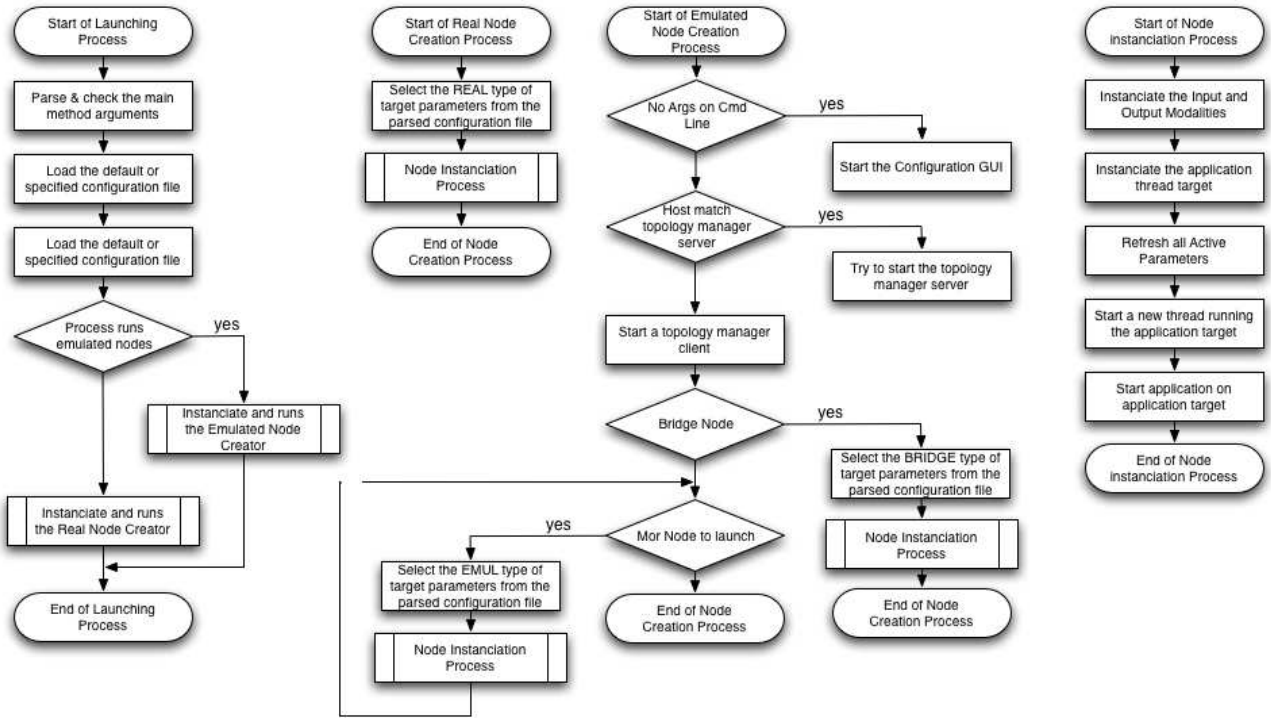
Software Architecture

Introduction

Refer to document *Overview of the Freemote Project* for an overview of the whole project and terminology.

Launching phase



Messages

The Real nodes are based on the 802.15.4 radio standard. This standard define many configurations for the messages. Refer to the norm[1] for details. We choosed to use the configuration used on the IEEE 802.15.4 compliant Berkeley Motes such as the MICAz or TelosB.

TinyOS uses the following configuration

1 802.15.4 IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. 2003.

TinyOS IEEE 802.15.4 DATA Messages Configuration	
field	***meaning***

fcfhi (Frame Control High Byte)	`Frame Type = 001 // DATA Frame`
	`Security Enable = 0 // Disable Security`
	`Frame Pending = 0 // No Frame is Pending`
	`Ack Request = [0|1] // Depends on the needs`
	`Intra PAN = 0 // Enable intra PAN communication`
	`Reserved = 0`
	
	`VALUE = 0x01 (ACK NOT REQUESTED) | 0x21 (ACK REQUESTED)`

fcflo (Frame Control Low Byte)	`Reserved = 00`
	`Dest Addr Mode = 01 // 16 BITS DEST ADDRESS & DEST PAN`
	`Reserved = 00`
	`Src Addr Mode = 00 // No SRC ADDRESS & NO SRC PAN`
	
	`VALUE = 0x08`

Tableau 1: TinyOS IEEE 802.15.4 DATA Messages Configuration.

Moreveover, TinyOS use a broadcast 0xFFFF PAN for the destination address. This

A quoi sert le PANID -> isoler differents reseaux sur le meme frequence -> utilise le broadcast et prennent un byte du payload 802.15.4 pour implementer cette feature (Group)

What about the AM type field

GRAPHIQUE FAUX, LE CRC N EST PAS ENVOYE NI POUR TELOSB NI POUR MICAZ

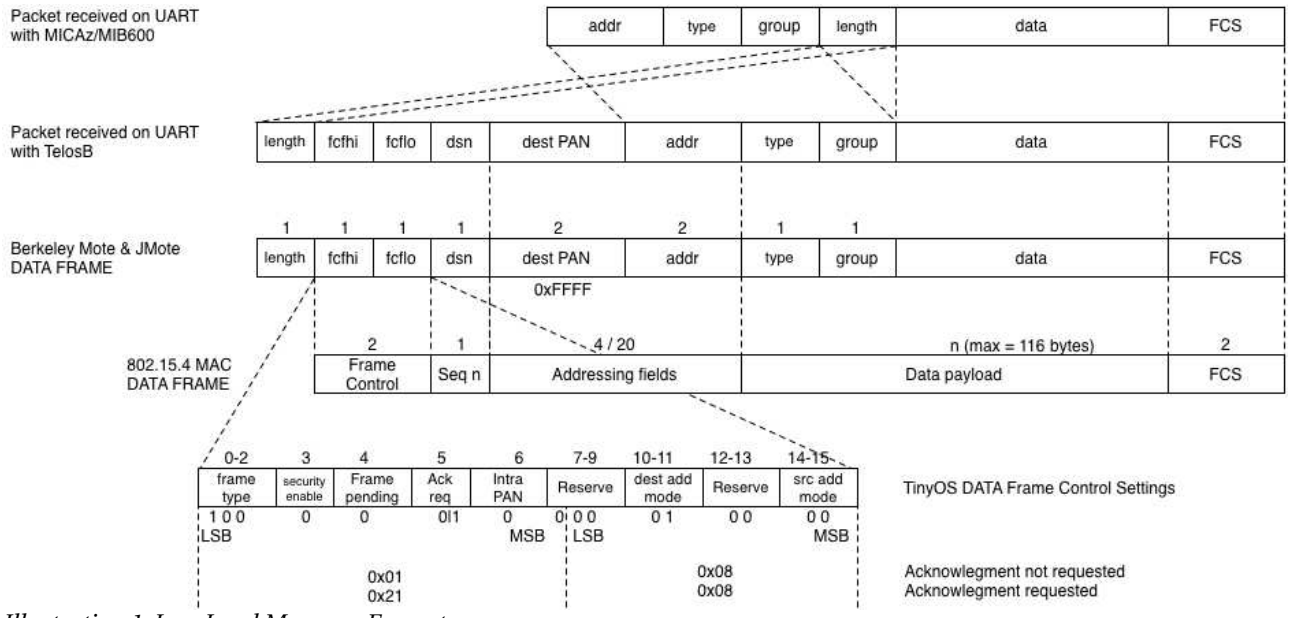


Illustration 1: Low Level Messages Format

Le champs length de la trame 802.15.4 definit la taille totale de la trame (HEADER + PAYLOAD + FCS) sans le byte de longueur lui meme. La taille maximale pouvant etre envoyee sur le chip CC2420 est 127. Avec les reglages definits de HEADER par TinyOS la longueur maximale de charge utile =

HEADER = Length + fcfhi + fcflo + dns + dest PAN + addr = 1 + 1 + 1 + 1 + 2 + 2 = 8 bytes

TOS_IN_PAYLOAD = type + group = 1 + 1 = 2bytes

FCS = 2 bytes

127 – (HEADER + TOS_IN_PAYLOAD + FCS – 1) = 127 – (8 + 2 + 2 – 1) = 116bytes

Par defaut, la longueur maximale des paquets de TinyOS sont limites a environs 36 bytes, il faudra modifier la variable TOSH_DATA_LENGTH dans les makefiles pour transporter plus d'information.

1.1 Passage pour le pont.

Le noeud pont est constitue d'une partie reelle et d'une partie emulee. La partie reelle est constituee d'un mote (MICAz + MIB600 ou TelosB + Virtual Com port on USB) et communiquent avec le programme Java grace a une API fournie par TinyOS. La communication se fait sur un port serie soit au dessus d'un Socket (MIB600) soit au dessus grace à un port com virtuel (USB).

Cependant les informations pouvant etre recues par ces deux configurations sont differentes et sont representees sur la figure ????. Avec le TelosB, les informations disponibles à la reception d'un message sont exactement les informations qui auraient été disponibles sur le mote ce qui permet une transparence complete. Avec le MICAz cependant, Les 6 premiers bytes de la tramme ne sont pas transmis.

La longueur peut etre aisément calculée, Le dest pan peut etre fixe a sa valeur par defaut (0xFFFF), le champs FCFLO peut lui aussi etre fixe a sa valeur par defaut (0x08). Cependant les champs DSN et FCFHI contiennent de l'information qui ne peut pas etre reconstruite tel que la demande d'acquittement ou non.

C'est pourquoi les accquittements se feront pour le pont au niveau du mote (si besoin le mote envoye un ACK).

Pour le TelosB les valeurs sont directement issues de la trame. Pour le MICAz le DSN est fixe à 0x00 et le FCFHI est fixé à 0x01 (pas d'acquittement).

Format reçu

ADDRESS:2,TYPE:1,GROUP:1,DATALENGTH(Correspond au length de 802.15.4):1,DATA:n,CRC:2

Cependant la classe implementatant la librairie Java TinyOS a ete modifiee comme suit

Retourne des messages du type net.tinyos.message.micaz.TOSMsg quel que soit le noeud reel

Plus de notion de listener associés a un template de message -> retourne toujours un message brut. -> s'enregistrer qu'une seule fois au listener

Ces modifications sont ajoutees dans la classe net.tinyos.message.FreemoteReceiver.

Quelques précisions sur ces messages

La methode get_data() sur un Message retourne un tybleau de byte contenant uniquement les données du message

CàD: Les données contenues a partir du champs goup (non compris) et jusqu'au CRC (non compris)

La methode dataGet permet de retourner tout le buffer (Header, etc.)

LE CHAMPS LENGTH DE LA TRAME 802.15.4 CORRESPOND A LA TAILLE DU PAYLOAD 802.15.4 SANS LE CRC

CàDire: get_data().length + 2 (group et type)

Limitations des Berkeley Motes par rapport au JMotes et noeuds emules, le promiscuous mode n'est pas possible. L'acquittement automatique est activé sur les Berkeley Motes et ceuzx cis ne recoivent que les messages de Bcast ou qu'ils leur sont destinées.

1.2 Passage des messages entre les noeuds emules -> UDP empaquetage, marshalling unmarch..

1.3 Gestion des ackuittements

AVEC LE BRIDGE; deux parametres importants a regler comme tradeoff

Soit le brdige a toutes les chances d'attraper un ack mais va attendre un monstre temps qui diminue les performances, soit il risque de ne jamais attraper d'ackittement et donc de modifier le comportement du protocole.

Regler dans la classe ch.unine.mac.MoteUARTCommunication la variable

MAX_EMULATED_BRIDGE_PART_WAITS_FOR_AN_ACK=300

ainsi que dans le make file du pont, la valeur -DCC2420_ACK_DELAY=200

All the package or software presented below can be found on the Freemote CD 1.0.

Si ce n'est pas clairement mentionne

2 Installing the Freemote Emulator

The Freemote Emulator can be distributed either as a Web Service or as a Developer form. Both distribution gives the full support of the Freemote Emulator features and can perform exactly the same experiments coupling or not the real nodes through the bridge. Moreover both distribution can perform user defined scenario by selecting the layer implementation and tunning the experimentation parameters. However, the Web Service distribution can use only the layer implementation already given in the distribution thus is not suitable for development. A developer must uses the Development distribution and could refer to the documentation (tutorial, architecture,).

This first part of this section describe how to install the environnment to provide a Freemote Emulator access through the Web. The second part describes the setup of the Development distribution.

2.1 Installing the Web Service Distribution

The Freemote Emulator can be distributed as a Web Service based ont he Java Web Start[2] technology. This technology is compatible with most of the web servers and is based on jnlp files. These files are XML formatted and mainly describe the classes that must be downloaded, the operating system dependant libraries and the parameters given to the main method. The web user starts a Freemote Emulator process by clicking on a hyperlink that refers to the jnlp file. The runtime environnment automatically fetches an up to date version on the website if needed. Moreover it is still possible to uses the Freemote Emulator offline once downloaded. We have build a web server[3] at University of Applied Science of Fribourg[4] that permanetly provide this service. This installation is mostly based on the following ressources.

Web Service Distribution Ressources to Install	
Ressource	***Description***
`apache_2.0.58-win32-x86-no_ssl.msi`	Apache web server
`mysql-4.1.20-win32.zip`	MYSQL SGBD
`php-5.1.4-Win32.zip`	PHP support for Apache

Tableau 2: Web Service Distribution Ressources to Install

There are numerous tutorials on the web that discribe how to install these softwares so we don't describe the operations in this documentation. However we used the followings documents[5] from the website www.developpez.com.

2.2 Installing the Developer Distribution

Developer Distribution Ressources to Install	
Ressource	***Description***
`JDK 1.5__?`	The Java Development Kit
`Eclipse___?`	An IDE to edit the source code ()
`Distribution___?`	The Developer Distribution of the Freemote Environment You can download the latest distribution directly from the Freemote Project website

Tableau 3: Developer Distribution Ressources to Install

How to? -> JDK 1.5 and that's it. Refer to the tutorial pour l'installation.

Importer le projet dans Eclipse avec les librairies, tout le toutim

Fixer la version du binaire generee a 1.3 pour compatibilite avec JMotes

Deux version, soit la version consistant à modifier le code existant -> installation avec les sources, ou alors pour le dev seulement -> avec le jar.

3 Installing the Real Nodes Environments

3.1 JMotes

2 http://java.sun.com/products/javawebstart/
3 Le fichier emulator_web_private_info.pdf contains the parameters to manages this server.
4 http://mote.tic.eia-fr.ch/
5 Installation et configuration d'un serveur web Apache 2.0 avec PHP 5 MySQL 4.1 et phpMyAdmin sous windows Java Web Start

JMote ressources to install	
Ressource	***Description***
`install3.16.09.exe`	aJile J2ME/CLDC 1.0 Runtime and Tools v. 3.16.9 Rev C
`JmoteAjileConfigs.zip`	Ajile tools configuration files for the JMotes
`BlueBee_MotherBoard1.1.1.zip`	CPLD design for the ML-053 board (AJ100-Pro)
`Bluebee_Card1.2.zip`	CPLC design for the pi044 board (ADS-BlueeBee)

Tableau 4: JMote ressources to install

The company aJile Systems provides the tools to create and load software on the JMote platform.

Design CPLD

Follow the instruction in the `rapportHardware1.0.pdf` to programm the two CPLD.

L'environnement de développement pour les Jmotes est fournit par aJile Systems[6]. Cet environnement inclue un linker JEMBuilder permettant de créer des binaires s'exécutant sur le processeur de la plateforme, ainsi que le chargeur Charrade permettant de programmer les JMotes. La version à installer est disponible (TODO). L'installateur peut être suivit dans sa configuration par défaut.

Lorsque l'installation est terminée, il faut encore désarchiver JmoteAjileConfigs.zip disponible dans TODO puis remplacer les fichiers C:\aJile\Charade\charade.wcf et C:\aJile\Charade\charade.cfg à partir de ceux contenus dans l'archive. Finalement, les configurations conteues dans l'archive doivent êtres copiées dans le répertoire C:\aJile\Configurations\.

3.2 Berkeley Motes

3.2.1 TinyOS Environment

TinyOS[7] is distributed in two main versions, the tinyos-1.x and the tinyos-2.x. The second is not backward compatible and is a clean redesign and re-implementation of the first version. We use the first version for the developement so we describe in this section how to install the developpment and deployment environment for TinyOS software. There are different ways to install this environment and we have choosen an installation based on a Linux live CD distributed by the Colorado School of Mines[8]. This distribution of Linux is based on the Xubuntu distribution and contains a pre-installed and up to date version of the two main version of TinyOS. This distribution is called XubuntOS.

You can download the lattest `ISO` archive of the bootable CD on the following website http://toilers.mines.edu/Public/XubunTOS or you can use the version stored on TODO. As you will need some third party software which run only on the Windows platform, we sugger to install this distribution on a virtual environment such as VMware on a Windows OS. To install these two softwares, follow the instructions given on this website http://sing.stanford.edu/klueska/installing_xubuntos_vm.html. You can found a pdf version of this document on TODO.

After the installation you may know some trouble to adjust the display settings of XubuntOS. In this case, install and use the `displayconfig-gtk` tool to fix this problem. Finally you may edit your `.bash_tinyos` in your home directory to set the first version of TinyOS as the default environment.

```
# Set your default version of TinyOS here
if [ "$TINYOS_VER" == "" ]; then
        export TINYOS_VER=1
fi
```

6 http://www.ajile.com
7 http://www.tinyos.net/
8 http://toilers.mines.edu/Public/

TODO Mettre à jour les fichiers sources de TINYOS qui ont été modifiés!!!! Modifications permttant de la gesetion des ackittements dans le driver du CC2420 dans /opt/tinyos-1.x/tos/lib/CC2420Radio

CC2420Const.h et CC2420RadioM.nc

Ces fichiers sont dans l'archive tos-patch.zip

TODO placer les fichiers de développement (Mbridge et TestAodve, ...)

3.2.2 MICAz Motes

asdf

3.2.2.1 DeviceInstaller

This tool is developped by Lantronx and permit to configure the programmation base MIB600CA. You can download the installator on this website http://www.lantronix.com/device-networking/utilities-tools/device-installer.html for free or you can found the version we used on TODO. You can follow the default installation for this tool.

3.2.3 TelosB Motes

Si utilisé pour le pont et l'émulateur s'exécute sur un environnement Windows (la distribution XubuntOS un tel outil installé) – alors installer FTDI virtual com port ou alors un adaptateur USB/Serie.

+ le javax.comm à installer

Execution

1 Executing the Freemote Emulator

Configurable par des fichiers XML, deux methodes d'envoi, soit en ligne de commande directement et edition du fichier à la main soit par l'edition du fichier totalement masquee a l'aide d'un user friendly GUI.

1.1 The Configuration GUI

The strcture and the meaning of the XML configuration file elements are detailed in WHERE PDF. The default configuration is WHAT. The configuration file GUI and the details of this interface are represented below.

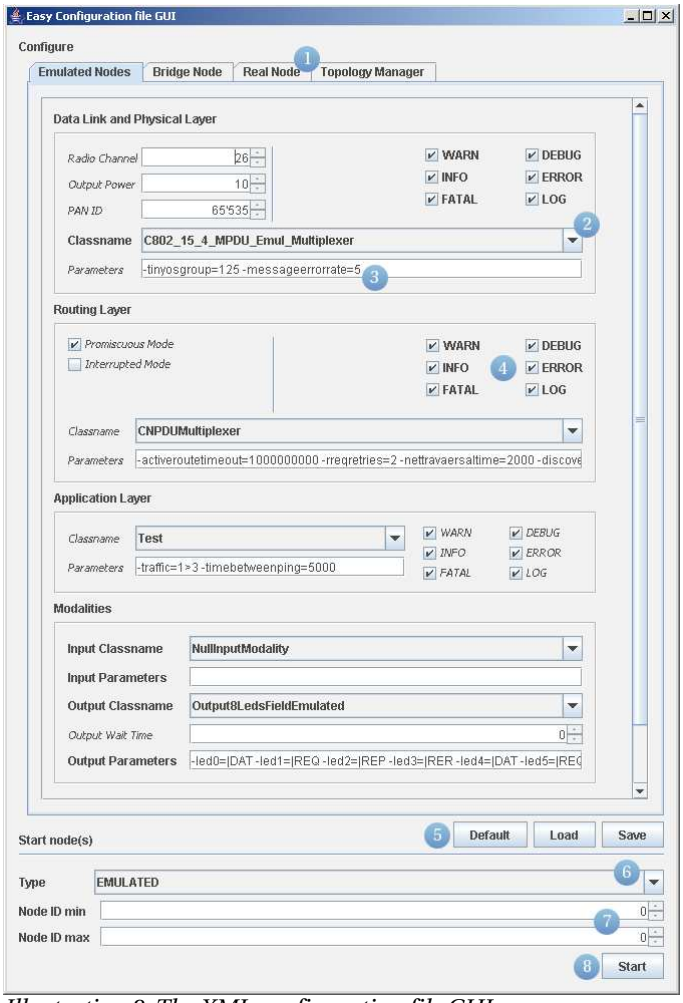


Illustration 2: The XML configuration file GUI

Details of the Configuration file GUI functions	
zone	***function***
1	In this zone you can edit the configuration file with your needs for a test.
2	This button loads the default configuration file in the editing zone.
3	This button open a file chooser frame. Use it to launch a preconfigured configuration file.
4	This button open a file saver frame. Use it to save your configuration file after you have sat it up.
5	This field contains the command line parameters that would have been used if you have started the application from a terminal. Notice that you don't need to specify the `type` and `configfile` parameters as the `type` is automatically set up to `EXEC_EMULATED` and the content of the configuration file will be the content of the editing zone.
6	This button loads the default command line parameters.
7	This button set the configuration file and the command line parameters to the emulator, close the configuration GUI and open the Freemote Emulator GUI.

Tableau 5: Details of the Configuration file GUI functions

1.1 Usage of the Freemote Emulator GUI

Once you have finished to configure the Freemote Emulator, a GUI which represents the nodes in a carthesian square map is started. An example of this GUI is shown on the Illustration 3. This GUI is splitted in three area. The first represents the emulated nodes on the left of the frame, the second represent the bridges nodes on the right. The third area is a console that displays the messages that would have been written on the standard output.

At the begining, the server that compute the physical connectivity of the emulated nodes network and the motion of the nodes is not active. This gives you time to starts other clients nodes by launching other Freemote Emulator instances. When you are ready, click on the following button to start the server.

Fisrt step, connect with start

Start motion si besoin

Connecter le bridge si matos sinon il ne va pas servir à grand chose... notez que le bridge n'a qu'un seul contact physique et qu'il restera identique tzout au long de la simulation quel que soit le mouvement des noeuds.

Bridge node takes the address 0 automatically.

The Tableau 1 contains details the components of the Freemote Emulator GUI.

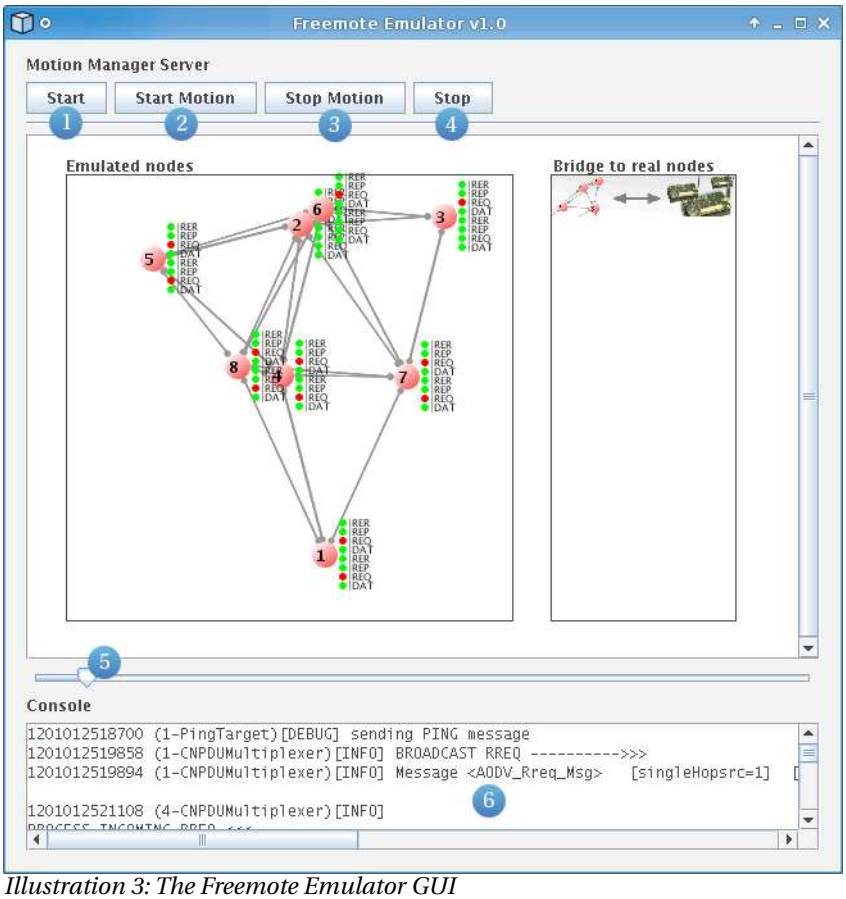


Illustration 3: The Freemote Emulator GUI

Details of the Freemote Emulator GUI	
zone	***function***
1	This button starts the process of computing the physical topology and the motion of the emulated nodes on the server.
2	This button starts the motion of the emulated nodes on the server.
3	This button stops the motion of the emulated nodes on the server.
4	This button quit the Freemote Emulator.
5	This button connect the bridge that make the connection between the emulated and the real motes. The bridge takes automatically the address 0.
6	This scroll bar controls the zoom level of the Emulated nodes and the Bridgre to real nodes area.

Tableau 6: Details of the Freemote Emulator GUI

1.2 Executing the Web Distribution

The freemote Emulator can be started directly from the http://mote.tic.eia-fr.ch website. At the beginning of the launch process,

This feature is based on the *Java Web Start*[9] technology (JSR-56) which let you starting standalone Java software applications in one click over a network. At the beginning of the launch process, a small GUI is started and let the user editing or choosing the XML configuration file and the command line parameters for the emulation. This documents

9 http://java.sun.com/products/javawebstart/

describes how to start the emulator and to configure it for your own tests.

1.2.1 Java Web Start installation check

The Freemote Emulator need a JRE 5.0 or an upper version to run. This version include by default the support for Java Web Start. You can check at the beginning of the launching page if your system is properly installed by reading the installation status. The table below gives a comprehensive explanation of these messages.

1.2.2 Launching and configuring the Emulator

If Java Web Start is properly installed on your system, you are ready to launch the Freemote Emulator by clicking on the following button.

Then the Java Web Starts system fetches the jar files that you need and then displays a security warning message that ask you if you accept to execute the application. Indeed the Freemote Emulator doesn't run in a sanboxed environment as the user may save and read XML configuration files from the disk. So the Freemote Emulator is a signed application and you must trust our certificate to run it.

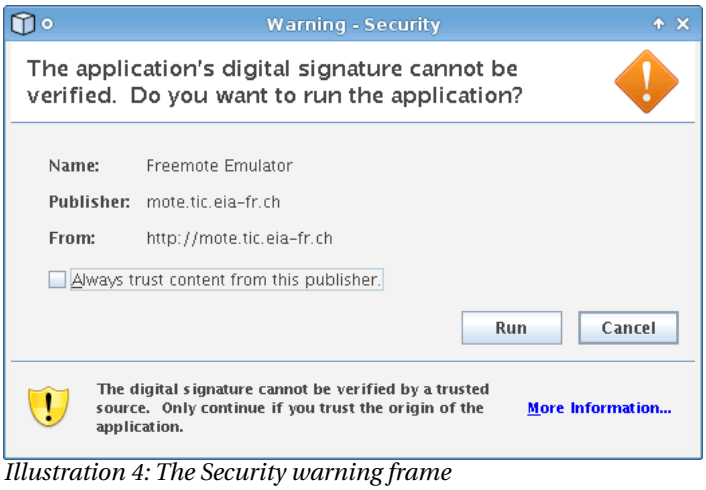


Illustration 4: The Security warning frame

Once you have accepted the certificate, the configuration GUI will be displayed. This interface let you configure the XML file and the command line parameters which are also needed if you starts the Freemote Emulator from a terminal. If you want to save time between two identical tests or to run a test not only with more than one computer, you may save the configuration file on a shared disk. When you have configured the emulator for your test, click on the following button which close the configuration GUI and open the Freemote Emulator GUI.

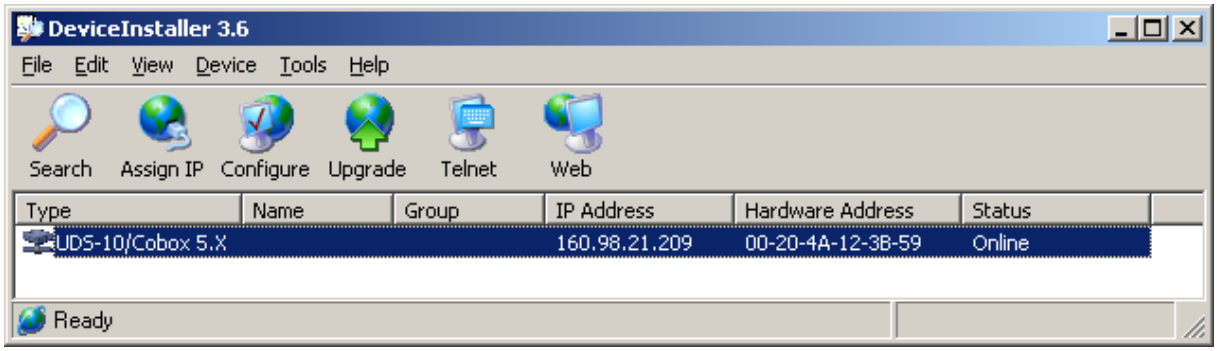
1.3 Executing the Developer Distribution

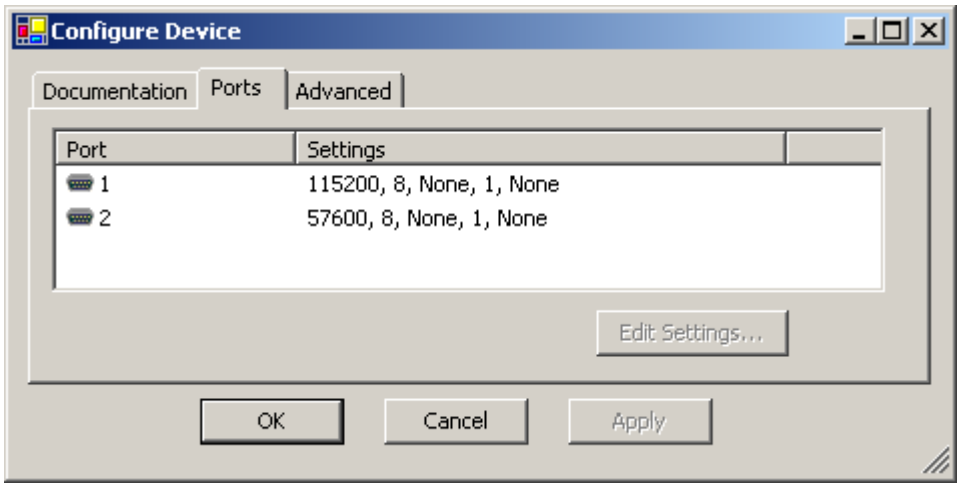
1.4 Executing the Bridge Node

Le programme a charger pour les Motes (Mbridge), avec quelle adresse (la meme que celle programmee sur la partie emulee, par convention on met 0)

Refer to the section ?????(Programming the MICAz Motes) to program the program the MICAz or (Programming the TELOSB)

1.4.1 Executing the Bridge Node with MICAz





1.4.2 Executing the Bridge Node with TelosB

1.4.3 Executing the Bridge Node With Real Part Based on MICAz

S'assurer que les settings dans DeviceInstaller soient bien configurés -> images et tout

1.4.4 Executing the Bridge Node with Real Part Based on TelosB

S'assurer que le PORT COM Virtuel soit bien configuré

Menu: VM > Removable Devices > USB Devices

2 Executing the Real Nodes

2.1 Executing the Jmote

Configuration des CPLD et autre FPGA s'il en est

Configurations JEMBuilder – différents printscreen ou seulement les configs

Les configs doivent de toute façon êtres adaptées ...

Ouvrir Charrade, et bla bla?

Comment placer les jumpers? Jusqu'ou est-ce qu'il faut aller?

2.2 Executing the Berkeley Motes

2.2.1 Programming the MICAz Motes

The MICAz Motes are programmed with the MIB600 programming base. Assure that a MICAz Mote is monted on the base and that the base is connected to your LAN and powered before any operation. First of all we need to know the IP address of the programming base. Launch the DeviceInstaller software and click on the the ???????? button. This opration will list all the bases reachable through the LAN and permit to know the IP address of the base.

??? FIGURE DU DEVICE INSTALLER AVEC UNE BASE TROUVEE

Then open a terminal and go on the directory of the application you need to programm on the Mote. Then type the following command

```
> make micaz [re]install,<nodeID> eprb,<MIB600 IP>
```

```
> make micaz [re]install,<nodeID> eprb,<MIB600 IP> (As an example)
```

The `<nodeID>` is the 802.15.4 16bits node address and the `<MIB600 IP>` is the IP addres of the base from copied from the DeviceInstaller. The `install` command compile and the program the Mote and the `reinstall` command only program the mote with the already created binary.

Once programmed the application will automatically run on the Mote and every time you turn off and on the program is also automatically launched.

2.2.2 Programming the TelosB Motes

First of all you need to connect a TelosB on an USB port of your computer. If you uses XuBunTOS on WMware, assume that the port is selected on the menu ?????????????. Then you can type the following command.

```
> motelist
```

This command displays the USB connected Motes. The output should look like the following.

```
Reference  Device         Description
---------- -------------- ----------------------------------------
XBNU4PE1   /dev/ttyUSB0   XBOW Crossbow Telos Rev. B
```

Then you can type the following command to program the Mote.

```
> make telosb [re]install,<nodeID> bsl,<Device>
```

```
> make telos install,55 bsl,/dev/ttyUSB0 (As an example)
```

The `<nodeID>` is the 802.15.4 16bits node address and the `<Device>` is copied from the `motelist` command output. The `install` command compile and the program the Mote and the `reinstall` command only program the mote with the already created binary.

Once programmed the application will automatically run on the Mote and every time you turn off and on the program is also automatically launched.

3 The emulator

If you want to access the serial (RS232) or parallel port with JAVA, you need to install a platform/operating system dependent library. Install either javax.comm from SUN (no windows version as of javax.comm version 3.0) or better install the rxtxSerial and/or rxtxParallel library from rxtx.org (Windows, Linux, Mac OS X).

rxtx-2.1-7-bins-r2.zip

– Download javacomm20-win32
  - win32com-dll to /bin
  - javax.comm.properties to /lib
  - comm.jat to /lib
–

and then install the FTDI Driver pour pouvoir simuler un port USB comme un port série sur votre machine. (pour la distribution de linux pas besoin - CDM 2.02.04 WHQL Certified.zip) Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the same way as it would access a

standard COM port. Ou alors utiliser un cable faisant la conversion serial / usb

TRES TRES IMPORTANT, apres avoir installé FTDI Virtual Com port, il faut impérativement fixer les paramètres du port tel que bit rate Dans le Device Manager de Windows en fonction de la plateforme -> référence vers la fin

Bridge node

Matériel requis	
Plateforme de programmation pour les motes ethernet	Crossbow MIB600CA
MICAz mote	Crossbow

Logiciel requis	
XubunTOS	
VMware	
DeviceInstaller	http://www.lantronix.com/
Arbre de développement (Mbridge, etc)	

Configuration Format

This section describes the format of the elements of the XML configuration file and the command line arguments. These are needed for the Java nodes emulated or the JMotes.

1 Command line arguments

The string of parameters for the main method must match the syntax listed in the Tableau 7.

Command line syntax

```
parameter  ::= "-type=" type ["-configgui" | "-configfile=" configfile "-nodeidmin=" nodeid "-nodeidmax=" nodeid]
type       ::= .. The type of target to launch in {EXEC_REAL, EXEC_EMULATED, BRIDGE, AGGREGATOR} ..
configfile ::= .. The path to the XML configuration file that contains setup for the environment
                  This path is either a relative path from the ch.unine.launcher package (eg. emulation.xml) or
                  an absolute path (eg. /ch/unine/launcher/emulation.xml) ..
nodeid     ::= .. The 16 bits (0 to 65535) MAC level and routing level address of a node ..
```

Tableau 7: Command line syntax

The `EXEC_REAL`, `AGGREGATOR` and `BRIDGE` target only starts one node with the address `-nodeidmin`. The `-configgui` type is not suitable for the `EXEC_REAL` type since this mode let the user edit the XML config file from a small GUI. As examples, the Tableau 8 contains valid strings of parameters.

Command line examples
-type=EXEC_EMULATED -configgui
-type=EXEC_EMULATED -configfile=/ch/unine/launcher/emulation.xml -nodeidmin=1 -nodeidmax=55
-type=EXEC_REAL -configfile=/ch/unine/launcher/emulation.xml -nodeidmin=33 -nodeidmax=33
-type=AGGREGATOR -configfile=emulation.xml -nodeidmin=30000 -nodeidmax=30000
-type=BRIDGE -configfile=emulation.xml -nodeidmin=0 -nodeidmax=0

Tableau 8: Command line examples

Type of target	
type	***description***
EXEC_REAL	An evaluation node that runs on a CLDC1.0 compatible device (launcher and application only use Java J2ME features)
EXEC_EMULATED	Some evaluation nodes that run on the emulated environment
BRIDGE	An evaluation node that runs on the emulated environment. Messages sent by this node are also sent on the real radio medium through a mote programmer
AGGREGATOR	An aggregator node that runs on the emulated environment

Tableau 9: Type of target

2 XML configuration file

This section presents the meaning of every textual elements and element parameters of the XML configuration file but does not describe the configuration file structure. We have seen in which part that some part are configurable for every experimentation such as the layers implementation, the mobility model, the output modylity implementation, etc. Every implementation is configurable following a specific format which are described in the second part of this section. The first part describe the format for the parameters that don't depend on the implementation.

2.1 General XML configuration file element content

General XML configuration file content	
name	***description***
`emulation-env.` `mac-layer#`**`radiochannel`**	The IEEE 802.15.4 radio channel in {11,...,26}
`emulation-env.` `mac-layer#`**`outputpower`**	The radio output power in {0,...,31} This parameter only affects the real nodes
`emulation-env.` `mac-layer#`**`panid`**	The IEEE 802.15.4 PANID in {0,...,65535}
`emulation-env.` `mac-layer.`**`parameters`**	The String of parameters for a specific implementation of the MAC layer for every type of target. (1)

`emulation-env.`
`mac-layer.`
`real-node#`**`classname`**
	The class name implementation of the MAC layer for the `EXEC_REAL` target.
	This class must be in the `ch.unine.mac` package and implement the `I802_15_4_MPDU_Multiplexer` interface.

`emulation-env.`
`mac-layer.`
`real-node#`**`logging-levels`**
	(2)

`emulation-env.`
`mac-layer.`
`emulated-node#`**`classname`**
	The class name implementation of the MAC layer for the `EXEC_EMULATED` and the `AGGREGATOR` targets.
	This class must be in the `ch.unine.mac` package and implement the `I802_15_4_MPDU_Multiplexer` interface.

`emulation-env.`
`mac-layer.`
`emulated-node#`**`logging-levels`**
	(2)

`emulation-env.`
`mac-layer.`
`bridge-node#`**`gateway`**
	A boolean value in {true,false}. If the value is false, the `BRIDGE` target use the MIB600CA to exchange messages with the real nodes. If the value is false, the `BRIDGE` use the gateway to exchange messages to the real nodes.
	Note: *The gateway mode is not implemented yet.*

`emulation-env.`
`mac-layer.`
`bridge-node#`**`url`**
	In gateway mode: The url of the gateway
	In non gateway mode: The url of the serial interface to the motes
	MICAz and MIB600CA: `network@<MIB600CA IP address>:10002`
	TELOS: `serial@<PORT>:<SPEED>`
	On a *nix OS, the PORT is one of those listed in the `motelist` command output. eg. `/dev/ttyUSB0`
	On a windows OS, the PORT is the name of the COM port. eg. COM0
	Note: *The Java Communication API (javax.comm) must be installed on your system*
	
	`PLATFORM SPEED`
	`----------`
	`telos 115200`
	`telosb 115200`
	`tmote 115200`
	`micaz 57600`
	`mica2 57600`
	`mica2dot 19200`
	`eyes 115200`
	`intelmote2 115200`

`emulation-env.`
`mac-layer.`
`bridge-node#`**`classname`**
	The class name implementation of the MAC layer for the `BRIDGE` target.
	This class must be in the `ch.unine.mac` package and implement the `I802_15_4_MPDU_Multiplexer` interface.

General XML configuration file content	
emulation-env. mac-layer. bridge-node#**logging-levels**	(2)
emulation-env. routing-layer#**classname**	The class name implementation of the routing layer for the EXEC_REAL, EXEC_EMULATED, BRIDGE and AGGREGATOR targets. This class must be in the ch.unine.routing package and implement the INPDUMultiplexer interface.
emulation-env. routing-layer#**promiscuousmode**	A boolean value in {true,false}. If the value is false, a node receives the NPDU sent by its physical neighbors only if the multi hop destination or the single hop destination match its address. If the value is true, a node receive every NPDU sent by its physical neighbors.
emulation-env. routing-layer#**interruptedmode**	A boolean value in {true,false}. If the value is true, the intermediates nodes on a route don't automatically forward the NPDU to the destination. This operation is done with an explicit call to the forwarding method. If the value is false, the routing layer automatically forwards the NPDU.
emulation-env. routing-layer.**parameters**	The string of parameters for a specific implementation of the ROUTING layer for every type of target. (1)
emulation-env. routing-layer. real-node#**logging-levels**	(2)
emulation-env. routing-layer. emulated-node#**logging-levels**	(2)
emulation-env. routing-layer. bridge-node#**logging-levels**	(2)
emulation-env. application-layer. execution-node#**logging-levels**	(2)
emulation-env. application-layer. execution-node.**classname**	The class name implementation of the APPLICATION layer for the EXEC_REAL, EXEC_EMULATED and BRIDGE targets. This class must be in the ch.unine.target package and implement the IapplicationLayerTarget interface.
emulation-env. application-layer. execution-node.**parameters**	The string of parameters for a specific implementation of the APPLICATION layer for the EXEC_REAL, EXEC_EMULATED and BRIDGE targets. (1)
emulation-env. application-layer. aggregator-node#**logging-levels**	(2)
emulation-env. application-layer. aggregator-node.**classname**	The class name implementation of the APPLICATION layer for the AGGREGATOR target. This class must be in the ch.unine.target package and implement the IapplicationLayerTarget interface
emulation-env. application-layer. aggregator-node.**parameters**	The string of parameters of the APPLICATION layer for the AGGREGATOR target. (1)
emulation-env. modalities. input-modality. real-node#**classname**	The class name implementation of the input modalities for the EXEC_REAL target. The class must be in the ch.unine.modality package and implement the IInputModality interface.
emulation-env. modalities. input-modality. emulated-node#**classname**	The class name implementation of the input modalities for the EXEC_EMULATED and AGGREGATOR targets. The class must be in the ch.unine.modality package and implement the IInputModality interface.
emulation-env. modalities. input-modality. bridge-node#**classname**	The class name implementation of the input modalities for the BRIDGE target. The class must be in the ch.unine.modality package and implement the IInputModality interface.

General XML configuration file content	
emulation-env. modalities. input-modality.**parameters**	The string of parameters for an implementation of the input modality for every type of target. (1)
emulation-env. modalities. output-modality#**waittime**	The time in ms that every method must wait after outputing something (slow down the execution).
emulation-env. modalities. output-modality. real-node#**classname**	The class name implementation of the output modalities for the EXEC_REAL target. The class must be in the ch.unine.modality package and implements the IOutputModality interface.
emulation-env. modalities. output-modality. emulated-node#**classname**	The class name implementation of the output modalities for the EXEC_EMULATED and AGGREGATOR targets. The class must be in the ch.unine.modality package and implements the IOutputModality interface.
emulation-env. modalities. output-modality. bridge-node#**classname**	The class name implementation of the output modalities for the BRIDGE target. The class must be in the ch.unine.modality package and implements the IOutputModality interface.
emulation-env. modalities. output-modality.**parameters**	The string of parameters for an implementation of the output modality for every type of target. (1)
emulation-env. connectivity-server#**displaymap**	A boolean value in {true,false}. If the value is true, the main GUI displays the map with the EXEC_EMULATED, AGGREGATOR and BRIDGE nodes.
emulation-env. connectivity- server#**mintimebetweenupdates**	The minimal time in ms between two updates of the emulated nodes physical topolgy by the server.
emulation-env. connectivity-server#**connectivity-server-id**	An id for the server on the MotionManagerDispatcher. Note: *This id let the MotionManagerDispatcher running more than one server on the same machine.*
emulation-env. connectivity-server.**host**	The IP address or the name of the machine that runs the MotionManagerDispatcher and the MotionManagerServer.
emulation-env. connectivity-server.**port**	The TCP port that the MotionManagerDispatcher opens to listen to client connections.
emulation-env. connectivity-server. mobilitymodel#**radiorange**	The default physical radio range in meter for the EXEC_EMULATED and AGGREGATOR targets.
emulation-env. connectivity-server. mobilitymodel#**connectivityaverage**	The average physical connectivity for the EXEC_EMULATED and AGGREGATOR targets.
emulation-env. connectivity-server. mobilitymodel.**class**	The class name implementation of the mobility model for the EXEC_EMULATED and AGGREGATOR targets. The class must be in the ch.unine.motionmanager.mobility package and implements the IMobilityModel interface.
emulation-env. connectivity-server. mobilitymodel.**parameters**	The string of parameters for the implementation of the mobility model. (1)

Tableau 10: Genereal XML configuration file content.

(1) The format depends on the implementation. See the implementation detail part of this document or the implementation file for details.

(2) The list of enabled logging levels in {WARN, INFO, FATAL, DEBUG, ERROR, LOG} . The levels separator is the pipe |, eg. DEBUG|WARN|LOG

2.2 Implementation specific XML configuration file element content

TODO Add a row that describe the parameters meaning and not only the syntax.

2.2.1 MAC layer

TODO Describe the 3 MAC layers and their parameters

2.2.2 Routing layer

Routing layer: TODO	
Class name	
Description	
Parameters syntax	
Example	

Tableau 11: TODO

1. TODO changer le nom de l'implémentation du mux qui utilise AODV pour CNPDUMultiplexerAODV

2.2.3 Application layer

Application layer: No application	
Class name	`NullTarget`
Description	This application does nothing.
Parameters syntax	`NO PARAMETERS`

Tableau 12: No application

Application layer: Ping application	
Class name	`PingTarget`
Description	This application defines some nodes as ping messages sender and some other as ping messages receivers. Every sender periodically send ping messages to a multihop destination node and display some statistics when they receive a ping response. These statistics include the precentage of successfully ping messages sent (request and reply), the RRT (Round Trip Time), the TTL (Time To Live) which is decremented at every node on the ping reply route (starts with 65535).

Parameters syntax	`parameter ::= "-traffic=" (srctodest (";" srctodest)*)? "-timebetweenping=" period`
	`srctodest ::= nodeid ">" nodeid`
	`period ::= .. The time in ms between two ping sending (long) ..`
	`nodeid ::= .. Identify the node address (int) ..`

Example	`-traffic=1>3;4>5;5>6 -timebetweenping=2000`

Tableau 13: Application layer: Ping application

Application layer: DHT aggregator application	
Class name	`DhtAggregatorTarget`
Description	This application collects the DHT evaluation result messages, manage the DHT evaluation operations and write the results in an outputfile. TODO explain the parameters + starts a small GUI + how does it send command to the nodes, how does it sends the parameters

Parameters syntax	`Parameter ::= "-mintimebetweenlookups=" time "-overlayinitnodeid=" nodeid "-expectednbofnodes=" nbofnodes "-caching=" boolean "-non=" boolean "-implicitoverlay=" boolean`
	`tim ::= .. The time between two LOOKUP requests in ms (long) ..`
	`nodeid ::= .. Identify the node address (int) ..`
	`nbofnodes ::= .. The number of nodes that executes the DhtExecutionTarget in the network + 1 (int) ..`
	`boolean ::= .. A value that indicates if a feature is enabled or not (boolean) ..`

Example	`-mintimebetweenlookups=5000 -overlayinitnodeid=99 -expectednbofnodes=101 -caching=true -non=true -implicitoverlay=true`

Tableau 14: DHT aggregator application

TODO inclure le tuto sur l'utilisation de l'application permettant de tester la DHT.

Application layer: DHT execution application	
Class name	`DhtExecutionTarget`
Description	This application build an overlay network and generates periodically random LOOKUP request over the DHT. TODO explain the effect of the parameters
Parameters syntax	`NO PARAMETERS`

Tableau 15: DHT execution application

2.2.4 Input modality

Input modality: No input modality	
Class name	`NullInputModality`
Description	This class implements the default input modality that capture nothing.
Parameters syntax	`NO PARAMETERS`

Tableau 16: No input modality

2.2.5 Output modality

Output modality: No output modality	
Class name	`NullOutputModality`
Description	This class implements the default output modality that display nothing.

Output modality: No output modality	
Parameters syntax	`NO PARAMETERS`

Tableau 17: No output modality

Output modality: Real node 8 leds field	
Class name	`Output8LedsFieldJMote`

Description	This class implements the `IOutputLedsField` interface which provides methods to activate and disactivate 8 leds one by one. This implementation is suitable for the `EXEC_REAL` target and more specifically the JMote platform with the `ML063` board (`ADAPTER 120p-WW-MICTOR`) and the `8xLEDS + LATCH` module.
	8 wires between these two boards are connected like the following
	`D0 <-> 118`
	`D1 <-> 117`
	`D2 <-> 116`
	`D3 <-> 115`
	`D4 <-> 114`
	`D5 <-> 113`
	`D6 <-> 112`
	`D7 <-> 111`

Parameters syntax	`NO PARAMETERS`

Tableau 18: Output modality: Real node 8 leds field

Output modality: Emulated node 8 leds field	
Class name	`Output8LedsFieldEmulated`
Description	This class implements the `IOutputLedsField` interface which provides methods to activate and disactivate 8 leds one by one. This implementation is suitable for the `EXEC_EMULATED`, `AGGREGATOR` and `BRIDGE` target. The leds are displayed on the GUI on the right of every nodes. This implementation also display a radio cover and the message content over the nodes that send message.

Parameters syntax	`parameter ::= "-led0=" description "-led1=" description "-led2=" description`
	`"-led3=" description "-led4=" description "-led5=" description`
	`"-led6=" description "-led7=" description`
	`description ::= .. The description which define the meaning of a led ..`

Example	`-led0=|DAT -led1=|REQ -led2=|REP -led3=|RER -led4=|DAT -led5=|REQ -led6=|REP -led7=|RER`

Tableau 19: Output modality: Emulated node 8 leds field

2.2.6 Mobility model

Mobility Model: Fixed model	
Class name	`FixedModel`
Description	This class implements a fixed mobility model which defines repeatable scenarios. The angle, speed, radio range and initial position could be fixed for every nodes. Use the class FixedModelGenerator to create random fixed model parameters.

Parameters syntax	`parameter ::= "-DESCRIPTION=" (description ("#" description)*)?`
	`description ::= nodeid "," initx "," inity "," speed "," angle "," radiovariation`
	`nodeid ::= .. Identify the node address (int) ..`
	`initx ::= .. Position in x for the node in percentage of the network map size (float in the range [0.0,...,1.0]) ..`
	`inity ::= .. Position in y for the node in percentage of the network map size (float in the range [0.0,...,1.0]) ..`
	`angle ::= .. The angle in radian for the motion (float in the range [0.0,...,2*Math.PI]) ..`
	`speed ::= .. The speed in meter per second that the node will move (float) ..`
	`radiovariation ::= .. The radio range variation in percent of the default range, 1.0 means no variation (float in the range [0.0,...,Float.MAX_VALUE]) ..`

Example	`-DESCRIPTION=1,0.16,0.5,0.0,0.0,0.0#2,0.8,0.5,0.0,0.0,0.0#3,0.32,0.5,0.0,0.0,0.0`

Tableau 20: Mobility Model: Fixed model

Mobility Model: Random uniform model	
Class name	`RandomUniformModel`
Description	This class implements the Random Uniform mobility model. The motion direction and speed is random but remain constant.
Parameters syntax	`parameter ::= "-MAXSPEED=" speed` `speed ::= .. A speed in meter per second (float) ..`
Example	`-MINSPEED=2.1 -MAXSPEED=3.9`

Tableau 21: Mobility Model: Random uniform model

Mobility Model: Random walk model	
Class name	`RandomWalkModel`
Description	This class implements the Random Walk mobility model. The nodes pick up randomly a speed and direction at every step.
Parameters syntax	`parameter ::= "-MINSPEED=" speed "-MAXSPEED=" speed` `speed ::= .. A speed in meter per second (float) ..`
Example	`-MINSPEED=2.1 -MAXSPEED=3.9`

Tableau 22: Mobility Model: Random walk model

Index of Tables

Summary

Summary

Document Versions		
Date / Version	***Comment***	***Author***
Jan 31, 2008 / 0.1	Redaction of the initial version	T. M.