

Project Acronym	CORMORAN
Document Title	D2.5 – First Mobility Enabled Physical Simulator and Coupling with Packet Oriented Simulator
Contractual Date of Delivery to ANR	M29 (30/05/2014)
Actual Date of Delivery	17/06/2014
Editor	UR1
Authors	Benoit Denis, Oudomsack Pierre Pasquero, Raffaele D’Errico (CEA), Nicolas Amiot, Bernard Uguen, Stéphane Avrillon, Meriem Mhedhbi (UR1), Jimenez Guizar Arturo Mauricio, Claire Goursaud, Jean-Marie Gorce (INSA), Jihad Hamie (TPT)
Participants	CEA, UR1, INSA, TPT
Related Task(s)	T2
Related Sub-Task(s)	T2.4
Security	Public
Nature	Technical Report
Version Number	1.0
Total Number of Pages	75

CONTACT AND CORRESPONDENCE

Bernard Uguen (UR1)

- Address: IETR Université de Rennes 1 – campus de Beaulieu, Bât 11C, 263 avenue du général Leclerc 35042 Rennes CEDEX, France
- E-mail : bernard.uguen@univ-rennes1.fr
- Tel : (+33) (0)2 23 23 60 33

Benoît Denis (CEA)

- Address: CEA-Leti Minatec, 17 rue des Martyrs, Bât. 51C, p. C455, 38054 GRENOBLE Cedex 9, France
- E-mail: benoit.denis@cea.fr
- Tel: + 33 (0)4 38 78 09 90

TABLE OF CONTENT

CONTACT AND CORRESPONDENCE	2
TABLE OF CONTENT	3
TABLE OF ACRONYMS	4
ABSTRACT	5
EXECUTIVE SUMMARY	6
1. INTRODUCTION.....	7
2. GENERAL DESCRIPTION OF THE PHYSICAL SIMULATOR.....	7
2.1. Layout Description.....	9
2.2. Agent Mobility.....	11
2.3. Body mobility: Multi-cylinders model C3D animated.....	16
2.4. Graph based description of the propagation channel	28
2.5. IR-UWB Channel Impulse Response Determination Example	32
3. WSNET DESCRIPTION.....	37
3.1. Underlying Methods	37
3.2. Channel model and current limits of the simulations	40
3.3. Interfacing	41
3.4. Files XML.....	42
4. FULL MESH MEASUREMENT BASED SIMULATION	45
4.1. CSV Files in WSNET	45
4.2. Full Mesh Propagation Model.....	48
5. PYLYERS-WSNET INTERFACING	55
5.1. Co-simulation Frameworks for Wireless Sensor Networks	55
5.2. Architecture of the co-simulator framework	56
5.3. WSN / Pylayers PHY layer interfacing	61
5.4. Data design	68
6. CONCLUSIONS.....	71
7. BIBLIOGRAPHY	72

TABLE OF ACRONYMS

ACS	Antenna Coordinate System
AF	Amplify and Forward
B2B	Body to Body
B2I	Body to Infrastructure
BAN	Body Area Network
BER	Bit Error Rate
BLER	Block Error Rate
CCS	Cylinder Coordinate System
CF	Compress and Forward
CIR	Channel Impulse Response
DCS	Device Coordinate System
DF	Decode and Forward
DPSK	Differential Phase Shift Keying
ED	Energy Detector
GCS	Global Coordinate System
IR	Impulse Radio
LDP	Location Dependent Parameter
LOS	Line of Sight
LQI	Link Quality Indicator
MAC	Media Access Control layer
NLOS	Non Line of Sight
NWK	NetWorK layer
OB	On-Body
OOK	On-Off Key
PDF	Probability Density Function
PER	Packet Error Rate
PHY	PHYSical layer
PPM	Pulse-Position Modulation
RT	Ray-Tracing
SINR	Signal to interference plus Noise Ratio
SNR	Signal to Noise Ratio
TOA	Time Of Arrival
TOF	Time Of Flight
UWB	Ultra Wide Band
WBAN	Wireless Body Area Network

ABSTRACT

This document describes the 2 simulation tools PyLayers and WSNET and the current work for interfacing each other for the purpose of producing test vector for CORMORAN scenario simulation.

EXECUTIVE SUMMARY

This document gathers a detailed description of the simulation approach developed in the CORMORAN project. The presented work consists in making necessary evolutions to the two simulators and in interfacing them in the frame of the defined CORMORAN scenario [1].

The first presented simulator is PyLayers [2], a site-specific Ultra Wide Band (UWB) oriented propagation simulator which has been evolved toward a full heterogeneous Body Area Network (BAN) simulator. Its purpose, in the CORMORAN project, is to produce the time-stamped radio observables coming up from the BAN radio multi-channels, in order to feed a physical layer abstraction block and flowing it to the upper MAC layer simulator. Several original PyLayers contributions related to the adaptation to the Wireless BAN (WBAN) context, developed along the project are presented. This mostly concern the human mobility but also the very heart of the ray tracing approach. The whole set of tools whose have reached a quite high level of realism are very well suited for the addressed objectives of the CORMORAN project. In particular, the tool has the ability to address the CORMORAN key Body to Body scenario for large scale navigation.

The second simulator described is WSNET [3], a discrete event simulator for wireless network on a large scale. This simulator is in charge of assessing performances of MAC and NET layers solutions and strategies. This document provides an extensive description of the workflow for doing so.

The third important point which is addressed in this document is the way to make the two simulators work together for producing higher level performance evaluation of the two main defined CORMORAN scenarios. Especially, the interfacing data format has been defined and is described in this document as well or several rules of thumb execution time predictions.

1. INTRODUCTION

Realistic cross layer BAN simulation which is highly required for evaluation of the defined localization CORMORAN scenario raises different challenges for simulation tools themselves and for their interfacing. In a first time, the deterministic approach is required at a very low level. The scene involving humans moving in their environment is described carefully with the proper level of realism imposed by the considered application. The description of site specific scenes including body motion at different level is described as well as the approach for deterministic propagation simulation.

Section 2 describes the processing of the physical simulator. This simulator is used to describe the synthetic environment, run a dynamic simulation of agents to generate trajectories and add the body mobility over those trajectories. Given this mobility description, the simulator allows to perform a deterministic simulation of all the different propagation channels created by the agents and the infrastructure nodes.

Section 3, is focused on the WSNET tool, which is an event-driven simulator for wireless networks. It is especially designed to evaluate performances of a system by simulating a MAC layer based on a dedicated physical layer abstraction. For instance, the modeling of collisions and interference can help to determine byte error rate (BER) and packet error rate (PER) can be obtained for a large number of nodes. The explicit and detailed workflow for producing a WBAN oriented simulation in the WSNET framework is explained. This section has the sufficient detailed granularity for being used as an user manual.

Section 4 specifies how WSNET is going to use external measures or PyLayers simulations to evaluate system performances. In particular, that section focuses on the file formats and configurations files required for such a mode.

In Section 5, PyLayers and WSNET interface is studied in order to produce high quality observables in the WBAN context. The options for bridging the two softwares are studied in terms of computational and storage costs. In particular, the emphasis is put on which simulator ought to be in charge of the PHY abstraction.

2. GENERAL DESCRIPTION OF THE PHYSICAL SIMULATOR

The physical simulator PyLayers [2] is an open source project aiming to address the problem of indoor radio propagation. To this end, the simulator embeds several complementary and independent modules including, in particular:

- An agent mobility simulator based on discrete event simulator to which a multi cylinders Body model description can be associated.
- A deterministic UWB radio simulator,

The synthetic environment, so called layout, is described at different levels using graphs to address the different specificities of the simulation. The agent mobility relies on a mixture of specific graphs description of the layout and on a virtual force model. In addition to that agent mobility which is only focused on the mobility of a point, a multi cylinders body model can be added to address WBAN problematic.

The deterministic simulation of the propagation channel is performed by a pre-processing on graphs, allowing an economy of computation required in the context of mobility. About the considered ray tracing approach, the simulator would be used to determine the propagation channel in two different situations: Body to body (B2B), and body to infrastructure (B2I). Due the phenomena appearing on the body (creeping waves ...) and its complexity deterministic modeling, the on Body (OB) simulation would be treated in two steps. The first step will use a statistical model provided in [4] to determine power losses and body shadowing effect. A second step consists in running the deterministic simulation in order to obtain the environment influence in the propagation channel. Figure 2-1 shows a bloc diagram of the physical simulator including the agent and body mobility simulation, the electro-magnetic simulation and the output of the simulator.

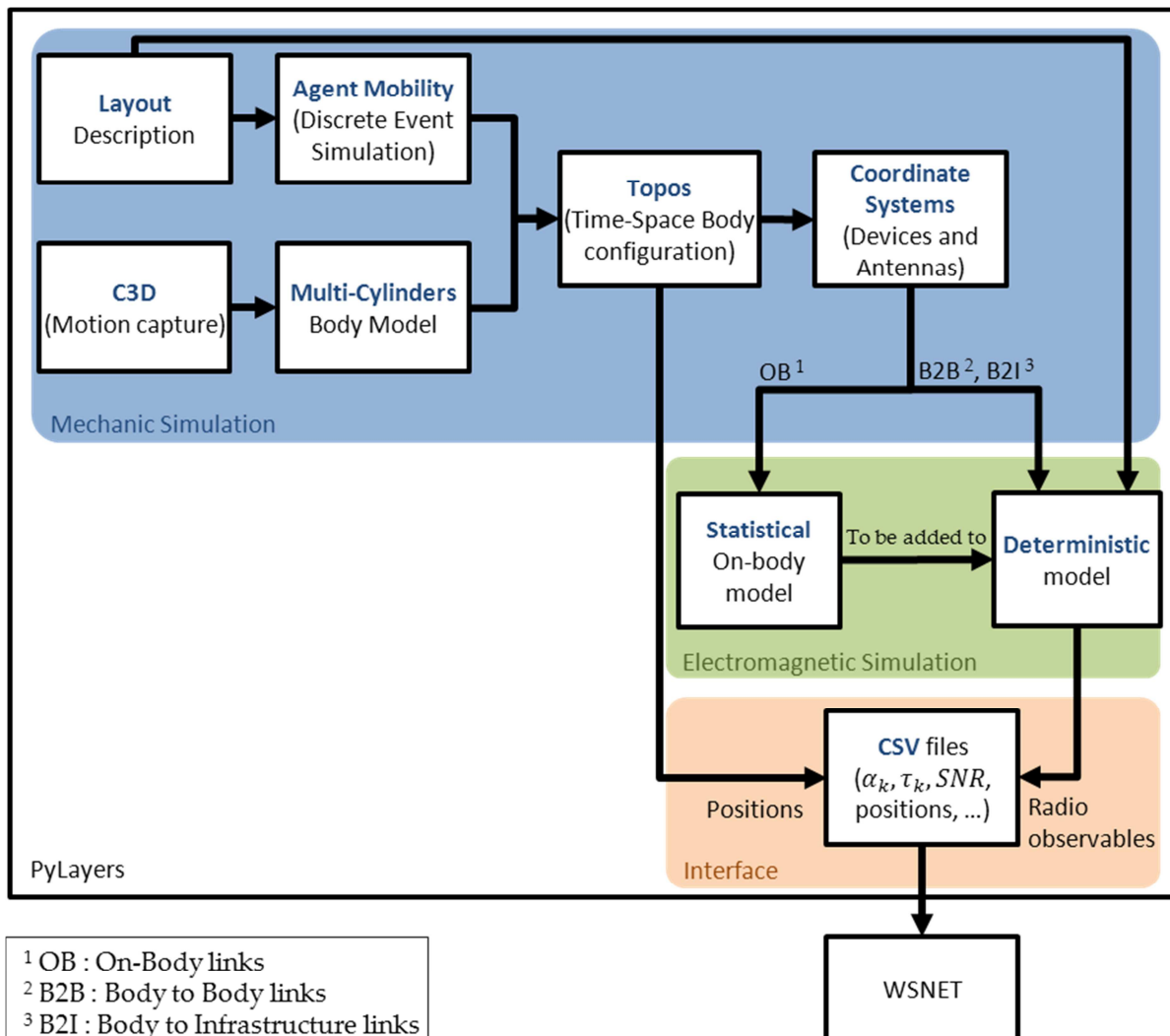


Figure 2-1 PyLayers' physical simulator bloc organization

2.1. LAYOUT DESCRIPTION

The layout is the description of the synthetic environment inside of which both mobility and ray tracing simulations takes place. That approach relying on a graph description is exposed in the following.

From the graph theory, a graph is a connected set of nodes. Graphs can be oriented or non-oriented. In the sequel, we will refer to graph also as a specific high level data structure. In that case, those graph data structures embed specific dictionaries associated with nodes or edges, to store the relevant piece of information related to their nature.

Throughout the following, the introduced graphs will be indexed with a letter x related to their name.

The following notations for graph description are used :

An graph indexed x from its name is noted $G_x(V_x, E_x)$ where V_x and E_x are respectively the set of nodes and the set of edges of the graph G_x .

For a graph $G_x(V_x, E_x)$ having N nodes and E edges,

$$V_x = \{v_x^0, \dots, v_x^{N-1}\} \quad (1)$$

$$E_x = \{e_x^0, \dots, e_x^{E-1}\} \quad (2)$$

where for an oriented graph

$$e_x^k = (\text{tail}(e_x^k), \text{head}(e_x^k)) \in V_x \times V_x \quad (3)$$

In PyLayers, the layout is described with the help of 2 non oriented graphs:

- the graph of structure $G_s(V_s, E_s)$,
- the topological graph $G_t(V_t, E_t)$.

2.1.1 THE GRAPH OF STRUCTURE G_s

The structure graph G_s is the first graph to be defined. This graph is directly obtained from a node list and an edge list. This graph is being modified all along the layout editing phase which is done with a dedicated built-in editor. It can also be generated from proper conversion from other existing format as XML open street map format (.osm) [5]. All the other graphs are derived from processing on its nodes and edges.

Considering the indoor case, a layout is generally a building floor plan. Each segment describing the layout outlines is associated to a node of G_s . As well, the two points bounding each segments are also associated to two nodes of G_s . Conventionally, nodes of G_s associated to segments are denoted with a positive integer, whereas nodes associated to points are denoted with a negative integer. By construction all nodes with a positive index (segments) are linked to two nodes with negative indexes (points). Note that the converse is not true, a point can be connected to several segments.

Figure 2-2 shows an example of the generated graph of structure G_s .

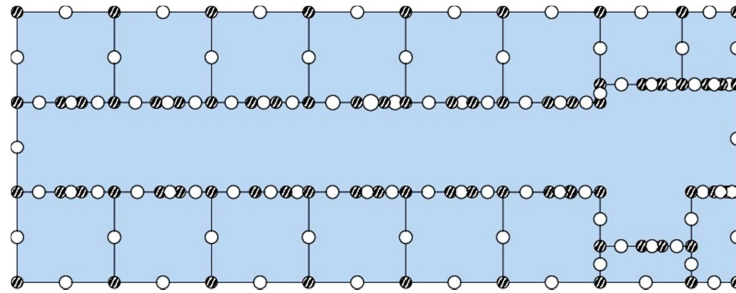


Figure 2-2 example of Graph of structure G_s . White dots represent segments (nodes with positive index) and hatched dots represents points (nodes with a negative index) of the layout.

Positive nodes of G_s have attached a meta-data dictionary containing the information listed below which is useful either for further mechanical or electromagnetic processing:

- a list of the connected points indexes (a.k.a. the nodes of G_s with a negative index),
- the name of the slab associated to the segment,
- a position,
- a height value,
- a normal vector associated to segment,
- a boolean transition indicator.

The list of connected points is directly deduced from the graph connectivity. The slab's name is specified during the layout edition.

A slab is a stack of different constitutive materials of different thickness. This description requires the knowledge of all slab elements with their electromagnetic properties.

The position is given as a vector in \mathbb{R}^3 . The height and elevation values allow to address situations where a segment is made with different slabs at different heights. The list of cycle numbers contains all cycles containing the node. At most, a node can be surrounded by two cycles. This information is filled once the topological graph G_t has been built. More information about G_t and cycles is provided in the following. The signed normal vector is used to determine the orientation of the segment. The transition indicator indicates if the segment can mechanically be crossed (by an agent).

In this connection, transition indicator is very useful for describing air walls. Air walls are virtual partition which affect neither mechanic nor electromagnetic computation. That abstraction has been introduced for reducing complexity during higher-level graph computation.

2.1.2 THE GRAPH OF TOPOLOGY G_t

The topological graph G_t aims to abstract the structural graph G_s to cycles representation.

A graph cycle is a closed path starting and finishing at the same node, where nodes and edges of that path can be only used once.

Cycles are convenient to obtain a high-level structure information of the layout for both increase the processing speed and derive other useful graphs. For the problem at hand, G_t is built from G_s nodes in respect of three rules :

1. all cycles have to be disjoint, which means none cycle must be included into another.
2. all nodes of G_s with a negative index has to be used for building cycles
3. Each of those nodes can only be used for building two cycles at most.
4. Each nodes of G_s with a positive index can link 2 cycles at most.

The first step to build G_t is to obtain a cycle basis from the decomposition of G_s . Because G_s is an undirected graph it can be defined a cycle space, it exists a set of linearly independent cycles named basis cycles, which can represent G_s [6]

As shown in Figure 2-3, nodes of the resulting G_t graph represent cycles. By construction, edges connect adjacent cycles. Two cycles are adjacent if they share at least one segment. Cycles which are only connected through a single point are not considered adjacent and though are not connected

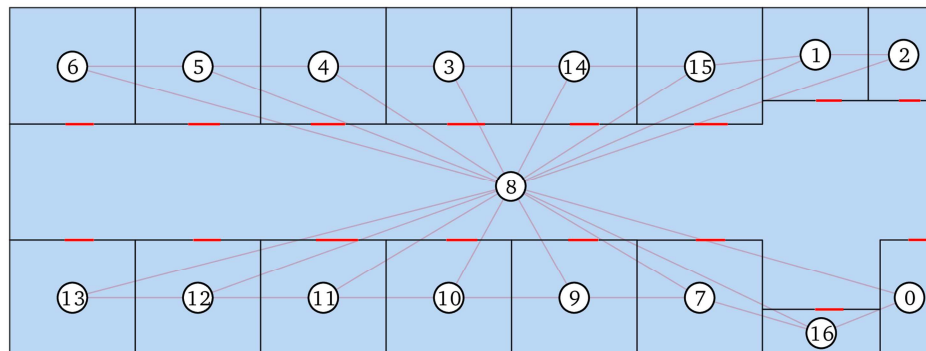


Figure 2-3 An example of topological Graph G_t . Nodes of this graph are cycles. Edges of this graph connect adjacent cycles

Both G_s and G_t are required for a precise description and organization of a layout. They are also mandatory to build higher level graph relative to propagation.

2.2. AGENT MOBILITY

The agent mobility describes the movement of agent inside a Layout. At this stage, only the center of mass of the agent is taken into account. The link between that center of mass and the body mobility based on a multi cylinders body model will be detailed in 2.3.

The proposed simulation of the agent mobility is based on a discrete event simulator. The movement of the agents description is divided into two modes. First the large scale mobility, which represents the *motivation* of the agent (choosing a destination room) is described in

2.2.1. Second, the small scale mobility, which constraints the center of mass of the agent to move and interact with the environment and the other agents is described in 2.2.2.

2.2.1 LARGE SCALE MOBILITY : A GRAPH AIDED DESCRIPTION

At large scale, the agents mobility can be seen as a stochastic succession of two different states :

- a state (SS) where the agent is static
- a state (MS) where the agent is mobile

Most of the time, in indoor environment, mobile agents are static, thus the static state can be fully considered as part of the overall mobility description. The large scale mobility consists in both defining a target for the agent and determining a path to reach that target.

The Graph of Room G_r and the Graph of Pathway G_w

In order to determine the different targets of the agents, the graph of room G_r is introduced.

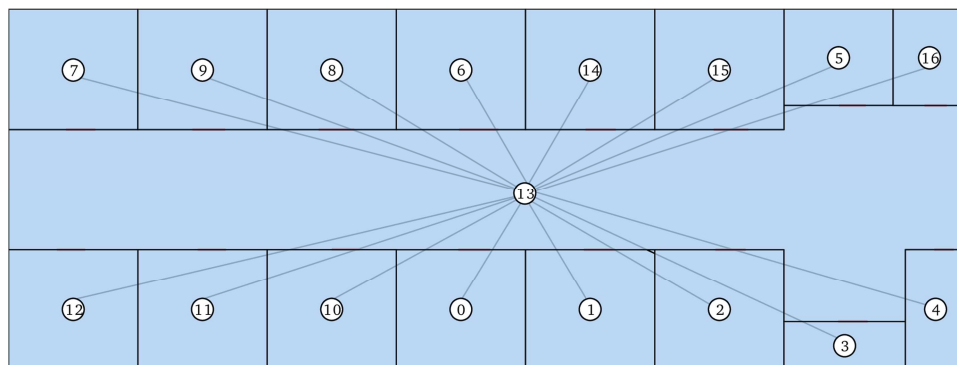


Figure 2-4 An example of Graph of rooms G_r .

The graph of room G_r , represented in Figure 2-4, is a simplified version topological graph G_t described in , especially design for the mobility purpose. The nodes of the G_r represent rooms. For this purpose, the nodes of G_r are the nodes of G_t limited to cycles (G_t 's nodes) with a door. In addition, each edge of G_r interconnects rooms sharing a door. With such a description, each node can determine a target by choosing a room in the nodes of G_r . Both the target and the path are determined with the help of graphs. For a more realistic movement, the graph of pathway G_w representing all the possible transitions between rooms is introduced.

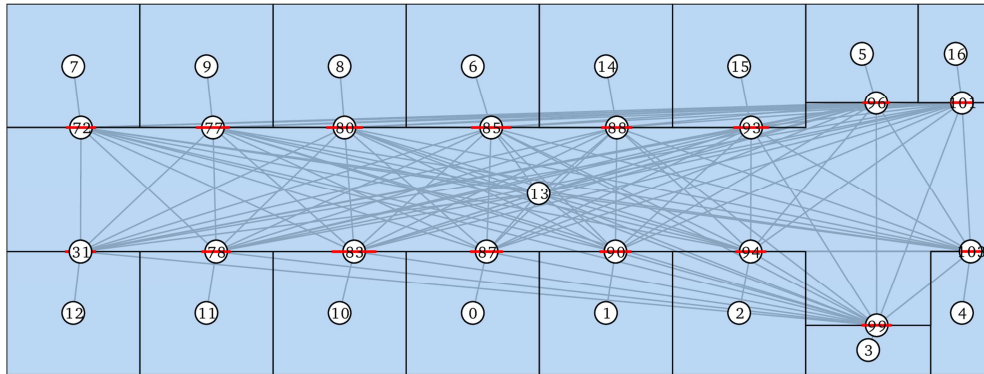


Figure 2-5 An example of Graph of pathway G_w , associated to the graph of room.

The graph G_w , represented in Figure 2-5, has the same "room" nodes than G_r , plus extra nodes associated to each doors. Those nodes are added to encourage the agent both to cross the doors and to avoid unreal trajectory e.g. going to the center of a corridor instead of directly reaching the opposite room. Nodes are connected using two rules :

- All room nodes are connected to a door node
- The doors of two rooms are connected together if the 2 rooms are connected in G_r .

Determination of a Target Using the Graphs.

Practically, the graph G_r allows the agent to randomly choose a target (a.k.a a destination room), and the graph G_w allows to find a path to this target. That path is a succession of G_w nodes.

Lets suppose a situation where an agent in room i chooses a *targeted* room t with the help of graph G_r . Rooms i and t correspond respectively to nodes v_r^i and v_r^t of G_r . Then, the path from the room i to the room t is obtained by processing a Djikstra [7] shortest path algorithm D on nodes v_w^i and v_w^t of the graph G_w :

$$V_p = D(v_w^i, v_w^t). \quad (4)$$

The obtained ordered set of nodes V_p describes the shortest path between v_w^i and v_w^t . To achieve the whole trajectory from i to t , the agent has to successively go through all the rooms and/or doors corresponding to the nodes of V_p . Each time a node of V_p is reached, the next one becomes an intermediate target it , until the node v_r^t is reached.

In Figure 2-5, an agent in room $i = 12$ with a target in room $t = 15$, corresponds to returned path $V_p = \{12,31,93,15\}$. Thus, the agent first moves from room 12 to its intermediate target 31 and so on until it moves to its final target in room 7.

2.2.2 SMALL SCALE MOBILITY : STEERING FORCES

Once their targets are known and their paths have been determined, agents can start moving in the environment, avoiding walls and going through doors. This particular ability is defined as the small scale description of the mobility and is described with the help of steering behaviors.

Steering behaviors have been introduced in [8] to describe interactions of agents with their environment. Agents are simply described by their mass m and two limiting parameters: their maximum acceleration a_{max} and their maximum velocity v_{max} . This description allow the agents to be driven by several steering forces applied on their center of mass. The different steering forces are described in sections the following.

The calculation of acceleration, velocity and position of agents is ensured by Euler integration [9]. At each time step the agent's acceleration \mathbf{a} is re-evaluated in regards with the resulting steering force applied on it. The true steering forces value is truncated in regard of the maximum acceleration parameter of the agent a_{max} . The steering force \mathbf{F} actually applied on the agent is defined by:

$$\mathbf{F} := \max(\|\mathbf{F}\|, ma_{max})\hat{\mathbf{F}} \quad (5)$$

where $\hat{\mathbf{F}}$ is the unitary vector obtained from normalization of \mathbf{F} . Then, the applied acceleration vector \mathbf{a}_n at time step n is given by :

$$\mathbf{a}_n = \frac{\mathbf{F}}{m} \quad (6)$$

As well, the true velocity \mathbf{v}_n at time step n is approximated by the Euler integration as the sum of the old velocity and the product of the current acceleration vector with $\delta\tau$, the time between $n - 1$ and n :

$$\mathbf{v}_n = \mathbf{v}_{n-1} + \mathbf{a}_n\delta\tau \quad (7)$$

Then, the applied velocity \mathbf{v}_n is truncated by the maximum velocity parameter of the agent v_{max} :

$$\mathbf{v}_n := \max(\|\mathbf{v}_n\|, v_{max})\hat{\mathbf{v}}_n \quad (8)$$

where $\hat{\mathbf{v}}_n$ is the normalization of \mathbf{v}_n .

The position \mathbf{p}_n at time step n is obtained by the Euler integration as the sum of the previous position \mathbf{p}_{n-1} and the current applied velocity :

$$\mathbf{p}_n = \mathbf{p}_{n-1} + \mathbf{v}_n\delta\tau \quad (9)$$

The steering forces applied on the agent correspond to human like behaviors. In the following, only three types of steering forces are considered : the seek behavior, the obstacle avoidance behavior and the agent avoidance behavior.

Seek Behavior

The seek behaviors aims to produce a steering force which attract the agent to a target. In the considered scenario, those targets are succession of intermediate targets as defined in section 2.3.1. The steering force direction will depends on the distance vector \mathbf{d}_n between the target \mathbf{t}_n and the actual position of the agent \mathbf{p}_n . Then, it is possible to determine a velocity vector \mathbf{v}_n^d defined with :

$$\mathbf{d}_n = \mathbf{t}_n - \mathbf{p}_n \quad (10)$$

$$\mathbf{v}_n^d = \max\left(\frac{\|\mathbf{d}_n\|}{\delta\tau}, v_{max}\right) \hat{\mathbf{d}}_n \quad (11)$$

where \mathbf{v}_n and \mathbf{v}_n^d are the current velocity vector and the desired velocity vector directed respectively; and $\hat{\mathbf{d}}_n$ is the normalization of \mathbf{d}_n

Thus, the velocity difference \mathbf{w}_n between the actual velocity \mathbf{v}_n and the desire velocity \mathbf{v}_n^d , allows to compute the seeking force \mathbf{F}_s to be applied on the agent.

$$\mathbf{w}_n = \mathbf{v}_n^d - \mathbf{v}_n \quad (12)$$

$$\mathbf{F}_s = m \frac{\mathbf{w}_n}{\delta\tau} \hat{\mathbf{w}}_n \quad (13)$$

where $\hat{\mathbf{w}}_n$ is \mathbf{w}_n normalized, and m the mass of the agent. This situation is represented in Figure 2-6.

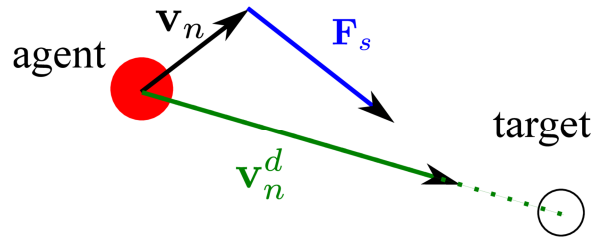


Figure 2-6 Seek behavior: The steering \mathbf{F}_s vector results from the current velocity vector of the agent \mathbf{v}_n and the desired velocity vector directed to the target.

Obstacle Avoidance Behavior

The obstacle avoidance behavior aims to produce a steering force avoiding the agent penetrating a wall or any part of the layout. The associated steering force \mathbf{F}_o is given by:

$$\mathbf{F}_o(\mathbf{d}_{k,w}) = \frac{m}{\delta\tau} \left((\mathbf{v}_{n-1} - \mathbf{v}_n) \frac{\alpha}{|\mathbf{d}_{k,w}|} \hat{\mathbf{n}} \right), \quad (14)$$

with $\mathbf{d}_{k,w}$ the distance vector from the agent k to a wall w , α a parameter properly chosen for avoiding the agent grazing the wall and $\hat{\mathbf{n}}$ a normalized vector orthogonal to the obstacle .

Agent avoidance

In [10], it is stated that it exists an inter-persons distance boundary. This boundary can be modeled with a circle with a radius r around the agent k [11] [12]. While another agent l penetrates inside that circle, an avoidance force F_a proportional to the distance vector between the agents $\mathbf{d}_{k,l}$ is applied. That force can be modeled using the previously defined obstacle avoidance force, regarding the 2 modifications:

1. The distance between the agent and the obstacle is replaced by the distances between the two agents,
2. A rotation matrix has to be introduced in order that the forced be applied orthogonally to the velocity vector, and thus creating an avoidance of both agents.

Thus, the agent avoidance force F_a can be written:

$$\mathbf{F}_a(\mathbf{d}_{k,l}) = \begin{cases} \mathbf{H}\mathbf{F}_o(\mathbf{d}_{k,l}) & , \|\mathbf{d}_{k,l}\| \leq r \\ \vec{0} & , \|\mathbf{d}_{k,l}\| > r \end{cases} \quad (15)$$

with \mathbf{H} a rotation matrix:

$$\mathbf{H} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad (16)$$

Resulting Force

An agent k is under the influence of different steering forces : a seek force which pull the agent to its intermediate target node v_{it} as seen in 2.3.1 and several repulsive forces to avoid obstacles or other agents. The resulting steering force \mathbf{F} is written as the sum of all the steering forces applied on agent k :

$$\mathbf{F} = \mathbf{F}_s + \sum_w^W \mathbf{F}_o(\mathbf{d}_{k,w}) + \sum_{l \neq k}^K \mathbf{F}_a(\mathbf{d}_{k,l}), \quad (17)$$

where W is the number of walls in the vicinity of the agent k , which is determined with a simple distance threshold. The use of that threshold avoids the useless computation of forces from all the walls of the entire layout.

2.3. BODY MOBILITY: MULTI-CYLINDERS MODEL C3D ANIMATED

The agent trajectory is obtained from the multi agent simulator which can produce an arbitrary number of realistic human trajectories into the Layout. Those trajectories can be seen as center of mass of a body. At each timestamp of a trajectory, each body requires to be inserted at the good location in time and space. This is a mandatory in order to determine the position and orientation of the devices attached on the body. For that purpose three steps

are required. First, the motion capture file and a multi cylinders model body representation have to be linked. Second, the multi cylinders model and the agent mobility must be synchronized. Third, the correct positions and orientations of the radio devices attached on the body have to be obtained at a specific time and space location.

2.3.1 MULTI CYLINDERS MODEL FROM THE MOTION CAPTURE FILE

This section presents the data structures and files used to describe the body into the PyLayers simulation.

Loading a Motion Capture File

The body mobility is imported from motion capture files. This is the chosen manner to achieve a high degree of realism for the modeling of the human motion and be able to address complex movements. Among the different existing motion capture data file formats, the C3D (Coordinates 3D) format is used here due to its compliance with most of motion capture systems. The data stored in C3D files are the whole 3D coordinates of a fixed number of points, representing human body sensors at very specific conventional location on the body at each time stamp. A "3D frame" consists of a set of those 3D points.

A C3D motion capture file is loaded with the method `loadC3D` with as arguments the motion capture file and the number of frames to load. The motion is represented as a sequence of frames stored in the `d` variable member. It is possible to get the information from the C3D header by using the verbose option of the `read_c3d` function

```
# Video Frame Rate
Vrate = 120
# Inter Frame
Tframe = 1./120
# select a number of frame
nframes = 300
# Time duration of the whole selected frame sequence
Tfseq = Tframe*nframes
#
# load a .c3dmotion capture file
# this update the g.pos
#
#bc.loadC3D(filename='07_01.c3d',nframes=nframes,centered=True)
```

It has to be emphasis that an upcoming foreseen CORMORAN measurement campaign is going to produce the human motion capture in this very format, which appears to have been a posteriori a quite happily chosen file format solution.

Body Cylinder data structure

To ease electromagnetic simulation a simplification of the motion capture data structure is necessary. Generally there are a large number of captured points, not all of them being useful for our modeling. The body model is a restriction of the key body segments which are transformed into K cylinders of radius r_k . As proposed in [13], from the 41 nodes available in the C3D file, only 16 are selected and used to create 11 cylinders of the body model:

- 4 cylinders describing the two arms,
- 4 cylinders describing the two legs,
- 2 cylinders describing the trunk,
- 1 cylinder for the head.

The body cylinder model is handled by a dedicated Python class call `Body`. To create a void body, simply instantiate a `Body` object from the class

```
>>> John = Body('john.ini')
```

This will create a `Body` instantiation using the `john.ini` configuration file.

Description of a body file

Each body is described using an `.ini` file which contains information about the mapping between the C3D file nodes and the restricted chosen nodes, the size of the linked cylinders, information about the radio devices equipping the body and the motion capture file to be used. This information is gathered in 4 different sections:

- `nodes` : This section associates a node number to a c3d file conventional node number `NodeId = C3DNODE`
- `Cylinder`: This section associates a cylinder `Id` to a dictionary which contains cylinder tail head and radius information
`CylId = {'t', NodeId1, 'h', NodeId2, 'r', float (m), 'name', }`
- `device` : This section associates a device name to a dictionary which contains cylinder device related information

```
DevId =
{'typ' : {static|mobile}
'cyl' : CylId
'l': length coordinate in ccs,
'h': height coordinate in ccs,
'a': angle coordinate in ccs,
'file': antenna file ,
'T' : Rotation matrix }
```

- `mocap`: This section informs about the chosen motion capture file associated to the body.

An example of such an `ini` file is given in the following:

Body configuration file of agent John: John.ini

```
[nodes]
0 = STRN
1 = CLAV
2 = RFHD
3 =RSHO
4 =LSHO
5 =RELB
6 =LELB
7 =RWRB
8 =LWRB
9 =RFWT
10 =LFWT
11 =RKNE
12 =LKNE
13 =RANK
14 =LANK
15 =BOTT

[cylinder]
trunku = {'t':0,'h':1,'r':0.18,'i':0} ; sternum (STRN) - clavicule (CLAV)
trunkb = {'t':15,'h':0,'r':0.17,'i':10} ; bottom (BOTT) sternum (STRN)
headu = {'t':1,'h':2,'r':0.12,'i':1} ; clavícula (CLAV) - head (RFHD)
armr = {'t':5,'h':3,'r':0.05,'i':2} ; right elbow (RELB) right shoulder (RSHO)
arml = {'t':6,'h':4,'r':0.05,'i':3} ; left elbow (LELB) left shoulder (LSHO)
forearmr = {'t':7,'h':5,'r':0.05,'i':4} ; right wrist (RWRB) right elbow (RELB)
forearml = {'t':8,'h':6,'r':0.05,'i':5} ; left wrist (LWRB) left elbow (LELB)
thighr = {'t':11,'h':9,'r':0.05,'i':6} ; right knee (RKNE) right hip (RFWT)
thighl = {'t':12,'h':10,'r':0.05,'i':7} ; left knee (LKNE) left hip (LFWT)
calfr = {'t':13,'h':11,'r':0.05,'i':8} ; right ankle (RANK) right knee (RKNE)
calfl = {'t':14,'h':12,'r':0.05,'i':9} ; left ankle (LANK) left knee (LKNE)

[device]
0 = {'typ':'static', 'cyl':'trunku', 'name':'cardio', 'l':0.0, 'h':0.02, 'a':0, 'file':'defant.vsh3',
'T':np.array([[0,0,1],[1,0,0],[0,1,0]])}
1 = {'typ':'dynamic','cyl':'forearml','name':'Galaxy Gear', 'l':0.0, 'h':.01, 'a':180, 'file':'defant.vsh3',
'T':np.array([[1,0,0],[0,1,0],[0,0,1]])}
2 = {'typ':'static', 'name':'BeSpoon Phone', 'cyl':'thighr', 'l':0.9, 'h':0.01, 'a':0, 'file':'S2R2.sh3',
'T':np.array([[1,0,0],[0,1,0],[0,0,1]])}

[mocap]
walk = '07_01_c3d'
```

Once loaded into an object, the information contained in that *.ini* configuration file can be also obtained by the representation function of the object:

```
>>> John
My name is : John
```

```
I have a Galaxy Gear device on the left forearm
I have a cardio device on the upper part of trunk
I have a BeSpoon Phone device on the right thigh
I am nowhere yet
```

```
filename : 07_01.c3d
nframes : 300
Centered : True
Mocap Speed : 1.36 m/s
```

Once instantiate, all the space-time configuration of the body nodes obtained from the C3D file are contained into a Multi-dimensional array of shape (3, npoint, nframe).

```
>>> John.d.shape
(3, 16, 300)
```

Where

- 3 : dimension of space
- 16 : number of nodes
- 300 : number of frames

In general it is supposed to be a cyclic motion as an integer number of walking steps. This allows to instantiate the body configuration anywhere else in space in the C3D trajectory. In the following, a specific space-time configuration of the body nodes is called a **topos**.

2.3.2 LINKING THE BODY NODES AND THE AGENT TRAJECTORY

The choice which has been made to address the global body mobility into a layout is to link the agent mobility from 2.2 and the body mobility. To this end, two steps are required on the body object. First, the motion obtained from the C3D has to be centered to express node positions relatively to the center of mass of the body. Second, those node positions have to be updated relatively to the agent position and velocity.

Centering the body nodes

The node positions contained into the C3D file are represented into a coordinate system originate at the beginning of the movement. It means that the node positions are given relatively to an origin set at the beginning of the C3D file trajectory. In order to be plugged

with the agent trajectory, the node positions have to be known relatively to the center of mass of the body.

Let \mathbf{n}_f^l denote respectively the position of node l at frame f . For each frame, it is possible to determine the center of gravity \mathbf{p}_f^s with:

$$\mathbf{p}_f^s = \frac{1}{L} \sum_l^L \mathbf{n}_f^l \quad (18)$$

Where L is the number of nodes.

Assuming the trajectory is in the $(0, \hat{\mathbf{x}}, \hat{\mathbf{y}})$ plan, it is possible to determine $\bar{\mathbf{n}}_f^l$ the position of node k centered relatively to the center of gravity of the body with:

$$\bar{\mathbf{n}}_f^l = \mathbf{n}_f^l - \underline{\mathbf{p}}_f^s \quad (19)$$

Where $\underline{\mathbf{p}}_f^s = \mathbf{p}_f^s \times [\mathbf{1}, \mathbf{1}, \mathbf{0}]^T$, in order to remain the elevation of the nodes. In the following, it is assumed that the under bar notation means that elevation component of the vector is set to zero.

At that point, the node positions at each frame are centered above the origin. For example for a walk motion the effect of the centering is just like if the body was still walking but not moving forward exactly in the same manner as a walk on a conveyor belt.

Evaluate topos

Once the node positions have been centered, they can be applied on the agent trajectory. However, that insertion requires both adjusting the velocity from the motion capture file to fit with the velocity of the agent trajectory and to bring the body coordinate system into the coordinate system of the agent trajectory.

First step is to evaluate the topos (and associated node positions) at a specific time t_k of the agent trajectory. Assuming the body motion is a periodic frame sequence with duration T_f , it is possible to get the frame index k_f corresponding to time t_k with:

$$k_f = \left\lfloor \frac{t_k \bmod(T_f)}{t_f} \right\rfloor, \quad (20)$$

where $t_f = \frac{T_f}{N_f}$ is the interframe time or frame sampling period, it is equal to the whole duration of the motion sequence divided by the total number of frames N_f .

Let $\hat{\mathbf{v}}_{k_f}^s$ denote velocity unitary vector along motion capture frame k_f defined as:

$$\hat{\mathbf{v}}_{k_f}^s = \frac{\mathbf{p}_{k_f}^s - \mathbf{p}_{k_f-1}^s}{\left| \mathbf{p}_{k_f}^s - \mathbf{p}_{k_f-1}^s \right|}, \quad (21)$$

and $\hat{\mathbf{v}}_{k_t}^t$ the velocity unitary vector along trajectory at time k_t :

$$\hat{\mathbf{v}}_{k_t}^t = \frac{\mathbf{p}_{t_k}^t - \mathbf{p}_{t_{k-1}}^t}{|\mathbf{p}_{t_k}^t - \mathbf{p}_{t_{k-1}}^t|} \quad (22)$$

Where $\mathbf{p}_{t_k}^t$ is the position of the agent at time t_k . Using the same previous notation where under bar means that elevation component of the vector is bring to zero, it is possible to build from the velocities 2 bases ($\underline{\mathbf{p}}_{k_f}^s, \underline{\hat{\mathbf{v}}}_{k_f}^s, \underline{\hat{\mathbf{v}}}_{k_f}^{s\perp}$) and ($\underline{\mathbf{p}}_{t_k}^t, \underline{\hat{\mathbf{v}}}_{k_t}^t, \underline{\hat{\mathbf{v}}}_{k_t}^{t\perp}$) in the motion capture and agent trajectory planes respectively. The two vectors $\underline{\hat{\mathbf{v}}}_{k_f}^{s\perp}$ and $\underline{\hat{\mathbf{v}}}_{k_t}^{t\perp}$ respectively orthogonal to $\underline{\hat{\mathbf{v}}}_{k_f}^s$ and $\underline{\hat{\mathbf{v}}}_{k_t}^t$ can be built with:

$$\underline{\hat{\mathbf{v}}}_{k_f}^{s\perp} = \mathbf{H} \underline{\hat{\mathbf{v}}}_{k_f}^s, \quad (23)$$

and
$$\underline{\hat{\mathbf{v}}}_{k_t}^{t\perp} = \mathbf{H} \underline{\hat{\mathbf{v}}}_{k_t}^t, \quad (24)$$

where the matrix \mathbf{H} is used to create an orthogonal vector can be is defined by:

$$\mathbf{H} = \begin{bmatrix} \mathbf{0} & -\mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \quad (25)$$

Note the two bases ($\underline{\mathbf{p}}_{k_f}^s, \underline{\hat{\mathbf{v}}}_{k_f}^s, \underline{\hat{\mathbf{v}}}_{k_f}^{s\perp}$) and ($\underline{\mathbf{p}}_{t_k}^t, \underline{\hat{\mathbf{v}}}_{k_t}^t, \underline{\hat{\mathbf{v}}}_{k_t}^{t\perp}$) will describe respectively the so called Body Local Coordinates System (BLCS) and the Global Coordinate System (GCS).

Link topos and agent trajectory

Once both coordinate systems from the motion capture file and from the agent trajectory have been evaluated, it is possible to link them. For that purpose, it is required that the motion capture coordinates system match the agent mobility coordinate system at a given time t_k . In other word, it consists to determine the correct affine transformation from ($\underline{\mathbf{p}}_{k_f}^s, \underline{\hat{\mathbf{v}}}_{k_f}^s, \underline{\hat{\mathbf{v}}}_{k_f}^{s\perp}$) to ($\underline{\mathbf{p}}_{t_k}^t, \underline{\hat{\mathbf{v}}}_{k_t}^t, \underline{\hat{\mathbf{v}}}_{k_t}^{t\perp}$). This transformation can be written:

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{B} \quad (26)$$

With:

$$\mathbf{X} = [\underline{\mathbf{p}}_{k_f}^s, \underline{\mathbf{p}}_{k_f}^s + \underline{\hat{\mathbf{v}}}_{k_f}^s, \underline{\mathbf{p}}_{k_f}^s + \underline{\hat{\mathbf{v}}}_{k_f}^{s\perp}] \quad (27)$$

and
$$\mathbf{Y} = [0, \underline{\hat{\mathbf{v}}}_{k_t}^t, \underline{\hat{\mathbf{v}}}_{k_t}^{t\perp}] \quad (28)$$

Both parameters \mathbf{A} and \mathbf{B} can be obtained with:

$$\mathbf{B} = [\underline{\mathbf{p}}_{t_k}^t, \underline{\mathbf{p}}_{t_k}^t, \underline{\mathbf{p}}_{t_k}^t] \quad (29)$$

and
$$\mathbf{A} = (\mathbf{Y} - \mathbf{B}) \cdot \mathbf{X}^{-1} \quad (30)$$

All the nodes positions $\mathbf{n}_{k_t}^l$ of a trajectory time k_t can be obtained with:

$$\mathbf{n}_{k_t}^l = \mathbf{A} \cdot \mathbf{n}_{k_f}^l + \mathbf{B} \quad (31)$$

That procedure is implemented into the `settopos` method, which takes into argument a trajectory and a time index.

```
>>> John.settopos(traj,t=5)
```

Once the topos has been set, the representation of the object evolves to :

```
>>> John
```

My name is : John

I have a Galaxy Gear device on the left forearm

I have a cardio device on the upper part of trunk

I have a BeSpoon Phone device on the right thigh

My centroid position is

```
[ 0.97911919  5.91836735]
```

filename : 07_01.c3d

nframes : 300

Centered : True

Mocap Speed : 1.36 m/s

Figure 2-7 represents the 16 node positions for 5 different topos linked to an agent trajectory. The 5 different topos have been sample with an interval of 1 second. The lines linking the nodes are just displayed for helping the visualization.

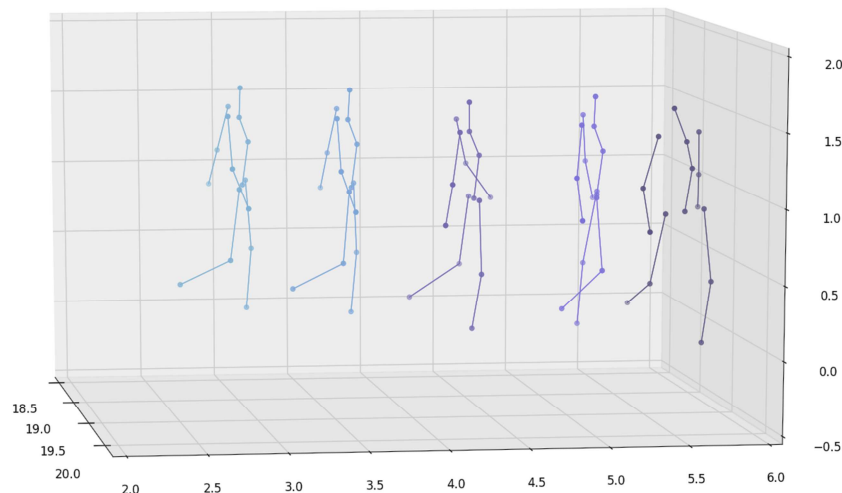


Figure 2-7 Node positions for 5 topos along a agent trajectory

At this point, the nodes positions can be retrieved for any point of the agent trajectory. The remaining work is to link those points to the multi-cylinders body model and to the devices placed on those cylinders.

2.3.3 OBTAINING RADIO DEVICES POSITIONS AND ORIENTATIONS

Due to the further ray tracing processing requiring to properly weighting the rays with the appropriate complex gain from the antenna pattern, it is mandatory to obtain the correct orientation of the radio devices on the body. At this point, body nodes can be expressed on the agent trajectory. The next steps consist in using those points to determine the body cylinders and radio device positions and orientations. To this end, the following chain of coordinate systems is introduced:

$$GCS \rightarrow CCS(t) \rightarrow DCS(t) \rightarrow ACS(t)$$

Where:

- GSS refers to the global coordinates system. The layout, the upcoming rays from the ray tracing and the body nodes (once the topos has been set using the method seen in 2.3.2) are all expressed in this GCS.
- CCS refers to Cylinder Coordinates System: each cylinder of the body model is associated with a dedicated CCS. A CCS is obtained from the large scale trajectory and the body frame corresponding to a given time on the body trajectory.
- DCS refers to Device Coordinates System: this CS is a rotated and/or translated version of the CCS. The origin of this coordinate system is the device position. A device is non-ambiguously placed on a body cylinder via 3 parameters ($l, h, cylId$). $0 < l < 1$ is a parameterization of the length of the cylinder, h is the height of the device w.r.t to the cylinder of Id $CylId$.
- ACS refers to Antenna Coordinate System: it is an arbitrarily rotated version of the DCS that defines the antenna local coordinates system. The radiation pattern of the antenna is known in this ACS. One device could have theoretically several different ACS. This possible feature is not exploited in the following, thus ACS and DCS will be considered to be the same.

Building the Cylinder Coordinate System (CCS)

Cylinders of the body Cylinder model are delimited by nodes of the body. As mentioned in 2.3.1 and shown the body configuration file, bounds of each cylinders are delimited by 2 known nodes. Assuming that $\mathbf{p}_{k_t}^{A,c}$ and $\mathbf{p}_{k_t}^{B,c}$ are the positions of node **A** and **B**, the boundaries of the cylinder c , at time t_k it is possible to define an unary vector $\hat{\mathbf{w}}_{k_t}^c$ for each cylinder with:

$$\hat{\mathbf{w}}_{k_t}^c = \frac{\mathbf{p}_{k_t}^{A,c} - \mathbf{p}_{k_t}^{B,c}}{\left| \mathbf{p}_{k_t}^{A,c} - \mathbf{p}_{k_t}^{B,c} \right|} \quad (32)$$

In addition, knowing $\hat{\mathbf{v}}_{k_t}^t$ the unit velocity vector along actual trajectory, it is possible to build 2 extra unitary vectors:

$$\hat{\mathbf{u}}_{k_t}^c = \frac{\hat{\mathbf{v}}_{k_t}^t - (\hat{\mathbf{v}}_{k_t}^t \cdot \hat{\mathbf{w}}_{k_t}^c) \hat{\mathbf{w}}_{k_t}^c}{|\hat{\mathbf{v}}_{k_t}^t - (\hat{\mathbf{v}}_{k_t}^t \cdot \hat{\mathbf{w}}_{k_t}^c) \hat{\mathbf{w}}_{k_t}^c|} \quad (33)$$

$$\hat{\mathbf{v}}_{k_t}^c = \hat{\mathbf{w}}_{k_t}^c \times \hat{\mathbf{u}}_{k_t}^c \quad (34)$$

The CCS matrix for a given cylinder n can be written:

$$\mathbf{T}_{k_t}^c = [\hat{\mathbf{u}}_{k_t}^c, \hat{\mathbf{v}}_{k_t}^c, \hat{\mathbf{w}}_{k_t}^c] \quad (35)$$

Practically, the `setccs()` method is used to associate a CSS to each cylinder of the body cylinder model. Figure 2-8 displays the CCS associated to each body cylinder.

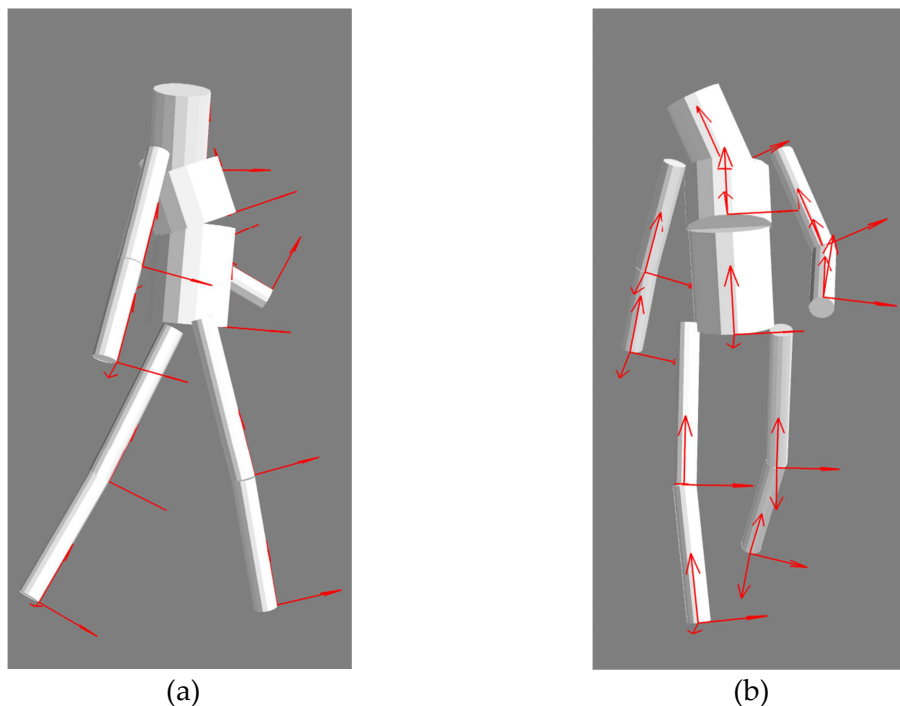


Figure 2-8 (a) Left side and (b) front of the 11 cylinders body model and the associated CCS represented with red arrows.

PLACING A DEVICE COORDINATE SYSTEM (DCS) ON THE CYLINDER

The device coordinate system is related to a given cylinder (and consequently to an ACS). As it can be observed in the configuration file presented in 2.3.1, a DCS referred by 4 numbers:

- c : The cylinder number, in order to determine on which body cylinder the radio device is placed,
- l_d : The distance from the boundary at position $\mathbf{p}_{k_t}^{A,c}$ of a cylinder c at time t_k ,
- h_d : The height above cylinder c generatrix,
- α_d : The angle from front direction in degrees.

Assuming that the angle α defines a rotation along the $\hat{\mathbf{z}}$ axis of the CCS, the associated rotation matrix $\tilde{\mathbf{R}}^d$ for a device d on cylinder c can be written:

$$\tilde{\mathbf{R}}^d = [\tilde{\mathbf{R}}^{d,x}, \tilde{\mathbf{R}}^{d,y}, \tilde{\mathbf{R}}^{d,z}] \quad (36)$$

With

$$\tilde{\mathbf{R}}^{d,x} = [\cos(\alpha) \quad -\sin(\alpha) \quad \mathbf{0}]^T, \quad (37)$$

$$\tilde{\mathbf{R}}^{d,y} = [\sin(\alpha) \quad \cos(\alpha) \quad \mathbf{0}]^T, \quad (38)$$

and

$$\tilde{\mathbf{R}}^{d,z} = [\mathbf{0} \quad \mathbf{0} \quad \mathbf{1}]^T \quad (39)$$

In GCS, this rotation matrix can be written:

$$\tilde{\mathbf{R}}_{t_k}^d = \tilde{\mathbf{R}}^d \cdot \tilde{\mathbf{T}}_{t_k}^c \quad (40)$$

Where $\tilde{\mathbf{T}}_{t_k}^c$ is the CCS matrix of cylinder c on which the device d stands at time t_k . The origin position $\mathbf{p}_{k_t}^d$ of that coordinate system at time k_t can be written:

$$\mathbf{p}_{k_t}^d = \mathbf{p}_{k_t}^{A,c} + (l_d * L_c)\tilde{\mathbf{R}}^{d,z} + (h_d + R_c)\tilde{\mathbf{R}}^{d,y} \quad (41)$$

Where L_c and R_c are respectively the length and the radius of the cylinder c on which the device d stands. Finally the DCS $\mathbf{T}_{k_t}^d$ of a device d expressed in the GCS can be written:

$$\mathbf{T}_{k_t}^d = [\mathbf{p}_{k_t}^d, \mathbf{R}_d] \quad (42)$$

Figure 2-9, shows the 3 DCS associated to the devices equipping John's Body (a watch device, a cardio device and phone device localized on its left forearm, on its upper part of trunk and on its right thigh respectively).

Those radio devices naturally embed antennas with their own coordinate system (ACS).

As mentioned before, at this stage of development, it is assumed that the ACS and DCS are the same. Hence it is possible to directly plug antennas on the body has shown in Figure 2-10.

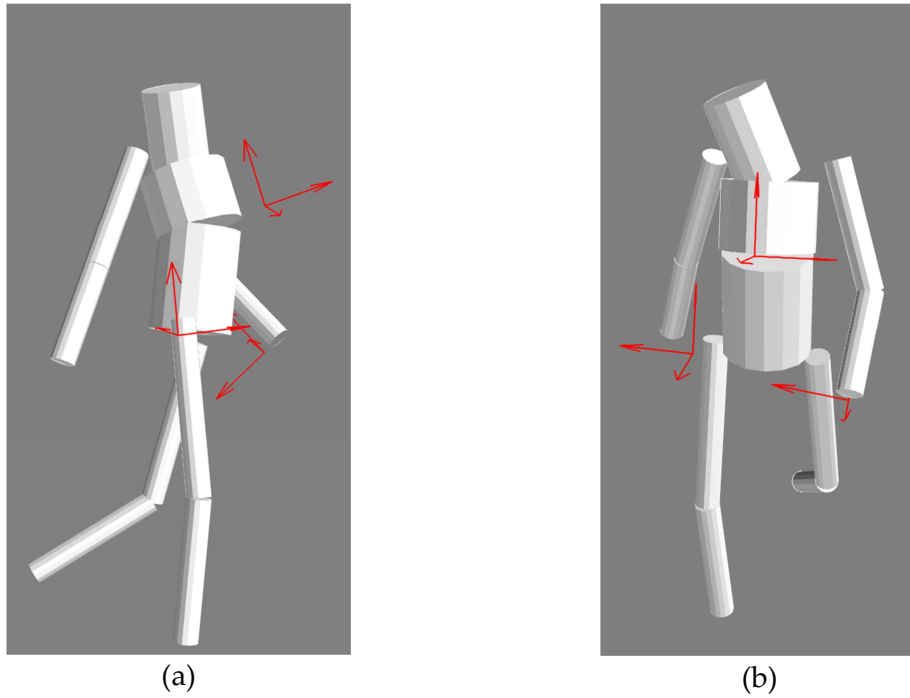


Figure 2-9 (a) Left side and (b) front of the multi-cylinders body model and the 3 DCS associated to the devices equipping the body, represented with red arrows.

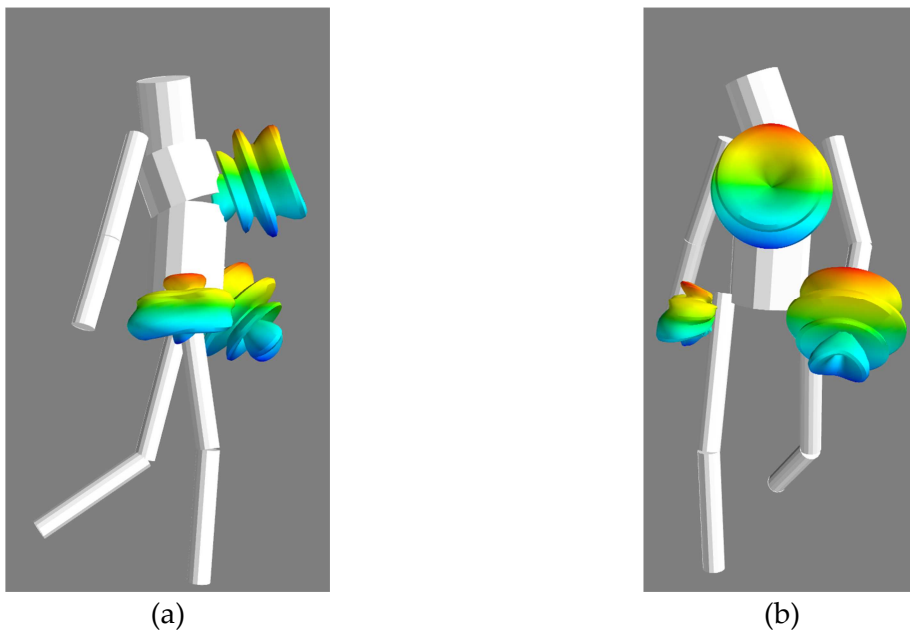


Figure 2-10 (a) Left side and (b) front of the multi-cylinders body model and antenna pattern associated to the 3 devices equipping the body.

2.4. GRAPH BASED DESCRIPTION OF THE PROPAGATION CHANNEL

Similarly to the agent mobility simulation, the propagation channel simulation takes advantage of a graph description. In addition to structural and topological information, propagation-specific graph are introduced to perform a geometric optic and uniform theory of diffraction based ray-tracing.

2.4.1 GRAPHS FOR SIMULATION ELECTROMAGNETIC SIMULATION

Graph of Visibility G_v

The graph of visibility G_v is used to describe the optical visibility between points and segments of a layout. The nodes of G_v are the same as those of G_s . The edges of G_v connect nodes related through an “optical” visibility relationship. Those visibility relations might concern two segments, a point and a segment or two points. Depending on the geometry, some points are candidate for becoming diffraction points. A simple rule for determining a diffracting point exploits the sign and the degree of the node.

The graph of visibility for each cycle of the layout is obtained independently and composed in a loop to obtained the overall graph. An example of the resulting graph processed is shown in Figure 2-11.

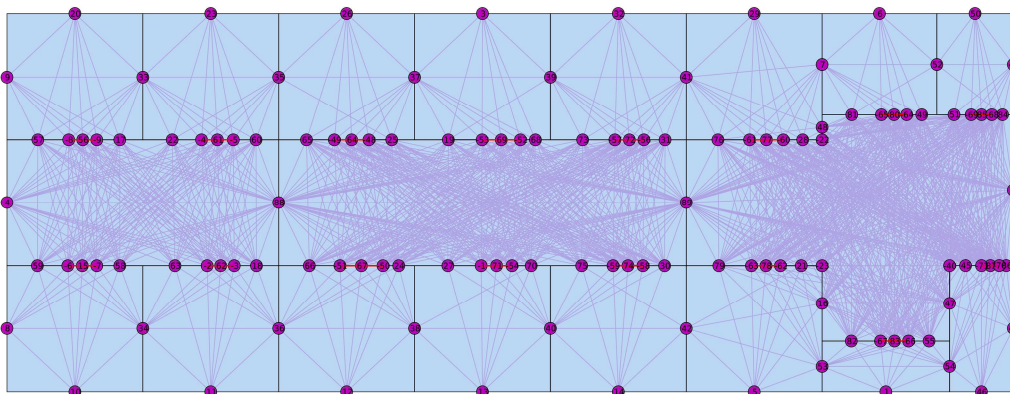


Figure 2-11 Graph of Visibility G_v

One important requirement for future ray estimation is to retrieve the type of interaction. Here, one difficulty is to make the distinction between reflection and transmission. The proposed manner to solve this problem is to introduced a new graph derived from G_v which make clear for each segment which is the involved type of interaction.

Graph of Interactions G_i

The graph of interactions G_i is derived from the G_v . The graph of interactions G_i is a directed graph which gathers all the possible interactions between nodes V_s of a given cycle of G_t .

Practically, a single interaction on a segment of G_v can potentially generate until 4 interactions in G_i : The reflection from a side of the interaction and/or the reflection from the other side and/or the transmission in one way and/or the transmission in the other way. Figure 2-12 shows these distinctions.

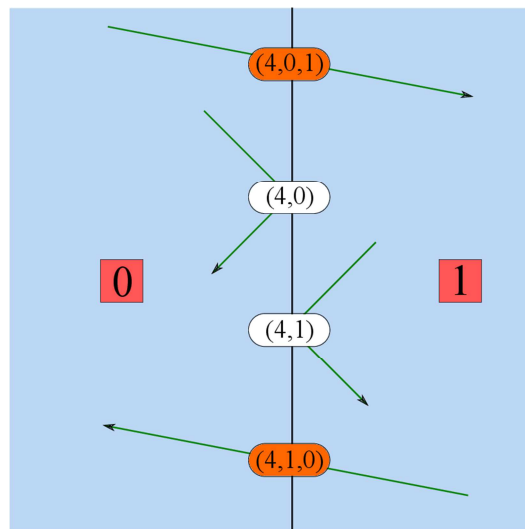


Figure 2-12 The 4 potential types of interactions on a given material

Three types of electromagnetic interactions are to be considered: the transmission (T), the reflection (R), and the diffraction (D). According to the notation defined in section 2.1 :

- (v_s^i, v_t^k) is a reflection on a segment $v_s^i > 0$ and cycle v_t^k
- (v_s^i, v_t^k, v_t^l) is a transmission across segment $v_s^i > 0$ and from cycles v_t^k to cycle v_t^l .
- (v_s^j) is a diffraction on node $v_s^j < 0$

An example of resulting graph of interactions G_i is shown in Figure 2-13 .

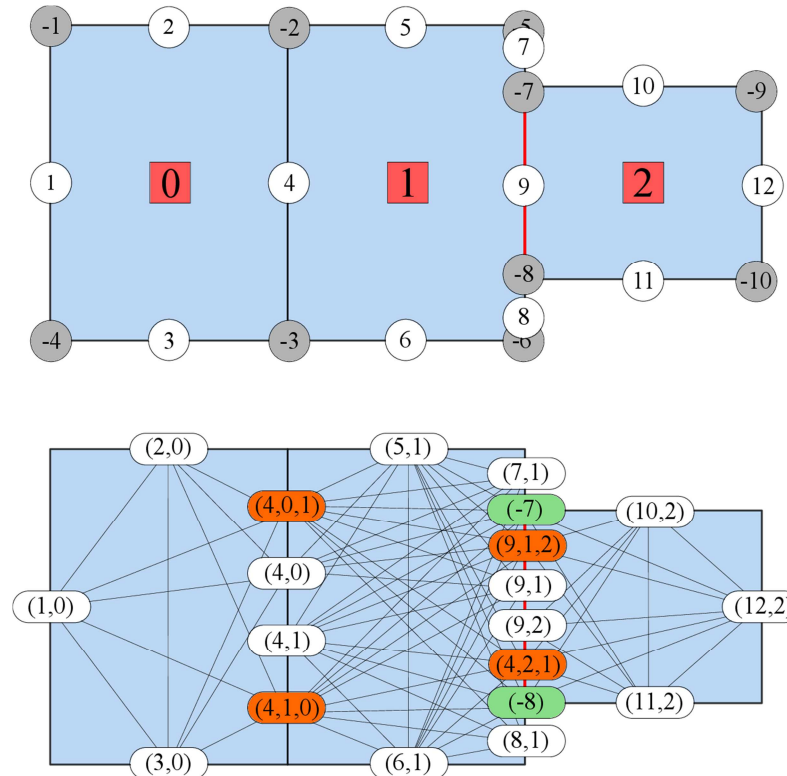


Figure 2-13 The graph on the top combines several graphs: segments nodes (positive number in white dots) and points nodes (negative numbers in grey dots) are from G_s , and the cycle number (number in red square) are from G_t . The graph below is the associated graph of interaction G_i . The nodes corresponding to reflection, transmission and diffraction are colored in white, orange and green respectively. Segment 4 corresponds to different type of interactions : (4,0) reflection toward cycle 0, (4,1) reflection toward cycle 1, (4,0,1) transmission from cycle 0 to 1, and (4,1,0) transmission from cycle 1 to 0

Nodes of G_i consists of heterogeneous interactions tuples. Hence written, it can be observed that the node names of G_i are tuples which contain information about:

- the index of involved segments or points from the G_s nodes,
- a type of interaction reflexion or diffraction depending on the number of elements in tuple.

That specific description would be very convenient in the new section, for the Signature description.

2.4.2 RAY SIGNATURES

The propagation channel parameters are time varying due to the motion of the extremities of the radio link or the motion of humans or objects evolving in the propagation environment. In spite of those movements, path persistency has been defined in [14] as the evolution of a particular path of the CIR which exhibits a differential change of a given feature in accordance with its differential motion. Considering a RT approach, such

differential change into the CIR, should be related to something observable into the rays, or more precisely into their related lists of interaction points. It appears that they are related by the **nature** of the involved slabs and the **type** of interactions on those slabs.

Based on this observation, it can be defined a ray-signature (or a signature) as an individual contributor of the stationary part of the CIR. By extension, the stationary part of the CIR is obtained by aggregating all the signatures.

Hence defined, it can be observed that a signature is not hardly related to a transmitter or a receiver position and can be the same for a small (or in some cases large) position change. This means that it can be used for generating a whole set of rays between two different regions. The use of graph for modeling stochastic channel has also been introduced and developed in [15] to describe the channel reverberation inside a room. This study has shown that the reverberating part of the CIR inside a reverberating volume can be explained and reproduced with high fidelity by using a graph description. Inspired by this representation, it is proposed in the following to take advantage of a graph description for building site specific signatures.

2D rays

The concept of signature is introduced to build 2D rays. 2D rays are 2D poly-lines whose each segment is delimited by interaction or termination points in $(O, \hat{\mathbf{x}}, \hat{\mathbf{y}})$. A 2D ray \mathbf{r} with L_h interaction points is an ordered list of $L_h + 2$ points, assuming that 2 points \mathbf{p}_{-1} and \mathbf{p}_{L_h} representing either the transmitter or the receiver are added in first and last position respectively. Thus, a 2D ray can be written:

$$\mathbf{r} = [\mathbf{p}_{-1}, \mathbf{p}_0, \dots, \mathbf{p}_{L_h-1}, \mathbf{p}_{L_h}] \quad (43)$$

2D rays are obtained in $(O, \hat{\mathbf{x}}, \hat{\mathbf{y}})$ and the tilde notation indicates that their components are given with only 2 dimensions. Hence, a 2D ray \mathbf{r} is a sequence of interactions points \mathbf{p}_i defined in $(O, \hat{\mathbf{x}}, \hat{\mathbf{y}})$.

Ray signatures

A signature is the association of 2 ordered sequences : the nature of the support (slab) and the type of the interactions. The nodes of G_i , as described in section 2.4.1, contain information about the cycles involved in the interaction, the nature of the interaction, the associated index corresponding to a slab. Those two last information can be advantageously used to build a signature.

Formerly, a ray signature $s = \mathcal{S}(\mathbf{r})$ of a 2D ray $\tilde{\mathbf{r}}$ with L_h interactions is the association of two ordered sequence of length L_h . A first sequence provides the nature of the slab and a second sequence provides the associated type of interaction among (R,T,D). Signatures can be obtained either from determination of paths in the oriented graph G_i or derived from an existing ray. Thus, a signature can be written:

$$\mathbf{s} = \begin{bmatrix} n_0 & n_l & n_{L_h-1} \\ t_0 & t_l & t_{L_h-1} \end{bmatrix} \quad (44)$$

Obtaining a signature from a sequence of nodes of G_i is a 2 steps process:

- determining two subsets of nodes $V_i^{p_{-1}}$ and $V_i^{p_{L_h}}$ from G_i which are in direct visibility of \mathbf{p}_{-1} and \mathbf{p}_{L_h} respectively.
- finding all simple paths (paths with no repeated nodes) between any distinct pair of interactions $(V_i^{p_{-1}}, V_i^{p_{L_h}})$.

Those return paths are ordered sequences of nodes G_i , which can be easily transformed into signatures. Algorithm 1 describes that procedure.

Algorithm 1 : Determination of signatures

Require: $G_i, \mathbf{p}_{-1}, \mathbf{p}_{L_h}$

$\mathbf{S}=[]$

$V_i^{p_{-1}} \leftarrow \text{get-visible-interactions}(G_i, \mathbf{p}_{-1})$

$V_i^{p_{L_h}} \leftarrow \text{get-visible-interactions}(G_i, \mathbf{p}_{L_h})$

For it **in** $V_i^{p_{-1}}$:

For ir **in** $V_i^{p_{L_h}}$:

$\mathbf{S}.\text{append}(\text{all-simple-path}(G_i, \mathbf{p}_{-1}, \mathbf{p}_{L_h}))$

End For

End For

$\mathbf{s} = \text{reshape-to-signature}(\mathbf{S})$

The procedure to obtain 3D rays from those signatures and to determine the corresponding evaluated field on each ray is detailed in [16]

2.5. IR-UWB CHANNEL IMPULSE RESPONSE DETERMINATION EXAMPLE

This section propose to illustrate the dynamic simulation and the propagation tool with the help of examples. For that purpose The CIRs of an agent moving into a layout are computed taking advantage of the proposed signature description. In this part, the Multi-cylinder body model is not added to the agent mobility, and the CIR is determined on the center of gravity of the agent.

2.5.1 DISPLAYING THE MOBILITY

Figure 2-14 is an example of a short mobile simulation.. In this example, an agent is moving into the layout from a room to antoher. It is potentially connected to 3 Acces points represented with triangles. The different ground truth positions of the agent are sampled every second and plotted as red dots. Between each dots, 1 second has elapsed. Every 10

seconds the small dots are replaced by larger dots with a number. These larger dots are *checkpoint positions*.

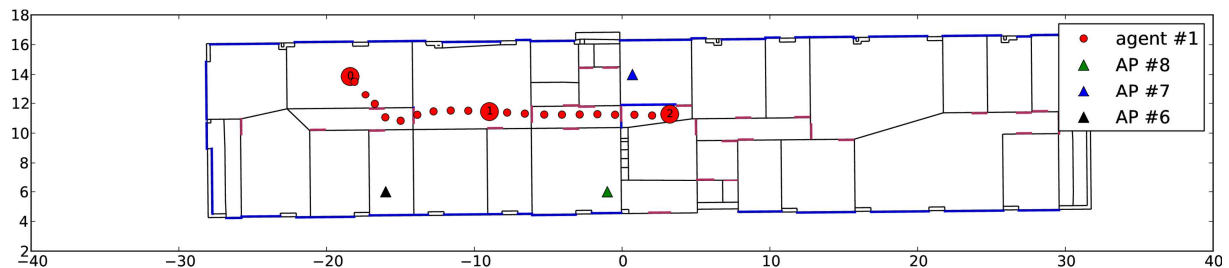


Figure 2-14 Simulation of a trajectory of 2 mobile agents and 4 static access points

From that simulation a log file is saved containing all the positions of each nodes (mobiles and statics) and the radio link between all nodes. Hence, the log file contains:

- The true node positions
- The estimated node positions (if one localization algorithm has been setup)
- The received power and delay for all the links at each timestamp.

2.5.2 COMPUTING THE CIR USING THE RT

From such a simulation a the log file, it is possible to post process the dynamic simulation with the RT tool and evaluate the CIR of all the links of the layout.

The advantage of the graph based RT is the signature description. As explained in section 2.4.2, signatures have been introduced to represent the persistent component of the radio channel. It means that for a given environment, the channel perturbation are caused by some specific signatures which can be directly related to specific pattern into the CIR.

From a room to another, signatures are assumed to be stationary. Practically it means that while the transmitter or the receiver remain in their respective rooms, the same set of signatures can be used. It can be noticed that the signature description is only related to the environment and do not involve the position of neither the receiver nor the transmitter. Consequently, as long as an agent belong to a same room, the same set of signatures can be reused to calculate associated rays.

As an illustration, CIRs of the mobile agent is computed for points of its trajectory belonging to the same room. On the left column of Figure 2-15 are displayed the obtained rays between 6 positions of the agent and an access point. Facing these layout representations, the corresponding obtained CIR are displayed. The 6 set of rays and CIRs have been obtained using the same set of signatures.

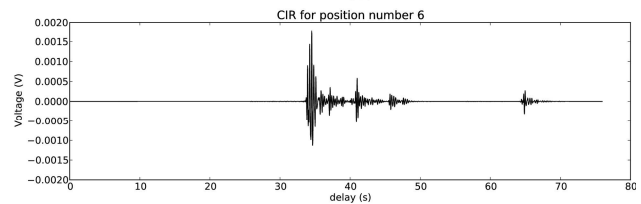
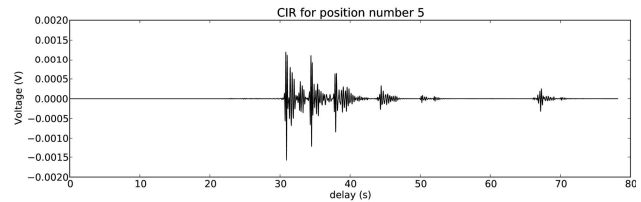
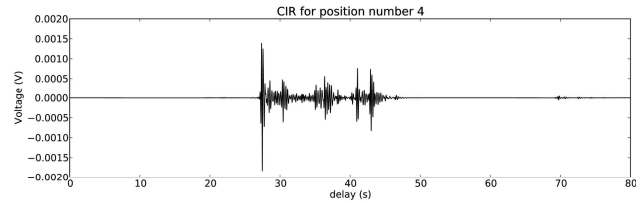
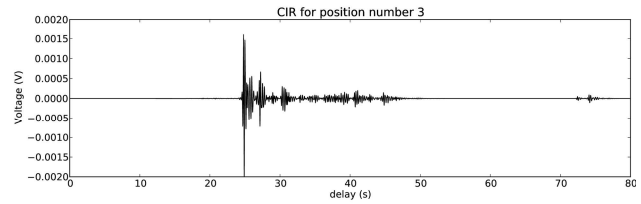
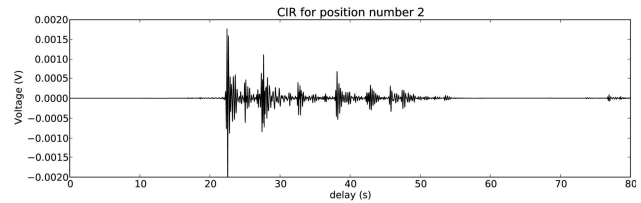
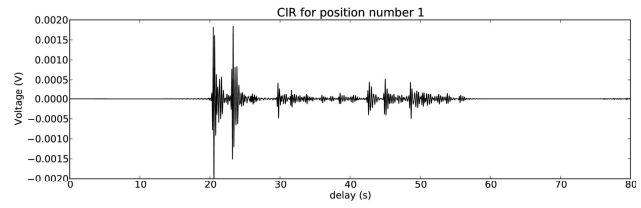
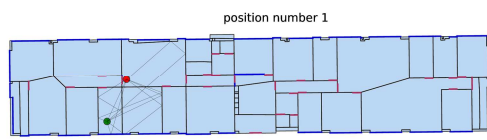


Figure 2-15 The figures on the left show computed rays between the mobile agent and an access point for different position of the mobile agent along the trajectory. On the right column are displayed the evaluated CIRs corresponding to the situation on the left column.

For the 6 CIRs obtained in Figure 2-15, Figure 2-16 displays an histogram of the computation time for each of those signatures. As expected, the computation of the first CIR is longer due to the cold start computation of the signatures. However, for the next 5 positions of the mobile agent (which still belong to the same cycle), CIRs are obtained significantly more rapidly.

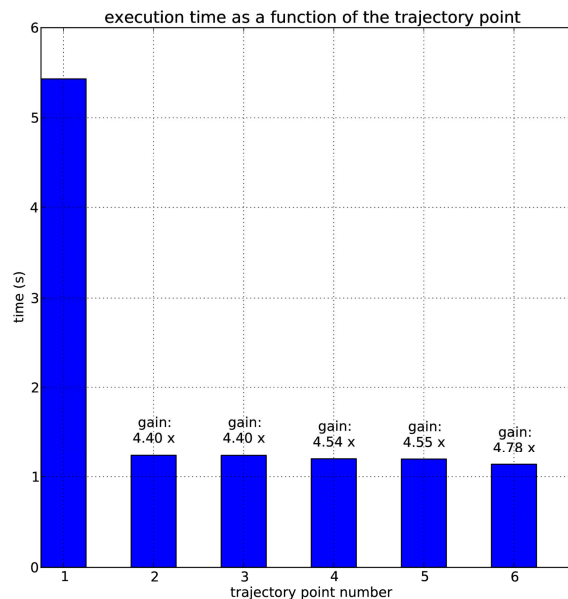


Figure 2-16 Time required producing CIR for 6 different positions of the mobile node. The gain indication informs about speed improvement in CIR computation relative to the first cold start requiring signature computation (position 1). The used computer to perform that simulation is an Intel Core I7 with 4Gb of RAM.

2.5.3 EXPLOITABLE OBSERVABLES FROM THE CIR

TOA Error Model

The channel modeling that has been presented before can be highly exploited for assessing the localization under real physical conditions. In this context, some future works would consider modeling the ranging error based on the Impulse Radio - Ultra Wideband (IR – UWB) Time of Arrival (TOA) estimation. Figure 17 shows a block diagram, which can be addressed in the modeling methodology. On this occasion, a TOA estimator should be involved, such as Energy Detector (ED) that is classified as a non-coherent estimator and characterized by low power consumption. The variation of the resulting model parameters should account for the multipath components and the Signal to Noise Ratio (SNR), as well as to the channel obstruction (LOS/ NLOS). Indeed, it is expected to observe ideal detections of the TOA, under high SNR and LOS conditions. However, we may find that the observable path in NLOS cases is shifted independently of the path power, hence leading to a non-neglected ranging bias.

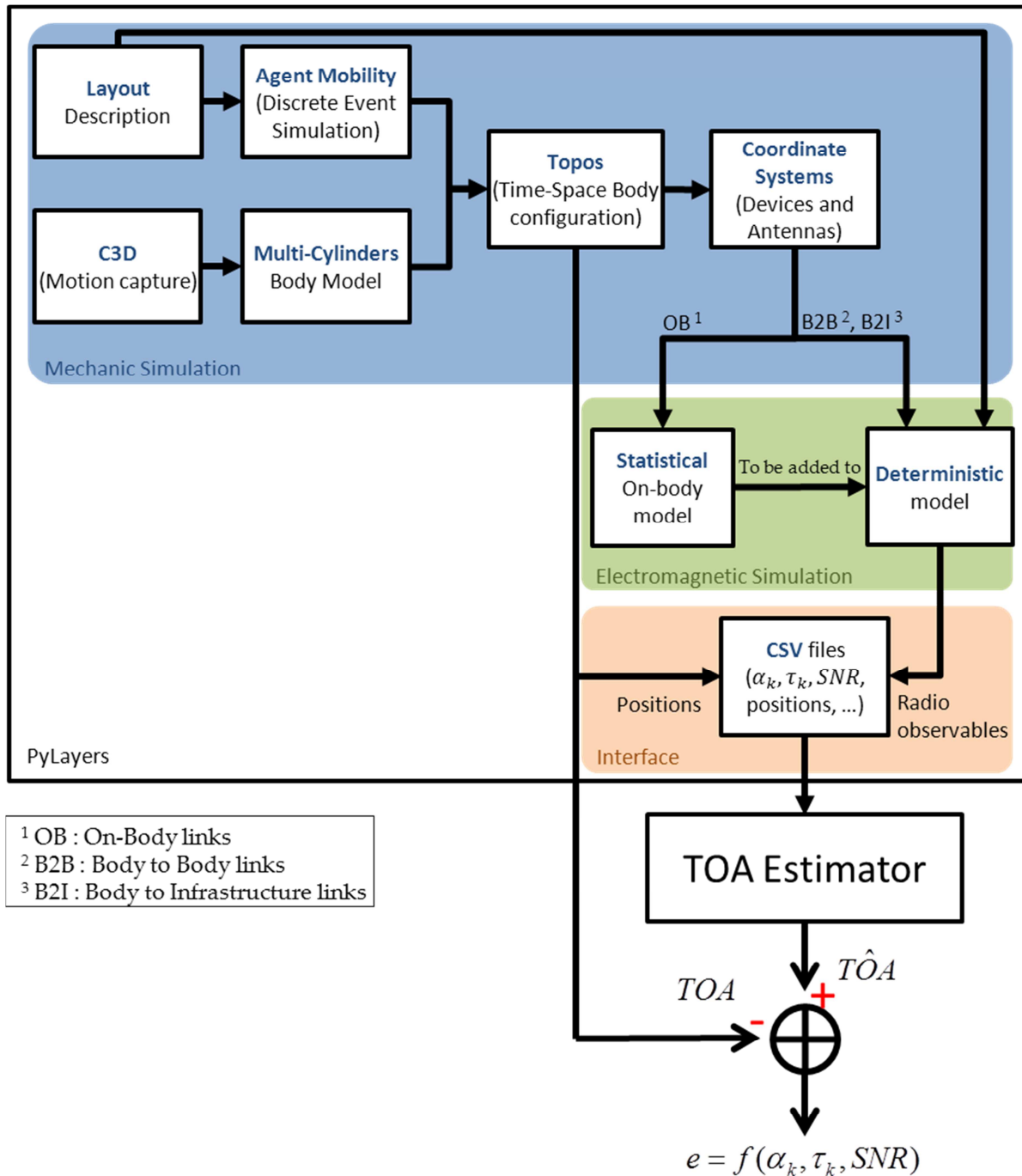


Figure 2-17: Block diagram of the expected ranging error methodology based on representative WBAN channels.

BER and PER

Obtaining the Bit Error Rate (BER) and/or Packet Error Rate (PER) from the CIR depends on a PHY layer emulation or simulation. The different approaches to address that particular problem are studied and discussed in section 5.3

3. WSNET DESCRIPTION

WSNet is an event-driven simulator for wireless networks on a large scale developed at the CITI laboratory [17]. Its principal features:

- **Written in C**
- Uses **free and accessible libraries**
- Configured using **XML files** to simulate
- Management of a **massive number of nodes**, up to several thousand, to meet the specific needs of the wireless sensor networks.
- Generation of random geometric graphs
- Uses a **“realistic” physical layer to develop algorithms for the upper layers** of the cell networks.

This last point is the most interesting to work within the CORMORAN project, which is the utility of the PHY and MAC layers. These layers can be defined and customized by the user. However, in the study of BAN networks behavior in a crowd simulation, the scaling can be interesting.

A number of secondary constraints were placed on the development of WSNet and are now as features of the simulator:

- Complete and modular model for the software architecture of nodes, i.e. the different layers of the OSI model.
- Taking into account the actual hardware architecture of sensors when the simulation is coupled to WSIM [18].
- Modeling of collisions and interference.
- Modeling of mobility and physical environment – such as the presence of obstacles or fire.

3.1. UNDERLYING METHODS

3.1.1 SIMULATION WITH DISCREET EVENTS

WSNet uses a mode of simulation said in discrete events. The principle is to consider that the system fits its behavior only to the appearance of events in time. By opposition to a simulation at continuous time, a simulator with discrete events can so save up a great deal of intermediate calculations, to focus only on the critical moments of the simulation. It supposes nevertheless that the evolution and the behavior of the system is in agreement with the hypothesis of departure.

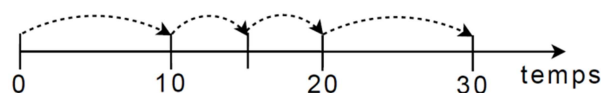


Figure 3-1 : Appearance of events at specific times of the temporal evolution line [17]

In the Figure 3-1, we can see the discreet moments when the simulator is going "to wake up" and to update the state of the system. The temporal notion is considered, but no update of the simulated system is sensible to be made outside these moments of awakening. Within the framework of simulation of network, these events will be mainly the sending and the reception of packages, and the evolution of the position of nodes and the physical environment within the framework of simulation with mobility.

3.1.2 ARCHITECTURE

WSNet architecture is represented as blocks modeling the properties of the radio medium, the simulation core and the characteristics of simulated nodes Figure 3-3:

- **The simulator** core allows generating and processing **the events and the packets** of the simulation. This is a key element of the simulator performance because it determines the speed degree and the effectiveness of simulations.
- **The radio medium abstraction** consists of three sub-blocks: **propagation, interference and modulation.**
- **The node abstraction** is composed of three sub-blocks:
 - o **Behavior** describes the mobility and the environment in which the nodes can evolve (fire, obstacles, etc)
 - o **Hardware** defines the different physical properties of each node (TX/RX frequency, TX power, battery)
 - o **Software** describes the upper layer of the simulation (MAC, NWK and application layer)

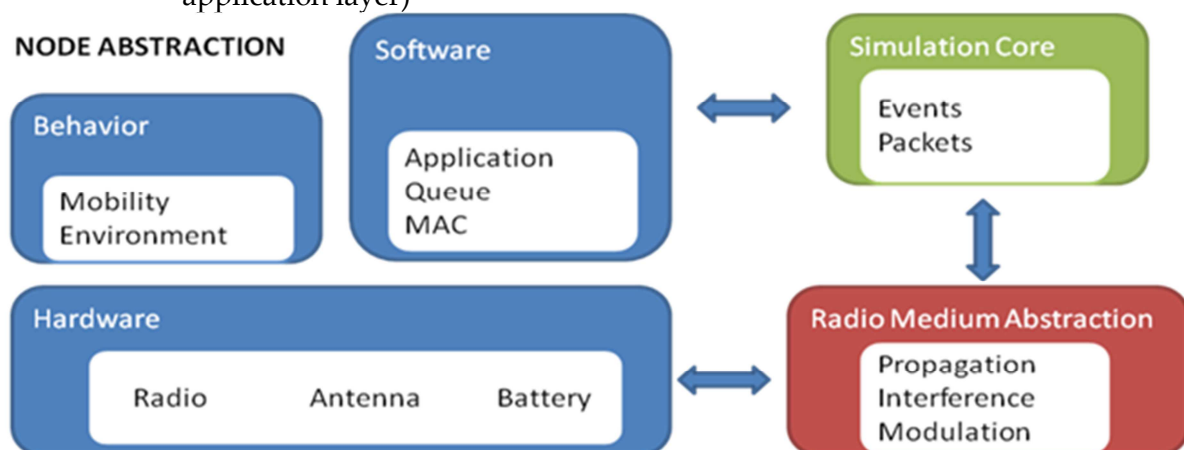


Figure 3-2 : WSNet architecture

WSNet possesses a modular structure, which is partially represented on the Figure 3-3. A node is a set of **superimposed layers**, which communicate directly with the lower or upper layers by the intermediaries of functions TX and RX. In every layer corresponds a particular

model, which must be written and compiled in C. These models include at the same time either the instances of the TCP/IP protocol for high layers, or else simple mathematical calculations of BER (Bit Error Rate) for a modulation layer.

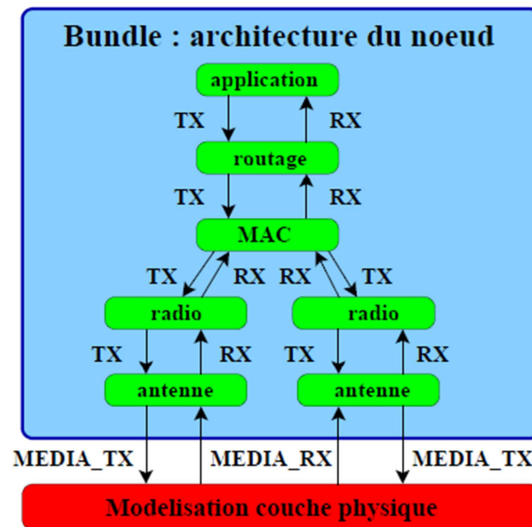


Figure 3-3 : Node architecture in WSNets and modules organization [17]

The modeling of the radio takes care of the decision of reception or non-reception of packages, leaving the MAC layer to arbitrate in the access to the channel. The models existing for the MAC layer concern mainly the protocol 802.15.4, in 868MHz, 902MHz and 2.4GHz. These models implement particularly the back-off described in the protocol.

The modeling of the physical layer is common to all the nodes and is composed of following modules:

- **Propagation:** this module is the lowest of the layer, and represents the path loss and all the effects bound to the power of the signal, such as the shadowing or the fading. The interferences are calculated by another module.
- **Interference:** this module gets back the value of the SNR of the propagation module and calculates the SINR, the ratio Signal on Interferences + Noise. This calculation can be made with an arbitrary granularity, which is once for the entire package, or once on every bit, or for any intermediate division.
- **Modulation:** this module sends back the BER following the value of the SNR or SINR of the propagation or interference layers.
- **Antenna:** this module allows integrating the properties of antennas, especially the different gains following the angle of transmission between the nodes.

3.2. CHANNEL MODEL AND CURRENT LIMITS OF THE SIMULATIONS

At present WSNNet integrates the following propagation layers into its basic version:

- **File Static:** this model simulate a binary radio link with a certain probability of error, defined first in a file for all the links.
- **Disk model:** this model is a link on-off where the path loss is neglected until a certain distance, then maximal beyond.
- **Free Space:** this model simulates a path loss in free-space depending on the distance between nodes ([19], p. 71)
- **Two-ray ground:** this model simulate a propagation of type double ray, by considering a perfectly flat ground ([19], p. 89)
- **Lognormal shadowing:** this model combines the *free-space* propagation, which is multiplied by a log-normal random variable for every drawing of the channel.
- **Rayleigh fading, Nakagami fading:** these two models combine the propagation free-space with a component of shadowing, which can follow a Rayleigh's law or a Nakagami's law.

Although WSNNet allows a finer simulation of the propagation layer than the other simulators, it remains however limited. The propagation layer generates in fact only a scalar value of SNR, which is passed in the interference layer, or modulation layer. Then, the modulation layer calculates a BER following this value of SNR (or of SINR if the interferences are considered), and the value of BER is then dismissed to the simulator which will update a probability of error for the package. The considered statistics are only of order 1, the spatial or temporal correlation of the channels must be integrated into the current models.

In order to deal with this problem, a usable temporary solution was to create a propagation model reusing measures or simulations made in an external way to WSNNet. At the first, all the links are generated outside WSNNet, and transmitted during the initialization of the simulation under the valuable shape of SNR. In every event, WSNNet gets back the SNR corresponding in the table of the pre-generated values, and adds to it a log-normal shadowing component and a fading component (Rayleigh, Rice or Nakagami).

This model has the advantage to be extremely flexible, because we can use as base of simulation of the measures already made in real conditions, either the successful tools of generation of signals contained in Matlab for example (This point explained in section 4). Furthermore, it is simple to integrate the spatial and temporal correlation of the state of the links. Nevertheless, it is necessary to pre-calculate all the values of SNR for the links, what reduces considerably the interest of the in time discreet simulator. When this calculation is made, we can nevertheless reuse channels generated for several simulations.

In the case of the BAN, following the models studied in particular in [20], the evolution of the random variable of masking follows a normal law, which is correlated in the time. This temporal evolution is modeled under the shape of an auto-regressive series. Consequently, to simulate such a series, it is necessary to apply an auto-regressive filter with a white noise. The generation cannot thus suffer from the absence of certain samples and it is not possible to jump from a time t_1 to another time t_2 without generating all the intermediate values.

3.3. INTERFACING

3.3.1 INPUTS:

The nodes are instantiated and described in the central XML file of configuration. The simulation chain is the following one:

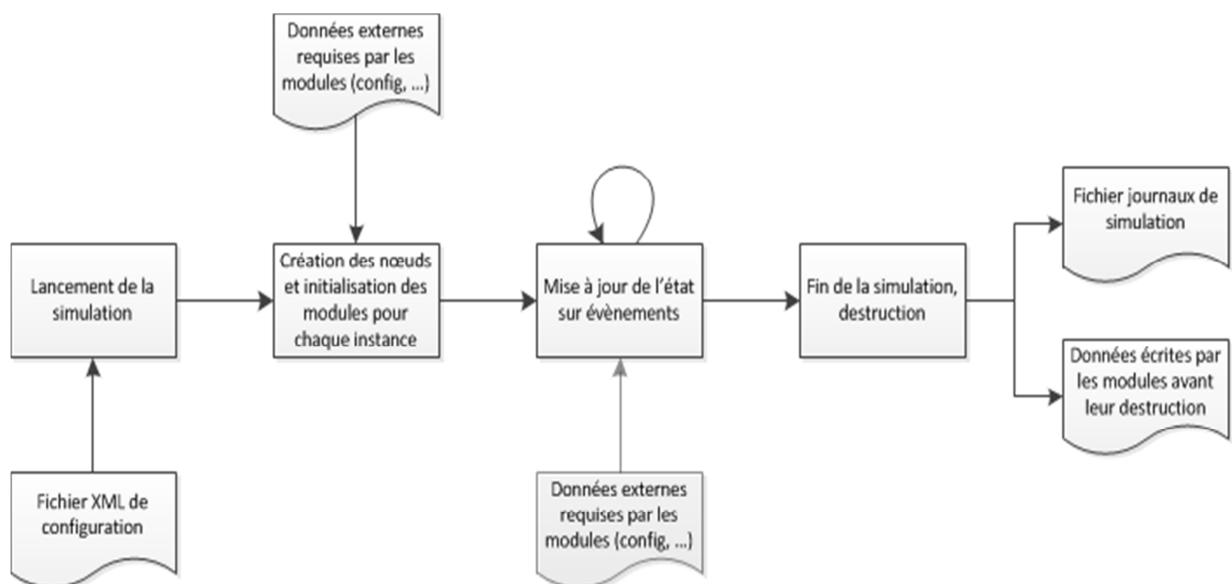


Figure 3-4 : Progress of a simulation in WSNet

We use xml files to configure a simulation (in the installation we ran cbr.xml as simulation). This file describes the simulation setup and specifies, for example, the number of nodes to simulate, the libraries used to model the radio medium and the nodes (e.g., for propagation, routing, etc).The file XML allows to store certain valuable number of initialization for modules, these values which can be then completed by external files as indicated on the Figure 3-4.

3.3.2 OUTPUTS:

In release, log files are generated, in particular for the mobility of nodes and these log files can be replayed in a graphic interface.

3.4. FILES XML

In WSNET, we use xml files to configure a simulation. This file describes the simulation setup and specifies, for example, the number of nodes to simulate, the libraries used to model the radio medium and the nodes (e.g., for propagation, routing, etc). It is composed of five main sections:

- **Global parameters** allow defining a simulation with a specific syntax. These parameters are *number of nodes, duration, width (x), length (y) and height (z) of the simulation*.
- **Entities** are an instantiation of dynamic libraries. These libraries can be instantiated several times in different entities. Each of these entities may have different global-parameter values and node-parameter default values. The parameters are *name of entity, name of the dynamic library, global parameters (init) and node parameters (default)*.
- **Environment** defines the radio medium and the physical environment of the simulation. The parameters are *the name of propagation, the name of interferences, the name of a monitoring entity, the name of one modulation entity (modulation should appear at least once) and the name of one environment entity (with - optional)*.
- **Bundles** correspond to the node architecture abstraction. As we said before for the node abstraction, the parameters are *the name of bundle, default bundle (to specify if this bundle is the default one or not, values true or false), default birth (optional, default value = 0), the mobility entity, the energy entity, the antenna entity*.

Then there are more options that can be repeated several times: with to define specific entities in the bundle such as the upper layers, the uplink, the downlink and the default values depending on the entity description ¹

- **Node** instantiates a bundle and this define the node behavior. The parameters are *the node id, the bundle name, the birth date of node (optional) and for (optional) that is used to override parameters of an entity defined in the bundle*.

The simulation configuration file is specified using the command line option -c.
For example: `wsnet -c config.xml`.

This allows you to launch the simulation. The next example is the one used for the CORMORAN project:

¹ It is advisable to see the parameters of libraries on *the models WSNet site* to configure entities, environment and bundles. For more details go to *the WSNet xml configuration link*.

```
<?xml version='1.0' encoding='UTF-8'?>
<worldsens xmlns="http://worldsens.citi.insa-lyon.fr">
```

All xml files for WSNET start and finish with <worldsens> instance. See at the end of the example.

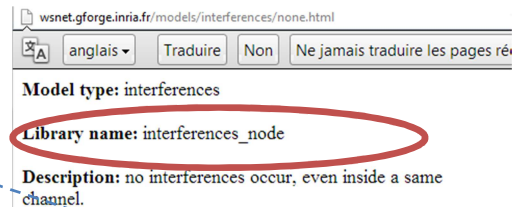
```
<!-- == Worldsens ===== -->
<simulation nodes="4" duration="50s" x="2" y="2" z="2"/>
```

This part corresponds to the **global parameters**. We assume that 4 nodes are deployed over an area of 2x2x2 during 50 seconds.

```
<!-- == Entities ===== -->
<!-- == PROPAGATION, INTERFERENCES, MODULATION and ENERGY ===== -->
<entity name="ban_propag" library="propagation_ban" >
  <init data_description_file="user_models/ban_links_description.csv"
  fading_model="rician" />
</entity>
```

After global parameters, we define the **entities** and the instances to the libraries and models. In the first entity, **we found the instance of our model ban_propag** explained in 4.2. **We configure our propagation model with a rician fading**

```
<entity name="mod_oqpsk" library="modulation_oqpsk">
</entity>
<entity name="interf" library="interferences_none">
</entity>
<entity name="bat" library="energy_linear">
  <default energy="100000" tx="0.00468" rx="0.0555" idle="0"/>
</entity>
```



Then we define modulation, interferences and energy. The user must be careful with the name of the library instance. This is show in the models table [WSNet site](#). The name is up to the user to define and this is the name to call the entity after.

```
<!-- == RADIO and ANTENNA ===== -->
<entity name="omnidirectionnal" library="antenna_omnidirectionnal" >
  <default loss="0" angle-xy="random" angle-z="random"/>
</entity>
<entity name="radio" library="radio_half1d" >
  <default sensibility="-92" T_s="90" dBm="0" channel="0" modulation="mod_oqpsk"/>
</entity>
```

Next, it is defined the radio and antenna entities of the PHY layer. In the example it is shown an omnidirectionnal antenna and the radio is half duplex.

```
<!-- == MAC ===== -->
<entity name="mac" library="mac_802_15_4_2400_oqpsk_u_csma_ca" >
</entity>
```

After that, it is defined the MAC layer as the IEEE 802.15.4 at 2.4 GHz using QPSK and CSMA.

```
<!-- == Application ===== -->
<entity name="hip_app" library="application_cbr" ></entity>
<entity name="back_app" library="application_cbr" ></entity>
<entity name="rthigh_app" library="application_cbr" ></entity>
<entity name="tfoot_app" library="application_cbr" ></entity>
<entity name="lthigh_app" library="application_cbr" ></entity>
<entity name="tfoot_app" library="application_cbr" ></entity>
<entity name="torso_app" library="application_cbr" ></entity>
<entity name="rarm_app" library="application_cbr" ></entity>
<entity name="rhand_app" library="application_cbr" ></entity>
<entity name="larm_app" library="application_cbr" ></entity>
<entity name="lhand_app" library="application_cbr" ></entity>
<entity name="rear_app" library="application_cbr" ></entity>
<entity name="lear_app" library="application_cbr" ></entity>
```

Subsequently, the application layer is defined for each node with a simple Content Based Routing.

```
<!-- == MOBILITY ===== -->
<!--
<entity name="static-pnc" library="mobility_static" >
  <default x="0.433" y="0.433" z="0"/>
</entity>
<entity name="static-dev1" library="mobility_static" >
  <default x="0.683" y="0.866" z="0"/>
</entity>
<entity name="static-dev2" library="mobility_static" >
  <default x="0.183" y="0.866" z="0"/>
```

Then it can be possible to choose the mobility of nodes. In the example, some parameters are shown as comments and the mobility is defined as a static mobility entity.


```
</entity> -->

<entity name="static" library="mobility_static">
  <default x="random" y="random" z="random"/>
</entity>
```

After entities, it is defined the **environment**, in this example, the propagation is the ban_propag entity created for BANET. Then it is chosen the environment without interferences and with a QPSK modulation.

```
<!-- == Environment ===== -->

<environment>
  <propagation entity="ban_propag"/>
  <interferences entity="interf"/>
  <modulation entity="mod_oqpsk"/>
</environment>
```

```
<!-- == Bundle ===== -->
<bundle name="hip" worldsens="false" default="true" birth="0">
  <mobility entity="static"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>
  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>
  <with entity="mac">
    <up entity="hip_app"/>
    <down entity="radio"/>
  </with>
  <with entity="hip_app">
    <down entity="mac"/>
  </with>
</bundle>
```

After the environment, the **bundles** must be defined, in this example there is a bundle for each node (it is shown only two of the bundles of the real file).

The first one is for the hip node. It is composed by the static mobility entity, followed by an omnidirectional antenna entity and remark the up entity that is the radio.

At the up and down entities, it is defined the chain of the PHY, MAC, NWK and application layers. Each layer must be defined in the correct order.

The last entity corresponds to the application layer hip_app defined before with the Content Based Routing.

```
.....

<bundle name="lear" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>
  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>
  <with entity="mac">
    <up entity="lear_app"/>
    <down entity="radio"/>
  </with>
  <with entity="lear_app">
    <down entity="mac"/>
  </with>
</bundle>
```

Then we do the same for the left ear node.

At the end, each node is defined with their corresponding bundle.

```
<!-- == Nodes ===== -->
<node id="0" as="hip"/>
<node id="1" as="back"/>
<node id="2" as="rthigh"/>
<node id="3" as="lear"/>
```

This define the end of the simulation xml file

```
</worldsens>
```


4. FULL MESH MEASUREMENT BASED SIMULATION

As we describe on the chapter 3, it is possible to customize the models for WSNET. We also introduced a propagation model that is capable to reuse external measures or simulations to WSNET. This is possible by using CSV files containing the parameters of each nodes and the state of all the links. Then the propagation model of WSNET will take these measures into a structure data base on C during the simulation.

In this chapter, we will describe how WSNET can use a full mesh measurements data base for simulation of the upper layers.

4.1. CSV FILES IN WSNET

CSV stands for 'Comma Separated Values,' (strict form as described in RFC 4180) so CSV files have a comma between each item of information they contain. Since a CSV file is a simple text file (ASCII or Unicode) it can be easily opened by almost any text editor.

4.1.1 CSV FORMAT

WSNET follows the conventions for CSV files described at [21] which seem to reflect the most common usage of the format, namely:

- Fields are separated with commas.
- Rows are delimited by newline sequences (see below).
- Fields may be surrounded with quotes.
- Fields that contains comma quote or newline characters **MUST** be quoted.
- Each instance of a quote character must be escaped with an immediately preceding quote character.
- Leading and trailing spaces and tabs are removed from non-quoted fields.
- The final line need not contain a newline sequence.

In strict mode, any detectable violation of these rules results in an error. A typical CSV file, where a comma is used to distinguish each record, looks like this:

```
Name1,Address1,Telephone1
Name2,Address2,Telephone2
Name3,Address3,Telephone3
```

Or if the "|" bar character is used to separate the data, it might look like:

Name1 | Address1 | Telephone1
Name2 | Address2 | Telephone2
Name3 | Address3 | Telephone3

In a practical context, WSNET may use two CSV files using comma to separate data. The first contains the measures of one link in a row. For example, the file named shadowing-Back.csv could contain the following measures:

11.875175,11.926053,11.976895,12.027699,12.078467,12.129197,12.179890,...

The second contains a description of the BAN links with the following parameters: transmitter node, receiver node, path to CSV file with the values of the BAN link, number of measures and time of simulation. For example, the file with the links description of a BAN with 4 nodes is showed bellow:

```
0,1,"csv/shadowing-Back.csv",3000,3
0,2,"csv/shadowing-R-Thigh.csv",3000,3
0,3,"csv/shadowing-R-Foot.csv",3000,3
1,0,"csv/shadowing-Back.csv",3000,3
1,2,"csv/shadowing-R-Thigh.csv",3000,3
1,3,"csv/shadowing-R-Foot.csv",3000,3
2,0,"csv/shadowing-Back.csv",3000,3
2,1,"csv/shadowing-R-Thigh.csv",3000,3
2,3,"csv/shadowing-R-Foot.csv",3000,3
3,0,"csv/shadowing-Back.csv",3000,3
3,1,"csv/shadowing-R-Thigh.csv",3000,3
3,2,"csv/shadowing-R-Foot.csv",3000,3
```

In this case, WSNET will read at the first time the file with the BAN links description. It will pass the parameters into a structure, which will create each link with their respective measures. For example, if we read the first row, we will create the link of node 0 to node 1 by reading the file shadowing-Back.csv that contains 3000 measures in a simulation of 3 seconds.

The simulation time of measures must be considered in the conception of the whole simulation of WSNET. In fact, as we work with a discrete time simulator we need to respect the time of events to collect samples to be simulated as realistically as possible. In the example, 3000 measures in 3 seconds means that there is not variation of the state of link in $3/3000 = 1\text{ms}$. So if some event arrives 2.5 ms after the start of the simulation, the state of link used will be the measure according to the second ms (Figure 4-1).

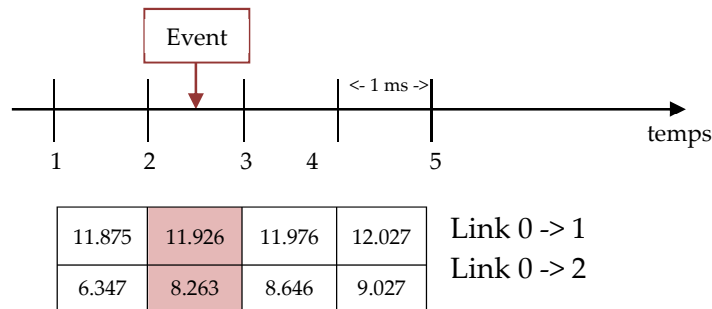


Figure 4-1 : Event arriving and evolution of the state of links

From this example, we can see that the resolution of the time is very important and it has to be as near as possible from the real variation of the channel.

4.1.2 LIBCSV LIBRARY

WSNET exploit CSV files by using a special library: **libcsv**. This library is free software under the terms of the GNU Lesser General Public License as published by the Free Software Foundation [21]. The CSV library provides a flexible, intuitive interface for parsing and writing csv data.

The idea behind parsing with libcsv is straight-forward:

- We initialize a parser object with **csv_init()** and feed data to the parser over one or more calls to **csv_parse()** providing callback functions that handle end-of-field and end-of-row events.
- **csv_parse()** parses the data provided calling the user-defined callback functions as it reads fields and rows.
- When complete, **csv_fini()** is called to finish processing the current field and make a final call to the callback functions if necessary.
- **csv_free()** is then called to free the parser object.
- **csv_error()** and **csv_strerror()** provide information about errors encountered by the functions.
- **csv_write()** and **csv_fwrite()** provide a simple interface for converting raw data into CSV data and storing the result into a buffer or file respectively.

4.2. FULL MESH PROPAGATION MODEL

4.2.1 CREATING A MODEL IN WSNET

During a simulation with WSNET, the entities on the xml file have an instance with a module. The modules are implementations of network models (radio propagation, interferences, mobility, MAC, application, etc).

There are a certain number of models included in the online version of WSNET. However, for the CORMORAN goal, we need to create our own models to support the different algorithms. Figure 4-2 shows the different models existing in WSNET, you can find it on [3]. There you can click on each model to have the details to configure it correctly such as model type, library name, description and entity parameters, these parameters are used in the xml file configuration. It is important to know the library name because is the name needed to put in the xml files to call the functions of the model. Then it is preferable to see the entity parameters description to know the values you can use for the simulation.

Models	Included in WSNet distribution
Radio propagation	file static , disk model (range), free space , tworay ground , lognormal shadowing , rayleigh fading , ITU indoor model , nakagami fading
Interferences	none , orthogonal , factor
Modulation	none , step , bpsk , oqpsk , mqam
Antenna	omnidirectional
Mobility	static , file static , billiard , torus central , torus plane , teleport
Battery/energy	linear
Environment	fire
Monitor	nodes , nrj
Radio	half1d , 802.15.4 868MHz bpsk , 802.15.4 902MHz bpsk , 802.15.4 2400MHz oqpsk
MAC	802.11 DCF , 802.15.4 868MHz bpsk , 802.15.4 902MHz bpsk , 802.15.4 2400MHz oqpsk , B-MAC , Ideal MAC
Routing	greedy geographic , file static
Application	CBR , CBR_v2 , Hello protocol , GHT , LBDD , XY , Data_Sink , Data_Source , GOSSIP , B-MAC application sample

Figure 4-2 : Event arriving and evolution of the state of links

Creation: The implementation of a new module is made by adding the source code of our module to the `/wsnet/user_models/` directory. Normally this directory is installed since the beginning, if not it is preferable to follow the instructions **to set up the working directory**.

Structure: Each module has a different architecture depending on the methods and the application of the model. But the basic architecture includes the following source code:

```
#include <include/modelutils.h>

model_t model = {
    "MODULE DESCRIPTION",
    "AUTHORS",
    "0.1",
    MODELTYPE_APPLICATION,
    {NULL, 0}
};

int init(call_t *c, void *params);

int destroy(call_t *c);

int setnode(call_t *c, void *params);

int unsetnode(call_t *c);

int bootstrap(call_t *c);

void rx(call_t *c, packet_t *packet);

/*
*****
*** */
application_methods_t methods = {rx};
```

This library is always present in the implemented modules. You can also include other libraries needed for your model.

This is a data structure describing the module (authors, version, etc). It is necessary to call the model with the xml files. Make sure there is only one model_t model structure in the code. You can also create more structures for your model as another C program.

The init function is necessary to initialize the application entity and the global entity parameters (values taken from the xml config file). Destroy is a function called at the end of the simulation to destroy the entity.

Setnode, unsetnode, bootstrap are methods used to make the interaction of the network nodes. These methods are useful for application implementation; however we are not using them in our example for BAN because we use csv files to take some nodes parameters for the simulation.

Finally, the rx method is the one who contains your application model and make the calculations when a packet is received. This function must be called at the end as an application method. The names can change between the different modules. In our example, this method is called propagation.

4.2.2 FULL MESH MODEL IN WSNET

As we said before, the goal is to external measures or simulations to WSNET. First, we need to implement a specific propagation model to compute the status of each channel link between the nodes. For that, the module will take entities and global parameters from the xml file (ban_cea.xml) presented in chapter 3.4. Then, the module will use csv files to take the shadowing values of the channel links of each BAN node and then compute the fading of these links. Finally, at the end of the simulation we can see all the status of links.

```
/**
 * \file ban-propag.c
 * \brief Deterministic BAN channel propagation layer
 * \author Paul Ferrand, Javier Cuadrado
 * \date 2010
 **/
```

The first comments are a description of the module

```
#include <include/modelutils.h>
#include <math.h> // Use fmod(x,y)
#include "libcsv/csv.h"
```

Here we call the libraries that are necessary for all implementation. We found the libcsv/csv.h library that contains the methods to manipulate the csv files

```
/* ***** */
/* ***** */
model_t model = {
    "BAN propagation model",
```

Data structure to describe the model: modeltype_propagation

```
"Paul Ferrand, Javier Cuadrado",
"0.1",
MODELTYPE_PROPAGATION,
{NULL, 0}
};

#define
...
#endif
```

```
/* ***** */
/* ***** */
typedef struct samples
{
    ... } t_samples;
struct entitydata
{
    ... };
typedef struct csv_params
{
    ... } t_csv_params;
```

After we create some structures to manipulate the csv files data. In this model, we want to parse the shadowing into tables to compute the fading.

```
/* ***** */
/* ***** Manipulate sample tables ***** */
/* ***** */

int init_tables(struct entitydata * entitydata){ ...}
__inline__
t_samples * get_samples_t( struct entitydata * entitydata, int src, int
dest){ ...}
int free_tables(struct entitydata * entitydata) { ...}
void reset_csv_params( t_csv_params * csv_params) { ... }
```

These are methods to manipulate the tables and the data contained in the csv files.

```
/* ***** */
/* ***** Callbacks for libcsv ***** */
/* ***** */

void _links_field_read(void *s, size_t i, void *p) { ...}
void _links_entry_read(int i, void *p){ ...}
void _config_field_read(void *s, size_t i, void *p) { ...}
void _config_entry_read(int i, void *p){ ...}
```

These are methods to call the libcsv library describe before

```
/* ***** */
/* ***** */
int init(call_t *c, void *params) { ...}
int destroy(call_t *c) {...}
```

We can find as well the init and destroy functions to create and destroy the entity.

```
/* ***** */
/* ***** Normal distribution ***** */
/* ***** */
double normal (double avg, double deviation) { ...}
```

This function generates a distributed random number (Normal distribution Box Muller transform) given a source of uniformly distributed random numbers. **It is used to compute the fading** (Rayleigh and Rice)

```
/* ***** */
/* ***** */
double compute_ban_fading(double pathloss, int fading_model)
{
    ...
    case BF_NONE:
    ...
    case BF_RAYLEIGH:
```

This function computes the fading, it takes into account the fading model chosen in the xml file in the entity propagation configuration. We call the normal method to compute normal random values. Then we compute the fading.

In the example we choose Rice fading that is computed according to the standard IEEE 802.15.6

```

...
case BF_RICE:
...
case BF_NAKAGAMI:
...
}
...
}

```

```

/* ***** */
/* ***** */
double propagation(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst,
double rxdBm)
{
...
fprintf(stdout, "[ban-propag] Time called : %f s (propagation).\n",
sim_time);
...

VERB(fprintf(stdout, "[ban-propag] P_tx (dBm) : %f.\n", rxdBm));
VERB(fprintf(stdout, "[ban-propag] Pathloss (shadowing) (dB) : %f.\n",
pathloss));
VERB(fprintf(stdout, "[ban-propag] Fading (dB) : %f.\n", fading));
VERB(fprintf(stdout, "[ban-propag] P_rx (dBm) : %f.\n", rx_dBm));
return rx_dBm;
}

/* ***** */
/* ***** */
propagation_methods_t methods = {propagation};

```

At the end we find the propagation method called by init to compute the receiver power and fading using the compute_ban_fading method.

This is also the responsible of the results showed during the simulation.

We see that the propagation method is defined at the end to be used by the xml file.

Once a model is created, it must be called by the XML file of the simulation on the entities section respecting the order of the transmission system. In this case, the full mesh propagation model will be declared before any upper layer (modulation, MAC, NWK, application). We choose a name for the entity (ban_propag), the library of the model (propagation_ban), the path to reach the full mesh ban links description file (user_models/ban_links_description.csv) and the propagation parameters (Rice, Rayleigh, etc). After what remains to be done is to create the library.

```

<?xml version='1.0' encoding='UTF-8'?>
<worldsens xmlns="http://worldsens.citi.insa-lyon.fr">

```

```

<!-- == Worldsens ===== -->
<simulation nodes="13" duration="50s" x="2" y="2" z="2"/>

```

```

<!-- == Entities ===== -->
<!-- == PROPAGATION, INTERFERENCES, MODULATION and ENERGY ===== -->
<entity name="ban_propag" library="propagation_ban" >
  <init data_description_file="user_models/ban_links_description.csv" fading_model="rician" />
</entity>

<entity name="mod_oqpsk" library="modulation_oqpsk">
</entity>

```

4.2.3 SIMULATION OF THE FULL MESH MODEL IN WSNET

Implementation: This section consists to start using the new model (ban_propag) by creating the model library and take a look at the propagation ban implementation: ban-propag.c (the model) and ban_cea.xml (the simulation), which are the files we explained before in our example. Then, all the files (csv libraries included) have to be within the $\$HOME/wsnet-module/user_modules/$ directory (Figure 4-3).

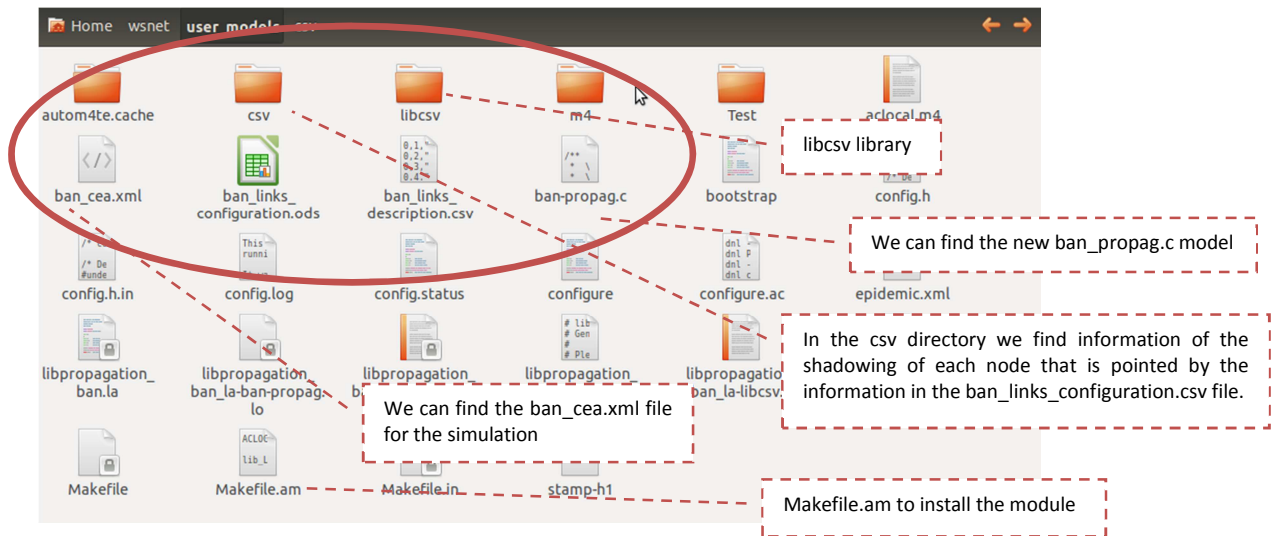


Figure 4-3 : Event arriving and evolution of the state of links

After this, we need to compile and install our new application module. For this in the user_models directory, we have to modify the content of Makefile.am. At the end, the file should contain the following entries:

```
ACLOCAL_AMFLAGS=-I m4
lib_LTLIBRARIES = libpropagation_ban_la
libpropagation_ban_la_CFLAGS = $(CFLAGS) $(GLIB_FLAGS) $(GSL_FLAGS) -Wall
libpropagation_ban_la_SOURCES = ban-propag.c libcsv/libcsv.c
libpropagation_ban_la_LDFLAGS = -module
```

Then, to compile ban-propag.c type the following commands at /user_modules/ directory:

```
$ cd ./wsnet/user_modules/
/user_models$ ./bootstrap
/user_models$ ./configure
/user_models$ make
```

After that, we have to install the module:

```
/user_models$ make install
```


If the installation was correct, then we will notice that some files have been created at the /user_models/ library (Figure 4-4).

```
arturo@arturo-OptiPlex-790:~/wsnet$ cd user_models
arturo@arturo-OptiPlex-790:~/wsnet/user_models$ ls
aclocal.m4          ban-propag.c      config.status     libcsv            libtool          resultats_ban.txt
autom4te.cache     ban-propag.c~    configure         libpropagation_ban_la  m4              stamp-h1
ban_cea.xml        bootstrap        configure.ac      libpropagation_ban_la-ban-propag.lo  Makefile        Test
ban_cea.xml~      config.h         csv              libpropagation_ban_la-ban-propag.o  Makefile.am
ban_links_configuration.ods  config.h.in     epidemic.xml     libpropagation_ban_la-libcsv.lo      Makefile.am~
ban_links_description.csv  config.log      epidemic.xml~    libpropagation_ban_la-libcsv.o       Makefile.in
arturo@arturo-OptiPlex-790:~/wsnet/user_models$
```

Figure 4-4 : Event arriving and evolution of the state of links

Simulation: Finally, we can simulate our module by typing in the wsnet directory:

```
~/wsnet$ wsnet -c user_models/ban_cea.xml
```

This will launch the simulation and it will start reading the xml parameters of the entities, then the bundles and finally it will show the results of the simulation.

```
Found 21 entities...
Found 1 environment...
Found 13 bundles...
```

```
Simulation will run using:
nodes      : 13
(x,y,z)    : (2.000000, 2.000000, 2.000000)
duration   : 50000000000ns
```

```
Entity 'ban_propag' (0)
  using model : propagation_ban
  using plugin : /usr/local/wsnet-2.0/lib//libpropagation_ban.so
  author      : Paul Ferrand, Javier Cuadrado
  version     : 0.1
  description  : BAN propagation model
[ban-propag] Fading set to Rician K.
[ban-propag] Initializing tables (init_tables).
[ban-propag] Entity data set!
[ban-propag] Reading the data description file (user_models/ban_links_description.csv) for 13 nodes.
[ban-propag] Configuration data read (_config_entry_read).
[ban-propag] Summary for the link :
[ban-propag] Source : 0
[ban-propag] Destination : 1
[ban-propag] File name : user_models/csv/shadowing-Back.csv
[ban-propag] Sample count : 3000
[ban-propag] Sampled time : 3.000000
[ban-propag] test 1: 3000 et 3.000000
[ban-propag] On est dedans
[ban-propag] Link data read (_links_entry_read).
[ban-propag] Entries : 3000.
[ban-propag] Configuration data read (_config_entry_read).
[ban-propag] Summary for the link :
```

```
[ban-propag] Source : 0
[ban-propag] Destination : 2
[ban-propag] File name : user_models/csv/shadowing-R-Thigh.csv
[ban-propag] Sample count : 3000
[ban-propag] Sampled time : 3.000000
[ban-propag] test 1: 3000 et 3.000000
[ban-propag] On est dedans
[ban-propag] Link data read (_links_entry_read).
[ban-propag] Entries : 3000.
[ban-propag] Configuration data read (_config_entry_read).
[ban-propag] Summary for the link :
[ban-propag] Source : 0
[ban-propag] Destination : 3
[ban-propag] File name : user_models/csv/shadowing-R-Foot.csv
...

```

The example shows the launch mechanism: First, the simulator loads the number of nodes, the size of the environment and the duration of the simulation in nanoseconds. After, it launches the full mesh model to load all the measures of the CSV files.

Finally, the simulation stats appear at the end of the process (Figure 4-5):

```
ban-propag] Pathloss (shadowing) (dB) : 20.633161.
ban-propag] Fading (dB) : -2.396227.
ban-propag] P_rx (dBm) : -23.029388.
ban-propag] Time called : 49.000000 s (propagation).
ban-propag] Fading envelope r.v. : 0.683211.
ban-propag] Source : 8
ban-propag] Destination : 10
ban-propag] P_tx (dBm) : 0.000000.
ban-propag] Pathloss (shadowing) (dB) : 33.554896.
ban-propag] Fading (dB) : -3.308901.
ban-propag] P_rx (dBm) : -36.863797.
ban-propag] Time called : 49.000000 s (propagation).
ban-propag] Fading envelope r.v. : 0.778638.
ban-propag] Source : 8
ban-propag] Destination : 3
ban-propag] P_tx (dBm) : 0.000000.
ban-propag] Pathloss (shadowing) (dB) : 29.109701.
ban-propag] Fading (dB) : -2.173284.
ban-propag] P_rx (dBm) : -31.282985.

simulation stats:
simulated time: 50000000000
simulation time: 232014000
speedup: 215.504237
events in queue: 1
events executed: 19539
ban-propag] Destroying the propagation framework!
ban-propag] Freeing tables (init tables).
rturo@arturo-OptiPlex-790:~/wsnet$ cd user models
```

You should find this message to show that the simulation has finished successfully.

Figure 4-5 : Event arriving and evolution of the state of links

5. PYLERS-WSNET INTERFACING

On the first chapters, we describe PyLayers and WSNet as the principal tools for the cross-layer simulator in CORMORAN. From one side, PyLayers is interesting for the capacity to simulate the IR-UWB channel impulse response and the mobility of agents, but also the implementation of human mobility. On the other hand, WSNET let us customize the upper layers and create large scale simulations, which are important features for CORMORAN.

However, the interfacing of these simulators must be well planned. For this, we described how to implement a full mesh measurements data base on WSNet. In this chapter, we are going to propose different solutions for the interfacing of WSNet and PyLayers.

5.1. CO-SIMULATION FRAMEWORKS FOR WIRELESS SENSOR NETWORKS

In order to simulate the network and physical systems, several simulators have been proposed. Network simulation tools, such as NS-2 [22], OPNET [23], OMNeT++ [24], WSNet [3], etc., support wireless modeling, a variety of communication protocols, and large-scale networks with up to thousands of wireless nodes. However, WSN simulators deal only with discrete-time models and they are not dedicated to the simulation of systems combining continuous and discrete dynamics. For that, they need to be co-simulated with physical systems simulators such as Matlab/Simulink, Modelica and SystemC. However, most co-simulators are not well adapted to evaluate protocols and algorithms for BANs or to the specifications of scenarios in CORMORAN.

COSMO [25] is a Simulation framework based on MATLAB and OMNeT++ for indoor wireless networks. COSMO provides accurate indoor channel models and 3-D models, and has the ability of self-validation. Channel models developed in Matlab are compiled directly into shared libraries and integrated into OMNeT++.

Piccsim [26] offers a simulation platform for wireless networked control systems (WiNCS), where both the wired or wireless network and the control system are co-simulated. In this framework, Simulink (used for system design and simulation) and NS-2 (for network simulation) are run on different computers and coordinated by TCP/UDP packets using the Piccsim Toolchain.

The OPNET-SIMULINK co-simulation platform [27] combines OPNET and Matlab to study the effect of node movement on WNCS. Similarly to Piccsim, this platform requires two computers, the first running OPNET as the master simulator and the second running Matlab. Both simulators are executed in parallel and are synchronized using UDP sockets.

In all of these co-simulation platforms, much development effort is needed to implement the synchronization mechanism between the two computers. HarvWSNet [28] was proposed by CEA-Leti/UR1-INRIA to provide adequate tools for the simulation of the network protocols

and the lifetime of energy harvesting wireless sensor networks (EH-WSN). It employs a simple synchronization process between WSNet and Matlab in one computer. In this co-simulator, the simple battery model of WSNet is replaced by the energy harvesting system implemented in MATLAB. Both MATLAB and WSNet are run interactively and exchange data using TCP/IP sockets (Figure 5-1).

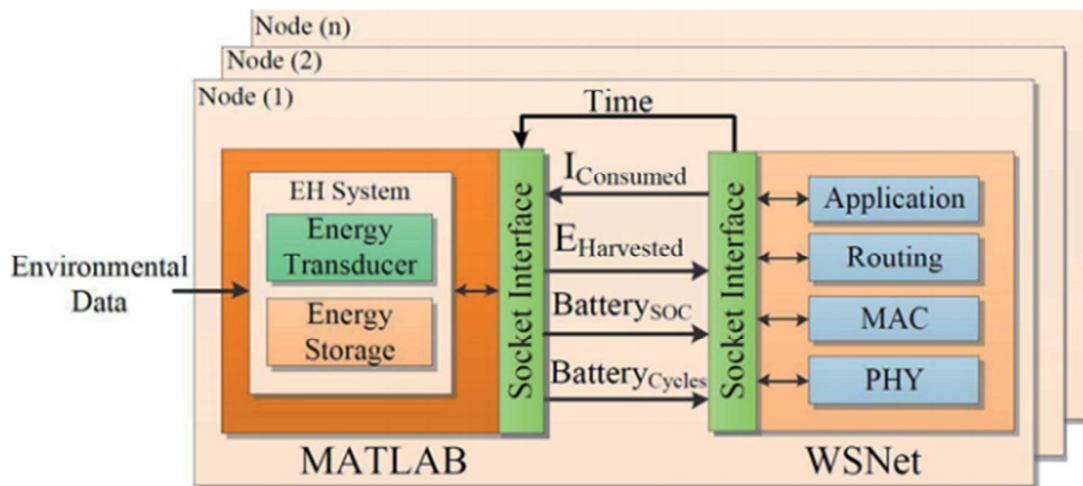


Figure 5-1 : The structure of HarvWSNET taken from [10]

In the context of the CORMORAN project, it is desirable to build a dedicated inter-WBAN multi-channel simulator with the following features:

- IR-UWB RAT capabilities
- Realistic short-term and long-term pedestrian mobility models
- On-body and off-body radio simulation capabilities
- Graph-oriented Ray Tracing tool for calculus acceleration and incremental prediction capabilities
- Multi-link and multi-RAT simulation capabilities required within cooperative scenarios

For this purpose, we propose to create a simulation tool with an interfacing between WSNet and PyLayers. PyLayers should provide the information of the quality channel links associated with network nodes. Then, the data should be processed by WSNet for the upper layers and followed by calculating the performance of the simulated scenario.

5.2. ARCHITECTURE OF THE CO-SIMULATOR FRAMEWORK

For the architecture of the simulator platform two approaches are proposed: simulator based on a database or real-time data (Figure 5-2).

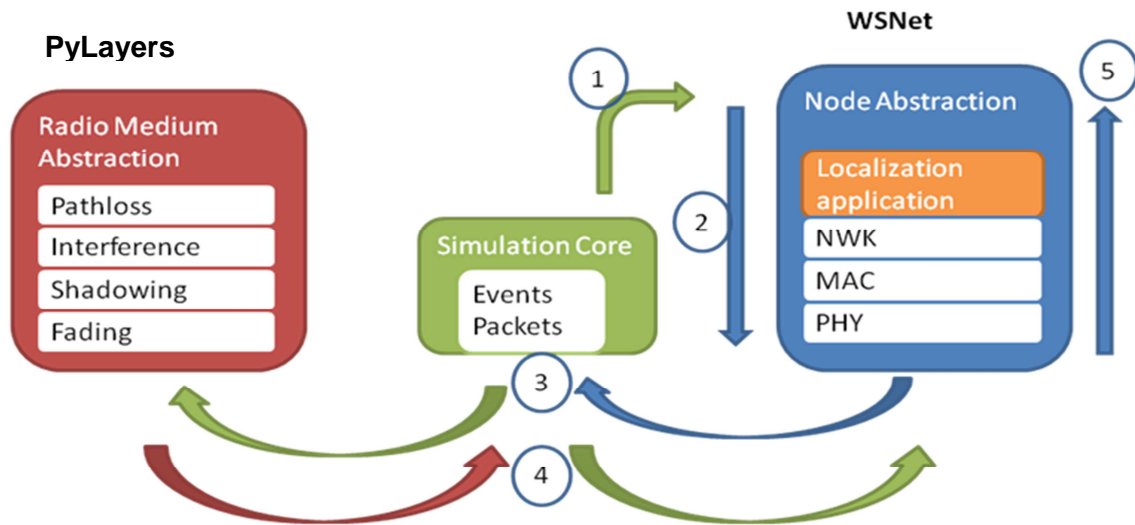


Figure 5-2 : WSNet and PyLayers co-simulation architecture

5.2.1 DATABASE SOLUTION

In this alternative, there is a database that stocks the results of the UR1 PyLayers deterministic simulator. The data is specially the quality of channel links and it is used after by WSNet for the upper layers overall network timeline.

The points to consider in this solution concern the PHY layer on WSNet:

- The PHY layer can be created by new models (this means to code a PHY layer for WSNet and this take some time).
- We use xml files where one would explicitly write traces (csv files) containing the radio metrics based on the mobility and results of PyLayers (apparently easier solution).
- Mobility can be supported by WSNet but it has to be synchronized with the mobility of PyLayers nodes.

Another problem for this solution resides in **choice of the master/slave mode of the simulation** for the layers management. It's logical that the upper layers (MAC/NWK) have to be the master to launch the PHY layer as slave. From this approach, WSNet will be the master simulator to launch the simulation and there is a scheduler that will manage the time to launch each process, including the PyLayers calculation. However, we have to consider the calculation time and the sampling method (Time Stamp, Node Id, X, Y, Z, Link Quality Indicator, trajectories calculation).

So one solution is to take PyLayers as slave of WSNNet (Figure 5-3):

- To reduce the generation of several xml files
- Mobility can be treated separately (e.g. through multi-agent model of UR1 then processed via database with respect to PyLayers)
- Physical time simulation by request is still an open question to validate the feasibility of this architecture

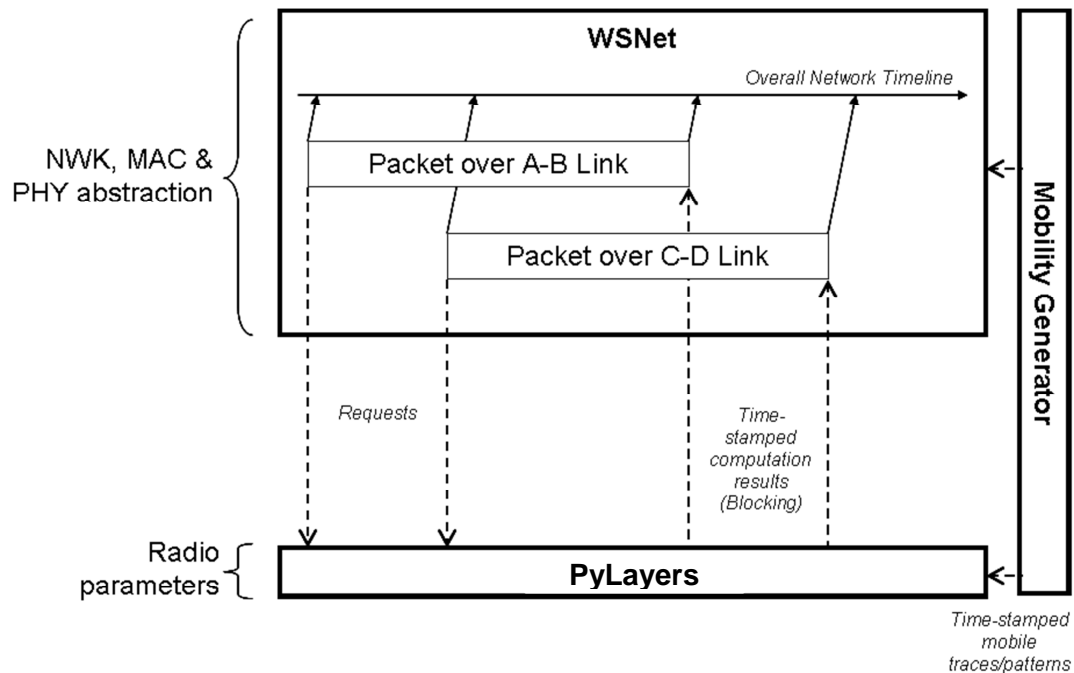


Figure 5-3 : master/slave option for WSNET and PyLayers

However, one last point to consider is the access to the data base at the same time. In fact, we can find some errors if WSNet try to recover data when PyLayers is writing on the data base. For that, we need to define a good synchronisation between WSNet and PyLayers to avoid having access at the same time.

5.2.2 REAL TIME SOLUTION

The idea is to simulate on real time the PyLayers mobility feature with statistical radio channel models (PHY layer) and WSNet (upper layers)

For this solution, the communication of simulators has to be considered:

- Unidirectional: WSNET → PyLayers, in this solution WSNet is the master that calls PyLayers functions when necessary.
- Bidirectional: WSNET ↔ PyLayers, both simulators can be used separately and they can be master or slave depending on the simulated scenario.

This solution is more complex to develop, that's why we should test the first solution in short-term. The first step is to test the interface of the multi-agent mobility model UR1 for off-body radio & body-to-body to generate the first useful lower-level simulation for the group navigation scenario and cooperative inter-WBAN communication.

We can also consider implementing a co-work flow as it was made for HarvWSNet [28], both PyLayers and WSNnet should run interactively and exchange data with sockets. WSNnet may be responsible for clock synchronization and simulation time handling. In this case, we can benefit from the WSNnet event-drive nature. WSNnet invokes PyLayers at each synchronization event, scheduled at fixed intervals of time and dedicated to maintaining the synchronization of the simulation clock and nodes mobility between WSNnet and PyLayers, as well as at each communication event, which arise when a node is in different states, such as transmitting, receiving, listening or sleeping Figure 5-4.

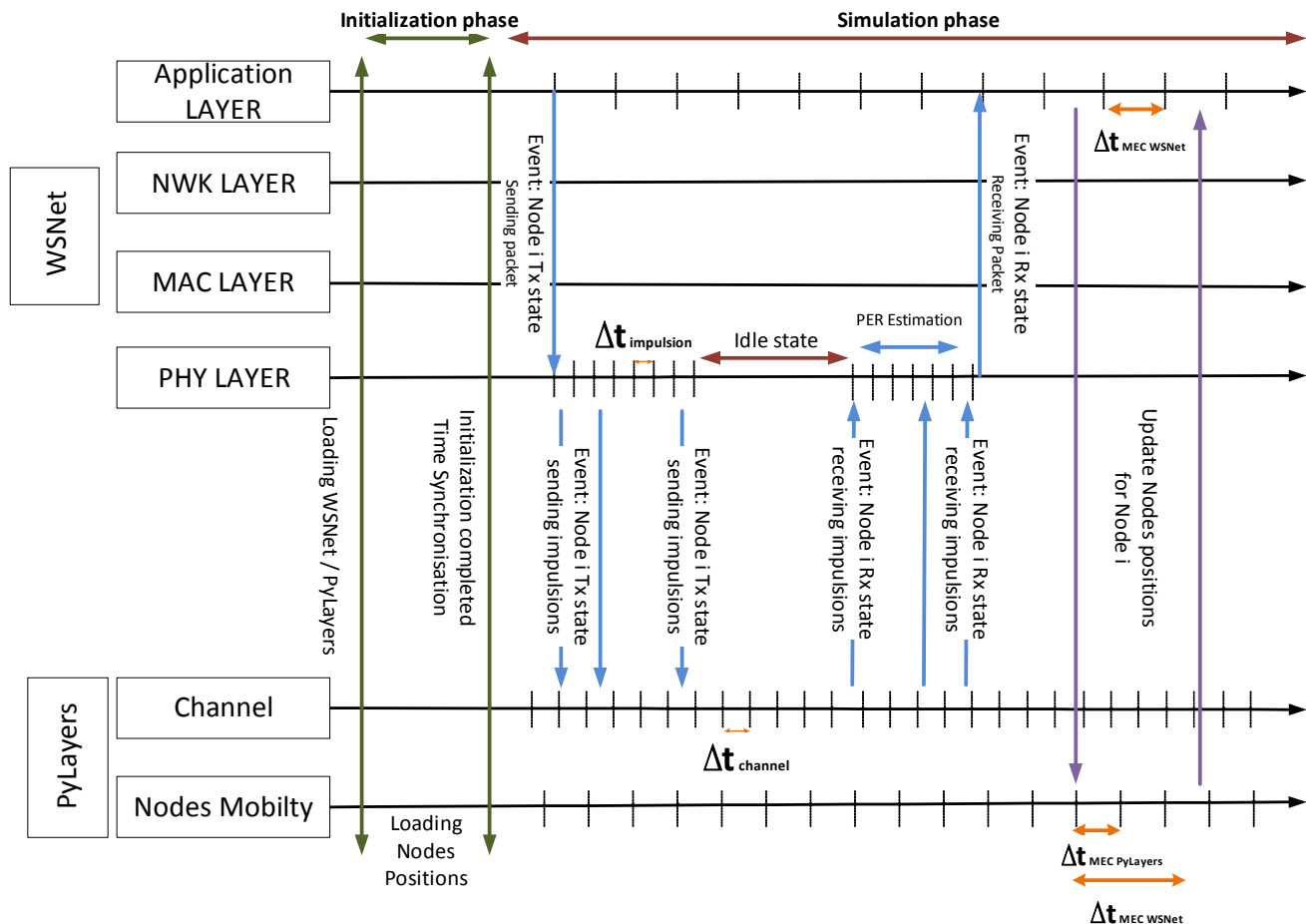


Figure 5-4 : Co-simulation of WSNnet and PyLayers

As shown on the figure, we find two phases: an initialization phase and a simulation phase. In the first phase, WSNnet and PyLayers are loaded; the starting nodes positions are settled in PyLayers and finally, the timers of both simulators are synchronized. In the second phase,

we start the simulation of a BAN scenario where the application layer is the one who start the communication. On the figure, we see a node i sending a single packet to estimate its position. First, the node changes its state into TX mode to send a packet, and then the PHY layer sends impulsions to PyLayers (i.e. with sockets). After that, PyLayers calculates the channel impulse response and send them to WSN_{Net}. When the node receives the packet correctly, it launches the localization logarithm at the application layer. The positions are estimated with the estimation of the time-of-flight but the estimated values can be also compared by updating nodes positions with PyLayers (i.e. with sockets), thus we can estimate the localization error rate.

However, this example is not the one that represents a real BAN scenario. In fact, PyLayers returns the channel impulse response to the nodes that listen the node that request its position. These nodes change their state into RX mode to receive a packet, thus they estimate the channel variations and the packet error rate. After that, they will answer to the node. Then, we can also consider that the nodes could consider helping in the communication as relays and so, we can evaluate the cooperative and localization algorithms.

Nevertheless, the big challenge of this co-simulation is the time management. As shown on the figure, PyLayers and WSN_{Net} must synchronize their timers to maintain the logic of the communication. This is not easy to accomplish because there is an abstraction of the time at the different layers. On the figure, there are represented four different times:

- $\Delta t_{\text{MEC WSN}_{\text{Net}}}$ represents the time of the application layer. In other words, it is the time starting when a node sends a position request and it finish when the node calculates its position with the TOF and clock drift of all the answers.
- $\Delta t_{\text{impulsion}}$ represents the time of sending impulsions by the PHY layer abstraction on WSN_{Net}. This layer represents the union between WSN_{Net} and PyLayers and it will be discussed on the next subchapter. This time should be in the order of nanoseconds.
- $\Delta t_{\text{channel}}$ represents the timestamps of channel time variation simulated by PyLayers. Thus, the impulsions sent by WSN_{Net} will be affected by the channel state at a specific $\Delta t_{\text{channel}}$.
- $\Delta t_{\text{MEC PyLayers}}$ represents the time of the nodes mobility. This can be calculated by both PyLayers and WSN_{Net}, but we supposed that PyLayers should manage the body motion. It is important to know that this time is different to $\Delta t_{\text{MEC WSN}_{\text{Net}}}$, but they have a relation because the time of application layer also represents the resolution of the localization algorithm that must be adapted to the body movement. For example, if a person is running, the hands can move from the front to the back in $\sim 250\text{ms}$, therefore the application layer has to adapt the calculation of positions in a time lower than 250ms. So we can imagine an application layer adapting the resolution and the refreshment of position requests depending to the body movement.

5.3. WSNET / PYLAYERS PHY LAYER INTERFACING

As described on the deliverable D2.4 Chapter 10, PyLayers will provide the information of the quality channel links (LQI) and then, WSNet will process the data with the upper layers for calculating the performance of the simulated scenario. However, one important piece of the simulation remains on the PHY layer. In fact, neither PyLayers nor WSNet developed the IR-UWB PHY layer. So, one of the question is which simulator should create this layer (. However, we have to consider the complexity for implementing a PHY Layer in terms of simulation time. As WSNet was conceived to simulate networks for upper layers at the packet level for quick simulations, a whole implementation of the PHY layer will increase the time of simulation which makes less interesting the simulation of WSNET.

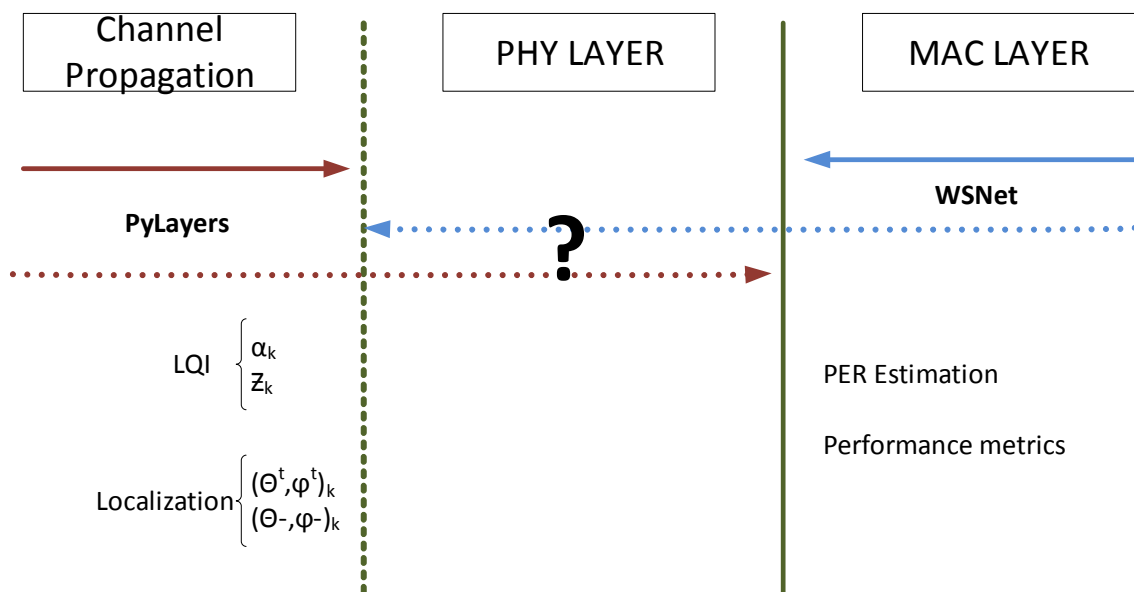


Figure 5-5 : PHY Layer PyLayers - WSNet interfacing

As PyLayers does not have a cross-layer approach, we suggest implementing the PHY Layer with WSNet. For that, we can imagine two ways of development: full PHY layer for the transceiver and receiver and an abstraction of the PHY layer at the receiver.

5.3.1 FULL PHY LAYER IMPLEMENTATION

The first is the implementation of a **full PHY Layer for the transceiver and receiver**. As explained on the D2.4 Chapter 10.1.2 PHY layer inputs, we have to consider creating modules in WSNet to simulate the code repetition coder (e.g. BCH encoder), the modulator

(e.g. OOK, DPSK as defined in the standard IEEE802.15.6), the pulse shaper and the coherent or not coherent receiver (Figure 5-6). For this task, we can inspire from some MATLAB models proposed in literature [29] [30] [31] [32].

One advantage of this proposal is that we can eventually study the performance of cooperative strategies at PHY and MAC level. For example, we can imagine comparing the performance of a specific MAC for PHY cooperative algorithms e.g. Amplify and Forward (AF), Decode and Forward (DF) or Compress and Forward (CF). Or study the impact of MAC/PHY cooperative strategies for the localization algorithms proposed for CORMORAN.

However, the calculation for a whole transmission chain will be more complex making WSNET to take more time for simulations. As PyLayers is capable to simulate IR UWB Ray Tracing, it is possible to get the Channel Impulse Response (CIR) which could be the only input needed to have a simulation for upper layers in WSNET.

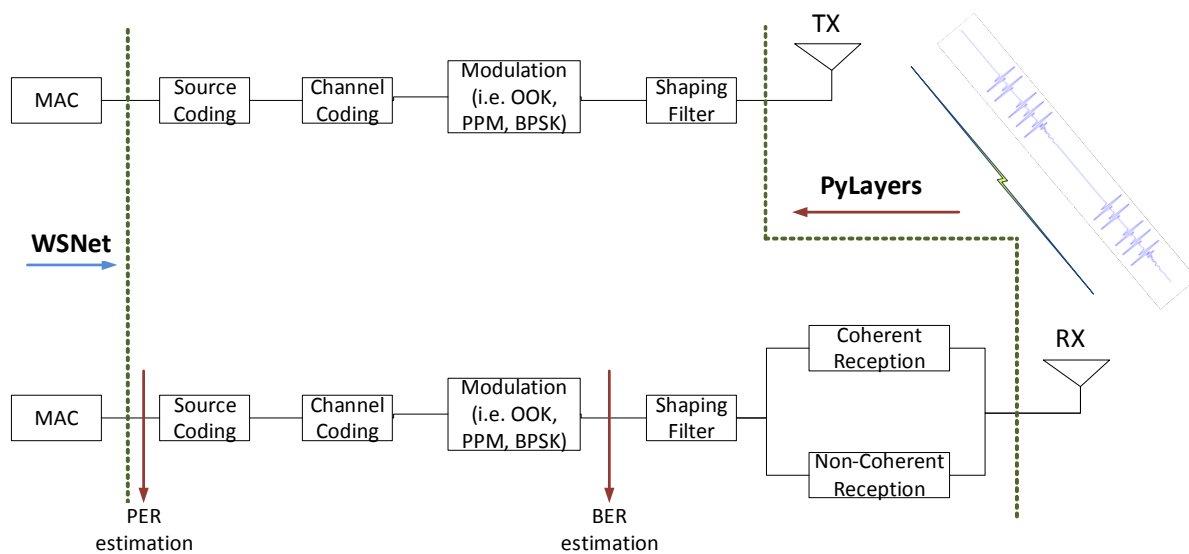


Figure 5-6 UWB transmission scheme

5.3.2 PHY LAYER ABSTRACTION AT RECEPTION

To simplify the simulation of the PHY layer, it is possible to get the performance values without simulating the whole transceiver chain transmission. The second idea is to implement **an abstraction of the PHY Layer at the receiver**, this can be seen as a WSNET module capable to give us the Packet Error Rate (PER) estimation with statistical models that use the LQI information of PyLayers. For that, a proposition for the PHY abstraction is a non-coherent receiver that detects the energy of the CIR at the moment that a packet may be sent, thus, we will be able to estimate the PER when a packet event is launched by WSNET. The

model will detect the CIR and with a hard decision we can estimate if a packet is loss or not [33] [34] [35].

Firstly, we can write a typical CIR in UWB like this:

$$h(t_n, \tau) = \sum_{k=1}^{k(t_n)} \alpha_k(t_n) * e^{j\theta_k(t_n)} * \delta(\tau - \tau_k(t_n))$$

where $k(t_n)$ is the total number of multipath, α_k , θ_k and τ_k are the amplitude, the phase and the delay of the k^{th} path.

Then, we suppose that we have perfect synchronisation and non-coherent receiver. Thus, we can split the modulation in two cases, mono-path demodulation ($T_{ED} \approx T_w$) and multi-path demodulation ($T_{ED} \gg T_w$), with T_{ED} as the integrator time of the energy detection and T_w as the pulse time defined by the standard IEEE802.15.6. This depends also on the modulation technique, in our case we will focus on OOK. Moreover, we can also split the cases where we have or not Time hop sequences and finally the case where the packets are or not coded by the BCH coder.

The general PHY receiver chain is composed by a non-coherent receiver employing a squaring device and an energy integrator as shown in (Figure 5-7). Then, a decision system is used in order to detect the moment where an impulse is received with a decision threshold. Thus the probability error is calculated for a whole packet.

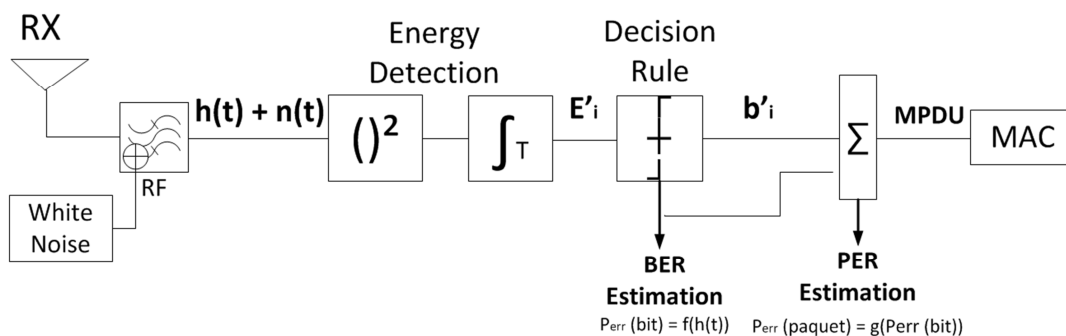


Figure 5-7 UWB PHY Layer abstraction

Mono-path OOK receiver case

The first approach to study is the case of mono-path demodulation with no time hop sequence and no BCH coding. In this case, the duration of the integration window is written as $T_{ED} \approx T_w$, which means that is approximately the same duration of a pulse waveform. However the pulse waveform duration depends on the data rate of the system, in our study,

we consider the default data rate (0.4875 Mbps, $T_w = 2054.3$ ns) defined in the standard IEEE802.15.6 (Figure 5-8).

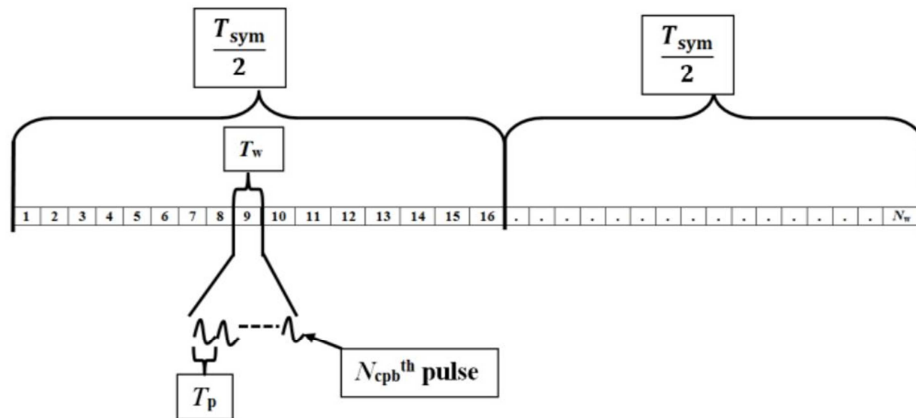


Figure 5-8 UWB PHY Symbol OOK 802.15.6

Statistical Hypothesis Testing. Considering the modulation of the standard IEEE 802.15.6 as explained in [36], the OOK is mixed with a group-PPM. So a bit is coded with a symbol that is divided into two halves. Moreover, we have 16 possible burst hopping positions within each half in the default mode and each half represents a bit of a code word. Therefore, a data bit [0] = [1 0] and the data bit [1] = [0 1], which means that when we send a zero bit, there is a pulse in the first half and no pulse in the second half; therefore, when there is an impulse in the second half and no impulse in the first half, a one bit is transmitted.

Then, the non-coherent receiver passes the received signal through a bass-pass filter for noise reduction. Thus, the decision variable x of the ED receiver is expressed with two hypotheses:

$$\begin{cases} H_0 : x = \int_0^{T_{ED}} [n(t)]^2 dt & \text{bit 0} \\ H_1 : x = \int_0^{T_{ED}} [r(t) + n(t)]^2 dt & \text{bit 1} \end{cases}$$

where $r(t) = \alpha p(t - \tau) + n(t)$ is the received signal for and emitted pulse with waveform $p(t)$. At this stage, we try to decide "at best" between H_0 and H_1 , intending to minimize the error probability. For this aim, we define the Probability Density Function (PDF) P_{x0} (resp. P_{x1}) under the hypotheses H_0 (resp. H_1) as a central (resp. non central) Chi-square distribution (χ^2) [37]:

$$\left\{ \begin{array}{l} H_0 : \\ H_1 : \end{array} \right. \quad P_{x0}(x) = \frac{1}{N_0} \frac{(x/N_0)^{M-1} e^{-x/N_0}}{\Gamma(M)}$$

$$P_{x1}(x) = \frac{1}{N_0} \left(\frac{x}{E}\right)^{\frac{M-1}{2}} e^{-\frac{x+E}{N_0}} I_{M-1}\left(2\sqrt{\frac{x}{N_0}E}\right)$$

where $2M = 2 B * T_{ED} + 1$ represents the degrees of freedom of these distributions (Shannon Sampling Theorem); B is the signal Bandwidth at the T_{ED} ; E is the estimation of the signal energy $E = \int_0^{T_{ED}} [r(t)]^2 dt$; $I_n()$ is the Bessel function of the first kind as $I_{M-1}(u) = K u^{-1/2} e^u$ for a fixed M and positive K; and $\Gamma(M)$ is the Euler function as $\Gamma(p) = \int_0^\infty t^{p-1} e^{-t} dt, p > 0$.

Then, we calculate the optimal decision rule with a threshold (ρ_{opt}) resolving the equation $P_{x0}(\rho_{opt}) = P_{x1}(\rho_{opt})$, Figure 5-9. In [38], we can find an approximation of ρ_{opt} for E/No between 0 and 20 dB: $\rho_{opt} = \frac{E}{4N_0} + M + \sqrt{M-1} \varphi\left(\frac{E}{N_0}\right)$, where $\varphi(x)$ is a third degree polynomial.

Moreover, we denote the pulse error probability as $PEP = \frac{1}{2}(P_\rho(1|0) + P_\rho(0|1))$, for an optimal threshold $\rho_{opt} = \varepsilon$, where $P_\rho(1|0)$ is the probability of deciding H_1 whereas no signal was sent and $P_\rho(0|1)$ is the probability of deciding H_0 by missing the pulse.

$$\left\{ \begin{array}{l} P_\rho(1|0) = e^{-\frac{\varepsilon}{N_0}} \sum_{k=1}^M \frac{\left(\frac{\varepsilon}{N_0}\right)^{M-k}}{\Gamma(M-k+1)} \\ P_\rho(0|1) = 1 - Q_M\left(\sqrt{\frac{2E}{N_0}}, \sqrt{\frac{2\varepsilon}{N_0}}\right) \end{array} \right.$$

where Q is the generalized Marcum function: $Q_m(a, b) = \frac{1}{a^{m-1}} \int_b^\infty x^m e^{-\frac{x^2+a^2}{2}} I_{M-1}(ax) dx$

Therefore, the pulse error probability can be expressed as a function of the received energy E/2 as:

$$P_e = \frac{1}{2} - \frac{1}{2} Q_M\left(\sqrt{\frac{2E}{N_0}}, \sqrt{\frac{2\varepsilon}{N_0}}\right) + \frac{e^{-\frac{\varepsilon}{N_0}}}{2} \sum_{k=1}^M \frac{\left(\frac{\varepsilon}{N_0}\right)^{M-k}}{\Gamma(M-k+1)}$$

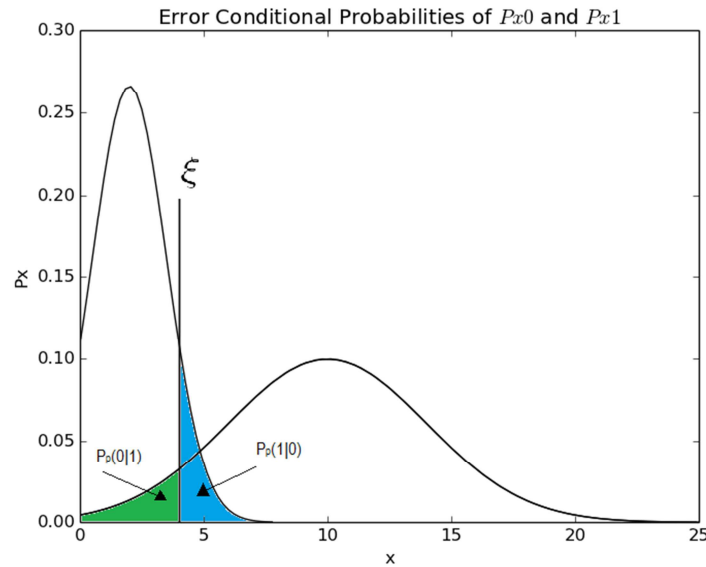


Figure 5-9 Probability Density Function under the hypothesis $(H_i)_{i \in \{0,1\}}$

However, as a OOK symbol is represented by many burst positions (Figure 5-8), the Bit Error Rate of the n th symbol can be seen as the comparison of the detected energy at the m th burst ($G_n^{(m)}$) with the optimal threshold (OOK approach) or as the comparison of the two energies detected at the m th burst between the two halves (PPM approach):

$$\begin{cases} BER_{OOK} = f(G_n^{(m)} \underset{1}{\leq} \varepsilon) \\ BER_{PPM} = f(G_0^{(m)} \underset{1}{\geq} G_1^{(m)}) \end{cases}$$

Thus, the Packet Error Rate without BCH code of a single transmission for a packet of q bits is computed as

$$PER_{CRC}(q) = 1 - (1 - BER)^q$$

In the case of a transmission with BCH block codes we can calculate the PER in function of the Block Error Rate (BLER) given by

$$BLER(n, k, t) = \sum_{i=t+1}^n \binom{n}{i} BER^i (1 - BER)^{n-i}$$

where n is the block length, k is the payload length and t is the error correcting capability in bits. Finally, the PER is given by

$$PER_{BCH}(q, n, k, t) = 1 - (1 - BLER(n, k, t))^{\lfloor \frac{q}{n} \rfloor}$$

Multi-path OOK receiver case

In the literature we find that the biggest challenge for the non-coherent receptor is the detection of multipath and the duration of integration interval T_{ED} [39]. There are many works [40] [41] [38] [42] [43] [44] [45] studying the non-coherent receptors and the performance of energy detection with fixed integration intervals T_{ED} . Therefore, there is a trend [33] [37] [46] [47] [48] to create a fusion of different integration intervals to consider the principal paths of the channel for the decision rule of the Hypotheses Test.

As for the mono-path case, we need to detect the energy of pulses before the hypothesis testing, however delay spread of multipath can reach values from 30 to 200 ns for distances varying from 1 to 10 meters [49] [50]. The main paths can be up to 60 to take 85% of pulse energy, that's why the integrator time T_{ED} should be higher than the delay spread. Therefore, the integrator time has to be higher than the burst time ($T_{ED} \gg T_w$) for any data rate of the IR UWB symbol. Thus, we apply the hypothesis test as for the mono path but considering the multipath at the received signal $r(t)$.

$$\begin{cases} H_0 : x = \int_0^{T_{ED}} [n(t)]^2 dt & \text{bit 0} \\ H_1 : x = \int_0^{T_{ED}} [r(t) + n(t)]^2 dt & \text{bit 1} \end{cases}$$

where $T_{ED} \gg T_w$ and $r(t) = \sum_k \alpha_k p(t - \tau_k) + n(t)$, k indexes the paths of the channel.

Then we apply the same process to calculate the pulse error rate. However, as the multipath delays can be bigger than a burst time or can interfere between the halves of the symbol, the BER_{OOK} could be not the best way to detect the bit error rate, so it is better to use the BER_{PPM} approach as a fusion of multiple ED for the principal paths of the channel (having an energy higher than the optimal threshold $\rho_{opt} = \varepsilon$), as shown in the Figure 5-10. This approach is similar to the multipath receptor architecture presented in [37] but adding the comparison of the two halves.

$$BER_{PPM} = f \left(G_0^{(m)} \underset{1}{\gtrless} G_1^{(m)} \right)$$

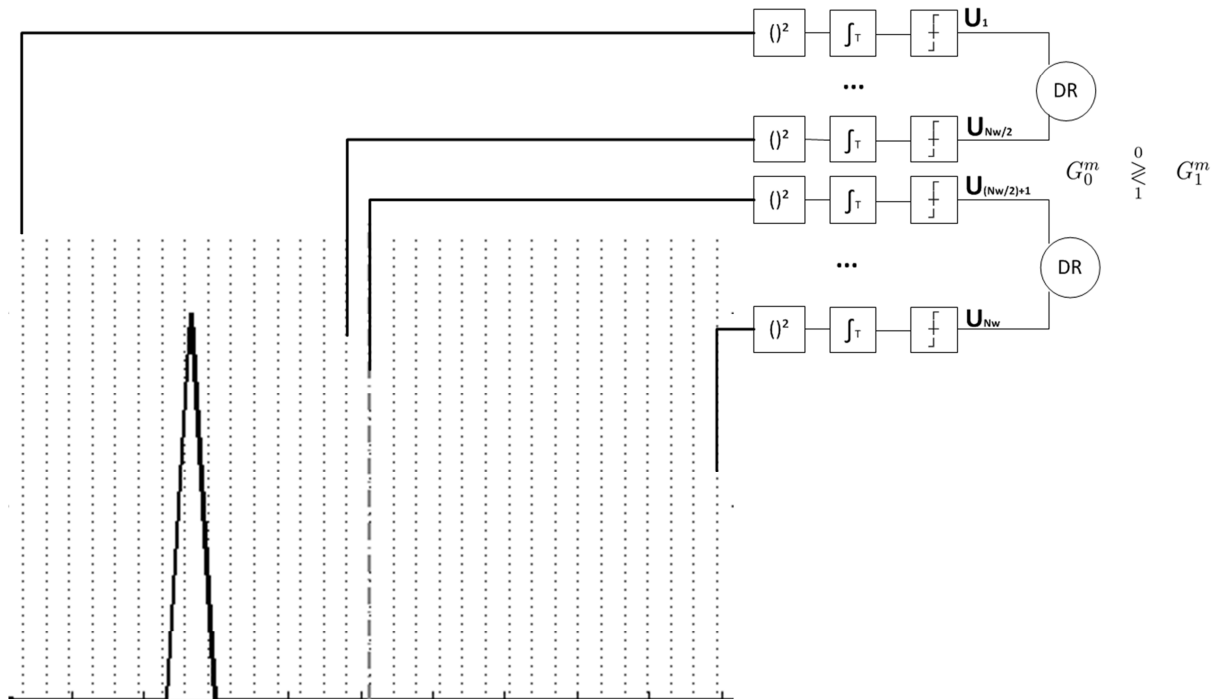


Figure 5-10 Bit Error Probability Decision Rule

5.4. DATA DESIGN

A final step is to properly define the data exchanged between PyLayers and WSNets. For that four points are presented to show the different challenges for the design of data exchanged in the co-simulator.

5.4.1 DATA FORMAT

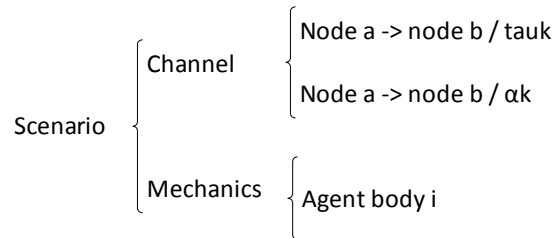
a. Delay and power:

As explained on chapter 2 and 4, PyLayers provides 2 CIR to WSNets: the **delays** (τ_k) represented in **nanoseconds** (ns) and the associated **power value** (α_k) represented in **decibels** (dBm). The samples are stored in two different **link CSV files** at a **constant timestamp** which is defined in the **description CSV file**, as showed on the chapter 4.1. Moreover, we have to consider that PyLayers has a maximum number of contributing rays (and associated delays and power) that is arbitrary limited to 100.

Storage requirements: 6 bytes are needed for each τ_k and 6 bytes for a power value. Then, as we have 100 contributing rays per link, we need 1.2 kilobytes per link, 600 bytes for each CSV file.

Storage organization: for this, we will need a repertory named scenario representing and specific simulation such as an on-body running, walking, etc. In this repertory, we will

define a channel repertory containing the links CSV files and a mechanics repertory containing the agent body description file:



For example, the file with the agent links description with n nodes is stored in the mechanics repertory and it may contain:

```

0,1,"scenario/channel/node0-node1_tauk.csv",timestamp,simulation_time
0,1,"scenario/channel/node0-node1_ak.csv",timestamp,simulation_time
...
0,n,"scenario/channel/node0-noden_tauk.csv",timestamp,simulation_time
0,n,"scenario/channel/node0-noden_ak.csv",timestamp,simulation_time
...
n,n-1,"scenario/channel/noden-noden-1_tauk.csv",timestamp,simulation_time
n,n-1,"scenario/channel/noden-noden-1_ak.csv",timestamp,simulation_time
    
```

After that, each link file may contain the 100 measures of tauk or ak. For example:

```

node0-node1_tauk.csv:      zk_0, zk_1, zk_2, ..., zk_99
node0-node1_ak.csv:       alphak_0, alphak_1, alphak_2, ..., alphak_99
    
```

Furthermore, we can also consider a file description containing the interference agents file description to define the interfering nodes of a BAN and the respective links description.

b. Position:

As explained on chapter 2.2, PyLayers simulates the **center of mass position** of the agents. Moreover, information about velocity vector is also provided. These body scale mobility is described on **C3D files** which contains the position of all antennas on a body in a local axis.

For this purpose two task are needed:

- First, we need to adjust the axis of C3D file to those of PyLayers;
- And secondly, we have to adjust the loop on body movement and the timestamps.

c. Time to consider:

As described on the chapter 5.2, a delta time mechanic is necessary to represent the time of the nodes mobility. As explained in section 2.3 the body motion has a delta mechanic of 1/10

Hz = 1 second. Then, we supposed that each scenario is simulated over 60 seconds. Finally, we have a simulation of **600 mechanical timestamps**.

$$\Delta t_{\text{MEC PyLayers}} = 1/10 \text{ Hz} = 0.1 \text{ s}$$

$$\text{Simulation time} = 60 \text{ s}$$

5.4.2 COMPUTATION COMPLEXITY EVALUATION

In the CORMORAN project, we want to simulate the coexistence of BANs for different scenarios, and for that we need to evaluate the computation complexity required. For this, we consider two cases: the first case considers the data recollected in the deliverable Draft 1.1 chapter 2.3 and the second is based with the CEA application scenarios.

a. Application needs case (worst case):

In this case, the worst scenario corresponds to the Coordinated Group Navigation where we can find a maximum of 10 persons per group with a maximum of 5 on-body nodes per person. As all links have to be simulated, we find $(10*5)*49 / 2 = 1225$ radio links per timestamp.

b. CEA application case:

In this case, one body is equipped with 12 antennas and 2 off-body anchors. As all links have to be simulated, we find $(12+2) * 13 / 2 = 91$ radio links per timestamp.

5.4.3 COMPUTATION STORAGE REQUIREMENT

In this subchapter, we take into account all the last values to calculate the final storage needed for the simulation. This storage can be calculated by considering the mechanical timestamps, the total number of links and the file storage.

- a. **Worst case:** 600 timestamps * 1225 links * 1,2 KB = **882 Megabytes**
- b. **CEA case:** 600 timestamps * 91 links * 1,2 KB = **65 Megabytes**

5.4.4 COMPUTATION SPEED EVALUATION

Finally, it is also needed to estimate the computation speed for a whole simulation. For this purpose, we have to consider the mechanical timestamps, the total number of links and the time for a single link

- a. **Worst case with 1s/link:** 600 timestamps * 1225 links * 1 s = 735000 s => **8.5 days**
- b. **Worst case with 0.1s/link:** 600 timestamps * 1225 links * 0.1 s = 73500 s => **20.5 h**
- c. **CEA case with 1s/link:** 600 timestamps * 91 links * 1 s = 54600 s => **15 h**
- d. **CEA case with 0.1s/link:** 600 timestamps * 91 links * 0.1 s = 5460 s => **1.5 h**

6. CONCLUSIONS

This document is a description of the first mobility enabled physical simulator and coupling with packet oriented simulator. All the main key features to get into this objective are presented with a sufficient level of details for building up. In particular this document covers the description of the two piece of software used for this purpose: PyLayers and WSNET as well as the modalities foreseen for their interfacing.

Let as a conclusion recalls how the whole simulation workflow is going to be implemented.

- 1- The whole Indoor scene including the floor plan description and the several moving agent involved are described in ad-hoc human readable configuration files
- 2- Each agent is equipped with a set of radio devices, each being carefully localized and oriented on the body abstraction.
- 3- The body abstraction is animated with motion capture files.
- 4- The ray tracing is calculated using the graph derived abstraction of signatures which allows important time saving when addressing moving channel simulation.
- 5- The channel radio observables are available including received power for each path as well as delay an angular values for all multi-channel. The antenna being included in an original and efficient manner.
- 6- The time stamped output of Pylayers feed a PHY layer abstraction block which provides 2 key quantities (for the CORMORAN algorithms assessment), the BER (or PER) and the TOA bias.
- 7- The MAC and NET, APP (localization block) solution and strategies are assessed under :
 - a. Realistic LDPs
 - b. With ground truth for all devices (which is key and the motivation of the deterministic initial choice)

The CORMORAN simulation endeavor is going smoothly and strengthens both tools capabilities. The further step which is ongoing is to actually present results obtained from the synergy of those two tools. This is going to be presented in the next deliverable about the CORMORAN simulator.

7. BIBLIOGRAPHY

- [1] B. Denis et al., "D1.1 Application Scenarios, System Requirements and Prior Models," Deliverable 2012.
- [2] N. Amiot B. Uguen. PyLayers. [Online]. <http://www.pylayers.org>
- [3] A. Fraboulet and E. Ben Hamida (s.d.) G. Chelius. (2012) consulted 2012 on WSNet / Worldsens simulator. [Online]. <http://wsnet.gforge.inria.fr/index.html>
- [4] Raffaele d'Errico, Meriem Mhedhbi, Bernard Uguen Oudomsack Pierre Pasquero, "D2.2 - 1st Channel Measurement campaign and," CORMORAN (ANR 11-INFR-010), Deliverable 2013.
- [5] OSM. (2012, Sep.) OpenStreetMap. [Online]. <http://www.openstreetmap.org>
- [6] R. Diestel, *Graph Theory*, Springer-Verlag, Ed.: Electronic library of mathematics, 2010.
- [7] E.W. Dijkstra, *A Short Introduction to the Art of Programming.*: Holland, 1971.
- [8] Craig W Reynolds, "Steering behaviors for autonomous characters," in *Game Developers Conference*, vol. 1999, 1999, pp. 763-782.
- [9] L. Euler, *Introductio in analysin infinitorum*, apud Marcum-Michaellem Bousquet & socios, Ed., 1748.
- [10] E.T. Hall, *The Hidden Dimension.*: Doubleday, 1966. [Online]. <http://books.google.fr/books?id=VtdOAAAAMAAJ>
- [11] Teknomo, "Microscopic Pedestrian Simulation Model to Evaluate Lane-like Segregation of Pedestrian Crossing," in *Proceedings of Infrastructure Planning Conference.*, Kouchi, Japan, 2001.
- [12] Kardi Teknomo, "Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model," Tohoku University, Japan, Sendai, Ph.D. dissertation 2002.
- [13] T. and Minseok Kim and Takada, J. Iswandi and Aoyagi, "The utilization of body skeleton model for modeling the dynamic BAN channels," in *Antennas and Propagation (EUCAP), 2012 6th European Conference on*, 2012, pp. 540-543.
- [14] Chia-Chin and Tan, Chor-Min and Laurenson, D.I. and McLaughlin, S. and Beach, M.A. and Nix, A.R. Chong, "A novel wideband dynamic directional indoor channel model based on a Markov process," *Wireless Communications, IEEE Transactions on*, vol. 4, no. 4, pp. 1539--1552, 2005.
- [15] T. and Fleury, B. H. Pedersen, "Channel Modeling; diffuse scattering; diffuse scattering," in *Communications, 2007. ICC '07. IEEE International Conference on*, 2007, pp. 2733--2738.
- [16] AMIOT N., "Design of a Simulation Platform Joining Site Specific Radio Propagation and Human Mobility for Localization Applications," Université de Rennes 1, Rennes, Thèse de Doctorat 2013.
- [17] E. Ben Hamida, "Modélisation stochastique et simulation des réseaux sans fils multi-sauts," INSA de Lyon, PhD Thesis in French Sept 2009.
- [18] G. Chelius, N. Boulicault, L. Lemaître, J. Nassimian and J. Carpentier, (s.d.) A. Fraboulet.

- (2012) WSim / Worldsens simulator. [Online]. <http://wsim.gforge.inria.fr/people.html>
- [19] T. Rappaport, *Wireless Communications: Principles and Practice*, IEEE Press Piscataway, Ed. NJ, USA, USA, 1998.
- [20] R. D'Errico and L. Ouvry, "Delay dispersion of the on-body dynamic channel," in *in Proc. EuCAP'10*, Barcelona, April 2010.
- [21] Dominic John Repici. (2013) consulted 2013 on CSV format / Creativyst Inc. [Online]. <http://creativyst.com/Doc/Articles/CSV/CSV01.htm>
- [22] The Network Simulator NS-2. [Online]. <http://www.isi.edu/nsnam/ns/>
- [23] OPNET Homepage. [Online]. <http://www.opnet.com/>
- [24] OMNeT++ Homepage. [Online]. <http://www.omnetpp.org/>
- [25] Z. Lu, Q. Chen, X. Yan, and L.-R. Zheng Z. Zhang, "Cosmo: Co-simulation with matlab and omnet++ for indoor wireless networks," in *GLOBECOM*, 2010.
- [26] M. Pohjola, J. Brand, and L. M. Eriksson T. Kohtamaki, "Piccsim toolchain - design, simulation and automatic implementation of wireless networked control systems," in *in Proc. Int. Conf. Networking, Sensing and Control ICNSC*, 2009, pp. pp. 49–54.
- [27] H. Yu, A. Carrington, and T. C. Yang M. S. Hasan, "Co-simulation of wireless networked control systems over mobile ad hoc network using simulink and opnet," *IET Communications*, 2009.
- [28] C. Bernier, D. Morche, O. Sentieys A. Didioui, "HarvWSNet: A Co-Simulation Framework for En-ergy Harvesting Wireless Sensor Networks," *IEEE ICNC*, January 2013.
- [29] B. Miscopein et J. Schwoerer, "Low complexity synchronization algorithm for non-coherent UWB-IR receivers," in *Vehicular Technology Conference VTC2007-Spring. IEEE 65th*, Dublin, Apr. 2007, pp. pp. 2344–2348.
- [30] B. Miscopein and J.-M. Gorce J. Schwoerer, "Procédé d'émission d'impulsuion dans un canal de transmission," INPI 0854137, Juin 23, 2008.
- [31] J. Schwoerer, L. Fesquet et R. Renaudin J. Hamon B. Miscopein, "Self-timed implementation of an impulse radio synchronisation acquisition algorithm," in *in DASIP, Conference on Design and Architectures for Signal and Image Processing*, Belgium , November 2008.
- [32] Maria-Gabriella Di Benedetto and Guerino Giancola, *Understanding Ultra Wide Band*, Prentice Hall PTR, Ed.: Radio Fundamentals, 2004.
- [33] L.-M. Aubert, and B. Uguen S. Paquelet, "An impulse radio asynchronous transceiver for high data rates," in *International Workshop on Ultra Wideband System joint with Conference on Ultrawideband Systems and Technologies. Joint UWBST & IWUWBS. , May 2004*, pp. pp 1-5.
- [34] T. Aoyagi, and R. Kohno K. Takizawa, "Channel Modeling and Performance Evaluation on UWB-Based Wireless Body Area Networks," in *Institute of Electrical and Electronics Engineers*, Jun 2009., pp. pp. 1–5.
- [35] M. Weisenhorn and W.Hirt, "ML Receiver for Pulsed UWB Signals and Partial Channel State Information," in *Institute of Electrical and Electronics Engineers*, 2005, pp. pp.

180–185.

- [36] Matti Hamalainen, and Jari Linatti Ville Niemela, *On IEEE 802.15.6 UWB symbol length for energy detector receivers performance with OOK and PPM*, Institute of Electrical and Electronics Engineers, Ed., Mar 2013.
- [37] Benoît Miscopein, "Systèmes uwb impulsions noncohérents pour les réseaux de capteurs: coexistence et coopération.," INSA Lyon, PhD thesis 2010.
- [38] A. Rabbachin, "Low complexity uwb receivers with ranging capabilities," Faculty of Technology of the University of Oulu, Ph.D. dissertation 2008.
- [39] U. Mengali, and E. Arias-de Reyna A. D'Amico, "Energy-Detection UWB Receivers with Multiple Energy Measurements," *IEEE Transactions on Wireless Communications*, vol. 6, no. 2, pp. pp. 2652–2659, Jul. 2007.
- [40] F. S. Lee, D. C. Daly, M. Bhardwaj, P. P. Mercier, and A. P. Chan-drakasan D. D. Wentzloff, "Energy Efficient Pulsed-UWB CMOS Circuits and Systems ," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on, 2007*, pp. pp. 282–287.
- [41] R. Merz, J.-Y. Le Boudec, and J. Zory M. Flury, "Performance evaluation of an IEEE 802.15.4a physical layer with energy detection and multi-user interference," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on, 2007*, pp. pp. 663–668.
- [42] M. Casu, M. Graziano, and M. Zamboni M. Crepaldi, "A low-power CMOS 2-ppm demodulator for energy detection IR-UWB receivers," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on, , 2007*, pp. pp. 461–466.
- [43] D. Kreiser, and R. Kraemer S. Olonbayar, "Performance and design of IR-UWB transceiver baseband for wireless sensors," in *IEEE International Conference on Ultra-Wideband (ICUWB), 2009*, pp. pp. 809–813.
- [44] B. Sallberg, J. Nordberg, and I. Claesson M. G. Khan, "Non-coherent detection of impulse radio UWB signals based on fourth order statistics," in *IEEE International Conference on Ultra-Wideband (ICUWB), 2009*, pp. pp. 824–828.
- [45] M. Costa, A. Bevilacqua, D. Vogrig, and A. Neviani A. Gerosa, "An energy-detector for non-coherent impulse-radio UWB receivers," in *Circuits and Systems, 2008. IS-CAS 2008. IEEE International Symposium on, 2008*, pp. pp. 2705–2708.
- [46] M. Wolf, and M. Haardt N. Song, "Low-Complexity and Energy Efficient Non-Coherent Receivers for UWB Communications," *IEEE PIMRC*, pp. pp. 1–4, Sept. 2007.
- [47] Z. Tian and B. Sadler, "Weighted energy detection of ultra-wideband signals," in *Signal Processing Advances in Wireless Communications, 2005 IEEE 6th Workshop on, 2008*, pp. pp. 1068–1072.
- [48] I. Guvenc, and H. Arslan M. Sahin, "Optimization of energy detector receivers for UWB systems," in *Vehicular Technology Conference, 2005. VTC 2005-Spring, 2005*, pp. pp. 1386–1390.
- [49] IEEE, "Channel modeling sub-committee report IEEE P802.15 Wireless Personal Area Network," IEEE, 2003.
- [50] Jean Schwoerer, "Études et implémentation d'une couche physique UWB impulsionnelle à bas débit et faible complexité," PhD thesis 2006.

- [51] SimPy. SimPy. [Online]. <http://simpy.sourceforge.net>
- [52] S. Zirari and B. Denis, "
- [53] Robert Gamble. (2013) consulted 2013 on Source Forge. [Online].
<http://sourceforge.net/p/libcsv/news/>
- [54] B. Miscopein, L. Ouvry, R. d'Errico, F. Dehmas, M. Maman, B. Denis, M. Pezzin, A. Tonnerre, J.-M. Gorce, E. Hamadani J. Schwoerer. FT, CEA, Thal'es, full proposal at IEEE 802.15.6 Task Group. [Online]. <https://mentor.ieee.org/802.15/dcn/09>
- [55] B. Miscopein et J. Schwoerer, "Low complexity synchronization algorithm for non-coherent UWB-IR receivers," in *Vehicular Technology Conference VTC2007-Spring. IEEE 65th*, Dublin, Apr. 2007, , pp. pp. 2344–2348.
- [56] B. Miscopein, J. Schwoerer, L. Fesquet et R. Renaudin] J. Hamon, "Self-timed implementation of an impulse radio synchronisation acquisition algorithm," in *DASIP, Conference on Design and Architectures for Signal and Image Processing*, Belgium 2008, November 2008.
- [57] Annamalai Annamalai Jr., and Amir I. Zaghloul Dongsong Zeng, "Pulseshaping_lter design in uwb system," in *Ultra Wideband Systems and Tech-nologies, 2003 IEEE Conference on*, Nov. 2003.
- [58] L. Euler, *Introductio in analysin infinitorum.*: apud Marcum-Michaelem Bousquet & socios, 1748. [Online]. <http://books.google.fr/books?id=iQ1bAAAAQAAJ>
- [59] Marco Zuniga and Bhaskar Krishnamachari, "Analyzing the transitional region in low power wireless links," in *In First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, 2004, pp. 517-526.
- [60] H. Wymeersch, J. Lien, and M.Z. Win, "Cooperative Localization in Wireless Networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427-450, 2009.
- [61] K. Das and H. Wymeersch, "Censoring for Bayesian Cooperative Positioning in Dense Wireless Networks," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 9, pp. 1835-1842, 2012.
- [62] Claus Pedersen, Troels Pedersen, and Bernard H. Fleury, "Exploiting Network Topology Information to Mitigate Ambiguities in VMP Localization," in *4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2011)*, San Juan, Puerto Rico, #dec# 2011, pp. 57-60.