FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Semestrální práce

Karetní hra prší pro dva hráče realizovaná nad TCP

Jakub Vokoun

PLZEŇ

2026

Obsah

Zadání 1

Pro karetní hru prší, v konkrétních pravidlech: nedá se přebíjet sedmička ani eso, měnič mění pouze na svojí barvu, naprogramujte server v jazyce C nebo C++ spustitelný běžnými nástroji na OS GNU/Linux. Vytvořte klienta v jiném jazyce (zvolen Python) jako přenositelnou aplikaci, která podporuje GNU/Linux i MS Windows.

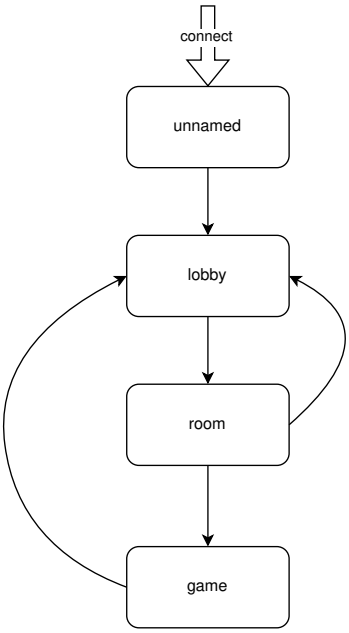
Síťová hra pro více hráčů s architekturou 1:N (server - klienti) fungující nad protokolem TCP, za použití standardních BSD socketů. Cílem semestrální práce je fungující aplikace s dobrým návrhem protokolu a jeho implementací.

Protokol 2

Komunikace mezi serverem a klientem není řízená celou dobu z jedné strany. Samotná hra je iniciovaná a řízená serverem, pohyb klienta po serveru je řízen klientem.

Klient se může v jednu chvíli nacházet pouze v jednom stavu. Stavy jsou následující:

- unnamed: Nepojmenovaný klient, pouze připojen na server.
- lobby: Pojmenovaný klient, který se může připojit do místností.
- room: Pojmenovaný klient v místnosti, čeká na její zaplnění.
- game: Pojmenovaný klient v místnost hraje/dohrál hru.



Obrázek 2.1: Průchod klienta serverem.

Každá existující místnost se musí nacházet v jednom ze stavů: OPEN, PLAYING, FINISHED. Přičemž stav OPEN znamená, že je volné místo pro dalšího hráče, stav PLAYING že probíhá hra, a stav FINISHED, že hra byla ukončena, ale ještě nejméně jeden klient neopustil místnost.

Zprávy v protokolu mají proměnnou délku, v závislosti jak na typu zprávy, tak i na přenášených datech. Protokol používá lidsky čitelná anglická slova psaná velkými písmeny. Jednotlivé části zprávy jsou odděleny mezerou, zprávy jsou od sebe odděleny znakem |. Každá zpráva začíná tzv. magickem PRSI. Protokol vypadá následovně:

Formát zprávy:

PRSI ... |

Tabulka 2.1: Popis veškerých zpráv v protokolu. Pokud je zpráva variabilní, část z ní se opakuje v závislosti na něčem, je tato opakující se část uzavřena mezi symboly # (které se v protokolu nevyužívají). Statické části zpráv jsou psány velkými písmeny, proměnné malými.

zpráva	**povolené stavy**	**popis**
Zprávy iniciované klientem		
NAME string	unnamed	Klient si zvolí libovolné jméno.
OK NAME	unnamed	Server potvrdí výběr jména.
LIST_ROOMS	lobby	Klient prosí server o seznam místností.
ROOMS count # id room-state #	lobby	Server posílá seznam místností, každá má svůj číselný identifikátor (int) a stav (string).
JOIN_ROOM id	lobby	Klient žádá o připojení do místnosti.
OK JOIN_ROOM	lobby	Server potvrzuje připojení do místnosti.
FAIL JOIN_ROOM	lobby	Server nemohl přiřadit klienta do místnosti (byla plná, neexistovala).
CREATE_ROOM	lobby	Klient se pokouší vytvořit místnost.
OK CREATE_ROOM	lobby	Server vytvořil místnost a přiřadil do ní klienta.
FAIL CREATE_ROOM	lobby	Server nemohl vytvořit místnost (dosažen limit místností).
ROOM_INFO	room/game	Klient žádá podrobnější informace o místnosti, ve které se nachází.

(tabulka pokračuje na další stránce)

Tabulka 2.1 *(pokračování z předchozí stránky)*

zpráva	**povolené stavy**	**popis**
ROOM id room-state PLAYERS count # name state #	room/game	Server posílá informace o místnosti.
LEAVE_ROOM	room/game	Klient opouští místnost.
OK LEAVE_ROOM	room/game (/lobby)	Server potvrzuje opuštění místnosti. Klientská aplikace tou dobou už může být v lobby, v závislosti na implementaci.
Zprávy iniciované serverem		
GAME_START	room	Server upozorňuje všechny hráče v místnosti, že začíná hra.
OK GAME_START	game	Klient potvrzuje serveru, že začíná hra.
HAND count # suitrank #	game	Server pošle hráči jeho karty, vždy ve formátu H7 například.
OK HAND	game	Klient potvzuje přijetí ruky.
TURN name TOP suitrank	game	Server oznamuje že je nové kolo, kdo je na tahu a jaká je svrchní karta.
OK TURN	game	Klient potvzuje přijetí TURN zprávy.
PLAY suitrank	game	Klient, který je na tahu posílá, jak chce hrát.
OK PLAY	game	Server potvrzuje klientovi jeho tah. Ekvivalentem pro draw je zpráva CARDS.
DRAW	game	Klient, který je na tahu, si líže.
PLAYED name suitrank	game	Server oznamuje, že hráč zahrál kartu - a jakou.
OK PLAYED	game	Klient potvzuje zprávu PLAYED.
DRAWED name count	game	Server oznamuje, že hráč si líznul počet karet.
OK DRAWED	game	Klient potvzuje zprávu DRAWED.
SKIP name	game	Server oznamuje, že hráč je přeskočen (kvůli kartě eso).
OK SKIP	game	Klient potvrzuje zprávu SKIP.
CARDS count # suitrank #	game	Server posílá hráči karty, které si líznul.
OK CARDS	game	Klient potvrzuje, že si líznul karty.

(tabulka pokračuje na další stránce)

Tabulka 2.1 *(pokračování z předchozí stránky)*

zpráva	povolené stavy	popis
WIN	game	Server oznamuje hráči, že vyhrál danou hru.
LOSE	game	Server oznamuje hráči, že prohrál danou hru.
OK WIN	game	Klient potvzuje zprávu WIN.
OK LOSE	game	Klient potvzuje zprávu LOSE.
Režijní zprávy		
PING	-	Server se dotazuje klienta, zda stále žije.
PONG	-	Klient odpovídá serveru, že stále žije.
SLEEP name	room/game	Server oznamuje hráčům v místnosti, že jiný hráč je dočasně nedostupný.
OK SLEEP	room/game	Klient potvruje zprávu SLEEP.
AWAKE name	room/game	Server oznamuje hráčům v místnosti, že hráč, který byl dočasně nedostupný, je opět online.
OK AWAKE	room/game	Klient potvrzuje zprávu AWAKE.
DEAD name	room/game	Server oznamuje hráčům v místnosti, že jiný hráč byl odpojen z důvodu nedostupnosti.
OK DEAD	room/game	Klient potvrzuje zprávu DEAD.
STATE	-	Klient se dotazuje serveru, ve kterém stavu se nachází. Užitečné po reconnectu.
STATE UNKNOWN	-	Server odpovídá na zprávu STATE, došlo k chybě a stav klienta je neznámý = třeba manuální reconnect.
STATE UNNAMED	-	Server odpovídá na zprávu STATE, klient se nachází ve stavu unnamed.
STATE LOBBY	-	Server odpovídá na zprávu STATE, klient se nachází ve stavu lobby.

(tabulka pokračuje na další stránce)

Tabulka 2.1 (*pokračování z předchozí stránky*)

zpráva	**povolené stavy**	**popis**
STATE ROOM _	-	Server odpovídá na zprávu STATE, klient se nachází ve stavu room. Na místo podtržítka dosaďte přesný formát zprávy ROOM.
STATE GAME _	-	Server odpovídá na zprávu STATE, klient se nachází ve stavu game. Na místo podtržítka dosaďte přesný formát zpráv ROOM, HAND, TURN.

Poznámky k protokolu:

- Výběr jména nemůže neuspět, jelikož pokud si klient zvolí jméno stejné, jako již existující hráč, server předpokládá změnu koncového zařízení klienta a hráči nastaví nový FD, tedy přiřadí hráče novému socketu.
- Zprávy ze serveru o stavu klienta mohou být velice dlouhé, zvlášť pokud je ve hře. Vyjde to ale levněji, než posílat tři zprávy zvlášť, kvůli magicu a oddělovači.
- Na jméno hráče nejsou kladeny žádné požadavky, kromě toho, že nesmí obsahovat oddělovač (|) a bílé znaky. To také znamená, že dlouhé jméno hráče může významně zpomalit přenos. (teoreticky) Útočníci s dlouhým jménem by mohli naschvál způsobovat krátkou nedostupnost a opět se připojovat a tím saturovat síť oběti, protože obě tyto zprávy obsahují jméno klienta.

Technická dokumentace implementace

3

3.1 Server

Třída config zajišťuje načtení proměnných z konfiguračního souboru a výchozí hodnoty. Třída logger nabízí možnost vypsat jakoukoliv zprávu i podle její vážnosti (error, info, warn).

Třídy room, card a player se starají hlavně o herní logiku a tolik se nezajímají o síť. Není to dokonalá dekompozice, v playerovi jsou uloženy důležité informace (např. čas posledního pingu) a některé pomocné funkce, ale ty jsou zpravidla volány ze třídy server. Původní plán byl rozdělit třídu player na třídy player a client, což mělo lepší dekompozici, ale nedokázal jsem to pořádně uchopit, tak je to dohromady.

Hlavní operativní třídou je server, pomocná pro ní je protocol. Protocol vytváří zprávy k odeslání centralizovaně. Server při inicializaci vytvoří epoll, který se kontroluje ve funkci run.

Vstupním bodem je funkce main, která vytvoří server a naplní ho daty z configu.

3.2 Klient

Klient je implementován jako python modul, v adresáři `prsi`. Je volán z vnějšku, ze souboru `run.py`. Základem je třída Client, která vlastní Ui a Net. Aplikace funguje na dvou vláknech: Ui a Net.

Pro Ui je použit tKinter, nad kterým je vystavěno uživatelské rozhraní se všemi tlačítky apod. Hrací karty jsou velmi odlišné od standardních (kreslila je kamarádka), ale mají stejné funkce jako standardní; načítají se pomocí třídy Img_Loader, která je ve třídě Ui.

Komunikace mezi ui a net třídami je zajištěna dvěma způsoby: třída net si sice vytáří nové vlákno na síťové poslouchání, ale její metody se dají normálně volat - takže například posílání síťových zpráv jde jednoduše. Komunikace směrem ze

sítě (z vlákna) do klienta/ui je přes frontu zpráv, do které vlákno přidává a klient pravidelně kontroluje.

V souboru `config.py` se nachází konstanty definující jak UI (barvy, velikosti, fonty), tak i síťovou komunikaci (konkrétní názvy stavů, zprávy, magic a oddělovač). V souboru `common.py` se nachází definice tříd Card, Player, Room. Kromě toho se tu vyskytuje i třída Client_Dummy, která existuje pouze kvůli rekurzivnímu vkládání - tato třída je proto v dědičnosti rodičovskou třídě Client, která implemetuje konkrétní funkcionalitu.

Uživatelská příručka 4

Server by mělo být možné spustit na libovolné GNU/Linux zařízení, kde je možné přeložit a spustit C++ 23, a kam může být připojena knihovna fmt. Pokud by byl problém s knihovnou, tím že je použitá pouze pro logger můžete třídu Logger upravit a o mnoho nepřijdete.

Kompilace/spuštění serveru:

Výpis 4.1: Ukázka kompilace a spuštění serveru.

```
user@machine: /$  mkdir -p build
user@machine: /$  cmake -S . -B build
...
user@machine: /$  cmake --build build
...
user@machine: /$  ./bin/ups
Server started.
...
```

Server lze nakonfigurovat v textovém souboru, který bude programu předaný při spuštění. Formát souboru je: jedna konfigurační proměnná na řádek, mezerou oddělená od hodnoty. Pokud se proměnná vyskytuje v souboru vícekrát, směrodatný je její poslední výskyt.

Tabulka 4.1: Seznam všech konfiguračních proměnných pro server.

proměnná	**výchozí**	**popis**
IP string	0.0.0.0	Na jaké IP server poslouchá.
PORT int	3750	Na jakém portu server naslouchá.
MC int	10	Maximální počet klientů.
MR int	10	Maximální počet místností.

(tabulka pokračuje na další stránce)

Tabulka 4.1 *(pokračování z předchozí stránky)*

proměnná	výchozí	popis
	nedoporučeno upravovat	
EME int	32	Epoll max events.
ET int	500	Epoll timeout [ms]. Za kolik ms přestane být epoll blokující.
PT int	2.000	Frekvence posílání pingu v ms.
ST int	5.000	Kolik ms bez pingu znamená, že je klient dočasně nedostupný.
DT int	180.000	Kolik ms bez pingu, než je klient prohlášen za nedostupného.

Klientská aplikace je napsána v jazyku Python 3 a měla by být spustitelná na jakémkoliv zařízení s interpretem pythonu v dané verzi a knihovnami pillow a typing_extensions. Doporučený postup je použít virtuální prostředí. Ukázkový skript pro GNU/Linux Debian:

Výpis 4.2: Ukázka instalace závislostí a spuštění klienta ve virtuálním prostředí venv.

```
user@machine: /$  sudo apt install python3-venv
user@machine: /$  python3 -m venv .venv
user@machine: /$  source .venv/bin/activate
(.venv) user@machine: /$  pip install pillow typing_extensions
...
(.venv) user@machine: /$  python3 run.py
...
```

Na MS Windows použijte ekvivalentní postup.

Závěr 5

Výsledná aplikace je dost robustní na problémy se sítí i uživatelem. Na začátku projektu jsem se snažil o dokonalou architekturu serveru, ale nakonec to pro mě bylo moc složité a raději jsem server navrhl jednoduše, ale pro mě srozumitelně. Teď, po jeho dokončení, bych už nejspíš byl schopen dosáhnout vyšší abstrakce a stále rozumět. Hlavní nevýhoda aktuální implementace je svázanost klienta-hráče se socketem. Ideálně bych měl odděleně hráče a klienty, přičemž pouze klient by byl svázán se socketem a hráč s maximálně jedním klientem.

Přišlo mi nepříjemné implementovat herní logiku na duplicitně ve dvou jazycích. Nevím, jestli existuje nějaké řešení, ale nechce se mi věřit, že by se takto opravdu operovalo mimo školu.

Musím se pochválit za využití epollu na serveru - podařilo se mi tím vyhnout se spoustě obtíží, které jsem slýchal od přátel.