Paweł Saniewski

s16996, WIs I.3 - 19c

Sobota, 29 grudnia 2018r.

Raport Projektu

1. Ogólny opis rozwiązania

Program składa się z trzech klas - Server odpowiedzialnej za obsługę klientów, Client umożliwiającej połączenie z serwerem oraz Main służącą do odpowiedniego uruchomienia serwera lub klienta. Program uruchamiany jest wywołaniem klasy Main przy użyciu konsoli i programu ‘java’. Poniżej przedstawiony jest format poprawnego uruchomienia programu:

```
java Main [argument1] [args…]
```

gdzie [argument1] jest liczbą portów UDP do otwarcia (bez portów stanowiących poprawną sekwencję), w przypadku uruchamiania serwera, lub adresem IPv4 serwera, gdy uruchamiamy proces klienta, a [args…] jest sekwencją portów UDP, każdy oddzielony pojedynczą spacją.

2. Szczegółowy opis rozwiązania

Klasa Server przechowuje adresy i numery portów klientów, którzy wysyłają pakiety na porty UDP serwera, oraz dla każdego portu UDP uruchamia nowy wątek korzystając z zagnieżdżonej klasy ServerThread.

Klasa ServerThread przechowuje podstawowe informacje o porcie UDP oraz numer ID portu, który jest niezbędny do weryfikacji kolejności przychodzących pakietów. Jeżeli klient wyśle pakiety o odpowiedniej treści w odpowiedniej kolejności, obiekt-wątek klasy ServerThread uruchamia nowy wątek, który otwiera port TCP, wysyła numer tego portu do klienta i obsługuje komunikację z klientem na podstawie połączenia TCP.

Klasa Client jest prostą klasą, która z utworzonego portu UDP wysyła na podany adres IP pakiety ponumerowane w kolejności, w jakiej zostały podane podczas wywołania. Jeżeli pakiet o odpowiednim numerze trafi do odpowiedniego portu UDP serwera, to serwer sprawdza, czy na wcześniejsze w kolejności porty trafił prawidłowy pakiet od nadawcy. Jeżeli tak, to w serwerze zapisywana jest informacja o dotarciu prawidłowego pakietu. Jeżeli nieodpowiedni pakiet trafi do portu stanowiącego element sekwencji lub pakiet trafi do portu nie będącego elementem sekwencji, to w serwerze usuwane są informacje o kliencie, co powoduje, że klient musi rozpocząć próby od nowa. Jeżeli klient nie wyśle pakietów w odpowiedniej sekwencji przez dziesięć sekund, następuje ‘time-out’ kończący pracę klienta po uprzednim wysłaniu pakietów ‘cleanup’. Pakiety te trafiają do serwera i powodują usunięcie informacji o próbach połączenia. W przypadku, gdy klient wyśle poprawie ponumerowane pakiety w odpowiedniej kolejności, to otrzymuje numer portu TCP, z którym następnie prowadzi prostą wymianę komunikatów.

3. Obserwacje, eksperymenty, wnioski i założenia

W trakcie pracy nad projektem spostrzegłem, że pakiety wysyłane przez klienta potrafią przychodzić szybciej, niż serwer jest w stanie wypisywać do standardowego wyjścia informacje, że owy pakiet dotarł do portu. Może również wystąpić sytuacja, że pakiety dotrą do portów w innej kolejności, niż zostały wysłane. Prawdopodobnie spowodowane jest to implementacją metody port knocking przez rozdzielenie portów UDP na wątki. Na potrzeby projektu opóźniłem wysyłanie pakietów o 100ms. Jest to bezpieczne, górne ograniczenie, które ma pozwolić na poprawne działanie programu na różnych (szybszych lub wolniejszych) komputerach i połączeniach.