

# NVE\_simulation manual

# Table of Contents

1Overview.....	3
2Simulation components.....	4
2.1Network topology.....	4
2.2Streams.....	5
2.3Messages.....	6
2.4Application layer protocol.....	8
3Running simulation.....	9
3.1Modules and dependencies.....	9
3.2Building NVE_simulation.....	9
3.3Executing a simulation.....	9
3.3.1Using graphical user interface.....	10
3.3.2Manual configuration .....	14
4Results.....	17

# 1 Overview

The purpose of this simulation tool is to provide a testing environment for networked virtual environment (NVE) applications.

This tool allows testing different types of traffic solutions when the number of clients, protocol configurations or network conditions change. The user of this tool is not required to do any programming to run the simulation, as this tool offers a graphical user interface (GUI) that allows the configuration of the simulation. The simulation output gives the user results of the executed simulation.

This tool uses ns-3 network simulator (<http://www.nsnam.org/>) modules in the actual simulation, i.e. all the networking functionality is provided by ns-3.

This manual is divided into following sections: Chapter 2 shows the conceptual architecture of the simulation, Chapter 3 deals with building and executing the simulation and Chapter 4 shows what the results are.

## 2 Simulation components

This chapter shows the conceptual architecture of the simulation tool.

### 2.1 Network topology

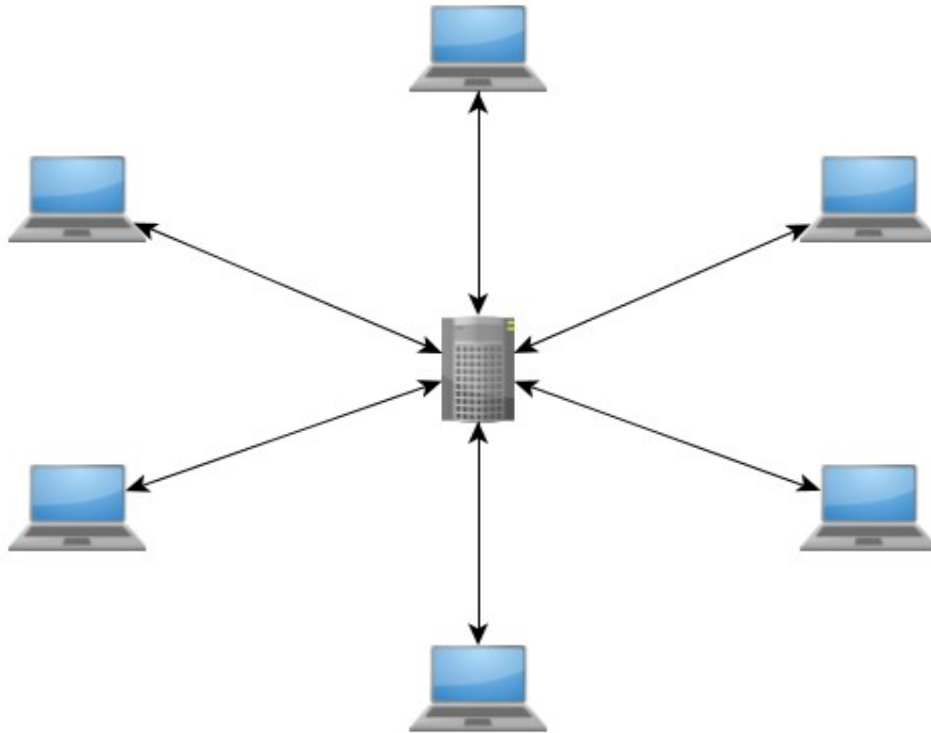


Figure above shows the network topology of the simulation. Multiple clients are connected to the server, and all client generated traffic travels via the server. Clients and the server are able to generate different types of application traffic, which can also be returned to the sender and forwarded to the other clients.

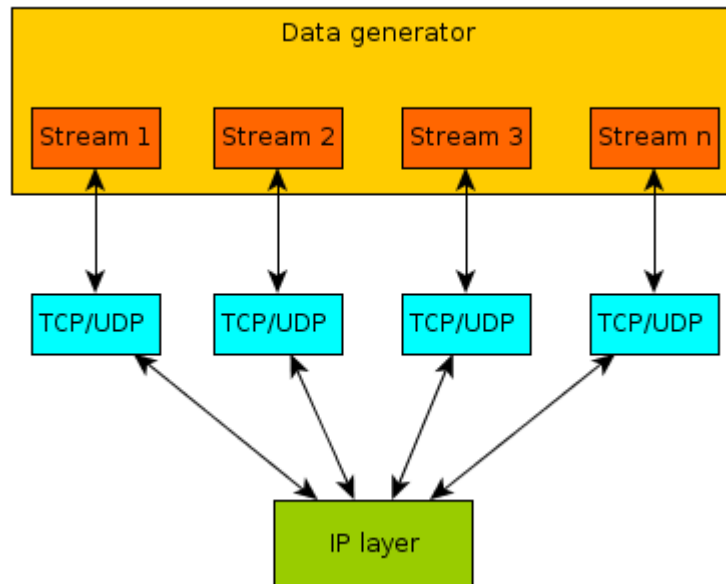
Each client has the following configurations:

- Network delay
- Packet loss
- Uplink bandwidth
- Downlink bandwidth
- Join time
- Exit time

In addition to the network conditions, clients can be configured to join and exit the server at any time in the simulation.

## 2.2 Streams

The following figure shows the model of the application that runs inside each client and the server.



The application can have multiple TCP or UDP streams. Each of the streams can contain different types of traffic.

Each stream has the following configuration options:

- Transport layer protocol (TCP and UDP available, SCTP not supported)
- Client game tick
- Server game tick
- Message types (see the next chapter)

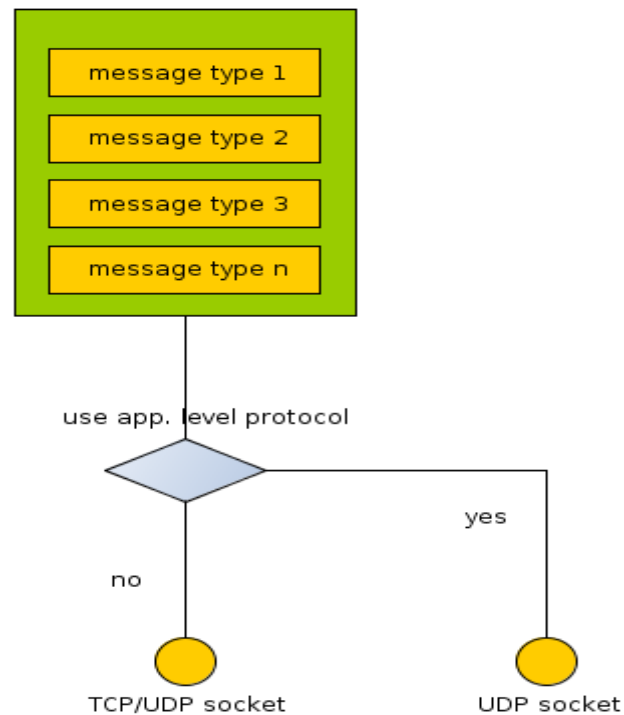
If UDP is used, the application layer protocol can also be enabled for the stream (see Chapter 2.4 for application protocol configuration). If the application protocol is enabled, the data of the stream can be declared to be ordered or unordered.

If TCP is used, Nagle's algorithm can be either disabled or enabled.

Client and server game tick values mean the send time interval of data from the application layer in client and server applications respectively. The data is buffered until the time runs out. Note that this is not necessarily the same as the time interval between packets in the network, as TCP may buffer data depending on the configuration.

## 2.3 Messages

Each stream can contain arbitrary number of messages, which all have different sizes and send time intervals. These values can either be constant or based on probability distributions. The next figure describes a single stream architecture, where multiple different messages are configured within a single stream.



Each message type contains the following configurable variables:

- Unique message name
- Reliability
- Message direction
- Message size
- Message time interval
- Return to sender
- Clients of interest
- Forward message size
- Time requirements

Each message type MUST have an unique name that identifies the message. The simulation tool uses the message type name and a ordinal number to identify the message. For example, if the message type name is “*test\_message*”, the messages generated from this message type are named “*test\_message:1*”, “*test\_message:2*”, “*test\_message:3*”, ..., “*test\_message:n*” etc.

Message direction declares whether the message is generated from the client to the server or from the server to the client.

Reliability option defines whether the message is reliable or not. This can only be used if UDP and the application protocol are used together.

Message size is the application data size of the message type. This size can be a constant number or it can be based on probability distributions.

NOTE: The generated message contents contain the message name, so the message size must always be big enough to contain the generated message name. For example, after running a simulation for an undefined period of time, the generated message name could be e.g. “*test\_message:10002*”. This whole name (including the “-signs”) must fit into the application data chunk, so the minimum application data size would be 20 in this case. The reason to include the message name is not only to identify messages, but to make it easier to track messages from Wireshark captures.

Message time interval is the time interval between two application generated messages. Like message size, it can be a constant value or it can be based on probability distributions.

Return to sender option defines if the message is returned to the sender. This works in both directions (client-to-server and server-to-client). Note that the message is not necessarily returned immediately, if there are e.g. game tick values present.

Clients of interest is a percentage that defines how many percentage of the clients receive the client generated message that the server receives. The server forwards messages to arbitrary clients when the server game tick runs out.

Forward message size is the message size of the message that is forwarded to other clients or returned to the sender. This value can be the same as the received message size, or it can be an independent constant value.

Time requirements define the time requirements that the message is desired to fulfill. This is used in result generation (see Chapter 4).

## 2.4 Application layer protocol

The tool contains also optional application layer protocol implementation, which offers reliable and ordered transfer also over UDP.

The application layer protocol has the following configuration options:

- Acknowledgement size
- Header size
- Retransmission timeout
- Delayed acknowledgment

Acknowledgement size means the size of the acknowledgement message (without network headers)

Header size means the size of the application protocol header that is added to application data packets. If there are multiple messages to be sent in a single socket write call, only one application header is used and messages are packed in the same application protocol datagram.

Retransmission timeout means the time interval when retransmissions are triggered, in case there are missing acknowledgements for the reliable data packets.

Delayed acknowledgement is the value to buffer the acknowledgement in order to include multiple acknowledgements inside the same network packet.



## 3 Running simulation

This chapter describes how to build the environment and how to configure and execute simulations. Note that this tool requires unix platform. These instructions have been tested on Ubuntu 12.04.

### 3.1 Modules and dependencies

The NVE\_simulation tool three modules: ns-3 network simulator, the NVE simulation model and graphical user interface. The included ns-3 version is ns-3.17 with small modifications to the probability distributions functionality. By default, only the required ns-3 modules are linked to the NVE\_simulation tool during compile time. If extra functionality from ns-3 is needed, the project file must be edited.

This tool requires *Rscript* and *tshark* to generate the statistics.

Qt is used to provide the GUI functionality.

### 3.2 Building NVE\_simulation

The NVE\_simulation tool contains a build script, that builds (probably) all the modules and dependencies. To run the script, go to the project folder and simply type

```
sh build.sh
```

This should build the tool. In case something fails, figure it out yourself!

### 3.3 Executing a simulation

After a successful build, this simulation tool can be executed by typing

```
./NVE_simulation
```

This starts the GUI. If there is something wrong with you and you don't want to use the GUI (or you want to automatize simulations) the tool can be executed from the command line. The options are

```
--filename <file>
```

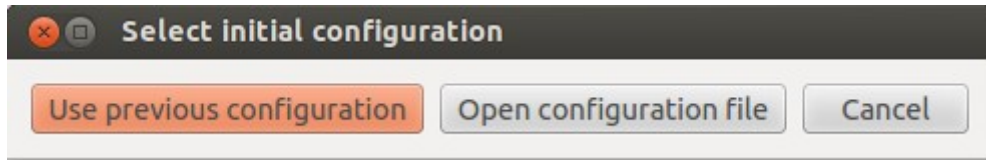
```
--help
```

```
--verbose
```

If any of these options are used, the GUI is not used. Running simulation this way requires that filename is given, and the file is a correct simulation configuration (see Chapter 3.3.2). Verbose option prints extra info about the configuration, and help surprisingly gives you some help.

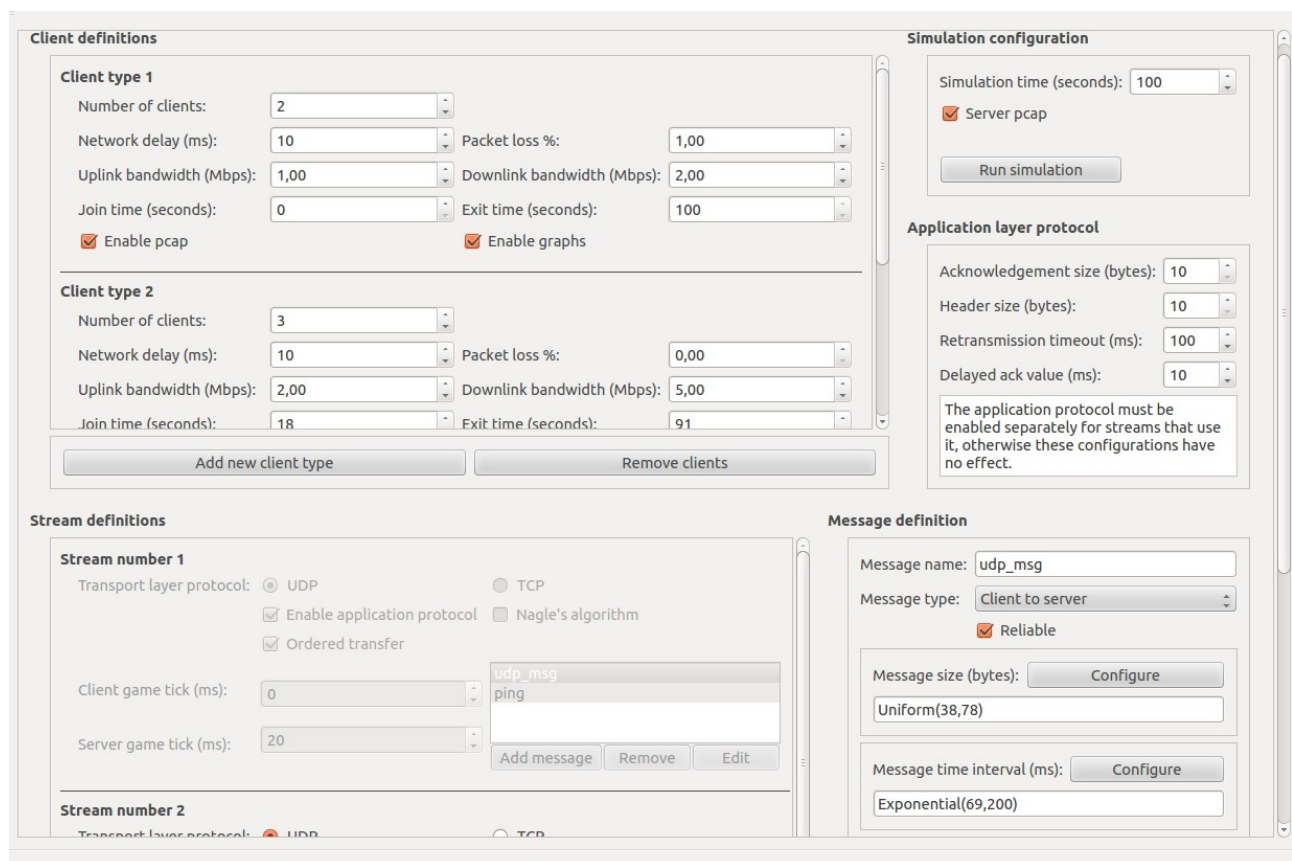
### 3.3.1 Using graphical user interface

When GUI is started, it asks the user if the user wants to use previous configuration.



If “Use previous configuration” is selected, the configuration is loaded from file named “configuration.txt”, which contains always the last simulation executed from the GUI. “Open configuration file” allows using other saved configurations, and “Cancel” lets the user build a simulation from a scratch.

The GUI looks like the following, and it contains the configuration options described in Chapter 2:



**Client definitions**

**Client type 1**

Number of clients: 2  
Network delay (ms): 10  
Uplink bandwidth (Mbps): 1,00  
Join time (seconds): 0  
☒ Enable pcap

Packet loss %: 1,00  
Downlink bandwidth (Mbps): 2,00  
Exit time (seconds): 100  
☒ Enable graphs

**Client type 2**

Number of clients: 3  
Network delay (ms): 10  
Uplink bandwidth (Mbps): 2,00  
Join time (seconds): 18

Packet loss %: 0,00  
Downlink bandwidth (Mbps): 5,00  
Exit time (seconds): 91

**Simulation configuration**

Simulation time (seconds): 100  
☒ Server pcap  
Run simulation

**Application layer protocol**

Acknowledgement size (bytes): 10  
Header size (bytes): 10  
Retransmission timeout (ms): 100  
Delayed ack value (ms): 10

The application protocol must be enabled separately for streams that use it, otherwise these configurations have no effect.

**Stream definitions**

**Stream number 1**

Transport layer protocol: ☒ UDP ☐ TCP  
☒ Enable application protocol ☐ Nagle's algorithm  
☒ Ordered transfer

Client game tick (ms): 0  
Server game tick (ms): 20

udp\_msg  
ping

Add message Remove Edit

**Stream number 2**

Transport layer protocol: ☒ UDP ☐ TCP

**Message definition**

Message name: udp\_msg  
Message type: Client to server  
☒ Reliable

Message size (bytes): Configure  
Uniform(38,78)

Message time interval (ms): Configure  
Exponential(69,200)

Client definitions box (the next figure) allows to configure different client types.

The 'Client definitions' window is divided into two sections: 'Client type 1' and 'Client type 2'. Each section contains several input fields for configuring client parameters. At the bottom, there are two buttons: 'Add new client type' and 'Remove clients'.

Parameter	Client type 1	Client type 2
Number of clients	2	3
Network delay (ms)	10	10
Uplink bandwidth (Mbps)	1,00	2,00
Downlink bandwidth (Mbps)	2,00	5,00
Join time (seconds)	0	18
Exit time (seconds)	100	91
Packet loss %	1,00	0,00
Enable pcap	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Enable graphs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

In addition to the client parameters described in Chapter 2, each client type can also have a number of clients that are created based on this client type. For example, in the configuration above, client type 2 has 3 clients which are connecting to the server after 18 seconds of simulation. It is also possible to disable graph generation and pcap file creation in order to speed up the execution time of the simulation (but less results are generated).

The figure on the right contains the overall simulation configuration and the application protocol configuration box. The server pcap file generation can also be disabled, but it reduces the amount of statistics created.

The 'Simulation configuration' window contains two main sections. The top section, 'Simulation configuration', includes a 'Simulation time (seconds)' field set to 100, a 'Server pcap' checkbox which is checked, and a 'Run simulation' button. The bottom section, 'Application layer protocol', includes four input fields: 'Acknowledgement size (bytes)' (10), 'Header size (bytes)' (10), 'Retransmission timeout (ms)' (100), and 'Delayed ack value (ms)' (10). A text box at the bottom of this section states: 'The application protocol must be enabled separately for streams that use it, otherwise these configurations have no effect.'

The stream configuration is shown in the following figure. Message types for each stream can be added and edited by using the message list.

**Stream definitions**

**Stream number 1**  
 Transport layer protocol: ☒ UDP ☐ TCP  
☒ Enable application protocol ☐ Nagle's algorithm  
☒ Ordered transfer  
 Client game tick (ms): 0  
 Server game tick (ms): 20  
 Message list: udp\_msg, ping  
 Buttons: Add message, Remove, Edit

**Stream number 2**  
 Transport layer protocol: ☒ UDP ☐ TCP  
☐ Enable application protocol ☐ Nagle's algorithm  
☐ Ordered transfer  
 Client game tick (ms): 0  
 Server game tick (ms): 0  
 Message list: unreliable\_udp\_msg  
 Buttons: Add message, Remove, Edit

**Stream number 3**  
 Transport layer protocol: ☐ UDP ☒ TCP  
☐ Enable application protocol ☒ Nagle's algorithm  
☐ Ordered transfer  
 Buttons: Add stream, Remove stream

Message definition box is shown on the right.

Message sizes and time intervals can be configured by clicking “Configure”, which opens a new window that allows to define probability distributions. If user wants to use a constant value, a Constant distribution must be chosen (example in the figure below).

**NVE\_simulation**

Constant  
 Constant value: 22,00  
 Buttons: Cancel, OK

**Message definition**

Message name: unreliable\_udp\_msg  
 Message type: Client to server  
☐ Reliable  
 Message size (bytes): Configure  
 Normal(100,5)  
 Message time interval (ms): Configure  
 Split(70%:Uniform(20,100),30%:Constant(200))  
☒ Return to sender  
 Clients of interest (%): 52,00  
 Forward message size (bytes):  
☒ Use received message size  
☐ Use constant size: 1  
 Time requirement to reach client (ms): 90  
 Time requirement to reach server (ms): 40  
 Buttons: Ok, Cancel

The distribution window contains all the probability distributions from ns-3, and it also has implementations of extreme and split distributions in addition to ns-3 distributions.

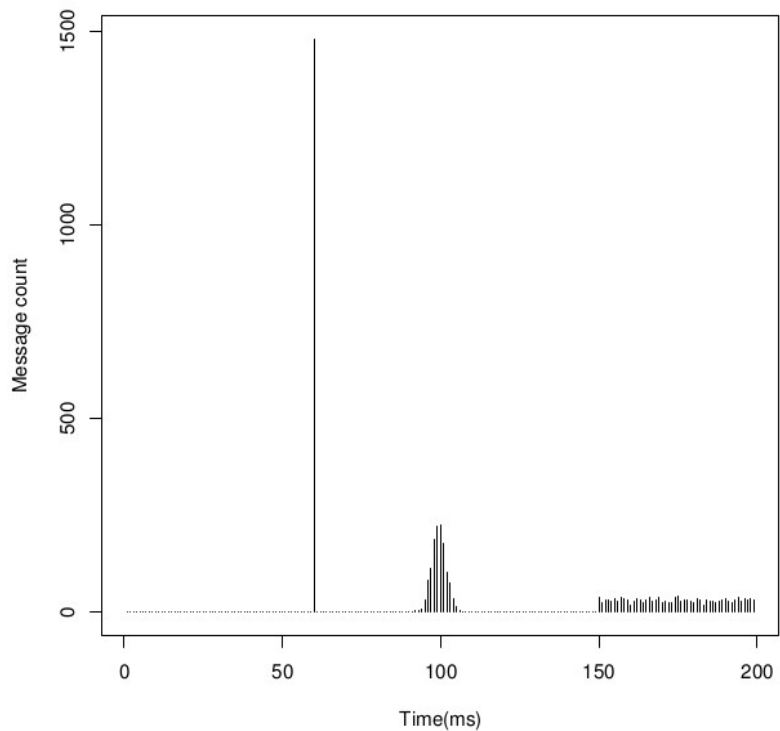
The split distribution allows the user to define a probability distribution that is comprised of multiple individual distributions. Each distribution included in split distribution must contain a percentage, which defines the density percentage of the distribution in the split distribution. The NVE\_simulation tool takes care that the density cannot exceed 1.

The two figures below show an example of a split distribution configuration and the followed results.

The screenshot shows a dialog box titled "NVE\_simulation" with a "Split" dropdown menu. Below it, there are three rows for defining distributions:

Distribution	Percentage
Distribution 1: Normal(100,5)	30,00
Distribution 2: Constant(60)	35,00
Distribution 3: Uniform(150,200)	35,00

At the bottom, there are "Add", "Cancel", and "OK" buttons. The "Add" button is highlighted.



ns-3 has also a probability distribution called *empirical distribution*. It allows ns-3 to generate random variables from empirical data set. This can be also used from the simulation tool, and it requires that counts of empirical values are located in a comma-separated file.

For example, saving values

10,0,20,21,21,30,40,41,42,43,44,44,44,44,45,45,46,47,48,49,50,100,79,55

to a text file "emp.txt", and by giving "emp.txt" as a parameter to a distribution, would mean that there is 10 number 1 values, 0 number 2 values, 20 number 3 values etc. I.e the index of the value (starting from 1) shows the number of values of the index.

### 3.3.2 Manual configuration

It is also possible to configure the simulation by modifying a text file. The configuration file uses an XML-like structure. In the following example, client and stream definitions are shortened with ... and are described later. The whole configuration is wrapped within <xml> tags. The ending tag of a struct has always extra / before the text, like </xml>. Single variable is written e.g. <runningtime="100"/>.

The xml-struct must contain the following elements (in the following example, client and stream definitions are shortened with ... and are described later):

```
<xml>
  <runningtime="100"/>
  <serverpcap="no"/>
  <clients>
    <client>
      ...
    </client>
  </clients>
  <appproto>
    <acksize="10"/>
    <delayedack="10"/>
    <retransmit="100"/>
    <headersize="10"/>
  </appproto>
  <streams>
    <stream>
      ...
    </stream>
    <stream>
      ...
    </stream>
  </streams>
</xml>
```

<clients> structure can contain multiple client definitions, like in the following example:

```

<clients>
  <client>
    <no="1:3"/>
    <delay="10"/>
    <uplink="1"/>
    <downlink="2"/>
    <loss="0.01"/>
    <jointime="0"/>
    <exittime="100"/>
    <pcap="yes"/>
    <graphs="yes"/>
  </client>
  <client>
    <no="4"/>
    <delay="10"/>
    <uplink="2"/>
    <downlink="5"/>
    <loss="0"/>
    <jointime="18"/>
    <exittime="91"/>
    <pcap="yes"/>
    <graphs="yes"/>
  </client>
</clients>

```

Client count and number are defined in “no” parameter. In case of a single client, only client ordinal number is used. If multiple clients are defined within single type, notation “x:y” is used, where x is the number from and y is the number to, e.g. 1:3 means 3 clients, numbers 1, 2 and 3.

The packet loss percentage value must be defined as a double from 0 to 1.

The following example shows how to configure streams. Each stream must have <messages> structure, that contains arbitrary number of messages defined in <message> structure. Message type is either “uam” or “odt” (“user action message” and “other data transfer”). “uam” stands for client-to-server message and “odt” server-to-client message. Probability distributions in message sizes and time intervals are defined with the distribution name and parameters, e.g. `Exponential(69,200)` or `Constant(30)`. Split distribution definition include also percentages, e.g. `Split(30%:Normal(100,5),35%:Constant(60),35%:Uniform(150,200))`

Clients of interest percentage must be a double value between 0 and 1.

```
<streams>
  <stream>
    <type="udp"/>
    <nagle="no"/>
    <appproto="yes"/>
    <ordered="yes"/>
    <servergametick="20"/>
    <clientgametick="0"/>
    <messages>
      <message>
        <type="uam"/>
        <name="udp_msg"/>
        <size="Uniform(38,78)"/>
        <reliable="yes"/>
        <timeinterval="Exponential(69,200)"/>
        <forwardmessagesize="rcv"/>
        <returntosender="no"/>
        <timerequirementclient="100"/>
        <timerequirementserver="50"/>
        <clientsofinterest="1"/>
      </message>
      <message>
        <type="odt"/>
        <name="ping"/>
        <size="Constant(30)"/>
        <reliable="no"/>
        <timeinterval="Constant(3000)"/>
        <forwardmessagesize="35"/>
        <returntosender="yes"/>
        <timerequirementclient="100"/>
        <timerequirementserver="50"/>
        <clientsofinterest="1"/>
      </message>
    </messages>
  </stream>
```



## 4 Results

The NVE\_simulation tool provides a various number of results that are saved in directory called “Results”.

Graphs are generated using *Rscript* and they are saved in a pdf file “resultgraphs.pdf”. Graphs include at least message transmission times for each stream, message transmission times for each message (including time requirement comparison) and bandwidth graphs. If the graphs option is enabled, each client gets an own graph that shows the actual packet time intervals and sizes in the network. If the server graphs option is enabled, an overall bandwidth usage graph over the simulation time is created.

Part of the results are generated from the pcap files, which are generated for each client and the server by default.

The NVE\_simulation tool also creates a text file “result.txt”, that contains numeric results of the transmission times.

The transmission times are measured from application to application, so the timer will start when the application decides to send a message (i.e. game tick buffering, protocol configurations and network delay affect the value).