In the name of God

CLIENT FRAMEWORK (CF)
USER MANUAL

PREPARED BY:

BEHNAM MOMENI

1 Introduction

Client Framework (CF) is a part of Partov project. This framework provides the ability to simulate variety of virtual networks by connecting to the central Partov server. Client Framework is consisted of two main parts: 1) base (which includes all of the operations for creating/instantiating topology instances (maps), making connections with Partov server and other systematic operations) and 2) user part (which includes the user program which will be executed over a virtual node within the map).

2 Method of Programming

For simulating the virtual node, you could add all required files of your program to the “user” folder and do required changes in the “sm.cpp” file in that folder. The “sm.cpp” file includes code of “SimulatedMachine” class. By running CF an instance of this class will be created and right after preparing virtual node's interfaces information (including IP address and MAC address for each interface) its “initialize” method will be called. You could initialize your data-structures in this method.

This class is child of “Machine” class. The “Machine” class provides access to information of node's interfaces through two protected fields, namely “iface” and “countOfInterfaces”. The “iface” is an array of “Interface” class instances.The “Interface” class structure is like followings:

```
Interface {
    byte mac[MAC_ADDRESS_LENGTH];
    uint32 ip;
    int32 mask;
}
```

Note that this array isn't valid before calling “initialize” method (in the constructor). After returning from “initialize” method, the “run” method will be called. The “run” method makes it possible to have two threads of execution out of the box (main and run). You could use “run” thread for operations which are not dependent on received packets (so you do not need to wait for the first incoming packet to start simulation). Returning from this method does not terminate the program and the main thread will continue execution.

Whenever a packet received by the virtual node on one of its interfaces, the “processFrame” method of “SimulatedMachine” class will be called by two arguments. The first argument has “Frame” type with following structure:

```
Frame {
    uint32 length;
    byte *data;
}
```

The second argument is an “index” which indicates the interface which packet is received by it (in the “iface” array). The “data” field of the received packet, points to the first byte of the frame. This frame is placed in a shared buffer and will be overwritten after returning from “processFrame” method. So if you need the content of the frame after returning from this method, you should copy data to another buffer. You could use each one of “processFrame” and “run” methods for sending packets over each one of interfaces. For this purpose you could use following method (from the “Machine” class):

```
bool synchronized sendFrame (Frame frame, int ifaceIndex) const;
```

This method has similar arguments to the “processFrame” method nad will send the packet over the requested interface. It return *true* on success and *false* on failure.

Also the user program could parse its own command line arguments using following static method (from the “SimulatedMachine” class):

```
static void parseArguments(int argc, char *argv[]);
```

The framework will call this static method after parsing its own arguments. The user program arguments (which should be marked with “--args”) will be passed to the “parseArguments” method. In this method, “argc” is count of user program arguments (count of arguments which are placed in front of “--args” when running CF) and “argv” is an array of those arguments.

For correct operation of the framework, no one of “base” folder files should be changed and address of files which are added to the “user” folder must be included in

the “USER_SOURCES” variable in the “Makefile” file to be compiled. You could compile the framework with *make* command in the CF root address and remove all created binary files by *make clean* command.

3 Running the Framework

After implementing your program and compiling the framework, the executable file, namely “cf.out”, will be created. Run “./cf.out --help” to see the internal help of the framework. A sample of running the framework is placed in the “run.sh” file. Framework acceptable command line arguments are shown in the Table 1.

Argument name	Example	Explanation
--ip <server-ipv4>	--ip 213.233.168.9	The IP (version 4) of the Partov server
--port <server-port>	--port 9339	The port of the Partov server (currently 9339)
--map <map-name>	--map bridge	The name of the map which you want to connect to
--user <user-name>	--user ali	Your username (for authentication)
--pass <password>	--pass sda;u0,4	Your password (for authentication)
--node <node-name>	--node ap1	(optional) The name of the node which you want to simulate it within selected map
--id <creator-username>	--id ali	Try to connect to the map instance which was created by <creator-username> user previously
--new	--new	Try to create a new map instance.
--free	--free	Free resources of the map instance which was created by this user.
--args <arg>[<arg>...]	--args 12 hl	(optional) The arguments which come after --args will be passed to the SimulatedMachine::parseArguments method (must be last argument).

Table 1: CF command line arguments

Each user requires a username/password (which will be sent by email) in order to connect to the Partov server. After identifying IP/port of the server and your username/password, you could connect to server (sign in). Then the desired map will be founded and one instance of it will be assigned to your session. In this phase you could only use one of three arguments “--id”, “--new”, or “--free”.

At the beginning you should use “--new” to instantiate a map (which is identified by the “--map” argument). The instantiated map will be assigned to you. Note that each user could only instantiate one map from each topology at a time. So if you try to instantiate more maps (of the same topology) you will face an error. After end of your simulation, it's required to run framework with “--free” argument so the instantiated map and its resources could be released. After releasing one map, you could “--new” it again.

When you connect to a map, the value of “--node” argument will be used to identify the virtual node (within the map) which you want to simulate it. If you want to simulate multiple nodes of a map concurrently (like simulating 4 bridges in a LAN), or if you were disconnected from the server accidentally and you want to resume the simulation, you could use “--id <username>” argument. For example for resuming simulation of “bdg1” you could use following command:

```
./cf.out --ip 213.233.169.0 --port 9339 --map bridge --node bdg1 --user ali --pass sda;u0,4 --id ali
```

Or for simulating three nodes, namely “bdg1”, “bdg2”, and “bdg3” in one map concurrently you could use following commands:

```
# first run this...
./cf.out --ip 213.233.168.9 --port 9339 --map bridge --node bdg1 --user ali --pass sda;u0,4 --new
# and in another terminal run this...
./cf.out --ip 213.233.168.9 --port 9339 --map bridge --node bdg2 --user ali --pass sda;u0,4 --id ali
# and in yet another terminal, run this...
./cf.out --ip 213.233.168.9 --port 9339 --map bridge --node bdg3 --user ali --pass sda;u0,4 --id ali
```

And at the end you should free the map with following command:

```
./cf.out --ip 213.233.168.9 --port 9339 --map bridge --user ali --pass sda;u0,4 –free
```

4 Possible Errors

In the Table 2 you could find a list of possible errors. If you encountered any issue, check this table to find possible solutions. If issue remained unsolved, please let us know about it via email describing the exact process of triggering the issue.

Code	Name	Description	Solution
5	Map not exists	The requested map does not exists on the server	Check the name after --map argument. Names are case-sensitive.
6	Duplicate map id	Another map with same ID exists currently	Each user could instantiate one map from each topology at most. Either free previous instance or resume it using --id argument.
7	Out of resource	Server does not have enough resources to instantiate a map	Each map requires some IP addresses. If other persons use all IP addresses by instantiating a lot of maps, server could not instantiate another map. You have to wait till another person release a map by --free argument.
12	Node not exists	There is no node with the requested name in the map	• Maybe node name entered incorrectly. Names are case-sensitive, • Maybe you are connected to that node from another terminal. Note that each node in each map could only be simulated by one CF. If node name is correct, --free map and --new it again.
31	Error in sending frame in simulated machine	Length of the sent packet is invalid	Length of frames which are sent by “sendFrame” method should be between 14 and 1514 bytes (do NOT include 4 bytes CRC of the Ethernet trailer).
42	Username or password is incorrect	Either your username or your password is misspelled (they are case-sensitive)	Check username/password again. If you saved your password in a file, check the file encoding.
Others	Negotiations failed	An error occurred in network communication	Check your network connection. Use w wired connection. Check your firewall settings. Retry.

Table 2: Possible errors and their solution