Dokumentace - Síťová hra Nim

Předmět: KIV/UPS - Úvod do počítačových sítí
Autor: David Sambazov
Datum: 2025

Obsah

1. Popis hry

1.1 Pravidla hry Nim (misère varianta)

Nim je strategická hra pro dva hráče. V této implementaci se hraje zjednodušená varianta s jednou hromádkou:

- **Počáteční stav:** Na začátku hry je na hromádce **21 kamínků**
- **Průběh tahu:** Hráči se střídají a v každém tahu odeberou **1 až 3 kamínky**
- **Podmínka výhry:** Kdo odebere **poslední kamínek, prohrává** (misère pravidlo)
- **Speciální pravidlo:** Každý hráč má možnost **jednou za hru přeskočit svůj tah**

1.2 Architektura

- **Server-klient** architektura (1:N)
- Více souběžných her v oddělených místnostech
- Textový protokol nad **TCP**

2. Specifikace protokolu

2.1 Formát zpráv

Protokol používá textové zprávy ve formátu:

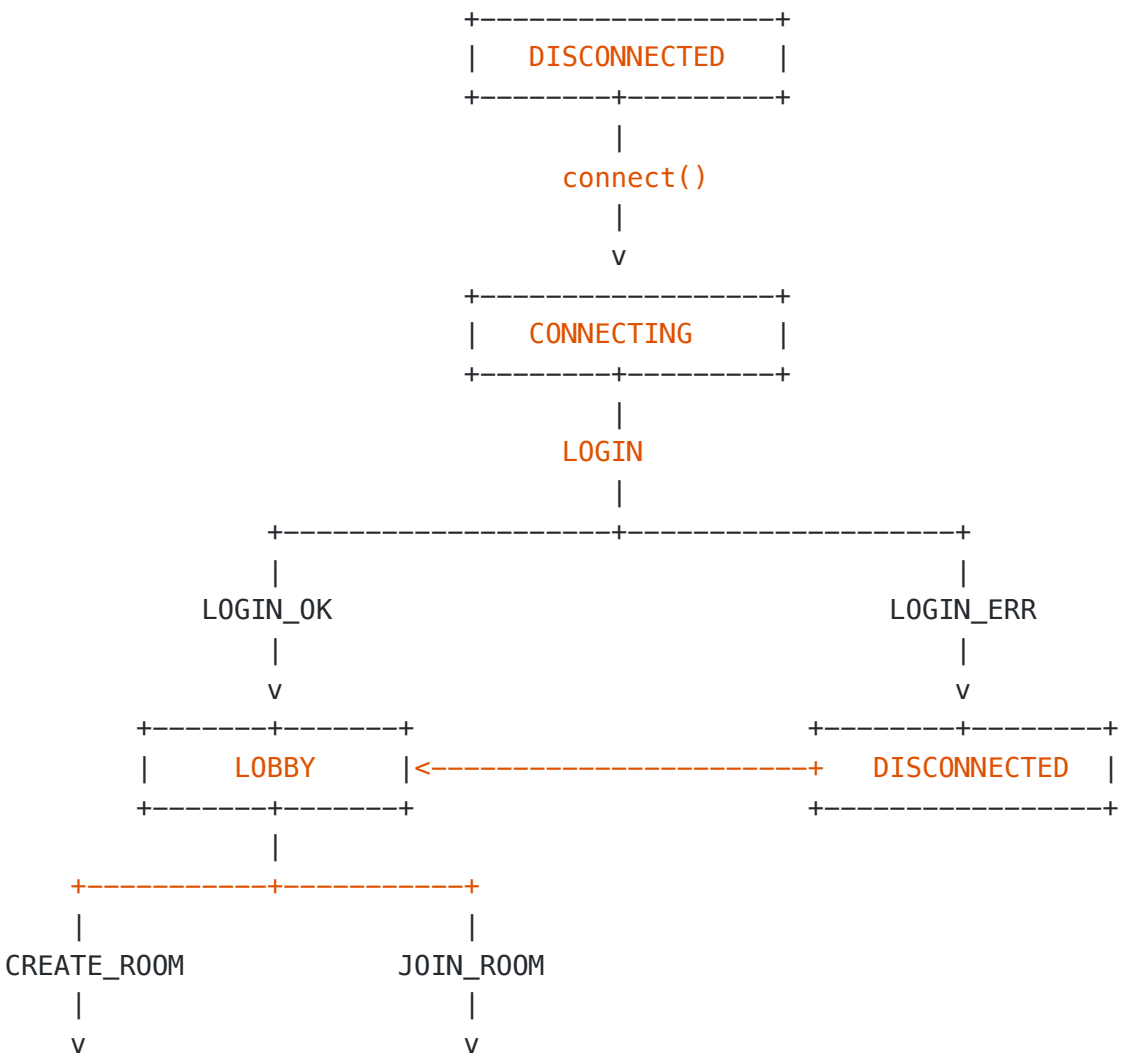
```
COMMAND;param1;param2;...\n
```

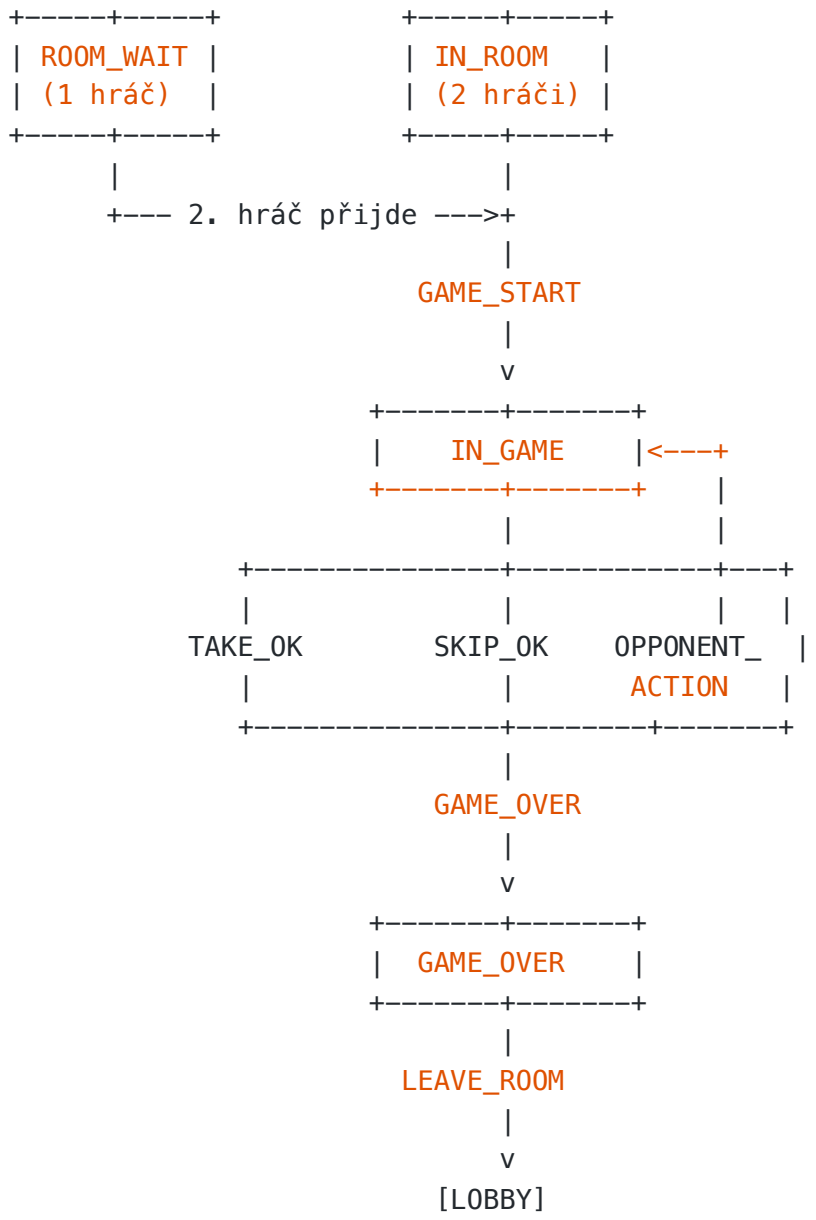
- **Oddělovač parametrů:** středník (`;`)
- **Ukončovač zprávy:** nový řádek (`\n` nebo `\r\n`)
- **Kódování:** UTF-8
- **Maximální délka zprávy:** 1024 bajtů

2.2 Datové typy

Typ	Popis	Příklad
STRING	Textový řetězec (bez středníků a \n)	`player1` , `MojeHra`
INT	Celé číslo (dekadicky)	`21` , `3` , `0`
BOOL	Logická hodnota (0/1)	`0` = false, `1` = true

2.3 Stavový diagram klienta





2.4 Klientské zprávy (klient → server)

Zpráva	Formát	Popis	Platný stav
LOGIN	`LOGIN;nickname:STRING`	Přihlášení hráče	CONNECTING
LIST_ROOMS	`LIST_ROOMS`	Žádost o seznam místností	LOBBY
CREATE_ROOM	`CREATE_ROOM;name:STRING`	Vytvoření nové místnosti	LOBBY
JOIN_ROOM	`JOIN_ROOM;room_id:INT`	Připojení do místnosti	LOBBY
LEAVE_ROOM	`LEAVE_ROOM`	Opuštění místnosti	IN_ROOM, IN_GAME
TAKE	`TAKE;count:INT`	Odebrání kamínků (1-3)	IN_GAME (na tahu)

Zpráva	Formát	Popis	Platný stav
SKIP	`SKIP`	Přeskočení tahu	IN_GAME (na tahu, má skip)
PING	`PING`	Kontrola spojení	kdykoli
PONG	`PONG`	Odpověď na PING	kdykoli
LOGOUT	`LOGOUT`	Odhlášení	kdykoli

2.5 Serverové zprávy (server → klient)

Zpráva	Formát
LOGIN_OK	`LOGIN_OK`
LOGIN_ERR	`LOGIN_ERR;code:INT;reason:STRING`
ROOMS	`ROOMS;count:INT;id,name,players,max;...`
ROOM_CREATED	`ROOM_CREATED;room_id:INT`
ROOM_JOINED	`ROOM_JOINED;room_id:INT;opponent:STRING`
ROOM_ERR	`ROOM_ERR;code:INT;reason:STRING`
LEAVE_OK	`LEAVE_OK`
WAIT_OPPONENT	`WAIT_OPPONENT`
GAME_START	`GAME_START;stones:INT;your_turn:BOOL;opponent:STRING`
TAKE_OK	`TAKE_OK;remaining:INT;your_turn:BOOL`
TAKE_ERR	`TAKE_ERR;code:INT;reason:STRING`
SKIP_OK	`SKIP_OK;your_turn:BOOL`

Zpráva	Formát
SKIP_ERR	`SKIP_ERR;code:INT;reason:STRING`
OPPONENT_ACTION	`OPPONENT_ACTION;action:STRING;param:INT;remaining:INT`
GAME_OVER	`GAME_OVER;winner:STRING;loser:STRING`
GAME_RESUMED	`GAME_RESUMED;stones:INT;your_turn:BOOL;your_skips:INT;opp_skips`
PLAYER_STATUS	`PLAYER_STATUS;nickname:STRING;status:STRING`
PING	`PING`
PONG	`PONG`
ERROR	`ERROR;code:INT;message:STRING`
SERVER_SHUTDOWN	`SERVER_SHUTDOWN`

2.6 Chybové kódy

Kód	Konstanta	Popis	Kdy nastane
0	ERR_NONE	OK	-
1	ERR_INVALID_FORMAT	Neplatný formát zprávy	Chybí parametry, špatný formát
2	ERR_UNKNOWN_COMMAND	Neznámý příkaz	Neexistující příkaz
3	ERR_INVALID_PARAMS	Neplatné parametry	count < 1 nebo > 3
4	ERR_NOT_LOGGED_IN	Hráč není přihlášen	Akce před LOGIN
5	ERR_ALREADY_LOGGED_IN	Hráč je již přihlášen	Opakovaný LOGIN
6	ERR_NICKNAME_TAKEN	Přezdívka je obsazena	Jiný hráč má stejnou přezdívku

Kód	Konstanta	Popis	Kdy nastane
7	ERR_NICKNAME_INVALID	Neplatná přezdívka	Zakázané znaky, příliš dlouhá
8	ERR_ROOM_NOT_FOUND	Místnost neexistuje	JOIN na neexistující room_id
9	ERR_ROOM_FULL	Místnost je plná	Místnost má 2 hráče
10	ERR_ROOM_NAME_TAKEN	Název místnosti je obsazen	CREATE s existujícím názvem
11	ERR_NOT_IN_ROOM	Hráč není v místnosti	LEAVE/TAKE mimo místnost
12	ERR_NOT_IN_GAME	Hráč není ve hře	TAKE/SKIP mimo hru
13	ERR_NOT_YOUR_TURN	Není váš tah	TAKE/SKIP když není na tahu
14	ERR_INVALID_MOVE	Neplatný tah	count > zbývající kamínky
15	ERR_NO_SKIPS_LEFT	Žádné přeskočení nezbývá	SKIP bez zbývajících skipů
16	ERR_SERVER_FULL	Server je plný	Dosažen limit klientů
17	ERR_MAX_ROOMS	Dosažen limit místností	CREATE při plném serveru
18	ERR_GAME_IN_PROGRESS	Hra již probíhá	-
19	ERR_GAME_PAUSED	Hra je pozastavena	TAKE/SKIP při odpojeném soupeři
99	ERR_INTERNAL	Interní chyba serveru	Neočekávaná chyba

2.7 Validace vstupů

Přezdívka (nickname):

- Délka: 1-32 znaků
- Povolené znaky: písmena (a-z, A-Z), číslice (0-9), podtržítko (_)
- Musí začínat písmenem
- Case-sensitive (rozlišuje velká/malá)

Název místnosti (room name):

- Délka: 1-64 znaků

- Povolené znaky: písmena, číslice, podtržítko, mezera
- Nesmí začínat/končit mezerou

Počet kamínků k odebrání (count):

- Rozsah: 1-3 (integer)
- Nesmí přesáhnout aktuální počet kamínků na hromádce

Room ID:

- Integer ≥ 0
- Musí odpovídat existující místnosti

2.8 Příklady komunikace

Úspěšné přihlášení a vytvoření místnosti:

```
C: LOGIN;player1
S: LOGIN_OK
C: CREATE_ROOM;MojeHra
S: ROOM_CREATED;0
S: WAIT_OPPONENT
```

Připojení druhého hráče a začátek hry:

```
C: LOGIN;player2
S: LOGIN_OK
C: JOIN_ROOM;0
S: ROOM_JOINED;0;player1
S: GAME_START;21;1;player2    (player1 dostane)
S: GAME_START;21;0;player1    (player2 dostane)
```

Průběh tahu:

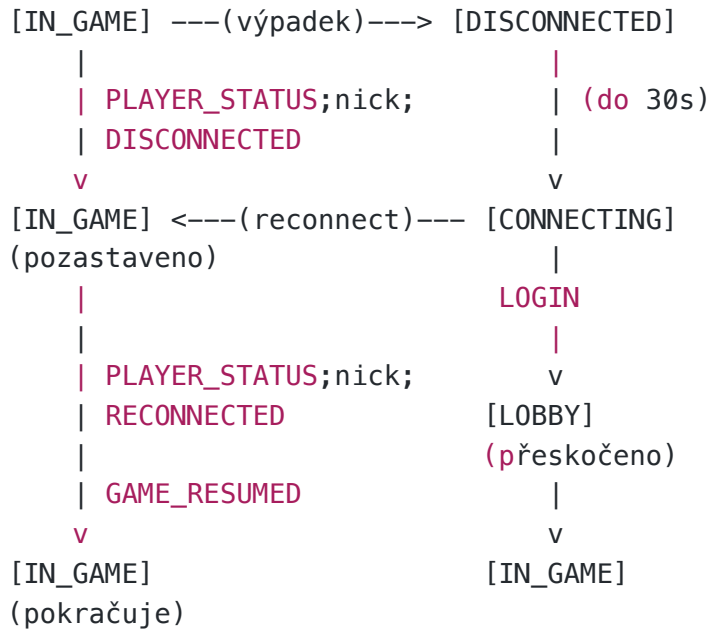
```
C: TAKE;3                      (player1 bere 3)
S: TAKE_OK;18;0                (zbývá 18, už nejsi na tahu)
(player2 dostane):
S: OPPONENT_ACTION;TAKE;3;18
```

Konec hry:

```
(player2 vzal poslední kamínek – prohrává)
S: GAME_OVER;player1;player2   (oba hráči dostanou stejnou zprávu)
```

2.9 Řešení výpadků

Diagram reconnectu:



Krátkodobý výpadek (do 30 sekund):

- Klient se automaticky pokouší o reconnect (max 10 pokusů)
- Exponenciální backoff: 3s, 6s, 9s...
- Server pozastaví hru (`PLAYER_STATUS;nick;DISCONNECTED`)
- Po reconnectu server pošle `GAME_RESUMED` a `PLAYER_STATUS;nick;RECONNECTED`

Dlouhodobý výpadek (nad 30 sekund):

- Odpojený hráč prohrává
- Protihráč dostane `GAME_OVER` a je vrácen do lobby
- Klient musí zahájit nové spojení

2.10 Ochrana proti útokům

Binární data (/dev/urandom):

- Server kontroluje, zda příchozí data obsahují pouze tisknutelné ASCII znaky (32-126) a \n, \r
- Nevalidní data jsou zahozena a počítána jako chybná zpráva
- Po 3 nevalidních zprávách je klient odpojen

Flood protection:

- Maximálně 20 zpráv za sekundu od jednoho klienta
- Zpráva bez ukončovacího znaku může mít max 256 bajtů

- Překročení = odpojení

Login timeout:

- Klient musí poslat LOGIN do 30 sekund po připojení
- Jinak je automaticky odpojen

TCP Keepalive:

- Server i klient používají TCP keepalive pro detekci mrtvých spojení
- `TCP_KEEPIDLE = 10s`, `TCP_KEEPINTVL = 5s`, `TCP_KEEPCNT = 3`

PING/PONG:

- Klient posílá PING každých 8 sekund
- Timeout pro PONG je 15 sekund
- Při timeoutu = předpokládá se ztráta spojení → reconnect

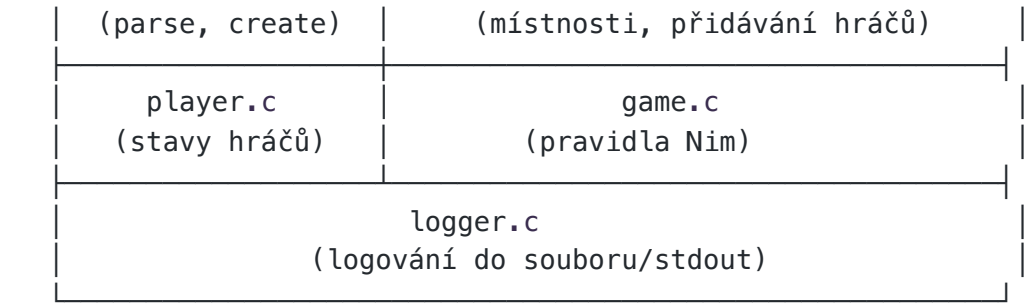
3. Implementace serveru

3.1 Struktura projektu

```
server_src/
├── Makefile              # Build script
├── include/
│   └── config.h          # Konfigurační konstanty
└── src/
    ├── main.c            # Vstupní bod, parsování argumentů
    ├── server.c/h        # Hlavní serverová logika, select()
    ├── protocol.c/h      # Parsování a tvorba zpráv
    ├── player.c/h        # Správa hráčů a jejich stavů
    ├── room.c/h          # Správa herních místností
    ├── game.c/h          # Herní logika Nim
    └── logger.c/h        # Logování
```

3.2 Rozvrstvení aplikace

```
┌─────────────────────────────────────────────────────┐
│                      main.c                         │
│              (spuštění, argumenty)                  │
├─────────────────────────────────────────────────────┤
│                     server.c                        │
│    (socket, select(), accept, read, write, timeouts)│
├──────────────────────┬──────────────────────────────┤
│    protocol.c        │              room.c          │
```



3.3 Dekompozice modulů

main.c

- Parsování argumentů příkazové řádky (`getopt`)
- Inicializace loggeru
- Volání `server_init()` a `server_run()`

server.c (900+ řádků)

- `server_init()` - vytvoření socketu, bind, listen
- `server_run()` - hlavní smyčka s `select()`
- `accept_new_client()` - přijetí nového spojení
- `read_from_client()` - čtení dat, buffering, parsování
- `server_handle_message()` - dispatch podle typu zprávy
- `server_send_to_player()` - odesílání zpráv
- `server_check_timeouts()` - kontrola ping/pong, login timeout
- `server_handle_disconnect()` - zpracování odpojení

protocol.c

- `protocol_parse_message()` - parsování příchozí zprávy
- `protocol_create_*()` - funkce pro tvorbu odchozích zpráv
- `protocol_validate_nickname()` - validace přezdívky

player.c

- Struktura `Player` s veškerým stavem
- `player_reset()` - reset hráče (s/bez zachování pro reconnect)
- `player_find_by_nickname()` - vyhledání podle přezdívky
- `player_reconnect_timeout_expired()` - kontrola timeoutu

room.c

- Struktura `Room` s hráči a hrou
- `room_create()` , `room_add_player()` , `room_remove_player()`

- `room_get_opponent()` - získání protihráče
- `room_find_by_id()`, `room_find_by_name()`

game.c

- Struktura `Game` se stavem hry
- `game_start()`, `game_take()`, `game_skip()`
- `game_is_over()`, `game_get_winner()`
- `game_pause()`, `game_resume()`

logger.c

- Makra `LOG_DEBUG`, `LOG_INFO`, `LOG_WARNING`, `LOG_ERROR`
- Thread-safe zápis do souboru nebo stdout

3.4 Metoda paralelizace

Server používá **single-threaded event-driven** architekturu s `select()`:

```
while (running) {
    // Připrav množinu file descriptorů
    FD_ZERO(&read_fds);
    FD_SET(listen_fd, &read_fds);
    for (int i = 0; i < max_clients; i++) {
        if (players[i].is_active && players[i].socket_fd >= 0) {
            FD_SET(players[i].socket_fd, &read_fds);
        }
    }

    // Čekej na aktivitu (max 1 sekunda)
    struct timeval timeout = {1, 0};
    int activity = select(max_fd + 1, &read_fds, NULL, NULL, &timeout);

    // Zpracuj nová spojení
    if (FD_ISSET(listen_fd, &read_fds)) {
        accept_new_client(server);
    }

    // Zpracuj data od klientů
    for (int i = 0; i < max_clients; i++) {
        if (FD_ISSET(players[i].socket_fd, &read_fds)) {
            read_from_client(server, &players[i]);
        }
    }

    // Kontroluj timeouty (ping/pong, login, reconnect)
    server_check_timeouts(server);
}
```

Výhody:

- Jednoduchá implementace bez synchronizace
- Nízká paměťová náročnost
- Žádné race conditions

3.5 Konfigurace

```
./nim_server [-a ADDRESS] [-p PORT] [-c MAX_CLIENTS] [-r MAX_ROOMS] [-v]
```

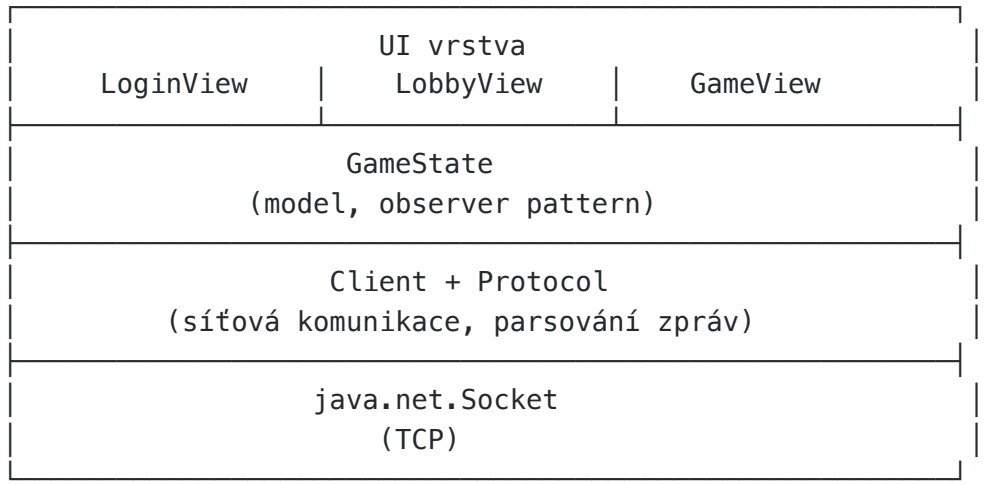
Parametr	Výchozí	Popis
-a	0.0.0.0	IP adresa pro bind
-p	10000	Port
-c	50	Maximální počet klientů
-r	10	Maximální počet místností
-v	false	Verbose režim (stdout místo souboru)

4. Implementace klienta

4.1 Struktura projektu

```
client_src/
├── build.xml            # Apache Ant build script
├── lib/                 # JavaFX knihovny (staženy automaticky)
└── src/main/java/nim/
    ├── Main.java        # Vstupní bod JavaFX aplikace
    ├── network/
    │   ├── Client.java  # Síťová vrstva, reconnect
    │   └── Protocol.java # Parsování a tvorba zpráv
    ├── game/
    │   └── GameState.java # Model stavu hry (Observer)
    ├── ui/
    │   ├── LoginView.java   # Přihlašovací obrazovka
    │   ├── LobbyView.java   # Lobby s místnostmi
    │   ├── GameView.java    # Herní obrazovka
    │   └── Components.java  # Sdílené UI komponenty a styly
    └── util/
        └── Logger.java  # Logování
```

4.2 Rozvrstvení aplikace



4.3 Dekompozice tříd

Main.java

- Entry point JavaFX aplikace (`Application.launch()`)
- Nastavení hlavního okna (Stage)

Client.java (480 řádků)

- Správa TCP socketu
- Background thread pro příjem zpráv (`ExecutorService`)
- Ping scheduler (`ScheduledExecutorService`)
- Automatický reconnect s exponenciálním backoff
- Callbacky: `messageHandler` , `connectionStateHandler` , `disconnectHandler`

Protocol.java

- `ParsedMessage` - parsovaná zpráva s typem a parametry
- `MessageType` enum - všechny typy zpráv
- `ErrorCode` enum - chybové kódy
- `parse()` - parsování příchozí zprávy
- `createLogin()` , `createTake()` , ... - tvorba odchozích zpráv

GameState.java

- Centrální model stavu aplikace
- `Phase` enum: DISCONNECTED, CONNECTING, LOGIN, LOBBY, IN_ROOM_WAITING, IN_GAME, GAME_OVER
- `OpponentStatus` enum: CONNECTED, DISCONNECTED, RECONNECTED
- Observer pattern - `setStateChangeListener()`

LoginView.java

- Formulář: IP adresa, port, přezdívka
- Validace vstupů před odesláním
- Zobrazení chybových hlášek

LobbyView.java

- Tabulka místností (TableView)
- Tlačítka: Vytvořit, Připojit, Obnovit
- Zobrazení stavu připojení

GameView.java

- Vizualizace kamínků (FlowPane s animovanými kruhy)
- Spinner pro výběr počtu + tlačítka Vzít/Přeskočit
- Indikace tahu, stavu protihráče, konce hry

Components.java

- Sdílené styly (barvy, fonty)
- Factory metody pro tlačítka, labely, pole
- Jemná barevná paleta (easy on the eyes)

4.4 Použité knihovny

Knihovna	Verze	Účel
JavaFX	21.0.1	GUI framework
java.net.Socket	JDK	TCP komunikace
java.util.concurrent	JDK	Vlákna, schedulery

Poznámka: Žádné externí knihovny pro síťovou komunikaci - pouze standardní BSD sockety přes `java.net.Socket`.

4.5 Non-blocking UI

Síťová komunikace probíhá v odděleném vlákně, aby UI nezamrzalo:

```
// Příjem zpráv v background threadu
receiverThread.submit(() -> {
    while (connected) {
        String line = reader.readLine();  // Blokující čtení
        if (line != null) {
            Platform.runLater(() -> processMessage(line));  // UI thread
```

```
            }
        }
    });

    // Ping scheduler
    pingScheduler.scheduleAtFixedRate(() -> {
        if (connected && !waitingForPong) {
            send(Protocol.createPing());
            lastPingTime = System.currentTimeMillis();
            waitingForPong = true;
        } else if (waitingForPong) {
            long elapsed = System.currentTimeMillis() - lastPingTime;
            if (elapsed > PONG_TIMEOUT) {
                handleConnectionLost();
            }
        }
    }, PING_INTERVAL, PING_INTERVAL / 2, TimeUnit.MILLISECONDS);
```

5. Požadavky a překlad

5.1 Požadavky na server

Požadavek	Verze
OS	Linux (GNU/Linux)
Kompilátor	GCC 9+ s podporou C11
Knihovny	POSIX (pthread, socket)

5.2 Požadavky na klienta

Požadavek	Verze
OS	Linux, Windows, macOS
Java	JDK 17+ (doporučeno 21)
Apache Ant	1.10+
Síť	Pro stažení JavaFX při buildu

5.3 Překlad serveru

```
cd server_src

# Release build (optimalizace)
```

```
make

# Debug build (s debug symboly pro valgrind/gdb)
make debug

# Vyčištění
make clean
```

5.4 Překlad klienta

```
cd client_src

# Kompilace (automaticky stáhne JavaFX při prvním spuštění)
ant compile

# Spuštění
ant run

# Vytvoření JAR
ant jar

# Vyčištění
ant clean
```

5.5 Spuštění

Server:

```
# Základní spuštění
./nim_server -p 10000

# S verbose logováním (výstup do terminálu)
./nim_server -p 10000 -v

# Plná konfigurace
./nim_server -a 0.0.0.0 -p 10000 -c 50 -r 10 -v
```

Klient:

```
# Přes Ant
ant run

# Nebo JAR přímo
java --module-path lib --add-modules javafx.controls,javafx.fxml -jar build/nim-client
```

5.6 Instalace na školních PC (UC-326)

```
# Aktualizace systému
sudo apt update

# Server – nástroje pro kompilaci C
sudo apt install build-essential gcc make

# Klient – Java a Ant
sudo apt install openjdk-17-jdk ant

# Volitelné – debugging nástroje
sudo apt install valgrind gdb

# Ověření instalace
gcc --version      # GCC 9+
java -version      # Java 17+
ant -version       # Ant 1.10+
```

6. Závěr

6.1 Dosažené výsledky

- ✅ Plně funkční síťová hra Nim pro dva hráče
- ✅ Stabilní server schopný obsluhovat více her současně
- ✅ Moderní GUI klient s příjemným designem
- ✅ Robustní protokol s validací a ošetřením chyb
- ✅ Podpora reconnectu při výpadku spojení
- ✅ Ochrana proti útokům (binární data, flood, timeout)
- ✅ Žádné memory leaky (ověřeno Valgrindem)

6.2 Testování

Aplikace byla testována na následující scénáře:

Scénář	Výsledek
Standardní průběh hry	✅ OK
Simulace výpadku klienta	✅ Reconnect funguje
Simulace výpadku serveru	✅ Klient se vrátí na login
Binární data (/dev/urandom)	✅ Server nepadá, odpojí klienta

Scénář	Výsledek
Zprávy ve špatném stavu	✅ Vrací chybový kód
Neplatné tahy	✅ Validace funguje
Valgrind memory check	✅ 0 errors, 0 leaks
InTCPtor (fragmentace, zpoždění)	✅ Aplikace funguje

6.3 Možná rozšíření

- Registrace hráčů s heslem a persistentní účty
- Historie her a statistiky vítězství
- Turnajový režim pro více hráčů
- Konfigurovatelný počet kamínků při vytváření místnosti
- Chatovací funkce během hry
- Podpora více typů her (varianty Nim)

Dokumentace vytvořena pro KIV/UPS, ZČU v Plzni, 2025