INSTRUKCJA do
ćwiczeń laboratoryjnych z PSI

Formuła: „Mini-projekt”, 3 osoby w zespole

Założenia: studenci rozwijają metodą inkrementalną program wg. podanych niżej ścieżek.

Punktacja: 20 p. Uwaga: nie ma progu minimalnego niezbędnego do zaliczenia laboratorium; niezgłoszenie zadania w terminie powoduje uzyskanie z niego 0 p. – bez możliwości poprawy.

Materiały dodatkowe:

- na serwerze studia2 w zasobach przedmiotu

Język i platforma:

C/C++ oraz Python – wg zaleceń w danym zadaniu. Inne języki (jak np. Java, C#) nie są dozwolone. Platformą do realizacji zadań jest serwer ***„bigubu”*** i środowisko Dockera (instrukcje w osobnych plikach).

Punktacja i kryteria oceny

Zadanie	Punktacja	Realizacja do ...
1. 1	5	20.11.2025
2.1a, 2.1b, 2.1c, 2.1d	8	27.11.2025
1.2	7	4.12.2025

Zespoły otrzymają indywidualnie informację o zestawie zadań do realizacji.

Sposób zaliczenia

- W terminie podanym w tabelce wyżej (odpowiednio dla kolejnych zadań 1.1, 1.2 i 2.1) należy przesłać link do repozytorium **GitHub**, w którym będą wszystkie wymagane pliki źródłowe oraz sprawozdanie (w formacie pdf) podsumowujące poszczególne zadania wraz z wynikami działania.
- Możliwy email zwrotny z uwagami do poprawy, zespół może (ale nie musi) być poproszony o konsultacje zdalne przez Teams. Na tym spotkaniu zespół musi mieć techniczną możliwość zdalnego zademonstrowania działającego kodu.
- Proszę w temacie maila napisać: „[PSI] [Zespół NN][Zadanie nr XX]”.

Uwaga:

- Dokumentacja musi być w formacie pdf.
- Proszę zadbać, aby repozytorium nie miało żadnych nadmiarowych plików (odpowiedni .gitignore).
- Proszę zadbać, aby poszczególne zadania dały się uruchomić minimalną liczbą komend, tj. klonuję repozytorium, wywołuję jedno polecenie i program działa (nie chcemy nadmiarowej konfiguracji itp.).

- Proszę umieścić plik README z krótkim opisem repozytorium (żeby ktoś z zewnątrz, np. ja, po przeczytaniu z grubsza wiedział co tam się dzieje i jak uruchomić program).
- **Tworzenie kodu za pomocą mechanizmów AI jest zakazane. Złamanie tego warunku powoduje automatyczne wyzerowanie punktów z laboratorium i może mieć dalsze konsekwencje formalne.**

Ocena wystawiana jest na podstawie:

- spełnienia wymagań,
- jakości kodu (patrz wskazówki na końcu w4.pdf) oraz zachowania ogólnej praktyki dobrego kodowania,
- jakości dokumentacji i raportu z testów.

Co powinno zawierać mini-sprawozdanie do zadań 1 i 2:

- Nagłówek – nazwę przedmiotu, **autorów** (skład zespołu z zaznaczonym nazwiskiem lidera zespołu), datę sporządzenia, wersję i historię zmian (jeśli będzie poprawka).
- **Treść** zadania.
- Opis rozwiązania problemu. Może to być fragment kodu z wstawkami informacyjnymi. Przykład - jeśli zadanie dotyczy wykorzystania nieblokującego we/wy, należy skupić się na opisie rozwiązania tej właśnie komunikacji.
- Uwagi dot. problemów, na które natrafił zespół i jak zostały one rozwiązane.
- Opis konfiguracji testowej (adresy IP, porty, interfejsy, etc.).
- Opis testowania i wydruki z testów (mogą być skrócone z istotnymi danymi).
- Wnioski końcowe i uwagi własne.
- NIE należy podawać informacji ogólnie znanych, np. opisywać działania funkcji gniazd i systemowych, działania protokołów L3 i L4, etc. Jeśli jednak natrafi się na nietypowe, nieoczekiwane, słabo udokumentowane, etc. działanie funkcji, to należy o tym napisać.
- Projekt dotyczy technik programowania protokołów internetowych, inne aspekty np. związane z interfejsem użytkownika nie będą brane pod uwagę przy jego ocenie, jednak wymaga się minimum zdrowego rozsądku przy obsłudze trybu command-line.
- Wolno korzystać z istniejącego kodu (np. realizującego logowanie), ale trzeba wyraźnie o tym napisać w dokumentacji. Kluczowy kod sieciowy powinien być stworzony własnoręcznie i od podstaw.
- Dokumentacja powinna być napisana poprawną polszczyzną, bez wyrażeń slangowych i niepotrzebnych anglicyzmów.

Zadania

Z 1 Komunikacja UDP

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.

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 eskperymentalnie 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.

Z 1.2 (to zadanie realizowane jest jako ostatnie)
Klient ma za zadanie odczytać plik z dysku (proszę wygenerować plik z losowymi 10000B) i wysłać do serwera jego zawartość w paczkach po 100B. Serwer ma zrekonstruować cały plik i obliczyć jego hash. Jako dowód działania proszę m.in. porównać hash obliczony przez serwer z hashem obliczonym przez klienta (może to być wydrukowane w konsoli klienta/serwera, hashe muszą być identyczne). Należy zaimplementować prosty protokół niezawodnej transmisji, uwzględniający możliwość gubienia datagramów. Gubione pakiety muszą być wykrywane i retransmitowane aby serwer mógł odtworzyć cały plik. Należy uruchomić program w środowisku symulującym błędy gubienia pakietów. (Informacja o tym, jak to zrobić znajduje się w skrypcie opisującym środowisko Dockera).

Z 2 Komunikacja TCP

Napisz zestaw dwóch programów – klienta i serwera komunikujących się poprzez TCP. Klient oraz serwer musi być napisany w konfiguracji C + Python (do wyboru co w czym).

Klient wysyła złożoną strukturę danych w postaci idealnego drzewa binarnego (co najmniej 15 węzłów). Każdy węzeł zawiera (oprócz danych organizacyjnych) liczbę całkowitą. Serwer powinien te dane odebrać (jeden węzeł drzewa na wiadomość) oraz dokonać poprawnej rekonstrukcji.
Wskazówka: można wykorzystać moduły Python-a: struct i io.