



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

PLATFORMA NA MONITOROVANIE UZLOV V P2P SIETI QTUM
BLOCKCHAINY A DECENTRALIZOVANÉ APLIKACE

Bc. Ján Jakub Kubík

24. apríla 2022

Obsah

1	Úvod	2
2	Teória	3
2.1	Peer-to-peer sieť	3
2.2	Qtum	3
2.3	Pripojenie a komunikácia uzlov v Qtum P2P sieti	4
3	Návrh	6
4	Implementácia	7
4.1	Použité technológie	7
4.2	Najdôležitejšie časti kódu	7
5	Testovanie a experimenty	9
5.1	Správanie platformy	9
5.2	Validácia výsledkov	10
6	Záver	11
	Literatúra	12
A	Súbory k projektu	13

1 Úvod

Kryptomeny sa v dnešnej dobe dostávajú čím ďalej viac medzi bežnú populáciu. Za prvú kryptomenu je považovaný Bitcoin. Vzniklo z neho mnoho klonov. Bitcoin používa takzvaný blockchain. Blockchain je zjednodušene povedané systém, v ktorom sa uchováajú všetky záznamy o uskutočnených transakciách v danej mene na uzloch blockchainovej siete. Táto sieť je typu peer-to-peer. Pri dostatočnej veľkosti siete nie sú útoky na túto sieť prakticky realizovateľné.

Ďalšiou evolúciou v oblasti kryptomien sú decentralizované aplikácie, ktoré sú umožnené vďaka smart kontraktom. Decentralizované aplikácie existujú a bežia na blockchainovej P2P sieti namiesto centrálného bodu ako je server. Smart kontrakty sú programy uložené na blockchaine, ktoré sa spúšťajú, keď sú splnené vopred určené podmienky. Medzi najznámejšie siete podporujúce smart kontrakty patrí Ethereum.

Ethereum a Bitcoin siete nie sú kompatibilné. Tento problém rieši Qtum blockchain.

Cieľami tohoto projektu je:

1. vysvetlenie a stručný popis P2P siete Qtum,
2. návrh a implementácia riešenia na mapovanie dostupných uzlov v sieti Qtum,
3. validácia a zhodnotenie výsledného riešenia.

Kapitola 2 obsahuje vysvetlenie a stručný popis P2P siete Qtum. V Kapitole 3 je návrh riešenia na mapovanie dostupných uzlov v sieti Qtum. Kapitola 4 popisuje najdôležitejšie časti implementácie. Kapitola 5 sa zaoberá validáciou a zhodnotením výsledného riešenia.

2 Teória

V tejto kapitole sú najskôr vysvetlené P2P siete. Ďalej Qtum blockchain. Na záver kapitoly je stručný popis komunikácie v Qtum P2P sieti.

2.1 Peer-to-peer sieť

Sieť peer-to-peer (P2P) sa skladá zo skupiny zariadení, ktoré spoločne používajú a zdieľajú zdroje. Každý účastník (uzol) vystupuje ako individuálny peer. Všetky uzly vykonávajú rovnaké úlohy. To znamená, že peer môže byť v roli klienta (využíva zdroje) a zároveň v roli serveru (poskytuje zdroje) [1]. Tri hlavné kategórie P2P sietí sú:

- **neštrukturované P2P siete** – uzly medzi sebou náhodne komunikujú,
- **štrukturované P2P siete** – predstavujú organizovanú architektúru, ktorá umožňuje uzlom efektívne vyhľadávať a poskytovať zdroje,
- **hybridné P2P siete** – kombinujú konvenčný model klient-server s niektorými aspektmi architektúry peer-to-peer.

2.2 Qtum

Qtum je blockchainová sieť založená v roku 2016, ktorá kombinuje možnosti smart kontraktov Ethereum s účtovným systémom Bitcoinu ktorý sa nazýva UTXO. Dosahuje to prostredníctvom technológie nazývanej Account Abstraction Layer. Táto technológia poskytuje spoločnosti Qtum výhodu v možnosti jednoduchšej implementácie aktualizácií z Bitcoinu ale aj z Ethereum. Qtum používa Proof of Stake konsenzus mechanizmus.

UTXO

V UTXO účtovnom systéme peňaženka používateľa sleduje zoznam nevyčerpaných transakcií spojených so všetkými adresami vo vlastníctve používateľa a zostatok peňaženky sa vypočíta ako súčet týchto nevyčerpaných transakcií. Account/Balance model, ktorý je používaný napr. Ethereum, sleduje zostatok každého účtu ako globálny stav.

Account Abstraction Layer

Blockchainy so smart kontraktami zvyčajne z technických dôvodov nepoužívajú účtovný systém UTXO. Qtum sa aj napriek tomu rozhodlo inak a používa UTXO spolu so smart kontraktami pomocou Account Abstraction Layer (AAL).

AAL funguje tak, že výstup transakcie UTXO používa na vytvorenie smart kontraktu. Potom odošle transakciu na požadovaný účet. A tak spustí vykonávanie kontraktu. AAL následne spracuje výsledky a prispôsobí ich UTXO [2].

2.3 Pripojenie a komunikácia uzlov v Qtum P2P sieti

Účastníkmi v P2P sieťach sú uzly. Uzly prijímajú nové pripojenia a odosielať údaje iným uzlom s ktorými majú aktívne spojenie. Výmena dát medzi uzlami závisí od komunikačného protokolu. Komunikačný protokol určuje všetko od začiatku komunikácie cez pripojenie k uzlom a odosielanie dát. Qtum používa na pripojenie sa do Qtum P2P siete upraveného Bitcoin core klienta. Komunikačný protokol až na pár výnimok zostáva takmer rovnaký ako v Bitcoine.

Qtum klient

Existujú full uzly a lightweight uzly. Full uzly majú stiahnutý celý blockchain a lightweight uzly majú stiahnuté len hlavičky blokov daného blockchainu. Uzly používajú špecializovaný software takzvaného klienta. Prostredníctvom klienta komunikujú s pripojenými uzlami. Klient sa skladá z 2 hlavných častí:

- **démon** – jadro celého klienta,
- **API** – slúži na komunikovanie užívateľa s démonom.

Démon najskôr vyhľadá dostupných peerov prostredníctvom rôznych techník. Následne sa na niekoľkých peerov pripojí a zahájí synchrizačný proces blockchainu. Po synchronizácii klient drží validný blockchain a je pripravený začať validovať nasledujúce bloky a zdieľať svoj pohľad na aktuálny blok s ostatnými peerami.

API existuje vo viacerých variantách ako napríklad nástroj na spúšťanie dotazov cez príkazový riadok alebo priamo prostredníctvom HTTPS dotazov s JSON POST parametrami. Tento spôsob sa nazýva JSON RPC a je veľmi bežný a často používaný. Prostredníctvom API sa dajú zisťovať rôzne informácie o P2P sieti, danom uzle alebo sa dajú pripájať/odpájať uzly v sieti od konkrétného uzlu s ktorými démon pracuje.

Komunikačný protokol

Komunikačný protokol Qtumu vychádza z komunikačného protokolu Bitcoinu. Použitý port je 3888. Dôležitou časťou pre komunikovanie je spôsob objavenia peerov. Všetky informácie o komunikačnom protokole Bitcoinu sú dobre spísané v práci [3] kapitola 2 sekcia 2.5.

Objavenie potenciálnych uzlov, ku ktorým sa dá pripojiť prebieha následovne. Po tom, čo klient začne fungovať, musí nájsť ďalšie aktívne uzly aby sa mohol pripojiť do P2P siete. Klient najskôr skontroluje svoju lokálnu databázu peers.dat. V prípade, že je lokálna databáza prázdna, tak sa klient pokúsi získať IP adresy pomocou DNS rezolúcie hardcodedných DNS seedov priamo v zdrojovom kóde klienta. Ak ani to nie je úspešne a nepodari sa klientovi nadviazať spojenie so žiadnym ďalším uzlom, tak zostáva už len manuálne nájsť fungujúce uzly a nastaviť ich klientovi natvrdo.

Po úspešnom pripojení sa k jednému alebo viacerým uzlom začne klient využívať inú metódu pre objavovanie potenciálnych ďalších uzlov. Konkrétne to je peers rumoring, kedy spojené uzly 'debatujú' o ďalších potenciálnych uzloch.

Uzly udržiavajú aktívne spojenie pomocou zasielania si správ ping, pong.

Qtum core API

Z API boli v projekte použité metódy ¹:

- **getpeerinfo** – vracia informácie o pripojených uzloch ako JSON objekty v poli. Každý objekt predstavuje jeden pripojený uzol.
- **getnodeaddresses** – vracia informácie o IP adresách, o ktorých daný uzol vie. Tieto adresy sa môžu použiť na objavovanie nových uzlov. Adresy sa vrátia ako JSON objekty v poli.
- **addnode** – Hlavným použitím addnode je pripojenie nového uzlu. Správanie addnone závisí od poskytnutých parametrov. Vyžaduje vždy 2 parametre. Prvým z nich je action a druhým address (adresa uzlu), na ktorý sa pokúsi pripojiť.
- **disconnectnode** – slúži na odpojenie jedného konkrétného uzla.

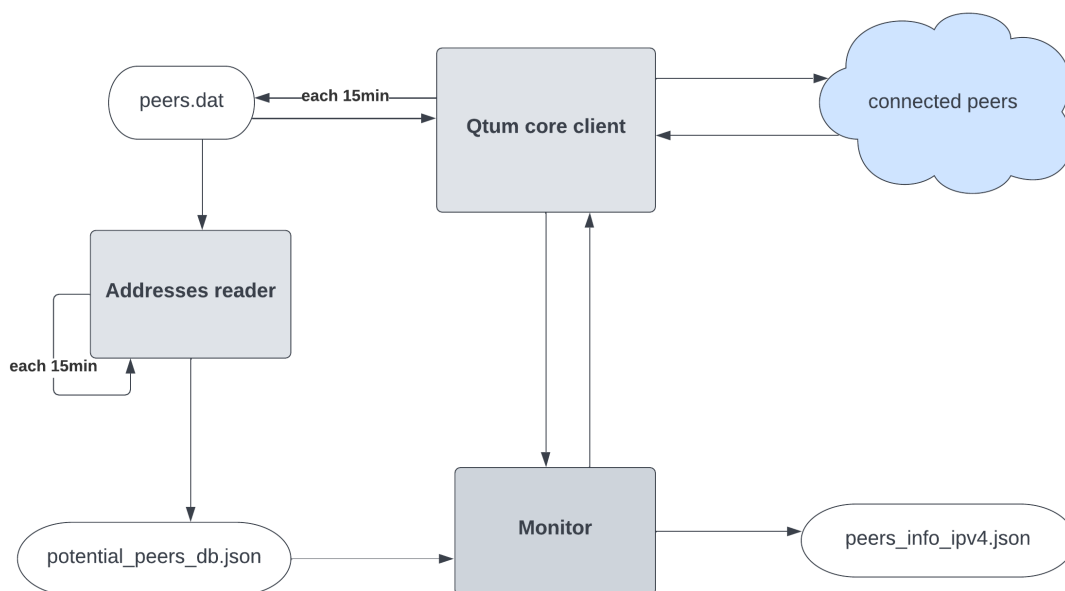
¹<https://docs.qtum.site/en/Qtum-RPC-API/network>

3 Návrh

Na obrázku 5.3 je návrh platformy na monitorovanie uzlov v sieti Qtum. Platforma pozostáva z 3 kľúčových komponent:

- **Qtum core client** – reprezentuje pripojený uzol do Qtum P2P siete. Cez jeho API sa zisťujú všetky potrebné informácie o pripojených peeroch a pripájajú/odpájajú sa peeri. Každých 15 minút si aktualizuje internú databázu dostupných peerov do binárneho súboru peers.dat.
- **Addresses reader** – pársuje každých 15 min peers.dat do súboru potential_peers_db.json.
- **Monitor** – súži na samotné monitorovanie siete. Z potential_peers_db.json a cez Qtum core client API si zistí zoznam použiteľných peerov, vyskúša sa cez API na nich pripojiť a zistiť informácie o nich. Zistené informácie pravidelne ukladá do súboru peers_info_ipv4.json.

Popísané činnosti robia všetky 3 komponenty dokola.



Obr. 3.1: Návrh platformy na monitorovanie uzlov

4 Implementácia

V tejto kapitole sú vysvetlené použité technológie a najdôležitejšie časti kódu.

4.1 Použité technológie

Jednotlivé komponenty boli rozdelené do Docker kontajnerov ¹ a nastavené pomocou Docker Compose ². Na samotné programovanie bol použitý jazyk Python a Bash. Ďalej boli použité Qtum core client ³ a bitpeers ⁴ na párovanie binárnej databázy peers.dat.

4.2 Najdôležitejšie časti kódu

Qtum core client

Docker file pre stiahnutie Qtum core klienta z githubu a jeho zostavenie:

```
WORKDIR /app

RUN apt-get -y update && \
    apt-get -y install git && \
    apt-get -y install bsdmainutils && \
    apt-get install gcc && \
    echo "Y" | apt-get install libdb++-dev && \
    echo "Y" | apt-get install build-essential g++ \
    python-dev autotools-dev libicu-dev libbz2-dev libboost-all-dev && \
    git clone https://github.com/qtumproject/qtum --recursive

RUN cd qtum && \
    ./autogen.sh && \
    ./configure --disable-wallet && \
    make -j2
```

Qtum core klient je spúšťaný z docker-compose ako služba node.

¹<https://docs.docker.com/get-started/>

²<https://docs.docker.com/compose/>

³<https://github.com/qtumproject/qtum>

⁴<https://github.com/RaghavSood/bitpeers>

Addresses reader

V Docker kontajneri sťahuje a inštaluje bitpeers z GitHubu. Pomocou tohoto nástroju vie párovať binárnu databázu uzlov peers.dat. Spársovanú databázu reader ukladá do JSON súboru. Databázu peers.dat si vytvára a aktualizuje Qtum core klient automaticky. Tak robí každých 15 minút. Bash kód:

```
BITPEERS_PATH="/app/go/bitpeers/cmd/bitpeers"
BINARY_PEERS_PATH="${PROJECT_PATH}/app/qtumd/peers.dat"
OUTPUT_FILE='/app/address_reader/potential_peers_db.json'
FORMAT='json'

SLEEP_SECONDS=$((15 * 60))
echo $BITPEERS_PATH

cd $BITPEERS_PATH

BITPEERS='bitpeers'
chmod '+x' $BITPEERS
COMMAND="${BITPEERS} '--filepath '
        '$BINARY_PEERS_PATH' '--format ' $FORMAT > $OUTPUT_FILE"
while true
do
    eval $COMMAND
    echo $COMMAND
    echo "Sleep for $SLEEP_SECONDS"
    sleep $SLEEP_SECONDS
done
```

Address reader je spúšťaný z docker-compose ako služba address_reader.

Monitor

Slúži na samotné monitorovanie uzlov. To znamená ich pripájanie/odpájanie a zisťovania informácií o nich cez Qtum core klientské API. Najdôležitejšiou funkciou je funkcia monitor. Tá si najskôr zistí zoznam použiteľných peerov cez API a zo súboru. Zlúči ich a postupne cez nich iteruje. V každej iterácii skúša či sa dá k danému peerovi pripojiť a zisťuje o nich informácie. Ak sa pokúsila už k peerovi pripojiť tak to už v ďalšej iterácii neskúša. Použité API metódy sú popísané v sekcii 2.3 časť Qtum core API. Ďalej je dôležitá trieda RequestWrapper, ktorá sa používa na volanie už zmienených API metód.

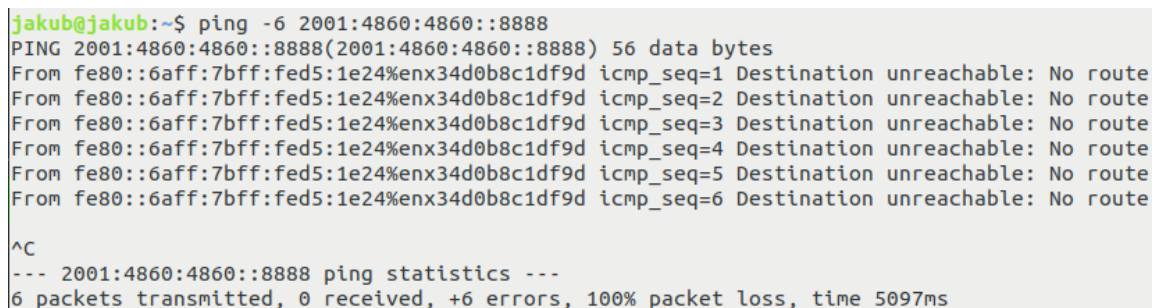
Monitor je spúšťaný z docker-compose ako služba monitor.

5 Testovanie a experimenty

V tejto kapitole je postupne vysvetlené správanie sa platformy na monitorovanie a problémy s IPv6. Ďalej je popísaný súhrn získaných dát a na záver je validovanie výsledkov.

5.1 Správanie platformy

Pôvodne som chcel zistiť informácie o všetkých uzloch používajúcich buď IPv4 alebo IPv6. Tak som skúšal všetky dostupné IPv4 a IPv6. Po pár hodinách som skontroloval získané dáta. Zistil som, že ani jeden pokus o pripojenie sa na uzol s IPv6 neuspel. Tak som sa teda rozhodol vyskúšať ping na nejakú známu IPv6 adresu. Použil som IPv6 adresu Google serveru (viď obrázok 5.1).



```
jakub@jakub:~$ ping -6 2001:4860:4860::8888
PING 2001:4860:4860::8888(2001:4860:4860::8888) 56 data bytes
From fe80::6aff:7bff:fed5:1e24%enx34d0b8c1df9d icmp_seq=1 Destination unreachable: No route
From fe80::6aff:7bff:fed5:1e24%enx34d0b8c1df9d icmp_seq=2 Destination unreachable: No route
From fe80::6aff:7bff:fed5:1e24%enx34d0b8c1df9d icmp_seq=3 Destination unreachable: No route
From fe80::6aff:7bff:fed5:1e24%enx34d0b8c1df9d icmp_seq=4 Destination unreachable: No route
From fe80::6aff:7bff:fed5:1e24%enx34d0b8c1df9d icmp_seq=5 Destination unreachable: No route
From fe80::6aff:7bff:fed5:1e24%enx34d0b8c1df9d icmp_seq=6 Destination unreachable: No route

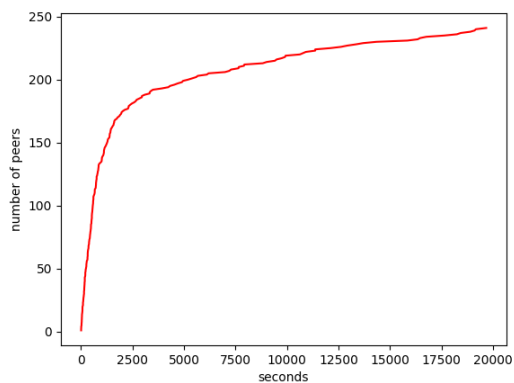
^C
--- 2001:4860:4860::8888 ping statistics ---
6 packets transmitted, 0 received, +6 errors, 100% packet loss, time 5097ms
```

Obr. 5.1: ping

Z toho som usúdil, že môj provider internetu nepodporuje alebo nemá správne nakonfigurovanú IPv6.

Finálne monitorovanie

Finálne monitorovanie bežalo po dobu približne 10 hodín. Z obrázku je zrejmé, že postupom času pribúdalo menej a menej nových peerov.

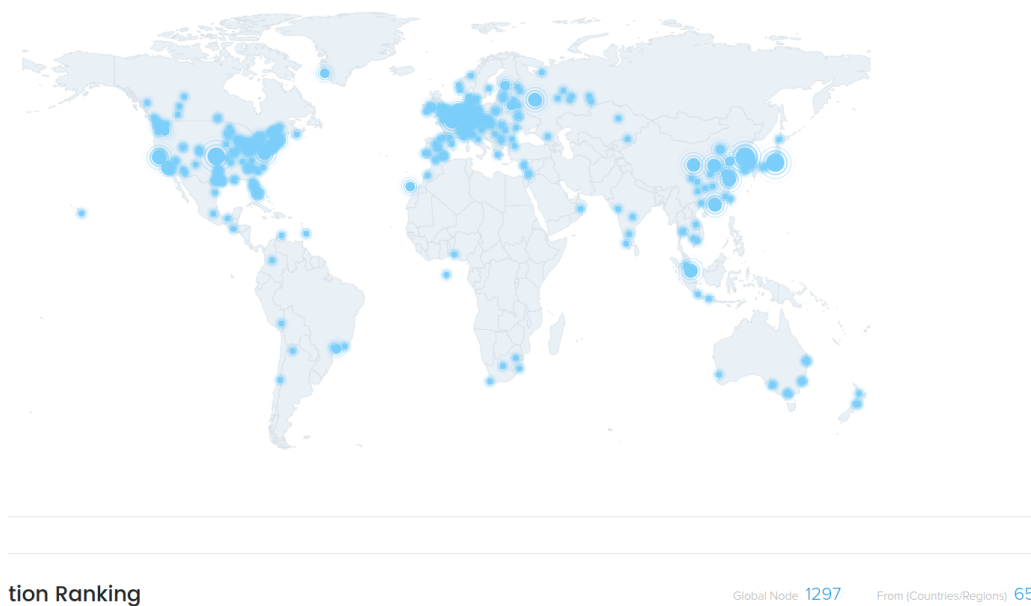


Obr. 5.2: Pripájanie nových peerov

Získané dáta o jednotlivých peeroch sú v súbore **peers_info_ipv4.json**.

5.2 Validácia výsledkov

Z oficiálnej stránky Qtum ¹ som zistil, že globálne je približne 1200 aktívnych uzlov.



Obr. 5.3: Globálna distribúcia uzlov

Mne sa bohužiaľ podarilo nájsť len približne 250 uzlov. Toto je s najväčšou pravdepodobnosťou zapríčinené absenciou IPv6 a Onion sietí v mojej platforme na monitorovanie.

¹<https://qtum.org/en/product/nodemap>

6 Záver

Vytvorená platforma na monitorovanie uzlov v P2P sieti Qtum nezmapovala celú Qtum sieť a našla pomene málo peerov v porovnaní s konkurenčnými nástrojmi. To je zapríčinené tým, že sa mi nepodarilo spustiť IPv6 monitorovanie a monitorovanie Onion časti siete som ani neimplementoval.

Výhoda tohoto nástroja je v jednoduchosti jeho spúšťania a používania.

Do budúcnosti by bolo vhodné spustiť nástroj v sieti s podporou IPv6 a doimplementovať monitorovanie Onion siete. V tomoto prípade by sa mohla porovnávať s už existujúcimi riešeniami ako je napríklad bitpeers.

Literatúra

- [1] BINANCE. *Peer-to-Peer Networks Explained*. Dostupné z: <https://academy.binance.com/en/articles/peer-to-peer-networks-explained>.
- [2] BINANCE. *What Is Qtum*. Dostupné z: <https://academy.binance.com/en/articles/what-is-Qtum-Qtum>.
- [3] ZAUJEC. *Cryptocurrency Node Monitoring*. 2020. Bc thesis. Brno University of Technology.

A Súbory k projektu

Dodatok obsahuje zoznam súborov k projektu.

- docker-compose.yaml – nastavenia pre všetky servisy
- Dockerfile_addresses_reader – Dockerfile pre zostavenie address_reader servisu
- Dockerfile_monitor – Dockerfile pre zostavenie monitor servisu
- Dockerfile_node – Dockerfile pre zostavenie node servisu
- README.md
- requirements.txt – Pythonom požadované balíčky pre monitor servisu
- app/
 - config.py – konfiguračný súbor pre monitor
 - monitor.log – logy z behu monitoru
 - monitor.py – logika monitoru
 - peers_info_ipv4.json – dostupný peeri a info o nich
 - potential_peers_db.py – zlučovanie adres dostupných peerov
 - requests_wrapper.py – zaobalené Qtum core API requesty
 - utils.py – pomocné funkcie
 - address_reader/
 - * address_reader.sh – hlavný skript pre address reader servisu
 - * potential_peers_db.json – spárované adresy peerov z peers.dat
 - qtumd/
 - * debug.log – logy z behu Qtum core klienta
 - * peers.dat – binárna databáza peerov z klienta