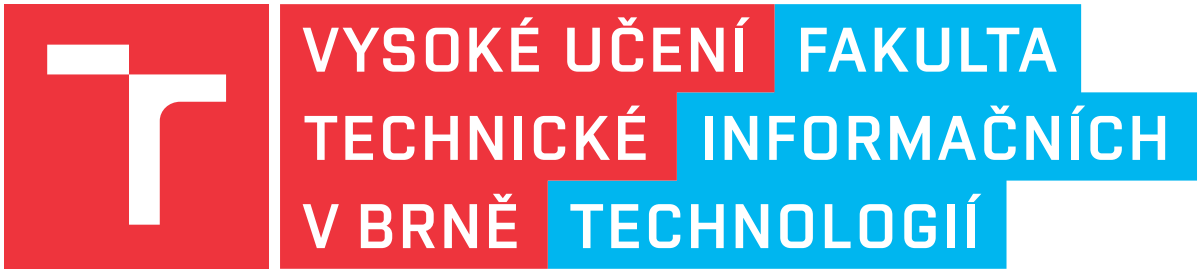
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Počítačové komunikace a sítě – 1. projekt

Klient-server pro jednoduchý přenos souborů

5. března 2018

Vladan Kudláč

Obsah

1 Úvod

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně. Dokumentace, uživatelská příručka a vestavěná nápověda je psána v češtině. Programová dokumentace je stejně jako samotný kód psána v angličtině a není součástí odevzdané dokumentace.

Cílem projektu bylo navrhnout vlastní aplikační protokol pro přenos souborů a naprogramovat klientskou a serverovou aplikaci využívající tento protokol.

2 Aplikační protokol

Aplikační protokol předpokládá spolehlivou komunikaci (TCP), správnost přenosu již nekontroluje.

Poté, co klient naváže se serverem spojení, zašle klient serveru požadavek v jednom ze dvou následujících tvarů:

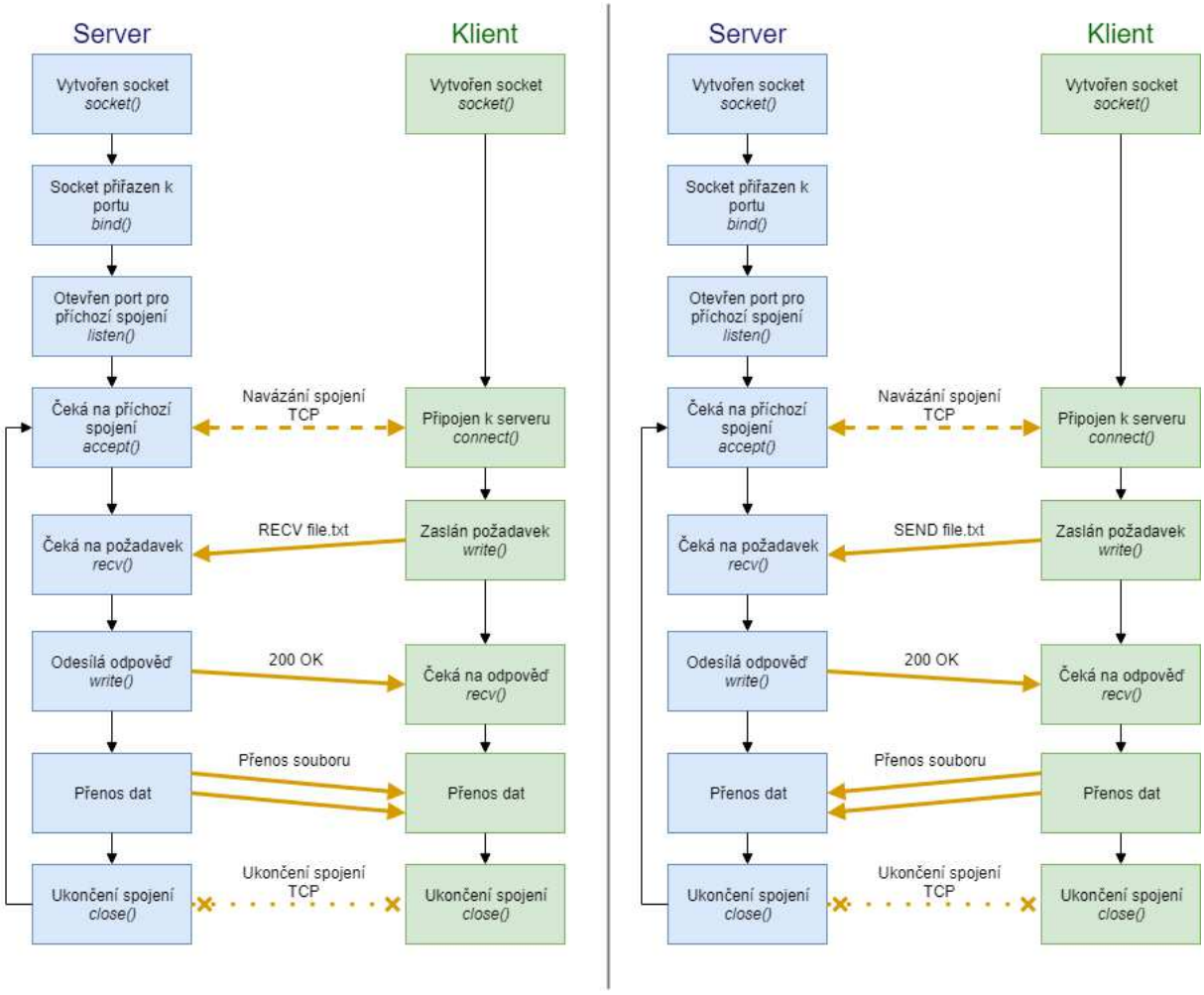
- **`SEND <filename>`** – soubor filename bude nahrán na server (klient->server)
- **`RECV <filename>`** – soubor filename bude stažen ze serveru (server->klient)

Zpráva obsahuje **klíčové slovo** `SEND` nebo `RECV`, **mezeru** (ASCII kód 32) a následuje **posloupnost libovolných znaků** určující název souboru. Server klientu odpoví jedním ze stavových kódů:

- **`200 OK`** – vše v pořádku, přenos může začít
- **`403 FILE_ERROR`** – klient žádá o práci se souborem, který nelze otevřít
- **`400 BAD_REQUEST`** – klient nesplnil předepsaný tvar požadavku

Odpovědi jsou inspirované stavovými kódy aplikačního protokolu HTTP. Jakákoliv jiná odpověď než `200 OK` je považována za chybu a k přenosu nedojde. Chybu je vhodné uživateli vypsat.

Pokud server vrátí kód 200, může být zahájen přenos. Od této chvíle je dohodnuto, která strana bude příjemcem a která odesílatelem, komunikace se stává jednosměrnou. Odesílatel odesílá data v binární podobě, příjemce data pouze přijímá. Jakmile odesílatel odešle všechna data, ukončí spojení.



Obrázek 1: Konečný automat aplikačního protokolu. Vlevo stahování ze serveru, vpravo nahrávání na server.

3 Programové řešení

Výsledný program je implementovaný v jazyce C++ dle standardu *C++11* a není zpětně kompatibilní s jazykem C, předpokládají se pouze adresy IPv4. Zdrojové kódy jsou členěny do tří souborů: `ipk-client.cpp`, `ipk-server.cpp` obsahující zdrovové kódy a `ipk-shared.cpp` obsahující společné vlastnosti – závislosti ipk-client a ipk-sever, makra definující konstanty, používaný jmenný prostor.

Server zvládá vyřizovat souběžně více požadavků. Pro každý příchozí požadavek vytvoří hlavní řídicí proces nový proces, který je zodpovědný za zpracování požadavku. Hlavní proces samotné požadavky nezpracovává. Pokud uživatel server ukončí signálem *SIGINT*, dojde k ukončení hlavního procesu, který přijímá nové požadavky, procesy s rozpracovanými požadavky ukončeny nejsou, nechají se, aby přenos dokončili.

Cílový i zdrojový adresář musí existovat. V případě, že neexistuje, končí program chybou, program adresáře nevytváří.

4 Uživatelská příručka

IPK: projekt č. 1 – klient-server pro jednoduchý přenos souborů. Verze 0.3 (5.3.2018).

4.1 Požadavky

Překladač

Program lze přeložit v překladači podporující standard *C++11*. Doporučuje se překladač ***gcc* verze 4.8.4 a novější**. Pro jiné překladače a starší verze nebyl program testován.

Knihovny potřebné k překladu

- iostream
- fstream
- unistd.h
- cstring
- sys/types.h
- sys/socket.h
- netinet/in.h
- netdb.h

4.2 Překlad

Překlad lze provést programem *make*. Pro přeložení serveru i klienta použijte příkaz **`make all`** v adresáři s projektem. Pokud potřebujete provést překlad klienta nebo serveru jednotlivě, lze použít příkaz `make ipk-client`, případně `make server-ipk`. Pokud není možné použít program *make*, lze programy přeložit následujícími příkazy:

```
g++ -std=c++11 ipk-client.cpp -o ipk-client
g++ -std=c++11 ipk-server.cpp -o ipk-server
```

4.3 Spuštění

ipk-client

```
./ipk-client -h host -p port [-r|-w] soubor
```

- `-h <host>` – (IP adresa nebo fully-qualified DNS name) identifikace serveru jakožto koncového bodu komunikace klienta
- `-p <port>` – cílové číslo portu v intervalu 0-65535
- `-r <soubor>` – klient stáhne soubor ze serveru
- `-w <soubor>` – klient nahraje soubor na server

ipk-server

```
./ipk-server -p <port>
```

- `-p <port>` – číslo portu v intervalu <0;65535>, na kterém server naslouchá

Při použití čísla portu menší než 1024 mohou být pro spuštění vyžadována administrátorská oprávnění. Server se ukončuje signálem *SIGINT* (Ctrl+C).

4.4 Chybové kódy

- 0 – program skončil v pořádku
- 1 – chyba při zpracování argumentu
- 2 – chyba síťového rozhraní (např: nelze vytvořit socket, přiřadit port, vytvořit spojení)
- 3 – chyba práce se soubory
- 4 – systémová chyba (selhal příkaz fork) – pouze u ipk-server
- 5 – chyba komunikace (klient zaslal neplatný požadavek)

5 Závěr

Program byl testován na referenčním virtuálním stroji pro předmět ISA, na serveru Merlin i Eva. Kromě nevytváření složek na straně příjemce není znám žádný problém. Aplikační protokol je jednoduchý, efektivní, ale význam zpráv je i přesto zřejmý a pro člověka čitelný. Jedná se o můj první program využívající síťové komunikace, takže jsem si osvojil mnoho nových znalostí. Projekt byl náročný, ale zajímavý.