

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

KIV/UPS

UDP - Piškvorky

Autor: Lukáš Haringer, A13B0306P

Akademický rok: 2015/2016

Zadání

Témata semestrálních prací

Téma semestrální práce si vybírá student sám, poté je konzultuje s cvičícím. Doporučuji vybrat si nějakou netriviální hru, kterou může hrát pomocí počítače více hráčů (dva a více). Řešení musí obsahovat program serveru, který bude obsluhovat více klientů - hráčů i her současně a bude schopen uchovávat stav hry. Klient bude po opětovném přihlášení pokračovat tam, kde přestal. Programové vybavení musí být odolné proti chybným zadáním a musí adekvátně reagovat na výpadky serveru i klientů.

Zásady vypracování semestrální práce

- Úlohu naprogramujte v programovacím jazyku C anebo Java. Pokud se jedná o úlohu server/klient, pak klient bude v Javě a server v C.
- Výstupy serveru budou v alfanumerické podobě, klient může komunikovat i v grafice (není podmínkou).
- Server řešte pod operačním systémem Linux, klient může běžet pod OS Windows XP. Emulátory typu Cygwin nebudou podporovány.
- Realizujte **konkurentní (paralelní)** servery. Server musí být schopen obsluhovat požadavky více klientů souběžně.
- V případě použití nespojovaných služeb (UDP) vyřešte na úrovni aplikačního protokolu problematiku ztráty příp. duplicity dat (např. číslování, metoda okénka, apod.).
- Součástí programu bude trasování komunikace, dovolující zachytit proces komunikace na úrovni aplikačního protokolu a zápis trasování do souboru.
- Každý program bude doplněn o zpracování statistických údajů (přenesený počet bytů, přenesený počet zpráv, počet navázaných spojení, počet přenosů zrušených pro chybu, doba běhu apod.).
- Zdrojové kódy organizujte tak, aby od sebe byly odděleny části volání komunikačních funkcí, které jste vytvořili na základě zadání, od částí určených k demonstraci funkčnosti vašeho řešení (grafické rozhraní).

Zvolená hra

Pro vypracování své semestrální práce jsem si zvolil známou hru piškvorky.

Pravidla hry

Piškvorky jsou strategická hra, ve které spolu soupeří dva hráči. Nejčastěji se hraje na čtverečkovaném papíře, na kterém se hráči střídají v kreslení křížku/kolečka. Vyhrává hráč, který jako první vytvoří nepřerušenu řadu pěti svých značek. V profesionálních pravidlech hry je dále takzvaný swap, který brání začínajícímu hráči získat výhodu. Většina lidí s ním však není obeznámena a docházelo by ke zmatení hráčů, proto jsem se ho rozhodl neimplementovat.

Analýza problému

V první řadě bylo potřeba se rozhodnout zda logika hry bude na serveru nebo tahy budou vyhodnocovat sami klienti a server bude sloužit jen jako prostředník pro přeposílání zpráv. Z důvodu požadavku na schopnost serveru uchovat stav hry jsem se rozhodl, že vyhodnocovat zprávy bude server. Tím bude mít server kompletní kontrolu nad hrou a zabráni podvádění klientů. Server bude nutné rozdělit do několika místností, aby si hráči mohli sami takovou místnost vytvořit se spoluhráčem, se kterým chtějí hru hrát. Pro obsluhu více klientů najednou jsem se rozhodl používat více vláken. Proto je pro každého nově připojeného klienta vytvořeno nové vlákno.

Popis protokolu

Pro komunikaci mezi serverem a klientem jsem se rozhodl použít znakově orientovaný protokol. Není tak efektivní jako bitově orientovaný protokol, ale je mnohem lépe čitelný pro člověka což pomáhá při debugingu. Frekvence zpráv mezi serverem a klientem není vysoká a zprávy jsou většinou krátké, takže nám nižší efektivnost nevadí. Každá zpráva má jako první sekvenční číslo, které nám umožňuje odhalit ztrátu datagramu. Další část zprávy obsahuje znak, který příjemci říká jaká akce je potřeba vykonat. Na oddělení jednotlivých částí zpráv je použit znak |. Po odeslání každé zprávy je čekáno na acknowledge zprávu p jako potvrzení, že předchozí zpráva dorazila.

Typy zpráv od klienta serveru

- | | |
|---|--|
| l | žádost o lognutí na server |
| a | vyžádá seznam otevřených her |
| b | založí hru |
| c | připojí do hry s identifikačním číslem 1 |
| d | zruší založenou hru |
| e | vzdá hru |
| f | žádost klienta o odpojení od serveru, pokud je ve hře tak ji vzdá |
| g | oznámí tah na pozici x y |
| v | po přijetí paketu se neprovádí žádná akce, slouží pouze k vyžádání ack paketu pro potvrzení, že klient je stále připojen |
| p | potvrzuje přijetí paketu číslo 0, na přijetí tohoto paketu se čeká po odeslání každé zprávy. Pokud nepřijde je klient odpojen. |
| r | vyžaduje znovu připojení do hry po pádu klienta nebo internetu |

Zprávy od serveru pro klienta

a	seznam otevřených her (1 otevřená hra hráče Hari)
b	partie založena úspěšně/neúspěšně
c	připojení do hry proběhlo úspěšně/neúspěšně
g	tah na pozici x y byl/nebyl zanesen do hrací plochy
0	určuje s jakými znaky kdo hraje a kdo začíná
k	konec hry
l	soupeř provedl tah na pozici x, y
h	slouží pro obnovení hry po pádu, paket obsahuje s jakými znaky hráč hraje, zda je na tahu a stav hracího pole
v	po přijetí paketu se neprovádí žádná akce, slouží pouze k vyžádání ack paketu pro potvrzení, že klient je stále připojen
p	potvrzuje přijetí paketu, na přijetí tohoto paketu se čeká po odeslání každé zprávy. Pokud nepřijde je klient odpojen. Ack paket je ve tvaru 0 p <sequence number potvrzovaneho>
q	server končí činnost

Příklady užití zpráv

Lognutí do hry

Klient zašle:

<sequence number>|1|jmeno

Server zašle ack packet pokud bylo lognutí úspěšné.

Reconnect po pádu hry

Klient zašle:

<sequence number>|r|jméno

Server odpoví ack paketem a zašle:

<sequence number>|h|0 nebo 1 (určuje symbol se kterým hráč hraje|0 nebo 1(určuje zda je hráč na tahu)|posloupnost znaků určující stav hrací plochy (0 kolečko, 1 křížek, 2 prázdné)

Vyžádání seznamu založených her

Klient zašle:

<sequence number>|a

Server odpoví:

<sequence number>|a|2(počet založených her)|1(id hry)|Hari(jméno zakládajícího)|2|Pepa

Založení hry

Klient zašle:

<sequence number>|b

Server odpoví:

<sequence number>|b|1 nebo 0 (úspěšně neúspěšně)

Připojení do hry

Klient zašle:

<sequence number>|c|1(id hry)|

Server obou klientům nejdříve zašle:

<sequence number>|c|1|jméno protihráče|

Poté zašle:

1 klientovi: Klient má kolečka.

<sequence number>|i|0|

2 klientovi: Klient má křížky a začíná.

<sequence number>|i|1|

Průběh hry

Klient zašle: Tah na pozici x=5 y=3

<sequence number>|g|5|3|

Server odpoví: Tah na pozici x=5 y=3 byl úspěšný/neúspěšný.

<sequence number>|g|1 nebo 0|5|3|

Server poté pošle druhému klientovi:

<sequence number>|l|5|3|

Čímž oznámí tah protihráče, a že je klient na tahu.

Konec hry

Klient zašle:

<sequence number>|e (vzdá hru)

Nebo

Server zašle

<sequence number>|k|kód konce

Kódy konce:

- 1 soupeř hru vzdal
- 2 ty jsi hru vzdal a proto jsi prohrál
- 3 regulérně jsi prohrál
- 4 regulérně jsi vyhrál
- 5 remíza

Implementace klienta

Jelikož klient slouží pouze k zobrazování zpráv od serveru, je velice jednoduchý. Skládá se ze dvou částí GUI a UDP klienta pro přijímání a odesílání zpráv.

Popis tříd klienta

Main.java - Hlavní třída programu. Načte údaje pro přihlášení k serveru a vytvoří instance všech ostatních tříd.

UdpClient.java – Nejdůležitější třída klienta. Stará se o odesílání zpráv, přijímání zpráv a vyvolává reakce na příchozí zprávy.

IUdpClient.java - Rozhraní třídy UdpClient.

Game.java – Třída pro ukládání aktuálního stavu hry.

GameWindow.java – Třída pro vytvoření okna hrací plochy.

Control.java – Třída pro vytvoření okna pro vytvoření a připojení do hry.

GameButton.java – Tlačítko, které reprezentuje jedno políčko hrací plochy. Ukládá informace o svém stavu a pozici.

Implementace serveru

Vlákna serveru

Main vlákno – Spustěno se startem programu. Zkontroluje parametry spustí receiveThread a čeká vstup pro ukončení serveru.

receiveThread – Příjmací vlákno. Přijímá všechny zprávy, které serveru přijdou a umísťuje je do spojového seznamu message.

connectThread – Když přijde zpráva z neznáme adresy receiveThread spustí connectThread, které rozhodne zda klienta připojí a spustí mu clientThread nebo odmítne.

clientThread – Vlákno pro obsluhu zpráv od jednoho klienta. Postupně vybírá přijaté zprávy ze spojového seznamu message a zpracovává je.

Struktury dat

Server používá tři druhy struktury dat. Message, player a game. Ve všech případech se jedná o spojové seznamy pro uchovávání dat.

Message

```
typedef struct message {
    /*text zpravy*/
    char *buf[1024];
    /*adresa zpravy*/
    struct sockaddr_in clientaddr;
    /*delka adresy zpravy*/
    int clientlen;
    /*odkaz na dalsi zpravu*/
    struct message *nextMessage;
} Message;
```

Player

```
typedef struct player {
    /*Adresa hrace*/
    struct sockaddr_in adresa;
    /*nick hrace*/
    char *nick;
    /*stav hrace*/
    int stav;
    /*indikator zda je pripojen*/
    int odpojen;
    /*zablokovani hrace pro odesilani*/
    int blocked;
    /*jeho index ve structure game*/
    int indexHrace;
    /*odkaz na struc game, ve ktere se nachazi*/
    struct game *partie;
    /*pocet prijatych zprav od toho klienta*/
    int packetNumberRecv;
    /*pocet odeslanych zprav pro tohoto klienta*/
    int packetNumberSent;
    /*cas posledniho kontaktu*/
    time_t checked;
    /*odkaz na dalsiho hrace v seznamu*/
    struct player *dalsiHrac;
} Player;
```

Game

```
typedef struct game {
    /*id hry*/
    int idGame;
    pthread_mutex_t zamekPartie;
    // 0 = prazdna, 1 - jen zakladajici hrac, 2 - oba hraci, 3
- probiha hra, 4 - konec;
    int stavGame;
    Player *tahneHrac;
    /*pole hraci plochy, 0 - kolecko, 1 krizek, 2 prazdne
pole*/
    char hraciPlocha[16][16];
    /*pocet zaplnenych policek*/
    int zaplnenoPolicek;
    /*odkaz na hrace ve hre*/
    Player *hraci[2];
    /*odkaz na dalsi hru v seznamu*/
    struct game *dalsiPartie;
} Game;
```

Popis modulů serveru

Main.c – Hlavní modul aplikace, zajišťuje síťovou komunikaci a spouští metody pro obsluhu přijatých zpráv.

Procces.c – Modul s metodami pro obsluhu přijatých zpráv.

Player.c – Modul se spojovým seznamem připojených hráčů.

Game.c – Modul se spojovým seznamem aktivních her.

Statistics.c – Modul pro vytvoření statistik síťové komunikace serveru.

winCheck.c – Modul s metodami pro ověření zda daným tahem neskončila hra.

Uživatelská dokumentace

Překlad serveru

Překlad serveru se provede pomocí programu make. Make Program make je součástí distribucí GCC (ve Windows MinGW, Cygwin). Překlad spustíme ve složce c_src příkazem:

make -f Makefile

Spuštění serveru

Server se spouští příkazem **./server <volitelná IP adresa> <volitelné číslo portu>** Při nezadání vstupních parametrů jsou použity defaultní hodnoty. Pokud program spustíme s parametrem **-h** vypíše se nápověda pro spuštění serveru.

Obsluha serveru

Během běhu server je možno vypsát statistiku serveru příkazem **S** a nebo ukončit jeho činnost příkazem **Q**.

Překlad klienta

Překlad klienta se provede pomocí programu **Apache Ant** (<http://ant.apache.org/>). Pro překlad a spuštění programu je potřeba nainstalovat java JRE 1.6 a java JDK 1.6. Po jeho instalaci stačí ve složce java_src spustit příkaz:

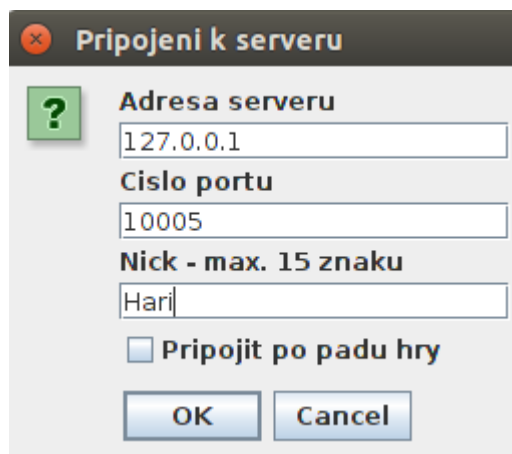
ant

Spuštění klienta


Klienta lze spustit dvojklikem na jeho ikonu nebo příkazem **java -jar piškvorky.jar**

Obsluha klienta

Do hry je potřeba se přihlásit vyplněním následujících údajů.



Připojení k serveru

 **Adresa serveru**
127.0.0.1

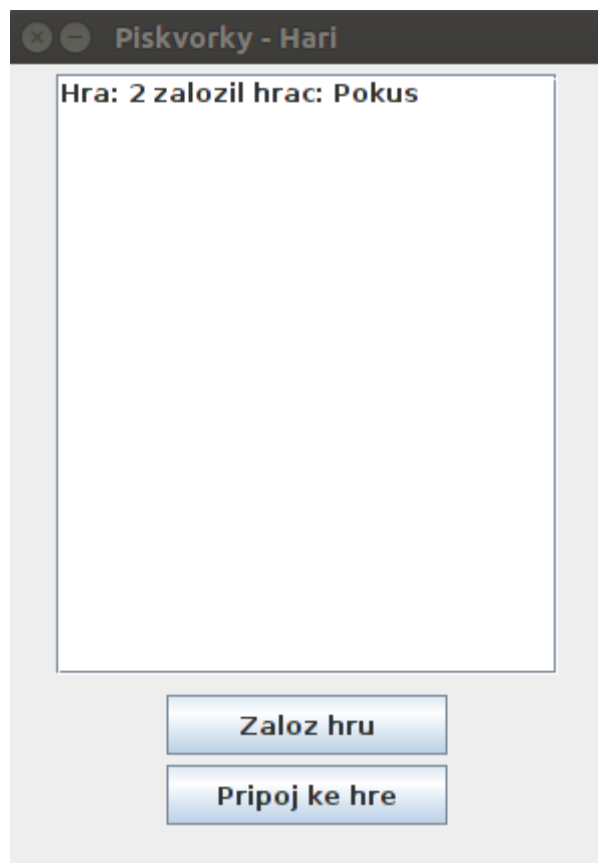
Cislo portu
10005

Nick - max. 15 znaku
Hari

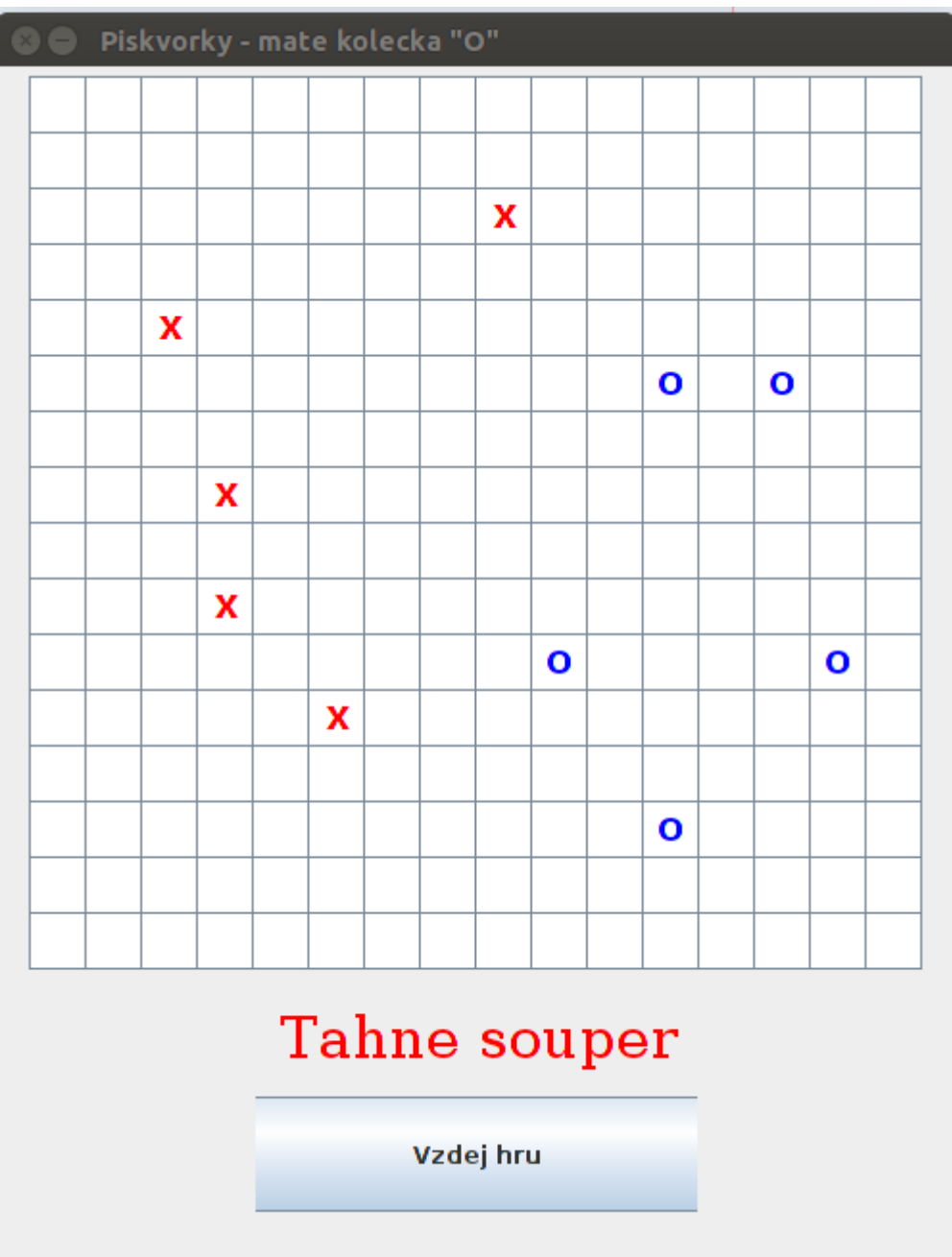
☐ Připojit po padu hry

OK Cancel

Po přihlášení na server lze založit novou hru nebo se připojit k nějaké vytvořené.



Po začátku hry lze kliknutím zvolit políčko pro tah nebo tlačítkem **Vzdej hru** hru vzdát.



Závěr

Tvorbou semestrální práce jsem se naučil základy síťové komunikace v jazycích Java a C, se kterou jsem se dosud při programování nesetkal. Server i klient pracují stabilně a jsou odolní proti výpadkům spojení. Zadání jsem splnil v plném rozsahu.