSAGE_H_
#define MYMESSAGE_H_
#include <omnetpp.h>
#include <vector>
using namespace std;

class MyMessage: public :: cMessage {

private:
    vector<int> v;
public:
    MyMessage();
    virtual ~MyMessage();

    void addSite(int);
    vector<int>& getSites();

};

#endif /* MYMESSAGE_H_ */
```

src/myMessage.cc

```
#include "MyMessage.h"
#include <vector>
using namespace std;

MyMessage::MyMessage() {
   v = vector<int>();
}

MyMessage::~MyMessage() {
   // TODO Auto-generated destructor stub
}

void MyMessage::addSite(int i){
   v.push_back(i);
}

vector<int>& MyMessage::getSites(){
   return v;
}
```

src/Site.h

```
#ifndef __RND_SITE_H_
#define __RND_SITE_H_

#include <omnetpp.h>
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

/**
 * TODO - Generated class
 */
class Site : public cSimpleModule
{
private:
    int id;
    ofstream fileout;
    static const string filename;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);

  public:

    void saveData(string,int);
};
    const string Site::filename="./data.txt";
#endif
```

src/Site.cc

```
#include "Site.h"
#include "MyMessage.h"
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

Define_Module(Site);

void Site::initialize()
{
    this->id = this->getIndex();
     if(this->id == 0){
        this->saveData(Site::filename.c_str(),this->id);
        MyMessage *m = new MyMessage();
        m->addSite(this->id);
        int k = intuniform(0,gateSize("neigh")-1);
        send(m,"neigh$o",k);
     }
}
```

src/Site.cc

```
void Site::handleMessage(cMessage *msg)
{

   this->saveData(Site::filename.c_str(),this->id);
   ev << "Message in " << this->id << endl;
   ((MyMessage *)msg)->addSite(this->id);
   ev << "Visited nodes : ";
   vector<int> v = ((MyMessage *)msg)->getSites();
   for(unsigned int i=0; i< v.size() ; i++){
      ev << v[i] << ";" ;
   }
   ev << endl;
   int k = intuniform(0,gateSize("neigh")-1);
   send(msg,"neigh$o",k);
}

void Site::saveData(string file,int data){
   this->fileout.open(file.c_str(),ios::app);
   this->fileout << "in " << data << endl;
   this->fileout.flush();
   this->fileout.close();
}
```

omnetpp.ini

```
[General]
network = rnd.simulations.Rnd
cpu-time-limit = 300s
seed-0-mt=85685
```