Newspaper P2P sharing, programátorská dokumentácia

Jazyk

Program je naprogramovaný v jazyku C++ vo verzii `C++17`.

Beh programu

Beh programu pozostáva s niekoľkých častí. Prvou je vytvorenie nového Peera a, ak je to žiadané, aj jeho novín. Na tento účel slúži GUI, ktoré je naimplementované použitím knižnice Qt. Následne už môžeme pridať nejaký článok.

Články a ich správa

Články sú uložené v databázi novín, ktorej kľúč tvorí identifikátor novín. Počas behu programu nie sú načítané v pamäti, je ale načítaná ich absolútna cesta na pevnom disku, pričom sú práve tam uložené. Na štrukturovanie programu sa používa Markdown, s tým ráta aj naimplementovaný parser. Článok sa načíta do pamäte, len ak je potrebné ho niekam vypísať, alebo, ak je potrebné ho poslať po sieti nejakému inému peerovi.

P2P komunikácia

Princíp programu spočíva v peer-to-peer komunikácii, namiesto tradičnejšieho, klient-server modelu. To zabezpečí decentralizáciu, ktorá je cieľom tohto programu. Každý peer má jedinečný identifikátor a spolu s so svojou IP adresou to tvorí základ celej komunikácie. Každý peer má v sieťovaní časť, ktorá prijíma, ale aj posiela informácie. Jeden peer sa teda vie napojiť, pomocou TCP, na druhého peera a navzájom si zdieľať články. Na posielanie po sieti bolo potrebné nájsť spôsob, ako dáta účinne a čo najvšeobecnejšie (teda s čo najširšou podporou) serializovať. Pre tento účel sa používajú tzv. protocol buffers.

Serializácia po sieti

Pre posielanie dát po sieti je potrebná ich serializácia. Na to slúžia tzv. protocol buffers, alebo tiež protobuf. Takáto protobuf správa obsahuje všetky potrebné informácie na to, aby pri posielaní tá druhá strana vedela, aká infromácia k nej dorazila. Každá správa má svoj typ (teda ktorej časti programu sa týka) a svoj kontext (teda ktorej časti komunikácie sa týka). Základ tvorí jedna spoločná správa `Message` a tá následne obsahuje všetky ostatné správy. Každá správa obsahuje svojho prijímateľa a odosielateľa a je, samozrejme, šifrovaná.

Šifrovanie

Takmer každá (až na výnimky, ktoré sú spomenuté neskôr) správa je šifrovaná. Šifrovanie je symetrické a je teda potrebné, aby si obe strany pred tým, než započnú posielanie nejakých užitočných dát, vymenili symetrické kľúče. To sa deje automaticky a bez zásahu užívateľa, rovnako tak sú aj generované. Práve posielanie týchto symetrických kľúčov tvorí výnimku zo správ, ktoré sú šifrované. Symetrický kľúč je ale asymetricky zašifrovaný verejným kľúčom prijímateľa a podpísaný súkromným kľúčom odosielateľa. Ďalšiu výnimku tvoria aj správy obsahujúce verejné kľúče, k výmene ktorých dôjde hneď pri pridaní novín do databázy. Vlastný verejný a súkromný kľúč sa vygenerujú pri zakladaní nového peera, symetrický kľúč sa vygeneruje vtedy, ak sa zistí (počas odosielania), že pre daného peera ešte neexistuje. Vtedy sa odoslanie pôvodnej správy mierne oneskorí. Ako svoje vlastné kľúče, tak aj kľúče ostatných je potrebné niekam uložiť. Všetky tieto a podobné dáta je potrebné ukladať do rôznych dátových štruktúr.

Dátové štruktúry

V programe sa používajú viaceré dátové štruktúry, najčastejšie však `std::unordered_map` a `std::unordered_set`. Medzi najdôležitejšie mapy a množiny patria:

- Mapa novín, indexovaná pomocou verejného identifikátora novín (ich ID) a ukazujúca na dátovú štruktúru `NewspaperEntry` ktorá obsahuje zoznam stiahnutých článkov z daných novín, celkový zoznam článkov, autority novín a neskôr aj verejný kľúč novín.
- Mapa užívateľov, ktorá slúži ako záznam o užívateľovi. Obsahuje úroveň užívateľa a jeho verejný identifikátor (jeho ID).
- Mapa článkov, ktorá obsahuje ako hlavičku článku, tak aj zoznam ľudí, ktorí oznámili danému peerovi, že daný článok majú stiahnutý (`ArticleReaders`). To sa neskôr použije na nasmerovanie peera, ktorý nejaký článok hľadá, ale jeho pôvodný majiteľ ho už, napríklad, vymazal.
- Mapa kategórií, obsahujúca observer na `ArticleReaders`.

- V rámci triedy `Networking` trieda `IpMap` obsahujúca mapu indexovanú verejnými identifikátormi peerov a obsahujúcu verejný a symetrický kľúč daného peera.

Verejný kľúč vs. verejný identifikátor

Verejný kľúč je kľúč typu `CryptoPP::RSA::PublicKey` a je to verejný kľúč určený pre asymetrické šifrovanie. Verejný identifikátor je identifikátor peera v P2P sieti, ale nie je natoľko bezpečný a nemá všetky potrebné vlastnosti na to, aby mohol byť používaný pri šifrovaní. Na druhú stranu sa však lepšie ukladá a aj serializuje. Serializácia dátových štruktúr však nie je robená pomocou protocol buffers, ale pomocou iného serializačného frameworku od `boost::serialization`.

Serializácia údajov na disk

Ak sa program vypína, alebo ak o to užívateľ požiada, tak je potrebné dáta serializovať. Na vrchole hierarchie je štruktúra `ProgramContext`, ktorá sa vie serializovať a rovnako sa vedia serializovať všetky ostatné triedy v hierarchii. Následne sa pomocou `boost::archive` celý obsah zapíše do textového archívu a uloží na disk. Pri deserializácii naopak.

Triedy

Krátky popis jednotlivých dôležitých tried:

- `Article` obsahuje hlavičku článku, ako napr. meno autora, kategórie hlavný hash a pod.
- `IpMap` spravuje zoznam IP (IPv4 aj IPv6) adries a aj verejné kľúče ostatných peerov a aj jednotlivé symetrické kľúče.
- `IpWrapper` obsahuje metódy na pridávanie IP adries, verejných kľúčov a pod. IP adresy sú uložené pomocou `QHostAddress`.
- `Margins` trieda spravujúca pridávanie a odstraňovanie marginálií z článkov.
- `Networking` obsahuje metódy na posielanie a prijímanie správ. Na tieto účely slúžia najmä triedy `PeerReceiver` na prijímanie a `PeerSender` na odosielanie.
- `NewspaperEntry` obsahujúca jednotlivé položky typické pre jedny noviny, ako napríklad zoznam ich článkov či ich meno a verejný identifikátor.
- `Peer` obsahuje hlavne metódy na správu peera, ako generovanie správ na sieť, sieťovanie, zoznam IP adries, zoznam článkov a novín, je to v podstate vrchol hierarchie tohto programu.
- `StringHelpers` pomocné metódy pre prácu so stringami.

Vlákna

Program nie je explicitne členený na vlákna a ak sa nejaké vytvárajú, je to spôsobené používanými knižnicami, najmä knižnicou Qt, ktorá obsluhuje ako používateľské rozhranie, tak aj sieťovanie. Práve pri sieťovaní sa používa princíp, ktorý zavádza Qt pre každý objekt, ktorý dedí od objektu `QObject` a to je princíp signálov a slotov. Takýto prístup sa ukazuje ako postačujúci a neblokuje hlavnú slučku pre užívateľské rozhranie. Ak by bolo potrebné, je možné odložiť vykonávanie slotov na neskôr, rovnako, ak by sa ukázalo, že je potrebné program prispôsobiť pre viacvláknové prostredie, je tento mechanizmus signálov a slotov prispôsobiteľný a na to dostatočne robustný.

Ukladanie článkov, metadát a konfigurácie

Články

Články sa ukladajú každý samostatne a jedná sa o jednoduché súbory typu Markdown, alebo hocijaká ich podmnožina, napríklad plaintext. Program podporuje operačné systémy Windows a tie, ktoré majú jadro Linux (rôzne GNU/Linux distribúcie) a tam sa súbory obsahujúce články ukladajú na nasledujúce miesta:

- **Linux**
    - Do adresára `$XDG_DATA_HOME/news_p2p_sharing/Articles`, alebo, ak premenná `$XDG_DATA_HOME` nie je k dispozícii, tak:
    - V adresári `$HOME/.local/sharing/news_p2p_sharing/Articles`, kde sa predpokladá, že premenná `$HOME` je definovaná a to v čase prihlásenia sa užívateľa ukazujúc na jeho domovský adresár (často `/home/meno užívateľa`)
- **Windows**
    - Do adresára `%LOCALAPPDATA%/NewsP2PSharing/Articles`

Názvy súborov sa neprenášajú a vytvárajú sa na mieste. Pozostávajú z nadpisu, ktorý v prípade Markdownu tvorí prvý riadok začínajúci znakom `#` a hashom, vypočítaným pre celý článok použitím `std::hash` na objekte `std::string`.

Metadáta a konfigurácie

Metadáta spolu s konfiguráciami jednotlivých častí programu, ako napríklad Meno a ID Peera a jeho novín, sú serializované a následne uložené do súboru:

- **Linux**

        - `$XDG_DATA_HOME/news_p2p_sharing/settings.txt`, alebo, ak nie je premenná k dispozícii, tak:
        - `$HOME/.local/sharing/news_p2p_sharing/settings.txt`
- **Windows**
        - `%LOCALAPPDATA%/NewsP2PSharing/settings.txt`

Celý kontext programu (metadáta, stav, konfigurácie) je teda serializovaný do jedného súboru. Túto serializáciu vykonáva cudzia knižnica `boost::serialization`.

Cudzie knižnice

Serializácia

Serializácia údajov prebieha pomocou už spomínanej, cudzej knižnice `boost::serialization` za pomoci `boost::archive`. Sú to jediné dve súčasti z rodiny boost knižníc. Tieto knižnice zvládajú všetky druhy C++ STL dátových štruktúr počnúc `std::vector` skrz `std::unordered_set` až po `std::shared_ptr`.

Jediným problémom pri používaní tejto knižnice je ten, že nepodporuje iterátory, ktoré mnohé dátové štruktúry v C++ používajú. V dôsledku toho je zvýšený počet tzv. "raw" pointerov.

Ostatné dátové typy, ktoré serializačný framework nepozná, sú serializované prevedením na `std::string`. Napríklad verejné a súkromné RSA kľúče z knižnice Crypto++. Tieto sa serializujú tak, že sa najprv pomocou metódy `.DEREncode(...)` prevedú na `std::string` a následne sa archivujú, alebo naopak, pri deserializácii sa z `std::string` prevedú pomocou `.BERDecode(...)` naspäť na svoje pôvodné triedy.

Serializačný framework podporuje aj použitie viacerých verzií nejakej triedy, takže pri ďalšom vývoji by nemal nastať výraznejší problém so spätnou kompatibilitou.

Na serializáciu po sieti sa používa Google protocol buffers, ktoré majú širokú podporu.

Kryptografia

Ďalšou, veľmi dôležitou súčasťou je kryptografia. V programe sa používa na tento účel knižnica Crypto++, ktorá zvláda ako symetrické, tak asymetrické šifrovanie, podpisy a pod.

Na symetrické šifrovanie sa používa `CryptoPP::EAX< CryptoPP::AES>`, na asymetrickú `CryptoPP::RSA` na podpisovanie a overovanie podpisov sa používajú `CryptoPP::RSASS< CryptoPP::PSSR, CryptoPP::SHA256>`.

Všetky tieto dátové typy je možné previesť na `std::string` a teda použiť jednoducho pri serializácii.

Užívateľské rozhranie (GUI)

GUI je vytvorené pomocou frameworku Qt, ako QWidget aplikácia. Použitie Qt je výhodné, vzhľadom na to, že komunikácia so sieťovaním v Qt bude lepšia a jednoduchšia, najmä v kritických okamihoch, ako napríklad pri čakaní na odpoveď po sieti, čo by pri nesprávnej implementácii mohlo spôsobiť "zaseknutie sa" programu, v tomto prípade jeho GUI.

Sieťová komunikácia

Sieťovú komunikáciu opäť zvláda framework Qt, konkrétne rôzne súčasti `QtNetwork`, ako `QTcpSocket` ako socket, `QTcpServer` ako server pri nových spojeniach, `QNetworkInterface` a ďalšie.

Vlastná implementácia

Parsovanie Markdownu

Markdown je parsovaný len veľmi jednoducho. Najprv sa nájde nadpis (nadpis prvej úrovne) pomocou `std::regex` a následne sa pomocou `std::hash` počítajú hashe jednotlivých odsekov.

Databáza

Jednotlivé databázy (ako napríklad databáza článkov) sú spravované použitím rôznych C++ STL dátových štruktúr, najmä ich "unordered" varianty (ako príklad slúži často používaná `std::unordered_map`). Kde je pri pridávaní relatívne nízka časová zložitosť, rovnako ako pri hľadaní položiek. Tam, kde na type presne nezáleží, sa ráta aspoň s tým, že daná dátová štruktúra obsahuje metódy `begin()` a `end()` a má naimplementovaný aspoň FrowardIterator.