

Innopolis University
SYSTEM AND NETWORKING ENGINEERING



Distributed Systems

INDIVIDUAL ASSIGNMENT 1

Chat Application

Student
Grebennikov Sergey

ID
47611

Lecturer
Konstantin Urysov

November 2, 2017

Contents

1	Project description	2
2	Architectural diagram	2
3	Design	2
4	Launch	3
5	Implementation requirements	3

1 Project description

Communication is the essential part of every distributed system. In this assignment you will have to develop simple chat application based on socket programming.

GitHub: https://github.com/vkaser/chat_DS_SNE_2017.git

DockerHub: <https://hub.docker.com/r/vkaser/server/> and <https://hub.docker.com/r/vkaser/client/>

2 Architectural diagram

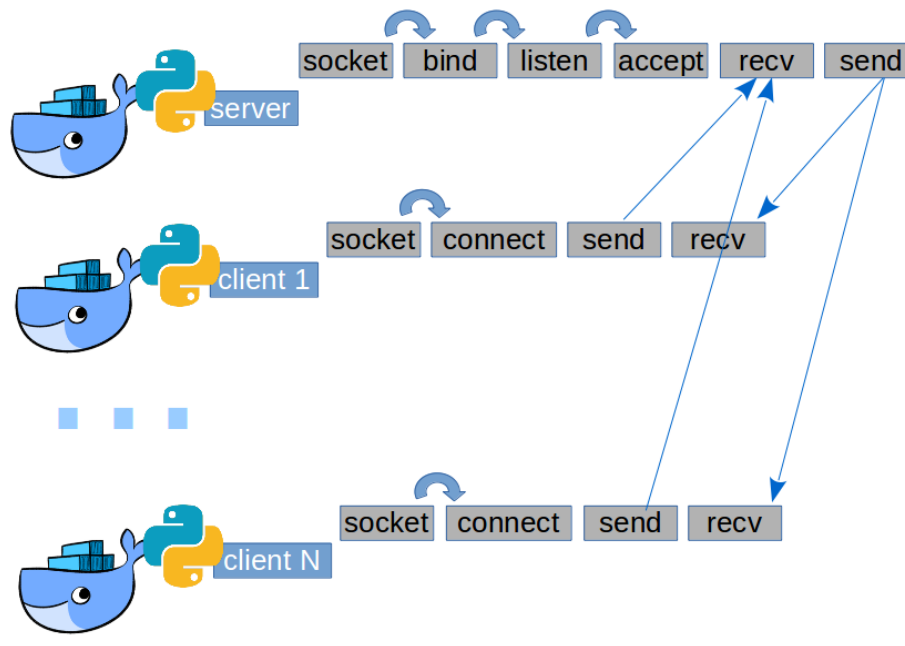


Figure 1: Architectural diagram

3 Design

Console Application:

```
sergey@ns1: ~/Documents/chat/client 66x8
Chat Server started on port: 5555
Client [172.17.0.3, 48126] connected
Client [172.17.0.4, 47092] connected
Client [172.17.0.5, 53086] connected
Distributed_
Systems_3-170928.
pdf

sergey@ns1: ~/Documents/chat/server 66x55
----- InnoChat -----
*** [172.17.0.4:47092] joined the chat ***
*** [172.17.0.5:53086] joined the chat ***
[172.17.0.5:53086]> Hello 😊
[172.17.0.4:47092]> Hi 😊
> Hello everyone!!! @frawn
> Hello everyone!!! @frown
> What are you doing guys? @wink
>
```

Figure 2: Client-server console application

4 Launch

Using source files

1. Run server:

```
$ python server.py
```

2. Run clients:

```
$ python client.py ns1.st13.os3.su 5555
or
$ python client.py 188.130.155.46 5555
```

where ns1.st13.os3.su - **domain**, 188.130.155.46 - **IP address**, and 5555 - **port number**

Using docker

1. Run server:

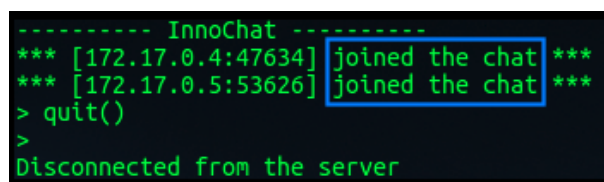
```
$ docker run -it --rm server
```

2. Run clients:

```
$ docker run --rm -it client
```

5 Implementation requirements

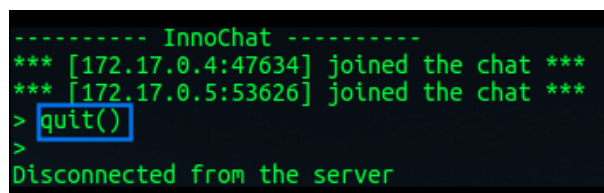
- Chat application is implemented using client-server architecture (see Figure 1)
- Each client receives messages from all other clients interactively (see Figure 2)
- Server handles two types of messages from all clients
 - Chat messages (see Figure 2)
 - Service messages:
 - * Connection Established (Figure 3)



```
----- InnoChat -----
*** [172.17.0.4:47634] joined the chat ***
*** [172.17.0.5:53626] joined the chat ***
> quit()
>
Disconnected from the server
```

Figure 3: Connection established

- * Command **quit()** and **CTRL-C** (Figure 4)



```
----- InnoChat -----
*** [172.17.0.4:47634] joined the chat ***
*** [172.17.0.5:53626] joined the chat ***
> quit()
>
Disconnected from the server
```

Figure 4: Quit the chat

- Server displays the list of connected clients (Figure 5)
- Server disconnects a particular client from the chat (Figure 6)
- Clients uses stickers (Figure 7)

```

Chat Server started on port: 5555
Client [172.17.0.3, 48506] connected
Client [172.17.0.4, 47634] connected
Client [172.17.0.5, 53626] connected
Client [172.17.0.3:48506] disconnected
Client [172.17.0.5:53626] disconnected

```

Figure 5: Connection information

```

Chat Server started on port: 5555
Client [172.17.0.3, 48506] connected
Client [172.17.0.4, 47634] connected
Client [172.17.0.5, 53626] connected
Client [172.17.0.3:48506] disconnected
Client [172.17.0.5:53626] disconnected
kill 47634
Client [172.17.0.4:47634] kicked

```

Figure 6: Disconnect a client

```

----- InnoChat -----
*** [172.17.0.4:48376] joined the chat ***
*** [172.17.0.5:54364] joined the chat ***
> Client [172.17.0.5:54364] is offline
[172.17.0.4:48376]> Hello 😊aaaa
> HHHug @hug
> @kiss
> @wink
> @frown
>

sergey@ns1: ~/Documents/chat/client$ vim Dockerfile
sergey@ns1:~/Documents/chat/client$ docker run --rm -it client
----- InnoChat -----
*** [172.17.0.5:54364] joined the chat ***
> Client [172.17.0.5:54364] is offline
> Hello @smile @monkey aaaa
[172.17.0.3:49414]> HHHug 😊
[172.17.0.3:49414]> 😊
[172.17.0.3:49414]> 😊
[172.17.0.3:49414]> 😊
>

```

Figure 7: Stickers