Ćwiczenia laboratoryjnie z PSI 2022Z
G. BLINOWSKI, wersja 2.kk1n
Prowadzący K. Kamiński (konrad.kaminski@orange.com)

Zasady

- Proszę kontaktować się używając mojego adresu w domenie orange.com. Konto pocztowe PW: konrad.kaminski3.dokt@pw.edu.pl sprawdzam zdecydowanie rzadziej.
- We wszystkich ćwiczeniach posługujemy się IPv4 w przygotowanym środowisku docker.
- Jest 5 zadań, dających razem 20 punktów (2+5+2+5+6).
- Zadania są przydzielone przez prowadzącego.
- Realizacja zdań Z1.1 oraz Z2.1 i Z2.2 jest wymagana do kolejnych zadań Z1.x i Z2.x.
- Określone zostały czasy realizacji zadań, każdy dzień opóźnienia sprawozdania proporcjonalnie obniża punktację, przekroczenie terminu o 1 tydz. powoduje wyzerowanie punktów z zadania.
- Czas realizacji zadań:
  Z1.1 + Z1.x (gdzie x>1) – sumarycznie 1 tydzień
  Z2.1 + Z2.1 – sumarycznie 1 tydzień
  Z2.x (gdzie x>2) – 2 tygodnie.
- Laboratorium nie jest obowiązkowe i nie ma minimum punktów, ale warto powalczyć.
- Sprawozdanie wraz ze spakowanym kodem proszę przesyłać mailem, proszę też nie usuwać kontenerów dockerowych.
- Konsultacje tylko zdalne (MS Teams: 00043333@pw.edu.pl), po wcześniejszym omówieniu mailem, preferuję środy 15:15-17:00.

Zadanie	Punkty	Realizacja	Czas [tyg]
1.1	2	wymagane	1
1.x (x>1)	5	alternatywnie	1 (razem z 1.1)
2.1	2	wymagane	1
2.2	5	wymagane	1 (razem z 2.1)
2.x (x>2)	6	alternatywnie	2

RAZEM: 20 punktów (2+5+2+5+6), czas realizacji: do 4 tygodni (1+1+2)

Z 1
Napisz zestaw dwóch programów – klienta i serwera wysyłające datagramy **UDP.** Wykonaj ćwiczenie w kolejnych inkrementalnych wariantach (rozszerzając kod z poprzedniej wersji).

Z 1.1
Klient wysyła, a serwer odbiera datagramy o stałym, niewielkim rozmiarze (rzędu kilkudziesięciu bajtów). Datagramy mogą zawierać ustalony „na sztywno” lub generowany napis – np. „abcde….”, „bcdef...”, itd. Powinno być wysyłanych kilka datagramów, po czym klient powinien kończyć pracę. Serwer raz uruchomiony pracuje aż do zabicia procesu.

Wykonać program w dwóch wariantach: C oraz Python.

Sprawdzić i przetestować działanie „między platformowe”, tj. klient w C z serwerem Python i vice versa.

Z 1.3
Na bazie wersji zadania 1.1 napisać w C klienta, który wysyła datagram zawierający strukturę zawierającą kilka liczb oraz napis: „struct { uint32_t a; int16_t b; char c[10];}”. Serwer napisany w

Pythonie powinien odebrać i dokonać poprawnego „odpakowania” tej struktury i wydrukowania jej pól. (Można też napisać klienta w Pythonie a serwer w C)

Wskazówka: wykorzystać moduły Python-a: struct i io.
Do zastanowienia, dlaczego warto stosować w programowaniu sieciowym typy określone w standardzie C99 (np. int32_t) zamiast typy proste (int)?