FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z předmětu KIV/UPS

# Síťová hra pro více hráčů - kostky

**Autoři:**
Filip Černý
A21B0100P
cernyf@students.zcu.cz

**Datum:**
31. ledna 2025

# Obsah

# Základní popis hry

Implementace hry s TCP protokolem pro N:1 server klient vztah. Hra kostek probíhá zde takto:

- Připojení do hry pomocí IP adresy, Portu a přezdívky
- Stránka Lobby - vytvoření místnosti nebo připojení do hry
- Místnost - začátek hry
- Hráči se střídají v kolech podle pořadí připojení
  - pokud nehrají vidí aktuální výsledky ostatních hráčů
- **Jednotlivé kolo**
  - Hráč hodí 6 kostkami
  - Vybere z hozených kostek možné libovolných počet z těchto možností nebo dostane zprávu o hození kombinace bez možnosti výběru kde pokračuje zpátky do místnosti a hraje jiný hráč
    - *Možnost*
      - hodnota 5 - skóre 50
      - hodnota 1 - skóre 100
  - hráč odešle vybrané kostky
    - následně buď pokračuje v kolu a hází znovu nebo vyhrál s maximálním skórem jako první

# Popis protokolu

## 2.1 Formát zprávy

```
struktura:
stamp;command_id;timestamp;{player_nickname};{args...}

příklad:
KIVUPS012024-12-31 15:30:00.000000{nickname}{}
```

### 2.1.1 Formát parametrů

**Jednoduché parametry formát:**

```
{"gameName":"Game3", "maxPlayers":"3"}
```

**Pole parametrů formát:**

```
{"gameList":"[{"gameName":"Game1","maxPlayers":"4","connectedPlayers":"2"};{"gameName"
```

### 2.1.2 Struktura dat

- zde popisuji různé verze pole parametrů, které mohou být přenášeny v tomto prokolu
- každá položka z loistu obsahuje tyto hodnoty

**gameList:**

- Název hry (**gameName**)
  - *string* - pouze velké znaky, malé znaky, čísla
- Maximální počet hráčů (**maxPlayers**)
  - *int* - přirozené číslo v definovaných hranicích v *config.json* na serveru
- Připojení hráči (**connectedPlayers**)
  - *int* - přirozené číslo v definovaných hranicích v *config.json* na serveru

**playerList:**

- Jméno hráče (**playerName**)
  - *string* - pouze velké znaky, malé znaky, čísla
- Připojení (**isConnected**)
  - *0* - nepravda
  - *1* - pravda

**gameData:**

- Jméno hráče (**playerName**)
  - *string* - pouze velké znaky, malé znaky, čísla
- Stav Připojení (**isConnected**)
  - *0* - nepravda
  - *1* - pravda
- Skóre (**score**)
  - *int* - přirozené číslo v definovaných hranicích v *config.json* na serveru
- Kdo je na tahu (**isTurn**)
  - *0* - nepravda
  - *1* - pravda

**cubeValues:**

- Hodnota (value)
  - *int* - v rozmezí (1-6)

## 2.2 Velikosti

- **Hlavička:**
  - stamp: 6 bajtů
  - command_id: 2 bajty
  - timestamp: 26 bajtů
  - formát: `%Y-%m-%d %H:%M:%S.%f`

## 2.3 Síťové zprávy

Každá zpráva je definována unikatáním identifikátorem a následně jejími parametry

### 2.3.1 Klient → Server

Zprávy, které klient odesílá serveru.

**ClientLogin**

- **CommandID:** 1
- **Params:** []
- Přezdívka hráče je součástí hlavičky.

**Odpověď:**

- **ResponseServerGameList** – seznam dostupných her
- **ResponseServerError**
  - Duplicitní přezdívka
  - Jiná chyba

**ClientCreateGame**

- **CommandID:** 2
- **Params:** [”gameName”, ”maxPlayers”]

**Odpověď:**

- **ResponseServerSuccess** – hra byla úspěšně vytvořena
- **ResponseServerError** – chyba při vytváření hry

**ClientJoinGame**

- **CommandID:** 3
- **Params:** [”gameName”]

**Odpověď:**

- **ResponseServerSuccess** – připojení do hry bylo úspěšné
- **ResponseServerError** – chyba při připojení

**ClientStartGame**

- **CommandID:** 4
- **Params:** []

**Odpověď:**

- **ResponseServerSuccess** – hra byla úspěšně spuštěna
- **ResponseServerError** – chyba při spuštění hry

### 2.3.2 Odpovědi Server → Klient

**Obecné odpovědi**

- **ResponseServerSuccess**
  - **CommandID:** 30
  - **Params:** []
- **ResponseServerError**
  - **CommandID:** 32
  - **Params:** ["message"]

**Specifické odpovědi**

- **ResponseServerGameList**
  - **CommandID:** 33
  - **Params:** ["gameList"] (List)
- **ResponseServerSelectCubes (ResponseServerDiceNext)**
  - **CommandID:** 34
  - **Params:** ["cubeValues"] (List)
- **ResponseServerEndTurn (ResponseServerDiceEndTurn)**
  - **CommandID:** 35
  - **Params:** []

### 2.3.3 Server → Klient

**Server → Všichni klienti**

**Jednorázové zprávy**

- **ServerUpdateStartGame**
  - **CommandID:** 41
  - **Params:** []
- **ServerUpdateEndScore**
  - **CommandID:** 42
  - **Params:** ["playerName"]

**Průběžné zprávy**

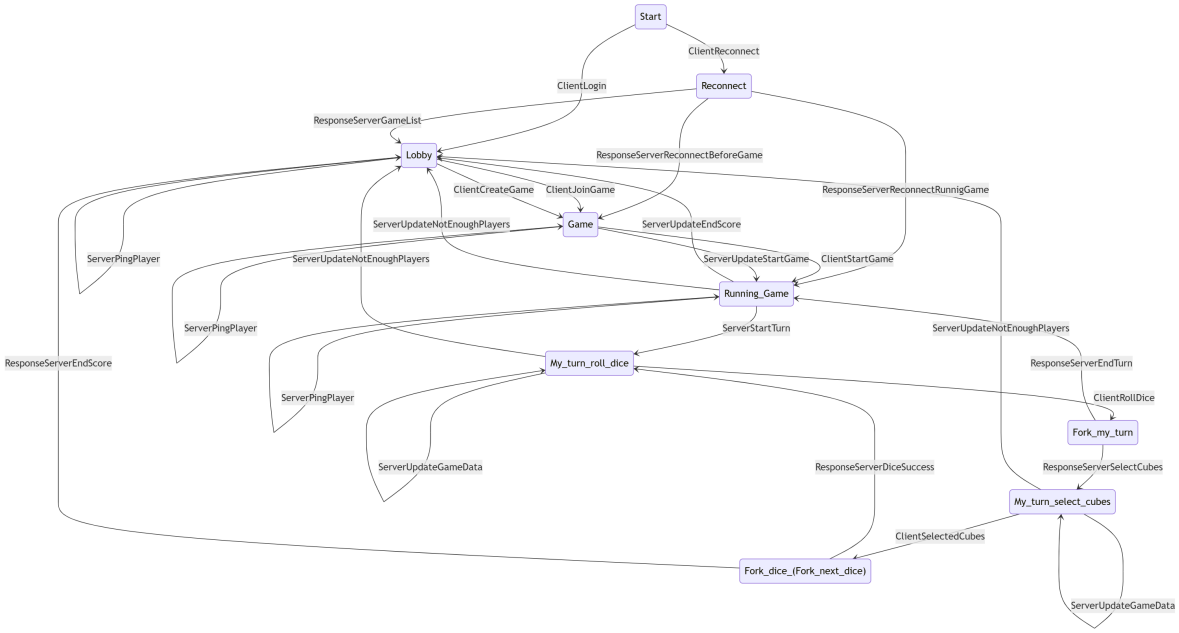
- **ServerUpdateGameData**
  - **CommandID:** 43
  - **Params:** ["gameData"] (List)
- **ServerUpdateGameList**
  - **CommandID:** 44
  - **Params:** ["gameList"] (List)

**Server → Jeden klient**

- **ServerStartTurn**
  - **CommandID:** 49
  - **Params:** []
- **ServerPingPlayer**
  - **CommandID:** 50
  - **Params:** []

### 2.3.4 Odpovědi Klient → Server

- **ResponseClientSuccess**
  - **CommandID:** 60
  - **Params:** []




Obrázek 2.1: Statový automat protokolu

# Popis implementace klienta a serveru

### 3.0.1 Klient

#### Dekompozice do modulů

Klientská aplikace je rozdělena na několik modulů:

- **main** – hlavní modul, který inicializuje a spouští klienta.

#### Backend

- **parser** – převádí textová vstupní data na interní reprezentaci a naopak.
- **server_communication** – zajišťuje síťovou komunikaci se serverem (odesílání a přijímání zpráv).
- **state_manager** – implementuje stavový automat, který řídí chování klienta.

#### Frontend

- **views** – moduly pro jednotlivé obrazovky klientské aplikace:
  - *start_page* – úvodní stránka pro zadání adresy serveru a jména hráče.
  - *lobby* – stránka pro čekání na ostatní hráče a správu herních místností.
  - *before_game* – stránka, kde se čeká na začátek hry nebo ji lze zahájit.
  - *running_game* – stránka, kde probíhá hra.
  - *my_turn_roll_dice* – fáze tahu hráče, ve které hází kostkou.
  - *my_turn_select_cubes* – fáze tahu hráče, kde vybírá kostky, které chce ponechat.
  - Pomocné funkce (*helper functions*) a rozhraní (*interface*) pro jednotlivé views.
- **ui_manager** – řídí zobrazení views a jejich vzájemnou komunikaci.

#### Shared (Sdílené moduly)

- Konstanty a funkce používané jak v backendu, tak ve frontendu.

#### Rozvrstvení aplikace

Hlavní částí aplikace je modul **server_communication**, který:

- Používá stavový automat (**state_manager**) a parser (**parser**).
- Reaguje na příchozí zprávy ze serveru a aktualizuje stav aplikace.

### 3.0.2 Použité technologie

#### Knihovny

- **Tkinter** – knihovna pro grafické uživatelské rozhraní klienta.
- **stateless** – knihovna pro implementaci stavového automatu.

#### Další technologie

- Programovací jazyk: Python 3.10 a standardní knihovny.

### 3.0.3 Metody paralelizace

- **GUI** – využívá více vláken pro současné zpracování zpráv ze serveru a aktualizaci zobrazených dat.
- Aplikace neustále naslouchá příchozím zprávám a v reálném čase aktualizuje uživatelské rozhraní.

### 3.0.4 Server

#### Dekompozice do modulů

Serverová aplikace je rozdělena do několika modulů:

- **internal/main.go** – hlavní vstupní bod aplikace obsahující funkci `main`, která inicializuje a spouští ostatní moduly.
- **Zpracování příkazů (Command Processing)** – modul zajišťující interpretaci a provádění příkazů od klienta.
- **Datové modely (Models)** – struktury potřebné pro běh aplikace:
  - *State Machine* – stavový automat reprezentující stav hry.
  - *Player* – hráč a jeho připojení k serveru.
  - *Game* – samotná hra a její stav.

  - *Message* – zpráva, kterou klient odesílá serveru.
  - Další datové struktury pro správu více instancí těchto objektů.

- **Síťová komunikace (Network)** – zajišťuje příjem a odesílání zpráv mezi klientem a serverem.
- **Parser** – převádí přijaté zprávy na vnitřní objekty a připravuje zprávy k odeslání.
- **Hlavní smyčka serveru (Server Listen)** – naslouchá zprávám od klientů a zpracovává je pomocí výše uvedených modulů.

### 3.0.5 Použité technologie

#### Knihovny

- **stateless** – knihovna pro implementaci stavových automatů.
- **github.com/pkg/errors** – nástroj pro lepší práci s chybami.
- **github.com/sirupsen/logrus** – knihovna pro logování událostí na serveru.

#### Další technologie

- Programovací jazyk: Go 1.23.

### 3.0.6 Metody paralelizace

Pro zajištění plynulého chodu serveru jsou využity následující paralelizační techniky:

- **Komunikace s klientem** – pro každého připojeného klienta je vytvořeno samostatné vlákno (goroutine), které zajišťuje příjem a odesílání zpráv.
- **Ukončení spojení** – pokud se klient odpojí, server po určitou dobu uchovává jeho stav a umožňuje mu opětovné připojení. Pokud se klient nepřipojí včas, spojení je definitivně ukončeno.
- **Asynchronní zpracování zpráv** – server využívá gorutiny k efektivnímu zpracování příchozích zpráv.

### 3.0.7 Spuštění aplikace

#### Spuštění serveru

1. Nainstalujte **Go 1.23** a **make**.
2. Stáhněte si zdrojový kód.

3. Otevřete terminál a přejděte do složky `GameServer`.
4. Spusťte aplikaci příkazem:

```
make
```

**Spuštění klienta**

1. Nainstalujte **Python 3.1** a **make**.
2. Stáhněte si zdrojový kód.
3. Otevřete terminál a přejděte do složky `GameClient`.
4. Spusťte aplikaci příkazem:

```
make
```

# Závěr

Projekt implementuje funkční síťovou hru, která umožňuje interakci hráčů přes internet a síťovou komunikaci. Hra zajišťuje znovu připojení při krátkodobé nedostupnosti serveru a následné vyhodnocení stavu při nedostupnosti klienta a nebo serveru. Aplikace dovoluje komunikaci jen podle předepsaného protokolu a s validními daty.

Tento projekt by se dal rozšířit o větší funkčnost a nebo i o lepší uživatelské prostředí na, které v tomto projektu nebyl dán důraz.