DANSE: INSTRUCTIONS FOR OPERATING THE SIMULATOR

Version 0.12 – 17th January 2012

Dave Pearce dajp1@ohm.york.ac.uk

This document contains the information required to operate the DANSE simulator.

If there's a function or feature that you would like but is not currently available, please ask the author, it's quite easy to modify the simulator to add new functions.

This is a new module, and the simulator has not been exhaustively tested, so it's likely that there are some bugs left in it. You're advised to check if things are working as you expect wherever, and report anything odd. Spotting errors, documenting and reporting them carefully will be rewarded.

All comments and feedback, please address to the author.

Contents

1 Introduction

DANSE is a wireless ad-hoc and sensor network simulator written specifically for use as a teaching aid in communications. It is optimised for ease-of use, rather than for speed or flexibility.

This document contains the information required to use the DANSE simulator with the built-in protocols. (For instructions on writing user-defined protocols, please see the document "DANSE: Instructions for Writing User Protocols".)

DANSE enforces a strict layered protocols architecture, with a well-defined interface between the layers. (There is some limited provision for passing information between layers for implementing cross-layer techniques, but these require the use of special function calls, and are not used by any of the built-in protocols.)

There is an emphasis on low-energy techniques, and energy considerations are integrated into the simulator, with nodes consuming energy at different rates when receiving, transmitting, actively listening, and asleep. The energy usage is shown on the front screen, to provide clear feedback to the users about the energy consumption of the nodes.

Up to one hundred nodes can be placed in a variety of configurations (including randomly), and packets sent either to random locations, or an access point / base station.

After a review of the protocol stack used and an overview of the user interface, this document is split into several chapters covering the operation of the simulator:

- Chapter Two: The user interface, and the controls for running simulations
- Chapter Three: Configuring the simulator and monitoring the simulations
- Chapter Four: Configuring the protocols
- Chapter Five: Outputting data from the simulator
- Chapter Six: Some technical details about the simulator

2 The User Interface

The simulator has a user interface with five tabs: Main, Setup Simulation, Setup Protocols, Outputs and PHY Activity. In this section, I'll describe the contents of the Main tab only, which divides into five areas, as shown in the figure below:

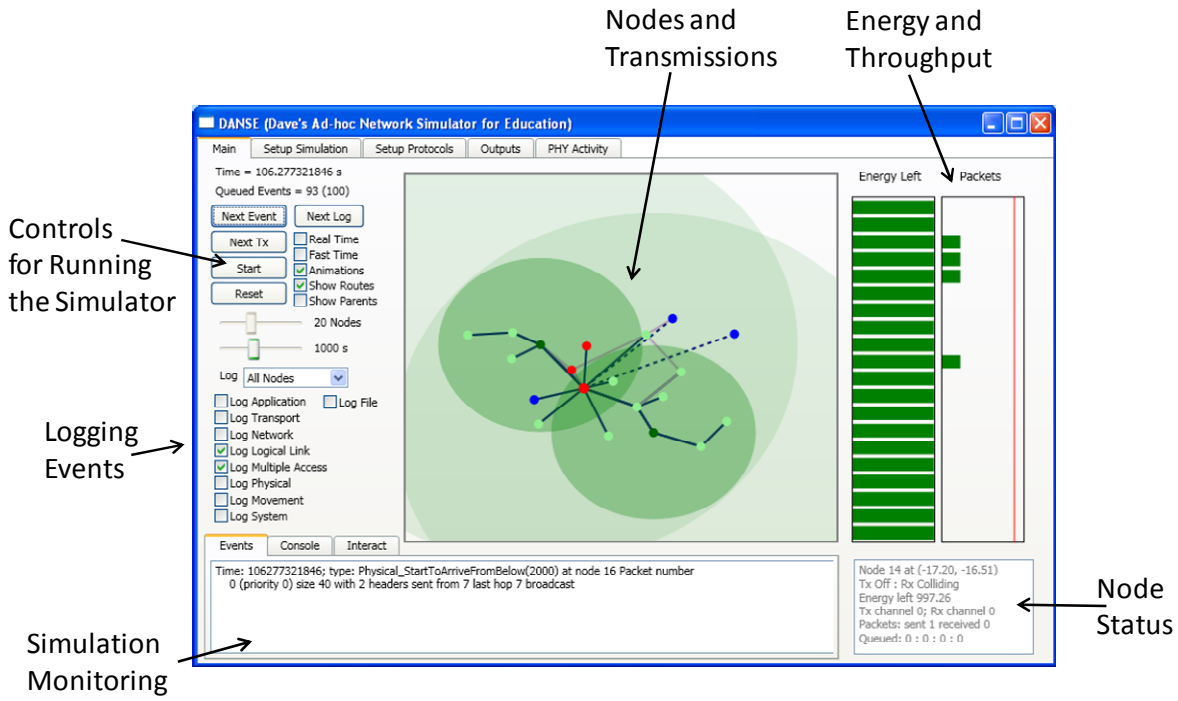


Figure 2-1 The Main Controls Tab

2.1 Changing Slider Values

Sliders give a good visual indication of the value set for the variables, but it can be hard to set sliders to an exact value if that is required. There are a couple of things you should know about the sliders in DANSE.

Firstly, many of the sliders have default values; and if you double-click on the slider bar for that parameter, it will return to its default value.

Secondly, many of the sliders have the value of the parameter printed next to them, for example:

Figure 2-2 Typical Slider Control

To set the parameter to an exact value, double-click on the value. It will turn into a textbox, into which you can type the exact value you want. Press return, and the slider will be set to the closest legal value.

2.2 Controls for Running the Simulator

This section of the Main tab contains controls for starting and stopping simulations, stepping through simulations, and configuring the display. This part of the display is shown in the figure below:

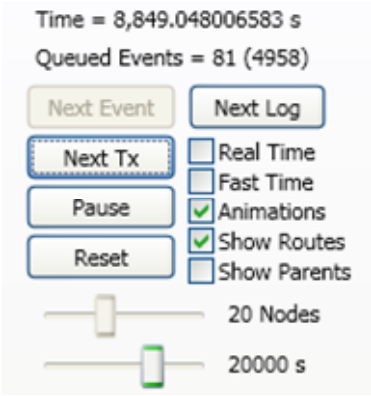


Figure 2-3 The Simulator Controls

2.2.1 The Time and Queued Events Indicators

The Time indicator gives the current global simulation time in seconds. The simulator works in nanoseconds, the displayed resolution is the maximum available. Note that nodes can have independent clocks, so the displayed time is not necessarily the time that the nodes internal clocks would give.

The Queued Events indicator displays how many events are currently waiting in the event queue. (The figure in brackets is the number of events that have been processed in the last interrupt, this will change depending on how fast a machine is being used to run the simulator, and how long it takes to process each event. The simulator is designed to take about 60% of the capacity of the processor it's running on.)

2.2.2 The Next Event, Next Log and Next Tx Buttons

The simulator can run in two modes: continuous and stepping. In step mode the simulator does one or more events, and then pauses, printing up in the events windows details of the event it has just done.

Pressing one of these three buttons puts the simulator in stepping mode (if it is not in stepping mode already), and proceeds to either the next event, the next event which would be logged, or the next event that transmits a packet.

2.2.3 The Run / Pause / Continue Button

For use in continuous running mode, pressing this button will start the simulator running if the simulation is at the start, or if already running, it will pause and continue the simulation.

At the end of the simulation time the button will be disabled until the simulation is reset.

2.2.4 The Reset Button

Press this at any time and the current simulation ends (if it is running), and the entire simulation is then reset. This means that all statistics and output information is lost.

2.2.5 The Real Time and Fast Real Time Checkboxes

If the Real Time checkbox is checked, the simulator will attempt to run in real time. This might be useful for demonstrating how the simulator works and watching packets being transmitting and hopping across multi-hop routes when debugging protocols. (If the computer running the processor is not powerful enough to complete simulator one second's worth of events in one second, then the simulation will just run as fast as it can.)

Fast Real Time is five times real time, if this checkbox is selected the simulator will attempt to do five second's worth of events every second. (It is impossible to check both the Real Time and the Fast Real Time boxes at the same time.)

2.2.6 The Animations Checkbox

When checked, the nodes display and the energy and packets received indicator bars will be constantly updated during the simulation. This however takes time, and the simulator will run faster if the animations are turned off, which can be done by unchecking this box.

When unchecked, the energy bars and packet received bars will only be set at the very end of the simulation.

2.2.7 The Show Routes Checkbox

When checked, placing the cursor on a node will display either the current routes to and from the selected node to the node with the mouse on it, or if there is no selected node, will display all the current routes to and from the node with the mouse on it. (See section 0 for more information about Selected Nodes.) Routes to are shown in blue, routes from are shown in plum (the colours were chosen to be easy to remember: "blue" is "to").

The information is taken from the in-built routeing tables, so if these are not being used, then the "no route available" indication (a dashed blue line direct to the destination) will always be shown.

If a route does exist to the destination in the in-built routeing tables, it will be shown as a solid line. If a partial route exists, the route will be shown as a solid line as far as it goes, followed by a dotted line to the destination. If a routeing loop is detected, then a dotted red line is shown from the loop to the destination.

2.2.8 The Show Parents Checkbox

Some MAC and LLC protocols establish groups or trees of nodes, structures that can be independent of the routes that packets take through the network. The simulator provides a way for these structures to be visible, by selecting this checkbox.

Checking this checkbox will show lines running from every node with a parent to its designed parent. Selecting a node while this checkbox is checked will indicate the parents of this node (all the way back to a co-ordinator node that does not itself have a parent node), and also all nodes that are children of the current node.

Note that currently none of the built-in protocols use this feature.

2.2.9 The Nodes Slider

The slider sets the number of nodes in the simulation. It is a copy of the slider on the "Setup Simulation" tab, changing one slider will automatically change the other one. The permitted values are a function of which node layout option is selected on the "Setup Simulation" tab.

(This copy has been put on the front page for convenience; it's easier to see what patterns are being generated by the hexagonal and square layout options in particular this way.)

2.2.10 The Simulation Time Slider

The slider sets the length of the simulation, in seconds. It is a copy of the slider on the "Setup Simulation" tab, changing one slider will automatically change the other one.

2.3 Logging Events

The logging section allows the user to select which nodes, which packets, and which events are logged in the simulation log. (Logging everything would result in very large files, and significantly slow down the simulator.)

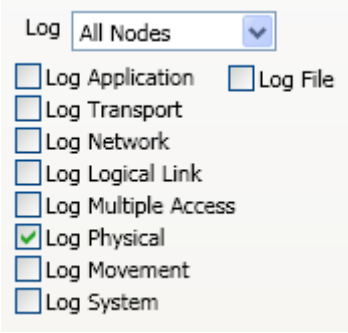


Figure 2-4 The Logging Controls

2.3.1 Selecting the Nodes / Packets for Logging

The combobox allows the choice of "All Nodes", "Node 0", "Node 0 and 1", "Packet 0", "Packet 0 and 1" and "Packet Non-Zero", to restrict the number of events that are logged to save time, and make the log files easier to read. "All nodes" saves information about all packets and all nodes; otherwise only the events at the nodes specified, or associated with the packets specified are logged.

(Data packets from the application layer start at packet number one, packets with number zero are usually packets generated by lower layer, for example routeing packets and packets for co-ordinating the multiple-access protocols).

2.3.2 Enabling a Log File

If this checkbox is checked then a log file is automatically generated by the simulator, and saved in whatever directory the simulator is running from. The log file contains the event descriptions of every logged event: the same logged events that can be seen in the "Log File" output in the "Output" tab of the main window.

The log files are given a name based on the time and date when the simulator is run, in the format LogFile_Year_Month_Day_Hour_Minute_Second_Millisecond. For example a file called "LogFile_2010_7_26_21_22_3_421.txt" would contain the record of every logged event in a simulator run that started at 22 minutes and 3.241 seconds after nine o'clock in the evening (21 hours), on the 26th of July, 2010.

Saving log files in this way can help with debugging, and ensures a complete record of runs for later analysis when running in batch mode, but it can significantly slow down the simulator, and produce a lot of large files which will need to be cleaned up afterwards.

2.3.3 The Log Selection Checkboxes

An additional way to specify what events are logged, to restrict the size of log files and speed up the simulator, is to check or uncheck the log selection boxes. Only events associated with the selected layer(s) of the protocol stack are logged. (If none of these boxes are checked, then no events will be logged, and any saved log file will be empty.)

The "Log System" checkbox logs events such as starting and stopping the simulator, and regularly updating the energy usage of the nodes.

2.4 Simulation Monitoring

The monitor window at the bottom of the main screen has three tabs: "Events", "Console" and "Interact".

2.4.1 The Events Tab

The "Events" tab shows details of the last event in stepping mode.

When not stepping through events, this tab shows a status message: at the start of the simulations (or when simulations are reset) this tab shows the current version number of the simulator; at the end if the simulations it just says "All done.".

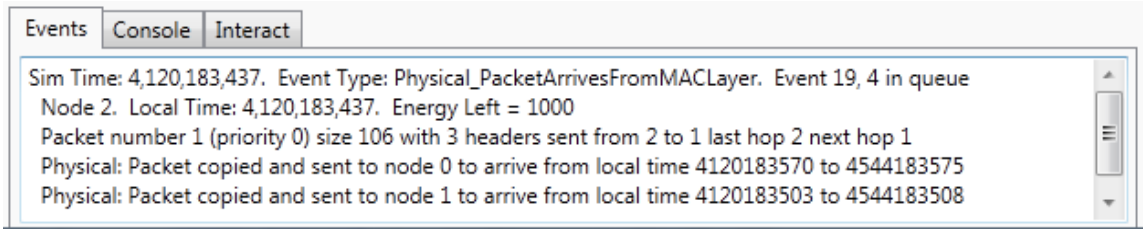


Figure 2-5 The Events Tab Showing a Current Event

2.4.2 The Console Tab

The console provides a space where user protocols can print status messages to help with debugging. Nothing in the simulator itself prints anything to the console.

2.4.3 The Interact Tab

The interaction tab allows the user to interrogate the status of the protocols during the simulation run. The query is typed into the upper textbox, the response is then printed below. The syntax is to type "X Y" where X is one of T (for transport), N (for network), L (for logical link) or M (for multiple

access), and Y is a number indicating the node. For example N 3 would query the network layer of node three.

User-defined protocols can override the default behaviour of the protocols, which are to print up the current contents of the transmit packet queue for the transport, logical link and multiple access layers, and print the current routeing table for the network layer.

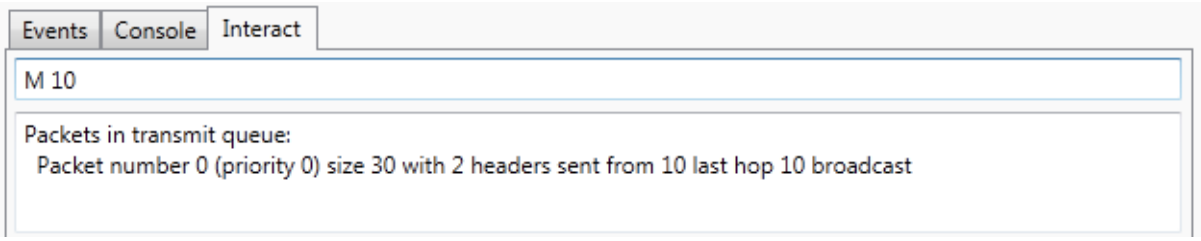


Figure 2-6 The Interact Tab

2.5 Nodes and Transmissions

The main display window shows the position of the nodes, and the range of any packet being transmitted.

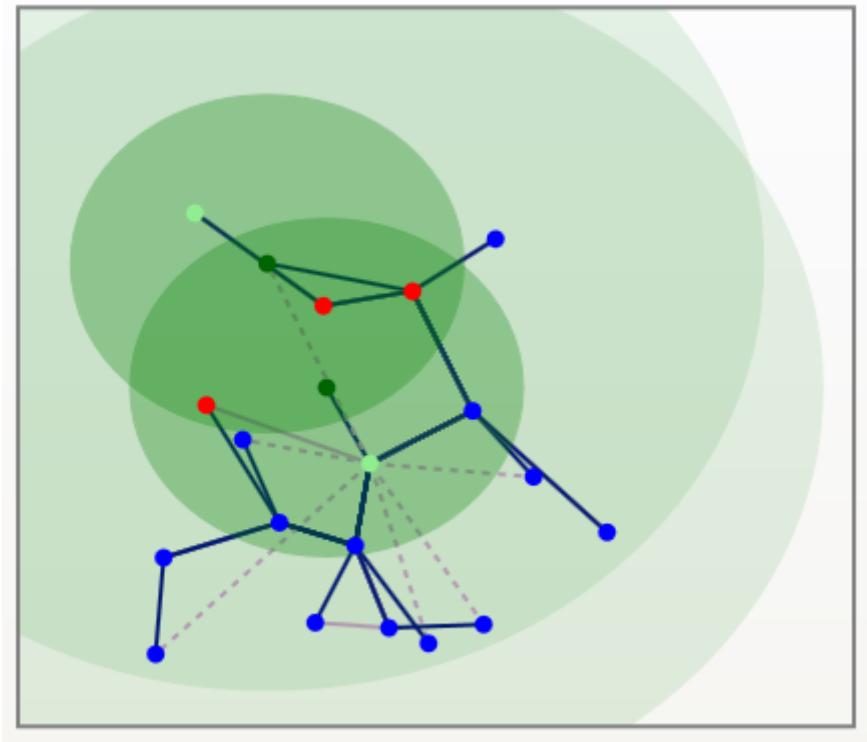


Figure 2-7 The Node Display

The status of the nodes is indicated by their colour:

Node Status	Colour
Actively Listening	Purple
Receiving	Light Green
Detecting a Transmission, but not Receiving	Blue
Unable to Receive due to Collision	Red
Transmitting	Dark Green
Asleep	Gray

Table 2-1 – Primary Node Colours

In addition, nodes can be several other colours when the mouse is placed over certain parts of the screen:

Condition	Colour
Highlighted by Mouse on Packet / Energy Bar	Orange
Source of Current Packet	Yellow
Destination of Current Packet	Gold

Table 2-2 – Secondary Node Colours

The two green circles around the nodes show the range in which the transmitting packet can be successfully received (the smaller, darker circle), and the range in which the transmitting packet can be detected (the larger, lighter circle). (It is usually possible to detect a transmission at a lower power than would be required to receive it correctly.)

The radius of the inner circle is a function of the transmit power and bit rate selected for each packet. The radius of the outer circle is a function of the transmit power and the detection threshold set by the physical layer for recognising transmissions (see section **Error! Reference source not found.**).

The diagram above shows the case when the "Show Routes" checkbox is checked, and two packets are currently colliding. The dotted plum lines indicate that no route is available from certain nodes.

Note that it is also possible to select a node by clicking on it, in which case a thin red line will appear around the node. To select a different node, just click on a different node; to unselect all nodes, click on whichever node was selected. If there is a selected node, then only routes to and from this node are shown when the "Show Routes" checkbox is checked.

2.6 Energy and Throughput

The Energy and Throughput bars give a visual indication of the amount of energy used and the number of packets received during the course of the simulation. The red line in the packets window shows the number of packets that would arrive at each node if packets were uniformly sent to all possible nodes, and all packets arrived safely.

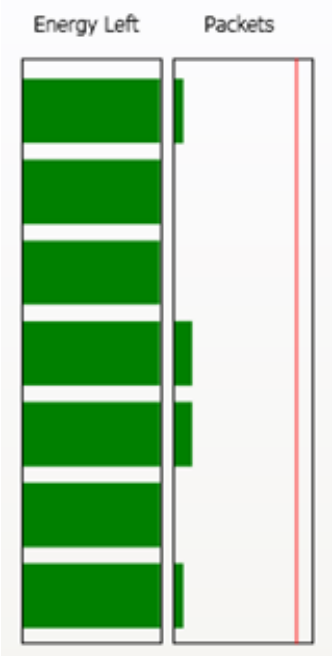


Figure 2-8 The Energy and Throughput Displays

Note that the "Packets" bar is not very useful when all packets are sent to the centre node, or node zero, as the bar tends to go off the screen very quickly.

Also, note that when the "Animations" box is not checked, these bars will not be updated until the end of the simulation.

2.6.1 Energy Used and Energy Left

Nodes can be in one of two energy modes, either counting down or adding up energy used. In the case of counting down, the energy bars start full, and reduce to zero, after which that node can do nothing more. In adding up, the nodes never run out of energy and stop working, but a red bar will appear at the nominal battery level, and the graph will automatically scale as more energy is used.

By default, nodes start with 1000 Joules of energy.

2.7 Node Status

The node status window shows the current status of the highlighted node: the node with the mouse cursor over it, or over its energy left or packet received bar.

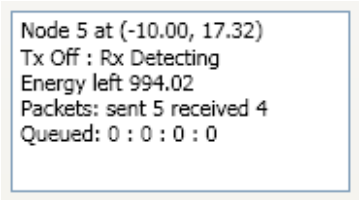


Figure 2-9 The Status Window

The status window gives the number and location of the node; the status of the transmitter and receiver; the energy used or energy left in the battery; the number of data packets generated at this

node and received at this node; and the number of packets currently waiting in the queues in the transport, network, logical-link and multiple-access layers respectively.

Reliable protocols will usually keep packets waiting in queues until an acknowledgement occurs, and many MAC protocols will delay transmitting packets, and have to store them until they can be sent.

The example above shows a node with co-ordinates (-10.0, 17.32) which is currently detecting a transmission, but is not receiving a sufficiently good signal to receive the packet. It has 994.02 Joules of energy left in the battery, has so far generated five packets and received four, and currently has no packets waiting in the queues at any protocol layer.

(Note: user protocols that do not use the built-in packet stores will not have the number of stored packets reflected in this window.)

3 Configuring the Simulation

The “Set-up Simulation” tab provides a screen where the current simulator parameters can be viewed and set, and configurations saved and loaded. There are nine control boxes grouping together functions that control different parts of the simulator: these control boxes are described in the following sections of this document.

3.1 Configuration Functions

Configurations and lists of packets can be saved and loaded from an XML format file; these XML files can then be modified by any text editor. If an attempt is made to load in a file with a parameter set to an invalid value, the program will warn the user, and then set the parameter to the closest possible valid value.

An automatically generated configuration file stores the value of every parameter that can be set up on the “Setup Simulation” and “Setup Protocols” tabs, but it does not include the settings of the logging or other checkbox options on the “Main” tab. Manually edited or produced configuration files do not have to include every parameter: any parameter not defined in the configuration file will retain its setting (which means it will be set to the default value unless it has been otherwise changed).

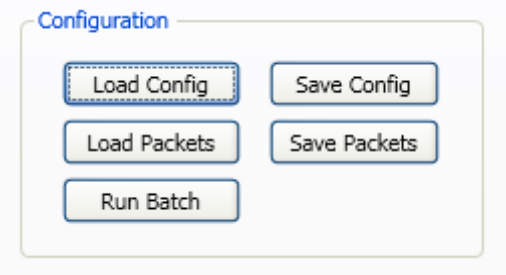


Figure 3-1 Configuration Controls

3.1.1 Adding Packets to the Configuration Files

A pre-defined sequence of packets can also be added to the XML configuration files. These are included in the following format:

```
<Packet>
 <Length>256</Length>
 <Priority>2</Priority>
 <Source>0</Source>
 <Destination>4</Destination>
 <Time>1.5e9</Time>
</Packet>
```

Any number of packets can be included in either the main configuration file, or a packet-specific configuration file loaded after the main configuration file (it has to be loaded after the main configuration file, as loading a configuration file clears any existing packets from the list to be sent).

The packet length is given in bytes; the priority must be an integer; the source and destination are numbers from zero to the number of nodes minus one; and the time is a double, measured in seconds, and indicates when the packet is generated by the application layer.

If the source or destination node indicated does not exist, they will be set to node zero.

Saving a configuration will also save any packets previously loaded in from a configuration file, but not any packets that have been randomly generated during the simulation run. To save randomly generated packets, use the "Save Packets" function (see below).

A set of packets can also be saved to a packets file which contains only packets: this is done using the “Save Packets” button.

3.1.2 Loading and Saving Packets

The "Load Packets" function will accept an .xml file of the same format as a configuration file, but it will only load in the packets from the specified .xml file - all other parameters are left unchanged. These packets will then be stored and inserted into the simulation and the specified times.

Note this function does not append a new set of packets to the list of packets waiting to be sent into the simulation: it resets this list first. If you want to combine two sets of packets to be run in the same simulation, you'll have to edit the two .xml files into one bigger file.

The "Save Packets" function will write an .xml file containing details of all of the packets sent during the simulation: both randomly generated during the simulation and preloaded from a configuration file.

3.1.3 Configuring the Random Number Seed

By default, DANSE generates a new series of random numbers every time it is run, by using the system clock to produce a new seed. However, it is possible to force DANSE to use the same set of random numbers for each simulation; this can be useful to compare the results of two slightly different protocols.

To do this requires the addition of another line into the .xml config file, which should look like this:

```
<Random_Seed>345</Random_Seed>
```

where in this case 345 will be used as the seed for the random numbers from then on. The seed should be a positive integer.

3.2 Running Batch Files

The “Run Batch” button allows a series of simulations to be run with different values for one or more parameters, with the results saved to files for later analysis. Note that in batch mode the simulator is configured to run as fast as possible, so the animations are turned off, and the packets and energy bars on the main screen are not updated.

Batch files are also XML files, and can contain elements that change various parameters of the simulator as well as specifying the name of the files that the outputs (in ASCII text format) should be saved to. An example batch file is shown below:

```
<?xml version="1.0"?>
<BatchFile>
 <Number_of_nodes>20</Number_of_nodes>
 <Run>TwentyNodes</Run>
 <Number_of_nodes>30</Number_of_nodes>
 <Run>ThirtyNodes</Run>
</BatchFile>
```

This file will run the simulation twice, once with twenty nodes, and once with thirty nodes, but otherwise will leave all the parameters unchanged. The raw results of the first simulation will be saved to the files "TwentyNodes_packets.txt" and "TwentyNodes_nodes.txt", and of the second simulation to the files "ThirtyNodes_packets.txt" and "ThirtyNodes_nodes.txt". All files will be saved into the same directory that contains the batch file being run.

The output format of these files are the same as the output formats used in the display window in the "Output" tab of the simulator, and are described in the sections below.

3.2.1 The Packets Output File

The first output file, which will be named something like "MyOutput_packets.txt" is a list of numbers in an array with eight columns: the packet number; the packet priority; the packet length; the source node; the destination node; the number of hops; and the start time and the end time, both in nanoseconds.

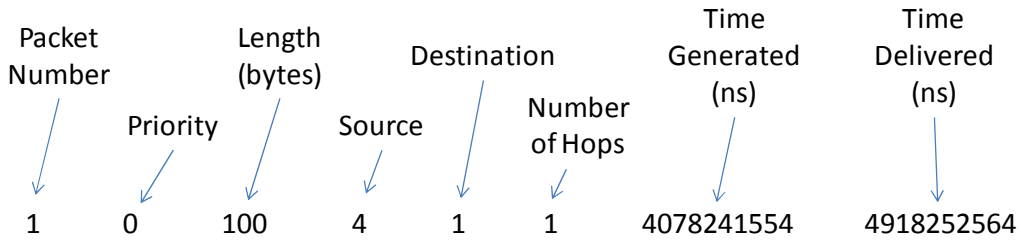


Figure 3-2 Fields in the Raw Packets Output

The "Start Time" is the time the packet was generated by the Application Layer, and the "End Time" is the time the packet successfully arrived at the Application Layer of the destination; both times are in nanoseconds. If the packet never arrived (or did not arrive by the end time of the simulation), the End Time is set to -1.

For example, a short file might look like this:

1	0	100	4	1	1	4078241554	4918252564
2	0	100	5	1	1	6351369231	-1
3	0	100	0	4	2	11528608436	-1
4	0	100	0	3	1	25212740940	26052751950
5	0	100	4	0	1	33962037731	-1
6	0	100	3	3	0	36134344872	36134348872

Figure 3-3 Sample Packet Output File

which shows six packets being sent in the first 50 seconds, three of which have failed to arrive. All packets were sent with a priority of zero, and have a length of 100 bytes. The first packet was generated at time 4.078241554 seconds, and was sent from node four to node one; it arrived at time 4.918252564 seconds after one hop. (Note that these times are global simulation times, not the times given by the individual clocks at the nodes.)

Packet six has taken zero hops, since node three is sending this packet to itself: it doesn't need to be transmitted over the radio interface at all.

3.2.2 The Nodes Output File

The second output file, which will be named something like "MyOutput_nodes.txt" is a list of numbers in an array with six columns: node number; x- and y- co-ordinates; the number of data packets sent; the number of data packets received; and the energy left or energy used (depending on the energy mode).

For example, a short file might look like this:

0	-10.00	-17.32	11	7	975.252
1	10.00	-17.32	11	7	975.082
2	-20.00	0.00	9	9	974.986
3	0.00	0.00	18	10	974.448
4	20.00	0.00	16	9	975.075
5	-10.00	17.32	22	4	974.948
6	10.00	17.32	9	9	974.864

Figure 3-4 Sample Nodes Output File

which shows the results of a seven node simulation, using a hexagonal layout. The centre node (node three) with co-ordinates (0.0, 0.0) has generated eighteen packets, received ten, and ended the simulation with 974.448 Joules of energy left in its battery.

In the case where the node has been moving, the co-ordinates given represent the location of the node at the end of the simulation. The packet counters count packets sent by and received by the application layer: packets sent by other layers (for example acknowledgements and routeing packets) are not counted.

3.2.3 Running Batch Files from Command-Line

It is also possible to run batch files from the command-line, so that the whole simulator can be run in a batch file or from another program. In this case, it will close as soon as it has finished running the simulations.

Just put the names of the batch files in .xml format (note the filenames must end in .xml or the simulator won't recognise them) as command-line options. If there is at least one .xml file included as a command-line option, the simulator will run the batch files, generate the relevant output files, and then exit.

3.3 System Functions

The system functions box contains the controls for setting how many nodes there are, what patterns they are in, where they choose to send packets, how long the simulation lasts for and for what proportion of this time the application layer attempts to send packets, and whether the clocks in all of the nodes are synchronised or not.

The system control box is shown in Figure 3-5; the individual controls are described in more detail below.

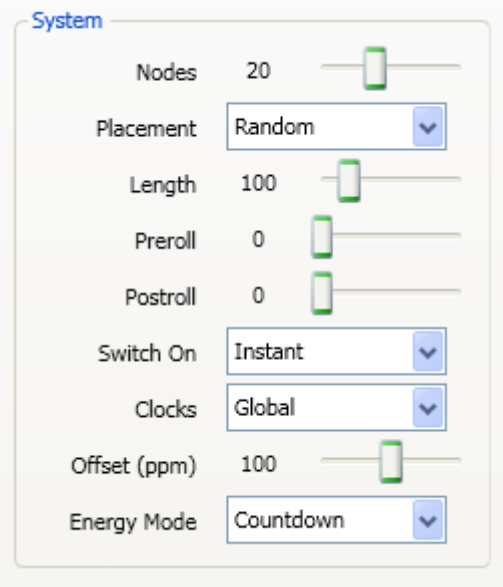


Figure 3-5 System Setup Controls

3.3.1 The Number of Nodes

The number of nodes can be set to between one and a hundred. Not all numbers are allowed. The possible numbers depend on the placement mode selected: for example in random mode the number of nodes must be a multiple of two between six and ten, a multiple of five between ten and fifty, and a multiple of ten between fifty and one hundred; in square mode there are a limited selection of symmetrical patterns with between one and one hundred nodes; in hexagonal mode there are a limited selection of hexagonal patterns with between one and ninety-one nodes.

3.3.2 The Placement of Nodes

Nodes can be placed either randomly, in a circle around node number zero, in a ring without a centre node, in a straight line, or in square, cross or hexagonal patterns. The total area available is a square with sides of length 200 metres.

Random placements are not entirely random: the idea is that it should be impossible to have all nodes except one in one corner of the screen, and the final node in the other corner, which could make it impossible to get a packet to or from this final node. It's also considered unrealistic to have two nodes very close together. So, the first node is always placed close to the centre of the area, then other nodes are placed at least 10 metres from any other node, and at most 30 metres from any existing nodes.

Note that these maximum and minimum separations are only for the initial placement of the nodes; if the nodes start to move then these minimum and maximum separations can be (and often are) exceeded.

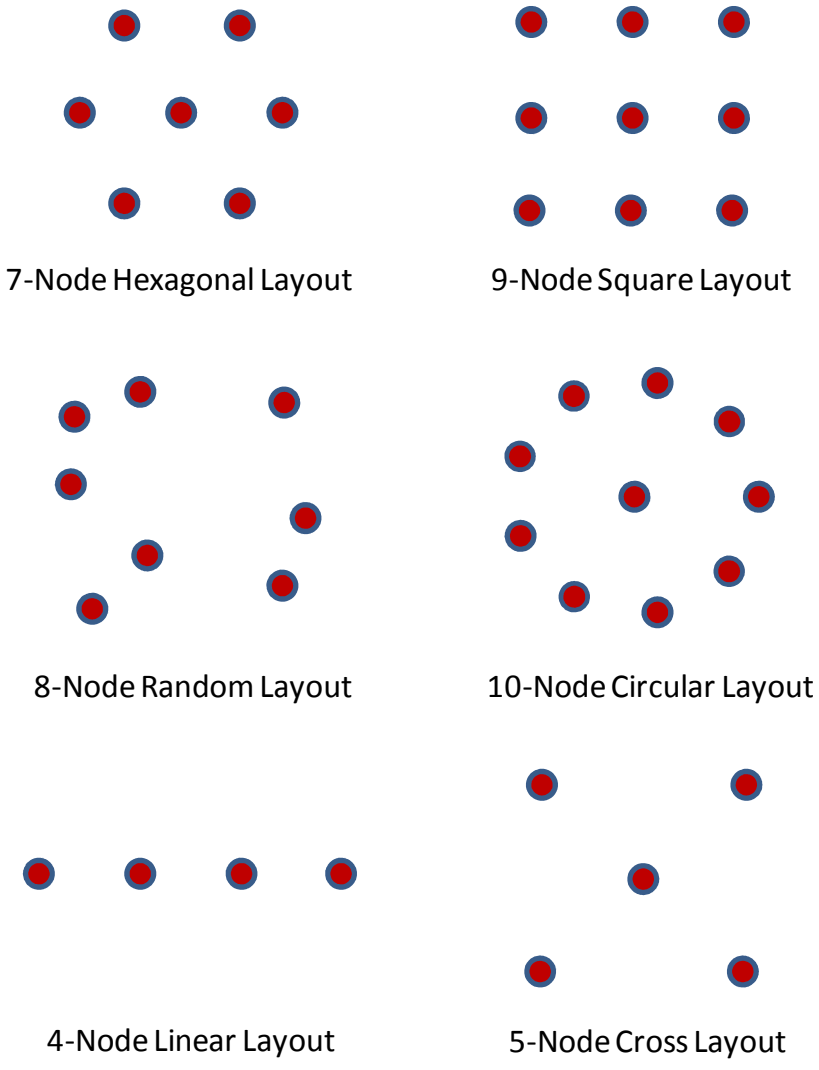


Figure 3-6 Illustration of Typical Node Layouts

In a circular placement, the first node (node zero) is placed in the centre of the area, and all other nodes arranged equally spaced around a circle of radius 30 metres. This placement is most commonly used when all packets are being sent to node zero, when it allows an easy comparison with theory. There is a also a ring placement, which is similar, except that there is no centre node.

Square placements have the rows and columns of the square a distance of 20 metres apart. Hexagonal placements have the distance between nearest neighbours set to 20 metres, except in the case of the 91-node hexagon, where the distance is slightly less in order to fit all of the nodes into the overall 200-metre square area.

Linear and cross layouts have nodes spaced 20 meters apart, except when there are more nodes than would fit in the total area available, in which case the space between nodes is reduced.

3.3.3 The Length of the Simulation

This slider sets the total time of the simulation (including any preroll and postroll times) in seconds. It can only be set to certain values (10, 20, 50, 100, 200, 500, 1000 etc), from 10 seconds to 100,000 seconds (about 27 hours and 47 minutes).

3.3.4 The Preroll Time

The preroll time is the time at the start of simulation before the application layer starts generating any packets. It can be useful to have a preroll period if you're using a pro-active routeing protocol that attempts to establish a set of good routes to use or a MAC protocol that works out what transmit power is required to reach every other node, before any packets are sent.

3.3.5 The Postroll Time

The postroll time is the time at the end of the simulation during which no new packets are generated by the application layer: only packets already generated will be forwarded by routers and re-transmitted by reliable layers. This feature is provided to prevent packets generated at the end of simulation period being counted as lost when in fact they just haven't had a chance to be delivered yet.

3.3.6 The Switch On Mode

The switch on combobox specifies when, and in what order, the nodes are first switched on. Some protocols specify that a node should transmit something when it is first switched on (to announce its presence, or register with a co-ordinator node), and if all nodes switch on at the same time, this results in a rather unrealistic burst of packets, many of which collide with each other.

To address this problem, the simulator has three different wakeup modes: "Instant" switches on all nodes at the start of the simulation; "Sequence" switches them on one-by-one in order of node number at regular intervals throughout the preroll time; and "Random" switches each node on at a random time during the preroll time, although in this case node zero is always turned on at time zero.

3.3.7 Synchronisation of Clocks

The "Clocks" parameter can be set to either "Global" or "Individual". When global clocks are selected, every node is assumed to have access to the same global, perfectly accurate clock (for

example, provided by a GPS receiver). In practice this is rare, and it is more usual for nodes to run individual clocks at slightly different speeds, and keep their own time.

Setting the “Clocks” parameter to “Individual” forces each node to use their own internal clocks, which are not synchronised, and which run at slightly different speeds. The maximum inaccuracy of these clocks is set by the “Offset ppm” parameter, in parts per million (ppm). For example, with “Offset ppm” set to 10000, the clocks in the nodes can be running up to +/– 1% out from the correct speed.

3.3.8 Energy Mode

The simulator keeps a note of how much energy is used by the physical layer of each node, and this can be done in one of three modes: "Countdown", "Addup" and "Down Not Zero". In "Countdown" mode, the nodes countdown from their initial energy, and when their energy reaches zero the node is not permitted to receive or transmit any more frames.

In "Addup" mode the nodes never run out of energy, they just keep a note of how much energy they have used. "Down Not Zero" mode is equivalent to "Countdown" mode for all nodes except node zero, which never runs out of energy; the idea is that this mode can be used for systems with a base-station or access point which has much larger reserves of energy.

3.4 Movement Functions

The movement controls are shown in Figure 3-7, they control the speed of the nodes, and how they move.

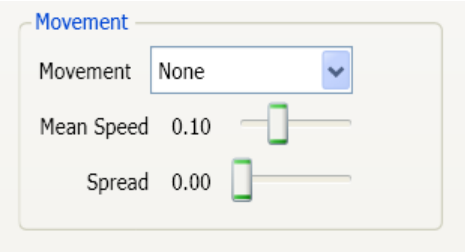


Figure 3-7 Movement Setup Controls

The movement type can be set to “None”, “Linear” or “Targetted”. If set to “None”, the nodes don’t move, the settings of the mean speed and spread are ignored; otherwise the speed of each node is calculated as:

$$Speed = Mean\,Speed \times \left(1 + 2\left(R - 0.5\right) \times Spread\right) \quad (1.1)$$

where *R* is a uniformly-distributed random number between zero and one. This means that with “Spread” set to zero, all nodes have the same mean speed; with “Spread” set to one, nodes are equally likely to have any speed between zero and twice the mean speed.

The units of speed are metres per second, and the location of users is updated twice every second.

If the movement type is set to “Linear”, each node is given a random direction and speed, and sets off in that direction at the start of the simulation, bouncing off the boundaries of the 200 meter

area. If the movement type is set to “Targetted”, then a random position in the 200 meter square area is chosen for each node, and each node sets off towards its target. When the target is reached, a new target is generated, and the node sets off towards the new target with a new random speed, and so on. This method is often considered to more accurately simulate the movement of real nodes.

Note that using movement slows down the simulator, since it has to re-calculate all the received power and interference levels every time the nodes mode. (If a node moves out of range during reception of a packet, the packet is not received.)

4 The DANSE Protocol Stack and Built-in Protocols

The protocol stack used by the simulator is shown in the table below:

Application Layer	Sources and receives information packets
Transport Layer	End-to-end flow and error control
Network Layer	Routeing
Logical-Link Layer	Hop-by-hop flow and error control
Multiple-Access Control (MAC) Layer	Multiple access, power control
Physical Layer	Transmission and reception of packets, collisions and interference

Table 4-1 The DANSE Protocol Stack

The application layer and physical layer are fixed and not accessible to users, they are part of the simulator itself. Protocols for the other four layers can be written and optimised by the users, although there are several simple protocols for these layers built into the simulator itself.

4.1 The Application Layer

The application layer sources data packets, as well as receiving them and updating the statistics (including the number of packets received and average delays). The user controls allow different distributions of packet sizes and times between packets to be selected.

Packets are sent to the transport layer with three additional pieces of information: the source port (a number between zero and 254), the destination port (another number between 0 and 254) and the address of the destination (a number between zero and $N - 1$, where N is the number of nodes in the simulation).

The controls for the application layer are on the Setup Simulation tab, and they are shown in the figure below:

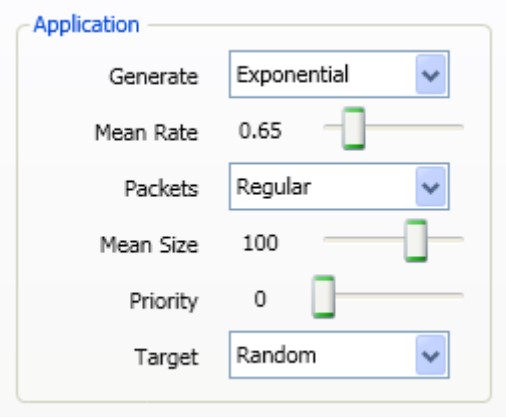


Figure 4-1 The Application Layer Controls

4.1.1 The Generate ComboBox

The options for when packets are generated can be set to “Exponential”, “Regular”, “Sequence” or “Uniform”.

The “Exponential” option generates packets such that the probability distribution of the times between packets is a negative exponential distribution: this reflects an ideal situation where packets are generated entirely randomly by all the nodes, so that the probability of generating a packet in the next second is constant.

The “Regular” option generates packets at regular intervals, so that the time between packets being generated is constant. Since all nodes start generating packets at the same time, this can result in a very large number of collisions. This is not very realistic, but it is good for testing multiple access protocols.

The “Sequence” option is similar to the “Regular” option in that packets are generated at regular intervals, however an offset is added to each node so that they start to generate packets in turn, avoiding collisions (provided the mean rate of packets is low). This is equally unrealistic, but can be useful for testing.

(Note that while both “Regular” and “Sequence” options start off with packets being generated by the nodes at synchronised times, if the System “Clocks” parameter is set to “Individual”, this synchronisation will slowly be lost as the clocks in the different nodes drift apart.)

With the “Uniform” option, packets are generated so that the time between packets is a uniform random variable between zero and twice the inverse of the mean rate. It’s not particularly representative of any real situation, it’s just easy to program.

4.1.2 The Mean Rate Slider

The mean rate slider can be set to any number between zero and three, and sets the average number of packets generated per second. These packets are in addition to any packets defined in the configuration or packet files.

The mean rate sets the rate of packet generation for the simulator as a whole, so for example setting a mean rate of one packets per second with four nodes would result in each node sending a packet on average once every four seconds. The mean rate can be set between zero and three packets per second, with the default value set to 0.1 packets per second.

4.1.3 The Packets ComboBox

The packets combobox determines the distribution of the size of the packets, and can be set to “Regular”, “Uniform” or “Exponential”.

The options for the distribution of packet sizes are similar to the options for the distributions of packet rates: exponential, regular and uniform. “Exponential” gives a negative exponential distribution of packet sizes, “Uniform” gives a uniform distribution of packet sizes chosen between one byte and twice the mean length minus one, and “Regular” forces all packets to be the same size.

4.1.4 The Mean Size Slider

The mean size of packets can be set to between 1 and 1,000 bytes. Note that the bit rate of the transmissions on the physical layer is by default 1000 bits per second, so a 1,000 byte packet will take at least 8 seconds to transmit (and possibly slightly more depending on the size of the headers added by the lower layers).

4.1.5 The Priority Slider

The priority slider can be set to any integer between 0 and 7, and determines the priority of any packet generated. None of the built-in protocols considers the priority of packets, so unless priority-sensitive user protocols are being used, this control has no effect.

4.1.6 The Target ComboBox

The target combobox determines where the packets are sent. The options are “Random”, “Centre”, “Zero” and “Random Other”. “Random” just selects any node (including the source node) at random; “Random Other” selects any other node (not including the source node) at random, “Centre” sends all packets to the node nearest the centre of the coverage area, and “Zero” sends all packets to node number zero.

Setting this parameter to “Random Other” prevents a node trying to transmit a packet to itself, so that choosing “Random Other” when there is only one node in the simulation will generate an error message when the simulation is run.

In cases such as the square array when there can be four nodes equally close to the centre of the pattern, and the centre node is chosen as the destination for all packets, one of these four middle nodes is chosen (usually the one on the top left).

4.2 The Transport Layer

The transport layer provides end-to-end flow and error control. A basic best-effort and reliable protocol come pre-defined with the simulator.

Packets are delivered to the transport layer from the application layer together with a destination address and a source and destination port number; all are numbers between zero and 254.

The built-in best-effort transport layer adds a header containing a source port and destination port field, each eight-bits long, but that’s all. The built-in reliable transport layer adds a header containing the source port and destination port, an eight-bit sequence number, and a Boolean flag indicating whether the packet is an acknowledgement or not, stored in an eight-bit flags field. The headers are shown in the figures below.

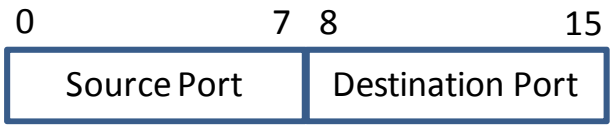


Figure 4-2 The Best-Effort Transport Layer Header

Figure 4-3 The Reliable Transport-Layer Header

Packets are passed to the network layer with a single additional piece of information: the destination address.

The transport layer protocols are controlled by the transport box, which is shown below:

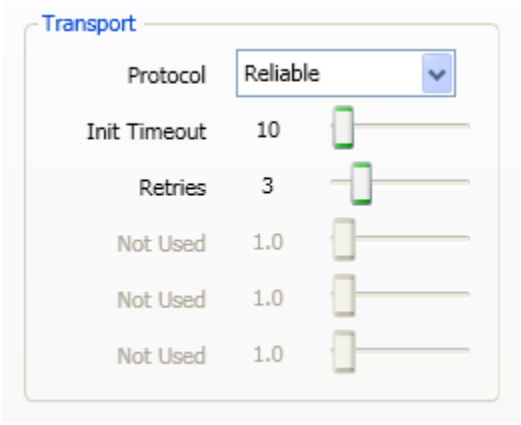


Figure 4-4 The Transport Control Box

4.2.1 The Protocol ComboBox

The protocol combobox provides the choice of "Reliable", "Best Effort" or "User", and selects between the reliable, best-effort and user-defined protocols. The best-effort protocols do not have any additional controls, the other controls specified here are for the reliable protocols.

4.2.2 The Init Timeout Slider

The initial timeout slider can be set to any value between one and 100 seconds, and determines how long the reliable layer waits for an acknowledgement before attempting to re-transmit packets.

4.2.3 The Retries Slider

The retries slider can be set to any integer between one and ten and determines how many attempts the transport layer makes to deliver a packet before it gives up.

4.3 The Network Layer

The network layer provides the routeing function. Four protocols are provided: a basic direct routeing protocol (everything is sent directly to the end-user in one hop); flooding; a simple Bellman-Ford routeing scheme, and an on-demand routeing scheme.

Packets are sent to the logical-link layer with the additional information of the next hop destination (for direct routeing this will be equal to the final destination of the packet).

4.3.1 The Direct Network Layer

When using direct routeing, the network layer adds a simple two-byte header that contains the original source and final destination only. (These will be the same as the MAC-layer source and final destination as well.)

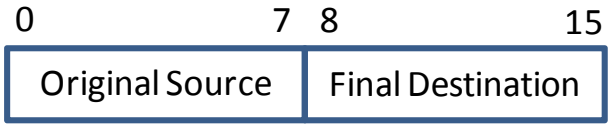


Figure 4-5 The Direct Routeing Header

4.3.2 The Flooding Network Layer

Flooding is a simple but often inefficient routeing protocol (in the sense that it requires a lot of energy to deliver a packet). All packets are broadcast, and any packets received by any node which is not the final destination are re-broadcast. At each transmission a field in the header "Hops Left" is decremented; when this field reaches zero the packet is not re-transmitted (this prevents packets going round in circles forever).

When using flooding, a header is added containing the original source and final destination (both as eight-bit numbers), and a hop count (also stored as an eight-bit number), as shown in the figure below.

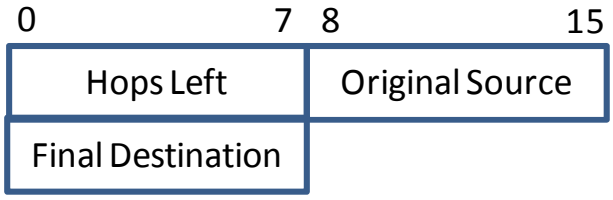


Figure 4-6 The Flooding Network Layer Header

The flooding control box is shown below:

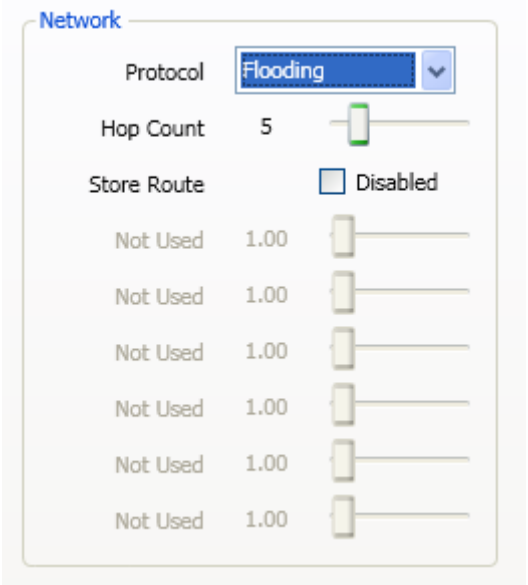


Figure 4-7 The Flooding Network Layer Control Box

There are only two parameters: hop count and the store route checkbox. The hop count sets the maximum number of hops a packet can make before being destroyed; store route (if checked) adds

the address of every node the packet has been through to the packet itself, so that a node, when receiving a packet it has seen before, can destroy it rather than re-transmit it. Both parameters are designed to save network energy and bandwidth by stopping packets from going round in circles.

4.3.3 The Bellman-Ford Network Layer

The Bellman-Ford protocol is a proactive distance-vector routeing protocol, and operates by each node broadcasting its routeing table at regular intervals. The built-in implementation has three parameters; the control box is shown in the figure below:

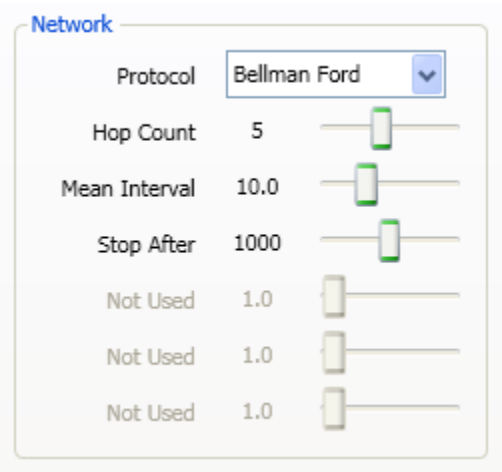


Figure 4-8 The Bellman-Ford Network Layer Control Box

"Hop Count" can be set to any integer between one and ten, and controls the maximum number of hops that a packet can take before it times out.

"Mean Interval" determines how often the routeing table is transmitted. The time between broadcasts of the routeing table is a uniform distribution, with a mean value that can be set between one and 1000 seconds.

"Stop After" sets a time after which no more routeing packets are transmitted. The idea is that in fixed networks, the routeing tables will not change once they are set up, and any further transmission of routeing packets is a waste of energy.

The Bellman-Ford algorithm requires a cost function to be associated with all hops, this implementation uses a cost given by:

$$Hop\_Cost = \max\left(1, \frac{1}{10^4 \times rx\_Power(\mathrm{W})}\right) \tag{1.2}$$

4.3.4 The On-Demand Network Layer

The On-Demand protocol is a reactive routeing protocol, and operates by attempting to find a route only when a node has a packet to transmit. It does this by broadcasting explorer packets which are passed on by every node that receives them, and these explorer packets collect the address of the nodes they have travelled through (so they get longer as they travel).

When the destination receives an explorer packet, it replies back along the path the explorer packet took.

The control box is shown in the figure below:

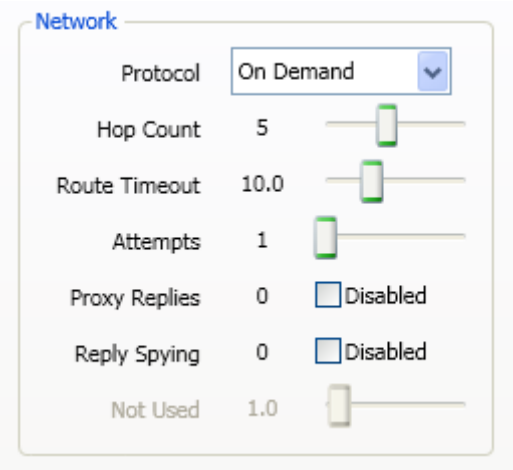


Figure 4-9 The On-Demand Network Layer Control Box

This protocol has five parameters:

"Hop Count" can be set to any integer between one and ten, and controls the maximum number of hops that a packet (including an explorer) can take before it times out.

"Route Timeout" can be set to any value between one and 10,000 seconds, and acts as a timeout for the routes. After this time, the route is forgotten, and has to be rediscovered.

"Attempts" can be set to any integer from one to ten, and controls how many explorer frames a node transmits before it gives up waiting for a reply and assumes that no route exists.

"Proxy Replies" if checked, allows an intermediate node, if it knows a route to the final destination, to send a reply to the explorer packet containing its best known route to the destination. If unchecked, all explorer packets have to travel to the final destination before a reply is sent.

"Reply Spying" if checked, allows other nodes along the path to look at the explorer replies, and store the best routes for future use, even when they do not have a packet to transmit to that destination.

4.4 The Logical-Link Layer

The logical-link layer provides hop-by-hop flow and error control. A basic best-effort and reliable protocol come pre-defined with the simulator, anything else must be written as a user routine.

Packets are delivered to the logical-link layer from the network layer together with a next hop address (which can be set to the defined constant BROADCAST for broadcast packets).

The built-in best-effort logical-link layer does not add a header at all; the built-in reliable logical-link layer adds a header containing an eight-bit sequence number and one flag stored in an eight-bit flags field to indicate whether this is an acknowledgement or not, as shown below.

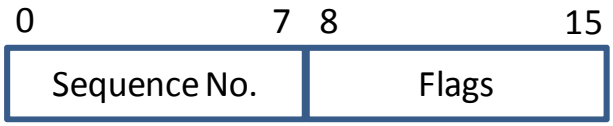


Figure 4-10 The Reliable Logical-Link Header

Packets are then delivered to the multiple-access control layer along with the next hop address.

The logical-link layer protocols are controlled by the logical link box, which is shown below:

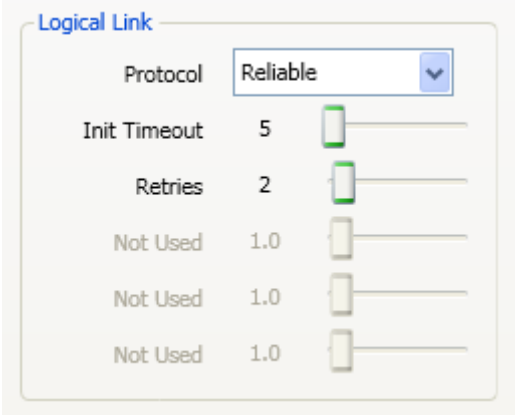


Figure 4-11 The Logical-Link Control Box

4.4.1 The Protocol ComboBox

The protocol combobox provides the choice of "Reliable", "Best Effort" or "User", and selects between the reliable, best-effort and user-defined protocols. The best-effort protocols do not have any additional controls, the other controls specified here are for the reliable protocols.

4.4.2 The Init Timeout Slider

The initial timeout slider can be set to any value between one and 100 seconds, and determines how long the reliable layer waits for an acknowledgement before attempting to re-transmit packets.

4.4.3 The Retries Slider

The retries slider can be set to any integer between one and ten and determines how many attempts the logical-link layer makes to deliver a packet before it gives up.

4.5 The Multiple-Access Control Layer

The multiple-access control (MAC) layer decides when to send packets, at what bit rate, and with what power. Also note that the next-hop destination and source are stored within the MAC header, rather than the logical-link header, since this is how things are usually done on the Internet.

Three multiple-access protocols are built into the simulator: ALOHA, CSMA and Polling.

Packets are delivered to the multiple-access layer from the logical-link layer together with a next hop address (which can be set to BROADCAST for broadcast packets). Packets are then delivered to the physical layer with an optional indication of the bit rate and the transmit power to use when

transmitting the packet; if these optional parameters are not set, then the default parameters taken from the physical layer control box are used (see the next section for more details). None of the built-in protocols specify these optional parameters, they all use the default transmit power and bit rate for all frames transmitted.

4.5.1 The ALOHA MAC Layer

ALOHA is the simplest possible MAC layer. In its simplest form it will attempt to transmit a packet as soon as it receives a packet from the logical-link layer above; the only possible delay occurs if the node is already transmitting a packet, in which case it will wait until the current transmission is finished before attempting to send the next packet.

In DANSE, the ALOHA scheme has one additional parameter: the "Initial Max Wait" parameter. If this is set to a non-zero value, then all frames arriving at the MAC layer from the logical link layer are delayed by a random amount chosen from a uniform distribution between zero and the maximum wait time specified by this parameter. The units of the waiting time are seconds.

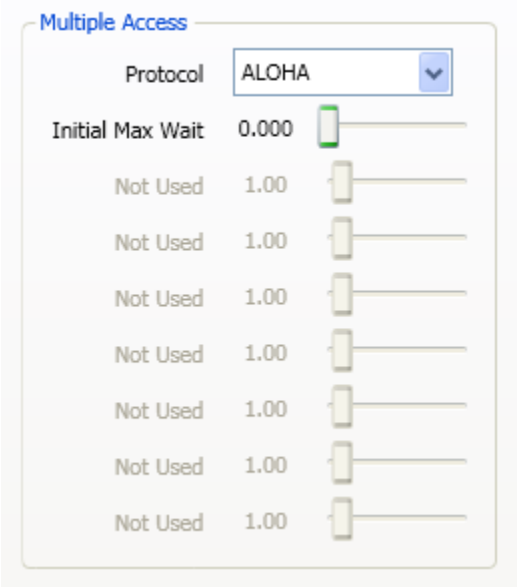


Figure 4-12 The ALOHA Multiple Access Scheme Control Box

This additional wait time can help avoid collisions when a higher layer protocol results in several nodes wanting to transmit at the same time (for example in flooding routeing).

The header used by the ALOHA multiple-access control scheme is shown in the figure below:

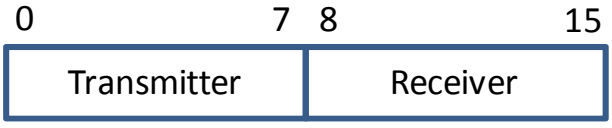


Figure 4-13 The ALOHA and CSMA MAC Header

This header contains no information other than the transmitter and receiver addresses.

4.5.2 The CSMA MAC Layers

The basic idea of Carrier-Sense Multiple Access (CSMA) is that a node listens to the channel first, and only starts to transmit a frame if the channel is clear (in other words the node cannot detect another packet already being transmitted).

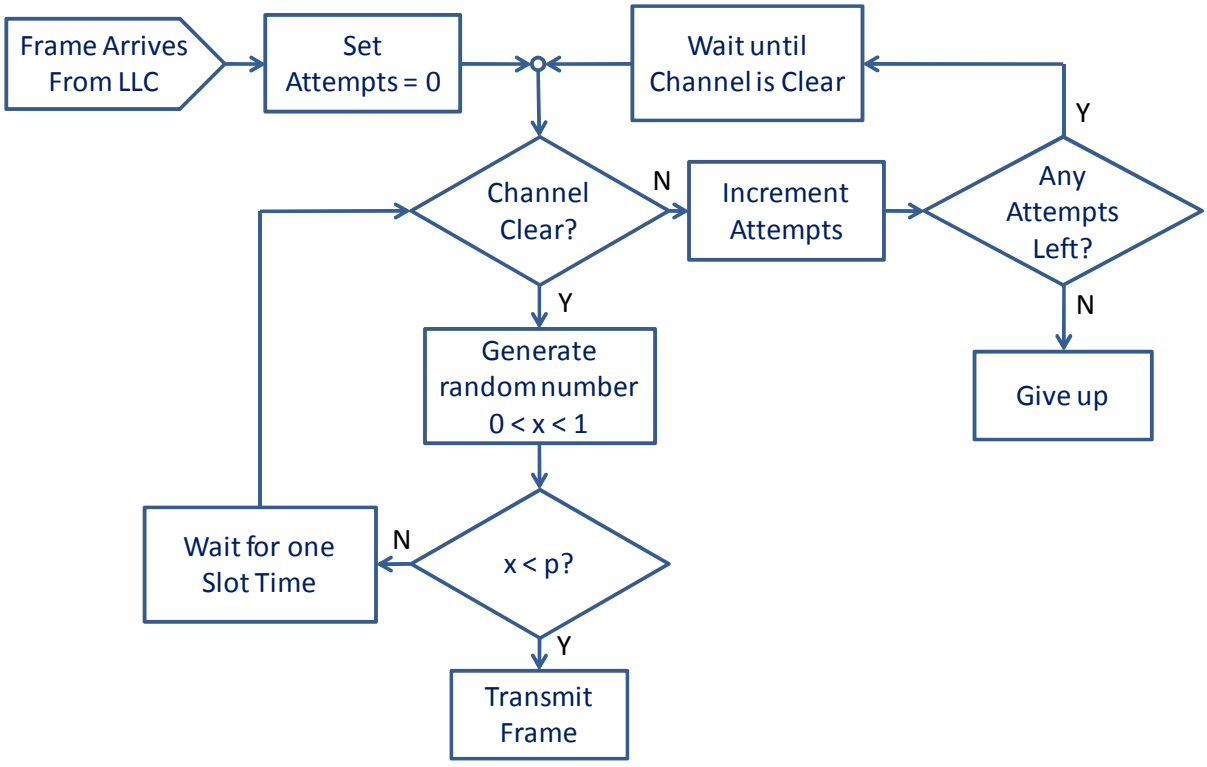
There are several forms of CSMA; the two implemented in DANSE are p-persistent CSMA and non-persistent CSMA. You can select between them by checking or unchecking the "Persistent" checkbox in the CSMA Multiple Access control box.

Both forms of CSMA use the same MAC header as the ALOHA MAC protocol.

4.5.3 The p-Persistent CSMA MAC Layer

p-Persistent CSMA works by waiting for the channel to become clear (if it is not already clear), and then generating a random number between zero and one and comparing this random number to the value of p. If the random number is less than p, the frame starts to be transmitted; if the random number is greater than p, the node waits for a slot time, then starts the process again: either waiting for the channel to become clear (if someone else has started to transmit) or generating another random number between zero and one and transmitting if the number is less than p, or waiting for another slot time if not.

A simplified[1] flow diagram of the operation of p-Persistent CSMA is shown below:



[1] Simplified, since it does not include the case where the transmitter is already transmitting or attempting to transmit a different frame when the frame arrives from the LLC. In these cases, the new frame is stored in a buffer until the earlier frames have either been successfully transmitted or the CSMA process has given up on them, and the minimum interframe space has expired.

Figure 4-14 Simplified Operation of p-Persistent CSMA

The CSMA control box (with the Persistent checkbox checked) is shown below:

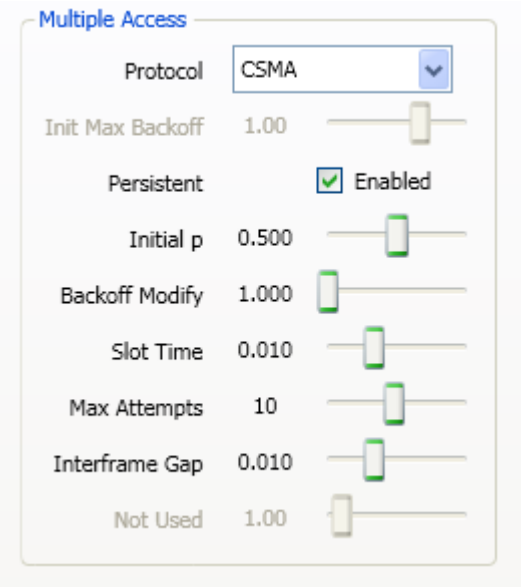


Figure 4-15 The p-Persistent CSMA MAC Control Box

"Initial p" sets the value of p for the first attempts to send the packet. (If "Backoff Modify" is set to 1, this value never changes.)

"Backoff Modify" attempts to intelligently change the behaviour of the algorithm depending on channel conditions. The value of p is set to the "Initial p" value at the first attempt to send a frame, but for each failed attempt (when another node starts transmitting first), the value of p is divided by this "Backoff Modify" coefficient. This means that the probability of starting to transmit a frame will decrease when the network is busy.

"Slot Time" defines the time that the node waits between generating random numbers while the channel is clear; it is measured in seconds.

"Max Attempts" sets the maximum number of attempts to transmit the frame before the algorithm gives up.

"Interframe Gap" sets the minimum time that a node waits after sending one frame before attempting to send another frame; this is measured in seconds.

4.5.4 The Non-Persistent CSMA MAC Layer

Non-Persistent CSMA in DANSE works by first waiting for a random time, then sensing the channel when a new frame arrives to be transmitted, and then starting to transmit immediately if the channel is clear. If the channel is not clear, the node starts the process again.

A simplified[2] flow diagram of the operation of Non-Persistent CSMA is shown below:

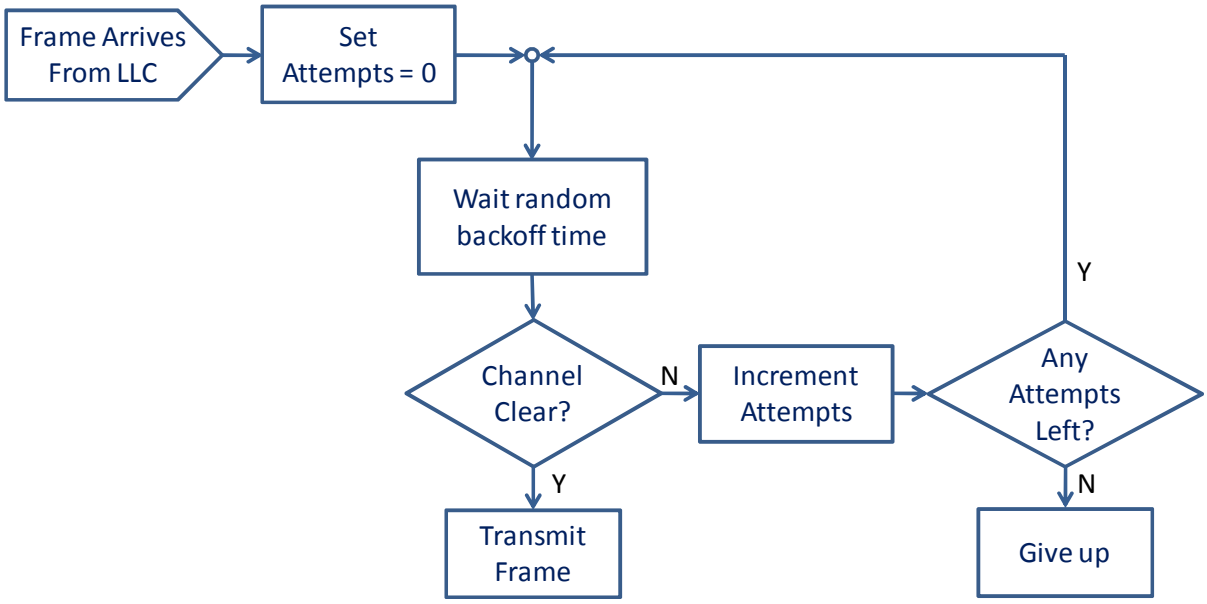


Figure 4-16 Simplified Operation of Non-Persistent CSMA

The CSMA control box (with the Persistent checkbox unchecked) is shown below:

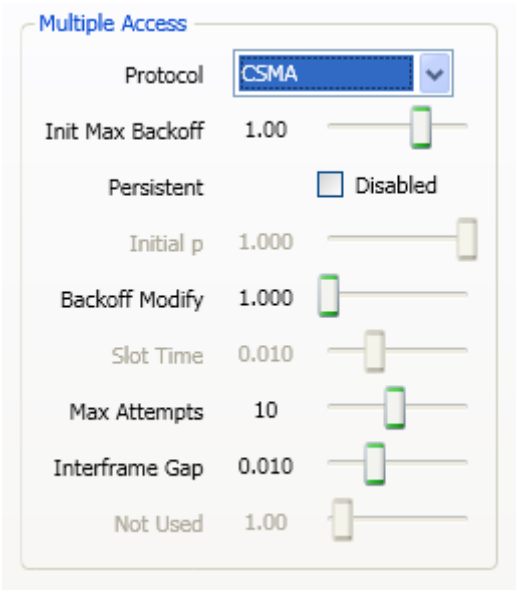


Figure 4-17 The Non-Persistent CSMA MAC Control Box

[2] Simplified, since it does not include the case where the transmitter is already transmitting or attempting to transmit a different frame when the frame arrives from the LLC. In these cases, the new frame is stored in a buffer until the earlier frames have either been successfully transmitted or the CSMA process has given up on them, and the minimum interframe space has expired.

"Initial Max Backoff" sets the maximum value of the random backoff period generated when the frame first arrives. If "Backoff Modify" is set to one, this will always be the maximum time that a node waits between attempts to transmit a frame.

"Backoff Modify" attempts to intelligently change the behaviour of the algorithm depending on channel conditions. The random backoff is a random time between zero and "Init Max Backoff" when the frame first arrives, but for each failed attempt (when the channel is sensed to be busy at the end of a backoff time), the maximum backoff time is multiplied by this "Backoff Modify" coefficient. This means that the maximum time a node may wait between attempts will increase when the network is busy.

"Max Attempts" sets the maximum number of attempts to transmit the frame before the algorithm gives up.

"Interframe Gap" sets the minimum time that a node waits after sending one frame before attempting to send another frame; this is measured in seconds.

4.5.5 The Polling MAC Layer

The Polling MAC layer assumes that the controller node is whichever node is closest to the centre of the coverage area. It operates a polling access scheme, with the co-ordinator asking each registered user in turn whether it has any packets to send or not. After a registered node has transmitted any waiting packets, it sends a polling reply back to the co-ordinator.

If a node is not registered, it waits for the start of a polling cycle, when the controller sends out a "Does Anyone Want to Join" request message, then waits for a short time for any non-registered nodes to reply. Non-registered nodes can transmit a "Registration Request" message during a special access period immediately following the "Does Anyone Want to Join" message.

The polling MAC header is shown below. Type can be set to 0 (for data frames), 1 (for polls from the co-ordinator), 2 (for poll finished replies to the co-ordinator), 3 (for "Registration Request" messages) or 4 (for invitations for join requests to be sent: the "Does Anyone Want to Join" messages).

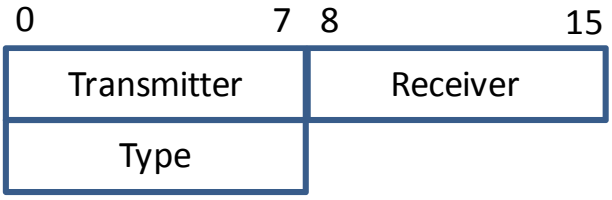


Figure 4-18 The Polling MAC Header

Five parameters can be set to configure the polling CSMA MAC protocol, as shown in the figure below.

"Max Reply Time" is the maximum time that the controller will wait for a response to a poll. If this time is reached without a "Poll Finished" reply being received, the controller will assume the poll has failed (perhaps a packet has been lost in noise) and move on to the next node. It is very important

that this time should not be less than the maximum time required to transmit the number of packets a node is allowed to send in one poll cycle, for obvious reasons. (At least I hope they're obvious reasons.) Like all the times in the polling control box, it is measured in seconds.

"Min Poll Time" is the minimum time taken for a complete polling sequence around all the nodes. If the polling sequence completes before this time, the co-ordinator waits until this period is over before starting the next polling sequence. (This parameter is here to allow the protocol to save a lot of energy sending polling messages when no-one has much to transmit.)

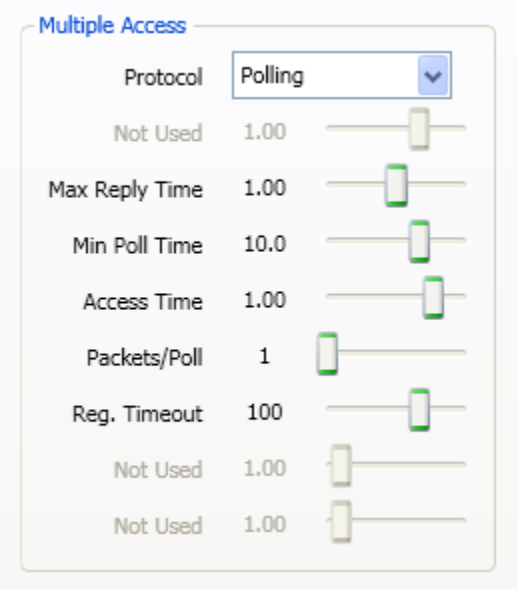


Figure 4-19 The Polling MAC Control Box

"Access Time" is the time at the beginning of a polling sequence during which the co-ordinator waits for any unregistered nodes to send a registration frame on a contention-basis.

"Packets/Poll" sets the maximum number of packets that a node is allowed to transmit before it has to stop and send a "Finished" message back to the co-ordinator.

"Reg. Timeout" is short for "Registration Timeout". If a node does not receive a poll within this time it will assume it has been de-registered, and attempt to register with the co-ordinator again.

4.6 The Physical Layer

The physical layer controls transmitting and receiving packets, and also keeps a note of the energy used so far by each node. There are several options that the user can set up, the control box is shown below.

"Preempt" can be set to Default (no pre-emption) or Preemptive (pre-emption turned on). With pre-emption on, if a new packet interrupts an existing packet, but the new packet is received so strongly that the signal to interference ratio would allow the new packet to be received, then the new packet is received. If pre-emption is switched off, then an interrupting packet will never be received, no matter how loud it is.

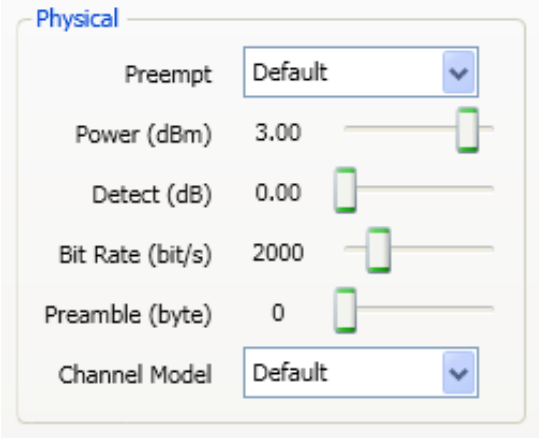


Figure 4-20 The Physical Control Box

"Power (dBm)" sets the default transmit power in dBm; "Bit Rate" sets the default Bit Rate in bit/s. Individual packets can be set at other powers, but if no power or bit-rate is specifically set by the MAC-layer protocol, it is these default values that are used. (All of the built-in protocols just use these default values).

"Detect (dB)" sets the signal to noise ratio required for a node to detect a transmitting packet (if the signal level is lower than this, the node will assume that the channel is clear).

"Preamble" sets the length of the preamble in bytes. The preamble is added to the frame by the transmitter and typically used to synchronise the receiver to the incoming transmission. It is sent at the same bit-rate as the packet. It can be thought of as a physical layer header.

"Channel Model" chooses between a default model (which assumes that the received SINR values determine the signal quality required for a bit error ratio (BER) of 1e-3), and a "Cliff Edge" model which assumes that any frame that has remained above the SINR throughout reception will always be successfully received, otherwise the frame will always be received with errors.

Higher bit-rates require higher signal to noise plus interference ratios (SINRs) for successful reception; default values used by the simulator are shown in the table below for bit error ratios of 1e-3. For more details on how the default model determines the probability of frame errors, see section 6.4.

Bit rate	SINR required (1e-3 BER)
500 bit/s	4 dB
1 kb/s	10 dB
2 kb/s	16 dB
4 kb/s	22 dB

Table 4-2 Bit Rates Supported and SINR Required

4.6.1 The Channel Loss Model

The channel loss model used by the simulator is a based on a mobile propagation model, with the loss of any channel expressed as:

$$Loss = kd^{\gamma} \tag{1.3}$$

where *d* is the distance in metres, and *k* and γ are constants, currently set to *k* = 0.001 and γ = 4.0.

The total effective noise at all receivers is 10 nW (-50 dBm).

For example, transmitting a 1 kb/s frame (which requires a SNIR of 10 dB) at -10 dBm (0.1 mW) in the absence of any interference gives a maximum range for a bit error ratio of 1e-3 of:

$$SNIR = \frac{Signal}{Noise} = \frac{0.1\times10^{-3} \div \left(0.001\times d^4\right)}{10\times10^{-9}} = 10 \tag{1.4}$$

$$d^4 = \frac{0.1\times10^{-3}}{10\times10\times10^{-9}\times0.001} = 10^3 \tag{1.5}$$

$$d = 31.6 \text{ m} \tag{1.6}$$

5 Outputting Data from the Simulator

The outputs of the simulator are given in the "Outputs" tab in the main window. There are six alternative outputs, accessed by selecting the radio buttons at the top of this tab, as shown in the figure below.

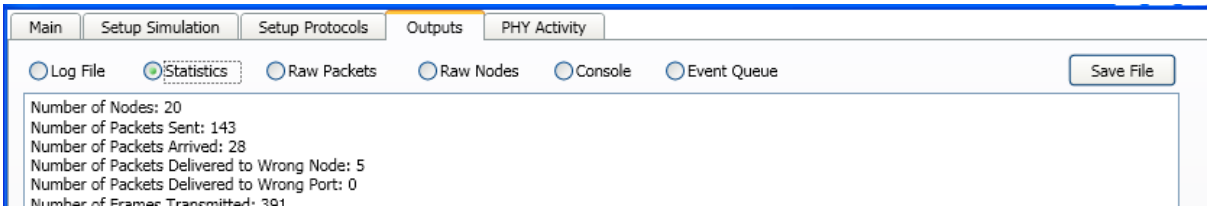


Figure 5-1 Radio Buttons in the Output Tabs

The "Save File" button brings up a standard Windows save file dialog box, which allows the current contents of the output window to be saved to a file.

5.1 The Log File Output

The log file output is a description of all the events logged in a run of the simulator. This list can get very long, and after this radio button is selected it can take a long time for the simulator to format and output the relevant list of events.

Some typical entries in a log file are shown in the figure below. Note that the local time at a node can be negative when individual clocks are used at each node.

Sim Time: 1,000,003,996. Event Type: LogicalLink_PacketArrivesFromNetworkLayer. Event 33, 3 in queue.
Node 5. Local Time: -144,319,611. Energy Left = 999.976
Packet number 0 (priority 0) size 4 with 1 header sent from 5 to 0 last hop 5 broadcast.
Logical Link: Packet sent down to multiple access layer, going to node 255

Sim Time: 1,000,004,995. Event Type: MAC_PacketArrivesFromLogicalLinkLayer. Event 34, 3 in queue.
Node 5. Local Time: -144,318,612. Energy Left = 999.976
Packet number 0 (priority 0) size 4 with 1 header sent from 5 to 0 last hop 5 broadcast.
Multiple Access: Packet sent to physical layer to transmit
Multiple Access: Channel clear, so sending immediately.

Sim Time: 1,000,004,995. Event Type: Physical_PacketArrivesFromMACLayer. Event 35, 3 in queue.
Node 5. Local Time: -144,318,612. Energy Left = 999.976
Packet number 0 (priority 0) size 6 with 2 headers sent from 5 to 0 last hop 5 broadcast.
Physical: Packet copied and sent to node 0 to arrive from local time 1944725761 to 1968724550
Physical: Packet copied and sent to node 1 to arrive from local time -199832071 to -175833417
Physical: Packet copied and sent to node 2 to arrive from local time -339898739 to -315896806
Physical: Packet copied and sent to node 3 to arrive from local time 2061325511 to 2085323782

Physical: Packet copied and sent to node 4 to arrive from local time -756352952 to -732354392

Figure 5-2 Excerpts from a Log File

For each event logged, the global simulation time is given, then the type of the event, the number of the event, and how many events are currently in the queue. If the event is associated with a node, the local time at the node is also given, along with the energy used or left in that node. (Note that it is possible for the local time to be negative when individual clocks are used at the nodes.)

If the event is associated with a packet, then the packet number, size, source and destination are also given. Finally, there is some detail about the event itself.

User protocols can also write events, for more details see the "Writing User Protocols" document.

5.2 The Statistics Output

The statistics output provides a few simple statistics about the run of the simulator, including the total number of packets generated and delivered, the mean delay times, the proportion of packets arrived successfully, etc. A typical output is shown in the figure below:

Number of Nodes: 6
Number of Packets Sent: 1
Number of Packets Arrived: 1
Number of Packets Delivered to Wrong Node: 0
Number of Packets Delivered to Wrong Port: 0
Number of Frames Transmitted: 63
Number of Frames Received: 198
Proportion of Packets Delivered: 100 %
Mean Frames Sent Per Packet: 63
Mean Delay: 4.739903455 s
Minimum Delay: 4.739903455 s
Maximum Delay: 4.739903455 s
Total Energy Used: 14.79
Maximum Energy Used by a Node: 2.483 by node 2

Figure 5-3 Excerpts from a Statistics Output File

(In this case, the reason that less than one frame is sent per packet generated is that any packets generated sent to the local node are never transmitted.)

5.3 The Raw Packets Output

This output shows the raw data for each packet: it's a matrix of numbers, with one row for each packet, and eight columns: the packet number (starting from one); the packet priority (by default set to zero); the packet length (the length of the packet when generated at the application layer, not including any headers added by lower layers); the source node; the destination node; the number of hops the packet has taken; the global simulation time the packet was generated (not the local time

at the generating node); and the global simulation time the packet was successfully received (set to -1 if the packet was not successfully received).

For an example output, see section 3.2.1.

5.4 The Raw Nodes Output

This output shows the raw data for each node: it's a matrix of numbers, with one row for each node in the simulation, and six columns: the node number, the x- and y- co-ordinates of the node, the number of packets generated at this node, the number of packets received by the application layer at this node (note that control packets and packets being forwarded to other nodes are not counted), and the total energy used or left in the node's battery.

For an example output, see section 3.2.2.

5.5 The Console Output

The console is not used by any of the in-built protocols, but can be used by user protocols as a general output space to write message to. It is available on this tab so that it can be easily saved if required.

5.6 The Event Queue

This lists the current events in the simulator's event queue. It's just here for interest, and to help optimise and debug the protocols. Typical contents look like:

Time: 20569656097; type: Physical_StartToArriveFromBelow(2000) at node 1 Packet number 0 (priority 0) size 6 with 2 headers sent from 5 to 1 last hop 5 broadcast.
Time: 20569656163; type: Physical_StartToArriveFromBelow(2000) at node 0 Packet number 0 (priority 0) size 6 with 2 headers sent from 5 to 1 last hop 5 broadcast.
Time: 20593658213; type: Physical_FinishArriving(2000) at node 1 Packet number 0 (priority 0) size 6 with 2 headers sent from 5 to 1 last hop 5 broadcast.
Time: 20593658280; type: Physical_FinishArriving(2000) at node 0 Packet number 0 (priority 0) size 6 with 2 headers sent from 5 to 1 last hop 5 broadcast.
Time: 20750000000; type: System_UpdateEnergy(0)
Time: 52682493210; type: Application_Callback(0) at node 0
Time: 56070644818; type: Application_Callback(0) at node 1
Time: 100000000000; type: System_Stop(0)

Figure 5-4 Excerpts from the Event Queue Output

5.7 The Physical Layer Activity Tab

The fifth tab on the screen shows a graph of the physical layer activity for the current run of the simulator.

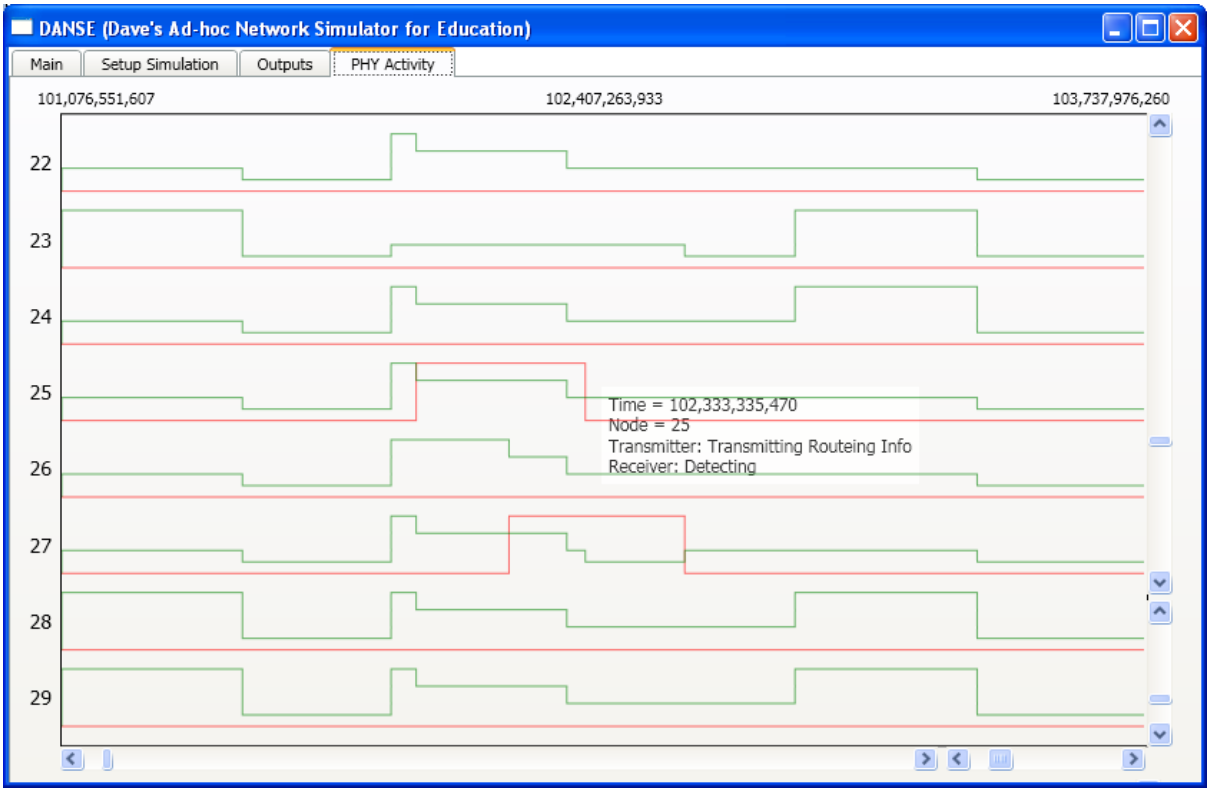


Figure 5-5 The Physical Layer Activity Tab

The sliders on the bottom of the display determine the centre time of the display and the range of the display; the sliders to the right-hand side determine the first node and the number of nodes shown in the display (it can get crowded and bit slow to draw if too many nodes are shown for too much of the time in a large simulation).

The red line indicates the transmit activity, and is either at the bottom (nothing being transmitted) or the top (the transmitter is currently active).

The green line indicates the receiver activity at each node, and is at the bottom of its range when the node is asleep; slightly higher when awake and listening; about half-way up when detecting another frame; slightly higher still when a collision is detected; and at the top of the range when the node is actively receiving a frame.

In all cases, putting the cursor on the graph will show was happening at that node at the time. The only point of interest about this is that when receiving a frame, a number in parenthesis appears after the number of the packet: for example “Receiving 3 (2.9)”. The number in parenthesis is the difference between the SINR at the node and the SINR required for a bit-error-ratio (BER) of 1e-3; if this is negative for large frames, the frames are unlikely to be received successfully.

As well as using the sliders, it is possible to zoom into the graph by pressing down the left mouse button on the graph, and dragging over the range to be displayed. Right-clicking the mouse zooms out by a factor of two.

Clicking the left mouse button adds a time cursor, so that transmission times and delays can be more accurately measured. Double-clicking turns the cursor off again.

6 Some Technical Notes about the Simulator

This section is mostly for interest, and for anyone thinking of modifying the simulator itself, rather than just using the simulator or writing custom protocols.

6.1 Event Timing and Delays

Any event queued by a node takes at least one microsecond[3] to do. This is implemented by adding one microsecond to the time at which an event is queued in the big event queue. So, for example, an event passed to the queue's AddEvent() method by a protocol layer at a node asking to be queued at the current time would actually be queued one microsecond into the future.

This means that it is possible for CSMA to result in collisions, as it takes at least one microsecond between the decision to start transmitting, and the actual start of the transmission itself. (It could also happen because of the propagation delay, but this is less likely, as a typical "long" range of 100 meters would imply a delay of only 0.33 microseconds).

6.2 The Energy Consumption of Nodes

Nodes consume energy according to what they are doing at the time, according to the following table:

Node State	Power Consumption
Asleep	24 μW
Actively Listening	24 mW
Receiving a Frame	48 mW
Transmitting a Frame	(0.03 + 10 * *TxPower*) W

Table 6-1 Power Consumption Defaults

These figures are adapted from the data sheet for the Crossbox IRIS XM2110CA nodes[4], which suggest typical current consumption from the 3V batteries that it uses: 8 μA when asleep, 8 mA when active (listening), 16 mA when receiving, and 10 mA, 13 mA and 17 mA respectively when transmitting -17 dBm, -3 dBm and 3 dBm.

A best-fit straight line through these points gives the power consumption when transmitting a frame as 0.03 + 10 * P, where P is the transmit power in Watts.

These constants are set in the Globals.cs file, in the "Energy Constants" region, along with the default energy in each node at the start of a simulation (1 kJ). (currently set to 2.0)

6.3 The Propagation Model

There's a very simple channel model in the simulator at the moment, of the form:

$$Loss = kd^{\gamma} \tag{1.7}$$

This was chosen because it is easy to implement.

[3] The value of one microsecond is the current default. This value is set by the Globals.MinimumEventTime constant set in the Globals.cs file.

[4] http://www.dinesgroup.org/projects/images/pdf_files/iris_datasheet.pdf [Accessed on 16/7/10]

The variables *k* and γ are specified by the global variables `PathLossExponent` (currently set to 4.0) and `PathLossConstant` (currently set to 0.001) in the Globals.cs file. Both values (and the default noise level) can be changed from a configuration file, but it will have to be manually edited outside DANSE.

This isn't a very realistic model for small networks, but it is used in a large amount of research due to being easy to implement and fast to run. Implementing a more realistic channel model is high on the wish list for the next version of the simulator.

6.4 The Bit Error Ratio Model

Each node keeps a list of the different SINR values during the reception of a frame, and how long they lasted. (For example, if a packet interrupts another packet the SINR will decrease during the transmission of the frame; this would result in two entries in the list: a larger SINR value at the start of the frame's reception, and a smaller SINR value at the end.)

When the channel model is set to default, the probability of a frame error is determined by considering the number of bits sent during each period in the list. First, a bit error probability is determined using:

$$x = \sqrt{9.45 \times \text{excessSNIR}} \tag{1.8}$$

$$ber = \frac{\left(1 - \exp\left(\frac{-1.98x}{\sqrt{2}}\right)\right)\exp\left(\frac{-x^2}{2}\right)}{1.135\sqrt{2\pi}\ x} \tag{1.9}$$

where excessSNIR is the ratio of the actual SNIR to the value required for a bit error ratio of 1e-3. (Equation (1.9) is an approximation to the Q-function[5].)

The probability of success for this portion of the frame is then determined using:

$$p = (1 - ber)^L \tag{1.10}$$

where *L* is the number of bits during this portion of the frame. For successful reception, all portions of the frame must be received successfully.

(This is equivalent to assuming that the modulation schemes using behave the same as BPSK or QPSK in terms of the general shape of their BER against SNR curves.)

Also note that if the initial SINR is more than 3 dB below the level required for BER of 1e-3, the node will not attempt to receive the frame. This corresponds to an expected BER of around 1.4 %, which means that a small 8-byte frame would have a probability of success of less than 40%. The theory is that any protocol that relies on that chance is unlikely to be a good design.

[5] G. K. Karagiannidis and A. S. Lioumpas, "An improved approximation for the Gaussian Q-function," *IEEE Communications Letters*, vol.11, no. 8, Aug. 2007, pp. 644–646.

6.5 Simulation Speed and Interrupts

When running, the simulator has an internal interrupt that is called every 100 ms, and events are done within the interrupt routine. In real time or fast real time modes events are done until the current time, or until 60 ms have elapsed, which ever happens sooner.

In theory, on a slow computer with a lot of nodes and a busy network, all the events required to be done for this 100 ms period could take longer than 60 ms to do, in which case the simulator will start running slower than the real time.

When running in free mode (as fast as possible), the simulator will do however many events it can within 60 ms. To avoid having to check the time after every event, this is done by maintaining a "how many events to do per interrupt" variable (called Globals.EventsPerInterrupt), and adjusting the value of this variable after the end of each interrupt run. The idea is to adjust the value until the interrupt takes about 60 ms to do the events, leaving 40 ms for other tasks.

When there is a selected node and chkRoutes is selected, the routes from the current selected node are recomputed and re-displayed as the simulation progresses. With a large number of nodes in the simulation, this can take quite a long time to do (greater than 250 ms) on slower machines. As a result, the updating of the routes is only done once per second. On slower machines, the simulation can seem to pause for a short time every second during the time when it is recalculating and redisplaying routes.