Programowanie sieciowe

Końcowa Dokumentacja Projektu

Mirosław Baca, Marcel Gacoń

Gra sieciowa w „Statki” z poczekalnią i obserwatorami

1. Opis:

Projekt jest implementacją gry w „Statki” po sieci. Serwer zarządza jednocześnie wieloma grami, po dwóch graczy, gdzie do każdej gry jest możliwość dołączenia się jako obserwator. Klient po uruchomieniu i połączeniu z serwerem (multicast) ma możliwość grania, lub obserwowania gry. Gracze dołączają do poczekalni (Lobby), do których może dołączyć drugi gracz, oraz obserwatorzy. Połączenie między graczami, oraz obserwatorami jest realizowane poprzez serwer (unicast). Ruchy i stan gry zapisywany jest u klientów, a serwer wysyła informacje do wszystkich jaki jest aktualny stan gry. Po zakończeniu rozgrywki gracze oraz obserwatorzy wracają do lobby. Serwer działa w trybie demona.

2. Działanie programu (klient-serwer):

Wyszukiwanie serwera: Klienci wysyłają zapytanie w grupie multicastowej, serwer odpowiada, podając swoje dane (adres, port), co umożliwia lokalizacje serwera.

Połączenie z serwera: Po zlokalizowaniu serwera następuje zestawienie połączenia TCP.

Poczekalnia: Po połączeniu z serwerem użytkownik musi wpisać jak się chce nazywać (nickname). Serwer sprawdza, czy wprowadzony „nick” nie jest przypisany do innego klienta. Po wprowadzeniu prawidłowej nazwy klient ma następujące opcje:

- /create - tworzy nową grę z id, które serwer przypisuje automatycznie.
- /join <id> - klient dołącza do gry z konkretnym id, po tym jak dwóch klientów dołączy się do danej gry to każdy kolejny dołącza się jako obserwator. Można połączyć się z grą również po jej rozpoczęciu.
- /list - wyświetla listę gier
- /exit - wyłącza klienta, zwalniając tym samym „nick”

Gra w „Statki”: Po dołączeniu klienta (gracza) do gry, mamy dostęp do następujących komend oraz funkcjonalności:

- /place - Po wpisaniu tej komendy program będzie pytał o koordynaty kolejnych statków. Nie można wykonywać innych komend dopóki wszystkie statki nie zostaną postawione.
- /start - Potwierdza gotowość gracza do rozgrywki, ta komenda nie może być wykonana dopóki gracz nie postawi swoich statków.
- /fire <x y> - gracz wykonuje strzał na podany koordynat

Gra rozpocznie się tylko wtedy gdy obaj gracze wpiszą komendę /start. Następnie pierwszy gracz może wykonać komendę /fire. Gracze podczas rozgrywki mogą komunikować się między sobą (implementacja chat'u). Gra kończy się jak jeden z graczy zatopi wszystkie statki oponenta.

Obserwatorzy: Obserwator otrzymuje aktualizacje stanu gry od serwera i nie ma możliwości ingerencji w jej przebieg, oraz nie są w stanie pisać wiadomości do graczy.

Zakończenie rozgrywki: Po zakończeniu rozgrywki klient (gracze oraz obserwatorzy) ponownie trafia do poczekalni.

Wyłączanie serwera: Jest obsługa sygnału Ctrl + C taka, aby gniazda po wyłączeniu serwera są zwalniane (multicast, unicast) (funkcja close()). W przypadku kiedy program działa w trybie demona po użyciu komendy kill gniazdo również będzie zwolnione.

3. Lista większości funkcjonalności/zabezpieczeń zastosowanych w projekcie:

- **Architektura klient-serwer oraz wielowątkowość** (każdy klient ma swój dedykowany wątek do obsługi).
- **Adresacja multicast i server discovery** (klient pyta na grupie multicastowej, na której nasłuchuje serwer o adres tego serwera).
- **Komunikacja TCP unicast** (po wykryciu serwera, klient łączy się z nim poprzez TCP dla stabilnej komunikacji lepszej niż UDP).
- **Wysyłanie tablicy gry do obserwatora przez TLV** (plansza gry jest przesyłana binarnie, a nie w formie tekstowej, co optymalizuje przesyłanie danych, chociaż bardziej chodzi o to aby zwiększyć zaawansowanie kodu i rozwinąć projekt).
- **Tryb demona** (serwer może działać w trybie demona, umożliwiając uruchamianie w tle bez terminala).
- **Logowanie do pliku battleship.log** (wyniki potyczek są zapisywane w pliku, rejestrując zwycięzców i przegranych).
- **Obsługa błędów funkcji sieciowych** (każda operacja sieciowa recv, send, socket, bind jest sprawdzana pod kątem błędów).
- **Zamykanie gniazd i zwalnianie pamięci** (po zakończeniu działania klient i serwer zamykają poprawnie gniazda TCP i UDP).
- **Obsługa sygnałów systemowych** (serwer obsługuje CTRL+C, poprawnie zamykając wszystkie zasoby przed zakończeniem działania, dokładnie wspomniane gniazda wcześniej).
- **Skalowalność kodu** (możliwość zmiany rozmiaru tablic statków, liczby graczy, wyboru trybu demona).
- **Ochrona przed duplikacją nazw użytkowników** (serwer sprawdza, czy podana nazwa użytkownika nie jest już zajęta przez innego aktywnego klienta).
- **Timeout klienta przy handshake username** (klient, który zbyt długo podaje nazwę użytkownika, jest automatycznie rozłączany aby umożliwić innym klientom dołączanie).
- **Konwersja adresów za pomocą inet_pton** (adresy IP wprowadzane przez użytkownika są konwertowane do formatu binarnego).
- **Blokowanie komendy /place w lobby** (gracz nie może rozpocząć ustawiania statków, jeśli nie znajduje się w pokoju gry).

- **Tablica strzałów i tablica trafionych** (klient przechowuje historię swoich strzałów i trafień w grze).
- **Łączenie się z serwerem poprzez opcję --serverIP <adres>** (przy uruchamianiu programu klienta można bezpośrednio podać adres IP serwera w parametrze i nawiązać połączenie na tym adresie).
- **Obsługa komendy /exit** (klient może opuścić grę, a jego nazwa użytkownika jest zwalniana, co pozwala innym na dołączenie).
- **Automatyczne zwracanie klientów do lobby po zakończeniu gry** (gracze i obserwatorzy są automatycznie przenoszeni do lobby po zakończeniu rozgrywki).

Kod podzielony jest na 3 pliki serwer.c, kient.c, oraz gra.h. Kod jest przejrzysty i dobrze sformatowany ze wszystkimi potrzebnymi komentarzami. Przy kompilacji nie wyświetlają się żadne błędy ani ostrzeżenia.

4. Wkład pracy:

Mirosław: logika serwera, struktura sieciowa serwera oraz realizacja pliku nagłówkowego gra.h, uruchomienie, testy, weryfikacja i poprawki stylistyczne

Marcel: pomysł gry, logika klienta oraz struktura sieciowa klienta, weryfikacja i poprawki stylistyczne

Sposób kompilowania programów:

```
root@debian-ps # gcc -o klient klient.c -pthread
root@debian-ps # gcc -o serwer serwer.c -pthread
```