A Simple Peer To Peer Network Implementation

Hoora Abootalebi
Nariman Aryan
Amin Isaai
Amirhossein Khajepour
Mahdis Tajdari
Ali Zeynali

November 2018

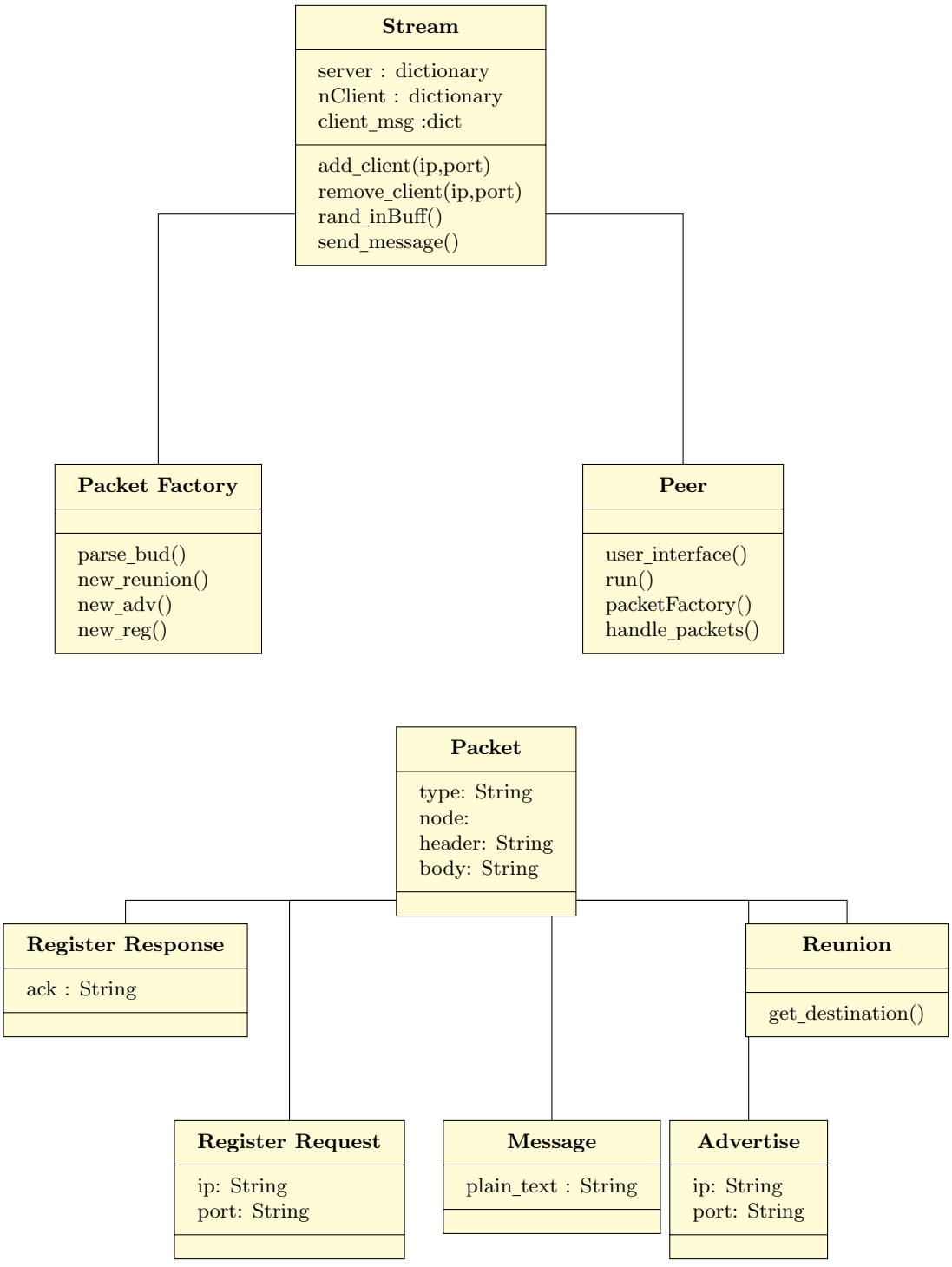
Contents

1 Introdeuction

This project aims to implement a peer to peer network. In the first step we design UML model and then we are going to explain each objects' attributes and methods.

2 UML model

We design the UML model in order to make the project more understandable, clearer and professional.


Stream
server : dictionary
nClient : dictionary
client_msg :dict
add_client(ip,port)
remove_client(ip,port)
rand_inBuff()
send_message()
Packet Factory
parse_bud()
new_reunion()
new_adv()
new_reg()
Peer
user_interface()
run()
packetFactory()
handle_packets()
Packet
type: String
node:
header: String
body: String
Register Response
ack : String
Reunion
get_destination()
Register Request
ip: String
port: String
Message
plain_text : String
Advertise
ip: String
port: String

3 Objects

Now it's time to explain every obejct's duty.

3.1 Streem

```
#stream()
add_client(ip,port)
remove_client(ip,port)
rand_inBuff()
send_message()
rand_inBuff()
```

We also need to add a dictionary to specify every client's message(s)

3.2 Peer

```
#Peer()
stream()
user_interface() #Which the user or client sees and works with.
run()           #This method runs every time to see
                #whether there is new messages or not.
packetFactory()
handle_packets()
```

3.3 Packet Factory

packetFactory() would generate the packets every node needs to connect another with.

```
#packetFactory
parse_buf()
new_reunion()
new_adv() #make a new advertise packet.
new_reg() #make a new register packet.
```

3.4 Packet

Every packet consists seven differntes parts: :**plain_text** which is the raw text message in the packet.

Node Sender Validator which make the packet valid.
Header where the information such as type of the packet and etc. are going to be there.
Body Action

3.5 Reunion

reunion(packet) checks the connection of the nodes to the root.

```
#reunion(packet)
 get_destination()
```

3.6 Node

Every node has two parameters: **IP** and **Port**.

3.7 Resgister Request

reg_req() sends IP/Port of a node to the root to ask if it can register it.

3.8 Register Response

reg_res() should just send an from the root *Ack* to inform a node that it has been registerd in the root if the reg_req() was successful.

3.9 Advertise

adv(packet)

3.10 Mesasge

msg(packet)