User's Manual

v.1.1

Table of Contents

1. General

Pamvotis 1.1 simulator is an advanced WLAN Simulator for the IEEE 802.11 protocol including all its physical layer extensions (IEEE 802.11a, IEEE 802.11b, IEEE 802.11g) and the IEEE 802.11e EDCA function for QoS. Basically it performs a very detailed simulation of the MAC protocol, based on the IEEE 802.11 specification.

Pamvotis was developed in Java, using only libraries included in the JRE. This way, it has limited hardware requirements and is compatible with any operating system with JRE installed.

The major advantages of the application concerning simulation of WLANs are the following:

- *Support of data rate agility*. This means that each node can work on its own data rate, depending on the distance from the receiver.
- *Support of the hidden terminal problem.* Nodes can be configured to be in LOS or NLOS, in order the hidden terminal problem to be investigated.
- *Support of "absence / existence of non-ERP nodes*". This advantage utilizes the option of the IEEE 802.11g protocol to adjust the network parameters depending if all nodes support IEEE 802.11g (absence of non-ERP nodes) or some nodes support only 802.11b or earlier (existence of non-ERP nodes).
- *Support of the new CTS-to-Self virtual carrier sense and protection mechanism*: This means that a node can support RTS/CTS or the new (defined in IEEE 802.11g specification) CTS-to-Self virtual carrier sense mechanism for better network performance.
- *Support of different types of traffic sources*: Pamvotis 1.1 supports more than one traffic sources on each node. Moreover, it supports different types of traffic sources, like FTP, HTTP, Video, as described in 3GPP standardization group.
- *Support of all new physical layers of the IEEE 802.11g specification which include*:
    - *ERP-DSSS-CCK*: Support of IEEE 802.11 data rates (1, 2 Mb/s) and IEEE 802.11b data rates (1, 2, 5.5 and 11 Mb/s).
    - *ERP-OFDM*: Support of all extended data rates defined in IEEE 802.11a and IEEE 802.11g (6, 9, 12, 18, 24, 36, 48, 54 Mb/s).
    - *ERP-PBCC*: Support of the extended data rates of 22Mb/s and 33Mb/s.
    - *DSSS-OFDM*: Support of the extended rates of the IEEE 802.11a with DSSS preamble and OFDM payload.
- *Support of the IEEE 802.11e EDCA* function for QoS and service differentiation in WLANs.
- *Support of many statistic results,* which show detailed information concerning the MAC layer performance. Those are: Throughput in bits and packets per

second, utilization, media access delay, queuing delay, total packet delay, delay jitter, mean number of retransmission attempts and mean packet queue length.*Very limited resource requirements*, which allows operation in any PC.

- *Cross platform operation,* meaning operation in any OS with Java Runtime Environment installed (Windows, Linux, Solaris, MacOS).
- *User friendly interface*, which allows quick and easy simulation configurations.
- *Many functions of the core simulation engine:* The core simulation engine provides many functions by which you can use Pamvotis as an embedded simulator, in order to build a more extended simulation platform (e.g. an integrated WiMAX/WiFi simulator). Analytical documentation exists for each one of these functions.
- *Open source code* in order to implement your own protocols.
- *FREE FOR ALL* (Major).

1.1. What's new history

Pamvotis 1.1 extends Pamvotis 1.0 with the following features:

- Correction of bugs reported by users in Pamvotis 1.0.
- Addition of FTP, HTTP and Video traffic sources, as described in 3GPP TR 25.892 V6.0.0 (see Appendix).
- Ability to have more than one traffic sources in each node.
- Useful functions to add / remove sources and nodes at any time during the simulation.
- Ability to simulate very long simulations (maximum is 50737 centuries of simulation time, but don't bother to try it because you will probably never see the simulation results).

2. Before you start

2.1. Application Requirements

Operating System: Windows (95/98/Me/NT/2000/XP/2003), Linux, Solaris, MacOS.

Disk Space: 800KB

Memory: 64MB

CPU Speed: 500MHz

Other: Java Runtime Environment 5.0 or higher.

2.2. Loading the application

If you installed the application on Windows, using the installer, simply click *Start*, go to *Program Files* and under the *Pamvotis 1.0* menu click *Pamvotis Simulator.*

If you use Linux, MACOS or Solaris, extract the .tar.gz folder, open a terminal, browse to the folder and run the command:

Java –jar pamvotis.jar

The above process must be followed on Windows, for the case where you didn't install the program through the installer, but downloaded the .tar.gz folder. In this case, you can additionally double click the file "pamvotis.jar" instead of opening a command terminal and executing the above command.

2.3. Uninstalling the application

Simply click the '*Uninstall Pamvotis* item of the *'Pamvotis 1.0'* menu under *Program* Files, or go to *Add remove Programs* on the *Control Panel* and select *Pamvotis 1.0* from the listed applications.

3. Performing a Simulation: Step by step

The steps for performing a simulation is to configure the simulation, run it and view the results.

3.1. Configuring the simulation

Once you have loaded Pamvotis, an application window is displayed named *Pamvotis WLAN Simulator*. All the simulation parameters are configured from this window and are explained below:

3.2. The events configuration group

Seed: Select the seed value which will be used by the random number generator. Different values of seed produce different results for the same configuration.

Simulation time: Select the simulation time in seconds. Only integer values are allowed.

3.3. The statistic results configuration group

Pamvotis provides an amount of useful statistic results concerning the MAC layer. Those can be configured from the results configuration group:

Collected Values per Statistic: Select the number of collected values per statistic. This is the number of values that will be written in the results files.

Choose statistics: Click on the button to choose the statistics you want to be collected. Note: Even if no statistics are chosen, a file containing the mean values (in relation to time) of all statistics will be produced. Below there is a brief explanation of each statistic.

- *Throughput (bits/sec):* The number of bits that a node successfully transmitted in a specific time interval.
- *Throughput (packets/sec):* The number of packets that a node successfully transmitted in a specific time interval.
- *Utilization:* The percentage of the channel capacity the node occupied. The utilization is the node's throughput in bits per second divided by the node's data rate.
- *Retransmission Attempts*: The mean number of retransmission attempts (collisions) until the packet is successfully transmitted.
- *Media access delay*: The delay of a packet from the time it is picked up from the transmitter until it is successfully received from the receiver. This statistic contains the delay due to retransmission attempts and the transmission delay. Note: The statistic is relative to the packet length. Moreover, it does not depend on the node's packet generation rate, but only on the network load.
- *Queuing delay*: The delay from the birth of a packet until the transmitter picks it up for transmission. It only contains the time a packet waits in the packet queue. It is relative to the packet generation rate and to the media access delay.

- *Total packet delay*: The sum of the media access delay and the queuing delay. It is the total delay from the birth of a packet until its reception from the receiver.
- *Delay Jitter:* The total delay jitter of each node. The delay jitter is the standard deviation of the total packet delay of each node.
- *Packet Queue Length*: The mean number of packets that wait to be transmitted in the packet queue.

For each one of the selected statistics, a text file is produced which contains the results. An extra file containing the mean values of all the results is also produced.

Once you select the statistics you want, press OK to return to the main configuration window.

Directory to save results: Choose the directory in which the results will be saved. The default directory is the application installation directory.

3.4. The MAC Layer Configuration group

In this group, the MAC layer parameters of the IEEE 802.11 protocol are configured:

Access Mechanism: Select one of the three access mechanisms to be used. (Basic access, RTS/CTS or CTS-to-Self). Note: CTS-to-Self is a new protection mechanism only supported by IEEE 802.11g protocol. If you are not familiar with this, or if you are simulating IEEE 802.11b or earlier networks, use RTS/CTS.

RTS Threshold: Specify the RTS threshold in bits. If the packet is larger than this threshold then the protection mechanism will be enabled. Only integer values are allowed. Note: If you want the protection mechanism to be always enabled set the RTS Threshold to zero.

IEEE 802.11e EDCA onfiguration: Pressing this button, a new window appears where the parameters of EDCA are configured. EDCA is the fundamental mechanism of IEEE 802.11e specification for QoS support.

Note: If you do not want to use QoS, just leave the parameters as is. Moreover if you use Network Configuration File, be sure to set the AC parameter to 0 for all nodes.

If you want to use service differentiation through EDCA, then three parameters must be defined for each Access Category (AC):

- *Minimum Contention Window (CwMin):* This parameter is defined as a submultiple of the original CwMin of DCF, which may be 16 (802.11a, 802.11g) or 32 (802.11, 802.11b). The definition is done using integer values as submultiples (division factors). For example, if a division factor of 2 is used for AC-1, and 802.11g is used, then the CwMin for AC-1 will be 16/2=8. Note: Be sure to use only integer values that are powers of 2, otherwise unexpected results will arise or the program will crash. Be sure to define the division factors so the CwMin value to be at least 1. For example, if you are certain that you will use 802.11b, then a division factor of 16 can be used for AC-3 (giving a CwMin of 32/16=2). But if you accidentally use 802.11g or 802.11a then the CwMin will become 1, which has no practical meaning. Correct parameter definition is at your own risk. Generally, smaller values of CwMin (meaning bigger values of the division factor) guarantee more priority.

- *Maximum Contention Window (CwMax):* This parameter is defined as a submultiple of the original CwMax of DCF, which is 1024. Same notes for CwMin stand for this case also. Generally, smaller values of CwMax (meaning bigger values of the division factor) guarantee more priority.
- *AIFS*: In original DCF the DIFS parameter is defined as DIFS=SIFS+2*slotTime. In EDCA, the DIFS parameter is named AIFSD, is different for each AC and is defined as AIFSD=SIFS+AIFS*slotTime. AIFS is an integer, different for each AC, that must be defined. Keep in mind that the default value of AIFS is 2, not 0 as many would expect. Correct definition of AIFS is at your own risk. Generally, bigger values of AIFS gurantee less priority.

3.5. The network configuration group

There are three available options there.

- Global configuration: Use this option if all nodes of your network have the same parameter values. Note: If you choose to configure nodes globally then LOS will be assumed for all nodes. If you plan to include the hidden terminal problem in your simulation scenario then you must configure the nodes using the Network Configuration File.
- Configuration with network configuration file: Use this option if nodes have different parameters. Also specify the location of the network configuration file. Note: The default location of the file is in the application installation directory. If you accidentally leave this location, then the default network configuration file located there, will be used.
- For information about how to use the network configuration file see the section "Using the Network Configuration File".
- Configuration with application's XML file: When configuring the simulation parameters (either globally or through network configuration file), Pamvotis creates an XML file, which is read by the simulation engine when the user presses the Run button. This XML file is located in the conf folder of the application directory. Users that are familiar with XML can use this file to define the simulation scenario. This gives also the ability to copy the XML file in a backup directory, if you intend to load the scenario some other time (you can do that even if you are not familiar with xml). Keep in mind that the new features supported by Pamvotis 1.1 can only be available if you use this option. For

In the remainder of this subsection, the meaning of each parameter for global node's configuration is explained.

Physical layer configuration: Select one of the available physical layers for your network. The last option concerns the case where IEEE 802.11b compliant nodes exist in an IEEE 802.11g network.

Nodes' configuration: The following parameters must be configured:

Number of Nodes: Insert the number of nodes that exist in your network.

Number of IEEE 802.11b Nodes: If you are simulating an IEEE 802.11g network select the number of IEEE 802.11b compliant nodes that exist in your network. Obviously, those must be less than the total number of nodes that exist in the network.

Packet length configuration: The following items must be configured:

Packet Length Distribution: Select one of the distributions the length of the generated packets obeys.

Packet Length mean: Select the mean value of the packet length in bits. Note that the maximum value the protocol allows is 32768bits. If you set the mean to a higher value then the packet will be fragmented.

Packet generation rate configuration: The following items must be configured:

Packet Generation Rate Distribution: Select the distribution the packet generation rate obeys.

Packet generation rate mean: Select the mean value of packet generation rate in packets per second.

Rate (only available in network configuration file): The node's data rate in b/s.

X-Position (only available in network configuration file): The node's horizontal coordinate in meters, related to an abstract coordination center (point 0).

Y-Position (only available in network configuration file): The node's vertical coordinate in meters, related to an abstract coordination center (point 0).

Coverage (only available in network configuration file): The node's coverage radius in meters.

Physical layer-PHY (only available in network configuration file): The node's physical layer type (802.11simple/a/b/g/mixed).

AC (only available in network configuration file): The node's access category for 802.11e QoS support (0, 1, 2 or 3).

3.6. Running the simulation

In this point the simulation configuration is completed. Click on the "Run" button to start the simulation execution. The simulation execution window is opened which shows the simulation progress. This may take from a minute to a century, mostly depending on the simulation time and your CPU speed. The simulation execution runs as a process with normal priority in the operating system. You can use the task manager to increase or decrease the priority of the task. It is recommended not to use high or real time priority because you will not be able to touch your PC until the simulation is complete. You can pause or stop the simulation any time you want. However, if you stop the simulation, results will be collected until this moment but mean values will not be collected at all. Once you have done, close the window to return to the main console.

3.7. Viewing the results

Click on the "View Results" button to view the results. Load the text file you are interested. The file shows the results in respect to time. The "Mean Values" file shows the mean values of the results related to time. You can copy the results and paste them to a spreadsheet in order to process them.

Pressing the 'View Results button opens the corresponding text file with Notepad (for Windows) or with gEdit (for Linux). If gEdit is not installed on Linux, then Mozilla Firefox is used. If Mozilla does not exist, Netscape is used. If Netscape does not exist either, then you should seriously consider stop using Linux.

For Linux users, in order to view the results in the correct format, you must configure gEdit so the tab width to be 8 chars (Edit→ Preferences → Editor tab → Tab width).

4. Using the Network Configuration File

The network configuration file is used when the nodes' configuration parameters are different. In this case, a special configuration is required for each node.

The source directory of the network configuration file is the 'conf' folder of the application directory. However, you can create your own configurations and load them to the application, from another directory.

There are two types of network configuration file that you can choose. The one is excel-based file and the other is an ASCII text-based file.

Excel file is easier to use and safer, concerning possible configuration mistakes. It is recommended for Windows and MacOS users.

Text-based file is the only option for Linux and Solaris users. At the moment Pamvotis 1.0 is developed, existing Java libraries for reading excel files from Java applications are not mature and stable enough. So, a text-based network configuration file is used. Note: Selecting an excel-based network configuration file while running the application is Linux or Solaris, will end up in unexpected results. The program does not prohibit you from doing this, so the correct network configuration file selection is on the user's responsibility.

4.1. Configuring the node's parameters

In order to configure the nodes' parameters you should open the network configuration file and edit it. Once you have configured your parameters, save the file (or save it with another filename) and return to Pamvotis main console.

The network configuration file is a simple Excel or ASCII file from which Pamvotis reads the nodes' parameters.

Move to the beginning of the network configuration file. An explanation section exists (yellow box for excel-based and a rectangle for the text-based). Read this section carefully. After the explanation section, the parameter's headers exist.

Begin to write after the parameters headers. This means, after the green line in the excel based file or after the dotted line in the text-based file.

The parameters to be configured are explained below:

- *Node:* The number of each node. The numbering starts from 1.
- *Data rate:* The node's data rate in bits/sec. Attention: The application does not check if the data rates are valid. Validation of data rates is on your own risk.
- *PktLength*: The mean value of the packet length in bits. Only integer values are allowed.
- *PktDistr*: The distribution of the packet length: c for constant, u for uniform and e for exponential. Be sure to write only one of the above three characters in lowercase. Write only one of the predefined characters. Any other character will be handled as exponential distribution.
- *genRate*: The mean value of the packet generation rate in packets/sec. Only integer values are allowed.

- *genDistr*: The distribution of the packet generation rate: c for constant, u for uniform and p for Poisson. Be sure to write only one of the above three characters. Write the characters in lowercase. Write only one of the predefined characters. Any other character will be handled as Poisson distribution.
- *xPosition:* The horizontal (x) coordinate of each node in meters. If you do not care about hidden terminals and want all nodes to be in LOS, set the same values for *xPosition* and *yPosition* parameters. Do not leave them blank.
- *yPosition:* The vertical (y) coordinate of each node in meters. If you do not care about hidden terminals and want all nodes to be in LOS, set the same values for *xPosition* and *yPosition* parameters. Do not leave them blank.
- *Coverage:* The coverage range of each node in meters.
- *Phy*: The node's physical layer (802.11 simple/a/b/g/mixed). s for simple, original 802.11, a for 802.11a, b for 802.11b and g for 802.11g. Be sure to write only one of the above four characters in lowercase. Any other character will be handled as 802.11g layer.
- *AC*: The node's access category for IEEE 802.11e EDCA (0, 1, 2 or 3). Be sure to use only one of the above values. Any other value will be considered as AC-0 (best effort).
- *Reserved:* This is reserved for future use. Do not write anything there.

5. Using the application's XML file

The application's XML file is an alternative method to configure a simulation. Using this method, a text-based configuration can be done, without having to pass through all configuration steps when there is the need to repeat a simulation with minimal changes.

Note: All new features supported by Pamvotis 1.1 are applicable only through the application's XML file. This means that, if you want to configure sources other than generic, or if you want to configure nodes with more than one sources, then you must do it through the application's XML file.

Despite the fact that the document is based on the XML language, its structure is simple, so there is no need to know XML in order to configure it.

The application's XML file is always stored in the folder PAMVOTIS_HOME/config. Hence, pay attention to always keep a backup of the configuration files you create. A folder named bkp exists for this purpose, where backups of the original configuration files exist. It is recommended not to modify the files of this folder.

The remaining of this section describes the structure of the XML network configuration file.

- *Seed*: An integer number from which the random number generator is initialized. Different values of seed produce different simulation scenarios.
- *Duration: The* duration of the simulation in seconds.
- *Values:* The number of collected values for each statistic, that will be written to the results files.
- *Node element:* We create as many node elements as the network nodes. Each node element has a number (ID) which is an integer number, unique for each node. Each element has the following characteristics:
    - *rate:* The node's data rate in bits.
    - *Coverage:* The coverage range of each node in meters.
    - *xPosition:* The horizontal (x) coordinate of each node in meters. If you do not care about hidden terminals and want all nodes to be in LOS, set the same values for *xPosition* and *yPosition* parameters. Do not leave them blank.
    - *yPosition:* The vertical (y) coordinate of each node in meters. If you do not care about hidden terminals and want all nodes to be in LOS, set the same values for *xPosition* and *yPosition* parameters. Do not leave them blank.
    - *AC*: The node's access category for IEEE 802.11e EDCA (0, 1, 2 or 3). Be sure to use only one of the above values. Any other value will be considered as AC-0 (best effort). Be also sure to set AC-0 for all nodes, if you do not want to use EDCA at all.

- *Source element*: This element describes a traffic source for a node. Each node can have on or more traffic sources, each one represented by a source element. Each source element has an ID, which is a unique integer number for each source in a node, and a type, which can be one of the following: *generic, ftp, http, video*. See the description of each one of the elements, further on this section.

- *nodes:* The number of nodes in the network. This parameter is not used at all and may be omitted.
- *mixedNodes:* The number of IEEE 802.11b compliant nodes in a mixed 802.11b/g network. Be careful with this parameter. If you set it different than 0, be sure to set the *phyLayer* parameter to *m*.
- *phyLayer:* The physical layer type. (802.11 simple/a/b/g/mixed). s for simple, original 802.11, a for 802.11a, b for 802.11b, g for 802.11g and m for mixed mode 802.11b/g. Be sure to write only one of the above four characters in lowercase. Any other character will be handled as 802.11g layer. Be also sure to set the number of 802.11b compliant nodes (*mixNodes* parameter) if you selected the mixed mode network.
- *RTSThr:* The RTS threshold in bits. Set it to 0 if you want to be always enabled. Set it to 999999 if you want to be disabled, or set it to a value you desire.
- *CTSToSelf:* The CTS-to-Self parameter for 802.11g networks. Set it to y if you want it to be enabled or to n if you want it to be disabled. Be careful to select the correct type of physical layer. Enabling the CTS-to-Self parameter will have an effect only in 802.11g networks. Enabling CTS-to-Self and selecting another type of physical layer will lead to unexpected results.
- *EDCA element:* This element contains parameters for the 802.11e EDCA function for QoS. Leave the default values if you do not want to use it. Default values may be seen if you run Pamvotis Simulator and create a scenario without using 802.11e EDCA.
  - *CWMinFact0/1/2/3:* The division factor for the minimum contention window for AC-0, AC-1, AC-2 and AC-3 respectively. See the user's manual for a definition of the division factor.
  - *CWMaxFact0/1/2/3:* The division factor for the maximum contention window for AC-0, AC-1, AC-2 and AC-3 respectively. See the user's manual for a definition of the division factor.
  - *AIFS0/1/2/3:* The value of AIFS for AC-0, AC-1, AC-2 and AC-3 respectively. See the user's manual for a definition of AIFS.
- *ResultsPath:* The path where the results files will be created.
- *OutResults*: This is a tricky string parameter. It is used to check which results the user wants to be printed. If a user wants to collect values for a specific result then a specific substring is concatenated to the string parameter. For each result, the *PrintStats* method of the Simulator class checks to see if a specific substring is contained in the *outResults* string variable. If it does, then the result is printed. The substrings representing each result are shown below:

- *tb*: Throughput in bits/s.
- *tp:* Throughput in packets/s.
- *ut:* Utilization.
- *md:* Media access delay.
- *qd:* Queueing delay.
- *td:* Total delay.
- *dj:* Delay jitter.
- *ra:* Retransmission attempts.
- *ql:* Queue length.

Thus, if we want to collect values for the first three results and for the last, we would have to define the outResults parameter as tb_tp_ut_ql_. Actually, the underscore is not necessary, but is useful for separating the substrings. If you do not want any results to be printed, do not leave this parameter empty. Just define it with an underscore or another character.

5.1. Description of source elements

In this subsection, a description of the parameters of each source element is outlined.

5.1.1. Generic source

This source represents a generic (abstract) source, producing abstract traffic, obeying specific parameters. Use this source by specifying *type="generic"* in the definition of the source element.

- *pktLngth:* The node's packet length mean value in bits. Can be a decimal number.
- *pktDist:* The distribution of the packet length: c for constant, u for uniform and e for exponential. Be sure to write only one of the above three characters in lowercase. Write only one of the predefined characters. Any other character will be handled as exponential distribution.
- *intArrTime*: The mean value of the packet generation rate in packets/sec. Can be a decimal number.
- *intArrDstr*: The distribution of the packet generation rate: c for constant, u for uniform and e for Poisson. Be sure to write only one of the above three characters. Write the characters in lowercase. Write only one of the predefined characters. Any other character will be handled as Poisson distribution.

5.1.2. FTP source

This source represents a source generating FTP traffic, as described in 3GPP TR 25.892 V6.0.0 (see Appendix). Use this source by specifying *type="ftp"* in the definition of the source element. For the explanation of the parameters that follow, please have a look at the appendix.

- *pktSize*: The MSDU packet size in bits. Can be a decimal number.
- *fileSizeMean*: The mean value of the file size in bytes. Can be a decimal number.

- *fileSizeStDev*: The standard deviation of the file size in bytes. Can be a decimal number.
- *fileSizeMax*: The maximum value of the file size in bytes. Can be a decimal number.
- *readingTime*: The reading time in seconds. Can be a decimal number.

5.1.3. HTTP source

This source represents a source generating HTTP traffic, as described in 3GPP TR 25.892 V6.0.0 (see Appendix). Use this source by specifying *type="http"* in the definition of the source element. For the explanation of the parameters that follow, please have a look at the appendix.

- *pktSize*: The MSDU packet size in bits. Can be a decimal number.
- *mainObjectMean, mainObjectStDev, mainObjectMin, mainObjectMax*: The mean, standard deviation, minimum and maximum values of the main object in bytes. Can be a decimal number.
- *embObjectMean, embObjectStDev, embObjectMin, embObjectMax*: The mean, standard deviation, minimum and maximum values of the embedded object in bytes. Can be a decimal number.
- *NumOfEmbObjectsMean, NumOfEmbObjectsMax*: The mean and maximum values of the number of embedded objects. Can only be integer values.
- *readingTime*: The reading time in seconds. Can be a decimal number.
- *parsingTime*: The parsing time in seconds. Can be a decimal number.

5.1.4. Video source

This source represents a source generating video traffic, as described in 3GPP TR 25.892 V6.0.0 (see Appendix). Use this source by specifying *type="video"* in the definition of the source element. For the explanation of the parameters that follow, please have a look at the appendix.

- *frameRate*: The frame rate in frames/sec. Can only be an integer number.
- *packetsPerFrame*: The number of packets per frame. Can only be an integer number.
- *pktSize*: The mean value of the packet size in bytes. Can be a decimal number.
- *pktSizeMax*: The maximum value of the packet size in bytes. Can be a decimal number.
- *pktIntArr*: The mean value packet interarrival time in seconds. Can be a decimal number.
- *pktIntArrMax*: The maximum value of the packet interarrival time in seconds. Can be a decimal number.

6. Attention !!!

In order to prevent possible errors of the application, pay attention to the following:

1. Make sure to use only integer values in all parameters or you will lead the application to buffer overflow.
2. As concerns the output results, some times negative values may arise (especially in the delay metric). This is not wrong, and means that the values are infinitive (very very big).
3. For Linux users, in order to view the results in the correct format, you must configure gEdit so the tab width to be 8 chars (Edit→ Preferences → Editor tab → Tab width).
4. Correct definition of the EDCA parameters is at your own risk. No validation mechanism is provided from the program. Incorrect definition will lead to unexpected output results or program crash.
5. Be sure to correctly specify the location of the network configuration file or the application will use the default file located in the installation directory.
6. Do not write or erase anything before the line that contains the headers of the nodes' parameters in the network configuration file or the application will crash.
7. As concerns the distribution and physical layer parameters of the network configuration file, write only predefined, single characters in lowercase.
8. When using the network configuration file, correct data rate specification according to each physical layer is on the user's risk. Correct physical layer specification is also on the user's risk.
9. Be sure to complete all the parameters of the network configuration file, even if some of them have no meaning for your simulation. For example, if you do not wish to use QoS, set the AC parameter to 0 for all nodes and do not leave it blank.
10. Generally, keep in mind that no validation is provided from the program for the correct use of the Network Configuration File. Any mistake will cause in a crash or in unexpected output results.
11. If you use the Network Configuration File and do not want to use QoS, be sure to set the AC parameter to 0 for all nodes.

A backup of the network configuration file can be found in the 'bkp' folder located in the application installation directory. It is recommended not to edit this file and keep it as a backup of the original network configuration file.

7. About Pamvotis

Pamvotis 1.1 Simulator was created by Dr. Dimitris El. Vassis and Vassilis Zafeiris, a telecommunications engineer and a software engineer respectively.

Pamvotis took its name from a lake in Ioannina City of Greece. Pamvotis (lake) is the only lake in the world that has an island with permanent residents. The logo of pamvotis (the *P* letter) depicts the lake.

Pamvotis was designed for academic purposes only. It is not a commercial application. The application and the source code may be distributed or modified under the terms of the GNU General Public Lisence. See the *Lisence.txt* file located in the 'doc' folder of the application directory for more information.

For any information and for any comment (well intentioned or not) concerning Pamvotis, please contact the authors:

support@pamvotis.org

Special regards to the following people:

Mr Gary Gaatz for his valuable remarks on some bugs concerning IEEE 802.11b.

Mr Stelios Koukoutsidis for his valuable remarks on some incorrect parameter definitions concerning IEEE 802.11a.

Ms. Rachel Vecchitto for providing a class for opening a URL with the default browser in older versions of Java.

8. Appendix – Part of 3GPP TR 25.892 V6.0.0

A.3.4 Traffic Sources

Three different traffic types are suggested for evaluation purposes.

A.3.4.1 HTTP Traffic Model Characteristics

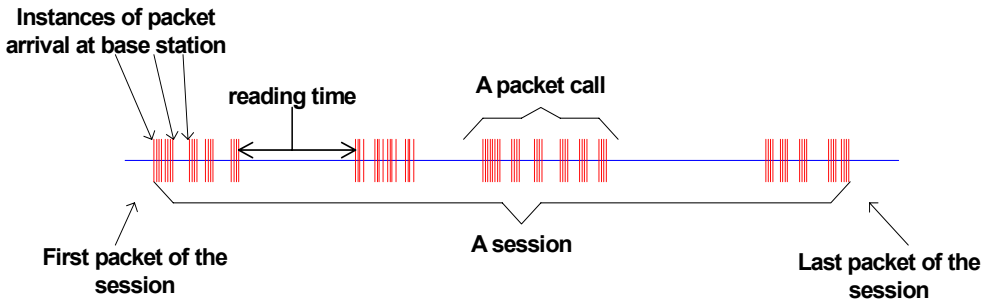

Figure 44: Packet Trace of a Typical Web Browsing Session

Figure 44 shows the packet trace of a typical web browsing session. The session is divided into ON/OFF periods representing web-page downloads and the intermediate reading times, where the web-page downloads are referred to as packet calls. These ON and OFF periods are a result of human interaction where the packet call represents a user's request for information and the reading time identifies the time required to digest the web-page.

As is well known, web-browsing traffic is self-similar. In other words, the traffic exhibits similar statistics on different timescales. Therefore, a packet call, like a packet session, is divided into ON/OFF periods as in Figure 44. Unlike a packet session, the ON/OFF periods within a packet call are attributed to machine interaction rather than human interaction. A web-browser will begin serving a user's request by fetching the initial HTML page using an HTTP GET request. The retrieval of the initial page and each of the constituent *objects* is represented by ON period within the packet call while the parsing time and protocol overhead are represented by the OFF periods within a packet call. For simplicity, the term "page" will be used in this paper to refer to each packet call ON period.

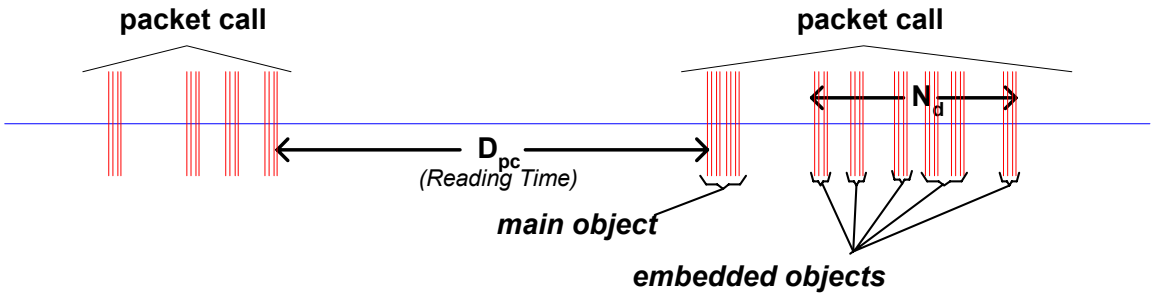

Figure 45: Contents in a Packet Call

The parameters for the web browsing traffic are as follows:

- S_M: Size of the main object in a page
- S_E: Size of an embedded object in a page
- N_d: Number of embedded objects in a page
- D_{pc}: Reading time
- T_p: Parsing time for the main page

HTTP/1.1 persistent mode transfer is used to download the objects, which are located at the same server and the objects are transferred serially over a single TCP connection as modelled in[5]. The distributions of the parameters for the web browsing traffic model are described in Table 17. Based on observed packet size distributions, 76% of the HTTP packet calls should use an MTU of 1500 bytes, with the remaining 24% of the HTTP packet calls using an MTU of 576 bytes. These two potential packet sizes also include a 40 byte IP packet header (thereby resulting in useful data payloads of 1460 and 536 bytes, respectively), and this header overhead for the appropriate number of packets must be added to the object data sizes calculated from the probabilistic distributions in Table 17.

Table 17: HTTP Traffic Model Parameters

Component	Distribution	Parameters	PDF
Main object size (S_M)	Truncated Lognormal	Mean = 10710 bytes Std. dev. = 25032 bytes Minimum = 100 bytes Maximum = 2 Mbytes	$f_x = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left[\frac{-(\ln x-\mu)^2}{2\sigma^2}\right], x \geq 0$ $\sigma = 1.37, \mu = 8.35$
Embedded object size (S_E)	Truncated Lognormal	Mean = 7758 bytes Std. dev. = 126168 bytes Minimum = 50 bytes Maximum = 2 Mbytes	$f_x = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left[\frac{-(\ln x-\mu)^2}{2\sigma^2}\right], x \geq 0$ $\sigma = 2.36, \mu = 6.17$
Number of embedded objects per page (N_d)	Truncated Pareto	Mean = 5.64 Max. = 53	$f_x = \frac{\alpha k^{\alpha}}{x^{\alpha+1}}, k \leq x < m$ $f_x = \left(\frac{k}{m}\right)^{\alpha}, x = m$ (NOTE) $\alpha = 1.1, k = 2, m = 55$
Reading time (D_{pc})	Exponential	Mean = 30 sec	$f_x = \lambda e^{-\lambda x}, x \geq 0$ $\lambda = 0.033$
Parsing time (T_p)	Exponential	Mean = 0.13 sec	$f_x = \lambda e^{-\lambda x}, x \geq 0$ $\lambda = 7.69$
Note: Subtract k from the generated random value to obtain N_d			

A.3.4.2FTP Traffic Model Characteristics

In FTP applications, a session consists of a sequence of file transfers, separated by *reading times*. The two main parameters of an FTP session are:

1. S : the size of a file to be transferred
2. D_{pc}: reading time, i.e., the time interval between end of download of the previous file and the user request for the next file.

The underlying transport protocol for FTP is TCP. The model of TCP connection described in [5]will be used to model the FTP traffic. The packet trace of an FTP session is shown in Figure 46.

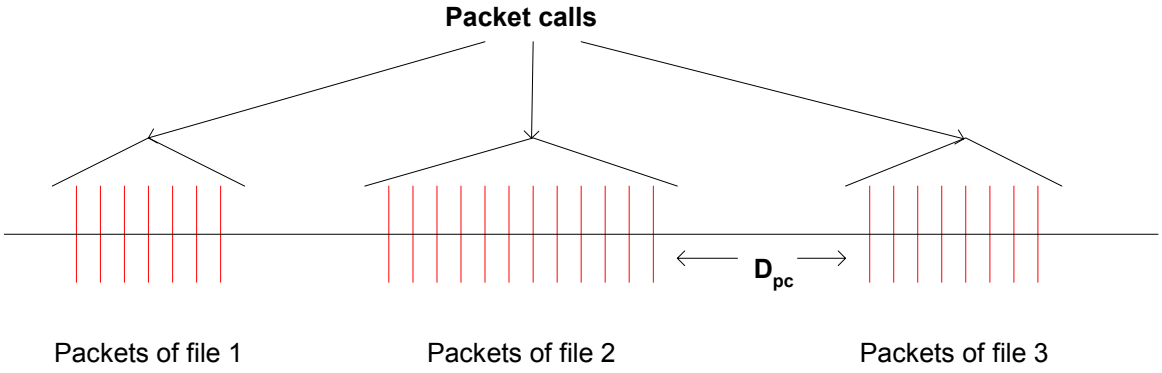

Figure 46: Packet Trace in a Typical FTP Session

The parameters for the FTP application sessions are described in Table 18.

Table 18: FTP Traffic Model Parameters

Component	Distribution	Parameters	PDF
File size (S)	Truncated Lognormal	Mean = 2Mbytes Std. Dev. = 0.722 Mbytes Maximum = 5 Mbytes	$f_X = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left[\frac{-(\ln x-\mu)^2}{2\sigma^2}\right], x \geq 0$ $\sigma = 0.35, \mu = 14.45$
Reading time (D_{pc})	Exponential	Mean = 180 sec.	$f_x = \lambda e^{-\lambda x}, x \geq 0$ $\lambda = 0.006$

Based on the results on packet size distribution, 76% of the files are transferred using and MTU of 1500 bytes and 24% of the files are transferred using an MTU of 576 bytes. Note that these two packet sizes also include a 40 byte IP packet header (thereby resulting in useful data payloads of 1460 and 536 bytes, respectively) and this header overhead for the appropriate number of packets must be added to the file sizes calculated from the probabilistic distributions in Table 18. For each file transfer a new TCP connection is used whose initial congestion window size is 1 segment (i.e. MTU).

A.3.4.3NRTV (Near Real Time Video) Traffic Model Characteristics

This section describes a model for streaming video traffic on the forward link. Figure 47 describes the steady state of video streaming traffic from the network, as seen by the base station. Latency at call startup is not considered in this steady-state model.

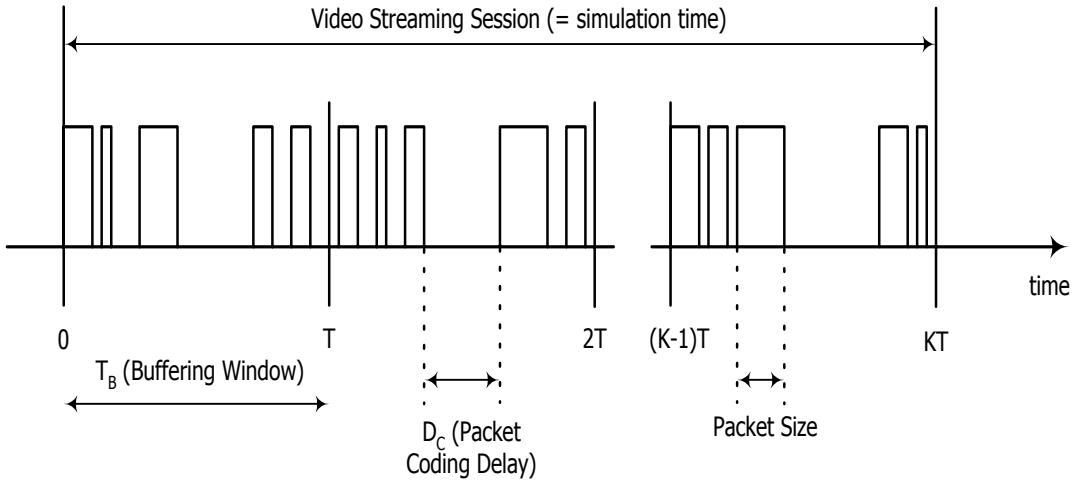

Figure 47: Video Streaming Traffic Model

A video streaming session is defined as the entire video streaming call time, which is equal to the simulation time for this model. Each frame of video data arrives at a regular interval T determined by the number of frames per second (fps). Each frame is decomposed into a fixed number of slices, each transmitted as a single packet. The size of these packets/slices is distributed as a truncated Pareto distribution. Encoding delay, Dc, at the video encoder introduces delay intervals between the packets of a frame. These intervals are modelled by a truncated Pareto distribution.
The parameter T_B is the length (in seconds) of de-jitter buffer window in the mobile station, and is used to guarantee a continuous display of video streaming data. This parameter is not relevant for generating the traffic distribution, but it is useful for identifying periods when the real-time constraint of this service is not met. At the beginning of the simulation, it is assumed that the mobile station de-jitter buffer is full with (T_B x source video data rate) bits of data. Over the simulation time, data is "leaked" out of this buffer at the source video data rate and "filled" as forward link traffic reaches the mobile station. As a performance criterion, the mobile station can record the length of time, if any, during which the de-jitter buffer runs dry. The de-jitter buffer window for the video streaming service is 5 seconds.
Using a source video rate of 64 kbps, the video traffic model parameters are defined in Table 19.

Table 19: Video Streaming Traffic Model Parameters.

Information types	Inter-arrival time between the beginning of each frame	Number of packets (slices) in a frame	Packet (slice) size	Inter-arrival time between packets (slices) in a frame
Distribution	Deterministic (Based on 10fps)	Deterministic	Truncated Pareto (Mean= 50bytes, Max= 250bytes)	Truncated Pareto (Mean= 6ms, Max= 12.5ms)
Distribution Parameters	100ms	8	K = 40 bytes α = 1.2	K = 2.5ms α = 1.2