Sprawozdanie – laboratorium 1 PSI

Przedmiot: Programowanie Sieciowe
Autorzy: Marek Dzięcioł, Jakub Mieczkowski, Tomasz Nejman-lider
Data sporządzenia: 19.11.2025
Wersja: 1.0

Treść zadania:

Z 1 Komunikacja UDP Napisz zestaw dwóch programów – klienta i serwera wysyłające datagramy UDP. Proszę napisać jedno zadanie w konfiguracji klient/serwer Python/C, a drugie w konfiguracji klient/serwer C/Python – do wyboru.

Z 1.1 Klient wysyła, a serwer odbiera datagramy oraz odsyła ustaloną odpowiedź. Klient powinien wysyłać kolejne datagramy o przyrastającej wielkości, tj. 2, 4, 8, 16, 32, itd. bajtów. Ustalić eksperymentalnie z dokładnością do jednego bajta jak duży datagram jest obsługiwany. Wyjaśnić. Zmierzyć czas pomiędzy wysłaniem wiadomości a odebraniem odpowiedzi po stronie klienta i zestawić wyniki na wykresie.

Opis rozwiązania problemu:

Utworzyliśmy dwa programy:
- klient udp w Pythonie
- serwer udp w C

Serwer nasłuchuje na danym porcie UDP (domyślnie 8888), używając funkcji recvfrom(). Po odebraniu datagramu, wysyła stałą odpowiedź z powrotem do klienta, używając adresu i portu nadawcy, uzyskanych z recvfrom().
Klient wysyła datagramy:
- Dla każdego rozmiaru generuje losową wiadomość o tej długości.
- Mierzy czas rozpoczęcia przed wywołaniem sendall()
- Czeka na odpowiedź, używając recvfrom()
- Mierzy czas zakończenia i oblicza różnicę
- Zebrane dane (rozmiar datagramu i czas odpowiedzi) umieszczane są na wykresie z modułu matplotlib.pyplot i zapisywane do pliku

Klient znajduje maksymalny rozmiar datagramu najpierw wysyłając datagramy rosnących rozmiarów: 2, 4, 8, 16, 32, itd. a gdy pierwszy raz maksymalny rozmiar zostanie przekroczony rozpoczyna szukanie największej możliwej wartości za pomocą algorytmu binary search.

Napotkane problemy

1. Po wysłaniu pierwszego datagramu przekraczającego maksymalny rozmiar w UDP otrzymywaliśmy błąd “[Errno 90] Message too long” nie tylko dla tego datagramu, ale też dla każdego następnego, nawet w mniejszym, odpowiednim rozmiarze.
   Rozwiązanie: nie czyściliśmy strumienia bajtowego BytesIO(), przez co wielkość datagramu wydawała się mniejsza, lecz tak naprawdę rosła wykładniczo.

2. Nie potrafiliśmy odzyskać z dockera pliku z zapisanym wykresem przedstawiającym zależność czasu odpowiedzi serwera od rozmiaru datagramu.
   Rozwiązanie: Stworzyliśmy dodatkowy volume na dockerze klienta, następnie skopiowaliśmy plik z dockera na maszynę lokalną za pomocą komendy:

docker container run --rm -v "z38_client_volume:/source" -v "$(pwd):/backup" -w /source debian tar czf /backup/result.tar.gz .

Opis konfiguracji

Serwer/środowisko: bigubu.ii.pw.edu.pl
Port domyślny: 8888
Sieć: z38_network
Adres sieci: 172.21.38.0/24, fd00:1032:ac21:38::/64
Nazwy kontenerów:
- Serwer: z38_server
- Klient: z38_client

Wolumen: z38_client_volume (podmontowany w kontenerze klienta)

Opis testowania i wyniki testów

Klient najpierw wysyła datagramy o rosnących rozmiarach:

```
Testing doubling size: 2 bytes
Interation: 1
Sending buffer size=  2
Received OK
Testing doubling size: 4 bytes
Interation: 2
Sending buffer size=  4
Received OK
Testing doubling size: 8 bytes
Interation: 3
Sending buffer size=  8
Received OK
…
```

Następnie rozpoczyna poszukiwanie maksymalnego rozmiaru datagramu za pomocą binary search:

```
Sending buffer size=  65536
[Errno 90] Message too long
Sending buffer size=  49152
Received OK
Sending buffer size=  57344
Received OK
Sending buffer size=  61440
Received OK
…
Sending buffer size=  65507
Received OK
Sending buffer size=  65508
[Errno 90] Message too long
Sending buffer size=  65507
Received OK
Max datagram size is: 65507
```

Serwer nasłuchuje na domyślnym porcie i odpowiada klientowi:

```
UDP server listening on port 8888...
./udp_server: -->GN
datagram size is: 2Read data successful. Sending: "OK"
./udp_server: -->GGGG
datagram size is: 4Read data successful. Sending: "OK"
./udp_server: -->ERGRGRNR
datagram size is: 8Read data successful. Sending: "OK"
./udp_server: -->INNIEEGGRGIGNNGG
datagram size is: 16Read data successful. Sending: "OK"
./udp_server: -->IEGIGGRNGEERGEGIEGRGGIGERGRGENIG
datagram size is: 32Read data successful. Sending: "OK"
./udp_server:
-->IGGIEGGNRINGGRRGNGIERNEGGGGRGIIGGNREIGRRGRNRRENRNEGNINI
GNRGGENGG
datagram size is: 64Read data successful. Sending: "OK"
```

Wnioski końcowe

Udało się nam uruchomić dwa komunikujące się programy. Udało nam się znaleźć eksperymentalnie maksymalny payload protokołu UDP. Wykorzystano nieliczne elementy przykładów z portalu studia3, np. użycie BytesIO() jako preferowanego strumienia bajtów do tworzenia zawartości datagramów. Plik png zawierający wykres zależności długości odpowiedzi serwera zależnie od rozmiaru datagramu znajduje się w repozytorium.