Εξαμηνιαία Εργασία στα Κατανεμημένα Συστήματα

Ακαδημαϊκό Έτος 2022 – 2023

Ονοματεπώνυμο	Αριθμός Μητρώου
Στεφανάκης Γεώργιος	el18436
Μπούφαλης Οδυσσεύς - Δημήτριος	el18118
Φώτης Νικόλαος	el15090

Noobcash

Το Noobcash μιμείται τις τεχνικές που χρησιμοποιούνται σε πολλά συστήματα blockchain που βασίζονται σε συναλλαγές, όπως η αποκέντρωση, η κρυπτογράφηση hash, το mining και το consensus που επιτυγχάνεται μέσω Proof-of-Work. Οι κόμβοι στους οποίους εκτελείται το backend χρησιμεύουν τόσο ως clients όσο και ως miners.

Το backend είναι μία single-threaded εφαρμογή που αναπτύχθηκε σε Node.js & TypeScript. Επιπλέον, πέρα από το CLI tool που ζητήθηκε, κατασκευάσαμε και μία web εφαρμογή με Angular, όπου κάθε κόμβος μπορεί να συνδεθεί και να δει το χρηματικό του υπόλοιπο, τις συναλλαγές του τελευταίου block της αλυσίδας σε πραγματικό χρόνο, καθώς επίσης και να πραγματοποιήσει μία νέα συναλλαγή προς κάποιον κόμβο του δικτύου.

GitHub Repository: https://github.com/stefanaki/noobcash

Backend

Το Noobcash Backend είναι ένα REST API, το οποίο παρέχει στους υπόλοιπους κόμβους του δικτύου αλλά και στους εξωτερικούς consumers τις εξής λειτουργίες:

- GET /transaction Επιστρέφει τις συναλλαγές του τελευταίου block που έχει γίνει mine
- POST /transaction Κατασκευάζει μία συναλλαγή με συγκεκριμένο παραλήπτη και ποσό
- PUT /transaction Ενεργοποιείται όταν γίνει broadcast ένα νέο transaction
- POST /block Αποθήκευση ενός νέου block που έγινε μόλις broadcast στο δίκτυο
- GET /blockchain Επιστρέφει το blockchain την στιγμή που έγινε το request
- POST /blockchain Καλείται όταν παρατηρηθούν conflicts για την ανανέωση του local state
- GET /balance Επιστρέφει το υπόλοιπο του κόμβου από τον οποίο έγινε το request
- GET /balances Επιστρέφει το υπόλοιπο όλων των κόμβων
- GET /ring Επιστρέφει όλους τους συνδεδεμένους κόμβους στο κατανεμημένο σύστημα
- POST /ring Ανανεώνει τον κατάλογο των συνδεδεμένων κόμβων όταν εισαχθεί νέος
- POST /node[i] Καλείται όταν ένας νέος κόμβος ζητήσει να εισαχθεί στο δίκτυο

Υποθέτουμε ότι δεν ξεκινάει κανείς κόμβος να κάνει mine πριν εισαχθούν όλοι οι κόμβοι. Όταν ένας νέος κόμβος ξεκινάει την εκτέλεσή του, στέλνει POST /node στον Bootstrap Node, και εκείνος με τη σειρά του κάνει broadcast POST /ring για να ανανεωθεί ο κατάλογος των κόμβων σε όλο το δίκτυο. Ο Bootstrap Node διαθέτει από την αρχή της εκτέλεσής, υπόλοιπο ίσο με $100 \times N$ NBC και το συναλλάσσει με τους νεοεισερχόμενους κόμβους. Οι συναλλαγές που πραγματοποιούνται εισέρχονται σε ένα Transaction Queue, υλοποιημένο αποδοτικά με σωρό, και όταν παρατηρηθεί ότι το μέγεθός του είναι μεγαλύτερο ή ίσο από το Block Capacity, τότε αφαιρούνται τα πρώτα έγκυρα transactions και προστίθενται στο τρέχον block. Τότε ξεκινάει και η διαδικασία mining. Ο πρώτος κόμβος που θα βρει το κατάλληλο nonce, θα κάνει broadcast στο δίκτυο το δικό του block, μαζί με ένα hash/checksum του τοπικού του state. Οι κόμβοι που θα λάβουν το block, θα ελέγξουν την συνέπεια του δικτύου συγκρίνοντας το checksum του δικού τους state με εκείνο που έλαβαν. Εάν παρατηρηθεί διαφορά, τότε έχει προκύψει διακλάδωση στα states των κόμβων

και καλείται η συνάρτηση επίλυσης συγκρούσεων. Η διαδικασία επαναλαμβάνεται έως ότου να αδειάσει το Transaction Queue.

Οι βασικές συνιστώσες τις εφαρμογής μας είναι οι οντότητες Node, Wallet, Block, Blockchain, Transaction, TransactionInput, TransactionOutput. Ο Bootstrap Node κληρονομεί όλες τις λειτουργίες ενός απλού κόμβου, ενώ υλοποιεί και τη λογική της κατασκευής του δικτύου και του Genesis Block. Οι βασικές λειτουργίες που αφορούν το validation των συναλλαγών και του Blockchain, την υπογραφή ενός transaction από τον κατασκευαστή του, την διαχείριση των UTXO's και του Transaction Queue, το mining κ.λπ., γίνονται στις κλάσεις/services της εφαρμογής, Transaction Service, Blockchain Service, Mining Service.

Για το εύκολο ανέβασμα και εκτέλεση της εφαρμογής στα VM's του ~okeanos-knossos, κάναμε containerize την εφαρμογή με Docker, και τρέξαμε το image στα μηχανήματα που μας δόθηκαν.

CLI Tool

Το CLI κατασκευάστηκε με Shell Scripting και παρέχει τις εξής λειτουργίες:

$ ./cli.sh help

Commands:

balance - view your wallet balance

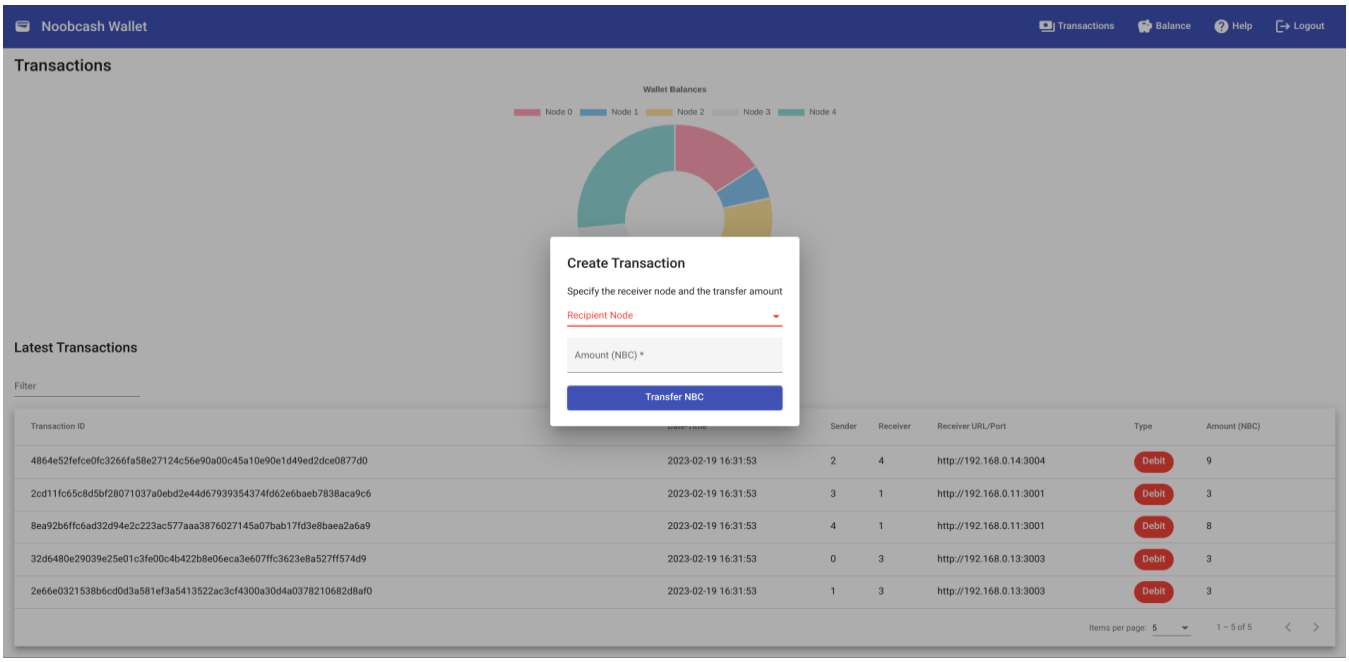
transaction <recipient_id> <amount> - make a transaction to the specified recipient with the specified amount

view - view all transactions

help - view this help message

Το CLI ενός κόμβου στέλνει HTTP request στα endpoints του ίδιου του backend του, στα GET /balance, POST /transaction, GET /transaction.

Web Application



Κάθε κόμβος μπορεί να εισέλθει στην δικτυακή εφαρμογή και να δει το υπόλοιπό του, καθώς και το υπόλοιπο των άλλων κόμβων του δικτύου, σε μορφή διαγράμματος. Μπορεί επίσης να κατασκευάσει ένα transaction με παραλήπτη κάποιον άλλο κόμβο. Οι αλλαγές στο σύστημα γίνονται αντιληπτές σε πραγματικό χρόνο, αφού γίνει mine το τρέχον block.

Benchmarks

5 Nodes

Throughput	Difficulty	
Block Capacity	4	5
1	0.9	0.11
5	2.1	0.24
10	2.32	0.36

Block Time	Difficulty	
Block Capacity	4	5
1	0.67	3.81
5	0.89	7.73
10	1.04	12.17

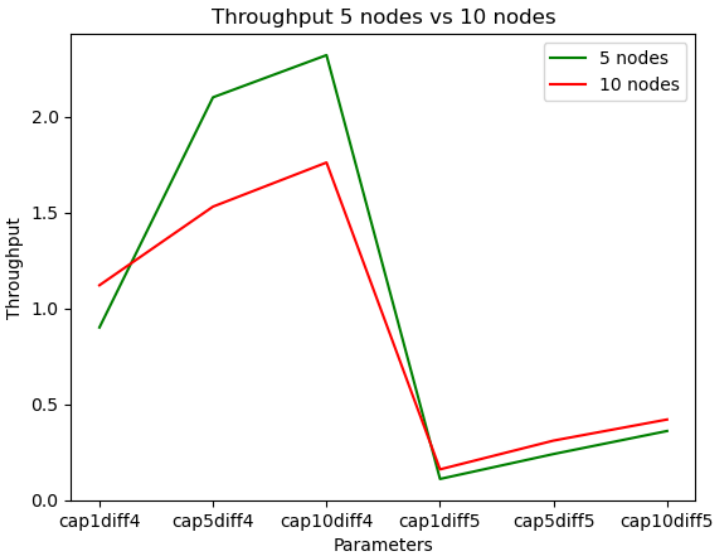
10 nodes

Throughput	Difficulty	
Block Capacity	4	5
1	1.12	0.16
5	1.53	0.31
10	1.76	0.42

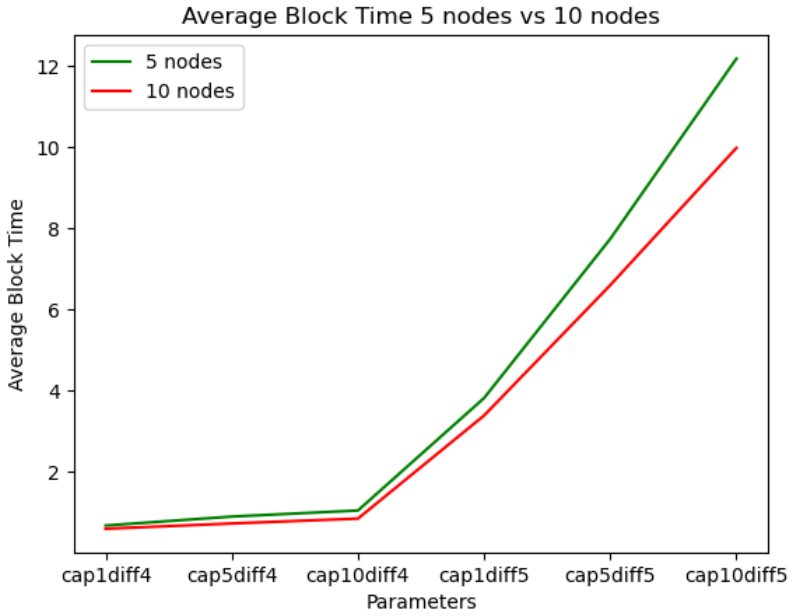
Block Time	Difficulty	
Block Capacity	4	5
1	0.59	3.38
5	1.53	0.72
10	0.84	9.97

Στη συνέχεια, απεικονίζουμε γραφικά τα παραπάνω αποτελέσματα για να δούμε ποιοτικά τη συσχέτιση των μετρικών throughput και block time με τις τιμές των block size και difficulty. Επίσης, βλέπουμε και το scalability του συστήματός μας καθώς απεικονίζουμε στα ίδια διαγράμματα τις τιμές των μετρικών για 5 και 10 κόμβους.

Παρακάτω, φαίνεται το throughput του συστήματός μας για 5 nodes και 10 nodes με block size 1,5,10 και difficulty 4 και 5. Βλέπουμε πως όσο αυξάνεται το block size αυξάνεται το throughput για σταθερό difficulty. Επίσης, βλέπουμε ότι η αύξηση του difficulty ρίχνει σημαντικά το throughput κάτι που καθιστά φανερό τον αντίκτυπο της δυσκολίας του PoW.



Αντίστοιχα, για το block time βλέπουμε το παρακάτω διάγραμμα για 5 nodes και 10 nodes με block size 1,5,10 και difficulty 4 και 5. Παρατηρούμε ότι όσο αυξάνεται το block size και το difficulty το block time αυξάνεται.



Όσον αφορά το scalability του συστήματός μας, παρατηρούμε ότι και με 5 και με 10 κόμβους οι τιμές των μετρικών εμφανίζουν ποιοτική συνέπεια μεταξύ τους.

[i] Μόνο ο bootstrap κόμβος ακούει σε αυτό το endpoint