*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

User Manual

Addigy4

Team # 01

**Team Members**

Carlos Ruiz

David Romero

**Product Owner**: Jason Dettbarn

**Instructor**: Masoud Sadjadi

**Table of Contents**

Introduction................................................................................................................................... 2

Server Installation......................................................................................................................... 2

Client/Tunnel Installation...............................................................................................................2

Using Client/Tunnel Software …………………………………………………………………………..2

**Introduction**

In today's computer world it is practically impossible to create a straight connection from one computer to the other in order to communicate, or transfer a file, or even doing remote control. This study describes and evaluates a solution for this problem working behind firewalls, routers, and any security system preventing this connections among computers to be created.

This system will use a third party (the server) in order to establish a connection between two registered systems. This dedicated server will be in charge of redirecting the communication using TCP connections and a personalized protocol build on top of it. Both of the machines will also have a service running which will be in charge of communicating with the server.

Communication between devices is a centerpiece of modern day computing as well. As such, there are many different ways in which devices can establish this communication, however how do we approach this in a way where we keep it scalable and simple to use. After much research, it was determined that a publish/subscribe model would be used.

This software will use RabbitMQ as the server that handles the creation and dispatching of messages. The software itself will simple facilitate the communication to said server,

abstracting the process in addition to adding new features such as presence and an idle timeout. The process is scaling off the existing APIs.

**Server Installation:**

1. Make sure server has python environment running. Target version python2.7
2. Download project zip file:
   1. Open terminal or from ssh session type: $ *wget https://github.com/FIU-SCIS-Senior-Projects/Addigy4/archive/master.zip*
   2. unzip master.zip file by typing: $ *unzip master.zip*
   3. navigate to Server software directory: $ cd Addigy4-master/Code/tunneler/server/
   4. Change setup.sh permission by: $ chmod +x setup.sh
   5. then run: $ ./setup.sh
   6. Thats it! Server shall be up and running.

**Client/Tunneler** installation:

1. Make sure computer has python installed. Target python version is python2.7.
2. Download project zip file:
   1. Open terminal or from ssh session type: $ *wget https://github.com/FIU-SCIS-Senior-Projects/Addigy4/archive/master.zip*
   2. unzip master.zip file by typing: $ *unzip master.zip*
   3. navigate to Server software directory: $ cd Addigy4-master/Code/tunneler/
   4. change permission of setup.sh files for **Client/Tunneler:** $ chmod +x client/setup.sh and then $ chmod +x tunneler/setup.sh
   5. run both directories setup.sh files: $ ./client/setup.sh and $ ./tunneler/setup.sh
   6. Now both directories are going to be located on /var/opt. To find them open terminate and type: $ cd /var/opt

Running **Client**/**Tunneler:**

1. After installation is completed, navigate to /var/opt.
   1. Open terminal for each program.
   2. Type: $ cd /var/opt
   3. **Client** and **Tunnel** boths will have to be updated:
      1. open their directory and navigate to "ServerObject.py"
      2. find "**def \_\_init\_\_(self):**" and update "**self.\_\_url**" with the url where the Server is going to be running
      3. save and exit
2. Running Tunneler
   1. Type: $ python tunneler/Main.py <tunnelId>

* tunnelId: needs to be 36 byte id and unique
* sample: **(fc86c7ef-f579-4115-8137-289b8a257803)**

3. Running Client

1. Type: $ python client/Main.py <tunnelId><local port><tunnel port>

* tunnelId: id for the tunnel you wish to communicate with
  + - sample: **(fc86c7ef-f579-4115-8137-289b8a257803 3000 22)**
  + local port: local port in which Client software is going to be listening
  + tunnel port: port number to which the Tunneler software is going to send the request

**PubSub Installation:**

1. A python installation is necessary in order to make PubSub function. Any installation post 2.7 will work, however the latest version is advised.
2. Download RabbitMQ from <https://rabbitmq.com/> and navigate to the download page. Here select your operating system and follow the subsequent steps.
   1. On any platform, if Erlang isn’t installed, the installation process will inform you and walk you through the process.
   2. Furthermore, if homebrew is installed on OS X, a simple brew install rabbitmq command can be issued. Similarly on Debian/Ubuntu systems, sudo apt-get install rabbitmq can be issued.
3. Download the project zip
   1. Download the project zip from <https://github.com/FIU-SCIS-Senior-Projects/Addigy4/>.
      * On Linux and OS X, wget <https://github.com/FIU-SCIS-Senior-Projects/Addigy4/archive/master.zip> will also do.
      * Once downloaded, unzip the project. On Linux and OS X systems, on the terminal enter unzip master.zip
4. If a version of Python below 2.7.9 and 3.4 is in use, setuptools will need to be used. This can be obtained from <https://pypi.python.org/pypi/setuptools>. Navigate to your operating system and follow the listed steps.
5. In our extracted project, move the setup bat or shell script with your platform name from the PubSub root folder to the sbin folder in your RabbitMQ installation location.
   1. Keep in mind that in order for the shell scripts to run, the permissions may need to be changed, which can be done by inputing chmod +x client/PubSubSetup[OS].sh in the terminal (where [OS] is replaced by your operating system).
6. In the command line or terminal, navigate to the installation location of RabbitMQ and enter either rabbitmq-server.bat (or sudo rabbitmq-server.sh on Debian/Ubuntu systems) start to successfully start the service.
   1. Note: on Linux, the script can be run from virtually anywhere.

**Running PubSub (Python):**

1. Start by opening the location in which the project was downloaded.
2. Move the class to the location in which the project implementing the solution.
3. Import the pubsub class. After, instantiate pubsub by the address our server, the name of the queue (please note it's important to specify this even if we are unsure if the queue exists already), the organization (channel) and user credentials (username and password).
   1. For local hosting, guest guest is the default user. In order to add new users, we must navigate to the same sbin folder and start rabbitmqctl.bat or sudo rabbitmqctl.sh on Mac OS X with add\_user, passing the username and password. On Debian/Ubuntu systems, sudo rabbitmqctl add\_user can be issued from anywhere
   2. After we must set a tag for our newly created user by issuing a rabbitmqctl.bat (sudo rabbitmqctl.sh) set\_user\_tags and passing the username and the tag. i.e. rabbitmqctl.bat david administrator
4. To publish, simply pass a routing\_key (which is often just the queue name) and the message we wish to send (routing\_key=[key], body=”text”).
5. To subscribe provide a callback function and a queue name (queue\_name=name)

**Running PubSub (Javascript):**

1. Start by opening the location in which the project was downloaded.
2. Move the class to the location in which the project implementing the solution.
3. Import the pubsub class. After, instantiate pubsub by the address our server, the name of the queue (please note it's important to specify this even if we are unsure if the queue exists already), the organization (channel), user credentials (username and password), heartbeats\_incomming and heartbeats\_outcoming (to disable, set to 0).
4. To publish, send either “/exchange/[organization].messages” or “/queue/[name]”, the type of content that being sent ({“content-type”:”text/plain”} for regular text), the callback function and the name of the queue. i.e. PUBSUB.publish(‘/exchange/testcorp.messages/test1’, {“content-type”:”text/plain”}, callback(param), ‘Test’);
5. To subscribe provide either “/exchange/[organization].messages” or “/queue/[name]”, a callback function and a queue name. i.e. PUBSUB.subscribe("/exchange/testcorp.messages", function(d) {console.log(); }, "Test")
   1. For more information on how to send information via the stomp adapter, visit <https://www.rabbitmq.com/stomp.html>