ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ ΣΤΑ ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Ακ. έτος 2021-2022, 9ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Πεγειώτη Νάταλυ – 03117707
Ομάδα 15

Noobcash Blockchain

Σκοπός

Σκοπός της εργασίας ήταν η δημιουργία ενός Blockchain (με το όνομα Noobcash) μέσω του οποίου ένας αριθμός N κόμβων μπορεί να εκτελεί συναλλαγές, χωρίς τη μεσολάβηση κάποιου TTP (Trusted Third Party, πχ κάποια τράπεζα). Ο κάθε κόμβος/χρήστης του συστήματος που θέλει να πληρώσει κάποιον άλλο, στέλνει την συναλλαγή σε όλο το σύστημα. Οι κόμβοι γεμίζουν το block τους με συναλλαγές και όταν αυτό γεμίσει ξεκινούν μια διαδικασία mining με Proof-of-Work, ψάχνοντας κατάλληλο nonce ώστε τα πρώτα d ψηφία του hash του block να είναι 0. Όταν κάποιος κόμβος βρει το κατάλληλο nonce, στέλνει το block τους στους υπόλοιπους και το προσθέτει και ο ίδιος στην αλυσίδα του.

Σχεδιασμός/Υλοποίηση

Για το σχεδιασμό του συστήματος χρησιμοποιήθηκε ο σκελετός που δόθηκε για python, στον οποίο προστέθηκαν άλλα 3 αρχεία, το cli.py στο οποίο περιέχεται η υλοποίηση για τον noobcash_client, το blockchain.py το οποίο περιέχει την υλοποίηση για τη λίστα με τα block του κάθε κόμβου και το constants.py στο οποίο απλά περιέχονται σταθερές για τη λειτουργία του συστήματος (difficulty, capacity και αριθμός κόμβων).
Ακολούθως περιγράφεται συνοπτικά ο σχεδιασμός και η χρήση κάθε κλάσης/αρχείου.
**Ο κώδικας συνοδεύεται με σχόλια*

rest.py

Στην κλάση αυτή περιέχονται τα endpoints ενός RESTful API το οποίο χρησιμοποιούν οι κόμβοι για επικοινωνία, αλλά και για εξυπηρέτηση αιτημάτων από κάποιο Client, όπως το CLI.
Κατά την εκκίνηση του rest.py ο χρήστης δίνει τα ορίσματα *-p [PORT] -a [ADDRESS] -l [BOOTSTRAPADDRESS] -m [BOOTSTRAPPORT] -b*, με την προφανή χρήση του καθενός και με τη σημαία -b να δίνεται μόνο αν ο τρέχον κόμβος είναι ο bootstrap κόμβος. Βάσει αυτών δημιουργείται ένα αντικείμενο της κλάσης node και σε αυτό αποθηκεύονται οι πληροφορίες για τον κάθε κόμβο. Τα endpoints εκτελούν διάφορους ελέγχους και επιστρέφουν ανάλογα τη ζητούμενη απάντηση με κωδικό 200 ή το σφάλμα και τον κατάλληλο κωδικό σφάλματος.

cli.py

Ο κώδικας του CLI δεν είναι παρά ένα ατέρμονο while loop το οποίο περιμένει να δώσει ο χρήστης κάποια εντολή. Κατά την εκτέλεση του αρχείου ο χρήστης δίνει με χρήση των flags *-a* και *-p* τη διεύθυνση ip και τη θύρα στην οποία λειτουργεί ο server του.
Το CLI προσφέρει 6 λειτουργίες, τις 4 που ζητούνται στην εκφώνηση καθώς και τις *exit* και *clear*.

block.py

Τα πεδία που περιέχει είναι ακριβώς αυτά που περιγράφονται στην εκφώνηση, ενώ επιπρόσθετο είναι το πεδίο creator, το οποίο δηλώνει το δημιουργό του block. Η προσθήκη αυτή γίνεται απλά για περαιτέρω διαφοροποίηση των block μεταξύ τους (διαφορετικά σε περίπτωση που το σύστημα δεν παρουσίαζε βλάβες και 2 κόμβοι δημιουργούσαν τα ταυτόχρονα το block τότε θα έκαναν mine ακριβώς το ίδιο block).

Η κλάση διαθέτει τις ακόλουθες μεθόδους

to_json(): για μετατροπή του αντικειμένου σε αναπαράσταση JSON, η οποία είναι αναγκαία τόσο για να γίνει hash το block αλλά και για να μπορεί ακολούθως να μεταδοθεί
MyHash(): για υπολογισμό του hash του block
add_transaction(): για την προσθήκη μιας συναλλαγής στο block και την κατάλληλη ενημέρωση στην τιμή του πεδίου current_hash

blockchain.py

Η κλάση blockchain περιέχει απλά δύο πεδία, τη λίστα με τα blocks και το μήκος αυτής.

Η κλάση διαθέτει τις ακόλουθες μεθόδους

add_block(): για προσθήκη block στην αλυσίδα
to_json(): για σειριοποίηση αντικειμένων της κλάσης
lastBlock(): για επιστροφή του τελευταίου block που προστέθηκε στην αλυσίδα

wallet.py

Τα πεδία που περιέχονται στην κλάση είναι αυτά που δίνονται στην εκφώνηση.
Κατά τη δημιουργία ενός αντικειμένου της κλάσης παράγεται ένα κλειδί RSA, το οποίο διανέμεται στους υπόλοιπους.

Η μοναδική μέθοδος της κλάσης είναι η εξής

balance(): για άθροιση των UTXOS του κόμβου και επιστροφή του αποτελέσματος, το οποίο αντιστοιχεί στα διαθέσιμα χρήματα που υπάρχουν στο πορτοφόλι.
Για λόγους μετάδοσης, το κλειδί RSA αποθηκεύεται σε μορφή hex, γεγονός που επιτρέπει την αναδημιουργία του στην “άλλη πλευρά του σύρματος”, με χρήση της *RSA.importKey()*

transaction.py

Η κλάση transaction περιέχει τα πεδία που αναφέρονται στην εκφώνηση.

Περιλαμβάνει τις μεθόδους

to_dict(): για τη μετατροπή σε dictionary ώστε να είναι δυνατή η μετάδοση αντικειμένων της κλάσης
myHash(): για υπολογισμό του hash, που χρησιμοποιείται σαν id της κάθε συναλλαγής
Το transaction_id υπολογίζεται πριν οριστούν όλα τα πεδία της κλάσης, καθώς η μοναδικότητα του αναγνωριστικού εξασφαλίζεται από το ότι τα transaction_inputs είναι προφανώς διαφορετικά
outputUTXOS(): για παραγωγή των transaction_outputs της συναλλαγής
add_utxos(): για προσθήκη των transaction_inputs
sign_transaction(): για υπογραφή με χρήση του private key του δημιουργού της συναλλαγής
Η υπογραφή υπολογίζεται με χρήση του private key του αποστολέα, κατά τη δημιουργία ενός transaction, χωρίς όμως το κλειδί να φυλάσσεται κάπου, για λόγους ασφαλείας.

node.py

Η κλάση node είναι η βασικότερη της υλοποίησης. Στα πεδία της περιέχονται χρήσιμες πληροφορίες όπως το id του κόμβου, η διεύθυνση ip του, η διεύθυνση του bootstrap, ο αριθμός των κόμβων που συμμετέχουν στο σύστημα, η αλυσίδα του κόμβου, καθώς και πληροφορίες για τον κάθε κόμβο που αποθηκεύονται στο dictionary ring.
Ανο κόμβος είναι ο bootstrap αρχικοποιούνται κατάλληλα κάποια πεδία, αλλιώς γίνεται ενημέρωση των τιμών τους από τον bootstrap.

Οι μέθοδοι που υλοποιούνται είναι οι ακόλουθες

generate_wallet(): για τη δημιουργία ενός νέου wallet για τον νέο κόμβο
create_transaction(): για τη δημιουργία μιας συναλλαγής (σαν αποστολέας ορίζεται ο ίδιος ο κόμβος, ενώ το ποσό και ο αποστολέας δίνονται ως ορίσματα)
broadcast_transaction(): για τη μετάδοση μιας συναλλαγής σε όλο το δίκτυο
verify_signature(): για επαλήθευση της υπογραφής ενός transaction
validate_transaction(): για έλεγχο εγκυρότητας μιας συναλλαγής, δηλαδή αν ο κόμβος που την δημιούργησε έχει αρκετό υπόλοιπο για την πραγματοποίησή της, αλλά και την εγκυρότητά της μέσω της *verify_signature()*
create_new_block(): για τη δημιουργία ενός νέου block
add_transaction_to_block(): για προσθήκη μιας συναλλαγή στο τρέχον block του κόμβου
mine_block(): για να γίνει mine ένα block το οποίο έχει “γεμίσει” με transactions σύμφωνα με τη σταθερά CAPACITY. Γίνεται αύξηση του nonce και υπολογισμός του hash μέχρι να βρεθεί κάποιο κατάλληλο βάσει της σταθεράς DIFFICULTY.
broadcast_block(): για τη μετάδοση ενός block σε όλο το δίκτυο.
validate_block(): για έλεγχο εγκυρότητας ενός ληφθέντος block. Ελέγχεται ότι το previous_hash του καινούριου block ταιριάζει με το current_hash του τελευταίου block στην αλυσίδα,
και ότι ο αριθμός των μηδενικών στην αρχή του hash είναι όντως σωστός.
validate_chain(): για έλεγχο εγκυρότητας της αλυσίδας, δηλαδή αν τα blocks σε ένα chain έχουν σωστές τιμές στο current_hash και previous_hash.
resolve_conflicts(): για έλεγχο δημιουργίας διακλάδωσης. Καλείται όταν ο κόμβος λάβει κάποιο block το οποίο δεν είναι valid. Ο κόμβος ρωτάει όλους τους υπόλοιπους για τα μήκη των αλυσίδων τους και κρατάει τη μακρύτερη.
register_node_to_ring(): καλείται μόνο από τον bootstrap όταν λάβει αίτημα για προσθήκη κόμβου στο σύστημα. Ο bootstrap ενημερώνει με τις καινούριες πληροφορίες το ring του και στέλνει το ring, την αλυσίδα του, 100 NBC coins και ένα αναγνωριστικό στον κόμβο που έκανε το αίτημα. Επιπλέον, ενημερώνει τους υπόλοιπους με τα στοιχεία του νέου κόμβου.
reconstruct_X(): για την αναδημιουργία αντικειμένων transcation, block και blockchain από μηνύματα JSON ή dictionaries

Προβλήματα/Παραδοχές

1. Για συντονισμό των κόμβων έγινε χρήση χρονοκαθυστερήσεων και event της κλάσης threading. Μάλλον μια καλύτερη προσέγγιση θα ήταν η χρήση locks.
2. Δεν επιτεύχθηκε ποτέ η εκτέλεση των πειραμάτων στα VMs που δημιουργήθηκαν στον Ωκεανό, παρά τις προσπάθειες.

Πειράματα

Όπως αναφέρθηκε και προηγουμένως η εκτέλεση των πειραμάτων στον Ωκεανό απέτυχε.
Αντ'αυτού τα πειράματα εκτελέστηκαν τοπικά, "σηκώνοντας' τους servers σε διαφορετικές θύρες και όχι σε διαφορετικά μηχανήματα.

Σαν αποτέλεσμα των πιο πάνω, τα αποτελέσματα για την ρυθμαπόδοση (throughput) και τον μέσο χρόνο mining κάθε block είναι πολύ κακά και δεν μπορούν να δώσουν μια σωστή εικόνα για αξιολόγηση του συστήματος.
Παρόλαυτα ακολούθως γίνεται η σύγκριση των αποτελεσμάτων για διαφορετικές παραμέτρους καθώς και για την κλιμακωσιμότητα του συστήματος.
Για την εκτέλεση των πειραμάτων με τα δοσμένα δεδομένα χρησιμοποιήθηκαν τα scripts reader.py τα οποία βρίσκονται στους φακέλους 5nodes και 10nodes, ανάλογα με το πλήθος των κόμβων.

Τα αποτελέσματα που λήφθηκαν από τα πειράματα είναι τα ακόλουθα:

5 Nodes

Throughput			
Difficulty/Capacity	1	5	10
4	0.373221	0.691885	0.667005
5	0.033649	0.043902	0.049479

Average Mining Time			
Difficulty/Capacity	1	5	10
4	2.893481	8.093199	17.261917
5	32.561444	127.546775	234.199357

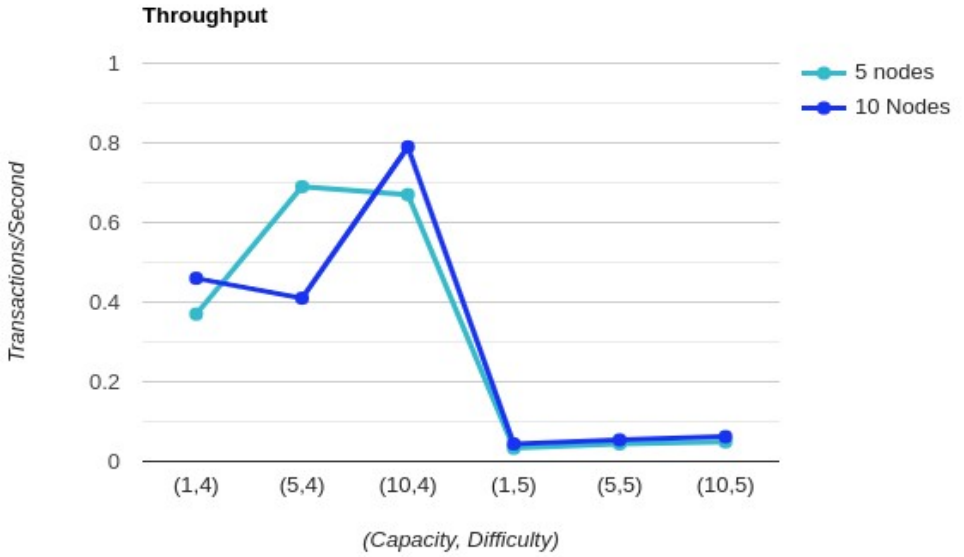
10 Nodes

Throughput			
Difficulty/Capacity	1	5	10
4	0.457984	0.413945	0.795247
5	0.044931	0.054539	0.061702

Average Mining Time			
Difficulty/Capacity	1	5	10
4	2.130026	12.260308	13.877761
5	22.370901	93.519655	178.320447

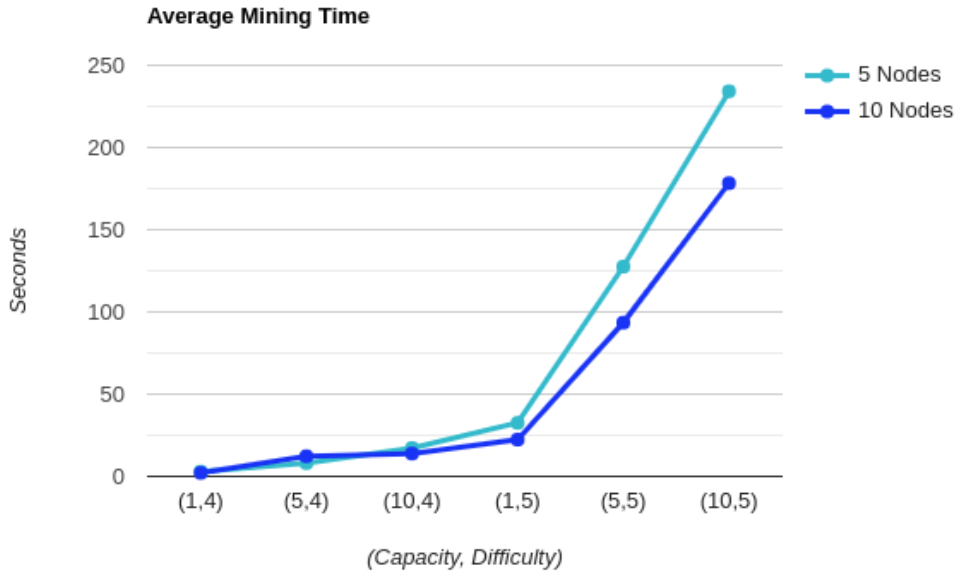
Σύμφωνα με τα πιο πάνω αποτελέσματα δημιουργήθηκαν τα ζητούμενα γραφήματα τα οποία παρουσιάζονται στη συνέχεια.

Όσον αφορά την ρυθμαπόδοση (throughput) προκύπτει το ακόλουθο γράφημα:



Παρατηρείται ότι μεγαλύτερες τιμές capacity οδηγούν σε καλύτερη ρυθμαπόδοση. Αυτό οφείλεται στο ότι για τον ίδιο αριθμό συναλλαγών χρειάζεται να γίνουν mine λιγότερα block. Επιπλέον, φαίνεται πως ο μεγαλύτερος αριθμός κόμβων επιφέρουν μεγαλύτερο throughput, γεγονός που ίσως οφείλεται στο ότι με μεγαλύτερο αριθμό κόμβων υπάρχουν περισσότερες πιθανότητες κάποιο nonce να είναι το κατάλληλο ώστε να γίνει mine πιο εύκολα.

Για τον μέσο χρόνο mining ενός block προκύπτει το ακόλουθο γράφημα:



Παρατηρείται ότι όσο μεγαλώνει η τιμή του capacity, αυξάνεται και ο μέσος χρόνος mining. Αυτό οφείλεται στο ότι ο υπολογισμός του hash παίρνει περισσότερο χρόνο, λόγω του μεγέθους του string που δίνεται σαν είσοδος στον SHA256. Επίσης, και σε αυτή την περίπτωση παρατηρείται καλύτερη απόδοση, δηλαδή μικρότερος χρόνος mining, στο δίκτυο με περισσότερους κόμβους, γεγονός που και πάλι ίσως οφείλεται στο ότι υπάρχουν μεγαλύτερες πιθανότητες για εντοπισμό του κατάλληλου nonce.