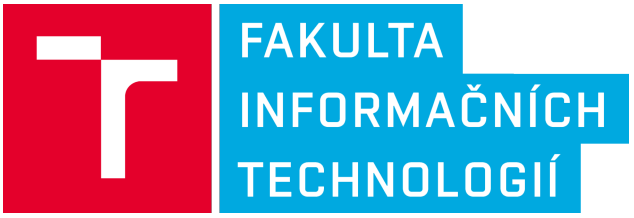

Vysoké učení technické v Brně

PDS

Přenos dat, počítačové sítě a protokoly

Dokumentace

Hybridní chatovací P2P síť

Autor:
Petr Buchal

Login:
xbucha02

28. dubna 2019

1 Úvod

Cílem projektu je implementovat několik separátních programů, které dohromady vytváří hybridní P2P síť umožňující chatování. Projekt jsem se rozhodl implementoval v Pythonu.

2 Implementace

Projekt jsem se rozhodl implementovat v jazyce Python. Pro bencoding využívám kód převzatý z [1]. Začal jsem implementací tříd pro zprávy, které si mezi sebou účastnící sítě posílají, ty nacházejí se v souboru *messages.py*. K třídám jsem vytvořil unit testy nacházející se v souboru *tests.py*. Soubor *tools.py* obsahuje metodu pro tisk na stderr a generování unikátního názvu roury. Následně popíšu 3 hlavní komponenty projektu.

2.1 RPC

RPC jsem implementoval pomocí pojmenovaných rour. Během volání *pds18-rpc.py* se pro konkrétního účastníka vytváří unikátní pojmenovaná roura na základě jeho typu (peer nebo uzel) a identifikátoru, na konci volaného programu se roura smaže. Každý účastník sítě nepřetržitě poslouchá na adrese své roury.

2.2 Peer

Peer obsahuje 3 vlákna, první vlákno se stará o poslouchání síťových zpráv. Zprávy, které nejsou peeru určeny a to jak konkrétně, tak i obecně jsou zahozeny. Ostatní zprávy jsou zpracovány na základě protokolu daného zadáním. Druhé vlákno se stará o naslouchání skrze pojmenovanou rouru. Pokud mu přijde pokyn, který není schopen zpracovat, zahodí jej, standardní zprávy zpracuje na základě zadání. Třetí vlákno se stará o hlídání timeoutů, tedy o včasné zasílání zpráv HELLO a ACK a případné zasílání zprávy ERROR, při nedoručení zprávy ACK.

2.3 Node

Implementace uzlu je velmi podobná implementaci peera. Rovněž obsahuje 3 vlákna, kde se první vlákno se stará o poslouchání síťových zpráv. Zprávy, které mu nejsou

určeny, jsou zahozeny. Ostatní zprávy zpracovává na základě protokolu daného zadáním. Druhé vlákno se stará o naslouchání skrze pojmenovanou rouru. Pokud mu přijde pokyn, který není schopen zpracovat, zahodí jej, standardní zprávy zpracuje na základě zadání. Třetí vlákno se stará o hlídání timeoutů, tedy o kontrolování doby uplynulé od zpráv UPDATE pro jiný uzel a HELLO pro registrované peery, včasné zasílání zpráv ACK, případné zasílání zprávy ERROR, při nedoručení zprávy ACK a vlastní zasílání zpráv UPDATE.

2.4 Technologie potřebné pro spuštění programu

- Standardní knihovny jazyka Python

3 Testování

V rámci verifikace jsem prováděl unitární testy tříd implementující typy zpráv, jak jsem již uvedl v sekci 2. Validační testování jsem provedl celkem se 4 lidmy. Nejvíce jsem projekt testoval s Richardem Hauerlandem (xhauer02, c++), jednalo se o vytvoření síťě skládající se z několika uzlů obou implementací a dále rovněž z několika peerů obou implementací, při čemž jsme zkoušeli, zdali se síť chová podle specifikace. Podobné testování v o něco menší míře jsem provedl s Janem Bartoňem (xbarto88, python). Následně jsme síť vytvořenou s Richardem Hauerlandem spojili se sítí vytvořenou s Janem Bartoňem a opět testovali její funkčnost. Kratší testování na větší síti obsahující cca 15 peerů se zhruba stejným počtem uzlů jsem provedl s Janem Zarskym (xzarsk03, python) a Jakubem Menšíkem (xmensi03, python). Jednalo se o zaslání vzájemné zaslání několika zpráv.

Reference

[1] Utku Demir. A simple bencode decoder-encoder library on pure python. https://bit.ly/2ZHgm4D.