Dokumentácia

Vlastná implementácia jednoduchého P2P protokolu „MatfyzPeer“

1. Stručný popis

Jedná sa o jednoduchý interaktívny P2P program, ktorý sa ovláda cez príkazový riadok. Interne program beží na vlastnom peer-to-peer protokole. Program dokáže seedovať alebo leechovať súbory a je plne automatický (uživateľ sa iba napojí na sieť a už nič nerieši).

2. Presný popis

Program je rozdelený do 2 hlavných častí. P2P a Tracker. Tracker je server, ktorý si pamatá ku každému súboru množinu IP adries ktore na ňom participujú. Bez trackeru by jednotlivci o sebe nemali ako vedieť. Tracker je jediná centrálna autorita vrámci tejto siete. Všetko ostatné je decentralizované. Tracker počúva na nejakom porte (osobne som používal port 5900) a pri pripojení nejakého node k nemu si automaticky spraví záznam a jemu vráti množinu adries k danému súboru.
V momente čo sa odpojí, tak ho z tabuľky vymaže.

Druhá, primárna časť je balíček p2p. Keďže každý uzol pripojený do siete je zároveň klient a aj server, je takto navrhnutý aj tento podprogram. Rozdelený je na dva časti, kde časť serverová je multithreadová aby vedela obsluhovať viacerých klientov naraz. Klientská časť je iba jednovláknová, pretože sťahovanie je lineárne (nesťahujem od viacerých seederov naraz, ale len od jedného). Na sťahovanie sa používa tzv. *.mff (metainfo file), ktorý obsahuje tagy potrebné na stiahnutie už konkrétneho reálneho súboru. Obsahuje adresu a port trackera, meno súboru, veľkosť jedného bloku (ktoré sa budú posielať potom medzi jednotlivými uzlami), veľkosť celého súboru a samozrejme najdôležitejšiu časť, hash súboru. Keď máme tento súbor, tak s jeho pomocou sa vieme napojiť na tracker, ktorý nám už následne vráti IP adresy ktoré participujú na danom súbore. Ak som seeder, tak ich všetky pingnem a dám im vedieť, že je tu nový seeder. Ak som leecher, tak budem postupne skúšať nájsť voľného seedera. Jeden uzol vie participovať iba na jednom súbore naraz (okrem iného aj kvôli tomu, že používam iba jeden port na komunikáciu; dá sa zmeniť, no konkrétne ja 34999).

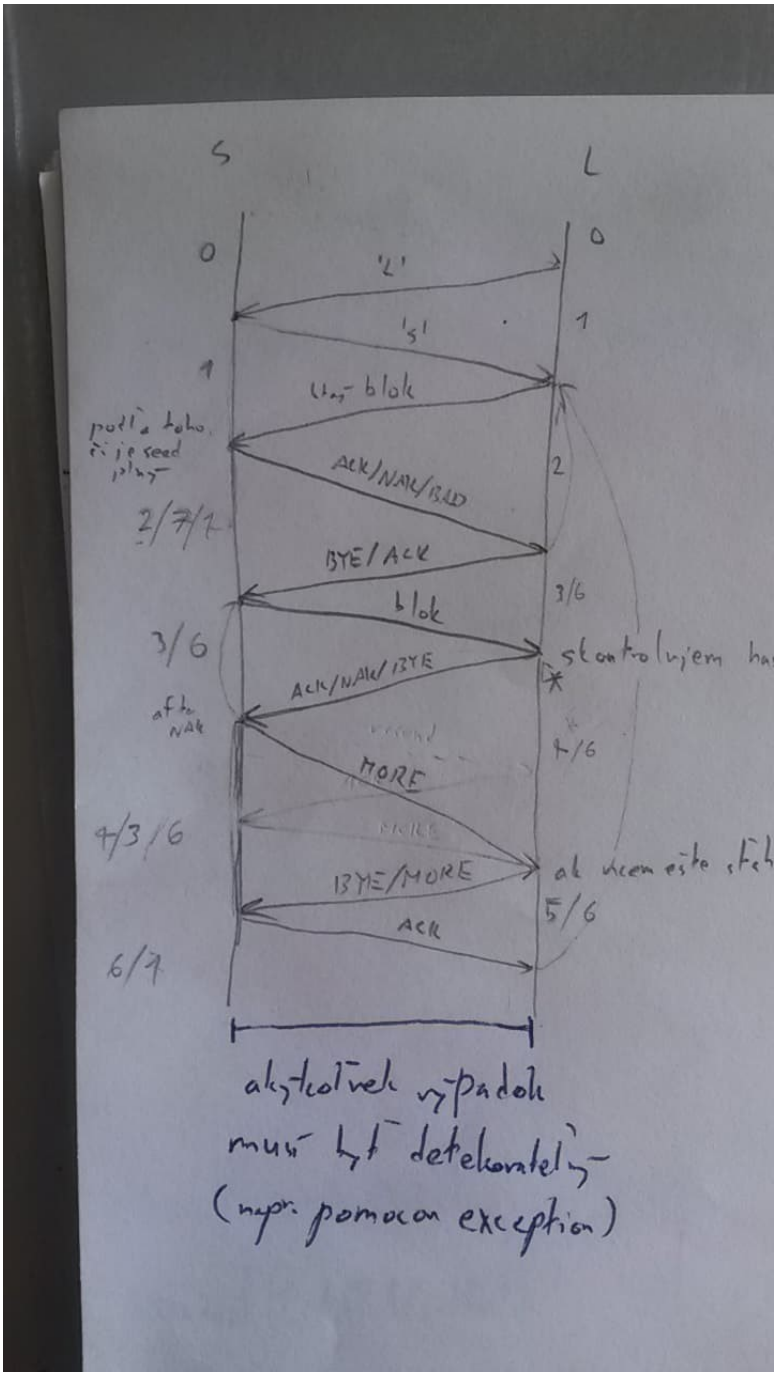
Program sa obsluhuje interaktívne pomocou príkazového riadku.

3. Popis protokolu

Keďže tu prebieha komunikácia medzi serverom a klientom, bolo treba vymyslieť nejaký protokol na komunikáciu medzi nimi. Keďže je to mnou vymyslený vlastný protokol, tak to nie je nič zložité, bolo to len pracné to vymsliet.

Najprv vyriešime patologické prípady a to L-L, S-S, L-S (na ľavo je server, na pravo klient; L je leech a S je seed). Ak L kontaktuje L, tak sa nič nedeje, lebo ani jeden z nich nie je oprávnený posielať data. Ak sa kontaktuje S a S, tak je to tiež vlastne jedno, pretože data netreba zas naopak ani jednému. Potom, ak S ako client kontaktuje L ako seed, tak si leecher uloží IP seedera a spojenie sa ukončí.

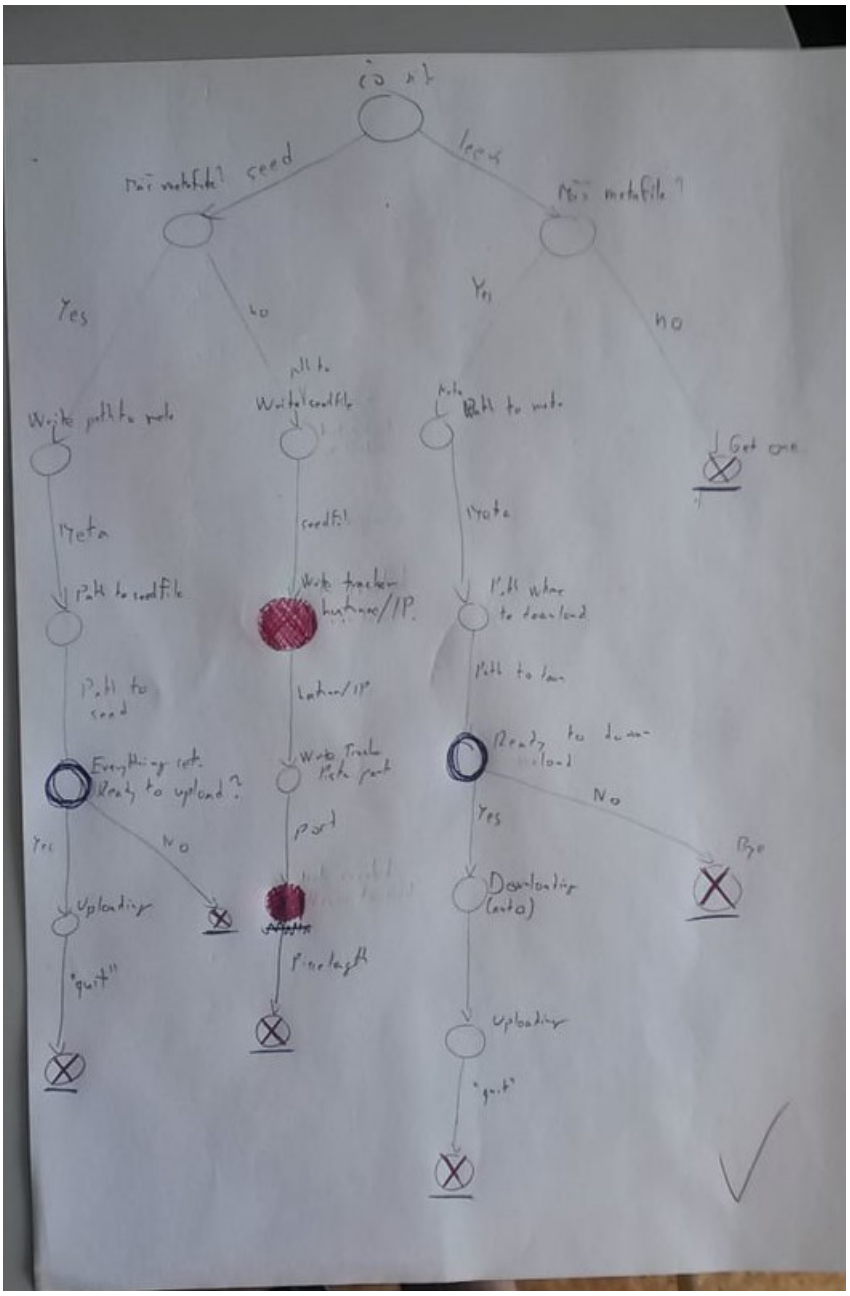
Najdôležitejší je prípad S – L. Komunikácia prebieha následovne:



Klient so serverom (leech a seed) si teda posielajú medzi sebou krátke správy, prípadne nejake bloky dát (byte[], používame serializáciu) a na základe toho prebieha komunikácia a protokol funguje.

4. Ovládanie programu

Keďže bol program zamýšľaný ako nejaký P2P filetransfer, tak bolo mojím cieľom spraviť ho čo uživateľsky najprívetivejší a zároveň nerozbitný. Uživateľské rozhranie bolo navrhnuté podľa stavového automatu, a teda v ňom je zahrnuté všetko a nič by ho nemalo rozbiť:



Rozhranie je celkom intuitívne, ak sa človek riadi pokynmi na obrazovke, tak by nemal byť žiaden problém, preto si myslím, že to bližšie netreba popisovať. Možno akurát by som spomenul, že kto nemá žiaden *.mff súbor, tak si ho môže vytvroriť.

5. Možnosti vylepšenia

Keďže P2P protokoly sú vďačná téma, je tu toho čo zlepšovať stále. Osobne chcem tento projekt rozšíriť a vylepšiť ako zápočťák na pokročilú Javu. Minimálne by som chcel dorobiť to, aby posielať data išlo aj medzi L-L, teda že si dvaja leecheri môžu medzi sebou vymieňať data ak jeden má také ktoré chce druhý. Potom by som sa chcel viac hĺbkovo pozrieť na to, ako to aplikovať na WAN nie iba na LAN (pretože tento teraz som testoval iba na LAN a neviem práve, ako by sa to správalo po internete). Ďalší veľmi dobrý upgrade by bol nevyužívať žiaden tracker, ale fungovať čisto decentralizovane iba na základe nejakých zdieľaných HashTables. S využitím pokročilých metód v Jave si myslím, že by sa z toho nakoniec mohol vykľuť aj celkom pekný osobný projekt do nejakého portfólia na GitHub ☺ a to by nebolo na škodu.

6. Záver

Tento zápočtový program hodnotím veľmi pozitívne, pretože som sa tu naučil veľmi veľa nových vecí; hlavne som si precvičil programovanie v Jave čo bolo primárny účel toho zápočtoveho programu. Musím však skonštatovať, že pracovať so socketami v Jave je trošku otravné a nepríjemné. Avšak vďaka tomu som sa ale konečne našiel v sietiach a pochopil, ako to funguje... protokol, port, socket, klient, server, tcp, udp už viac niesú pre mňa cudzie pojmy ktoré som vedel, že akurát niečo znamenajú v networkingu, ale konečne som si v nich spravil poriadok a už tomu rozumiem, čo s čím ako funguje a ako spolu interagujú tieto jednotlivé networking komponenty. Zaujalo ma to natoľko, že aj keď to bolo pomerne náročné zadanie na pochopenie a vymyslenie, tak to chcem rozšíriť ďalej a pokračovať s tým na zápočťák na pokročilú Javu.

Vyhotovil: Matúš Maďar