WZR-lab Porównanie architektur sieciowych - instrukcja do zadania

Celem zadania jest porównanie architektur klient-serwer oraz rozproszonej (multicast) pod kątem różnych aspektów budowy systemów zarządzających wirtualnymi zespołami roboczymi, takimi jak bezpieczeństwo, niezawodność, spójność środowiska, złożoność obliczeniowa, itd.

Zadanie podstawowe:

Aplikację klienta z przykładowego programu działającego w architekturze rozproszonej należy zmodyfikować tak, by przesyłanie wszystkich informacji odbywało się z wykorzystaniem architektury klient-serwer lub na odwrót w zależności od zestawu. Dodatkowo należy napisać aplikacje prostego serwera odbierającego i przesyłającego informacje z i do klientów, rejestrującego i usuwającego klientów.

Uwagi:

- W ramach serwera należy odpowiednio zmodyfikować program bazowy lub stworzyć oddzielny program (nowy projekt typu **Console Application**)
  Program powinien zawierać plik **main.cpp** (ciało serwera) oraz moduł net obsługujący sieć. Należy dodać do opcji projektu (opcje kompilatora) bibliotekę **-libws2_32** (w przypadku środowiska DEV-cpp) lub **WSOCK32.LIB MPR.LIB** w przypadku środowiska Visual C++.

- Wysyłanie i odbieranie musi odbywać się na oddzielnych portach (klient nadaje na porcie, na którym serwer odbiera i na odwrót) . Numery portów można przydzielić podczas tworzenia do odbierania i wysyłania komunikatów:

```
uni_recv = new unicast_net(1001);
uni_send = new unicast_net(1002);
```

- Praca najprostszego serwera powinna polegać na rejestrowaniu klientów, przyjmowaniu komunikatów (funkcja **reciv**) oraz rozsyłaniu ich do wszystkich zarejestrowanych klientów (funkcja **send**). Serwer powinien również informować aplikacje o klientach nieaktywnych.
- Sprawdzenie poprawności całego systemu wymaga użycia 3 komputerów (jeden z aplikacją serwera, dwa z aplikacją klienta)
- Numer IP serwera można uzyskać wywołując program **ipconfig** z linii poleceń

Rozszerzenia (dla osób nieobecnych na zajęciach):

R1. System dwóch serwerów, z których jeden jest aktywny a drugi rezerwowy staje się serwerem aktywnym w przypadku awarii pierwszego. System taki powinien zapewniać ciągłość pracy bez dodatkowego, znaczącego obciążania sieci (zakładamy, że serwer rezerwowy komunikuje się tylko z serwerem aktywnym i nie prowadzi żadnych obliczeń).

R2. Każda aplikacja kliencka jest w stanie w każdej chwili dodatkowo przyjąć rolę serwera.

R3. Symulacja wszystkich obiektów ruchomych na serwerze.