FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Katedra informatiky a výpočetní techniky

Semestrální práce z předmětu
Úvod do počítačových sítí

Síťová hra „Lodě“

Autor:
Stanislav Kafara
A21B0160P
skafara@students.zcu.cz

Obsah

Kapitola 1

Zadání

Úlohu naprogramujte v programovacím jazyku C/C++ anebo Java. Klient bude v Javě a server v C/C++.

Komunikace bude realizována textovým nešifrovaným protokolem nad TCP nebo UDP protokolem.

Výstupy serveru budou v alfanumerické podobě, klient implementuje grafické uživatelské rozhraní.

Server řešte pod operačním systémem Linux, klient může běžet pod OS Windows nebo Linux.

Realizujte konkurentní (paralelní) servery. Server musí být schopen obsluhovat požadavky více klientů souběžně.

Součástí programu bude trasování komunikace, dovolující zachytit proces komunikace na úrovni aplikačního protokolu a zápis trasování do souboru.

Zdrojové kódy organizujte tak, aby od sebe byly odděleny části volání komunikačních funkcí, které jste vytvořili na základě zadání, od částí určených k demonstraci funkčnosti vašeho řešení (grafické rozhraní).

Kapitola 2

Popis hry

„Lodě“ je strategická hra pro dva hráče. Hra se odehrává na mřížkách, kde si hráči rozmístí lodě. Vítězem je ten, kdo první uhodne všechny pozice polí, kde má protihráč rozmístěné lodě a všechny je tak „potopí“.

2.1 Pravidla

Pravidla jsou popsána v [1]. Typ flotily je „ruský“.

Kapitola 3

Popis protokolu

3.1 Formát zpráv

Protokol je textový, nešifrovaný a je postaven nad protokolem TCP. Jednotlivé zprávy jsou odděleny znakem `[LF]` (Line Feed). Oddělovač parametrů zprávy je `|` (roura). Pro zajištění transparentnosti přenosu je znak \ (zpětné lomítko) definován jako escape sekvence. Escapování je nejprve aplikováno na neřídící znaky `|` ve zprávě a poté na neřídící znaky `[LF]` ve zprávě.

3.2 Popis zpráv

Protokol definuje tři typy zpráv:

1. požadavky klienta,
2. aktualizace stavu,
3. udržování spojení.

Klient posílá zprávy serveru jako požadavek, aby provedl nějakou operaci. Jako výsledek mu server vždy odpoví zprávou s výsledkem operace. Server nezávisle posílá klientům zprávy s aktualizacemi stavu. Nezávisle si klient a server vyměňují zprávy se záměrem udržení aktivního spojení.

Po navázání spojení server posílá klientovi `WELCOME` zprávu informující o úspěšném připojení k serveru nebo `LIMIT_CLIENTS|⟨count⟩` informující o překročení limitu počtu klientů připojených k serveru.

Všechny zprávy jsou popsány v tabulkách 3.1, 3.2, 3.3, 3.4 a 3.5.

Požadavky klienta mají smysl jen v některých stavech klienta, jak ukazuje diagram 3.1 a v kladném případě klient přejde do patřičného stavu. Pokud

Tabulka 3.1: Požadavky klienta

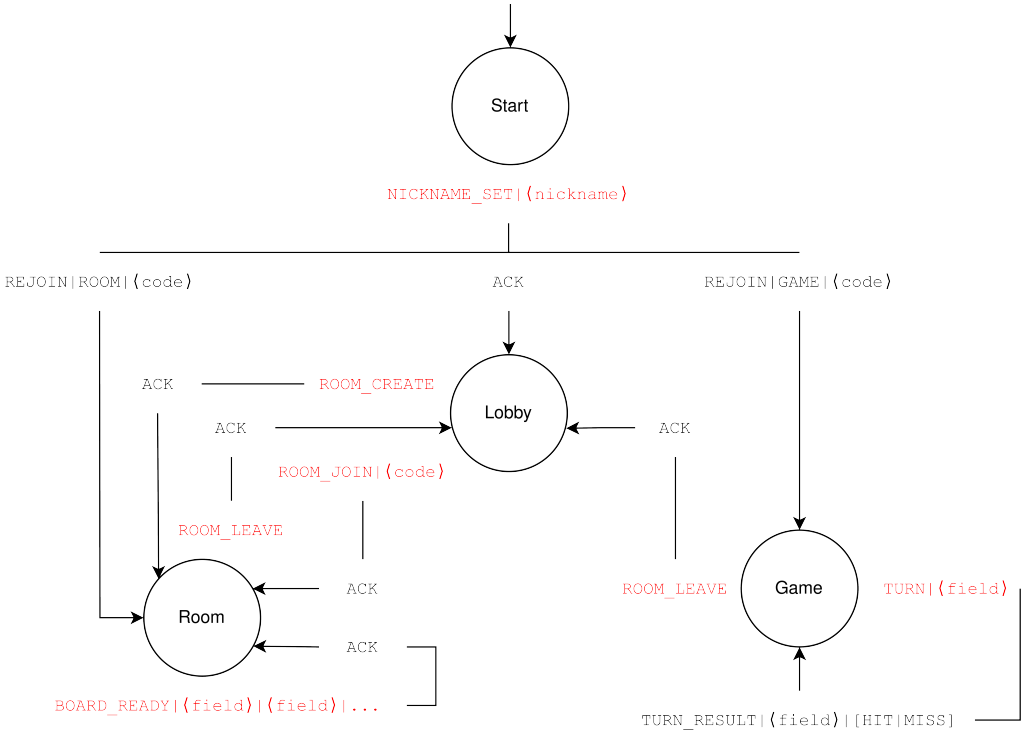
Zpráva	**Popis**
NICKNAME_SET|⟨nickname⟩	Nastavení jména
ROOM_CREATE	Vytvoření herní místnosti
ROOM_JOIN|⟨code⟩	Připojení do herní místnosti
ROOM_LEAVE	Opuštění herní místnosti
BOARD_READY|⟨field⟩|...	Potvrzení hracího pole
TURN|⟨field⟩	Tah

Tabulka 3.2: Požadavky klienta a možné kladné odpovědi serveru

Zpráva	**Možné kladné odpovědi**
NICKNAME_SET|⟨nickname⟩	ACK, REJOIN|[ROOM|GAME]|⟨code⟩
ROOM_CREATE	ROOM_CREATED|⟨code⟩
ROOM_JOIN|⟨code⟩	ACK
ROOM_LEAVE	ACK
BOARD_READY|⟨field⟩|...	ACK
TURN|⟨field⟩	TURN_RESULT|⟨field⟩|[HIT|MISS]

Tabulka 3.3: Požadavky klienta a možné záporné odpovědi serveru

Zpráva	**Možné záporné odpovědi**
NICKNAME_SET|⟨nickname⟩	NICKNAME_EXISTS
ROOM_CREATE	LIMIT_ROOMS|⟨count⟩
ROOM_JOIN|⟨code⟩	ROOM_NOT_EXISTS, ROOM_FULL
ROOM_LEAVE	
BOARD_READY|⟨field⟩|...	BOARD_ILLEGAL
TURN|⟨field⟩	TURN_ILLEGAL, TURN_NOT_YOU



Obrázek 3.1: Stavový diagram klienta

server obdrží požadavek, který není relevantní ke stavu klienta, je mu zaslána zpráva `CONN_TERM` a spojení s ním je ukončeno.

Symbol ⟨`nickname`⟩ představuje jméno klienta, ⟨`code`⟩ kód herní místnosti, ⟨`count`⟩ počet a ⟨`field`⟩ pozici herního pole.

Zpráva `BOARD_READY` má 20 parametrů a jsou jimi pozice lodí na herním poli.

Pozice herního pole je reprezentována dvěma znaky. První určuje řádek a druhý sloupec pole. Pole jsou indexovány od nuly. Možný rozsah tedy je `00` až `99`.

Hodnoty ostatních parametrů nejsou nijak omezeny.

Hra začíná po tom, co si oba klienti nastaví hrací pole. Pokud klient potopí loď, herní pole okolo potopené lodě jsou invalidována. Vítěz hry je uveden jako parametr zprávy `GAME_END`.

Pokud od klienta po určitou dobu nepřijde žádná zpráva, je mu zaslána zpráva `CONN_TERM` a spojení s ním je ukončeno. Pokud je klient nedostupný dlouhou dobu (`LONG`), herní místnost je ukončena, klient zapomenut a protihráč přesunut do stavu v lobby. Pokud klient obnoví spojení a je v nějaké herní místnosti, je mu obnoven stav hry, herní pole zprávou

Tabulka 3.4: Zprávy spravující stav hry

Zpráva	**Popis**
OPPONENT_NO_RESPONSE|[SHORT|LONG]	Protihráč není dostupný.
OPPONENT_BOARD_READY	Protihráč je připraven.
OPPONENT_NICKNAME_SET|⟨nickname⟩	Jméno protihráče
OPPONENT_REJOIN	Protihráč je dostupný.
GAME_BEGIN	Hra začíná.
GAME_END|[YOU|OPPONENT]	Hra končí.
TURN_SET|[YOU|OPPONENT]	Hráč na tahu
OPPONENT_TURN|⟨field⟩|[HIT|MISS]	Tah protihráče
OPPONENT_ROOM_LEAVE	Protihráč opustil místnost.
BOARD_STATE|[YOU|OPPONENT]|⟨state⟩|...	Stav herního pole
INVALIDATE_FIELD|[YOU|OPPONENT]|⟨field⟩	Invalidace herního pole

Tabulka 3.5: Zprávy spravující spojení

Zpráva	**Popis**
KEEP_ALIVE	Odesílatel má v úmyslu ponechat spojení aktivní.
CONN_TERM	Server ukončuje spojení s klientem.

BOARD_STATE a stav protihráče. Zpráva BOARD_STATE má 1 parametr určující klienta a 100 parametrů, každý odpovídající stavu jednoho pole linearizovaného herního pole. Stav ⟨state⟩ herního pole může nabývat hodnot [NONE|SHIP|HIT|MISS|INVALIDATED]. Stav NONE odpovídá prázdnému poli, SHIP poli s lodí, HIT poli se zasáhnutou lodí, MISS poli s nezasáhnutou lodí a INVALIDATED invalidovanému poli.

3.3 Příklad zprávy

- Založení místnosti: ROOM_CREATE[LF]
- Nastavení jména: NICKNAME_SET|skafara[LF]

Kapitola 4

Popis implementace

4.1 Server

Obsluha klienta je realizována stavovým strojem, který ve smyčce čte zprávy od klienta, provede patřičnou operaci na serveru, změní stav klienta, případně i protihráče a pošle zprávy klientovi, případně i protihráči.

Používané síťové rozhraní jsou BSD sockety z jádra UNIX přes jejich knihovní rozhraní v programovacím jazyce C.

4.1.1 Dekompozice problému

Problém je dekomponován do tříd poskytující příslušná rozhraní. Vybraná struktura programu je následující:

- `Server`
- `game`
  - `StateMachine`
  - `Room`
  - `Client`
  - `Board`
- `ntwrk`
  - `SocketAcceptor`
  - `Socket`
- `msgs`

- `Communicator`
- `Message`

Třída `Server` poskytuje operace obsluhující klientovy požadavky. Přijímá spojení reprezentované třídou `Socket` pomocí třídy `SocketAcceptor`. Třída `StateMachine` reprezentuje stavový stroj a řídí logiku obsluhy klienta. Klient je zapouzdřený třídou `Client`. Logika herní místnosti je zapouzdřená ve třídě `Room`. Herní pole je reprezentováno třídou `Board`. Stavový stroj komunikuje s klientem pomocí třídy `Communicator` a vyměňuje si s ním zprávy reprezentované třídou `Message`, mění stav hry a volá operace serveru poskytované třídou `Server`.

4.1.2 Paralelizace

Všechny paralelní činnosti jsou řešeny pomocí vláken. V programu běží vlákna pro následující činnosti:

1. přijímání nových spojení,
2. řešení dlouhodobě nedostupných klientů,
3. udržování spojení s klienty a
4. běh statového stroje každého klienta.

Pro zabránění souběhu na serveru musí vlákno stavového stroje klienta v době vyřizování požadavku vlastnit zámek ve třídě `Server`. Tak je zajištěno, že každá operace proběhne samostatně. Zároveň vlákno pro přijímání nových spojení, vlákno řešící dlouhodobě nedostupné klienty a vlákno pro udržování spojení s klienty musí vlastnit tento zámek, když pracují se sdílenými zdroji serveru.

4.2 Klient

Klient používá architekturu `MVC`. Používané síťové rozhraní jsou sockety poskytované standardní knihovnou `Javy`.

4.2.1 Dekompozice problému

Problém je dekomponován do tříd poskytující příslušná rozhraní. Vybraná struktura programu je následující:

- `Application`
- `models`
  - `ApplicationState`
  - `ClientState`
  - `BoardState`
- `views`
  - `StageManager`
  - `scenes`
  - `components`
- `controllers`
  - `workers`
    - `MessagesManager`
    - `StateMachine`
    - `KeepAlive`
  - `messages`
    - `Communicator`
    - `Message`
  - `Controller`
  - `StateMachineController`

Model je rozdělen na stav aplikace, stav klientů a stav jejich herních polí.

Jednotlivé scény jsou poskládáné z komponent. Řízení zobrazování scén je zapouzdřeno ve třídě `StageManager`. Scény pozorují model a jeho změny.

Zprávy jsou reprezentované třídou `Message`. Controllery jsou rozděleny na `Controller` ovládající požadavky klienta a `StateMachineController` ovládající změny stavu hry. Controllery mění model a případně mění scény.

4.2.2 Paralelizace

V aplikaci běží uživatelská vlákna pro následující činnosti:

1. udržování spojení se serverem,
2. řízení zpráv,

3. běh stavového stroje a

4. vyřizování každého požadavku klienta.

Vlákno `KeepAlive` periodicky posílá serveru zprávu `KEEP_ALIVE` pro ponechání aktivního spojení.

Vlákno `MessagesManager` přijímá zprávy od serveru. Controllery mohou `MessagesManager` informovat o tom, že posílají zprávu a čekají odpověď. `MessagesManager` v tomto případě očekávanou odpověď směřuje controlleru formou splnění `CompletableFuture<Message>`, v jiném případě zprávu předává do fronty třídě `StateMachine`. Pokud odpověď nepřijde do určité doby, směřuje controlleru výjimku `TimeoutException` informující o vypršení času požadavku, nebo se pokusí o znovu-připojení klienta k serveru.

Vlákno `StateMachine` ve smyčce vybírá zprávy z fronty a volá příslušné operace z třídy `StateMachineController`, které vyřídí aktualizaci stavu.

Jelikož více vláken chce posílat zprávy serveru, třída `Communicator` musí vlastnit zámek pro zápis do socketu. Jelikož více vláken přistupuje ke sdíleným zdrojům třídy `MessagesManager`, musí v té době, tj. v době nastavování očekávané odpovědi, nebo v době vyřizování přijaté zprávy, vlastnit zámek třídy `MessagesManager`.

Kapitola 5

Uživatelská příručka

Server vyžaduje překladač jazyka C++ podporující standard jazyka `C++20` a utilitu `cmake` verze alespoň 3.0. Klient vyžaduje `Javu` verze alespoň 17 a utilitu `maven`.

5.1 Server

5.1.1 Překlad a sestavení

Server je přeložitelný a spustitelný v prostředí Unix. Překlad a sestavení programu je řízeno skriptem utility `cmake`. K překladu a sestavení programu dojde po zadání příkazu v adresáři `server`:

```
server$ cmake -Bbuild -Ssrc && cd build && make
```

Výsledný spustitelný soubor se nachází v adresáři `server/build` a jmenuje se `bserver`.

5.1.2 Spuštění

Program přijímá právě čtyři vstupní argumenty. Vzor příkazu spuštění je:

```
bserver --ip=⟨ip⟩ --port=⟨port⟩ --lim-clients=⟨lim-clients⟩
--lim-rooms=⟨lim-rooms⟩
```

Symboly ⟨`ip`⟩ a ⟨`port`⟩ představují IP adresu a port, na kterém bude server naslouchat a symboly ⟨`lim-clients`⟩ a ⟨`lim-rooms`⟩ představují limit počtu hráčů, resp. místností, které server bude odbavovat.

Příkaz spuštění programu může vypadat následovně:

```
bserver --ip=127.0.0.1 --port=50000 --lim-clients=20
--lim-rooms=20
```

V případě zadání nesprávného počtu nebo neplatných hodnot argumentů program vypíše chybovou hlášku a příručku k ovládání programu. V případě,

že dojde k chybě při běhu programu, program vypíše chybovou hlášku a je bezpečně ukončen.

5.2 Klient

5.2.1 Překlad, sestavení a spuštění

Klient je přeložitelný a spustitelný v prostředí Unix a Windows. Překlad, sestavení a spuštění je řízeno skriptem utility `Maven`. K překladu, sestavení a spuštění programu dojde po zadání příkazu v adresáři `client`:

```
client$ mvn javafx:run
```

Kapitola 6

Závěr

Implementované řešení splňuje veškeré požadavky zadání. Programy jsou navrhnuty tak, že jsou přehledné a případně rozšiřitelné.

Seznam obrázků

Seznam tabulek

Bibliografie

[1] Přispěvatelé Wikipedie, *Lodě*, https://cs.wikipedia.org/wiki/Lod%C4%9B, [Online; přístup k datu 17. 1. 2024], 2024.