Projekt zaliczeniowy - Sieci Komputerowe 2 laboratorium

Michał Boroń
mboron@cs.put.poznan.pl
www.cs.put.poznan.pl/mboron

Podstawowe zasady:

- Wszystkie projekty należy realizować w architekturze klient-serwer z użyciem protokołu TCP, chyba że uzgodniono inaczej z prowadzącym.
- Implementacje serwerów **współbieżnych** należy wykonać dla systemów operacyjnych GNU/Linux z użyciem języka C/C++ wykorzystując API BSD sockets.
- Implementację klienta można wykonać na dowolnej platformie z użyciem dowolnego języka programowania, po uzyskaniu zgody prowadzącego.
- Klient musi posiadać interfejs graficzny, chyba że uzgodniono inaczej.
- Stan musi być przechowywany na serwerze (np. stan rozgrywki).
- Kody projektów są utrzymywane na repozytorium git w systemie GitLab https://gitlab.cs.put.poznan.pl lub w innym repozytorium dostępnym dla prowadzącego. Korzystając z serwera gitlab dostępnego na uczelni należy dodać prowadzącego do grupy projektu w systemie GitLab z uprawnieniami "Developer".
- Elementem zaliczenia jest sprawozdanie (maksymalnie 2 strony A4) w formacie PDF, które zawiera: opis protokołu komunikacyjnego, opis implementacji (struktury projektu), opis sposobu kompilacji i uruchomienia projektu.

Projekty nie będą oceniane lub ocena zostanie obniżona, w przypadku:

- implementacji serwera, która nie spełnia przynajmniej jednego z wymagań: współbieżna, napisana w C/C++, używa API bsd-sockets, kompilowana w systemie Linux,
- braku sprawozdania,
- niemożliwości prezentacji działania projektu w laboratorium lub przy użyciu własnego sprzętu,
- rażącej niezgodności funkcjonalności z uzgodnionymi wymaganiami,
- rażącego zaniedbania czytelności kodu,
- uzasadnionych wątpliwości dotyczących samodzielności pracy,
- innych odstępstw od ustalonych zasad.

Kryteria oceny projektu zaliczeniowego

Kategoria	Komentarz
Aspekty związane z komunikacją przez sieć.	Bardzo ważne (główna składowa oceny).
Protokół komunikacyjny.	
Implementacja aspektów sieciowych w tym błędy funkcji sieciowych, fragmentacja, sklejanie, obsługa sigpipe przy write.	
Konfiguracja maksymalnej liczby połączonych klientów, rozłączenie nadmiarowych klientów.	
Brak wysyłania nadmiarowych danych przez sieć.	
Inne.	
(nieobowiązkowe) Obsługa komunikatów niezgodnych z protokołem.	
Zgodność funkcjonalności z wymaganiami.	Bardzo ważne (główna składowa oceny).
Specyficzne dla projektu.	
Przechowywanie stanu na serwerze.	
Brak wbitych na stałe adresów ip / localhost.	
Inne.	
Poprawność aspektów związanych ze współbieżnością.	Ważne.
Błędy w obsłudze współbieżności.	
Wycieki pamięci.	
Aktywne czekanie.	
Inne.	
Sprawozdanie (opis protokołu, struktury projektu, sposobu uruchomienia).	Zazwyczaj mały wpływ na ocenę. Przy czym brak sprawozdania lub bardzo słaba jakość obniżają ocenę o pół stopnia. **Brak sprawozdania może skutkować brakiem możliwości oddania projektu!**
Pozostałe	Zazwyczaj mały wpływ na ocenę.
Brak błędów i ostrzeżeń przy kompilacji (z flagą -Wall).	
Interfejs graficzny klienta nie “zawiesza się” podczas operacji sieciowych.	
Wybór tematu w terminie.	
Utrzymywanie projektu na repozytorium git (realne wykorzystanie repozytorium).	
Czytelność kodu, architektura, UX.	
Projekt oddany w pierwszym terminie.	

Przykładowe tematy projektów (można zaproponować własne)

W nawiasach kwadratowych przykładowe wymagania.

Jednoosobowe:

1. Sieciowa turowa gra logiczna, np.: reversi, szachy, warcaby

   Jeden serwer do którego podłączają się gracze. Wspiera wiele równoległych rozgrywek między parami graczy.

   [walidacja ruchów po stronie serwera, powiadomienie o wyniku rozgrywki, wykrycie i obsługa rozłączenia gracza]

2. Zdalne zamykanie systemów operacyjnych

   Jeden program-agent per węzeł z systemem operacyjnym do zamknięcia. Jeden serwer, który wie o wszystkich agentach oraz zna uprawnienia klientów.

   [uprawnienia per klient – które maszyny może wyłączać, możliwość wyłączenia wskazanych przez klienta maszyn, możliwosć dynamicznej rejestracji nowych agentów, monitorowanie stanu agentów – wł./wył.]

Dwuosobowe:

3. System komunikacji grupowej typu IRC [dołączenie do pokoju, tworzenie pokoju, wysyłanie wiadomości w pokoju, odbieranie wiadomości z pokoju, możliwość usunięcia użytkowników z pokoju przez jego właściciela]

4. Komunikator internetowy typu GG [tworzenie konta, logowanie z hasłem, lista znajomych, prowadzenie wielu konwersacji jednocześnie]

5. Komunikator internetowy typu Skype [tworzenie konta, logowanie z hasłem, lista znajomych, rozmowa głosowa lub wideo]

6. System wymiany komunikatów publish/subscribe [tworzenie nowych tematów, publikowanie wiadomości w określonym temacie, subskrybowanie tematu, odbieranie wiadomości w subskrybowanych tematach, anulowanie subskrypcji]

7. Grupowy edytor plików tekstowych [tworzenie nowego dokumentu, poprawna współbieżna edycja dokumentu, udostępnienie dokumentu innym użytkownikom, usuwanie dokumentu]

8. Prosty serwer HTTP zgodny z RFC 2616 co najmniej w zakresie żądań GET, HEAD, PUT, DELETE

9. Prosty serwer FTP zgodny z RFC 959 co najmniej w zakresie komend: ascii, binary, mkdir, rmdir, put, get

10. E-mail: system poczty elektronicznej [wysyłanie wiadomości do lokalnego serwera, przesyłanie wiadomości między serwerami, pobieranie wiadomości ze skrzynki, logowanie hasłem]

11. System mnożenia dużych macierzy kwadratowych (lub innych obliczeń algebry liniowej). [rozproszenie obliczeń na wiele maszyn, przesłanie zadania, odebranie wyniku]

12. System wymiany plików w architekturze peer-to-peer

13. Implementacja dowolnej usługi sieciowej zgodnie z RFC

W projektach wystarczy utrzymywać stan w pamięci, nie ma konieczności zapisywania danych na dysku (poza FTP, wymiana plików P2P).