Programowanie sieciowe

zadanie 2

1.12.2024

Bogumił Stoma

Aleksander Stanoch

Sebastian Abramowski

Treść zadania

Z 2 Komunikacja TCP

Napisz zestaw dwóch programów – klienta i serwera komunikujących się poprzez TCP. Transmitowany strumień danych powinien być stosunkowo duży, nie mniej niż 100 kB.

Klient TCP wysyła złożoną strukturę danych. Przykładowo: tworzymy w pamięci listę jednokierunkową lub drzewo binarne struktur zawierających (oprócz danych organizacyjnych) pewne dane dodatkowe: np. liczbę całkowitą 16-o bitową, liczbę całkowitą 32-u bitową oraz napis zmiennej i ograniczonej długości. Serwer napisany w Pythonie/C powinien te dane odebrać, dokonać poprawnego „odpakowania" tej struktury i wydrukować jej pola (być może w skróconej postaci, aby uniknąć nadmiaru wyświetlanych danych). Klient oraz serwer powinny być napisane w różnych językach.

Wskazówka: można wykorzystać moduły Python-a: struct i io.

Rozwiązanie zadania

Zadanie zostało zrealizowane poprzez implementację następujących elementów:

Klient (C)

1. **Tworzenie struktury danych**

   Klient generował listę jednokierunkową zawierającą 1000 węzłów. Każdy węzeł miał dwie właściwości tekstowe: `text1` i `text2`. Przykładowe wartości:

   - `text1`: "Node 0 Text1"
   - `text2`: "Node 0 Text2 - but longer".

2. **Serializacja**

   Dane z każdego węzła były serializowane do bufora binarnego. W tym celu:

   - Dodano rozmiar napisu (32-u bitowa liczba całkowita).
   - Skopiowano zawartość napisu.

3. **Transmisja**

   Klient wysyłał zserializowany bufor binarny do serwera przy użyciu gniazda TCP.

Serwer (Python)

1. **Nasłuchiwanie połączeń**

   Serwer nasłuchiwał na określonym porcie (`12345`). Po zaakceptowaniu połączenia od klienta, odbierał cały strumień binarnych danych.

2. **Deserializacja**

   Dane były odczytywane i odpakowywane przy użyciu modułu `struct`. Serwer:

   - Odczytywał rozmiar napisu.
   - Pobierał ciąg znaków odpowiadający temu rozmiarowi.
   - Powtarzał powyższe kroki dla każdego węzła.

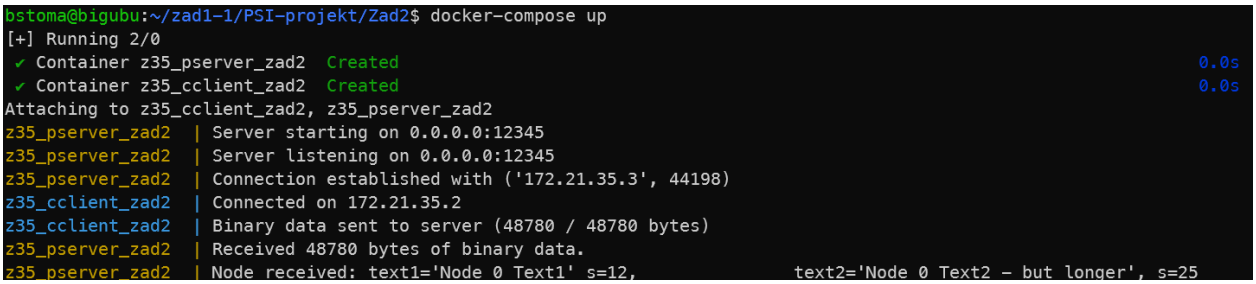
3. **Wyświetlanie danych**

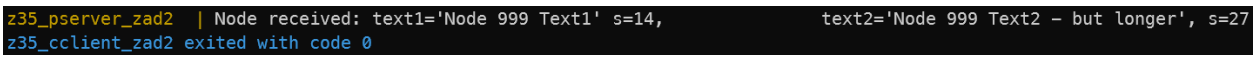
Zawartość pól `text1` i `text2` dla każdego węzła była wypisywana w skróconej formie w konsoli.

Opis konfiguracji testowej

- **Adres IP serwera**: `172.21.35.2` (może być przekazany jako argument w linii poleceń).
- **Port**: `12345` (domyślnie, ale można zmienić, podając go jako argument w linii poleceń).
- **Docker**: Środowisko testowe zostało uruchomione przy użyciu Docker Compose, co umożliwiło uruchomienie dwóch kontenerów w jednej sieci.

Screeny

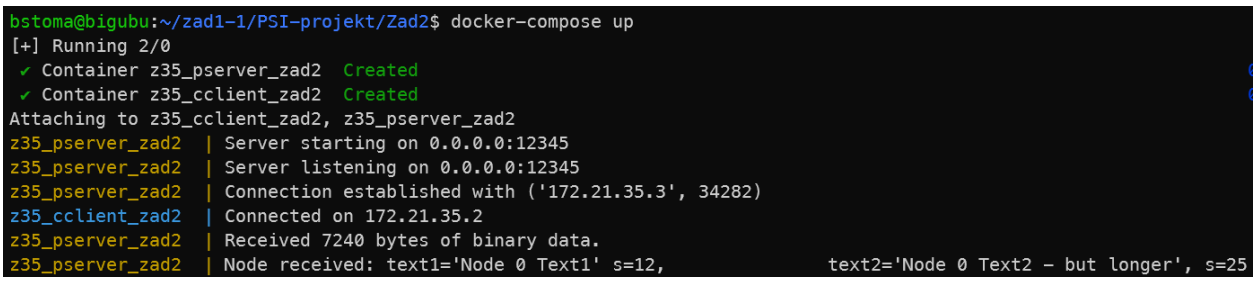


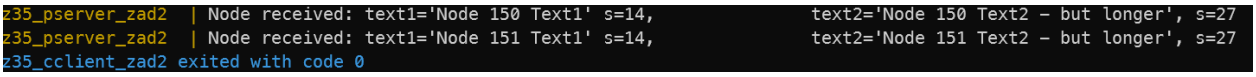


Serwer poprawnie nasłuchuje na określonym porcie, przyjmuje połączenia i odbiera przesyłane dane binarne od klienta. Klient skutecznie nawiązuje połączenie z serwerem, przesyła dane w sposób ciągły i zapewnia odpowiedni format przesyłanych pakietów, co widać po odebranych komunikatach. Logi potwierdzają, że serwer poprawnie przetwarza dane i zapisuje je w określonym formacie.

Problemy

Czasami kilent wysyłał jedynie do 151 węzłów czyli, 7240 bajtów, screeny poniżej, zgadujemy że jest to spowodowane jakimś ograniczeniem sieci laboratoryjnej, bo błąd nie pojawiał się przy uruchomieniu lokalnym.





Wnioski

Zadanie zostało zrealizowane zgodnie z założeniami, a komunikacja między klientem i serwerem przy użyciu TCP przebiegła pomyślnie. Klient w języku C generował i serializował listę jednokierunkową zawierającą dane tekstowe, a następnie przesyłał je w formacie binarnym do serwera. Serwer w Pythonie poprawnie odbierał dane, deserializował je i wyświetlał. Moduł struct pomógł w czytaniu danych wysyłanych przez klienta. Poprawność transmisji została potwierdzona w logach.