Programowanie sieciowe

Laboratorium 4 **dr hab. inż. Radosław Wajman**

Wielożytkownikowy serwer ECHO (TCP)

Cel ćwiczenia:

W trakcie ćwiczenia studenci wykonają pierwszy prosty program sieciowy wielożytkownikowego serwera usługi ECHO.

Usługa Echo: serwer czeka na połączenie klienta. Po zaakceptowaniu połączenia oczekuje na wiadomość. Zaimplementowany program serwera można testować napisaną na poprzednich zajęciach własną aplikacją klienta bądź aplikacją dostępną w folderze *PROGRAMY* w sekcji „*Połączenie TCP klient-serwer*".

Polecenie ćwiczeniowe:

Do zaliczenia ćwiczenia wymagane jest napisanie programu udostępniającego kilku użytkownikom jednocześnie usługę ECHO. Możliwa dowolna platforma programistyczna.

Zadaniem programu jest:

- udostępnianie usługi dla wybranych interfejsów sieciowych (domyślnie dla wszystkich, a tylko nie localhost!),
- udostępnienie usługi ECHO dla protokołu TCP/IP na podanym porcie (wartość domyślna 7).
- udostępnienie możliwości wprowadzenia innego portu komunikacji,
- rejestrowanie wszystkich połączeń klientów, przypisywanie im numeru # i wyświetlanie w stałym miejscu (znajdującym się 80% od lewej krawędzi okna oraz 10% od górnej krawędzi) na bieżąco aktualizowanych informacji o podłączonych komputerach odległych – adres IP wraz z numerem portu komunikacji, np. #1 192.168.0.10:34567
  #2 192.168.0.11:76543
- posiadać limit aktywnych połączeń do 3, a po jego przekroczeniu natychmiast rozłączać nadliczbowe połączenie (po stronie klienta powinno być to interpretowane jako SERVER BUSY),
- odporność na niewłaściwe dane wpisane przez użytkownika, niewłaściwe zamknięcie gniazda po stronie klienta i zwalnianie zasobów (w tym wszystkich wątków) w chwili zamykania programu,
- natychmiastowe odsyłanie wiernej kopii otrzymanej wiadomości (tzn. funkcja wysyłająca wysyła TYLKO tyle danych, ile serwer otrzymał od klienta),
- informowanie użytkownika o każdej otrzymanej wiadomości z zaznaczeniem numeru # przypisanego nadawcy wiadomości oraz treści i rozmiaru wiadomości,
- informowanie o akceptacji nowego połączenia (w tym nadliczbowego) oraz rozłączenia klienta.

Algorytm programu powinien być zgodny z tym przedstawionym na diagramie na następnej stornie:

UWAGA!!

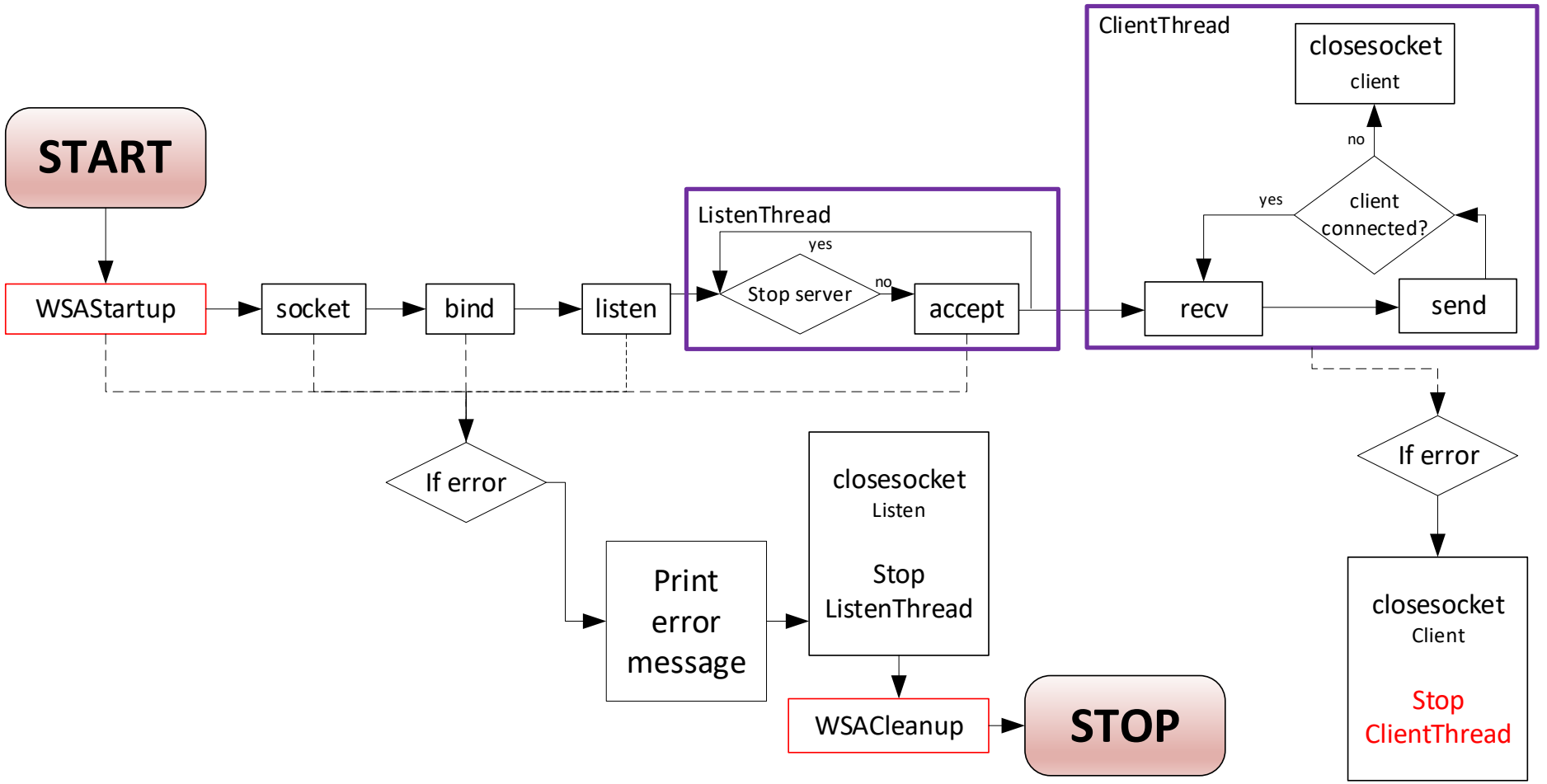
Program można zrealizować na dwa sposoby. Pierwszy, to implementacja poszczególnych bloków programu (np. oczekiwanie na połączenie klienta lub oczekiwanie na wiadomość od klienta) w oparciu o wątki. Inną możliwością jest wykorzystanie metod asynchronicznych do obsługi blokujących operacji komunikacji sieciowej.

Uwagi do nadsyłanych programów

1. Program powinien spełniać wszystkie funkcjonalności opisane powyżej i żadnej więcej,
2. Powinien być wynikiem samodzielnej pracy,
3. Dozwolone jest użycie różnych języków programowania przy założeniu, że każda z operacji zaznaczonych na diagramie zostanie zaimplementowana (wywołana / obsłużona) z osobna.
   Wyjątek w tym przypadku stanowią bloki WSAStartup i WSACleanup.
   Wykonanie zadania w oparciu o automatyzację wynikającą z wywołania jedynie metody konstruktora spowoduje obniżenie oceny o 1.
4. Zaimplementowana procedura obsługi błędów i sytuacji wyjątkowych (jej brak obniża ocenę o jeden punkt),
5. Interfejs użytkownika dowolny tj. konsola, GUI.
6. Czytelny i przejrzysty kod uwzględniający wysoki poziom hermetyzacji i synchronizację wątków.

Odwołania

1. C#: https://social.msdn.microsoft.com/Forums/en-US/d9e4bcbc-3149-4817-8d7c-37cdb29f8be0/tcp-server-with-multiple-clients?forum=ncl
2. JAVA: http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html



Bloki zaznaczone na czerwono są wymagane tylko w przypadku implementacji w języku C/C++.