

Klient usługi ECHO na TCP

Cel ćwiczenia:

Usługa Echo na bazie protokołu TCP: serwer czeka na połączenie klienta. Po zaakceptowaniu połączenia oczekuje na wiadomość. Każdą odebraną od klienta wiadomość serwer natychmiast odsyła do klienta bez zmian.

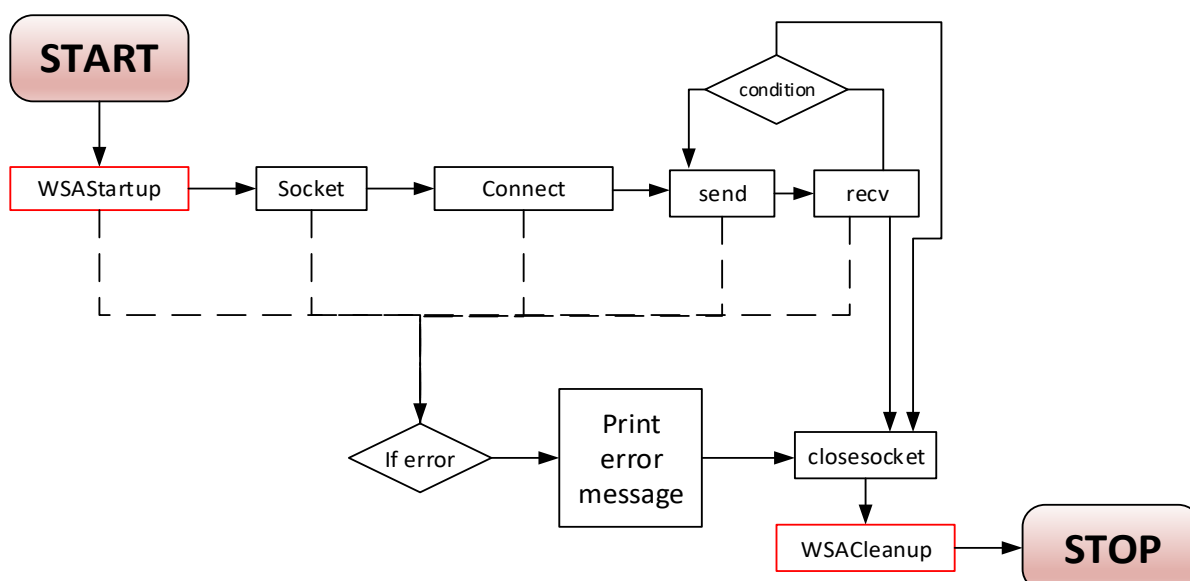
W trakcie tego ćwiczenia studenci wykonają pierwszy prosty program sieciowy klienta usługi ECHO, jak np. TCP_Client.jar (do pobrania w folderze PROGRAMY). Zaprojektowaną aplikację klienta można testować z serwerem TCP_Server.jar, który podobnie można pobrać z sekcji PROGRAMY.

Polecenie ćwiczeniowe:

Zadaniem programu jest połączyć się za pomocą protokołu TCP/IP z podanym serwerem na podanym porcie. Program powinien wysyłać znaki wpisywane przez użytkownika do zdalnego serwera oraz wyświetlać na ekranie wszystkie znaki nadsyłane przez serwer.

Ponadto:

- Serwer odległy powinien być identyfikowany nazwą domenową lub adresem IP (należy unikać łączenia z użyciem adresu 127.0.0.1),
- Klient powinien udostępniać możliwość wprowadzenia adresu IP i portu komunikacji do połączenia z serwerem, (na systemach z rodziny Linux – w tym MAC, zajęcie portu poniżej 1024 wymaga uprawnień root)
- Po ustanowieniu połączenia lub błędzie, aplikacja klienta powinna o tym informować użytkownika, dodatkowo status połączenia powinien być aktualizowany na bieżąco i wyświetlany ciągle w stałym miejscu znajdującym się 80% od lewej krawędzi okna oraz 10% od górnej krawędzi,
- Dane (wysyłane i odbierane) mają być wyświetlane w taki sposób, aby łatwo można było je porównać (tj. dodać informację o ilości bajtów wysłanych i odebranych),
- Program musi być odporny na niewłaściwe dane wpisane przez użytkownika,
- Używanie funkcji typu „readln” do odczytu danych z gniazda jest niewskazane; Funkcje te z natury czekają (blokują) na dane z gniazda do czasu nadejścia w wiadomości znaku nowej linii lub przepełnienia buforu, natomiast serwer sam z siebie nigdy nie dodaje znaku końca linii, chyba, że otrzymał go od klienta.
- Algorytm programu powinien być zgodny z tym przedstawionym na poniższym diagramie:



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

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 WSAShutdown 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.

Odwołania

1. C: <http://www.binarytides.com/winsock-socket-programming-tutorial/>
2. C#: [https://msdn.microsoft.com/en-us/library/4xzx2d41\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/4xzx2d41(v=vs.110).aspx)
3. JAVA: <https://systembash.com/a-simple-java-tcp-server-and-tcp-client/>
4. JS: <https://github.com/socketio/socket.io-client/blob/master/docs/API.md>
5. Python: <https://docs.python.org/3/howto/sockets.html>