FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z předmětu

UPS

Prší

15. ledna 2025

Autor:

Martin Reich
A22B0123P
reichm@students.zcu.cz

Cvičící:
Ing. Martin Úbl

Obsah

1 Zadání semestrální práce z předmětu KIV/UPS

- síťová hra pro více hráčů, architektura server-klient (1:N), PC
  - server: C/C++
  - klient: Java, C (i např. Unity), Kotlin nebo jiný vysokoúrovňový jazyk (musí schválit cvičící)
- Varianty zadání:
  - tahová hra (prší, mariáš, šachy, piškvorky, ...)
  - real-time hra (různé skákačky a střílečky, tanky, "Bulánci", ...)
  - pseudo-real-time (Pong, Arkanoid, ...)
- Požadavky na protokol:
  - textový protokol nad transportním protokolem TCP nebo UDP
    - při shodě zadání v rámci cvičení (max. 2) bude každý student používat jiný
  - bez šifrování
  - využijte znalostí ze cvičení při návrhu (např. transparentnost při přenosu dat, apod.)
  - když se nic neděje (žádný hráč nic nedělá), nic se neposílá
    - výjimkou může být občasná ping zpráva
  - na každý požadavek přijde nějaká reakce (byť by šlo pouze o jednoznakové potvrzení, že se operace podařila)
- Legenda:
  - červeně jsou označeny body, jejichž nesplnění automaticky vede k vrácení práce
  - oranžově jsou označeny body, které nemusí vést k vrácení práce (stále jsou ale povinné)
  - modře jsou označeny body, kvůli kterým práce již nebude vracena
- Požadavky na aplikace:
  - aplikace jsou při odevzdání přeloženy standardním nástrojem pro automatický překlad (make, ant, maven, scons, ...; nikoliv bash skript, ani ručně gcc/javac, ani přes IDE)
    - pokud potřebujete nějakou knihovnu, verzi Javy, ... před prezentací si ji zajistěte a nainstalujte

- je zakázáno využití jakékoliv knihovny pro síťovou komunikaci a serializace zpráv – to řeší váš kód sám pouze s BSD sockety (server) a nativní podporou ve standardní knihovně (klient; Java, C, ...); není dovoleno použít ani C++2y networking rozhraní
- kód aplikací je vhodně strukturovaný do modulů, resp. tříd
- kód je dostatečně a rozumně dokumentovaný komentáři
- aplikace (server i klient) jsou stabilní, nepadají na segfaultu (a jiných), všechny výjimky jsou ošetřené, aplikace se nezasekávají (např. deadlock)
- počet hráčů ve hře je omezen pouze pravidly dané hry; vždy by však měla jít dohrát ve 2 lidech (abychom ji mohli testovat)
- po vstupu se hráč dostane do "lobby"s místnostmi, hráč má možnost si vybrat místnost a vstoupit do ní (pokud nepřesahuje limit hráčů); případně je zařazen do fronty a čeká na naplnění herní místnosti
- hra umožňuje zotavení po výpadku způsobené nečekaným ukončením klienta, krátkodobou nebo dlouhodobou síťovou nedostupností
  * dle pravidel hry se pak buď:
    · čeká na návrat hráče (hra se pozastaví)
    · nečeká na návrat hráče, hra pokračuje a po obnovení se hráč připojí do následujícího kola hry (ale hráči je stále po připojení obnoven stav)
    · nečeká na návrat hráče, hra pokračuje (ale hráči je stále po připojení obnoven stav)
  * krátkodobá nedostupnost nesmí nutit hráče k manuálnímu pokusu o připojení (klient vše provede automaticky)
  * dlouhodobá nedostupnost už by naopak měla (i s příslušnou zprávou hráči)
  * všichni hráči v dané hře musí vědět o výpadku protihráče (krátkodobém, dlouhodobém)
  * hráč, který je nedostupný dlouhodobě, je odebrán ze hry; hra pak může místnost ukončit (dle pravidel, většinou to ani jinak nejde) a vrátit aktivního protihráče zpět do lobby
- hráči jsou po skončení hry přesunuti zpět do "lobby"
- obě aplikace musí běžet bez nutnosti je restartovat (např. po několika odehraných hrách)
- obě aplikace ošetřují nevalidní síťové zprávy; protistranu odpojí při chybě

    - náhodná data nevyhovující protokolu (např. z /dev/urandom)
    - zprávy, které formátu protokolu vyhovují, ale obsahují očividně neplatná data (např. tah figurky na pole -1)
    - zprávy ve špatném stavu hry (např. tah, když hráč není ve hře/na tahu, apod.)
    - zprávy s nevalidními vstupy dle pravidel hry (např. šachy – diagonální tah věží)
    - aplikace můžou obsahovat počítadlo nevalidních zpráv a neodpojovat hned po první nevalidní zprávě, až po např. třech
  - obě aplikace mají nějakou formu záznamu (log)
    - zaznamenávají se informace o stavech hráčů, her, popř. chybové hlášení, apod.

- Server:
  - server je schopen paralelně obsluhovat několik herních místností, aniž by se navzájem ovlivňovaly (jak ve smyslu hry, tak např. synchronizace)
  - počet místností (limit) je nastavitelný při spouštění serveru, popř. v nějakém konfiguračním souboru
  - celkový limit hráčů (dohromady ve hře a v lobby) je omezen; rovněž se dá nastavit při spuštění serveru nebo konfigurací
  - stejně tak lze nastavit IP adresu a port, na které bude server naslouchat (parametr nebo konfigurační soubor; ne hardcoded)

- Klient:
  - klient implementuje grafické uživatelské rozhraní (Swing, JavaFX, Unity, popř. jiné dle možností zvoleného jazyka a prostředí) (ne konzole)
  - klient umožní zadání adresy (IP nebo hostname) a portu pro připojení k serveru
  - uživatelské rozhraní není závislé na odezvě protistrany - nezasekává se v průběhu např. připojení na server nebo odesílání zprávy/čekání na odpověď
  - hráč a klient je jednoznačně identifikovaný přezdívkou (neřešíme kolize)

  nepovinné chcete-li, můžete implementovat i jednoduchou registraci (přezdívka + heslo), aby se kolize vyřešily

  - všechny uživatelské vstupy jsou ošetřeny na nevalidní hodnoty
    - totéž platí pro např. ošetření tahů ve hře (např. šachy, aby věž nemohla diagonálně, apod.)

  - klient vždy ukazuje aktuální stav hry - aktuální hrací pole, přezdívky ostatních hráčů, kdo je na tahu, zda není nějaký hráč nedostupný, atp.
  - klient viditelně informuje o nedostupnosti serveru - při startu hry, v lobby, ve hře
  - klient viditelně informuje o nedostupnosti protihráče - ve hře
- Dokumentace obsahuje:
  - základní zkrácený popis hry, ve variantě, ve které jste se rozhodli ji implementovat
  - popis protokolu dostatečný pro implementaci alternativního klienta/serveru:
    * formát zpráv
    * přenášené struktury, datové typy
    * význam přenášených dat a kódů
    * omezení vstupních hodnot a validaci dat (omezení na hodnotu, apod.)
    * návaznost zpráv, např. formou stavového diagramu
    * chybové stavy a jejich hlášení (kdy, co znamenají)
  - popis implementace klienta a serveru (programátorská dokumentace)
    * dekompozice do modulů/tříd
    * rozvrstvení aplikace
    * použité knihovny, verze prostředí (Java), apod.
    * metoda paralelizace (select, vlákna, procesy)
  - požadavky na překlad, spuštění a běh aplikace (verze Javy, gcc, ...)
  - postup překladu
  - závěr, zhodnocení dosažených výsledků

2 Popis hry

Prší je karetní hra pro dva hráče. Každý hráč začíná se čtyřmi kartami. Hráči střídavě pokládají karty na odkládací balíček, přičemž karta musí mít buď stejnou hodnotu nebo stejnou barvu jako vrchní karta na odkládacím balíčku. Pokud hráč nemůže položit kartu, musí si vzít kartu z dobíracího balíčku. Speciální karty mají následující efekty:

- Sedmička - následující hráč si musí vzít dvě karty a vynechává tah

- Eso - následující hráč vynechává tah

Vítězem se stává hráč, který se první zbaví všech karet.

3 Síťový protokol

3.1 Formát zpráv

Komunikace probíhá pomocí UDP protokolu s JSON zprávami. Každá zpráva obsahuje povinné pole "type", které určuje typ zprávy.

3.2 Typy zpráv

Připojení ke hře

```
{"type": "connect",
  "name": "jméno_hráče"
}

{
    "type": "connect_ack",
    "player_id": "jméno_hráče",
    "waiting_for_player": true/false,
    "players": ["hráč1", "hráč2"],  // pouze pokud
       waiting_for_player=false
    "current_player": "jméno_hráče", // pouze pokud
       waiting_for_player=false
    "deck_size": 31,                  // pouze pokud
       waiting_for_player=false
    "discard_pile": 1,               // pouze pokud
       waiting_for_player=false
    "top_card": {"suit": "♡", "value": "7"} // pouze
       pokud waiting_for_player=false
}
```

Herní akce

```
{
    "type": "play_card",
    "card": "7♡",
    "player_name": "jméno_hráče"
}

{
    "type": "draw_card",
```

```
    "player_name": "jméno_hráče"
}
```

Aktualizace stavu hry

```
{
    "type": "game_state_update",
    "players": ["hráč1", "hráč2"],
    "current_player": "jméno_hráče",
    "deck_size": 23,
    "discard_pile": 1,
    "top_card": {"suit": "♡", "value": "7"},
    "hráč1": [{"suit": "♡", "value": "8"}, ...],
    "hráč2": [{"suit": "♠", "value": "K"}, ...]
}
```

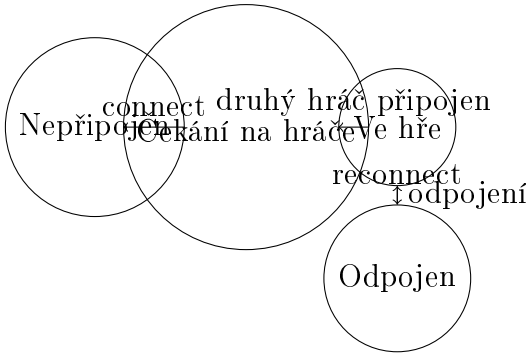
Chybové stavy

```
{
    "type": "error",
    "message": "popis chyby"
}

{
    "type": "name_taken",
    "message": "Jméno je již používáno"
}
```

Odpojení a reconnect

```
{
    "type": "player_disconnected",
    "player": "jméno_hráče",
    "message": "Hráč se odpojil"
}

{
    "type": "player_reconnected",
    "player": "jméno_hráče",
    "message": "Hráč se znovu připojil"
}
```

3.3 Stavový diagram



4 Implementace

4.1 Server (C++)

Server je implementován v C++ s následující strukturou:

- **game_server** - Hlavní třída serveru, zpracovává síťovou komunikaci
- **Lobby** - Třída reprezentující herní místnost pro dva hráče
- **Player** - Třída reprezentující hráče
- **Deck** - Třída reprezentující balíček karet
- **SimpleJSON** - Třída pro práci s JSON zprávami

Paralelizace je řešena pomocí vláken:

- Hlavní vlákno - zpracování příchozích zpráv
- Kontrolní vlákno - kontrola odpojených hráčů

4.2 Klient (Python)

Klient je implementován v Pythonu s GUI v Tkinter:

- **MainMenu** - Třída pro hlavní menu a připojení ke hře
- **CardGameGUI** - Třída pro herní rozhraní
- **NetworkClient** - Třída pro síťovou komunikaci
- **Player** - Třída reprezentující hráče

Paralelizace je řešena pomocí vláken:

- GUI vlákno - hlavní vlákno aplikace
- Přijímací vlákno - příjem zpráv ze serveru

- Heartbeat vlákno - odesílání heartbeat zpráv

5 Požadavky a překlad

5.1 Server

- C++17 nebo novější
- CMake 3.10 nebo novější
- Windows: Visual Studio 2019 nebo novější
- Linux: GCC 7.0 nebo novější

Překlad:

```
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
```

5.2 Klient

- Python 3.8 nebo novější
- Tkinter (součást standardní knihovny Pythonu)

Spuštění:

```
python main.py
```

6 Závěr

Implementace splňuje všechny základní požadavky na síťovou karetní hru. Server zvládá paralelní obsluhu více herních místností a řeší odpojení hráčů. Klient poskytuje přehledné grafické rozhraní a implementuje všechny herní mechaniky. Možná vylepšení zahrnují:

- Implementace více variant pravidel
- Přidání podpory pro více než dva hráče
- Implementace statistik a žebříčku hráčů
- Přidání chat systému