WiSE-MNet++

Wireless Simulation Environment for Multimedia Sensor Networks

Installation manual

version 1.4

Original version by Christian Nastasi (nastasichr@gmail.com)
and Andrea Cavallaro (a.cavallaro@qmul.ac.uk)

Updated by Juan C. SanMiguel (juancarlos.sanmiguel@uam.es)

October 11, 2017

Contents

1 Introduction

The Wireless Simulation Environment for Multimedia Sensor Networks (WiSE-MNet++) has been designed to simulate distributed algorithms for Wireless Multimedia Sensor Networks (WMSNs) under realistic network conditions. The simulation environment is based on one of the most popular network simulator: *OMNeT++* . Among the several simulation models for the OMNeT++ environment, *Castalia* is the one that has been designed with similar goals, although it focuses on classic Wireless Sensor Networks (WSNs).

WiSE-MNet++ is proposed as an extension of the Castalia/OMNeT++ simulator. The main extensions provided to the Castalia simulation model can be summarized as:

- generalization of the sensor data-type (from scalar-based to any type);
- idealistic communication and direct application communication;
- concrete modules for sensing and processing: moving target, camera modeling, target tracking application.
- simple GUI for 2D world representation;

We assume the reader to be familiar with the OMNeT++ environment and to know the basics about the Castalia simulation model. The reference versions over OMNeT++ and Castalia are respectively the 5.0 and the 3.1 . Documentations and tutorials about OMNeT++ can be found at the project documentation page `http://www.omnetpp.org/documentation`. We particularly suggest to read the User Manual first and then to use the API Reference when developing new modules. A copy of the Castalia user's manual is available at `http://castalia.npc.nicta.com.au/documentation.php`.

2 Prerequisites: OMNeT++

WiSE-MNet++ is based on OMNeT++ and is an extension of the Castalia simulation model. WiSE-MNet++ has been developed using the version 5.0 of OMNeT++ and the 3.1 version of Castalia. Although OMNeT++ is available for Windows systems, Castalia has been designed for *GNU/Linux*-like systems (see Castalia reference manual). For this reason we strongly recommend to use a *GNU/Linux*-like system to use WiSE-MNet++ (the Ubuntu GNU/Linux distribution has been successfully used). However, installation for Windows systems might be possible through the *Cygwin* environment, although this has not been tested.

2.1 Installing OMNeT++

WiSE-MNet++ requires to install OMNeT++ and OpenCV. For this tutorial, Ubuntu 16.04 LTS is used and the Ubuntu-specific command are indicated. Please refer to the Omnet or OpenCV installation guides for other operative systems (OS). Please launch a *bash* shell and type the following commands:

1. Before installing please make sure that the system is updated and upgraded (*Ubuntu-specific* command)

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

2. Get the latest source files from github at `https://github.com/MultiCameraNetworks/wisemnet`

```
$ wget https://git.io/vdrEd -O wisemnet-v1.4.zip
$ unzip -qq -o wisemnet-v1.4.zip -d .
```

A folder `wisemnet-v1.4` will be created.

2.1.1 Bash script

For full instructions and details about the installation of OMNeT++ , refer to the Linux section of the OMNeT++ installation guide (also available at `http://www.omnetpp.org/pmwiki/index.php?n=Main.InstallingOnUnix`). The following steps should be performed for a fresh installation of OMNeT++ .

If Omnetpp is not installed, please run the script *install_opencv.sh* to proceed with the installation steps.

```
$ cd wisemnet-v1.4/utils
$ sudo ./install_omnetpp.sh
```

2.1.2 Instructions

In this guide, we used the version 5.0 which is available at download page. However, newer omnetpp versions can be installed (available at download most recent version) but they have not been tested..

1. We assume to work in the home directory

   ```
   $ cd ~
   ```

2. Install required dependencies for OMNeT++ . (*Ubuntu-specific* command)

   ```
   $ sudo apt-get install build-essential gcc g++ bison flex perl qt-default tcl-dev tk-
       dev libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-3.0-0
   ```

3. To enable parallel suport, install the required packages

   ```
   $ sudo apt-get install openmpi-bin libopenmpi-dev
   ```

4. Get the OMNeT++ sources for the 5.0 Linux version here. A file `omnetpp-5.0-src.tgz` will be created.

5. Extract the source files

   ```
   $ tar xvzf omnetpp-5.0-src.tgz
   ```

   A folder `omnetpp-5.0` will be created.

6. Set the environment variables to point to the OMNeT++ binary paths:

   ```
   $ export PATH=$PATH:~/omnetpp-5.0/bin
   $ export LD_LIBRARY_PATH=~/omnetpp-5.0/lib
   ```

   These two lines should be also appended to the `~/.bashrc` file.

7. Compile OMNeT++ [1]

   ```
   $ cd omnetpp-5.0
   $ . setenv
   $ ./configure
   $ make
   ```

8. OMNeT++ should be successfully installed. The following command can be used to verify that the OMNeT++ executables are in the execution path.

   ```
   $ which opp_makmake
   ```

9. Testing the installation

   ```
   $ cd samples/dyna
   $ ./dyna
   ```

   After clicking through some options, you should see the output depicted in Figure 1.

WiSE-MNet++ is also compatible with Omnet++ 4.4.1 and 4.6, which can be downloaded at download page . Please follow the instructions (steps 4 to 7) to test the correct installation and setup of Omnet++.

[1] NOTE: if you have a multi-core machine, compilation will be faster by running the make command with the '-j' option and passing the number of cores plus one as argument. For instance, in a dual-core machine use '`make -j 3`'.

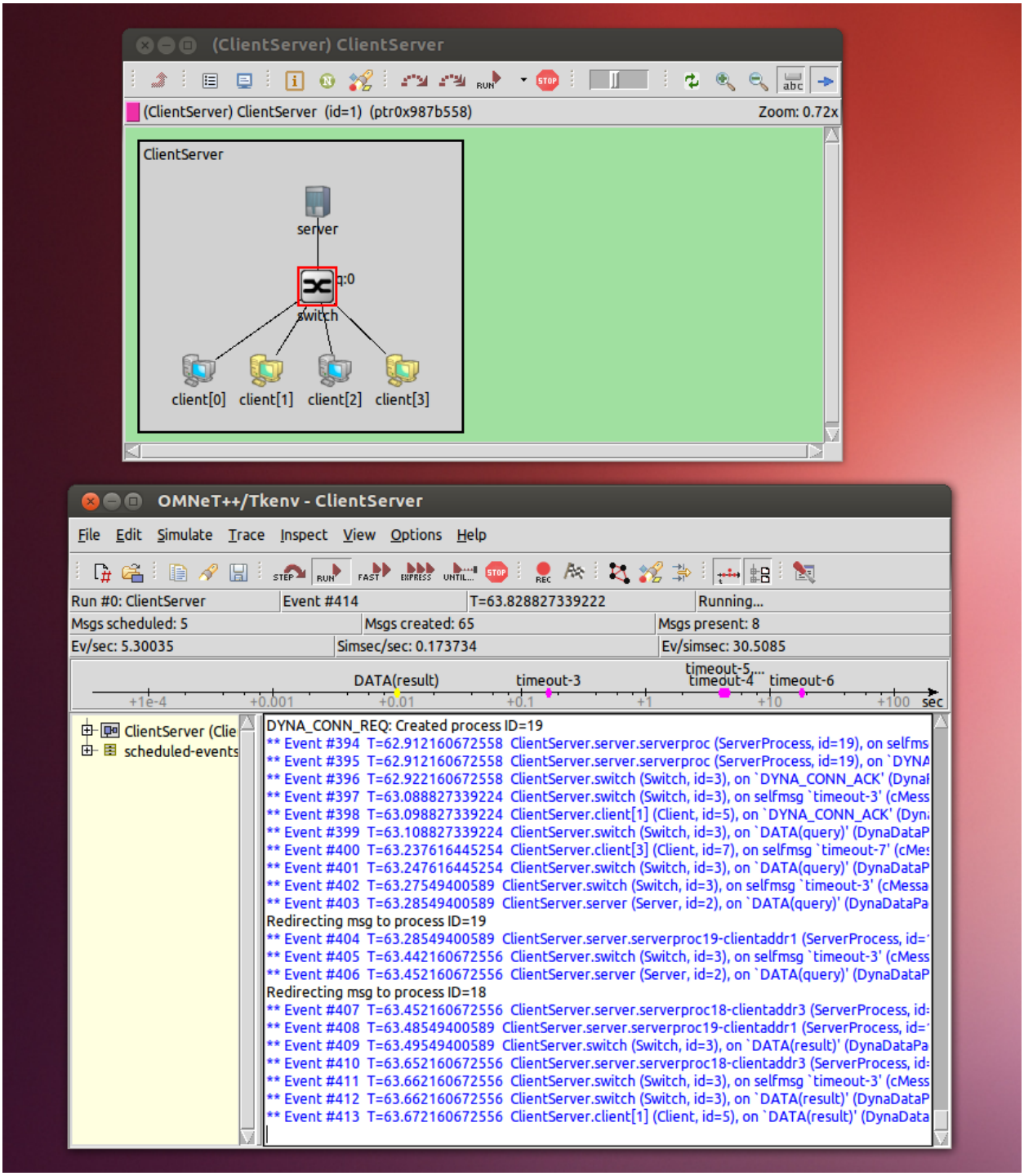

Figure 1: Output of the *dyna* application of OMNeT++ after a successful install which uses the Tkenv GUI to show simulation results. For tutorials and help regarding OMNeT++ and Tkenv, please refer to `https://omnetpp.org/doc/omnetpp/UserGuide.pdf`

3 Prerequisites: OpenCV

For installation instruction of the OpenCV library under GNU/linux, please refer to here. The following steps should be performed for a fresh installation of OpenCV after OMNeT++ . In this guide, we used the version 3.2.0 which is available at github. You can the current version installed in your OS by typing:

```
$ pkg-config --modversion opencv
```

3.1 Bash script

If OpenCV is not installed, please run the script *install_opencv.sh* to proceed with the installation steps.

```
$ cd wisemnet-v1.4/utils
$ sudo ./install_opencv.sh
```

3.2 Instructions

In this document, we provide snippets from a short tutorial but other sources may be employed as long as OpenCV is properly installed.

1. We assume to work in the home directory

   ```
   $ cd ~
   ```

2. Install required dependencies for OpenCV. (*Ubuntu-specific* command)

   ```
   $ sudo apt-get -y install build-essential cmake git libgtk2.0-dev pkg-config python-
       dev python-numpy libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev libjasper
       -dev libavcodec-dev libavformat-dev libswscale-dev libgstreamer0.10-dev
       libgstreamer-plugins-base0.10-dev libv4l-dev libtbb-dev libqt4-dev libfaac-dev
       libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev
       libvorbis-dev libxvidcore-dev x264 v4l-utils unzip
   ```

3. Install the version 3.2.92 of the *Eigen3* package. While WiSE-MNet++ may work with other *Eigen3* versions, some compatibility issues may exist among the different version of *Eigen3*. We advice here to install the version used during the testing phase. (*Ubuntu-specific* command)

   ```
   $ sudo apt-get install libeigen3-dev=3.2.92
   ```

4. Get the OpenCV sources from the download page or type.

   ```
   $ cd ~
   $ wget https://github.com/opencv/opencv/archive/3.2.0.tar.gz && mv 3.2.0.tar.gz
       opencv-3.2.0.tar.gz
   ```

5. Extract the source files

   ```
   $ tar -zxvf opencv-3.2.0.tar.gz
   ```

   A folder `opencv-3.2.0` will be created.

6. Similarly we download and extract the source files for the extended module of OpenCV

   ```
   $ wget https://github.com/opencv/opencv_contrib/archive/3.2.0.zip && mv 3.2.0.zip
       opencv_contrib-3.2.0.zip
   $ unzip opencv_contrib-3.2.0.zip -d opencv-3.2.0/
   ```

   A folder `opencv-3.2.0/opencv_contrib-master` will be created.

7. Generate the Makefile to compile OpenCV

```
$ cv opencv-3.2.0
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -DWITH_OPENGL=
    ON -DWITH_VTK=ON -DWITH_TBB=ON -DWITH_GDAL=ON -DWITH_XINE=ON -DBUILD_EXAMPLES=ON
     -DBUILD_TIFF=ON -D BUILD_PYTHON_SUPPORT=ON -DOPENCV_EXTRA_MODULES_PATH=../
    opencv_contrib-3.2.0/modules ..
```

A folder `build` will be created.

8. Compile OpenCV sources

```
$ make
$ sudo make install
```

9. Configure OpenCV (*Ubuntu-specific* command) by editing the configuration file

```
$ sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
$ sudo ldconfig
```

10. To check whether the OpenCV library paths are correctly installed

```
$ pkg-config opencv --cflags
$ pkg-config opencv --libs
$ pkg-config --modversion opencv
```

These should print the include paths and linker options of OpenCV to be used by WiSE-MNet++ .

11. Testing the installation

```
$ cd ~/OpenCV-3.2.0/samples/c
$ chmod +x build_all.sh
$ ./build_all.sh
$ ./facedetect --cascade="/usr/local/share/OpenCV/haarcascades/
    haarcascade_frontalface_alt.xml" --scale=1.5 lena.jpg
```

After compiling and running the example, you should see the output depicted in Figure 2.

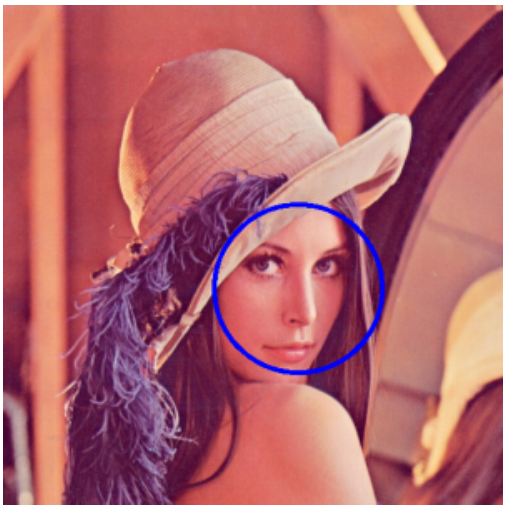

Figure 2: Output of the *facedetect* application of OpenCV 3.2.0 after a successful install.

4 Installing WiSE-MNet++

4.1 Instructions

The simulator source files are distributed as an extension of the Castalia simulation model. The steps required to compile the simulator are equivalent to those for Castalia. We describe two methods using the terminal and the OMNeT++ IDE.

4.2 From terminal

The simulator source files are distributed as an extension of the Castalia simulation model. The steps required to compile the simulator are equivalent to those for Castalia. We describe the method employ the bash shell.

We assume to work in the home directory (type '`cd ~`' to enter it) in a *bash* shell. The following procedure creates an executable file located in `wisemnet-v1.4/out/gcc-release/` or `wisemnet-v1.4/out/gcc-debug/` which can be also accessed via the symbolic link `wisemnet-v1.4` located in the directory `wisemnet-v1.4` .

1. Extract the source files

```
$ wget https://git.io/vdrEd -O wisemnet-v1.4.zip
$ unzip -qq -o wisemnet-v1.4.zip -d .
$ cd wisemnet-v1.4
```

   A folder `wisemnet-v1.4` will be created.

2. Set the environment variables to point to symbolic link `wise-mnet` of the WiSE-MNet++ executable:

```
$ export PATH=$PATH:~/wisemnet-v1.4
```

   This line should be also appended to the `~/.bashrc` file.

3. Create the makefiles to compile Castalia with the WiSE-MNet++ extensions

```
$ ./makemake
```

4. Build [2]

```
$ make
```

   A symbolic link `wise-mnet` will be created in the directory `wisemnet-v1.4` after the successful build.

5. To properly clean the last WiSE-MNet++ build, the following can be used

```
$ ./makeclean
```

6. Testing the installation

```
$ cd ~/wisemnet-v1.4/wise/simulations/wisetest/
$ wisemnet-v1.4 -r 0 -u Cmdenv -c  FOV_directional demo2_network2D.ini
```

   After compiling and running the example, you should see the output depicted in Figure 3.

[2] NOTE: if you have a multi-core machine, compilation will be faster by running the make command with the '-j' option and passing the number of cores plus one as argument. For instance, in a dual-core machine use '`make -j 3`'.

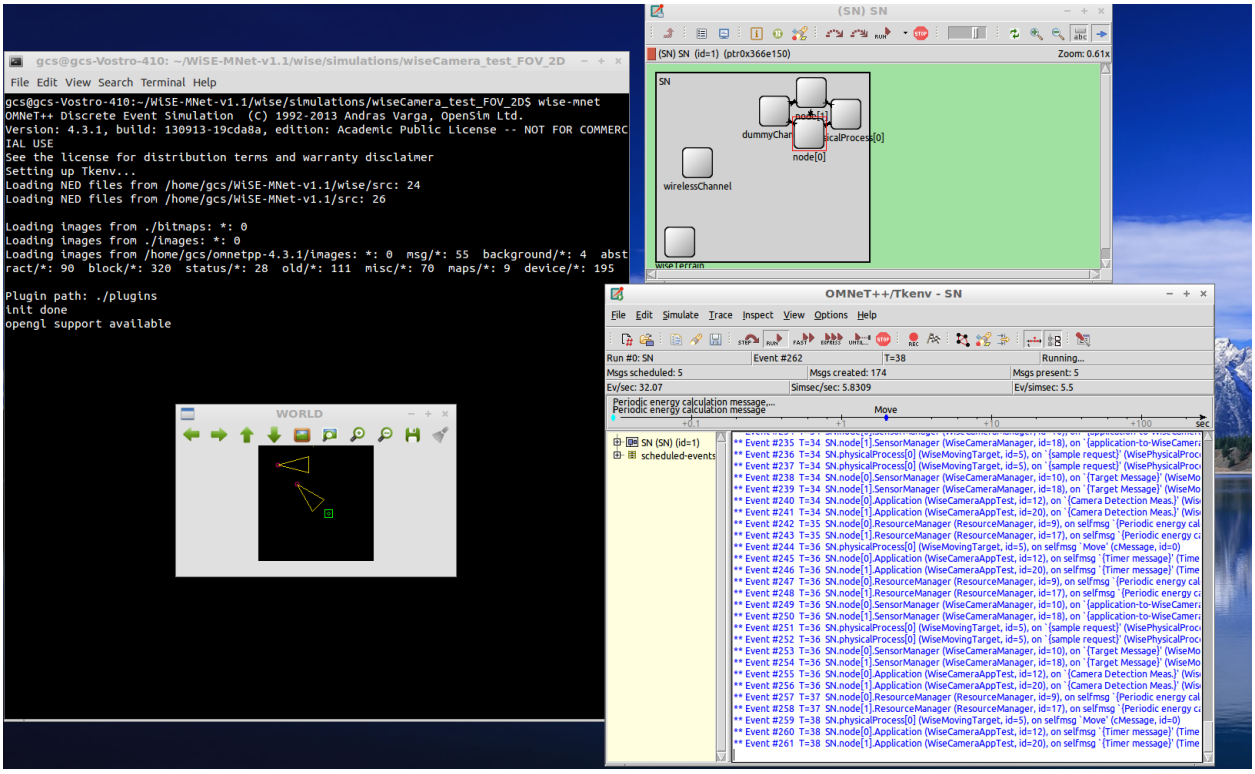

Figure 3: Example of succesffull execution of WiSE-MNet++ .

4.3 From OMNeT++ IDE: import project

The OMNeT++ IDE can be also used to develop applications for WiSE-MNet++ . A quick overview of the IDE is available at `htpp://www.omnetpp.org/doc/omnetpp/IDE-Overview.pdf`.

A first method to setup the IDE for WiSE-MNet++ , we propose to import a project sample which can be distributed here.

1. Extract the source and the project files

```
$ unzip -qq -o wisemnet-v1.4_omnetpp5.0.zip -d .
$ cd wisemnet-v1.4_omnetpp5.0
```

A folder `wisemnet-v1.4_omnetpp5.0` will be created.

2. Start the IDE

```
$ omnetpp
```

If the command does not work or there no symbolic link to the OMNeT++ executable, please go to the directory `~/omnetpp-5.0/ide` and type again the command. As a result, you should see the IDE environment as illustrated in Figure 4.

3. Then import the project downloaded

   (a) Create a new OMNeT++ imported project (*File*→Import...)

   (b) Select the option "Existing Projects into Workspace" as shown in Figure 5.

   (c) Select the folder `wisemnet-v1.4_omnetpp5.0` at the location where the it was extracted in the step 1, as shown in Figure 6.

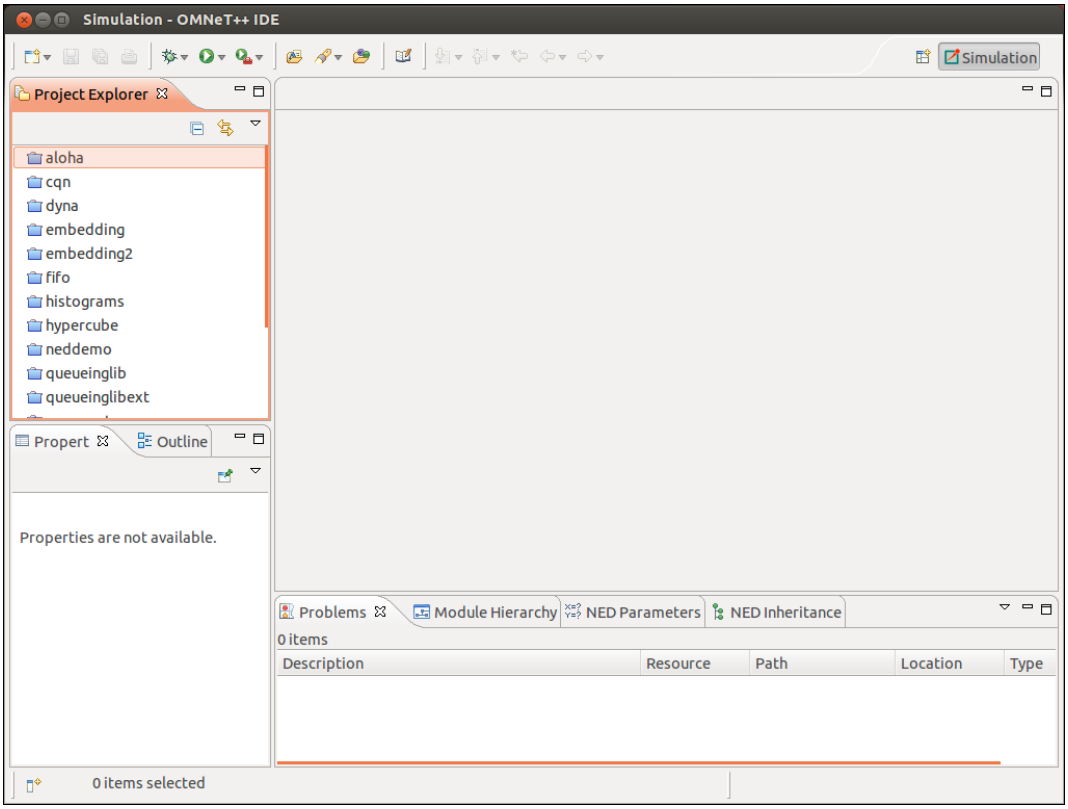

Figure 4: OMNeT++ IDE main window to develop applications based on Omnet++.

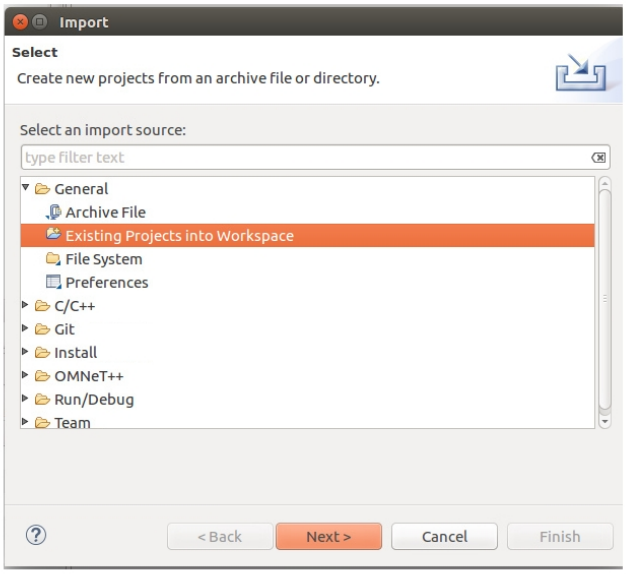

Figure 5: Import OMNeT++ project sample for WiSE-MNet++ (step 1).

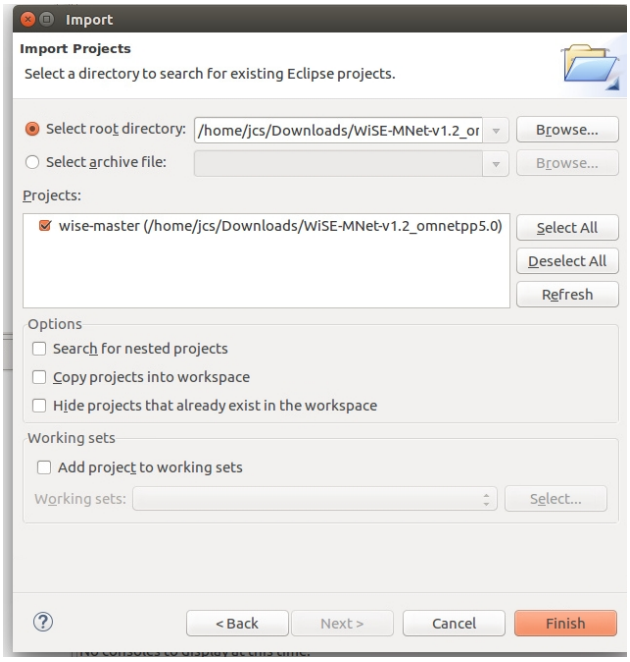

Figure 6: Import OMNeT++ project sample for WiSE-MNet++ (step 2).

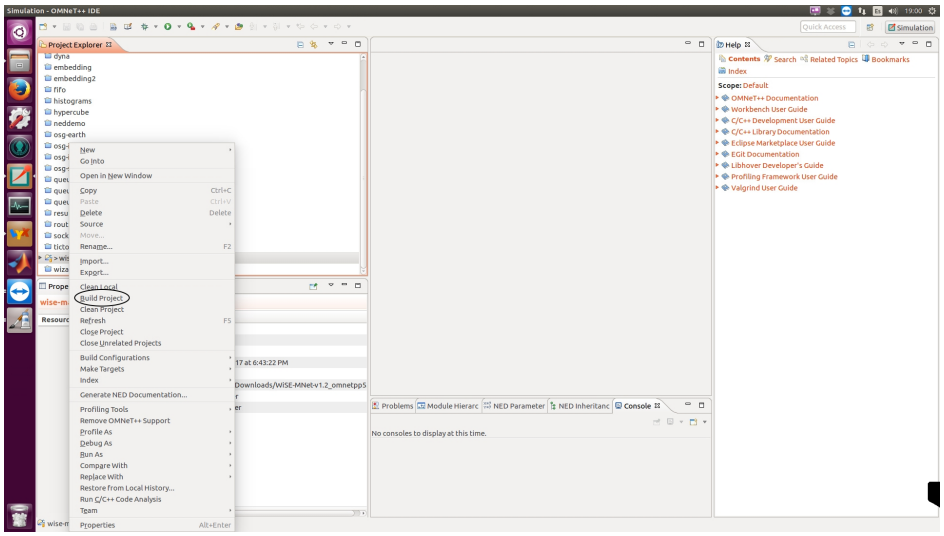

Figure 7: Import OMNeT++ project sample for WiSE-MNet++ (step 3).

(d) Click *Finish* to import the project

4. To check that the project is succesfully imported, perform the following steps:

   (a) Compile the projects as shown in Figure 7.

   (b) The project will start compiling... as shown in Figure 8.

   (c) After a while, the project will have completed the compilation as shown in Figure 9.

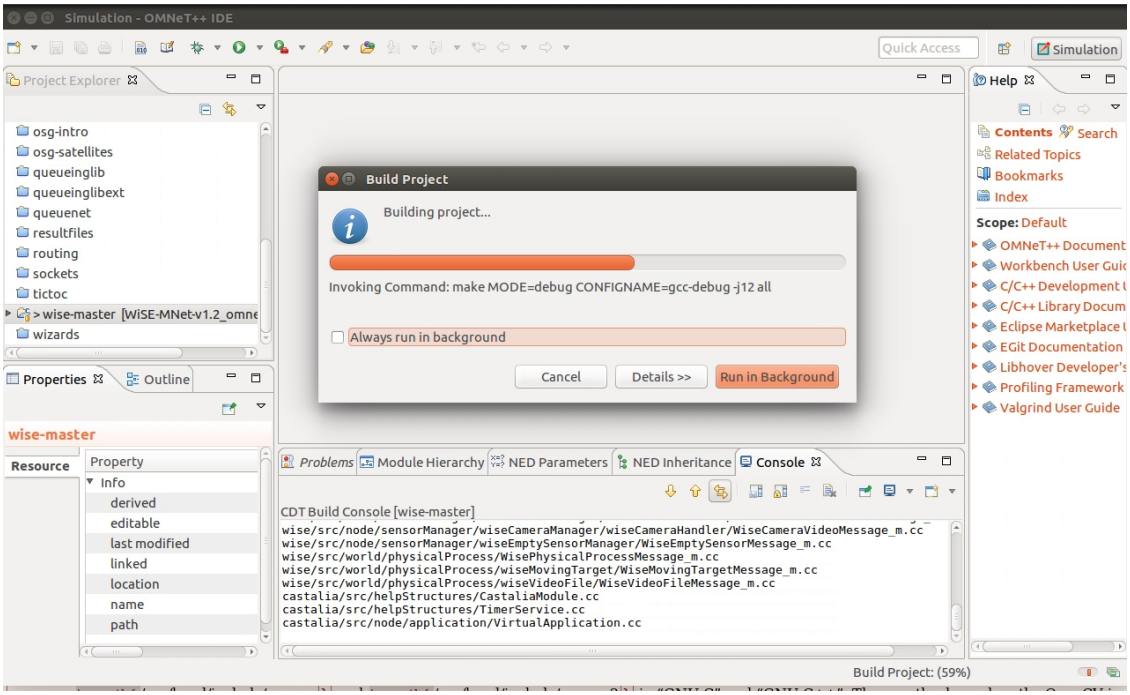

Figure 8: Import OMNeT++ project sample for WiSE-MNet++ (step 4).

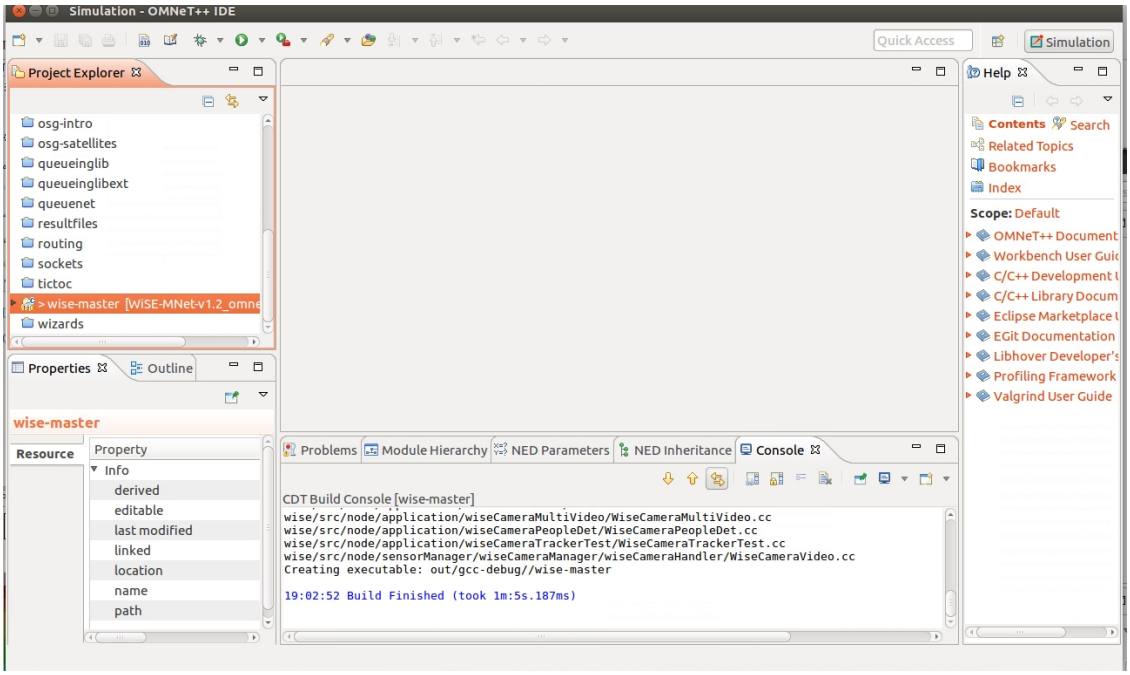

Figure 9: Import OMNeT++ project sample for WiSE-MNet++ (step 5).

(d) Open the file `wisemnet_omnetpp5.0/wise/simulations/wiseTest/demo2_network2D.ini`. You should see the content of this configuration file.

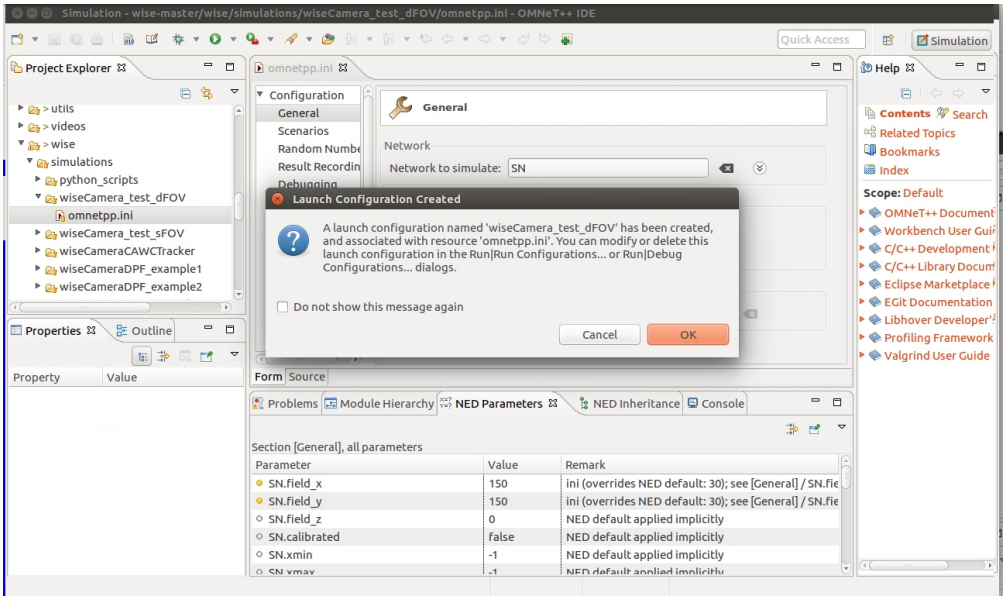

Figure 10: Import OMNeT++ project sample for WiSE-MNet++ (step 5).

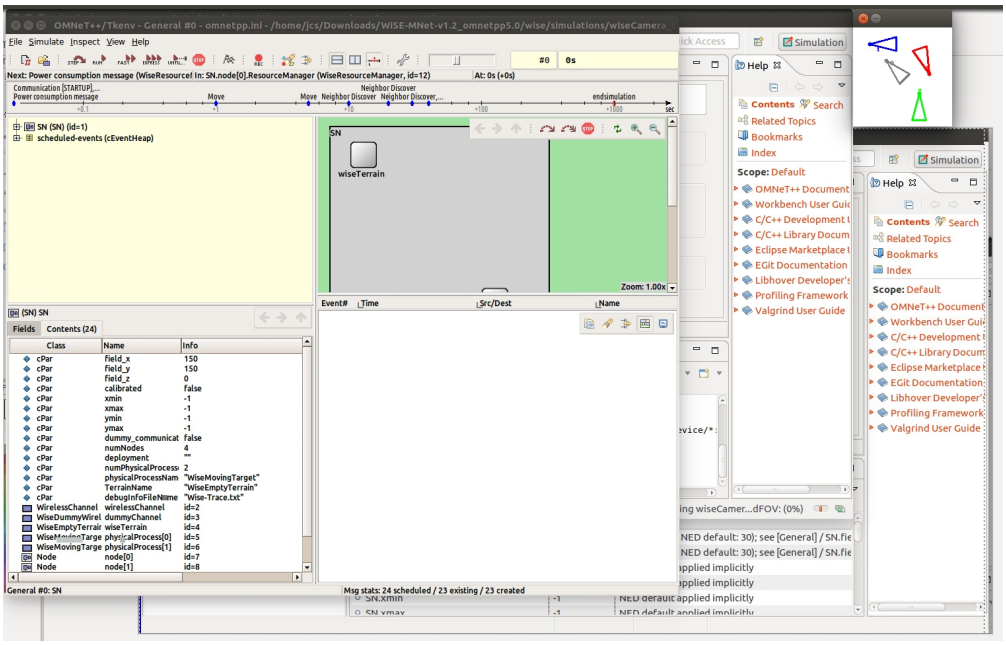

Figure 11: Import OMNeT++ project sample for WiSE-MNet++ (step 5).

(e) Then, generate a new simulation by pressing "Crtl + F11" or via the upper menu (*Run*→Run). A dialog should open as shown in Figure 10.

(f) Press OK and the simulation environment TKenv should appear as shown in Figure 11. At the left you have TKenv whereas at the right you have a visualization of the camera network with the Fields of View of cameras.

(g) Use the controls ("Step", "Run", "Fast",...) to start the simulation and visualize the target moving

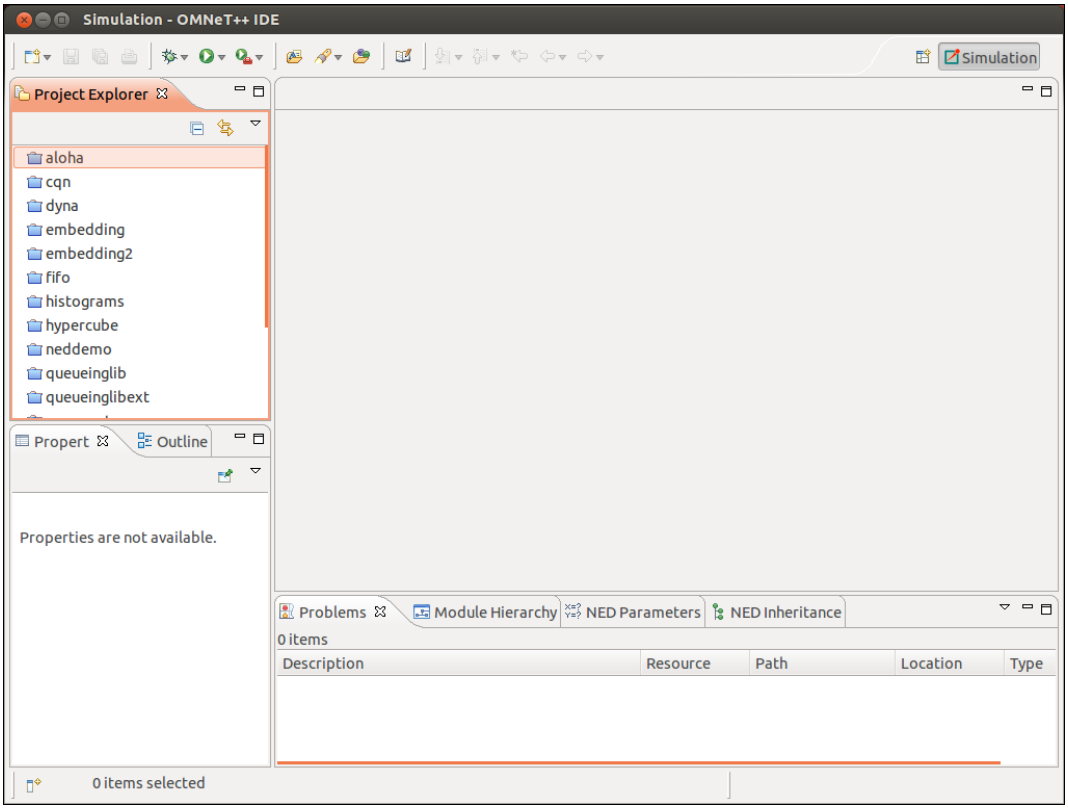

Figure 12: OMNeT++ IDE main window..

in the camera network.

4.4 From OMNeT++ IDE: new project

If the above importing method does not succesfully work, another option is create a new project, include the source files and manually adjust the settings of the new project. The steps are:

1. Extract the source files

```
$ unzip -qq -o wisemnet-v1.4.zip -d .
$ cd wisemnet-v1.4
```

A folder `wisemnet-v1.4` will be created.

2. Start the IDE

```
$ omnetpp
```

If the command does not work or there no symbolic link to the OMNeT++ executable, please go to the directory `~/omnetpp-5.0/ide` and type again the command. As a result, you should see the IDE environment as illustrated in Figure 12.

3. Create and configure the WiSE-MNet++ project. A video tutorial has been created at `https://github.com/MultiCameraNetworks/wisemnet`. Here we summarize the main steps.

   (a) Create a new OMNeT++ empty project (*File→New→OMNeT++ project*)

(b) Import all source files into the newly created project (*"Right Click" on the project name→Import*)

(c) Go to the source NED files manager (*"Right Click" on the project name→Properties→*OMNeT++ →NED Source Folder) and tick two "src" boxes in the paths `wisemnet-v1.4/castalia/src` and `wisemnet-v1.4/wise/src`

(d) Add required include OpenCV files. Go to the "includes" tab in the "Path and Symbols" menu (*"Right Click" on the project name→Properties→*C/C++ General→Path and Symbols). Then, add the paths `/usr/local/include/opencv` and `/usr/local/include/opencv2` in "GNU C" and "GNU C++". These paths depend on the OpenCV installation and can be obtained by running:

```
$ pkg-config opencv --cflags
```

(e) Moreover, add the following paths `/usr/include/eigen3` and `/usr/include/libxml2`

(f) Add required libraries files for OpenCV. Go to *the Makemake* options (*"Right Click" on the project name-→Properties→*OMNeT++ →Makemake). Then, click on `makemake (deep, recurse->)` and the button on the left will be enable. After clicking on this button, go to the link tab and include the Opencv libraries `opencv_core, opencv_features2d opencv_imgproc opencv_highgui opencv_video opencv_legacy` in the "Additional libraries box". In the "Additional objects" box, include the path `wise/src/wise/utils/gmm/c-gmm/c-gmm-64bit.a` The final result can be checked in the "Preview" tab which should similar to the following:

```
--deep -O out -lxml2 -lopencv_tracking -lopencv_videoio -lopencv_objdetect -
    lopencv_imgcodecs -lopencv_imgproc -lopencv_highgui -lopencv_video -
    lopencv_core -lopencv_features2d --meta:recurse --meta:auto-include-path --
    meta:export-library --meta:use-exported-libs --meta:feature-cflags --meta:
    feature-ldflags -- wise/src/wise/utils/gmm/c-gmm/c-gmm-64bit.a
```

(g) Finally, run the desired simulation by creating a configuration of the selected *.ini file (*"Right Click" on the project name-→Run or Debug as→*OMNeT++ *simulation*)

5 Dataset downloading

WiSE-MNet++ provides the following features to support the use of visual data within the simulation environment:

- Multiple video streams captured in multi-camera settings.
- Sensing at different framerates according to the configuration.
- Calibration information to map each camera view to a common ground plane.

Currently, the following datasets can be employed in WiSE-MNet++ :

- Sequence S2_L1 from Crowd_PETS09 dataset (7 views) available at Reading FTP.
- Sequence AB_HARD from AVSS2007 dataset (1 view) available at QMUL FTP.
- Sequence CHAP from ICGLab6 dataset (4 views) available at Datasets Tugraz webpage.

Sample bash scripts are provided to download the video frames and the associated data (groundtruth, calibration,...) in the directory `videos` These scripts prepare all data to be used directly in WiSE-MNet++ . Type the following commands to download:

```
$ cd WiSE-MNet-v1.4/videos
$ ./download_ICG6Lab_chap.sh
$ ./download_PETS09_S2_L1.sh
$ ./download_AVSS07_AB_Hard.sh
```

6 Software organization

The WiSE-MNet++ root directory (according to installation instructions is `~/wisemnet-v1.4/`) contains the original Castalia source files and the extensions provided by WiSE-MNet++ . All WiSE-MNet++ source and simulations files are contained in the `wise/` folder in the root directory. The native Castalia files can still be found in `Castalia/`. In this section, we give an overview of the software organization in folders to help the reader browsing the source code.

The structure of the WiSE-MNet++ root directory is:

`out/`	output generated after compilation of WiSE-MNet++ files
`castalia/`	native Castalia NED/C++ sources, header and simulation files*
`doc/`	documentation of WiSE-MNet++ files
`wise/`	WiSE-MNet++ NED/C++ sources and Simulation setups
`videos/`	datasets, video files and download scripts
`utils/`	Utilties, scripts and other supporting files
`makemake`	Script to configure the WiSE-MNet++ makefiles
`makeclean`	Script to properly clean-up the WiSE-MNet++ build
`...`	others

*Castalia is included in the Wise-Mnet distribution to support wireless and radio communications.

The WiSE-MNet++ simulation setup files are contained in the `wise/Simulations/` sub-folder. The definitions/redefinitions of the OMNeT++ modules (NED/C++ files) can be found in the `wise/src/` sub-folder, which contains the main part of the software.

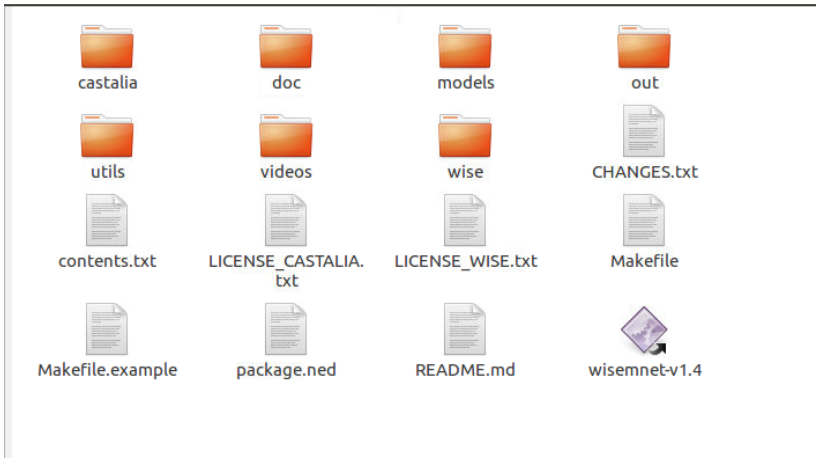

Figure 13: Main directory structure for WiSE-MNet++ .

The structure of the `wise/simulations` subtree is the following:

`wise/simulations/`	Simulation files and display utilities
`/wiseTest/`	Simulations files for testing basic functionality such as 2D networks, OpenCV integration and multi-camera analysis..
`/Tutorial_ICIP17/`	Simulations files presented at the tutorial@ICIP2017.
`/wiseMultiCameraPeriodicApp_MTIC/`	Simulations files for demo and testing the MTIC tracker.
`/wiseMultiCameraPeriodicApp_ICF/`	Simulations files for demo and testing the ICF tracker.
`/wiseMultiCameraPeriodicApp_DPF/`	Simulations files for demo and testing the DPF tracker.
`/wiseMultiCameraPeriodicApp_KCF/`	Simulations files for demo and testing the KCF tracker.

The structure of the `wise/src/` subtree is the following:

`wise/src/`	Wise-Mnet source code files.
`/node/`	Definition of the Wise-Mnet node's components.
`/world/`	Definition of the world's elements (PhysicalProcess, terrain).
`/wirelessChannel/`	WirelessChannel and WiseDummyWirelssChannel.
`/gui/`	simple GUI code.
`/utils/`	Utilities (GMM, ParticleFilter, helper classes).

The structure of the `wise/src/node` subtree is the following:

`node/application/`	Application layer files.
`/wiseBaseApp/`	Baseline interface to develop applications in WiseMnet (network functionality and connection to other layers).
`/wiseBaseAppTest/`	Test class for the interface *wiseBaseApp*.
`/wiseCameraApp/`	Extension of *wiseBaseApp* interface to support visual inputs (i.e. cameras).
`/wiseCameraAppTest/`	Test class for the *wiseCameraApp* interface.
`/wiseCameraPeriodicApp/`	Extension of *wiseCameraApp* interface to support periodic analysis (i.e. sensing, processing and communication).
`/wiseCameraPeriodicAppTest/`	Test class for the *wiseCameraPeriodicApp* interface.
`/wiseCameraPeriodicAppDet/`	Sample class implementing the *wiseCameraPeriodicApp* interface for background subtraction using OpenCV.
`/wiseCameraPeriodicAppTracker/`	Sample class implementing the *wiseCameraPeriodicApp* interface for target tracking using OpenCV.
`/wiseMultiCameraPeriodicApp_DPF/`	Implementation of the Distributed Particle Filter (DPF) using the *wiseCameraPeriodicApp* interface for multi-camera single target tracking.
`/wiseMultiCameraPeriodicApp_ICF/`	Implementation of the Information Consensus Filter (ICF) using the *wiseCameraPeriodicApp* interface for multi-camera single target tracking.
`/wiseMultiCameraPeriodicApp_KCF/`	Implementation of the Kalman Consensus Filter (KCF) using the *wiseCameraPeriodicApp* interface for multi-camera single target tracking.
`/wiseMultiCameraPeriodicApp_MTIC/`	Implementation of the Multi-Target Information Consensus filter (MTIC) using the *wiseCameraPeriodicApp* interface for multi-camera multi-target tracking.
`node/sensorManager/`	Sensor layer files.
`node/sensorManager/sensorManager`	Base interface for Sensor Manager modules.
`/wiseEmptySensorManager`	Dummy sensor producing random numbers.
`/wiseCameraManager`	Camera Manager module supporting two inputs (moving targets and video files) and calibration information.
`node/resourceManager/`	Resource manager for Wise-Mnet++ accouting for sensing, processing and communication.

7 Documentation

Doxygen style documentation has been created in HTML format for WiSE-MNet++ .

1. To generate this documentation, please use the provided script

```
$ sudo ./utils/generate_doc.sh
```

   A folder `doc/html` will be created.

This documentation would be available at `doc/html/index.html`. Figures 7 and 7 depict examples for the application layer *WiseMultiCameraPeriodicApp_MTIC* and the corresponding packet to exchange data. Additionally, a copy of the user manuals for Castalia, OpenCV and OMNeT++ is included in the directory `doc`.

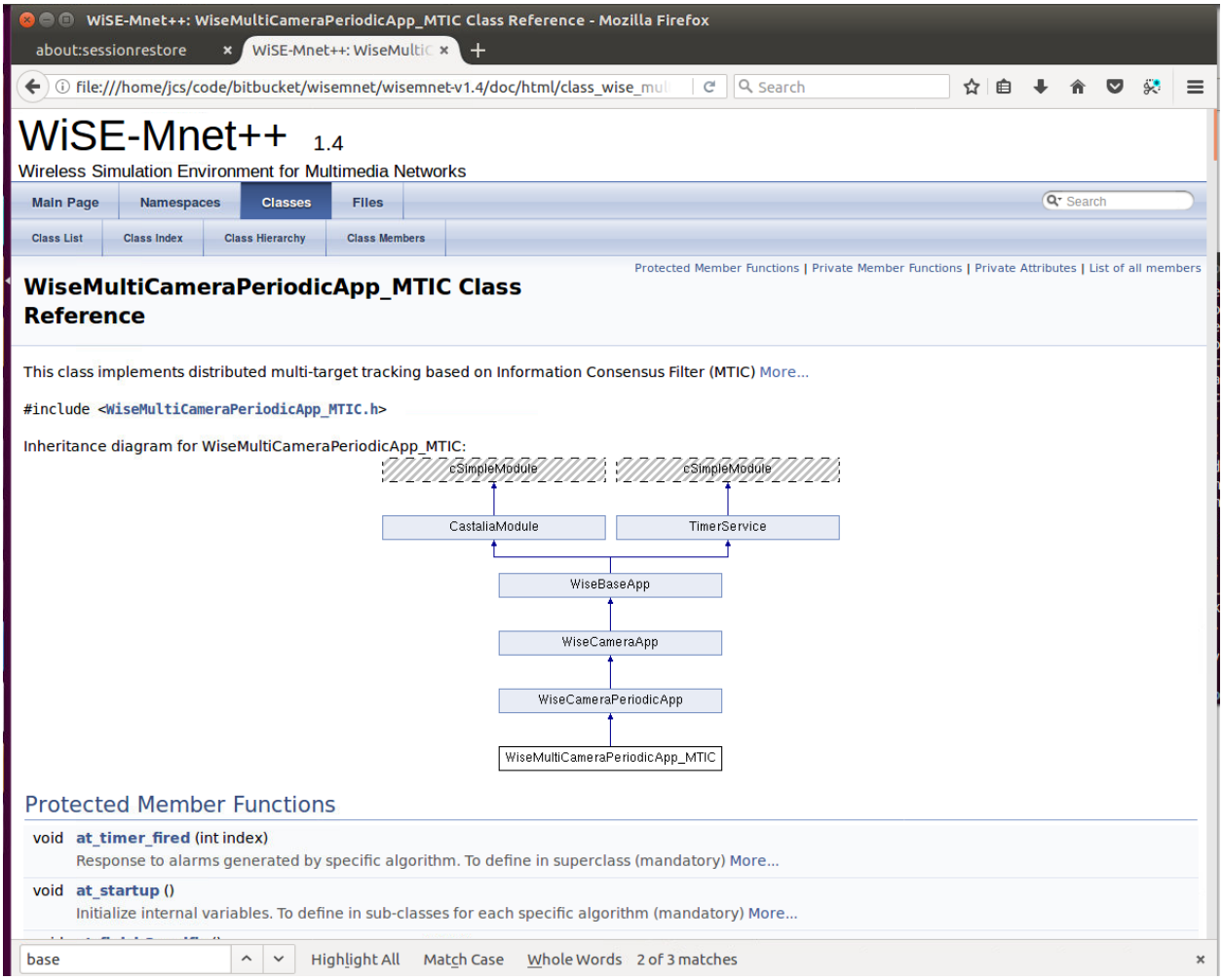

Figure 14: Example of documentation for application layer *WiseMultiCameraPeriodicApp_MTIC* of WiSE-MNet++ .

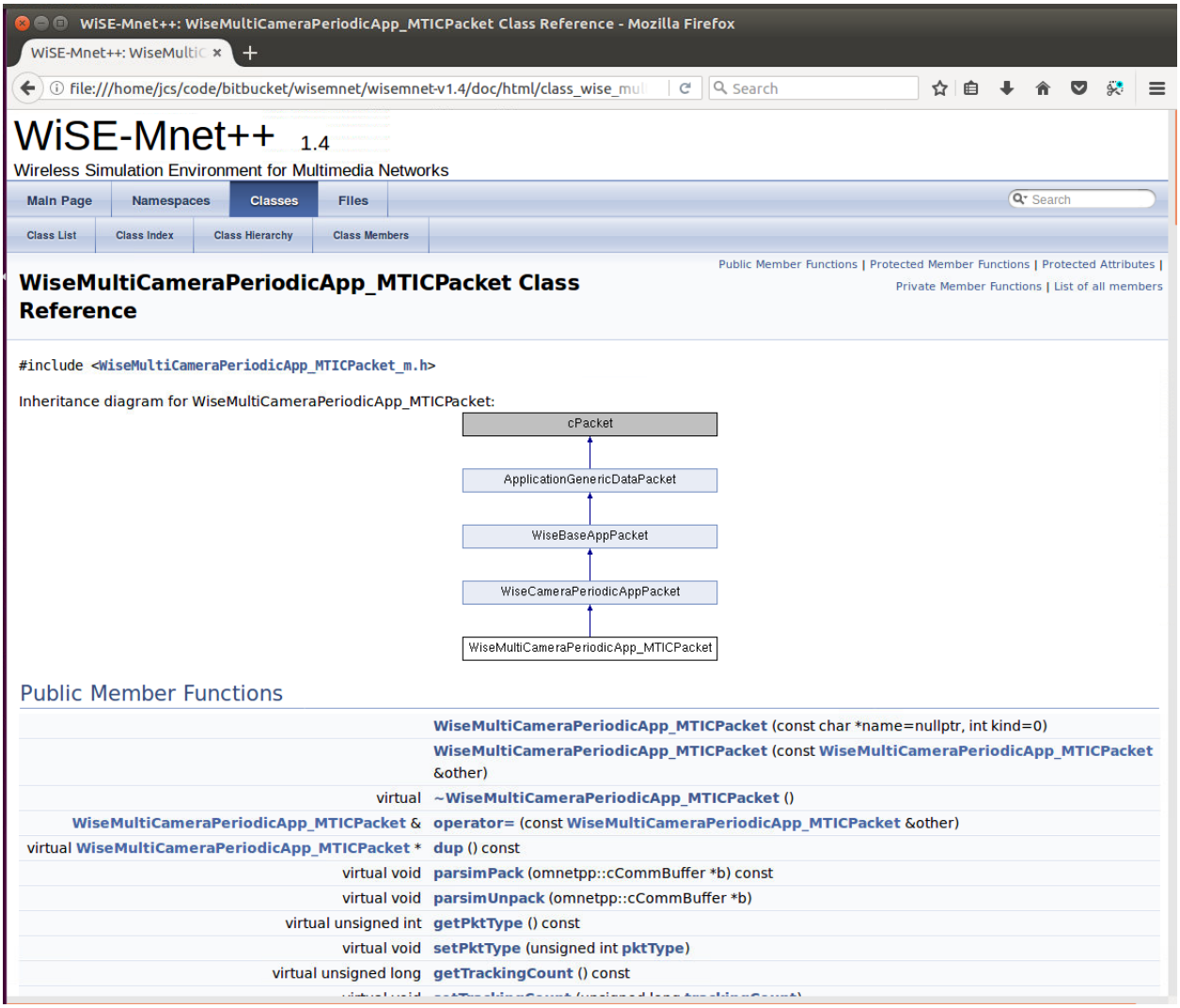

Figure 15: Example of documentation for application layer *WiseMultiCameraPeriodicApp_MTIC* of WiSE-MNet++ .

8 Compatibility issues

WiSE-MNet++ has been successfully tested in various Linux systems: Ubuntu (12.04 LTS, 14.04 LTS & 16.04 LTS), Debian 7.5 and Fedora 20-17. However, a compatibility issue has been detected for Fedora 17 regarding the drivers for the video graphics card and the use of OpenCV within OMNeT++ , reporting the error *X Error: BadWindow (invalid Window parameter)*.

For the lastest Ubuntu version 16.04 LTS, this compatibility issue appears when simultaneously using TKenv (to visualize events and packets exchanged) and OpenCV windows (to visualize simulation results). A temporary solution to this issue is to employ only one method to visualize results for each simulation. The command terminal option *Cmdenv* is suggested as illustrated in Figure 8

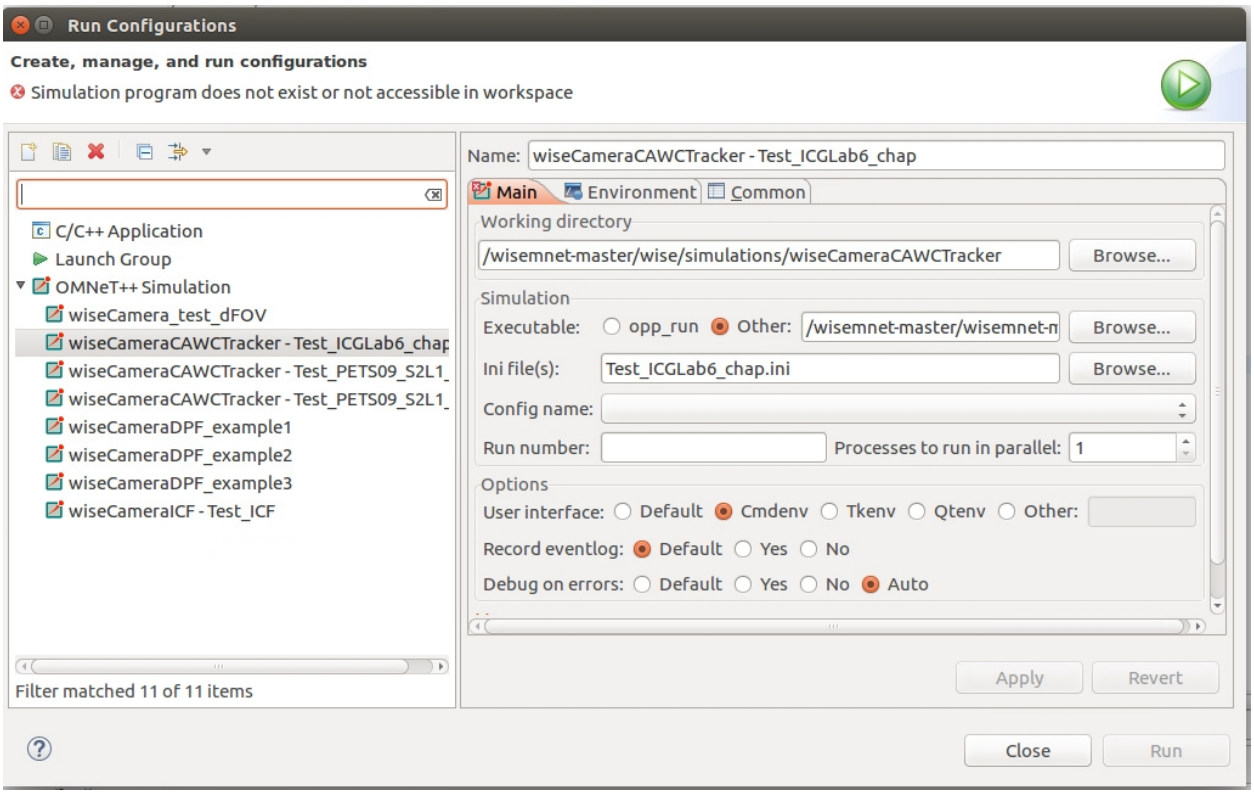

Figure 16: Example to select the option “Cmdenv” to avoid the use of TKenv for simulations in WiSE-MNet++ .