Zadanie 4 28 marca 2022 r.

zaawansowane technologie Javy

aplikacje klient-serwer w sieci

Instytut Informatyki
Uniwersytetu Wrocławskiego Paweł Rzechonek

Wybierz jedno spośród dwóch poniższych zadań i je zaprogramuj.

Zadanie 1

TCP jest protokołem działającym w trybie klient-serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie. Klient inicjalizuje połączenie do serwera. W przeciwieństwie do UDP, protokół TCP gwarantuje wyższym warstwom komunikacyjnym dostarczenie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów. Zapewnia to wiarygodne połączenie kosztem większego narzutu w postaci nagłówka i większej liczby przesyłanych pakietów.

Stwórz aplikację sieciową pracującą w modelu klient-serwer, która będzie umożliwiała publiczne wyrażanie swoich myśli a nawet prowadzenie rozmów przy dobrej woli uczestników spotkania. Aplikacja ma przesyłać dane po protokole TCP/IP wykorzystując do komunikacji obiekty `Socket` po stronie klienta i `ServerSocket` po stronie serwera.

Najpierw napisz program serwera jako aplikację okienkową w technologii *Swing* o nazwie `Komunikator`. Serwer ma nasłuchiwać na porcie 2022. Po nawiązaniu połączenia z nowym klientem powinien być utworzony osobny wątek do pracy serwera z tym klientem. Jednocześnie może być zalogowanych co najwyżej 10 klientów. Komunikaty wysyłane przez dowolnego klienta mają trafiać do wszystkich pozostałych klientów oraz do serwera (administrator serwera ma mieć możliwość śledzenia całej rozmowy). Na serwerze powinni być uwidocznieni wszyscy zalogowani klienci, tak aby w razie potrzeby administrator serwera mógł wyrzucić wybranego klienta z komunikatora. Liczba wszystkich pamiętanych przez klienta i przez serwer komunikatów nie powinna przekraczać 100. Przetestuj działanie serwera za pomocą programu *telnet*.

Następnie napisz program klienta także jako aplikację okienkową w technologii *Swing* o nazwie `Rozmowca`. Klient ma nawiązywać połączenie z serwerem podając swoją niepustą nazwę (imię, przezwisko itp.). Połączenie powinno być nawiązane jeśli

- serwer jest włączony,
- podana nazwa jest różna od wszystkich innych nazw zalogowanych klientów,
- liczba klientów nie przekracza ustalonego limitu (10).

Klient może wysyłać komunikaty do wszystkich pozostałych klientów i komunikaty te powinny być podpisane jego nazwą.

Zadanie 2

UDP jest protokołem internetowym, stosowanym w warstwie transportowej modelu OSI. Nie gwarantuje on dostarczenia datagramu. Jest to protokół bezpołączeniowy, więc nie ma narzutu na nawiązywanie połączenia i śledzenie sesji (w przeciwieństwie do TCP). Nie ma też mechanizmów kontroli przepływu i retransmisji. Korzyścią płynącą z takiego uproszczenia budowy jest szybsza transmisja danych i brak dodatkowych zadań, którymi musi zajmować się host posługujący się tym protokołem. Z tych względów UDP jest często używany w takich zastosowaniach jak wideokonferencje, strumienie dźwięku w Internecie i gry sieciowe, gdzie dane muszą być przesyłane możliwie szybko, a poprawianiem ewentualnych błędów zajmuje się aplikacja. Kolejną cechą odróżniającą UDP od TCP jest multicasting, czyli możliwość transmisji do kilku adresów docelowych naraz.

Stwórz aplikację sieciową pracującą w modelu klient-serwer, która będzie rozpowszechniała w sieci wybrane utwory muzyczne (pliki w formacie `aiff`, au, wav itp.) albo dzieła graficzne (pliki w formacie `jpg`, png, `gif` itp.). Aplikacja ma przesyłać dane typu `DatagramPacket` po protokole UDP/IP wykorzystując do komunikacji obiekty `DatagramSocket` po stronie klienta i serwera.

Napisz parę aplikacji: serwer o nazwie `Rozglosnia` do rozpowszechniania utworów oraz klient o nazwie `Odbiorca` do konsumowania udostępnionych utworów. Zarówno serwer jak i klient powinny być aplikacjami napisanymi w technologii *Swing*.

Rolą serwera ma być rozsyłanie (co 60 sekund) losowo wybranych plików muzycznych albo graficznych z pewnej ustalonej puli. Klient z kolei ma być aplikacją, która będzie odtwarzała nadesłana muzykę albo wyświetlała nadesłaną grafikę (na samym początku jeszcze w trakcie prac projektowych należy zdecydować, czy będą to pliki audio czy pliki graficzne).

Pliki muzyczne albo graficzne mają być rozsyłane z wykorzystaniem multicastingu. Do pracy serwisu można wykorzystać lokalny host multicastowy 224.0.0.1. Pliki przesyłaj w małych fragmentach – klient powinien je poskładać w całość i dopiero wtedy odtworzyć.

Spróbuj zasymulować działanie serwera w trudnych warunkach: raz na jakiś czas serwer powinien zagubić pakiet i raz na jakiś czas przestawić kolejność wysyłanych pakietów. Klient powinien umieć odtworzyć plik muzyczny albo wyświetlić plik graficzny nawet w przypadku nieodebrania jakiegoś środkowego fragmentu danych.

Do ściągnięcia i odtworzenia pliku muzycznego wykorzystaj klasy z pakietu `javax.sound.sampled`. Przykład:

```
URL url = new URL("http://jakis.host/media/plik.wav");
// getAudioInputStream() also accepts a File or InputStream
AudioInputStream ais = AudioSystem.getAudioInputStream(url);
Clip clip = AudioSystem.getClip();
clip.open(ais);
clip.start();
```

Do ściągnięcia pliku graficznego wykorzystaj klasę `java.awt.image.BufferedImage`. Przykład:

```
URL url = new URL("http://jakis.host/media/plik.jpg");

// read() also accepts a File or InputStream

BufferedImage img = ImageIO.read(url);
```