The WINSS Manual

April, 2016

This document presents Wireless Network Sensor Simulator (WINSS), a simulation platform of the IEEE 802.15.4 Standard for NS-2. The main characteristics and functionalities of the WINSS are described in this document. At the end, an example of simulation is presented.

Contents

How to install NS-2.35 + WINSS in Ubuntu 14.04

1- **Step 1: Setting up the prerequisites.**

1- Download NS-2+WINSS.tar.gz from <https://www.dropbox.com/s/m66tmr74dpz97h1/NS-2%2BWINSS.tar.gz?dl=0>. Copy the downloaded file to your home directory.

2- Before installing the NS-2 + WINSS, we have to install some essential packages required by NS-2 + WINSS. Open up a terminal and execute these commands one by one:

```
sudo apt-get update
sudo apt-get install build-essential autoconf automake
sudo apt-get install tcl8.5-dev tk8.5-dev
sudo apt-get install perl xgraph libxt-dev libx11-dev libxmu-dev
sudo apt-get install gcc-4.4 g++-4.4
sudo apt-get install default-jre
```

2- **Step 2: Extract and install NS-2.35 + WINSS.**

1- Open up a terminal and run the following commands:

```
cd /home/user_name
tar –zxvf NS-2+WINSS.tar.gz
cd NS-2+WINSS/ns-allinone-2.35
./install
```

3- **Step 3: Set the environment variables.**

1- Assuming that you have successfully installed your NS-2.35 + WINSS. There are some environment variables that need to be added to your profile. This can be done by editing the *.bashrc* file. Open up a new terminal and open the file using:

```
gedit ~/.bashrc
```

2- Add the following lines **at the end** of the file. Be sure to change “/path-to” to the path of where you have extracted the NS-2.35 + WINSS.

```
#LD_LIBRARY_PATH
OTCL_LIB=/path-to/ns-allinone-2.35/otcl-1.14
NS2_LIB=/path-to/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB
#TCL_LIBRARY
TCL_LIB=/path-to/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIBRARY:$TCL_LIB:$USR_LIB
#PATH
XGRAPH=/path-to/ns-allinone-2.35/xgraph-12.2/:/path-to/ns-allinone-
2.35/bin/:/path-to/ns-allinone-2.35/tcl8.5.10/unix/:/path-to/ns-allinone-
2.35/tk8.5.10/unix/
NS=/path-to/ns-allinone-2.35/ns-2.35/
NAM=/path-to/ns-allinone-2.35/nam-1.15/
export PATH=$PATH:$XGRAPH:$NS:$NAM

export NS=/path-to/ns-allinone-2.35/ns-2.35
export NSVER=2.35
```

Save the file and reload *.bashrc* as:

```
source ~/.bashrc
```

4- Step 4: Validate the installation.

1- You need to validate NS-2. It will take a lot of time. Open up a terminal and move to the directory “home/user_name/NS-2+WINSS/ns-allinone-2.35/ns-2.35” and run:

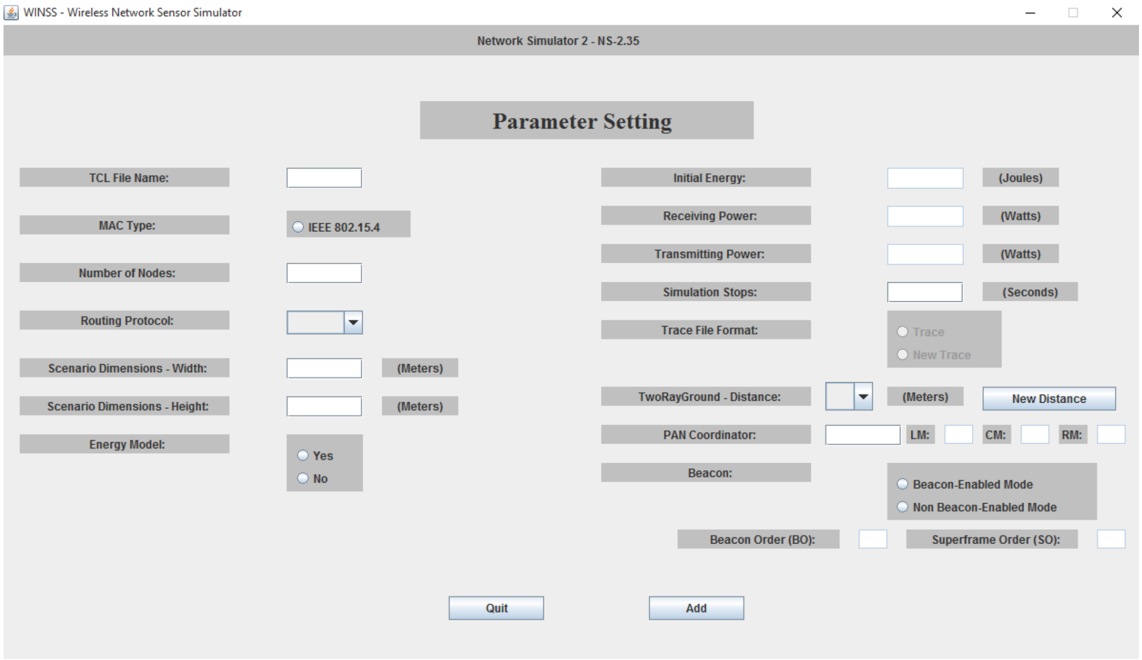
```
./validate
```

WINSS – Wireless Network Sensor Simulator

WINSS is a simulation platform of the IEEE 802.15.4 Standard for Network Simulator 2 (NS-2). WINSS works with the NS-2 (version 2.35) and its auxiliary tools (NAM and XGRAPH). The WINSS consists of four basic steps: The Parameter Settings step (Figure 1) is responsible for setting the general characteristics of the network; The Wireless Sensor Nodes step (Figure 3) is responsible for setting the characteristics of each one of the network nodes; The Communication Setting step (Figure 4) is responsible for setting the communication between the nodes; The Simulation Results step (Figure 6) allows to check the simulation results as well as to visualize the .tcl and .scn file of the network created. The positions of all nodes are saved in the scenario file (.scn).

1. The Parameter Settings step

Figure 1 The Parameter Setting Step.



1- TCL File Name: set the TCL File name.

2- MAC Type: Medium Access Control (MAC). The IEEE 802.15.4 is available.

3- Number of Nodes: set the number of mobile nodes.

4- Routing Protocol: Routing Protocol used. Available protocols are AODV, AOMDV, DSDV and ZBR. The ZBR protocol is not a standard NS-2 protocol, therefore it needed to be implemented and installed in the simulator. The implementation used was developed by Bilel Nefzi and Ye-Qiong Song [1]. For ZBR protocol, you need to set the LM, CM and RM parameters.

 4.1-LM: the minimum number of hops to reach the coordinator using only parent-child link.

 4.2-CM: the maximum number of children.

 4.3-RM: the maximum number of routers a parent may have as children.

5- Scenario Dimensions – Width: Size of the simulation scenario (width), in meters.

6- Scenario Dimensions – Height: Size of the simulation scenario (height), in meters.

7- Energy Model: To use the energy model, or not. For energy model, you need to set the initial energy, receiving power and transmitting power parameters.

 7.1-Initial Energy: the level of energy that the node has at beginning of the simulation, in joules.

 7.2-Receiving Power: energy usage for receiving one packet, in watts.

 7.3-Transmitting Power: energy usage for transmitting one packet, in watts.

8- Simulation Stops: the end time of the simulation, in seconds.

9- Trace File Format: set the Trace File Format. The available formats are Old Wireless Trace and New Wireless Trace. The new trace file format logs more information and is less ambiguous.

10- TwoRayGround – Distance: Radio Propagation model used. TwoRayGround is available. The user can choose one available distance (5, 9, 10, 11, 12, 13, 14 or 15), in meters, or to configure the radio propagation model through the "NEW DISTANCE" button, in Figure 2.

11- PAN Coordinator: set the corresponding node will serve as the PAN coordinator.

12- Beacon: The IEEE 802.15.4 MAC layer can support both beacon-enabled and non beacon-enabled mode. In beacon-enabled mode the nodes will receive beacon frames periodically from the coordinator. The beacon frame consists of active and inactive parts. The active part of the beacon frame is called *superframe*. The Beacon Order (BO) and Superframe Order (SO) determine the *superframe* structure. The values of BO and SO are determined by coordinator. These two attributes define the interval at which beacons are sent by a coordinator and the length of a superframe's active period, respectively [2].

After all parameters are configured, you can pass to next step. Click the "Add" button.

2. Two-Ray Ground Reflection Model

The Two-Ray ground reflection model is represented by Equation 1

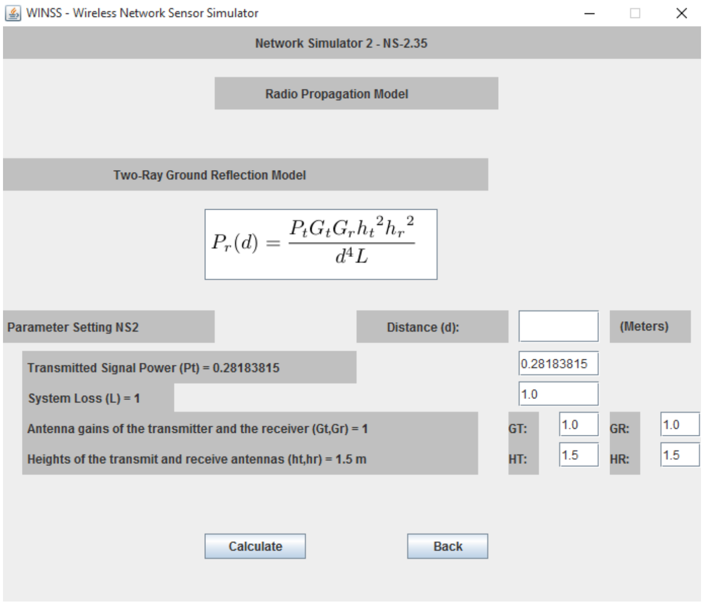
$$P_r\,(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{1}$$

where

1- *d*: the distance.
2- *Pr*: the received signal power
3- *Pt*: the transmitted signal power.
4- *L*: the system loss.
5- *Gt*: the antenna gain of the transmitter
6- *Gr*: the antenna gain of the received
7- *ht*: the height of the transmit antennas
8- *hr*: the height of receive antennas

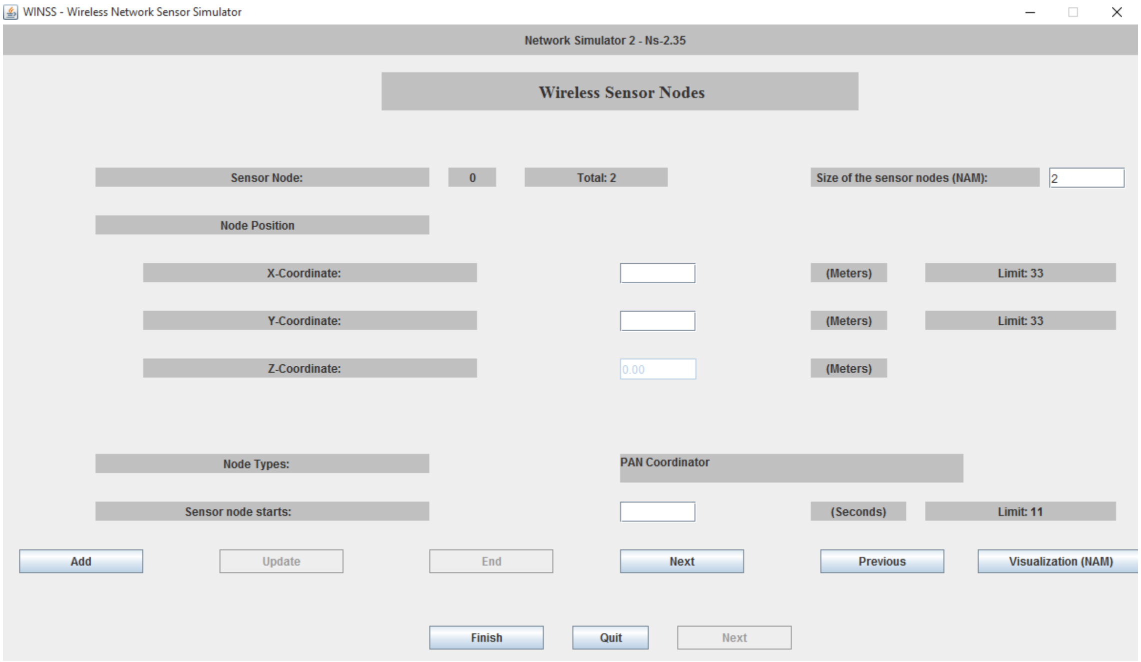
You can configure the radio propagation model (Two-Ray ground model). For this, you need to set all parameters of the radio propagation model and click the "Calculate" button.

Figure 2 The configuring the radio propagation model (Two-Ray ground model).



3. The Wireless Sensor Nodes step

Figure 3 The Wireless Sensor Nodes step.



1- X-Coordinate: set the node position in x-coordinate, in meters.
2- Y-Coordinate: set the node position in y-coordinate, in meters.
3- Z-Coordinate: set the node position in z-coordinate, in meters (Zero).
4- Node Types: Two different types of devices are defined in an 802.15.4 network, a full function device (FFD) and a reduced function device (RFD). An FFD can talk to RFDs and other FFDs, and operate in three modes serving either as a PAN coordinator, a coordinator or a device. An RFD can only talk to an FFD [3].
5- Sensor node starts: the startup time of the node, in seconds.

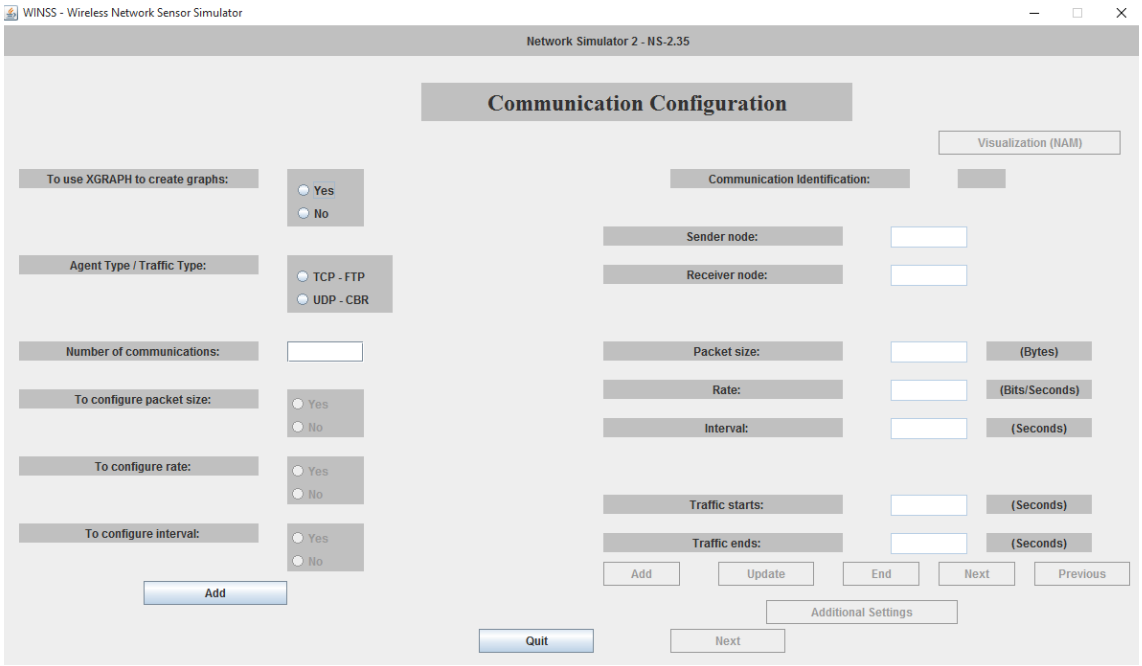
Click the “Add” button to store the information about the corresponding node. Click the “End” button to finish the configuration about the corresponding node and to pass to next sensor node. Press the "Update" button to update the information about one sensor node. Click the "Next" button to move to the next sensor node. Click the "Previous" button to move to the previous sensor node.

The NAM (Network Animator) is an animation tool for viewing network simulation. Click the "Visualization (NAM)" button to view how are configured the sensor nodes in the network. You can change the size the node in NAM. For this, set the size of the sensor nodes (NAM).

After all sensor nodes are configured, you can pass to next step. Click the "Finish" button and "Next" button.

4. The Communication Setting step

Figure 4 The Communication Setting step.



1- To use XGRAPH to create graphs: Some results can be plotted using XGRAPH. LossMonitor objects are a subclass of agent objects that implement a traffic sink which also maintains some statistics about the received data, number of bytes received, number of packets lost and etc. All of the data from the UDP simulation were collected using the LossMonitor class. All of the data from the TCP simulation were collected using the TCP Sink Monitor [4]. TCP Sink Monitor is a simple TCP Sink Agent that keeps some statistics about TCP connection on the sink (receiver) side. This agent was created by Luigi Iannone [4]. Using this statistics data, we were able to compute the throughput, packets loss, delay and packets received. The energy Consumption is measured through of analysis of the trace files using AWK (an interpreted programming language designed for text processing). For the throughput, delay, packets loss and packets received, the graphs will have a maximum limit of 7 communications.

2- Agent Type / Traffic Type: Transport Protocol used. Available protocols are TCP and UDP. The variants of the TCP protocol are available: TCP Reno, TCP NewReno and TCP Vegas.

3- Number of communications: set the number of communications in the simulation.

4- To configure packet size: to configure packet size, or not. The default TCP packet size has a size of 1000 bytes. The default UDP packet size has a size of 210 bytes.

4.1-Packet size: set the packet size, in bytes.

5- To configure rate: to configure rate, or not. The default data rate of CBR in NS-2 is 448 kbps.

 5.1- Rate: set the transmission rate, in bits/seconds.

6- To configure interval: to configure interval, or not.

 6.1- Interval: set the time interval between transmissions of packets, in seconds.

7- Sender node: set the sender node.

8- Receiver node: set the receiver node.

9- Traffic starts: set the beginning of the traffic, in seconds.

10- Traffic ends: set the end of the traffic, in seconds.

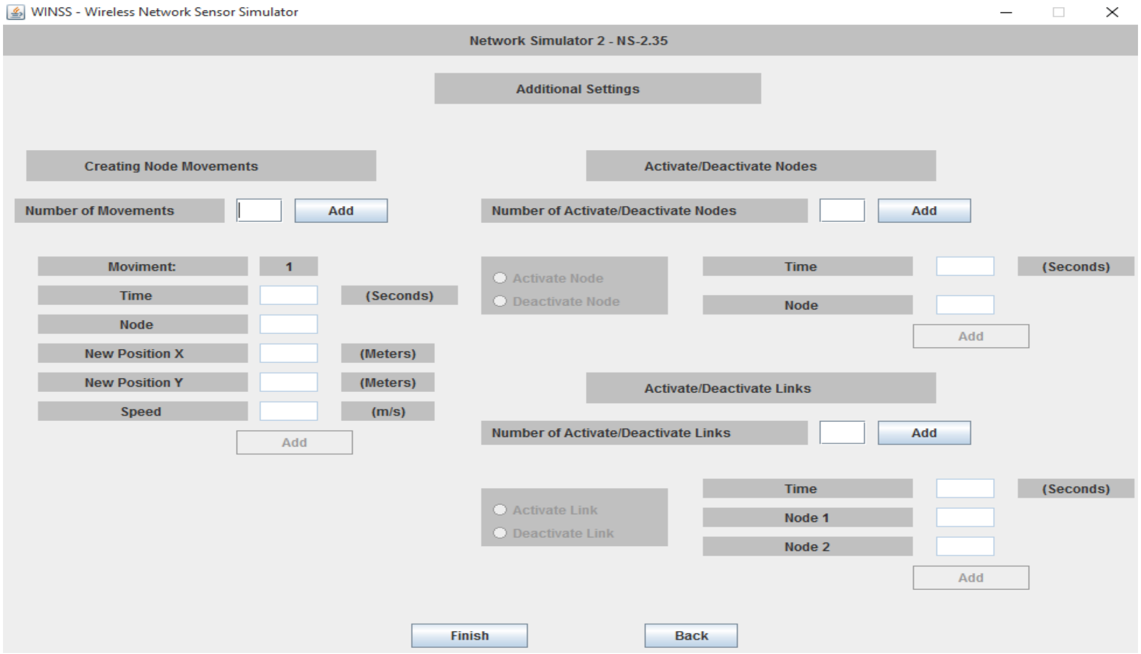
Click the “Add” button to store the information about the corresponding communication. Press the "Update" button to update the information about one communication. Click the "Next" button to move to the next communication. Click the "Previous" button to move to the previous communication.

If necessary, click the "Visualization (NAM)" button to view the topology layout.

After all communications are configured, you can pass to next step. Click the “End” button and “Next” button.

You can use the network fault model and mobile nodes model, in Figure 5. Click the “Additional Settings” button.

Figure 5 The configuration of the mobile nodes model and network fault model.



- Creating Node Movements

1- Number of Movements: set the number of movements in the simulation, after click the "Add" button.
2- Time: set the time that the node would start moving, in seconds.
3- Node: set the node would start moving.
4- New Position X: set the new destination position in x-coordinate, in meters.
5- New Position Y: set the new destination position in y-coordinate, in meters.
6- Speed: set the speed of movement of the node, in m/s.

Click the "Add" button to store the information about the node movements.

- Activate/Deactivate Nodes

1- Number of Activate/Deactivate Nodes: set the number of activate/deactivate nodes in the simulation, after click the "Add" button.
2- Activate/Deactivate Nodes
   2.1-Activate Nodes: Node restoration, bring up the failed node.
   2.2-Deactivate Nodes: Node failure, bring down the node.
3- Time: set the time of the node failure or restoration, in seconds.
4- Node: set the node would fail or restore.

Click the "Add" button to store the information about the activate/deactivate nodes.

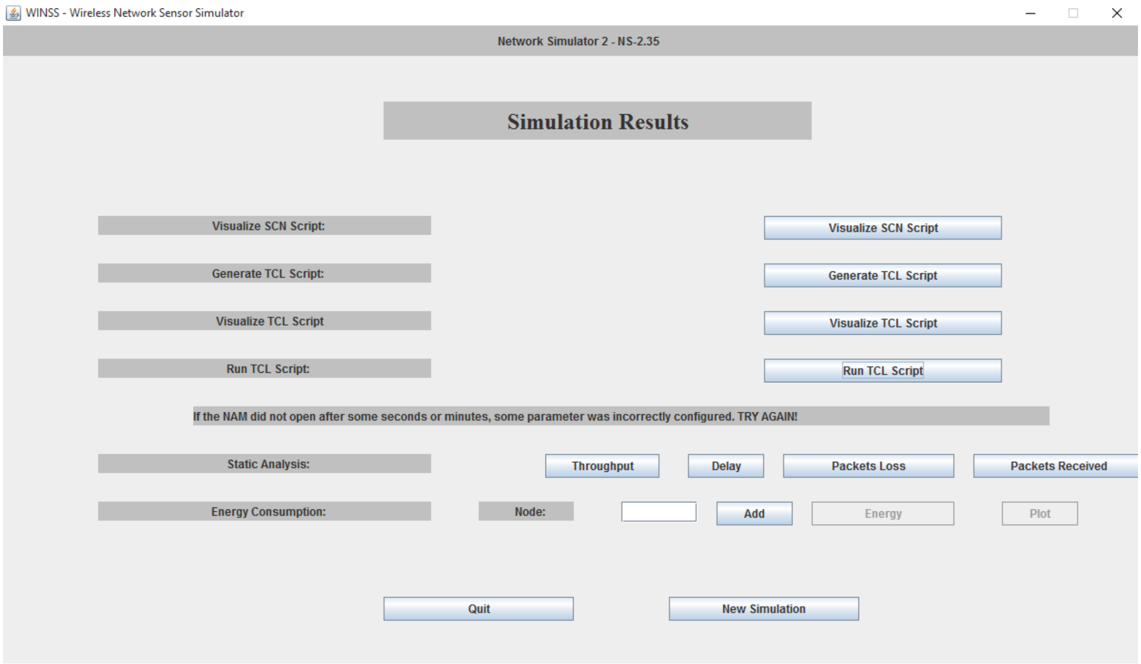
- Activate/Deactivate Links

1- Number of Activate/Deactivate Links: set the number of activate/deactivate links in the simulation, after click the "Add" button.
2- Activate/Deactivate Links
   2.1-Activate Links: Link restoration, bring up the broken link.
   2.2-Deactivate Links: Link failure, bring down the link.
3- Time: set the time of the link failure or restoration, in seconds.
4- Node 1 and Node 2: Bring down the link between node 1 and node 2 / Bring up the broken link between node 1 and node 2.

Click the "Add" button to store the information about the activate/deactivate links.

5. The Simulation Results step

Figure 6 The Simulation Results step.



Click the “Visualize SCN Script” button to view the .scn file. Press the "Generate TCL Script" button to create the .tcl file. Click the "Visualize TCL Script" button to view the .tcl file. Click the "Run TCL Script" button to execute the simulation in the NS-2. The NAM should automatically starts after the execution of the simulation.

Click the “Throughput” button to plot the throughput. Click the “Delay” button to plot the delay. Click the “Packets Loss” button to plot packets loss. Click the “Packets Received” button to plot the packets received.

The energy Consumption is measured for each node. You need to specify the node. After that, click the "Add", "Energy" and "Plot" buttons, respectively.

Example of Simulation

1- **Step 1: Run the WINSS.**

Note: The WINSS requires Java Runtime Environment version 1.7 .

Open up a terminal and run these commands:

```
cd NS-2+WINSS/ns-allinone-2.35/winss
java –jar WINSS.jar
```

2- **Step 2: Setting up the simulation.**

An example of simulation was carried out using the WINSS simulation platform. The simulation settings are shown in the Table 1, 2, 3 and 4.

Table 1 Simulation Parameters (The Parameter Setting Step)

Simulation Parameters	Value
TCL File Name	Example
PHY/MAC Protocol	IEEE 802.15.4
Number of nodes	6
Network Protocol	AODV
Network size	50m X 50m
Energy Model	Yes
Initial Energy	2.0
Receiving Power	0.1
Transmitting Power	0.2
Simulation time	100 sec
Trace File Format	New Trace
TwoRayGround-Distance	15 m
PAN Coordinator	0
Beacon	Non Beacon-Enabled Mode

Table 2 Simulation Parameters (The Wireless Sensor Nodes Step)

	Node 0	**Node 1**	**Node 2**	**Node 3**	**Node 4**	**Node 5**
X – Coordinate	30 m	22 m	35 m	13.5 m	29 m	40 m
Y – Coordinate	40 m	32 m	32 m	24 m	24 m	23 m
Z – Coordinate	0	0	0	0	0	0
Node Type	PAN Coord.	FFD	FFD	RFD	RFD	RFD
Sensor Node starts	0 s	0.5 s	1.5 s	2.5 s	3.5 s	4.5 s

Table 3 Simulation Parameters (The Communication Configuration Step)

Simulation Parameters	**Value**
Use XGRAPH	Yes
Agent Type / Traffic Type	UDP - CBR
Number of Communication	3
Configure Packet Size	Yes
Configure Rate	No
Configure Interval	Yes

Table 4 Communication Parameters (The Communication Configuration Step)

Parameters	**1º Communication**	**2º Communication**	**3º Communication**
Sender Node	3	4	5
Received Node	0	0	0
Packet Size	70 bytes	70 bytes	70 bytes
Interval	0.2 sec	0.2 sec	0.2 sec
Start/End time	10 - 100 sec	20 - 100 sec	30 - 100 sec

After running the simulation, it is possible to measure simulation results: Throughput (Figure 7); Delay (Figure 8); Packets Received (Figure 9); and Packet Loss (Figure 10). Besides the possibility of graphic visualization of the simulation by NAM tool shown in Figure 11.

Figure 7 Throughput.

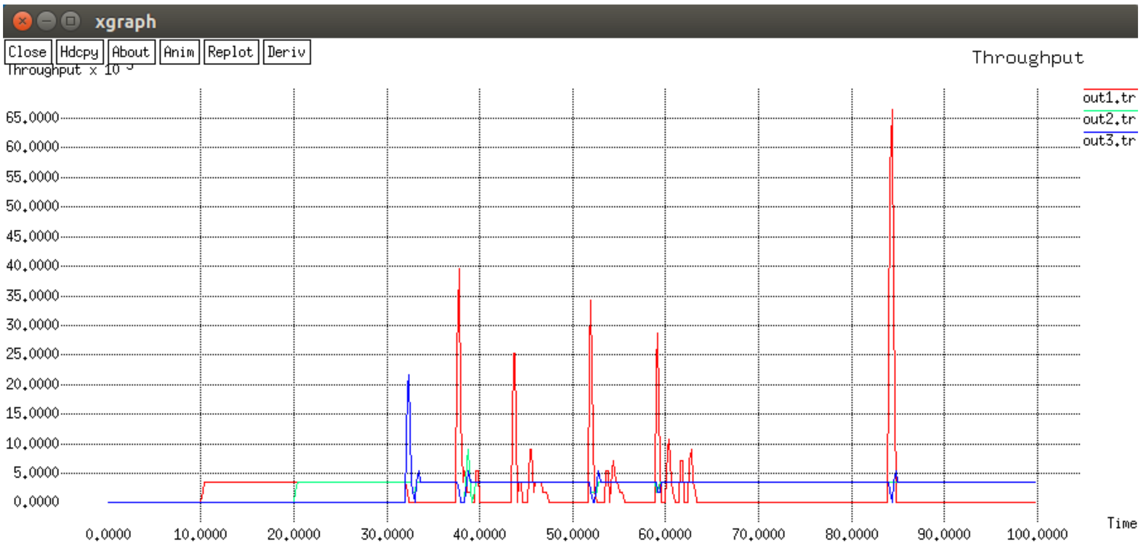


Figure 8 Delay.

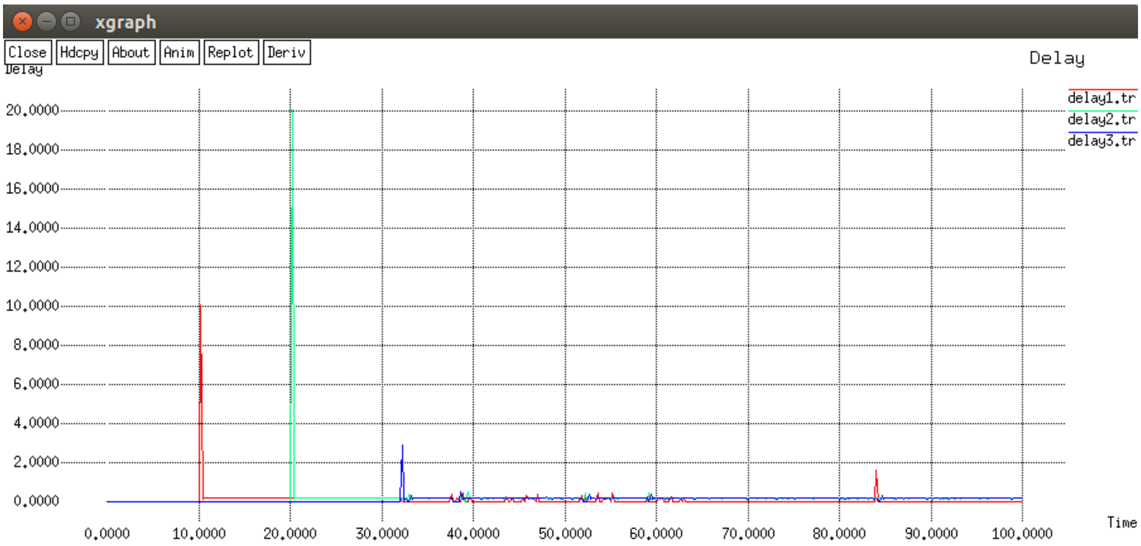


Figure 9 Packets Received.

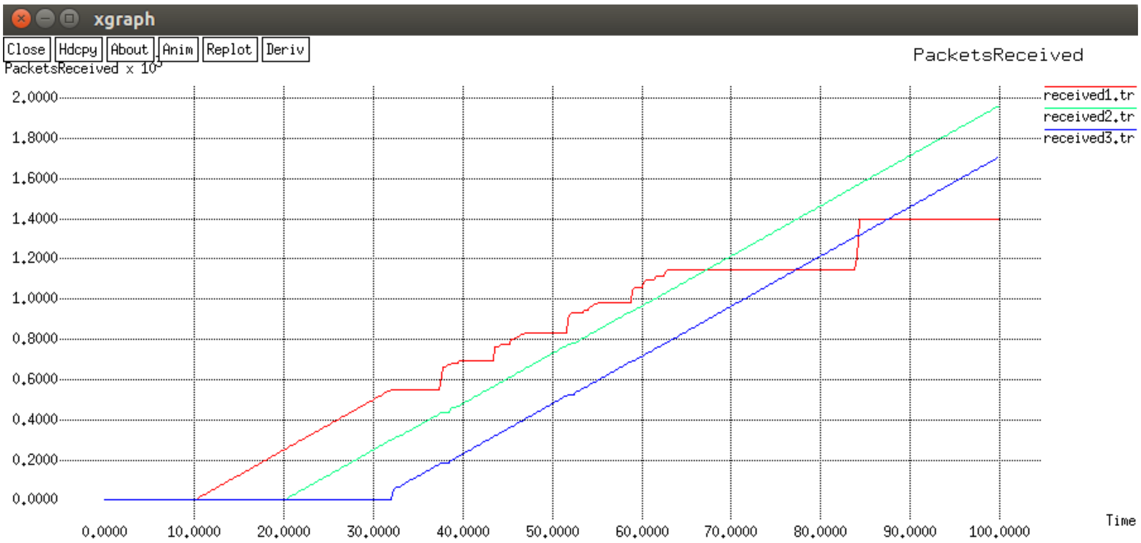


Figure 10 Packets Loss.

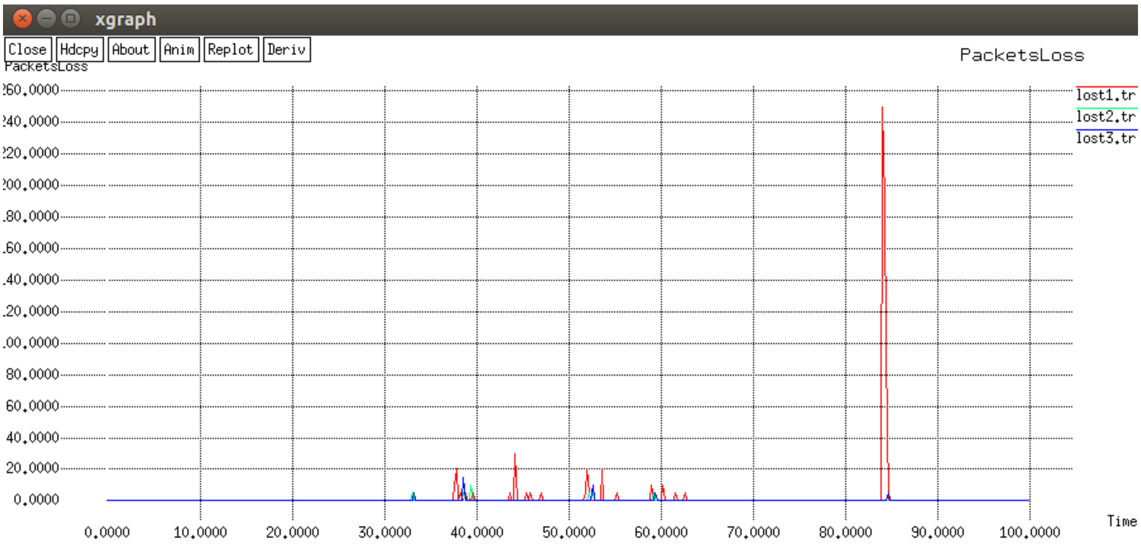
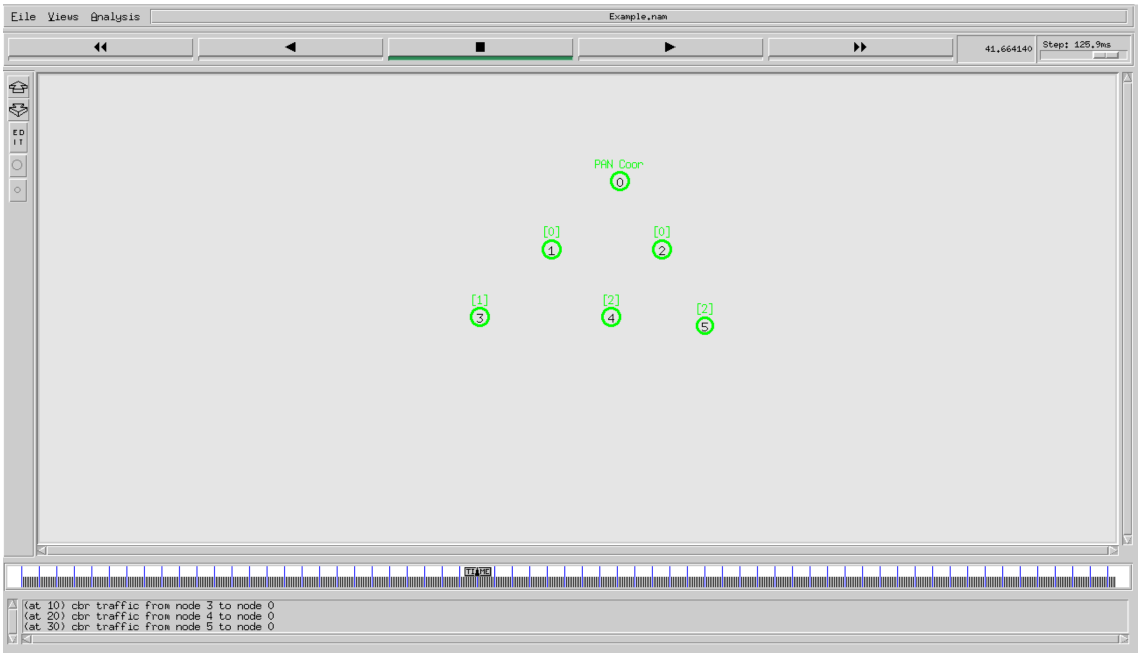


Figure 11 NAM.



The Fig. 12, 13, 14, 15, 16 and 17 show the energy consumption each node.

Figure 12 Energy Consumption by node 0.

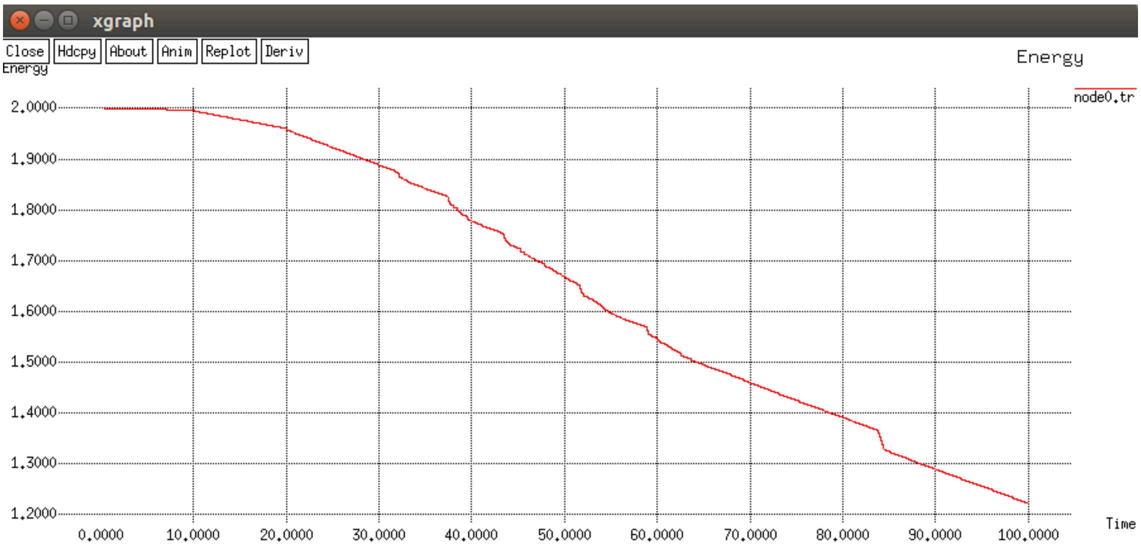


Figure 13 Energy Consumption by node 1.

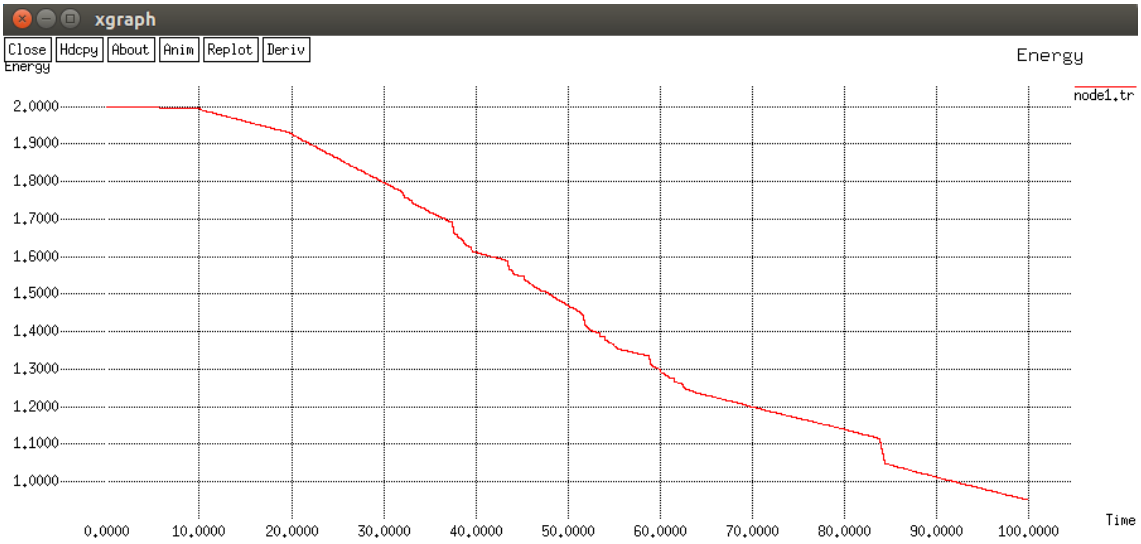


Figure 14 Energy Consumption by node 2.

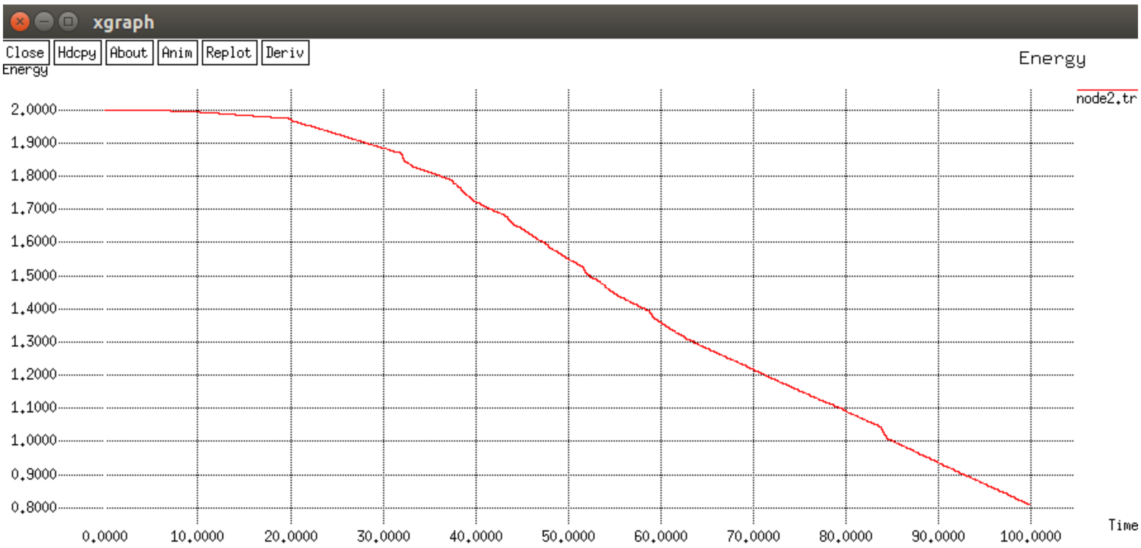


Figure 15 Energy Consumption by node 3.

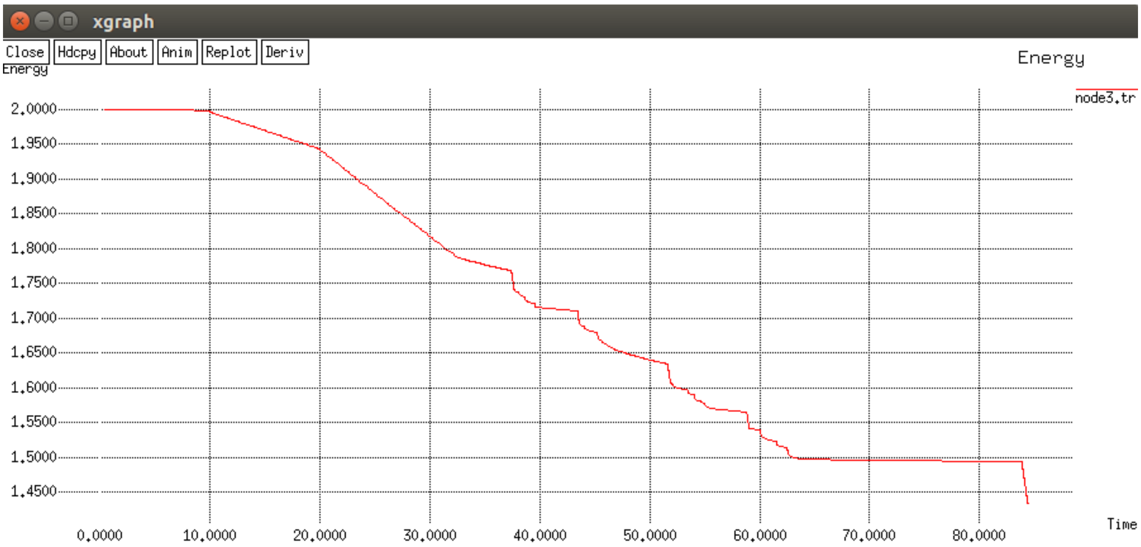


Figure 16 Energy Consumption by node 4.

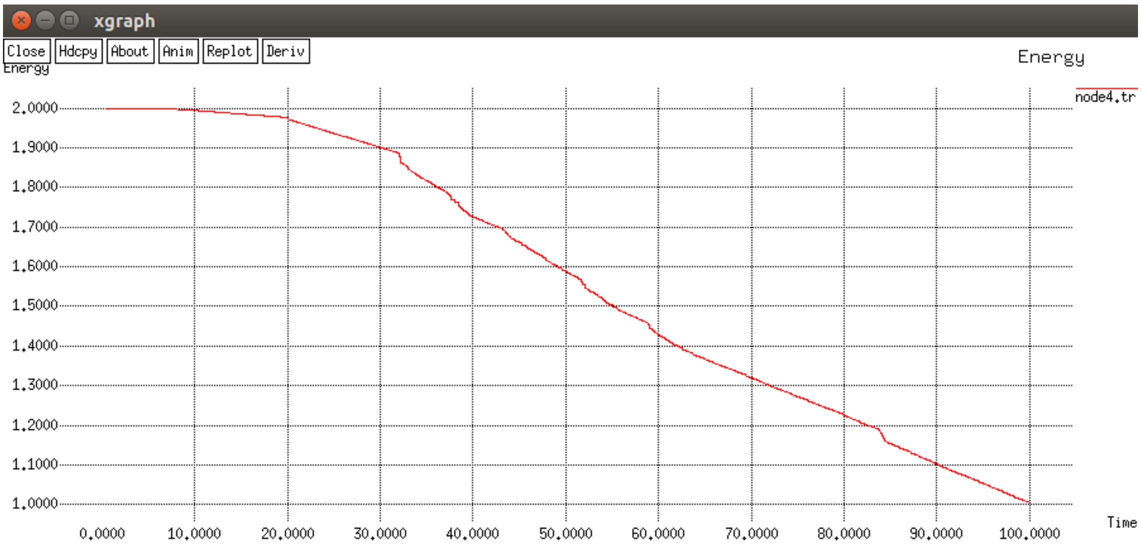
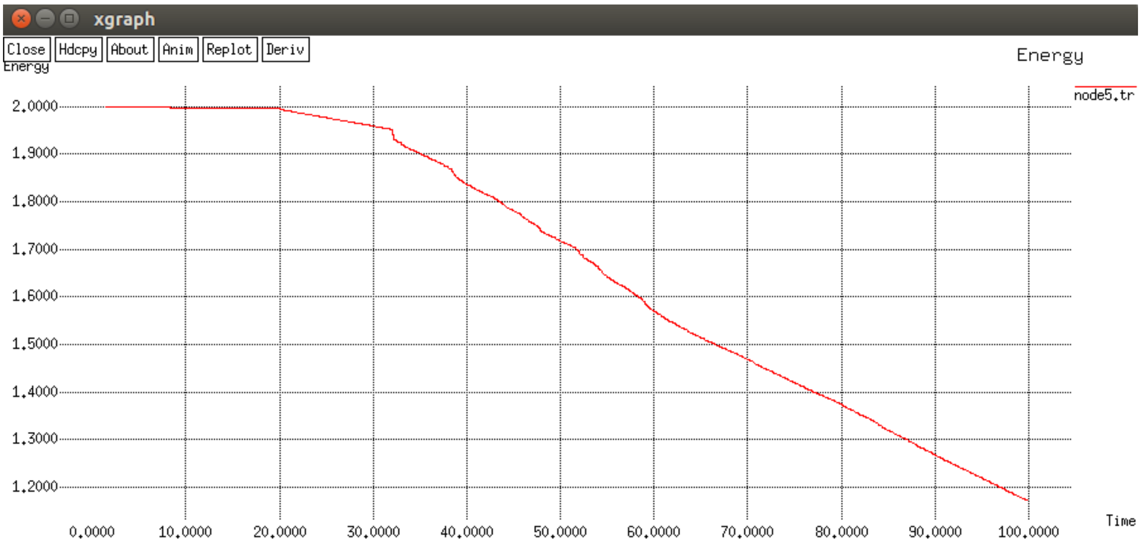


Figure 17 Energy Consumption by node 5.



References

[1] B. Nefzi and Y. Song, “Performance Analysis and Improvement of ZigBee Routing Protocol”, 7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT’2007, Nov 2007, Toulouse, France.

[2] J. Hoffert, K. Klues and O. Orjih, “Configuring the IEEE 802.15.4 MAC Layer for Single Sink Wireless Sensor Network Applications,” Real Time Systems Class Project, Washington University, St. Louis, MO, December 2005.

[3] S. Phoha, T. Porta, and C. Griffin, Sensor Network Operations, Wiley, John & Sons, Incorporated, pp. 222, 2006.

[4] TCP Sink Monitor, Available: http://www.gigix.net.

Appendix

Example.scn

```
$node_(0) set X_ 30
$node_(0) set Y_ 40
$node_(0) set Z_ 0.00
$node_(1) set X_ 22
$node_(1) set Y_ 32
$node_(1) set Z_ 0.00
$node_(2) set X_ 35
$node_(2) set Y_ 32
$node_(2) set Z_ 0.00
$node_(3) set X_ 13.5
$node_(3) set Y_ 24
$node_(3) set Z_ 0.00
$node_(4) set X_ 29
$node_(4) set Y_ 24
$node_(4) set Z_ 0.00
$node_(5) set X_ 40
$node_(5) set Y_ 23
$node_(5) set Z_ 0.00
```

Example.tcl

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)          Channel/WirelessChannel    ;
set val(prop)          Propagation/TwoRayGround   ;
set val(netif)         Phy/WirelessPhy/802_15_4
set val(mac)           Mac/802_15_4
set val(ifq)           Queue/DropTail/PriQueue    ;
set val(ll)            LL                         ;
set val(ant)           Antenna/OmniAntenna        ;
set val(ifqlen)        150                        ;
set val(nn)            6                          ;
set val(rp)            AODV                       ;
set val(x)       50
set val(y)       50
set val(energymodel)    EnergyModel             ;
set val(initialenergy)  2.0                  ;

set val(nam)       Example.nam
set val(traffic)    cbr ;
```

```
set appTime1        10;
set appTime2        100;
set appTime3        20;
set appTime4        100;
set appTime5        30;
set appTime6        100;

set stopTime        100    ;# in seconds

#=======================================================================
# Initialize trace file desctiptors
#=======================================================================

# *** Throughput Trace ***
set f1 [open out1.tr w]
set f2 [open out2.tr w]
set f3 [open out3.tr w]

# *** Packet Lost Trace ***
set f4 [open lost1.tr w]
set f5 [open lost2.tr w]
set f6 [open lost3.tr w]

# *** Packet Delay Trace ***
set f7 [open delay1.tr w]
set f8 [open delay2.tr w]
set f9 [open delay3.tr w]

# *** Packet Received Trace ***
set f10 [open received1.tr w]
set f11 [open received2.tr w]
set f12 [open received3.tr w]

# Initialize Global Variables
set ns_      [new Simulator]
$ns_ use-newtrace          ;
set tracefd    [open ./Example.tr w]
$ns_ trace-all $tracefd
if { "$val(nam)" == "Example.nam" } {
    set namtrace    [open ./$val(nam) w]
    $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
}

$ns_ puts-nam-traceall {# nam4wpan #}      ;# inform nam that this is a trace file for wpan (special handling needed)

Mac/802_15_4 wpanCmd verbose on       ;
Mac/802_15_4 wpanNam namStatus on       ;

# For model 'TwoRayGround'
set dist(5m)  7.69113e-06
set dist(9m)  2.37381e-06
```

```
set dist(10m) 1.92278e-06
set dist(11m) 1.58908e-06
set dist(12m) 1.33527e-06
set dist(13m) 1.13774e-06
set dist(14m) 9.81011e-07
set dist(15m) 8.54570e-07
Phy/WirelessPhy set CSThresh_ $dist(15m)
Phy/WirelessPhy set RXThresh_ $dist(15m)

# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create God
set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]

# configure node
$ns_ node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace ON \
            -energyModel $val(energymodel) \
            -initialEnergy $val(initialenergy) \
            -rxPower 0.1 \
            -txPower 0.2 \
            -channel $chan_1_

for {set i 0} {$i < $val(nn) } {incr i} {
          set node_($i) [$ns_ node]
          $node_($i) random-motion 0       ;
}

source ./Example.scn

$ns_ at 0.0 "record"

$ns_ at 0.0    "$node_(0) NodeLabel PAN Coor"
$ns_ at 0.0    "$node_(0) sscs startPANCoord 0 "        ;
$ns_ at 0.5     "$node_(1) sscs startDevice  "        ;
$ns_ at 1.5     "$node_(2) sscs startDevice  "        ;
$ns_ at 2.5     "$node_(3) sscs startDevice  0 "        ;
```

```
$ns_ at 3.5     "$node_(4) sscs startDevice  0 "       ;
$ns_ at 4.5     "$node_(5) sscs startDevice  0 "       ;

# Setup traffic flow between nodes
if { ("$val(traffic)" == "cbr") } {
   puts "Traffic: $val(traffic)"
   puts [format "Acknowledgement for data: " [Mac/802_15_4 wpanCmd ack4data]]
   #Communication1
   set udp1 [new Agent/UDP]
   set sink1 [new Agent/LossMonitor]
   $ns_ attach-agent $node_(3) $udp1
   $ns_ attach-agent $node_(0) $sink1
   $ns_ connect $udp1 $sink1
   set cbr1 [new Application/Traffic/CBR]
   $cbr1 attach-agent $udp1
   $cbr1 set packetSize_ 70
   $cbr1 set interval_ 0.2
   $cbr1 set random_ 0
   $ns_ at 10 "$cbr1 start "
   $ns_ at 100 "$cbr1 stop "

   #Communication2
   set udp2 [new Agent/UDP]
   set sink2 [new Agent/LossMonitor]
   $ns_ attach-agent $node_(4) $udp2
   $ns_ attach-agent $node_(0) $sink2
   $ns_ connect $udp2 $sink2
   set cbr2 [new Application/Traffic/CBR]
   $cbr2 attach-agent $udp2
   $cbr2 set packetSize_ 70
   $cbr2 set interval_ 0.2
   $cbr2 set random_ 0
   $ns_ at 20 "$cbr2 start "
   $ns_ at 100 "$cbr2 stop "
   #Communication3
   set udp3 [new Agent/UDP]
   set sink3 [new Agent/LossMonitor]
   $ns_ attach-agent $node_(5) $udp3
   $ns_ attach-agent $node_(0) $sink3
   $ns_ connect $udp3 $sink3
   set cbr3 [new Application/Traffic/CBR]
   $cbr3 attach-agent $udp3
   $cbr3 set packetSize_ 70
   $cbr3 set interval_ 0.2
   $cbr3 set random_ 0
   $ns_ at 30 "$cbr3 start "
   $ns_ at 100 "$cbr3 stop "

   Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
   Mac/802_15_4 wpanNam FlowClr -p ARP -c green
   Mac/802_15_4 wpanNam FlowClr -p MAC -s 0 -d -1 -c navy
   if { "$val(traffic)" == "cbr" } {
```

```
      set pktType cbr
   } else {
      set pktType exp
   }

   $ns_ at $appTime1 "$ns_ trace-annotate \"(at $appTime1) $val(traffic) traffic from node 3 to node 
0\" "
   Mac/802_15_4 wpanNam FlowClr -p $pktType -s 3 -d 0 -c blue
   $ns_ at $appTime3 "$ns_ trace-annotate \"(at $appTime3) $val(traffic) traffic from node 4 to node 
0\" "
   Mac/802_15_4 wpanNam FlowClr -p $pktType -s 4 -d 0 -c green4
   $ns_ at $appTime5 "$ns_ trace-annotate \"(at $appTime5) $val(traffic) traffic from node 5 to node 
0\" "
   Mac/802_15_4 wpanNam FlowClr -p $pktType -s 5 -d 0 -c cyan4
}

# defines the node size in nam
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns_ initial_node_pos $node_($i) 2
}

# Initialize Flags
set holdtime1  0
set holdseq1  0

set holdtime2  0
set holdseq2  0

set holdtime3  0
set holdseq3  0

set holdrate1  0
set holdrate2  0
set holdrate3  0

# Function To record Statistcis (Bit Rate, Delay, Drop)
proc record {} {

   global sink1 sink2 sink3 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 holdrate1 holdrate2 holdrate3 holdtime1 
holdseq1 holdtime2 holdseq2 holdtime3 holdseq3
   set ns [Simulator instance]
   set time 0.2 ;#Set Sampling Time to 0.2 Sec

   set bw1 [$sink1 set bytes_]
   set bw2 [$sink2 set bytes_]
   set bw3 [$sink3 set bytes_]

   set bw4 [$sink1 set nlost_]
   set bw5 [$sink2 set nlost_]
   set bw6 [$sink3 set nlost_]

   set bw7 [$sink1 set npkts_]
```

```
set bw8 [$sink2 set npkts_]
set bw9 [$sink3 set npkts_]

set bw10 [$sink1 set lastPktTime_]
set bw11 [$sink1 set npkts_]
set bw12 [$sink2 set lastPktTime_]
set bw13 [$sink2 set npkts_]
set bw14 [$sink3 set lastPktTime_]
set bw15 [$sink3 set npkts_]

set now [$ns now]

# Record Bit Rate in Trace Files
puts $f1 "$now [expr (($bw1+$holdrate1)*8)/(2*$time*1000000)]"
puts $f2 "$now [expr (($bw2+$holdrate2)*8)/(2*$time*1000000)]"
puts $f3 "$now [expr (($bw3+$holdrate3)*8)/(2*$time*1000000)]"

# Record Packet Loss Rate in File
puts $f4 "$now [expr $bw4/$time]"
puts $f5 "$now [expr $bw5/$time]"
puts $f6 "$now [expr $bw6/$time]"

# Record Packet Received Rate in File
puts $f10 "$now [expr $bw7/$time]"
puts $f11 "$now [expr $bw8/$time]"
puts $f12 "$now [expr $bw9/$time]"
# Record Packet Delay in File
if { $bw11 > $holdseq1 } {
   puts $f7 "$now [expr ($bw10 - $holdtime1)/($bw11 - $holdseq1)]"
} else {
   puts $f7 "$now [expr ($bw11 - $holdseq1)]"
}
if { $bw13 > $holdseq2 } {
   puts $f8 "$now [expr ($bw12 - $holdtime2)/($bw13 - $holdseq2)]"
} else {
   puts $f8 "$now [expr ($bw13 - $holdseq2)]"
}
if { $bw15 > $holdseq3 } {
   puts $f9 "$now [expr ($bw14 - $holdtime3)/($bw15 - $holdseq3)]"
} else {
   puts $f9 "$now [expr ($bw15 - $holdseq3)]"
}

# Reset Variables

$sink1 set bytes_ 0
$sink2 set bytes_ 0
$sink3 set bytes_ 0

$sink1 set nlost_ 0
$sink2 set nlost_ 0
$sink3 set nlost_ 0
```

```
    set holdtime1 $bw10
    set holdseq1 $bw11
    set holdtime2 $bw12
    set holdseq2 $bw13
    set holdtime3 $bw14
    set holdseq3 $bw15

    set  holdrate1 $bw1
    set  holdrate2 $bw2
    set  holdrate3 $bw3

    $ns at [expr $now+$time] "record"; # Schedule Record after $time interval sec
}

# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stopTime "$node_($i) reset";
}

$ns_ at $stopTime "stop"
$ns_ at $stopTime "puts \"\nNS EXITING...\""
$ns_ at $stopTime "$ns_ halt"

proc stop {} {

    global ns_ tracefd val env f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12

    # Close Trace Files
    close $f1
    close $f2
    close $f3
    close $f4
    close $f5
    close $f6
    close $f7
    close $f8
    close $f9
    close $f10
    close $f11
    close $f12

    # Plot Recorded Statistics
    #exec xgraph out1.tr out2.tr out3.tr -geometry 900x400 &
    #exec xgraph lost1.tr lost2.tr lost3.tr -geometry 900x400 &
    #exec xgraph delay1.tr delay2.tr delay3.tr -geometry 900x400 &
    #exec xgraph received1.tr received2.tr received3.tr -geometry 900x400 &

    $ns_ flush-trace
    close $tracefd
    set hasDISPLAY 0
```

```
    foreach index [array names env] {
        #puts "$index: $env($index)"
        if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
            set hasDISPLAY 1
        }
    }

    if { ("$val(nam)" == "Example.nam") && ("$hasDISPLAY" == "1") } {
        exec nam Example.nam &
    }

    exit 0

}

puts "\nStarting Simulation..."
$ns_ run
```