



ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA APLIKOVANÝCH VĚD
KATEDRA INFORMATIKY A VÝPOČETNÍ TECHNIKY

SEMESTRÁLNÍ PRÁCE KIV/UPS

Síťová hra Oběšenec
za použití protokolu UDP

Jaroslav Klaus
A13B0347P
jklaus@students.zcu.cz
31. ledna 2016, Plzeň

Obsah

1	Zadání	2
2	Popis hry	2
3	Protokol	2
3.1	Typy zpráv a dat	3
4	Implementace	4
4.1	Server	5
4.2	Klient	6
5	Uživatelská příručka	6
6	Závěr	7

1 Zadání

Vytvořte hru, kterou může hrát více hráčů. Řešení musí obsahovat program serveru, který bude obsluhovat více klientů i her současně. Klient bude po opětovném přihlášení pokračovat tam, kde přestal. Program musí být odolný proti chybným zadáním a musí adekvátně reagovat na výpadky serveru i klientů. Klient bude naprogramován v jazyce *Java* a server v jazyce *C* nebo *C++*. Konkrétně vytvořte hru *Oběšenec* za použití protokolu *UDP*.

Program bude doplněn o zpracování statistických údajů (Počet odeslaných a přijatých zpráv a bytů, počet chybných zpráv, počet znovu odeslaných zpráv apod.). Dále nesmí chyby v komunikaci znemožnit případně znesnadnit používání grafického rozhraní klienta.

2 Popis hry

Hra *Oběšenec* je relativně populární a jednoduchá. Cílem hráčů je uhodnout slovo na základě znalosti jeho délky. Hráč, který je na řadě, hádá písmenko, které si myslí, že je obsaženo ve slově. Pokud má pravdu, hádá znovu, pokud ne, hádá další hráč v pořadí. Za každé špatně hádané písmeno se hráči přikreslí část obrázku šibenice s oběšencem (odtud název hry) a pokud se obrázek kompletní, znamená to prohru. Výhra znamená, že hráč uhodl poslední skryté písmenko a slovo je tak kompletní. Další možností je pokusit se uhádnout celé slovo. Pokud jej uhádneme, vyhráváme, pokud ne, prohráváme.

3 Protokol

Z důvodu použití protokolu *UDP* bude potřeba řešit možnost ztráty, případně duplicity zpráv, nebo jejich špatné pořadí a poškození dat během přenosu. To bude vyřešeno tak, že každá zpráva bude obsahovat pořadové číslo a bude se na něj vyžadovat odpověď. Možnost ztráty spojení bude vyřešena tak, že se zprávy budou posílat opakovaně, dokud nebudou potvrzeny, a pokud klient některou nepotvrdí do stanovené doby, bude prohlášen za odpojeného. Stejně tak pokud neodpoví server, bude prohlášen za nedostupný. Zároveň se bude jednou za čas posílat zpráva typu *“ping”* proto, aby se zamezilo výpadku spojení během tahu, tedy v době, kdy se žádné zprávy nezasílají. Možnost poškození dat během přenosu bude vyřešena pomocí *checksumy*.

Komunikační protokol bude textový s oddělovačem ‘;’. Bude se skládat z pěti částí, viz tabulka 1. Data budou jako oddělovač používat ‘;’. Klient bude pro jeho identifikaci serverem posílat v každé zprávě své jméno.

Tabulka 1: Vzhled zprávy

číslo zprávy	typ zprávy	checksum	velikost dat	data
--------------	------------	----------	--------------	------

3.1 Typy zpráv a dat

Typ zprávy bude uveden číslem, u kterého bude dále popsán význam a naznačeno, jak budou vypadat data. Jako první v popisu bude uvedeno, zda bude paket od klienta na server (C), nebo od severu ke klientovi (S).

1. S, C, **potvrzení doručení paketu**
S: číslo potvrzovaného paketu (int)
C: jméno (string), číslo potvrzovaného paketu (int)
2. S, **ukončení hry z důvodu nedostupnosti hráče**
Data: jméno (string)
3. C, **žádost o připojení**
Data: jméno (string), počet soupeřů (int)
4. S, **odpověď na žádost o připojení**
Data: 0 pokud vše proběhlo v pořádku, 1 pokud je jméno ve hře a hraje, 2 pokud je jméno ve hře a je odpojené
5. C, **žádost o navázání přerušené hry**
Data: jméno (string)
6. S, **odpověď na žádost o navázání přerušené hry**
Data: stav hry = stav hry (int, pokud -1, tak znamená, že hráč nebyl v žádné hře nalezen), počet hráčů (int), stav hádaného slova (string), již hádaná písmena (string), počet špatně hádaných písmen (int)
7. S, **začátek hry**
Data: počet hádaných písmen (int)
8. C, **odpojuji se**
Data: jméno (string)
9. S, **oznámení o někom, kdo se odpojil**
Data: jméno (string)
10. S, **prohrál jsi**
Data: -

11. S, **oznámení o někom, kdo prohrál**
Data: jméno (string)
12. S, **vyhrál jsi** (= konec hry, uhodl slovo)
Data: -
13. S, **oznámení o někom, kdo vyhrál**
Data: jméno (string)
14. S, **jsi na tahu**
Data: -
15. S, **oznámení, kdo je na tahu** (pro ostatní hráče)
Data: jméno (string)
16. C, **tah**
Data: jméno (string), hádané písmeno (char)
17. S, **odpověď na tah**
Data: pozice s uhodnutými písmeny (string 0 a 1, kde 1 znamená shodu)
18. S, **oznámení tahu**
Data: jméno (string), písmeno (char), pozice s uhodnutými písmeny (string 0 a 1, kde 1 znamená shodu)
19. C, **hádám celé slovo**
Data: jméno (string), slovo (string)
20. S, **oznámení hádání slova**
Data: jméno (string), slovo (string)
21. S, C, **ping**
S: -
C: jméno (string)
22. S, **oznámení o klientovi, který chvíli neodpovídá**
Data: jméno (string)

4 Implementace

Zde je popsána dekompozice, rozvrstvení, metoda paralelizace a další implementační záležitosti programů.

4.1 Server

Program serveru je rozdělen do několika částí a postup zpracování zpráv je založený na vzoru producent – konzument. První část je soubor *main.c*, ve kterém probíhá kontrola počtu zadaných parametrů a volají se funkce pro vytvoření socketu, bind socketu a spuštění hlavní části serveru, které jsou implementované v souboru *server.c*. Při vytvoření socketu se kontroluje, zda je zadaná adresa platná a port v rozmezí 1 až 65535.

V hlavní části server inicializuje některé potřebné proměnné a struktury a spustí vlákna konzumenta (ta, která zpracovávají příchozí zprávy) a jedno vlákno, které periodicky posílá zprávy typu “*ping*” všem klientům. Poté přejde do nekonečné smyčky, ve které se vždy zablokuje při čekání za příchozí zprávu. Pokud zpráva dorazí, server se ji pokusí rozparsovat do struktury *struct message* – zjistí pořadové číslo, typ zprávy, checksum, velikost dat, data a jméno hráče, který zprávu poslal. Pokud se mu to podaří, přidá zprávu do seznamu *struct list* a upozorní konzumenty na novou zprávu. Pokud ne, tak byla zpráva při přenosu poškozená, případně nastal timeout. Ještě je zkontrolován seznam odeslaných zpráv, ve kterém čekají zprávy na potvrzení. Pokud se nalezne zpráva, která nebyla potvrzena po určitou dobu, odešle se znovu a pokud nebude ani po několika pokusech potvrzena, klient bude prohlášen za nedostupného a hra bude ukončena.

Další část programu obsahuje soubor *communication.c*, který implementuje funkce podstatné jak pro komunikaci (např. odeslání zprávy, výpočet checksumy apod.), tak pro zpracování příchozích zpráv (např. kontrolu checksumy, kontrolu pořadového čísla, reakce na zprávy apod.). Vlákno konzumenta čeká blokováno na semaforu, dokud není v seznamu příchozích nějaká zpráva. Tu ze seznamu vyjme a zpracuje. Nejprve zkontroluje checksumu, poté pořadové číslo zprávy, pokud vše souhlasí, odešle potvrzení a na zprávu zareaguje dle protokolu výše.

Soubor *game.c* obsahuje funkce pro práci se strukturami *struct player* a *struct game*, které uchovávají informace o hráčích a hrách. Mezi tyto funkce patří např. vytvoření nového hráče, přiřazení hráče do hry, vytvoření nové hry, zrušení hry, kontrola tahu hráče atd.

Soubor *list.c* obsahuje funkce pro práci se seznamy zpráv, jako přidání zprávy do seznamu, vyjmutí zprávy ze seznamu, nalezení a vyjmutí zprávy po zpracování potvrzení. Posledním souborem je *structures.h*, který obsahuje definice používaných datových struktur.

4.2 Klient

Postup práce klienta je téměř identický jako ten na serveru. Liší se např. v tom, že zde existuje pouze jedno vlákno pro zpracování zpráv. Další vlákno slouží pro příjem zpráv, jiné pro odesílání "pingu" a poslední k ovládání GUI.

První částí je soubor *Main.java*, který zkontroluje počet argumentů a vytvoří okno klienta. To je implementováno v souboru *Window.java* a po jeho vytvoření se spustí výše zmíněná vlákna. Stav hry se zobrazuje pomocí textových upozornění v okně hry, které také obsahuje tzv. plátno (definované v *Canvas.java*), ve kterém se vykresluje příslušná šibenice s oběšencem.

Soubor *Connection.java* uchovává informace o spojení a obsahuje metody pro komunikaci, jako např. odeslání a příjem zprávy.

Receiver.java představuje vlákno pro příjem zpráv. Jeho práce je téměř identická s tou, kterou provádí hlavní vlákno serveru – čeká na příchozí zprávu, kterou přidá do seznamu, zkontroluje a případně znovu odešle nepotvrzené zprávy a v případě dlouho nepotvrzené zprávy oznámí nedostupnost serveru.

Soubor *ProcessMessage* reprezentuje vlákno konzumenta, které čeká nad semaforem a po přijetí zprávy ji zpracuje a reaguje na ni dle protokolu (viz výše). Průběh zpracování zprávy je opět téměř symetrický se serverovou stranou.

Dále jsou zde dva soubory *Message.java* a *Game.java*, které slouží jako datové struktury.

5 Uživatelská příručka

Server je napsán v jazyce *C* v operačním systému Linux a je k němu vytvořen příslušný *Makefile*, kterým je možné server pomocí nástroje *make* jednoduše přeložit. K přeložení je potřebná knihovna *pthread* pro možnost vytvoření a práce s vlákny apod. Server se spouští se dvěma parametry, adresou pro naslouchání a portem, např.

```
./server INADDR_ANY 10000
```

Ukončení serveru je možné provést zasláním signálu *SIGINT* (stisknutí Ctrl + C).

Klient je napsán v jazyce *Java* za použití *JDK 6* a fungovat by tedy měl jak v OS Linux, tak Microsoft Windows. Pro jeho přeložení je tedy potřeba *JDK 6* nebo vyšší, pro spuštění by měl stačit *JRE 6* nebo vyšší. Příložen je soubor *build.xml* pro automatický překlad nástrojem *ant*, který se provede např. příkazem

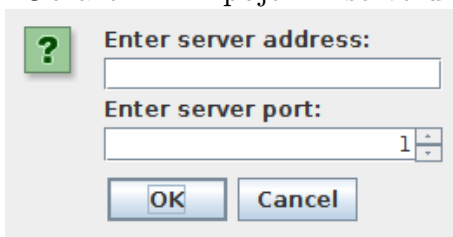
```
ant jar
```

Vytvoří se soubor *Hangman.jar*, který lze spustit např. příkazem
`java -jar Hangman.jar`

Po spuštění je požadováno zadání adresy a portu serveru, viz obrázek 1.

Po spuštění vypadá klient jako na obrázku 2. Novou hru vytvoříme z nabídky menu *Game*, *New Game* a do dialogu 3 zadáme svou přezdívku a počet protihráčů. Server nás automaticky přiřadí do hry se stejným počtem protihráčů, případně takovou hru vytvoří. Hra se poté ovládá tak, že pokud jste na tahu, tak hádání provedete tak, že daný znak stisknete na klávesnici. Pokud chcete hádat celé slovo, stiskněte klávesu *Enter*. Pokud se ze hry odpojíte, máte v blízké době šanci se do ní opět vrátit a to pomocí nabídky menu *Game*, *Reconnect*, viz 4, kam zadáte svou původní přezdívku a budete vráceni do hry.

Obrázek 1: Připojení k serveru



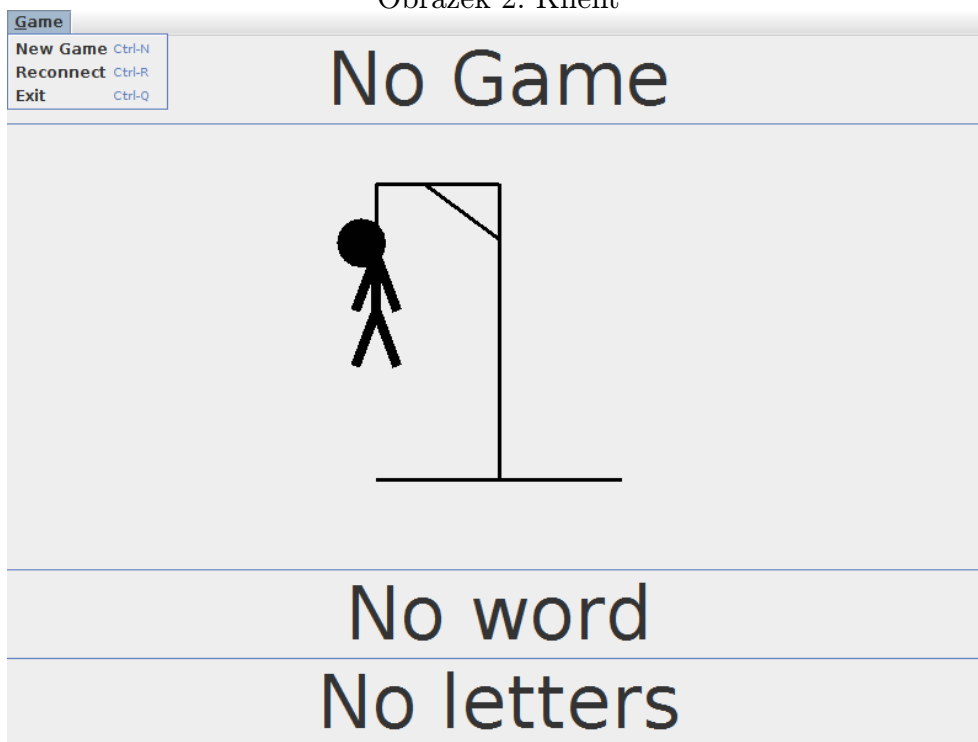
The image shows a standard Java Swing dialog box. On the left is a green square icon with a white question mark. To its right, the text 'Enter server address:' is followed by a single-line text input field. Below that, the text 'Enter server port:' is followed by a text input field containing the number '1' and a small spinner control. At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

6 Závěr

Vytvoření hry bylo mnohonásobně těžší, než jsem očekával. Zabrало mi relativně velkou spoustu času, cca 80 až 120 hodin. Jedním z důvodů bylo to, že nebylo jednoduché správně implementovat komunikaci pomocí UDP protokolu a také to, že jsem si při návrhu nedokázal uvědomit všechny možné problémy a situace, které mohou nastat.

Navrhl jsem komunikační protokol a vzhled zpráv, implementoval a otestoval program klienta i serveru. Zadání se mi tedy podařilo splnit.

Obrázek 2: Klient



Obrázek 3: Nová hra

The dialog box features a green question mark icon in the top-left corner. It contains two input fields: 'Enter nickname:' followed by a text box, and 'Enter number of opponents:' followed by a spin box currently set to '1'. At the bottom, there are 'OK' and 'Cancel' buttons.

Obrázek 4: Reconnect

The dialog box features a green question mark icon in the top-left corner. It contains one input field: 'Enter nickname:' followed by a text box. At the bottom, there are 'OK' and 'Cancel' buttons.