PSI Zadanie 1

Z37 - Aleksandra Szymańska, Angelika Ostrowska, Jakub Bąba

15.11.2024

1. Treść zadania

Z 1 Komunikacja UDP

Napisz zestaw dwóch programów – klienta i serwera wysyłające datagramy UDP. Wykonaj ćwiczenie w kolejnych inkrementalnych wariantach (rozszerzając kod z poprzedniej wersji). Klient jak i serwer powinien być napisany zarówno w C jak i Pythonie (4 programy). Sprawdzić i przetestować działanie „między-platformowe", tj. klient w C z serwerem Python i vice versa.

Z 1.1
Klient wysyła, a serwer odbiera datagramy o stałym rozmiarze (rzędu kilkuset bajtów). Datagramy powinny posiadać ustaloną formę danych. Przykładowo: pierwsze dwa bajty datagramu mogą zawierać informację o jego długości, a kolejne bajty kolejne litery A-Z powtarzające się wymaganą liczbę razy (ale można przyjąć inne rozwiązanie). Odbiorca powinien weryfikować odebrany datagram i odsyłać odpowiedź o ustalonym formacie. Klient powinien wysyłać kolejne datagramy o przyrastającej wielkości np. 1, 100, 200, 1000, 2000… bajtów. Sprawdzić, jaki był maksymalny rozmiar wysłanego (przyjętego) datagramu. Ustalić z dokładnością do jednego bajta jak duży datagram jest obsługiwany. Wyjaśnić.

2. Opis rozwiązania problemu

Zapoznaliśmy się z treścią zadania, dostarczonymi materiałami i notatkami z wykładów. Na tej podstawie zaczęliśmy pisać kod w Pythonie realizujący komunikację UDP dla klienta i serwera.

Przetestowaliśmy możliwe rozmiary wiadomości i ustaliliśmy jaki jest ich górny limit.

Przetestowaliśmy komunikację między klientem i serwerem w Pythonie.

Wiedząc, że zarówno serwer jak i klient w języku Python działają prawidłowo, postanowiliśmy przygotowywać kody w C "po kawałku".

Najpierw przygotowaliśmy klienta w języku C, testując go na bieżąco z serwerem w Pythonie i rozwiązując problemy. Po otrzymaniu prawidłowego, stabilnego rozwiązania, przeszliśmy do przygotowania serwera w języku C, podobnie wysyłając dane za pomocą klienta w języku Python.

Gdy przetestowaliśmy także serwer, przeszlliśmy do testowania połączenia serwera oraz klienta, obu w języku C.

3. Napotkane problemy

3.1. Format wiadomości

Musieliśmy wymyślić jak przesłać datagram o wymaganym formacie.

Ustaliliśmy, że potrzebujemy 2 bajty określające rozmiar wiadomości i następnie wiadomość. Przykładowo, datagram „2AB” reprezentowany byłby bitowo jako

„00000000 00000010 01000001 01000010”

Rozmiar datagramu, a w związku z tym i wiadomości zwiększa z każdą iteracją. Zapisujemy go korzystając z funkcji to_bytes(2) – zapisującą rozmiar na 2 bajtach.

Następnie ustalamy ile bajtów datagramu pozostało na wiadomość. Tworzymy ją z powtarzających się cyklicznie wielkich liter alfabetu A-Z. Jedna litera w ASCII kodowana jest na 1 bajcie, więc liczba bajtów dostępnych na wiadomość jest równa liczbie liter w wiadomości.

3.2. Maksymalny rozmiar wiadomości

W kolejnych iteracjach rozmiar wiadomości miał się zwiększać. Założyliśmy, że początkowy rozmiar datagramu to 2 bajty i będzie się on zwiększał poprzez mnożenie przez 2.

W taki sposób otrzymujemy następujące rozmiary datagramów:

2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536,131072,262144,524288

Istnieje jednak limit wielkości wiadomości, po którego osiągnięciu wyświetlany był komunikat „Message to long”. Drogą prób i błędów doszliśmy do tego, że maksymalnym rozmiarem datagramu jest 655007.

Dlatego w naszym programie, aby lepiej to ukazać, najpierw podwajamy rozmiar datagramu, a potem, przy zbliżaniu się do maksymalnej wartości inkrementujemy go lekko aż dotrzemy do limitu.

Zdecydowaliśmy się na rozmiary:

2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,40000,48000,56000,60000,64000,65000,65200,65400,65500,65502,65504,65505,65506,65507

4. Konfiguracja testowa

Nasza sieć testowa typu bridge nazywa się „z37_network”

Adres IP podsieci to 172.21.37.0/24

W tej sieci serwer jest lokalny i ma adres 127.0.0.1 i łaczymy się z nim na porcie 8000.

5. Testy

Po napisaniu kodu sprawdzaliśmy czy działa on z zamierzonym efektem. Uruchamialiśmy odpowiednie serwery i klienty za pomocą run.sh i testowaliśmy komunikację.

Po napisaniu serwera i klienta w Pythonie, uruchomiliśmy serwer, a w kliencie przesyłaliśmy wiadomości o zwiększacym się rozmiarze. Doszliśmy do wniosku, że maksymalnym rozmiarem jest 65507. Po tym przygotowaliśmy w ramach klienta testowego tablicę wybranych przez nas rozmiarów, aby zademonstrować działanie.

Testując kolejne konfiguracje serwerów i klientów, założyliśmy że limit rozmiaru wiadomości jest taki sam, a więc każde kolejne zestawienie serwera i klienta testowaliśmy używając tej samej tablicy rozmiarów dla wiadomości, otrzymując ten sam wniosek.

6. Wnioski końcowe

Udało nam się osiągnąć komunikację między klientem a serwerem. Klient może wysyłać, a serwer odbierać wiadomości. Korzystaliśmy z gniazd w C i Pythonie.