

OMNeT++



Thi-Mai-Trang Nguyen

*Maître de conférence
Sorbonne Université - LIP6*

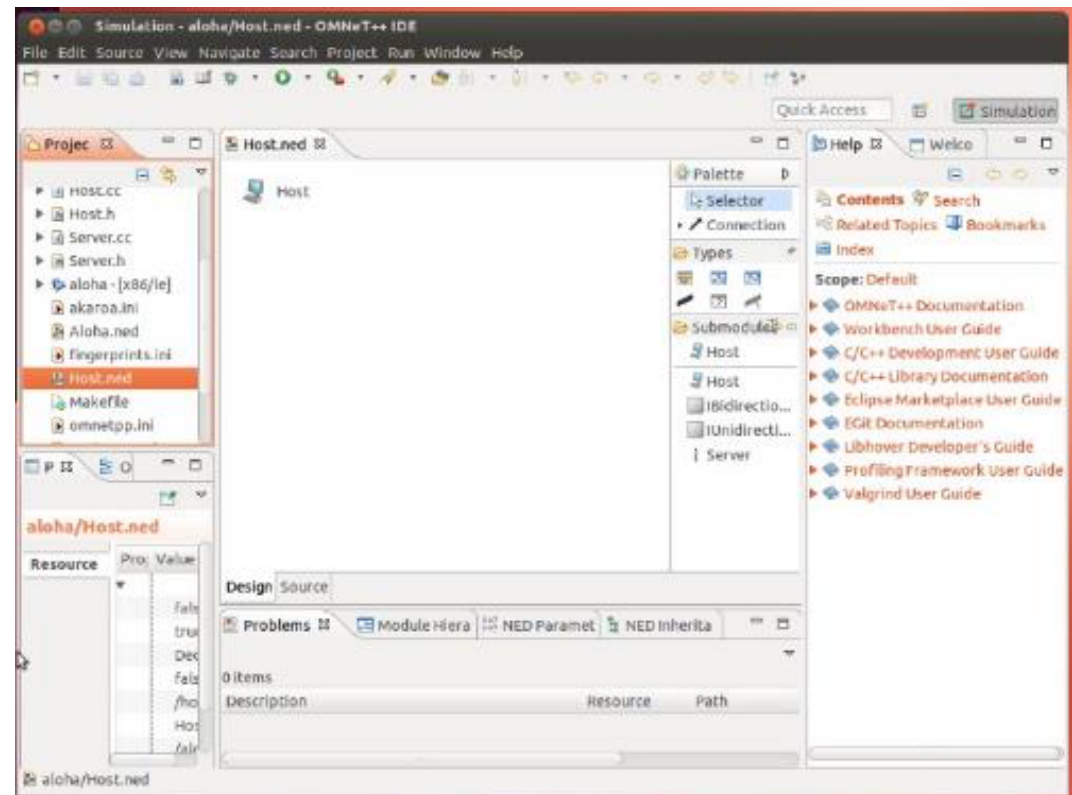
trnguyen@lip6.fr

Plan

- Ordonnancement d'événements et temporisation
 - Modélisation d'un délai de traitement dans un nœud
- Configuration d'un paramètre
- Génération de nombres aléatoires
- Observation d'une variable
- INET Framework et d'autres exemples de OMNeT++
- Importation d'un projet existant dans OMNeT++

OMNeT++ IDE

- Environnement avec l'interface graphique pour
 - Créer un projet
 - Créer un module simple
 - Créer un réseau
 - Compiler et lancer les simulations
 - Analyser les résultats

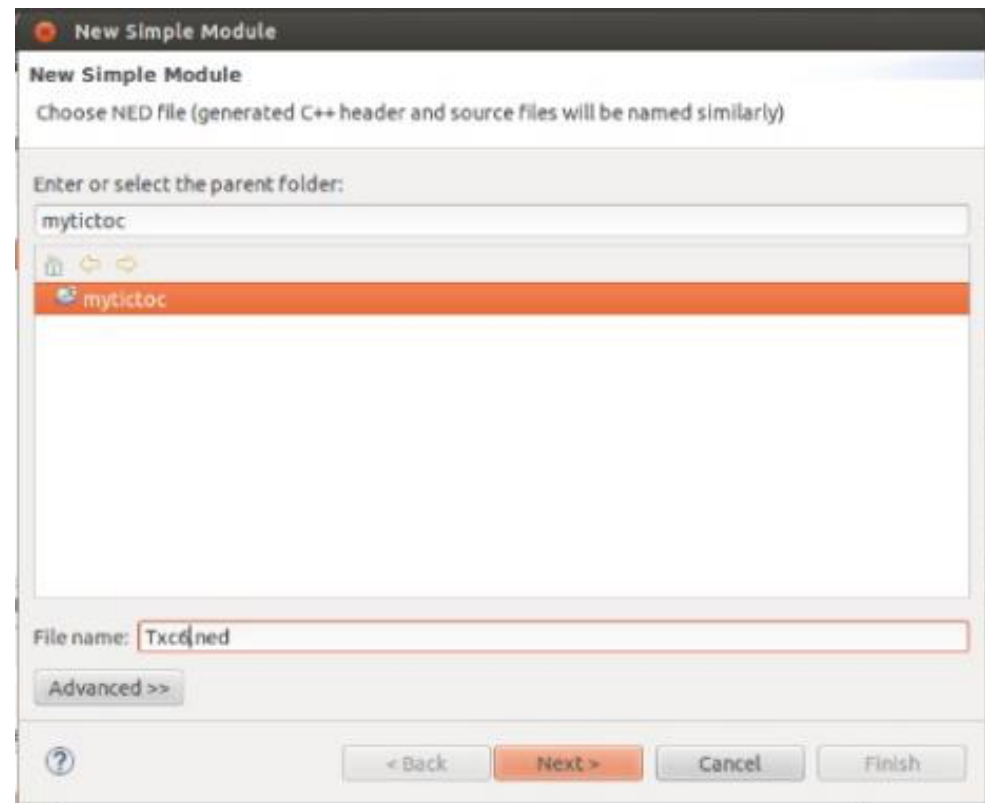


Modélisation d'un délai de traitement (1)

- Créer un réseau de deux nœuds : Tic et Toc
- Le canal entre deux nœuds a un délai de 100 ms
- Tic envoie un message à Toc qui le renvoie et ainsi de suite
- + Délai de traitement
 - A chaque réception du message, Tic et Toc gardent le message pendant 1 seconde avant de l'envoyer sur le canal

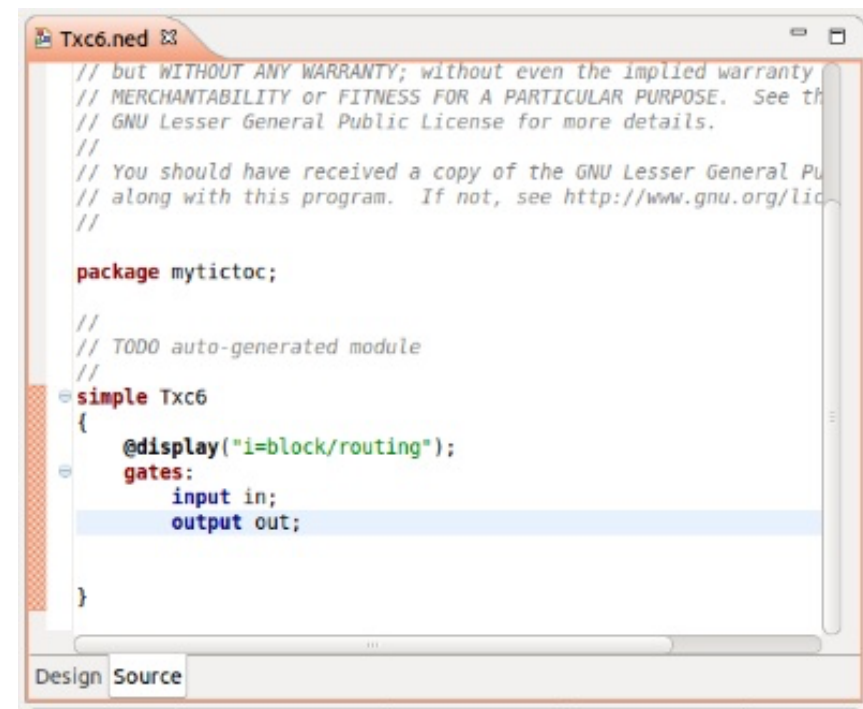
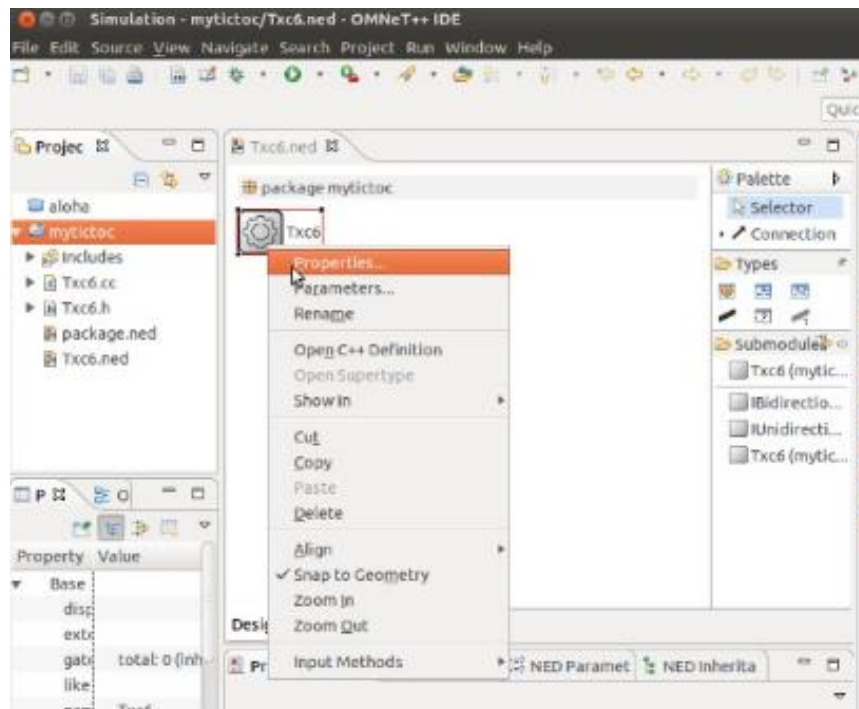
Modélisation d'un délai de traitement (2)

- Créer un projet mytictoc
 - *New → Project OMNeT++ → mytictoc → empty project*
- Créer un module simple appelé Txc6
 - *New → Simple Module → Txc6.ned → a simple module → Finish*
 - Trois fichiers `Txc6.ned`, `Txc6.cc` et `Txc6.h` sont créés automatiquement

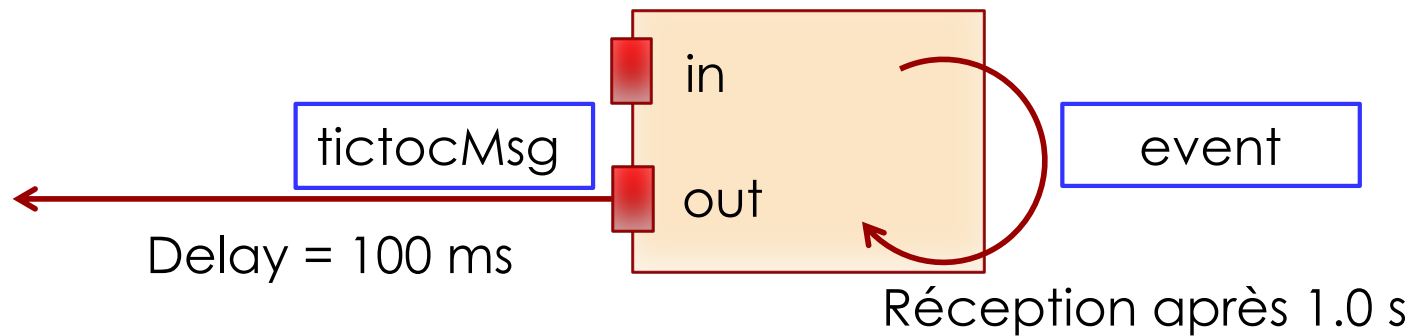


Modélisation d'un délai de traitement (3)

- Sélectionner le fichier `Txc6.ned`
- Dans la fenêtre `Design`, clique droit sur le module → choisir *Properties* → *Apparence* → *Image* → icône « bloc/routing »
- Dans le source code, définir deux ports : entrée et sortie



Modélisation d'un délai de traitement (5)

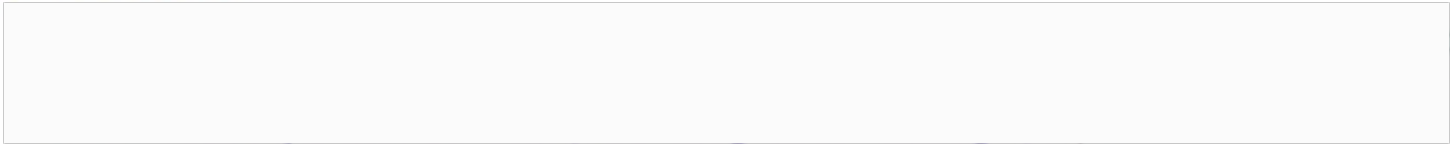


Modélisation d'un délai de traitement (6)

- Dans le fichier `Txc6.cc`, implémenter le délai de traitement en utilisant la fonction **`scheduleAt()`**
 - **`scheduleAt(simTime() + 1.0, event)`**
- Avec cette méthode, le module envoie le message `event` à lui-même au moment planifié (i.e. `simTime() + 1.0`)
- On dit que cette fonction envoie un « *self-message* » (c'est aussi la manière pour implémenter un temporisateur)
- Ensuite, dans la méthode `handleMessage()`, il faut différencier le traitement du message `event` (le *self-message*) et le traitement du message `tictocMsg` (venant du canal)

Modélisation d'un délai de traitement (7)

- Il est possible d'utiliser `if (msg->isSelfMessage())` à la place de `if (msg==event)`

```
void Txc6::handleMessage(cMessage *msg)
{
    if (msg==event)
    {
        
        send(tictocMsg, "out");
        tictocMsg = NULL;
    }
    else
    {
        
        tictocMsg = msg;
         scheduleAt(simTime()+1.0, event);
    }
}
```

Modélisation d'un délai de traitement (8)

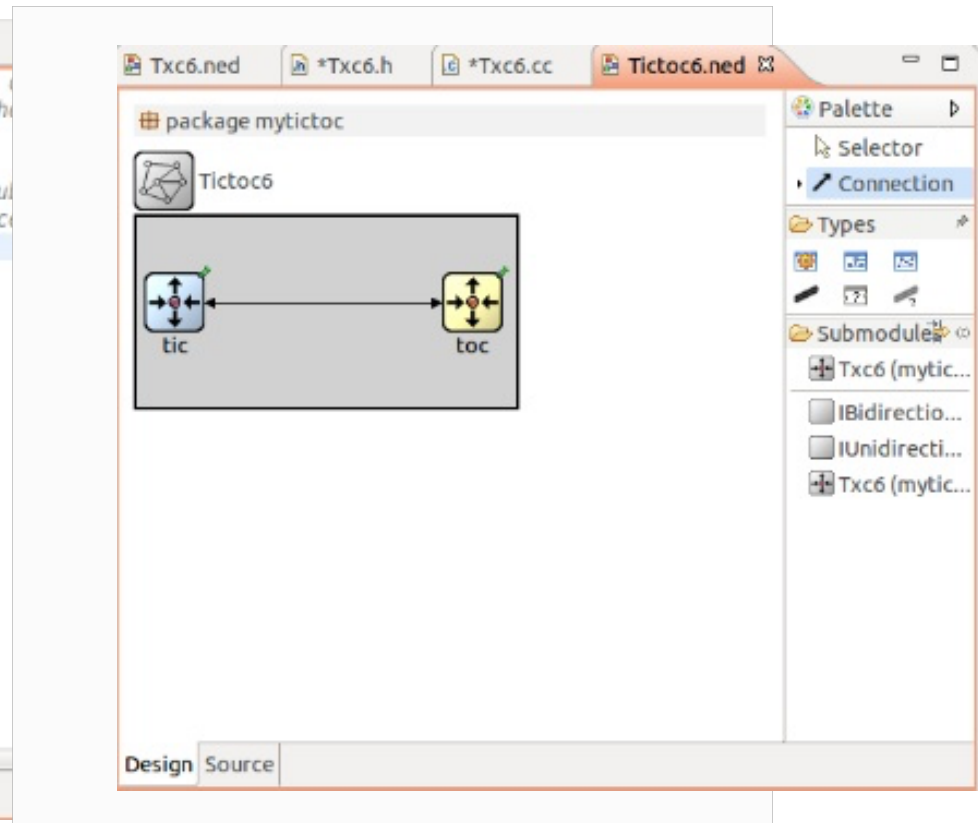
- Créer un réseau de deux nœuds Tic-Toc
 - *New → Network → Tictoc6.ned → an empty network → Finish*
 - Utiliser l'IDE pour personnaliser le réseau comme le suivant

```

// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public
// License along with this program. If not, see http://www.gnu.org/licenses/lgpl.html
//

package mytictoc;

network Tictoc6
{
    submodules:
        tic: Txc6 {
            @display("p=25,57;i=,#729FCF");
        }
        toc: Txc6 {
            @display("p=222,57;i=,#FCE94F");
        }
    connections:
        tic.out --> { delay = 100ms; } --> toc.in;
        toc.out --> { delay = 100ms; } --> tic.in;
}
  
```



Modélisation d'un délai de traitement (9)

- Tic commence à envoyer un message à 5 s
- Chaque nœud garde le message reçu pour 1 s avant de le renvoyer sur le canal



| Event# | Time | Src/Dest | Name | Info |
|--------|------|-------------|-----------|-------------|
| #1 | 5 | tic --> toc | tictocMsg | id=1 kind=0 |
| #3 | 6.1 | toc --> tic | tictocMsg | id=1 kind=0 |
| #5 | 7.2 | tic --> toc | tictocMsg | id=1 kind=0 |
| #7 | 8.3 | toc --> tic | tictocMsg | id=1 kind=0 |
| #9 | 9.4 | tic --> toc | tictocMsg | id=1 kind=0 |
| #11 | 10.5 | toc --> tic | tictocMsg | id=1 kind=0 |
| #13 | 11.6 | tic --> toc | tictocMsg | id=1 kind=0 |
| #15 | 12.7 | toc --> tic | tictocMsg | id=1 kind=0 |
| #17 | 13.8 | tic --> toc | tictocMsg | id=1 kind=0 |
| #19 | 14.9 | toc --> tic | tictocMsg | id=1 kind=0 |

Paramètres (1)

- Les paramètres qui peuvent être configurés dans le fichier `omnetpp.ini` doivent être déclarés dans la partie **parameters** du fichier **.ned**
- Exemple :
 - Dans le fichier `nœud.ned`


```
simple Noeud
{
    parameters:
    → int limit = default(2);

    @display("i=abstract/router");
    gates:
        input in;
        output out;
}
```


Paramètres (2)

- Dans le fichier `omnetpp.ini`

```
[General]
network = Reseau
*.*.limit = 5
```



- Dans le fichier `noeud.cc`
 - Utiliser la fonction **par()** pour obtenir la valeur configurée dans le fichier `.ini`



```
#include "noeud.h"

Define_Module(Noeud);

void Noeud::initialize()
{
    // TODO - Generated method body
    cMessage *msg = new cMessage("data");
    send(msg, "out");
    num_pkt = par("limit");
}

void Noeud::handleMessage(cMessage *msg)
{
    // TODO - Generated method body
    num_pkt--;
    if (num_pkt == 0){
        delete msg;
    } else{
        send(msg, "out");
    }
}
```

Génération de nombres aléatoires (1)

- Réseau Tictoc7
 - On souhaite que le délai de traitement du message de `Tic` et de `Toc` soit une variable aléatoire suivant la loi exponentielle
- Dans le fichier `Txc7.ned`, créer un paramètre `delayTime`

```
//  
simple Txc7  
{  
    parameters:  
        → volatile double delayTime @unit(s);    // delay before sending back message  
           @display("i=block/routing");  
    gates:  
        input in;  
        output out;  
}
```

Génération de nombres aléatoires (2)

- Dans le fichier de configuration `omnetpp.ini`
 - Générer les variables aléatoires pour Tic et Toc

```
[Config Tictoc7]  
network = Tictoc7  
Tictoc7.tic.delayTime = exponential(3s)  
Tictoc7.toc.delayTime = exponential(5s)
```

- Donner une valeur de la graine

```
[General]  
seed-0-mt=532569
```

Numéro du
générateur

Type (algorithme)
du générateur

Génération de nombres aléatoires (3)

- Dans le fichier d'implémentation `Txc7.cc`, lire le résultat du paramètre `delayTime` dans la méthode `handleMessage()`

```
// we'll get a different delay every time.
```

```
simtime_t delay = par("delayTime");
```

```
EV << "Message arrived, starting to wait " << delay << " secs...\n";
```

```
tictocMsg = msg;
```

```
scheduleAt(simTime()+delay, event);
```

Observation d'une variable (1)

- Pour observer les valeurs d'une variable durant la simulation, utiliser la fonction `WATCH()`
- Exemple
 - Dans le fichier `noeud.h`

```
#include <omnetpp.h>

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



Observation d'une variable (2)




- Dans le fichier `noeud.cc`

- Durant la simulation

(Noeud) Reseau.noeud1

Fields

Contents (3)

| Class | Name | Info |
|---|---------|----------------|
|  cGate | in | <-- noeud2.out |
|  cGate | out | --> noeud2.in |
|  int | counter | 15557 |

```
#include "noeud.h"

Define_Module(Noeud);

void Noeud::initialize()
{
    // TODO - Generated method body
    WATCH(counter);
    cMessage *msg = new cMessage("data");
    send(msg, "out");
}

void Noeud::handleMessage(cMessage *msg)
{
    // TODO - Generated method body
    send(msg, "out");
    counter++;
}
```

Observation d'une variable (3)

- Dans le fichier noeud.cc

```
if (ev.isGUI())  
    updateDisplay();
```

```
void Noeud::updateDisplay()  
{  
    char buf[40];  
    → sprintf(buf, "counter: %ld", counter);  
    getDisplayString().setTagArg("t",0,buf);  
}
```

- Durant la simulation

→ counter: 4521



Observation des événements (1)

- Une fois le code de la simulation est terminé

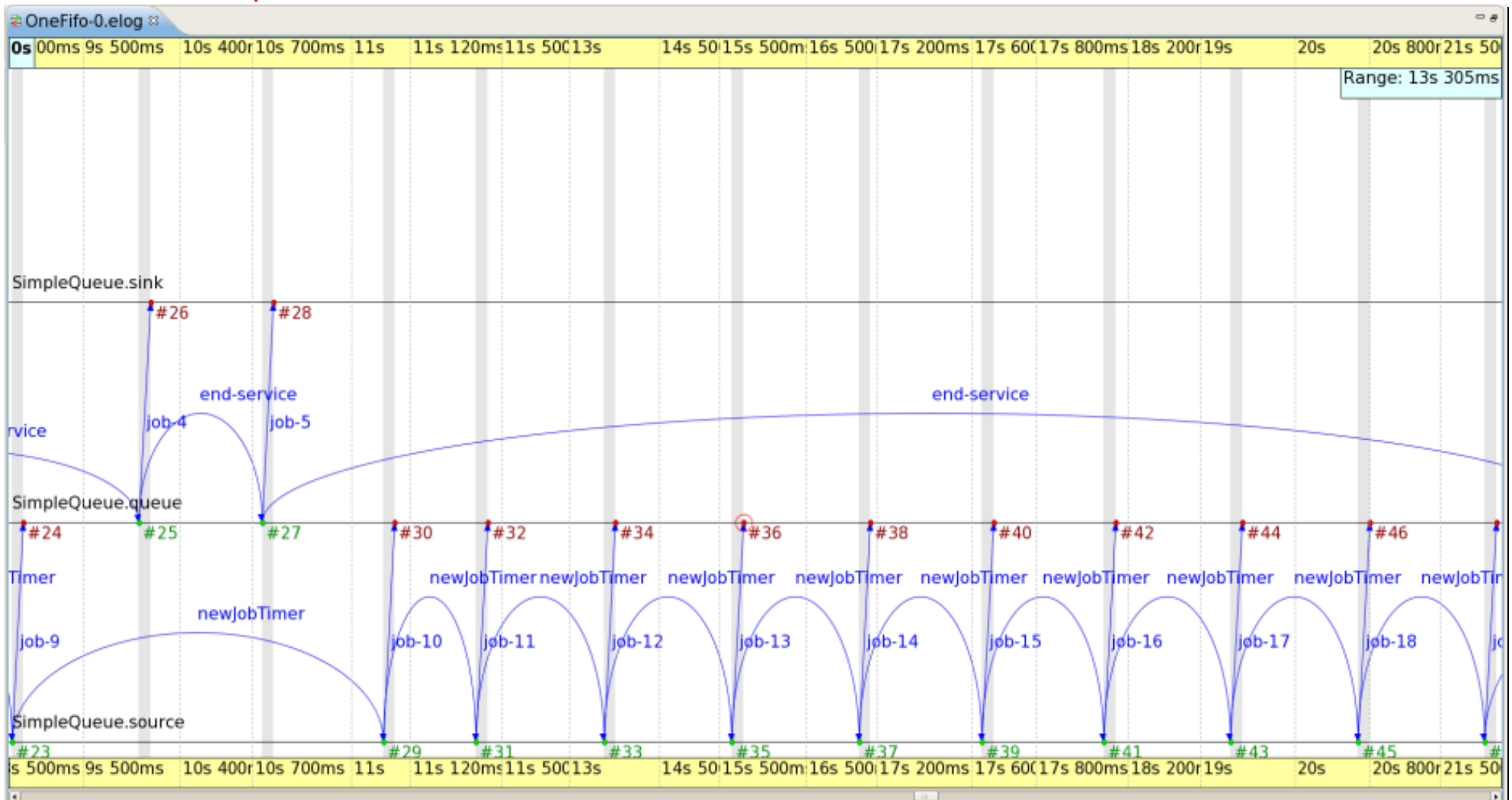
On peut observer la simulation :

- Animation graphique montrant l'évolution dans le temps de certains objets de la simulation
 - Vérification visuelle que le code implante a priori le comportement souhaité du système
 - Compréhension et observation du système
 - Certains événements peuvent être mis en avant
- Le déroulement des événements et leur instant d'occurrence
 - Permet de déboguer les événements (plus précis, automatisation possible)

Observation des événements (2)

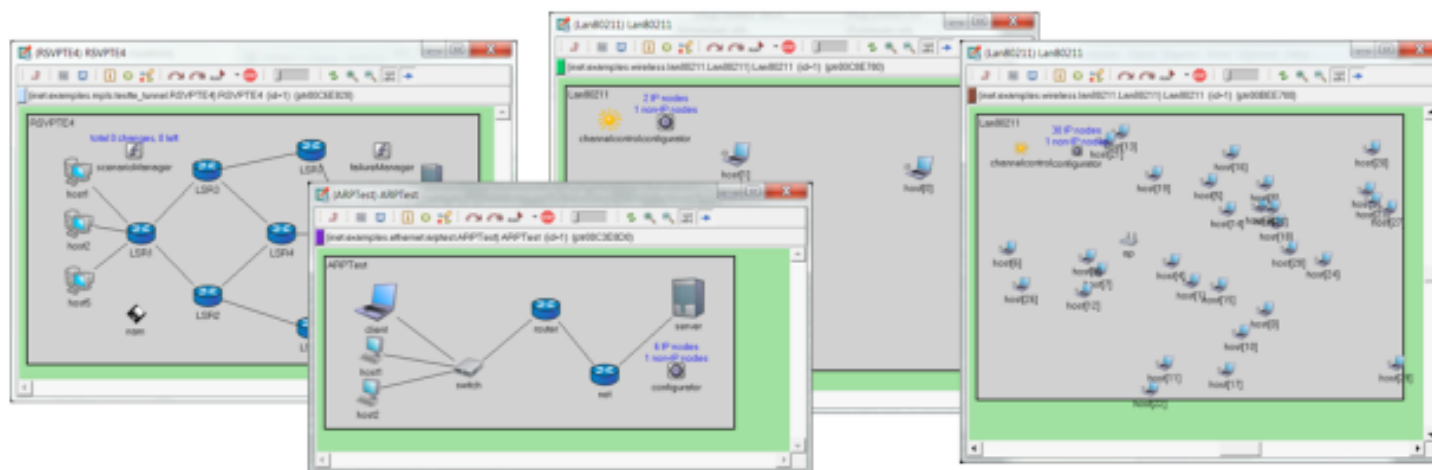
22

- Le déroulement des événements et leur instant d'occurrence
- Exemple :



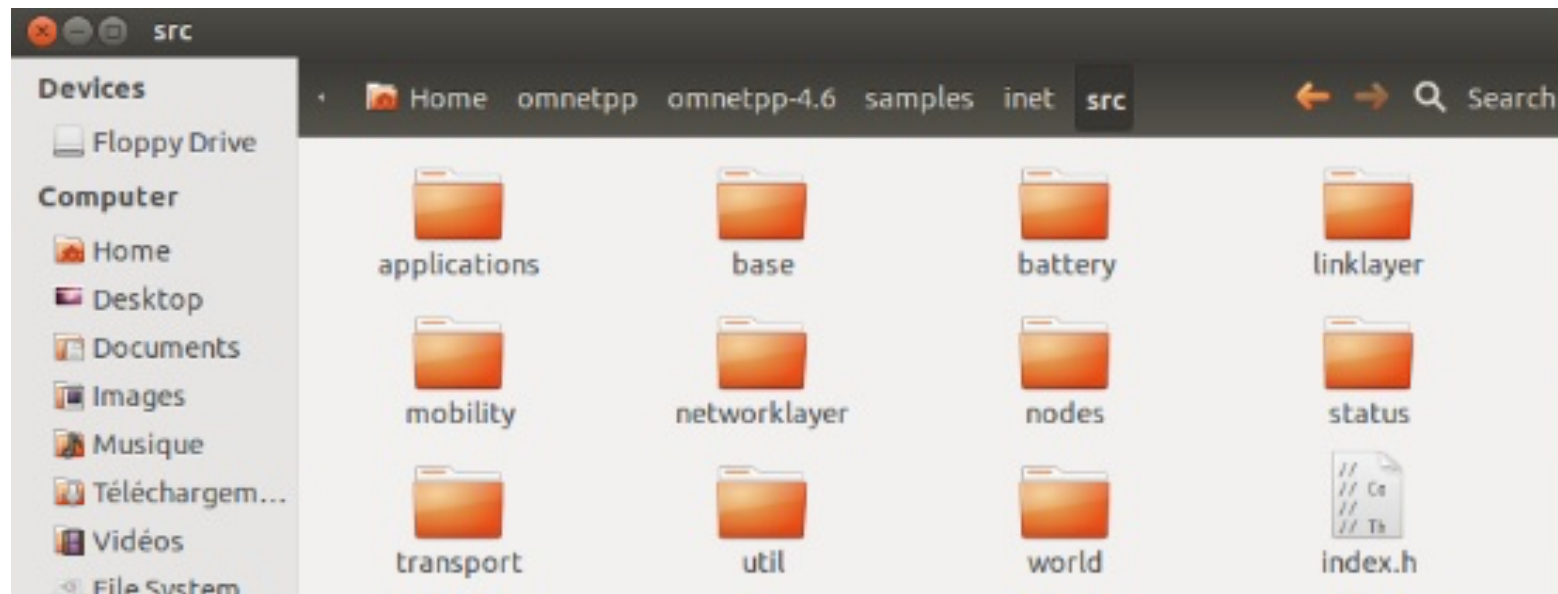
INET Framework

- Modèles de simulation pour l'Internet
 - UDP, TCP, SCTP
 - IPv4, IPv6
 - Ethernet, PPP, 802.11
 - MPLS, OSPF, MANET
 -
- <http://inet.omnetpp.org/>



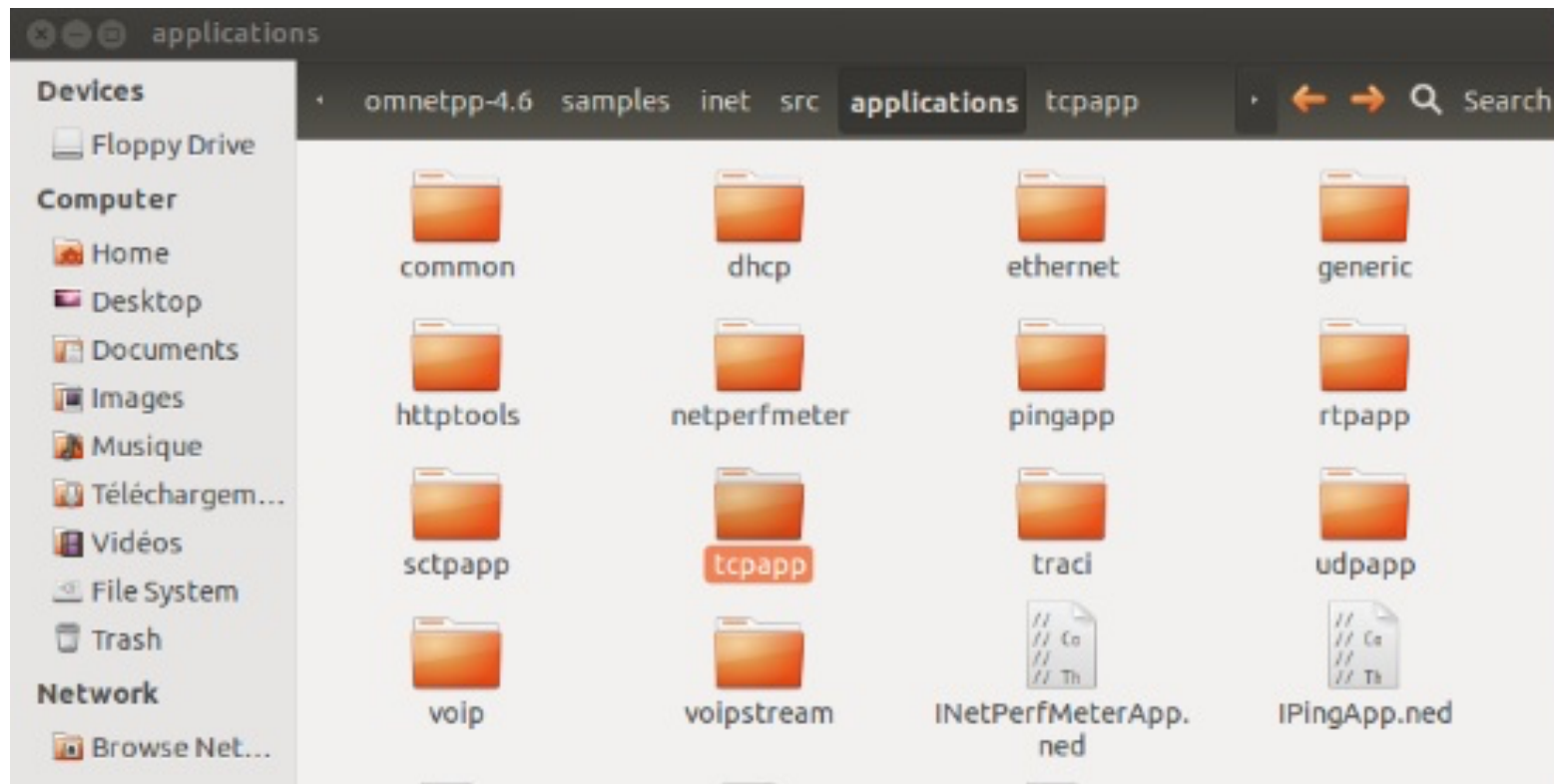
Modules dans INET Framework

- Les modules sont organisés sous la forme des packages hiérarchiques correspondant à l'arbre des dossiers (comme les packages dans Java)
- Les packages INET sont organisés suivant les niveaux du modèle OSI



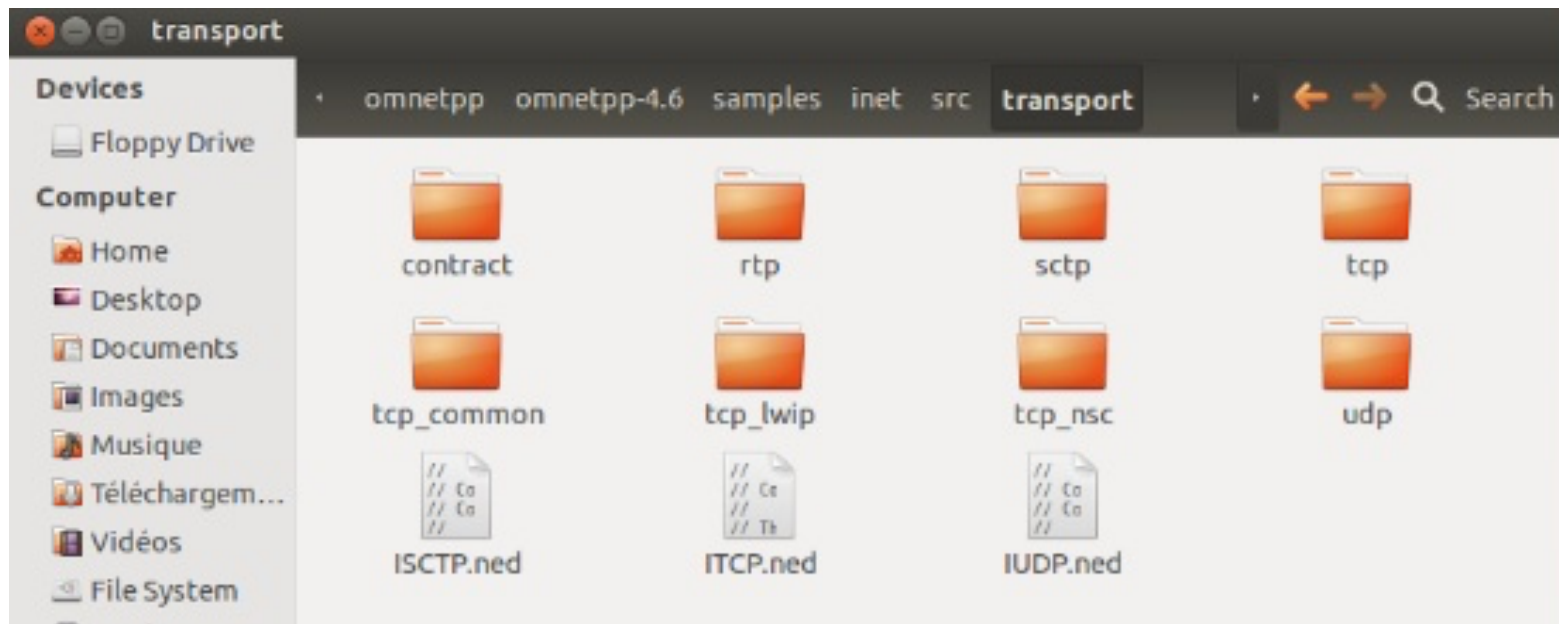
Package inet.applications

- Niveau application



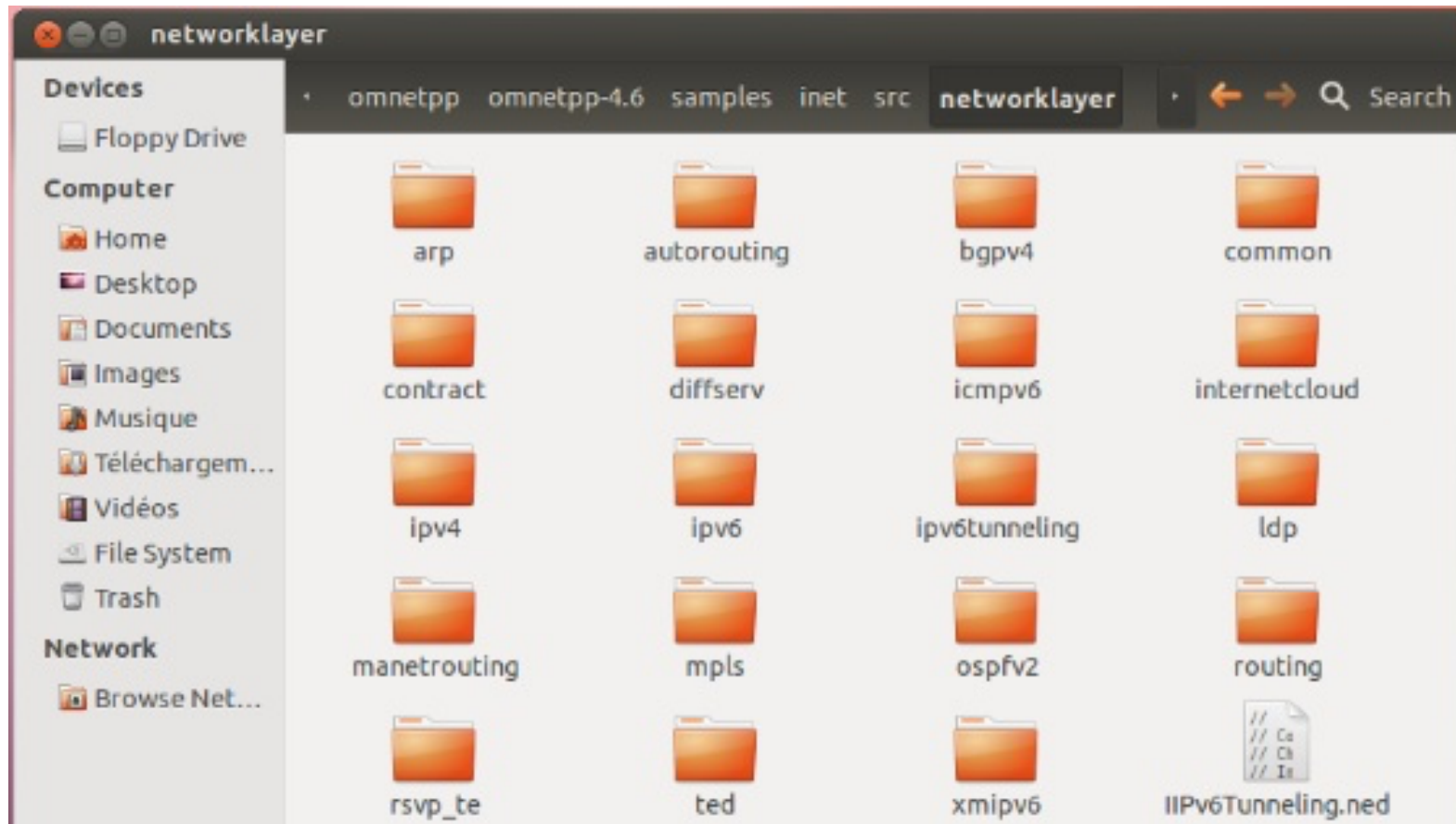
Package inet.transport

- Niveau transport (TCP, UDP, SCTP)



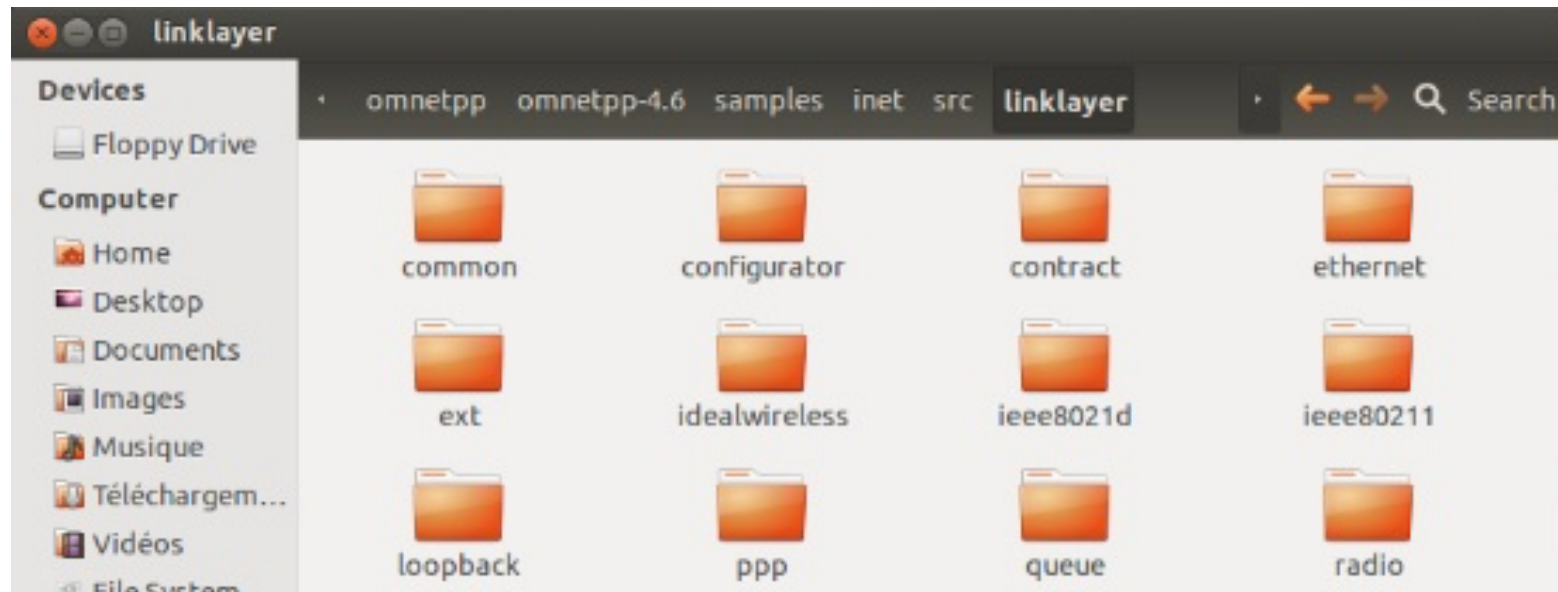
Package inet.networklayer

- Niveau réseau (IPv4, IPv6, protocoles de routage)



Package inet.linklayer

- Niveau liaison (Ethernet, 802.11)

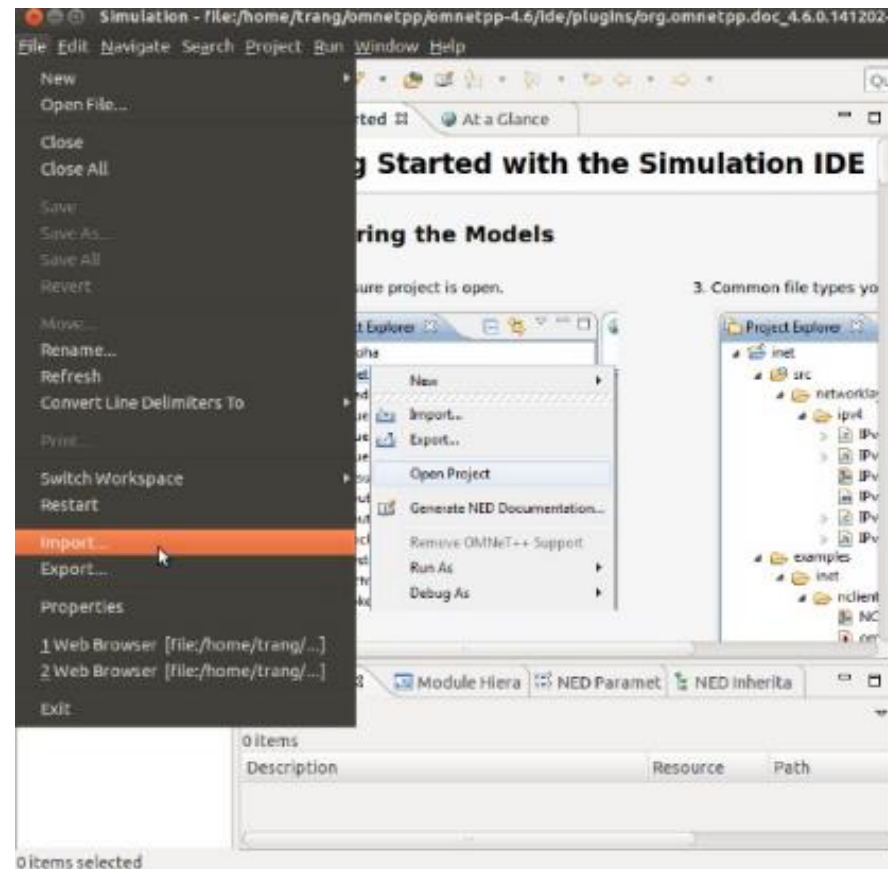


Importer un projet existant (1)

- Pour examiner en détail un projet existant et éventuellement le modifier, il est préférable de travailler seulement sur une copie de celui-ci
 - En cas d'erreurs durant les modifications, le projet existant reste intact
- Pour importer un projet existant
 - Créer un espace de travail qui n'est pas le même répertoire du projet existant
 - Exemple: `~/omnetpp/exercices`
 - Aller dans le nouvel espace de travail
 - File → Switch Workspace → `~/omnetpp/exercices`

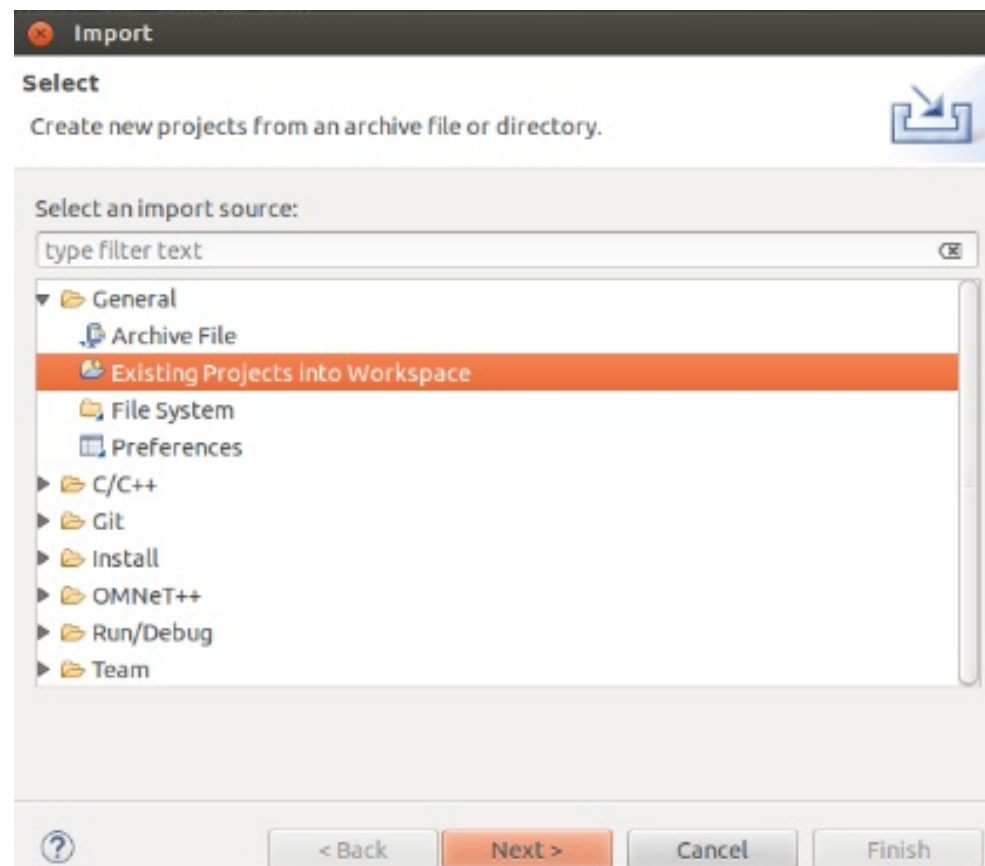
Importer un projet existant (2)

- Sélectionner File → Import ...



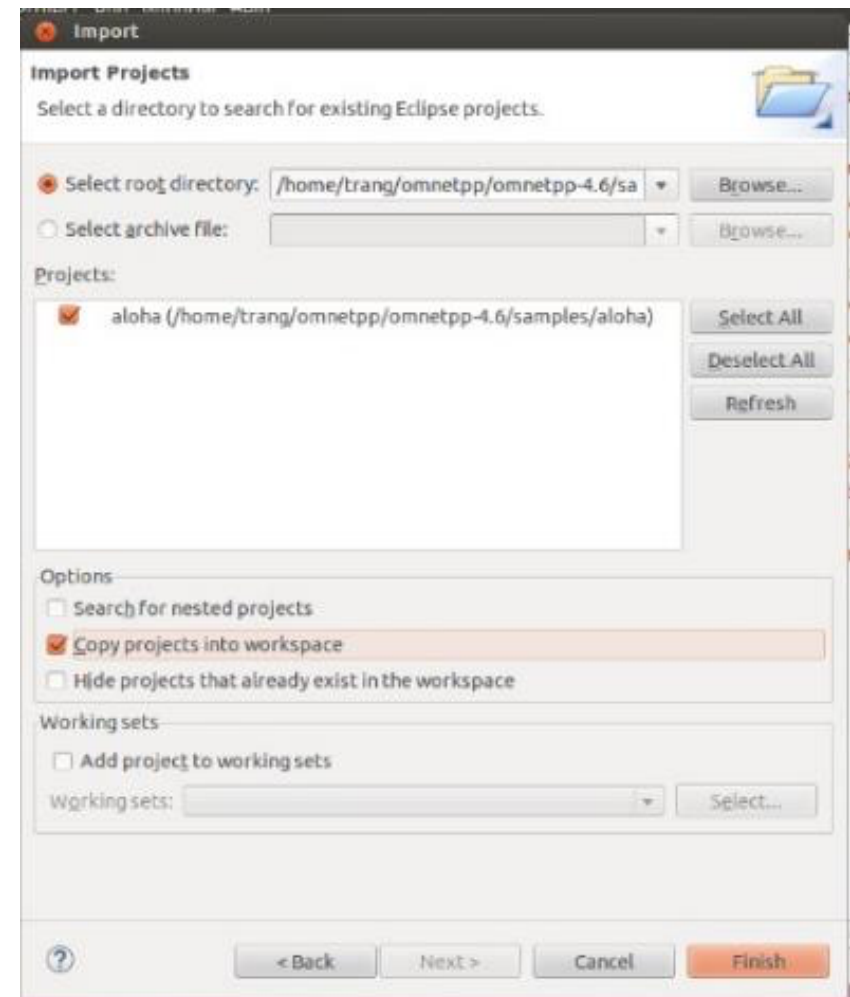
Importer un projet existant (3)

- Choisir « Importer des projets existants dans l'espace de travail »



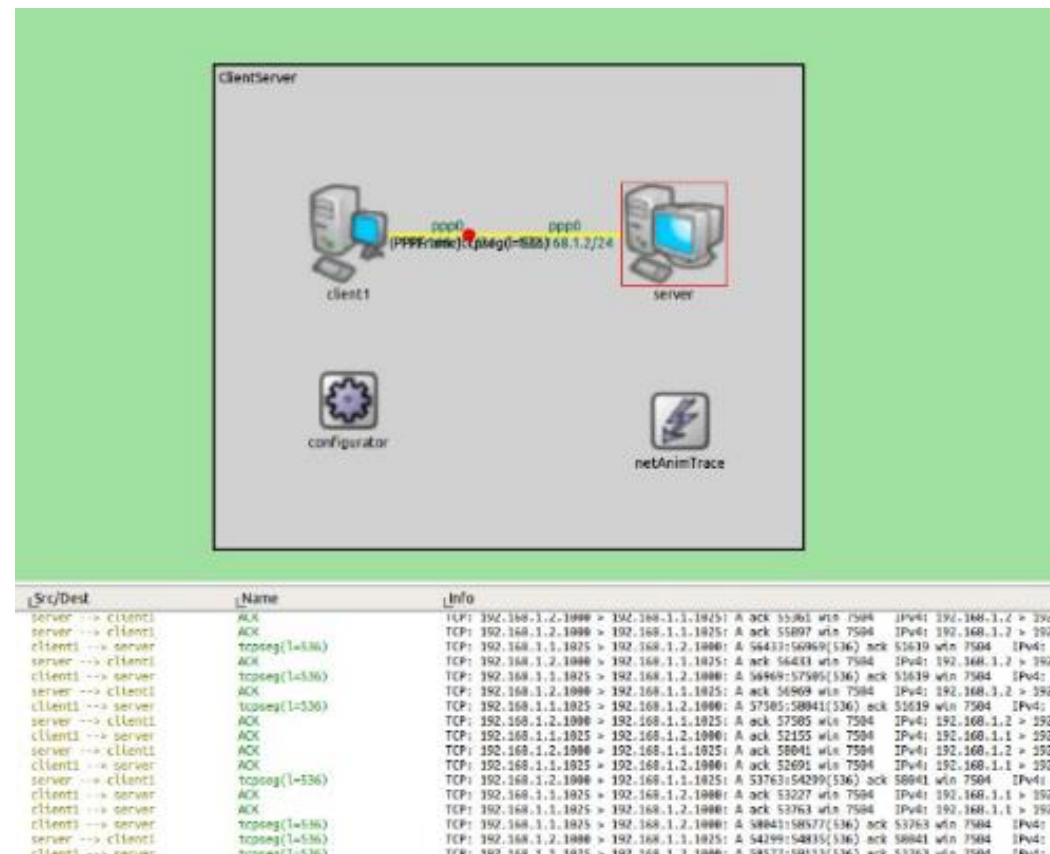
Importer un projet existant (4)

- Choisir le répertoire source qui contient le projet existant
 - Ex: `~/omnetpp/omnetpp-4.6/samples/aloha`
- Sélectionner l'option « *Copy projects into workspace* »
 - Un nouveau répertoire `~/omnetpp/exercices/aloha` sera créé
 - Tous les fichiers dans `~/omnetpp/omnetpp-4.6/samples/aloha` seront copiés dans `~/omnetpp/exercices/aloha`



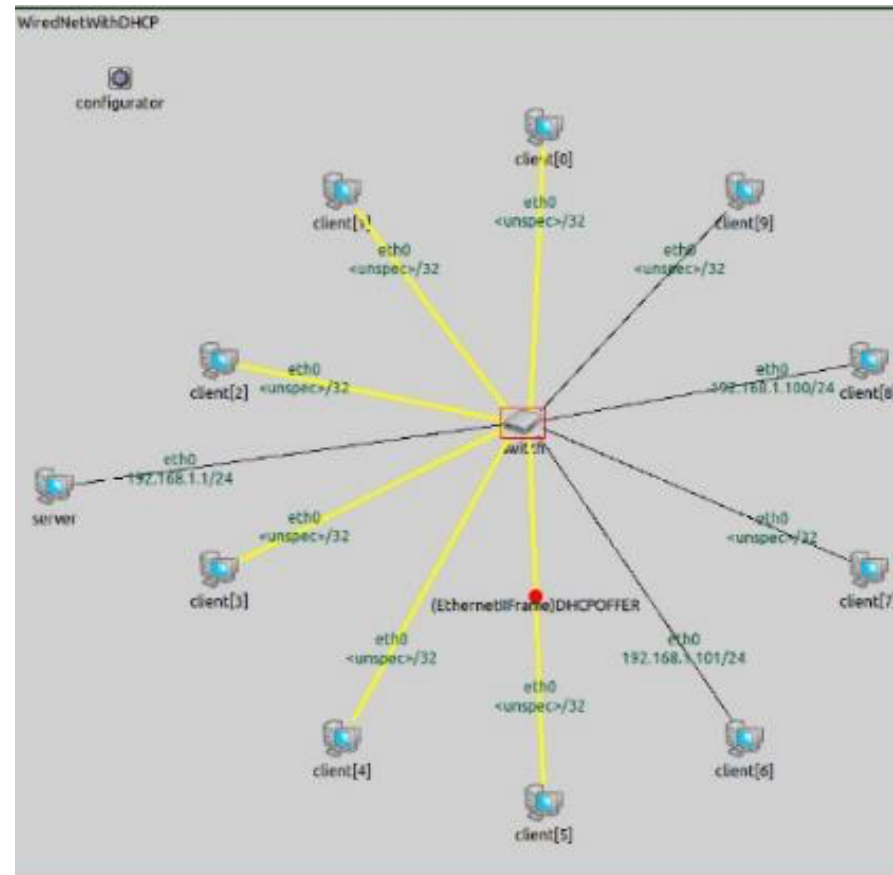
Transmission Control Protocol (TCP)

- ...\samples\inet\examples\inet\tcpclientserver



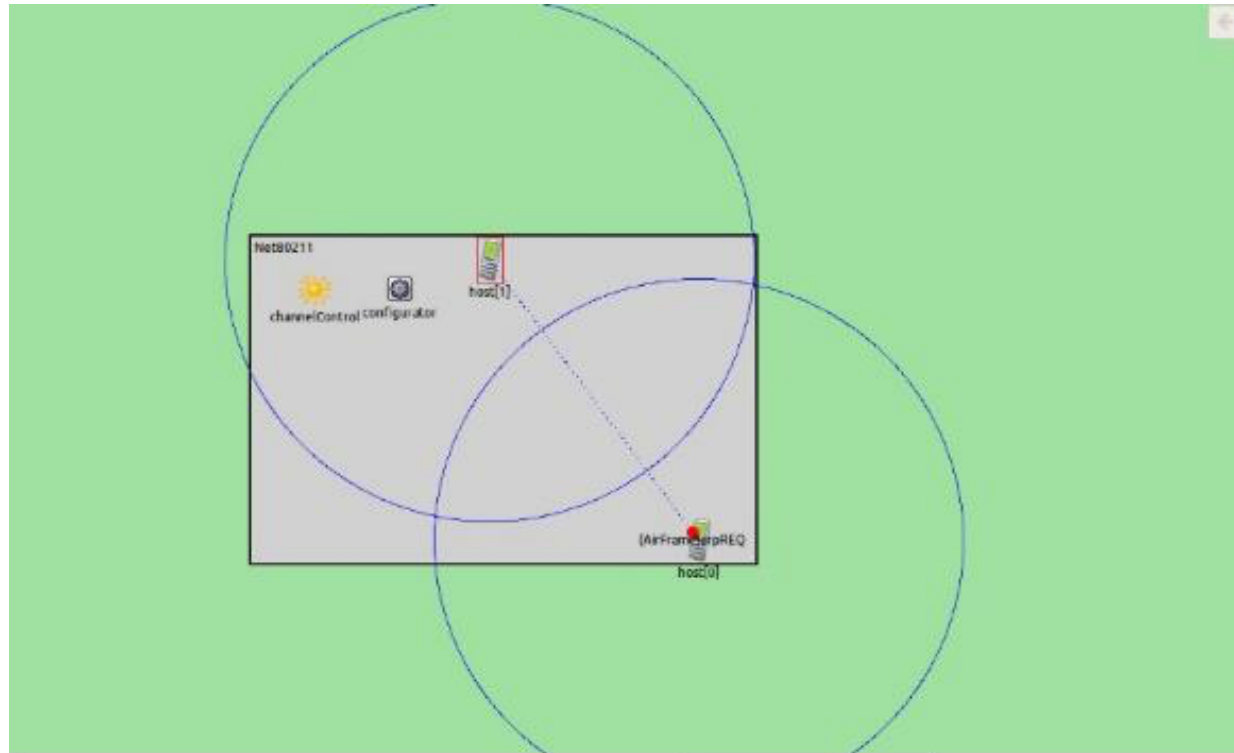
Dynamic Host Configuration Protocol (DHCP)

- ...\samples\inet\examples\dhcp



802.11 en mode ad-hoc

- ...\samples\inet\examples\adhoc



Méthode d'accès aléatoire Aloha

- ...\samples\aloha

