PSI Sprawozdanie v1.0

Sofiya Yedzeika
Matvii Ivashchenko
Katsyaryna Anikevich

18 Listopada 2025

Treść zadania

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

Rozwiązanie składa się z dwóch programów realizujących komunikację typu klient–serwer z użyciem protokołu UDP. W przedstawionej konfiguracji klient został zaimplementowany w języku Python, natomiast serwer w języku C. Zadaniem klienta jest wysyłanie datagramów o rosnących rozmiarach, mierzenie czasu RTT (Round Trip Time) oraz eksperymentalne określenie maksymalnego rozmiaru wiadomości, który serwer jest w stanie poprawnie obsłużyć.

Klient

Klient generuje kolejne rozmiary pakietów zgodnie z zadaniem(tj. 2, 4, 8, 16, 32 itd. bajtów.). Po przekroczeniu granicy lub możliwości bufora odbiorczego serwera pojawi się brak odpowiedzi. Od tej wartości klient wyszukiwaniem binarnym szuka wartość graniczną.

Serwer

Serwer w języku C wykonuje typowe operacje dla gniazd UDP:

- Utworzenie gniazda: socket(AF_INET, SOCK_DGRAM, 0)
- Dowiązanie do portu 8000 przy użyciu bind().
- Wejście w nieskończoną pętlę:
  - odbiór danych recvfrom() do statycznego bufora bajtów,
  - wypisanie liczby odebranych bajtów (diagnostyka),
  - odesłanie dokładnie tej samej treści sendto().

Serwer działa jak echo server, co umożliwia klientowi pomiar czasu RTT. Serwer nie stosuje segmentacji — jeśli wiadomość jest większa niż obsługuje sieć lub kernel, pakiet zostanie utracony. To właśnie wykorzystujemy do określenia maksymalnego poprawnie obsługiwanego rozmiaru datagramu.

Pomiar czasu RTT

Pomiar składa się z:

- Zanotowania czasu wysłania (perf_counter_ns()).
- Wysłania datagramu.
- Oczekiwania na odpowiedź z limitem czasu (timeout 5s).
- Porównania treści odpowiedzi z wysłanym pakietem.

- Wyznaczenia średniej RTT dla danego rozmiaru.

Jeśli:

- nastąpi timeout,
- odpowiedź nie zgadza się z wysłaną treścią,

• klient kończy i binarnie szuka wartość graniczną z precyzją do 1 bajtu
• zapisuje maksymalny rozmiar.

Generowanie wyników

Klient:

- wypisuje wyniki w konsoli,
- zapisuje wszystkie RTT do pliku CSV results.csv.

Drugi skrypt (plot_result.py) wczytuje CSV i generuje wykres:

- oś X — rozmiar datagramu (skala log2),
- oś Y — średni RTT,
- wykres zapisany do pliku rezults_plot.png.

Pozwala to łatwo ocenić wpływ wielkości pakietu na opóźnienie.

Napotkane Problemy

Podczas realizacji napotkano problem związany z bardzo małym poprawnie obsługiwanym pakietem:

```
size_B,avg_rtt_ms
2,FAIL

MAX poprawnie obsługiwany payload ≈ 0 B
Wyniki zapisane do rezults.csv
```

Problem polegał na tym ze klient słuchał pod nieprawidłowym portem. Po zmianie na właściwy, problem był usunięty.

Opis konfiguracji testowej:

Testy przeprowadzono w środowisku opartym o dwa kontenery Docker: serwer i klient, uruchamiane w dedykowanej sieci z33_network.

• Sieć Docker: z33_network (bridge)
• Kontenery:

- z33_server – aplikacja echo UDP
- z33_client – generator zapytań i miernik RTT

Adresacja i porty

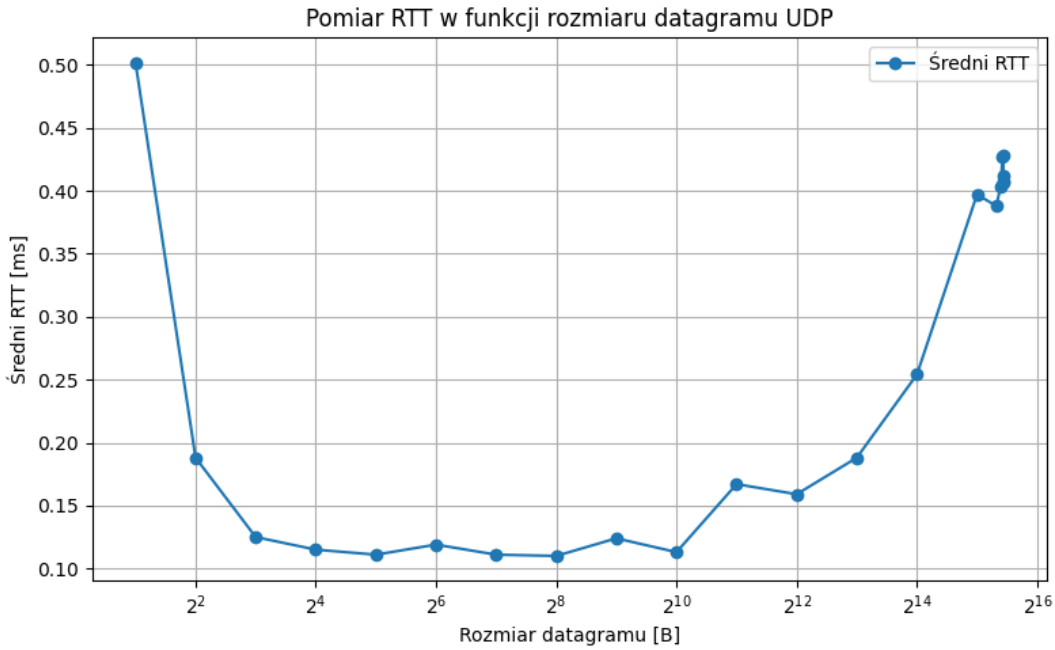
Serwer UDP:

- IP: 172.21.33.2
- Port: 8000

Klient wysyła pakiety bezpośrednio na powyższy adres i port (HOST="172.21.33.2", PORT=8000).

Opis wydruków:

Najpierw program drukuję komunikaty o zebraniu dokerów. Dalej wyniki pomiarów, które tez zapisywane do pliku results.csv:
size_B,avg_rtt_ms
2,0.170
4,0.079
.
.
.
44444,0.412
Pierwsza liczba to rozmiar pakietu w bajtach, druga czas RTT. Ostatni wydrukowany to ostatni udany test. Później dane zapisane w results.csv są wykorzystywane to tworzenia wykresu demonstracyjnego: Ostatni wydruk informuję użytkownika o końcowym wyniku testów.



Pierwszy test jest tak długi z powodu inicjalizacji sieci