УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Никола Богдановић, PR114/2018

Михаило Богдановић, PR73/2018

P2P

Индустријски и комуникациони протоколи у електроенергетским системима

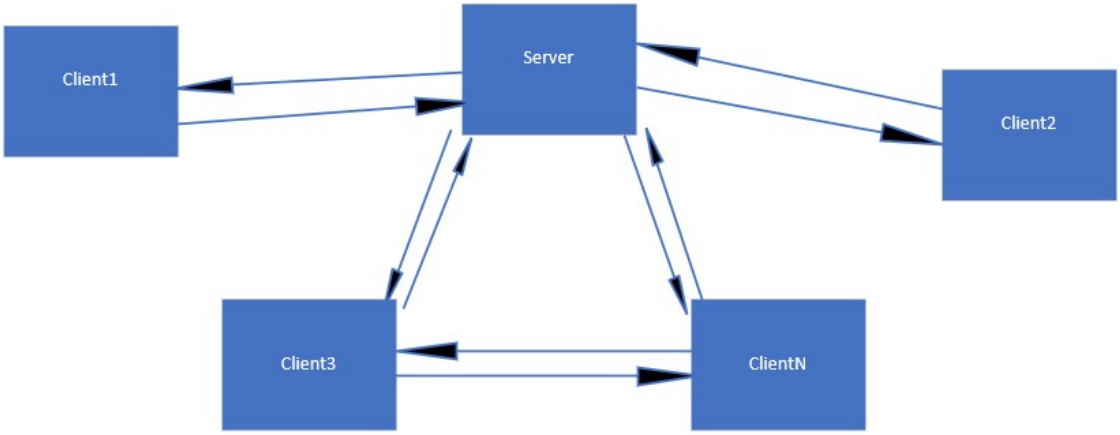
- Примењено софтверско инжењерство (OAC) –

Нови Сад 21.01.2022.

Садржај

1. Увод

У софтверској архитектури, *P2P* је образац за размену порука код којег пошиљаоци порука, названи *client*-и или *user*-и размењују поруке на два начина.
Први начин комуникације јесте преко самог сервера.
Client-и шаљу захтев серверу за комуникацију са другим *client*-ом и након успешне обраде остварују комуникацију са њим која се одвија преко сервера.
Други начин комуникације јесте директна комуникација са другим *client*-ом.
Client-и шаљу захтев серверу за податке о другом *client*-у са којим желе да комуницирају и након успешног добијања података остварују директну комуникацију са њим.



Слика 1. Архитектура *P2P* сервиса

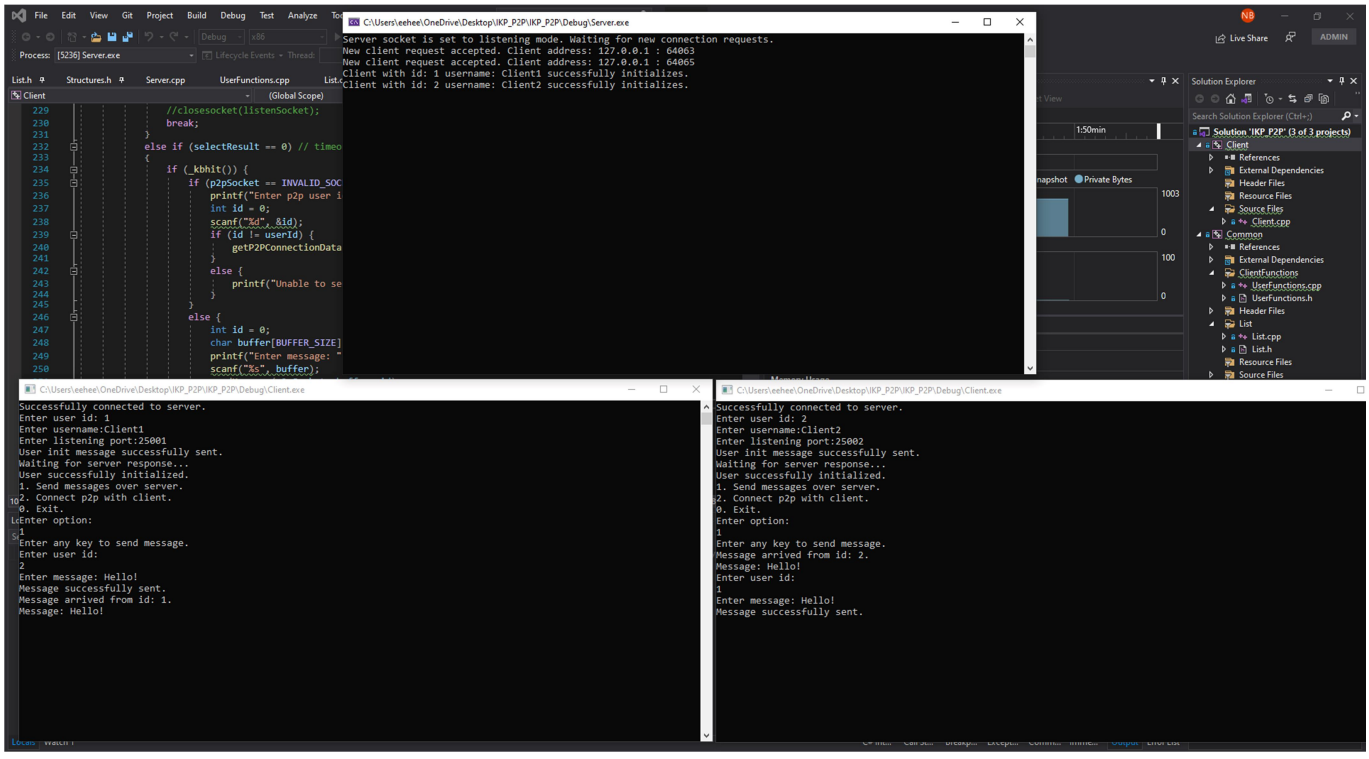
2. Дизајн

У апликацији су дефинисане две кључне компоненте:

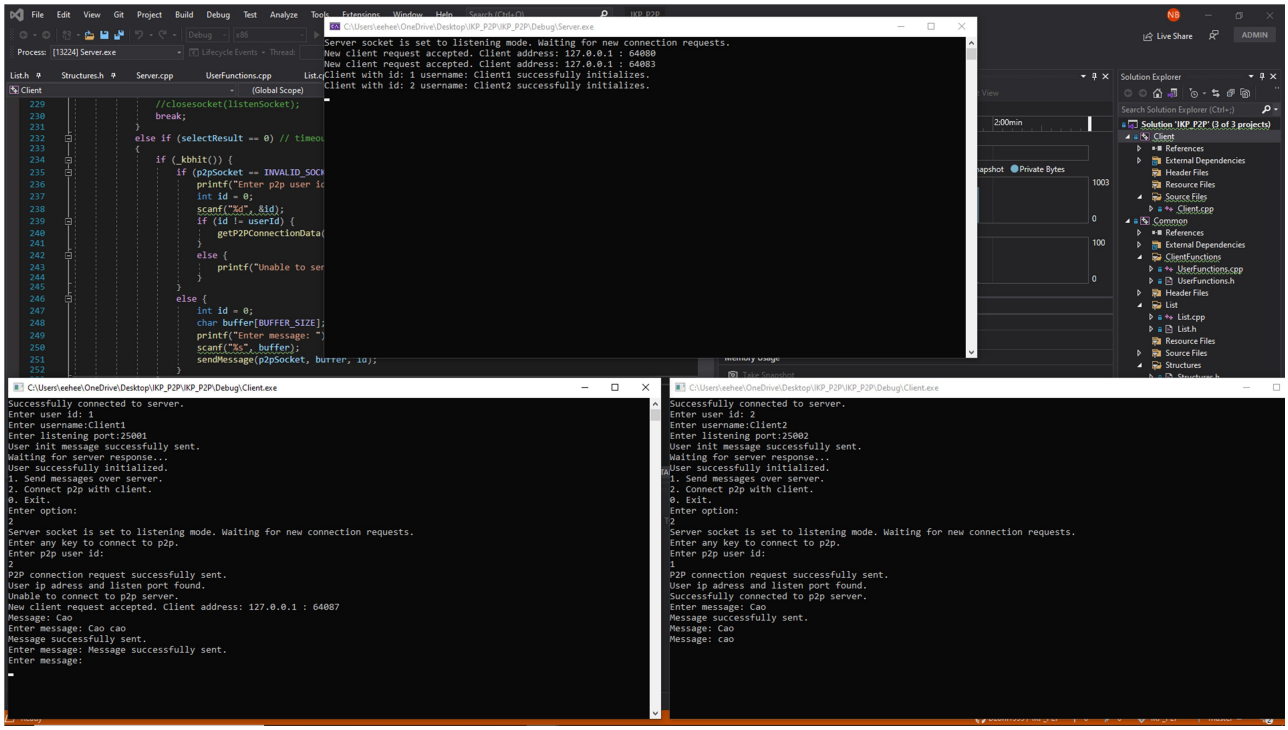
1) *Server*
2) *Client*

Client се повезује на познату адресу *Serva-a* помоћу *Socket-a*. Бира који вид комуникације жели да успостави са другим клијентом/клијентима. У обзир долазе директна комуникација између клијента или комуникација преко сервера. Након одабира, *Client* генерише жељени садржај и прослеђује.

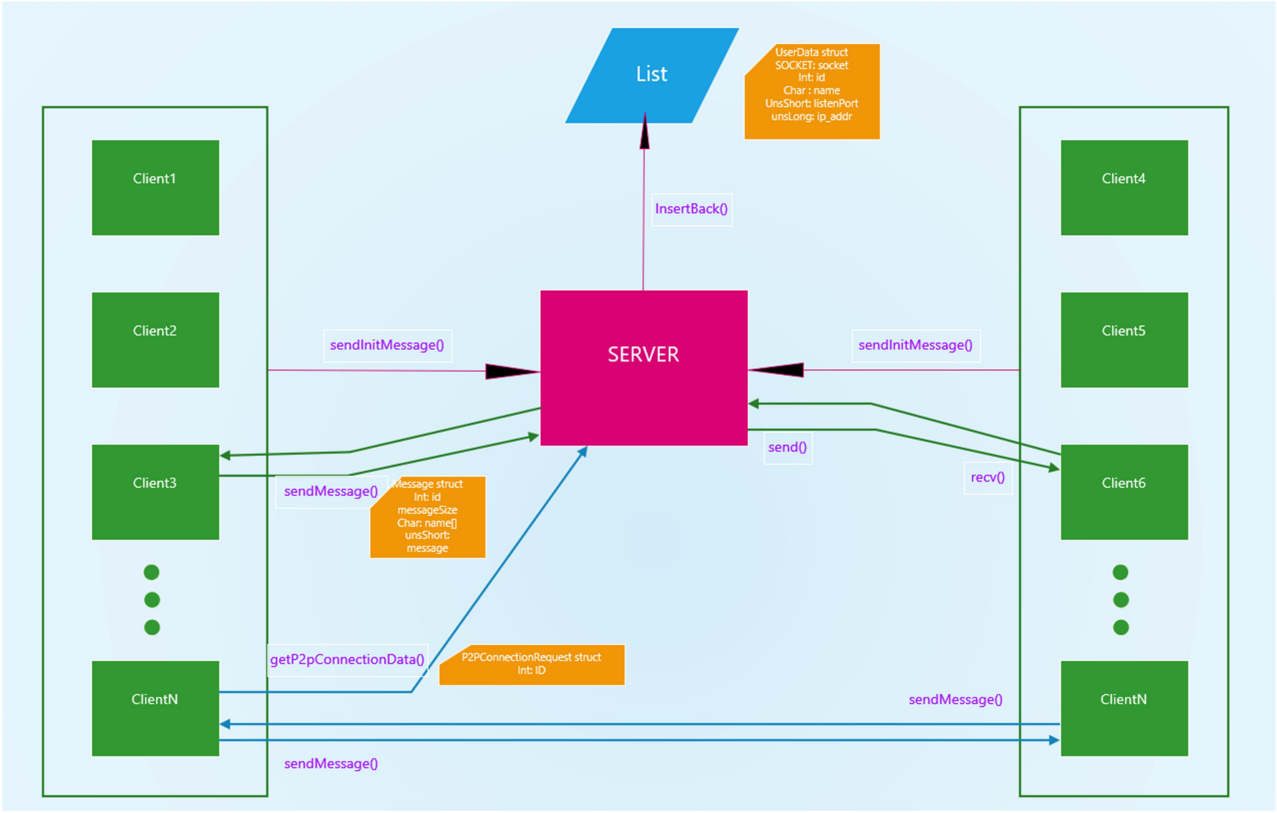
Server је у могућности да прихвати већи број клијената. У зависности од опција *Server* ће или проследити одговарајуће информације потребне клијенту да успостави директну везу са другим клијентом или ће приликом комуникације преко *Server-a* прослеђивати поруку одговарајућем клијенту.



Слика 2. *Client* и *Server* након што *Client* одабере комуникацију преко *Server-a*



Слика 3. *Client* и *Server* након што *Client* одабере директну комуникацију



Слика 4. Дизајн архитектуре систем

3. Структуре података

Подаци о клијентима/user који су регистровани чувају се у посебној структури *UserInit*(слика 5.).

Поруке које размењују клијенти, смештају се у структуру *Message*(слика 6.).

Структура *P2PConnectionRequest* користи се за чување -адреса клијената који комуницирају директно(слика 7.).

Структура *MessageType* чува тип поруке(слика 8.) коју клијент шаље серверу приликом одабира начина комуникације(слика 11.).

Структура *UserInitResponse* чува информацију да ли је клијент успешно регистрован(слика 9.).

Структура *P2PConnectionResponse* чува информације клијената са којима се остварује директна комуникација(слика 10.).

List(слика 12.) је структура података у којој су елементи повезани помоћу показивача. Чвор представља елемент на повезаној листи који има неке податке и показивач који показује на следећи чвор у листи. Листа је динамичка структура података, тако да може расти и смањивати се током извршавања додељивањем и уклањањем меморије (стога нема потребе за предефинисањем величине листе). Величина листе се може повећавати или смањивати током рада апликације, тако да нема губитка меморије.

```
typedef struct user_init {
        int id;
        char name[25];
        unsigned short listen_port;
}UserInit;
```

Слика 5. *UserInit* структура.

```
typedef struct message {
        int messageSize;
        int id;
        char message[512];
}Message;
```

Слика 6. *Message* структура.

```
typedef struct p2p_connection_request {
        int id;
}P2PConnectionRequest;
```

Слика 7. *P2PConnectionRequest* структура.

```
typedef struct message_type {
        int message_type;
}MessageType;
```

Слика 8. *MessageType* структура.

```
typedef struct user_init_response {
        int result;
}UserInitResponse;
```

Слика 9. *UserInitResponse* структура.

```
typedef struct p2p_connection_response {
        unsigned short listen_port;
        int ip_address;
}P2PConnectionResponse;
```

Слика 10. *P2PConnectionResponse* структура.

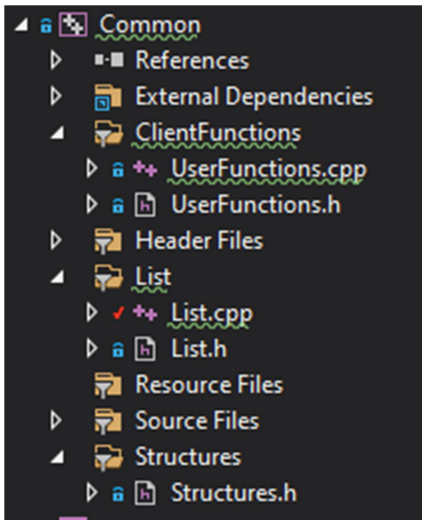
```
typedef enum enum_message_type {
        INIT_MESSAGE = 5,
        SEND_MESSAGE = 1,
        P2P_CONNECTION_REQUEST = 2,
        P2P_CONNECTION_RESPONSE = 3,
        INIT_MESSAGE_RESPONSE = 4
}EMessageType;
```

Слика 11. Типови порука.

```
typedef struct listitem {
        user_data data;
        struct listitem* next;
}ListItem;

typedef struct list {
        listitem* head;
}List;
```

Слика 12. Структура *List.*



Слика 13. *Common* пројекат са структурама.

4. Резултати тестирања

/

5. Закључак

Апликација се у целини понаша спрам дизајнираног сценарија рада, способна је да опслужује истовремено већи број корисника.

6. Потенцијална унапређења

Једно од могућих унапређења је да приликом директне комуникације између клијената, уместо да се клијенти проналазе преко свог *ID-a*, могли смо увести опецију проналажења или преко *ID-a* или *UserName-a*.