Z 2 Komunikacja TCP

Napisać zestaw programów – klienta i serwera – komunikujących się poprzez TCP. Wykonać ćwiczenie w kolejnych inkrementalnych wariantach, rozszerzając kod z poprzedniej wersji.

Z 2.1
Napisać w języku C/Python klienta TCP, który wysyła złożoną strukturę danych, np. uttworzoną w pamięci listę jednokierunkową lub drzewo binarne struktur zawierających (oprócz danych organizacyjnych) pewne dane dodatkowe, np. liczbę całkowitą 16-bitową, liczbę całkowitą 32-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 Pythona: struct i io.

Z 2.2

Zmodyfikować programy z zadania 2.1 tak, aby posługiwały się IPv6.

Z 2.3a

Zmodyfikować programy z zad. 2.1 w następujący sposób:

Klient powinien wysyłać do serwera strumień danych w pętli (danych powinno być przynajmniej kilkaset kB). Serwer powinien odbierać dane, ale między odczytami realizować sztuczne opóźnienie (np. za pomocą funkcji sleep()). W ten sposób symulowane będzie przeciążenie odbiorcy. Stos TCP będzie spowalniał nadawcę, aby uniknąć tracenia danych. Zidentyfikować objawy tego zjawiska po stronie klienta (dodając pomiar i logowanie czasu, monitorując ruch sieciowy np. za pomocą narzędzi tcpdump lub Wireshark) i krótko przedstawić swoje wnioski poparte uzyskanymi obserwacjami i statystykami czasowymi. Przeprowadzić eksperymenty z różnymi rozmiarami bufora nadawczego po stronie klienta (np. 100 B, 1 kB, 10 kB).
Zadanie należy wykonać korzystając z kodu klienta i serwera napisanych w języku C.

Z 2.3b

Zmodyfikować program klienta z 2.1 lub 2.2 tak, aby jednorazowo wysyłane były małe porcje danych (mniejsze od pojedynczej struktury) i wprowadzić dodatkowe sztuczne opóźnienie po stronie klienta (za pomocą funkcji sleep()). W programie serwera zorganizować kod tak, aby serwer kompletował dane i drukował je „na bieżąco".

Z 2.3c

Na bazie wersji 2.1 zmodyfikować serwer tak, aby miał konstrukcję współbieżną, tj. obsługiwał każdego klienta w osobnym procesie. Przy czym:

- Dla C należy posłużyć się funkcjami fork() oraz (obowiązkowo) wait().
- Dla Pythona posłużyć się wątkami, do wyboru: wariant podstawowy lub skorzystanie z ThreadPoolExecutor

Przetestować dla kilku równolegle działających klientów.