Dawid Królak, 145383, grupa I2.2, semestr III

Protokół komunikacyjny publish-subscribe

Spis treści:

1. Informacje ogólne:

- Protokól opiera się na jednej kolejce komunikatów o numerze 0x420.
- Maksymalna liczba zalogowanych użytkowników wynosi 16
- Maksymalna liczba tematów (kanałów komunikacyjnych) wynosi 16
- Maksymalna łączna liczba subskrybcji wynosi 64
- Maksymalna łączna liczba zablokowanych użytkowników wynosi 64

2. Struktura komunikatu.

```
struct message
{
     long receiver;
     int sender;

     int type;

     char shortText[16];
     char longText[256];
};
```

- **receiver** - oznacza dla kogo przeznaczona jest wiadomość. Dla serwera zawsze przyjmuje wartość 1, dla klientów jest zależna od ich PID. Przekierowując wiadomości innych klientów przyjmowana jest wartość
- **sender** - dla serwera zawsze przyjmuje wartość 1, dla klientów jest równe PID.
- **type** - oznacza typ wiadomości wysyłanych w sygnałach wewnętrznych
- **shortText** - krótki ciąg znaków. Zawiera nick podczas logowania, numer kanału komunikacyjnego, lub nick blokowanego użytkownika.
- **longText** - długi ciąg znaków. Zawiera treści wiadomości wysyłanych przez klientów. Jest również wykorzystywany do określenia typu subskrybcji - trwałej lub tymczasowej.

3. Struktura subskrypcji.

```
Struct Subscription
{
     int id;
     int channel;
     int quantity;
};
```

- **id** - PID użytkownika subskrybującego
- **channel** - numer subskrybowanego kanału
- **quantity** - trwałość subskrypcji. Przyjmuje dowolną wartość, dla wartości większych od 0 do użytkownika zostanie dostarczonych określona liczba wiadomości. Dla mniejszych subskrypcja jest trwała

4. Struktura klienta.

```
Struct Client
{
     int id;
     char name[16];
};
```

- **id** - PID klienta
- **name** - nick klienta, maksymalnie 16 dowolnych znaków

5. Struktura blokady.

```
Struct Blockade
{
     int blocker;
     int blocked;
};
```

- **blocker** - PID klienta blokującego
- **blocked** - PID klienta blokowanego

6. Sygnały wewnętrzne.

Protokół określa pewne sygnały o ustalonym znaczeniu. Służą one do komunikacji między serwerem a klientem.

Sygnały wysyłane przez klienta:

- **CONNECTION** - próba połączenia się z serwerem, przekazanie PID
- **LOGIN** - próba logowania, przekazanie nazwy użytkownika
- **LOGOUT** - próba wylogowania
- **SUBSCRIBE** - próba zasubskrybowania określonego kanału komunikacyjnego w określony sposób (trwale lub na k wiadomości)
- **UNSUBSCRIBE** - próba odsubskrybowania określonego kanału komunikacyjnego
- **MESSAGE** - próba wysłania wiadomości na określonym kanale komunikacyjnym
- **MSGREG** - zarejestrowanie nowego kanału komunikacyjnego
- **BLOCK** - zablokowanie odbierania wiadomości od określonego użytkownika, bez względu na kanał, którym ten się posłuży

Sygnały wysyłane przez serwer:

- **OPERATION_SUCCESS** - powodzenie dowolnej operacji wymagającej potwierdzenia
- **INVALID_USERNAME** - nick nie spełnia ograniczeń wynikających z poprawności danych
- **SENDER_NOT_RECOGNIZED** - użytkownik nie jest zalogowany
- **SERVER_OVERFLOW** - serwer osiągnął maksymalną liczbę zalogowanych klientów
- **UNAVAILABLE** - sygnał wysyłany w przypadku prób ponownego zasubskrybowania/odsubskrybowania tego samego kanału komunikacyjnego, zarejestrowania już istniejącego kanału komunikacyjnego, zablokowania tego samego użytkownika lub próby zalogowania na nick istniejącego już użytkownika,

- **NEW_CHANNEL** - sygnał asynchroniczny wysyłany do wszystkich zalogowanych użytkowników, gdy powstanie nowy kanał komunikacyjny
- **NO_CHANNEL** - sygnał wysyłany gdy kanał, przez który użytkownik próbuje wysłać wiadomość, lub który próbuje zasubskrybować nie istnieje

7. Poprawność danych.

Nick może składać się z dowolnych znaków. Jego długość to od 3 do 16 znaków.

Długość wiadomości nie może przekroczyć 256 znaków.

8. Scenariusze komunikacji.

a) nawiązanie połączenia

Klient:

```
receiver = 1;
type = CONNECTION;
sender = PID; //getpid()
```

Serwer:

```
receiver = PID;
type = [kod_sygnalu] //OPERATION_SUCCESS- połączono
```

b) logowanie

Klient:

```
receiver = 1;
type = LOGIN;
sender = PID; //getpid()
shortText = nick
```

Serwer:

```
receiver = PID;
type = [kod_sygnalu] //OPERATION_SUCCESS- zalogowany, INVALID_USERNAME- zly nick, UNAVAILABLE- nick zajety, SERVER_OVERFLOW- brak miejsca
```

c) wylogowanie

Klient:

```
receiver = 1;
type = LOGOUT;
sender = PID; //getpid()
```

Serwer:

```
receiver = PID;
type = [kod_sygnalu] //OPERATION_SUCCESS- klient wylogowany, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany
```

d) subskrypcja

Klient:

```
receiver = 1;
type = SUBSCRIBE;
sender = PID; //getpid()
shortText = channel;
longText = quantity;
```

Serwer:

```
receiver = PID;
type = [kod_sygnalu] //OPERATION_SUCCESS- kanał zasubskrybowany, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany, UNAVAILABLE- kanał już zasubskrybowany, NO_CHANNEL- kanał nie istnieje
```

e) odsubskrybowanie

Klient:

```
receiver = 1;
type = UNSUBSCRIBE;
sender = PID; //getpid()
```

```
shortText = channel;
```

Serwer:

```
receiver = PID;
type = [kod_sygnalu] //OPERATION_SUCCESS- kanał odsubskrybowany, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany, NO_CHANNEL- kanał nie istnieje, UNAVAILABLE- kanał nie był zasubskrybowany
```

f) rejestrowanie nowego typu wiadomości

Klient:

```
receiver = 1;
type = MSGREG;
sender = PID; //getpid()
shortText = channel;
```

Serwer:

```
receiver = PID;
type = [kod_sygnalu] //OPERATION_SUCCESS- kanał zarejestrowany, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany, UNAVAILABLE- typ wiadomości jest już zarejestrowany
```

g) nadawanie wiadomości

Klient:

```
receiver = 1;
type = MESSAGE;
sender = PID; //getpid()
shortText = channel;
longText = "[channel] [username]: text"
```

Serwer:

```
receiver = PID;
```

type = [kod_sygnalu] //OPERATION_SUCCESS- wiadomość wysłana, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany, NO_CHANNEL- kanał nie istnieje

h) odczytywanie wiadomości

Serwer:

receiver = PID zasubskrybowanego klienta;

type = messageCount //messageCount to licznik kolejnych wiadomosci wysylanych na serwerze, sluzacy do ich identyfikacji

longText = [tresc wiadomosci wyslanej przez klienta]

i) blokowanie użytkownika

Klient:

receiver = 1;

type = BLOCK;

sender = PID; //getpid()

shortText = nick;

Serwer:

receiver = PID;

type = [kod_sygnalu] //OPERATION_SUCCESS- użytkownik zablokowany, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany, INVALID_USERNAME- użytkownik o tej nazwie nie jest zalogowany, UNAVAILABLE- użytkownik o tej nazwie jest już zablokowany

j) odblokowanie użytkownika

Klient:

receiver = 1;

type = UNBLOCK;

sender = PID; //getpid()

shortText = nick;

Serwer:

receiver = PID;

type = [kod_sygnalu] //OPERATION_SUCCESS- użytkownik odblokowany, SENDER_NOT_RECOGNIZED-nadawca nieznany/niezalogowany, INVALID_USERNAME- użytkownik o tej nazwie nie jest zalogowany, UNAVAILABLE- użytkownik o tej nazwie nie jest zablokowany

k) zamknięcie klienta

Klient:

receiver = 1;

type = QUIT;

sender = PID; //getpid()