**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

## Ροή Υ : Κατανεμημένα Συστήματα

# Αναφορά Εξαμηνιαίας Εργασίας
# Σύστημα Noobcash

Γλυτσός Μάριος 03117055

Κριθαρούλας Διονύσιος 03117875

Αγερίδης Γιώργος 03117204

Ομάδα 29

Εξάμηνο: 9ο

**Εισαγωγή**

Το blockchain είναι η τεχνολογία πίσω από τα περισσότερα κρυπτονομίσματα και αποτελεί στην πραγματικότητα μια κατανεμημένη βάση που επιτρέπει στους χρήστες της να κάνουν δοσοληψίες (transactions) μεταξύ τους με ασφάλεια, χωρίς να χρειάζονται κάποια κεντρική αρχή (π.χ. τράπεζα).

Στην εργασία αυτή κατασκευάσαμε το noobcash, ένα απλό σύστημα blockchain, όπου καταγράφονται οι δοσοληψίες μεταξύ των συμμετεχόντων και εξασφαλίζεται το consensus με χρήση Proof-of-Work.

Συνοπτικά, το σύστημά σας θα πρέπει να υλοποιεί τις παρακάτω λειτουργίες:

1. Ο κάθε χρήστης του noobcash (το κάθε process δηλαδή) θα έχει ένα noobcash wallet για να να πραγματοποιεί transactions. Το κάθε wallet αποτελείται από: (α) ένα private key γνωστό μόνο στον χρήστη-κάτοχο του wallet, που του επιτρέπει να ξοδεύει χρήματα (να στείλει δλδ χρήματα σε άλλο wallet) και (β) το αντίτοιχο public key, που απαιτείται για να λάβει ο κάτοχος του wallet χρήματα από άλλον χρήστη. Το public key είναι και η διεύθυνση του wallet του χρήστη.

2. Ο κάθε κάτοχος wallet μπορεί να κάνει συναλλαγές ξοδεύοντας NBC (noobcash coins). Αν η Αλίκη θέλει να στείλει 1 NBC στον Μπόμπ, τότε θα δημιουργήσει ένα transaction που θα περιέχει το ποσό που θέλει να στείλει και τη διεύθυνση του wallet του Μπομπ , δηλαδή το public key του Μπομπ. Το transaction αυτό θα το υπογράψει χρησιμοποιώντας το private key της.

3. Το κάθε transaction που δημιουργείται γίνεται broadcast σε όλο το δίκτυο noobcash.

4. Οι miners που λαμβάνουν το νέο transaction το επικυρώνουν: Στο παράδειγμά μας, πρώτα ελέγχουν ότι το transaction προήλθε από την Αλίκη χρησιμοποιώντας το public key της, πιστοποιούν ότι η Αλίκη έχει αρκετά χρήματα στο wallet της για να εκτελέσει το transaction (1 NBC στην περίπτωσή μας) και αν όλοι οι έλεγχοι επιτύχουν, προσθέτουν το transaction στο τρέχον block. Για την παρούσα άσκηση θεωρούμε ότι όλοι οι χρήστες είναι και miners.

5. Όταν το τρέχον block γεμίσει, τότε οι miners ξεκινούν τη διαδικασία mining με proof-of-work. Όποιος βρει το σωστό nonce κάνει broadcast το επικυρωμένο block σε όλο το δίκτυο, ώστε να γίνει γνωστό σε όλους τους nodes που συμμετέχουν.

6. Σε περίπτωση που 2 ή περισσότεροι miners κάνουν ταυτόχρονα mine ένα block, οι παραλήπτες των διαφορετικών αυτών blocks προσθέτουν στην αλυσίδα τους το πρώτο block που λαμβάνουν. Αυτό μπορεί να οδηγήσει σε 2 ή περισσότερες διακλαδώσεις της αλυσίδας. Για να καταλήξουν τελικά όλοι οι κόμβοι με την ίδια αλυσίδα του blockchain, τρέχουν τον αλγόριθμο consensus, σύμφωνα με τον οποίον σε περίπτωση conflict υιοθετούν την αλυσίδα με το μεγαλύτερο μέγεθος.

**Δίκτυο**

Υποθέτουμε ότι το δίκτυό μας έχει μέγεθος n, δηλαδή n κόμβοι συμμετέχουν στο δίκτυο με ids από 0 έως n-1. Ο κάθε κόμβος του δικτύου μας θα πρέπει να μπορεί να ανταλλάσει μηνύματα με όλους τους υπόλοιπους κόμβους του συστήματος γνωρίζοντας την ip address και το port στο οποίο ακούνε. Για λόγους απλότητας, θεωρούμε ότι κάθε κόμβος διατηρεί λίστα με όλους τους υπόλοιπους κόμβους που συμμετέχουν κάθε στιγμή στο noobcash σύστημα. Η επικοινωνία μπορεί να υλοποιηθεί είτε με sockets (TCP ή UDP) είτε με REST api, χρησιμοποιώντας όποια βιβλιοθήκη θέλετε.

Ο πρώτος κόμβος του δικτύου, ο bootstrap node (id 0), είναι και ο κόμβος που δημιουργεί το genesis block, το πρώτο δλδ block του blockchain μας. Στο genesis block θέτουμε previous_hash=1, nonce=0 και η λίστα από transactions περιλαμβάνει μόνο ένα transaction που δίνει στον bootsrap κόμβο 100*n NBC coins από την wallet διεύθυνση 0. Αυτό είναι το μοναδικό block που δεν επαληθεύεται. Σταδιακά προστίθενται οι υπόλοιποι n-1 κόμβοι του δικτύου.

Για να εισαχθεί ένας νέος κόμβος στο σύστημα επικοινωνεί πρώτα με τον bootstrap κόμβο, του οποίου θεωρούμε γνωστά σε όλους τα στοιχεία επικοινωνίας ip address/port, και του στέλνει το public key του wallet του. Από τον bootstrap κόμβο λαμβάνει το μοναδικό id του (ο πρώτος το id 1, ο δεύτερος το id 2 κ.ο.κ) . Όταν εισαχθούν όλοι οι κόμβοι, ο bootstrap κάνει broadcast σε όλους τα ζεύγη ip address/port καθώς και τα public keys των wallets όλων των κόμβων που συμμετέχουν στο σύστημα. Θεωρείστε ότι από εδώ και πέρα δεν υπάρχουν εισαγωγές ή αποχωρήσεις κόμβων από το σύστημα. Επιπλέον, κάθε νέος κόμβος λαμβάνει από τον bootstrap κόμβο το blockchain όπως έχει διαμορφωθεί μέχρι εκείνη τη στιγμή, το κάνει validate και μετά από αυτό το σημείο μπορεί να κάνει transactions.

Ο bootstrap κόμβος, όπως αναφέρθηκε παραπάνω, έχει 100*n NBC coins από το genesis block. Μετά τη σύνδεση κάθε νέου κόμβου στο δίκτυο, ο bootstrap κόμβος εκτελεί ένα transaction όπου του μεταφέρει 100 NBC. Έτσι, μετά την εισαγωγή όλων των κόμβων, ο καθένας τους έχει 100 NBC στο wallet του.

## Backend

Στην συνέχεια αναλύουμε και επεξηγούμε τα βασικά συστατικά του Noobcash Backend αλλά και τον τρόπο λειτουργίας του:

### Wallet

Η κλάση wallet αναφέρεται στο «πορτοφόλι» του κάθε χρήστη που συνδέεται και χρησιμοποιεί το noobcash σύστημα. Τα πεδία της συγκεκριμένης κλάσης είναι τα ακόλουθα:

- *public_key:* Είναι το δημόσιο κλειδί του χρήστη του wallet. Αποτελεί την διεύθυνση του wallet μέσω της οποίας μπορεί οποιοσδήποτε άλλος χρήστης να στείλει χρήματα στον κάτοχο του συγκεκριμένου πορτοφολιού. Την συγκεκριμένη διεύθυνση την γνωρίζουν όλοι οι χρήστες ώστε να έχουν την δυνατότητα εάν το επιθυμούν να μεταφέρουν χρήματα στον χρήστη του wallet αυτού.
- *private_key:* Είναι το ιδιωτικό κλειδί του χρήστη του wallet. Το κλειδί αυτό είναι φανερό μόνο στον κάτοχο του wallet και σε κανέναν άλλον χρήστη. Χρησιμοποιείται προκειμένου ο χρήστης να μπορεί να υπογράφει τα transactions (συναλλαγές) που επιθυμεί να πραγματοποιήσει μεταφέροντας χρήματα σε κάποιον άλλο χρήστη. Με τον τρόπο αυτό εξασφαλίζεται ότι μόνο ο κάτοχος του συγκεκριμένου wallet μπορεί να ξοδέψει χρήματα μεταφέροντας τα σε κάποιο άλλο wallet. Αντίστοιχα οι υπόλοιποι χρήστες έχοντας στην διάθεση τους την διεύθυνση του συγκεκριμένου wallet μπορούν να ελέγξουν ότι πράγματι ένα transaction που ξοδεύει χρήματα από το wallet αυτό έχει δημιουργηθεί από τον κάτοχο του.

### Transaction

Η κλάση transaction περιγράφει την οντότητα ενός transaction το οποίο γίνεται αποκλειστικά μεταξύ δύο κόμβων του συστήματος. Περιέχει στην ουσία τις βασικές πληροφορίες ενός transaction. Πιο συγκεκριμένα:

- *Sender address:* Στο πεδίο αυτό αποθηκεύουμε το public key του χρήστη ο οποίος στέλνει τα νομίσματα.
- *Receiver address:* Στο πεδίο αυτό αποθηκεύουμε το public key του χρήστη ο οποίος λαμβάνει τα νομίσματα.
- *Amount:* Το πεδίο αυτό αντιπροσωπεύει το ποσό νομισμάτων που μεταφέρεται.
- *Timestamp:* Η χρονική στιγμή που δημιουργείται το συγκεκριμένο αντικείμενο που αντιστοιχεί σε ένα transaction.
- *Inputs:* Το πεδίο αυτό αντιστοιχεί σε μία λίστα από Unspented transactions (UTXOs) από τα οποία προέρχεται το ποσό που μεταφέρεται. Τα transactions

αυτά είναι αποκλειστικά unspent transactions του κόμβου ο οποίος αποτελεί τον αποστολέα χρημάτων του συγκεκριμένου transaction. Στην λίστα αυτή αποθηκεύονται μόνο τα Ids των συγκεκριμένων transactions. Απαραίτητη προυπόθεση προκειμένου να γίνει επιτυχώς validate ένα transaction είναι να υπάρχει ο απαραίτητος αριθμός από τέτοια transactions (δηλαδή το άθροισμα των χρημάτων τους να είναι μεγαλύτερο ή ίσο από το πεδίο amount .)

- *Outputs:* Το πεδίο αυτό αντιστοιχεί σε μία λίστα από δύο transactions. Το πρώτο transaction αντιστοιχεί στα χρήματα που μεταφέρονται στον παραλήπτη και το δεύτερο στο υπόλοιπο των χρημάτων που μένουν στον αποστολέα (αφαιρούμε δηλαδή από τα νομίσματα που κατέχει ο αποστολέας μέχρι εκείνη την στιγμή τα νομίσματα που μεταφέρει μέσω του συγκεκριμένου transaction). Η δομή των δύο αυτών transactions είναι η ακόλουθη:
    - Id: Ένα μοναδικό αναγνωριστικό.
    - Transaction_id: Το αναγνωριστικό του transaction από το οποίο προέρχεται.
    - Recipient/Sender: Το public key του χρήστη ο οποίος θα λάβει τα χρήματα από το συγκεκριμένο transaction/ Το public key του χρήστη ο οποίος στέλνει τα χρήματα μέσω του συγκεκριμένου transaction.
    - Amount/Balance: Το ποσό των χρημάτων που μεταφέρονται και τα οποία θα λάβει ο recipient / Το υπόλοιπο των χρημάτων που μένουν στον sender μετά το συγκεκριμένο trasnsaction.

  Όταν ένα transaction γίνει επιτυχώς validate από έναν κόμβο τα δύο αυτά Output Transactions πρστίθενται στην λίστα των Unspented Transactions (UTXOs) του κόμβου αυτού.

- *Transaction_Id:* Το πεδίο αυτό αποτελεί το αναγνωριστικό του transaction. Υπολογίζεται ως το hash των πεδίων Sender address , Receiver address , Amount , Timestamp
- *Signature:* Το πεδίο αυτό αποτελεί την υπογραφή του συγκεκριμένου transaction. Η υπογραφή υπολογίζεται κρυπτογραφώντας το hash (πεδίο Transaction_Id) του συγκεκριμένου Transaction μέσω του private_key του wallet του Sender. Όπως αναφέραμε και στην περίπτωση του wallet η υπογραφή αυτή στέλνεται μαζί με τα υπόλοιπα δεδομένα του transaction σε όλους τους κόμβους και χρησιμοποιείται προκειμένου εκείνοι να ελέγξουν ένα το transaction που ξοδεύει χρήματα από το wallet αυτό έχει δημιουργηθεί από τον κάτοχο του. Ο έλεγχος αυτός πραγματοποιείται με την κλίση στην συνάρτηση Verify_Transaction() την οποία αναλύουμε στην συνέχεια:
- Verify_Transaction(): Η συνάρτηση αυτή λαμβάνοντας την υπογραφή και το hash του συγκεκριμένου transaction αποκρυπτογραφεί την υπογραφή μέσω του public key του sender του transaction αυτού. Εάν η αποκρυπτογράφηση αυτή είναι επιτυχής σημαίνει ότι πράγματι το συγκεκριμένο transaction που

ξοδεύει χρήματα από ένα wallet έχει δημιουργηθεί από τον κάτοχο του wallet αυτού. Η συνάρτηση αυτή επομένως επιστρέφει το hash μετά την αποκρυπτογράφηση.

- *to_send():* Η μέθοδος αυτή μετατρέπει το αντικείμενο transaction σε dictionairy (λεξικό) έχοντας ένα key για κάθε πεδίο του αντικειμένου αυτού και value την αντίστοιχη τιμή του συγκεκριμένου πεδίου. Η συνάρτηση αυτή χρησιμοποιείται όταν χρειάζεται να μεταφέρουμε ένα transaction στους υπόλοιπους κόμβους (broadcast_transaction) μέσω API.

**Block**

Η κλάση αυτή αντιστοιχεί σε ένα block το οποίο ενδέχεται να προστεθεί στο Blockchain. Τα πεδία της κλάσης αυτής είναι τα ακόλουθα:

- *Index:* Ο αύξων αριθμός του block (ξεκινάει με το genesis block το οποίο έχει index = 0)
- *Timestamp:* Το timestamp της δημιουργίας του συγκεκριμένου block
- *listOfTransactions:* Η λίστα με τα transactions που περιέχονται στο block. Η λίστα αυτή αρχικοποιείται σε κενή κατά την δημιουργία του block.
- *Nonce:* Ο αριθμός αυτός αρχικοποιείται σε μηδέν κατά την δημιουργία του Block. Όταν ένα Block γεμίσει (όταν δηλαδή ο αριθμός των transactions που υπάρχουν στην λίστα του είναι ίσος με την σταθερά capacity η οποία ορίζεται κατά την δημιουργία του Blockchain) τότε το block αυτό πρέπει να γίνει mine (καλείται η συνάρτηση mine_block η οποία υπάρχει στο αρχείο node.py). Κατά την διαδικασία του mine ο κόμβος του συγκεκριμένου block αναζητά τον κατάλληλο αριθμό nonce ο οποίος πετυχαίνει το hash του συγκεκριμένου block να ξεκινάει από έναν συγκεκριμένο αριθμό μηδενικών. Ο αριθμός αυτός προσδιορίζεται μέσω της σταθεράς difficulty η οποία και πάλι ορίζεται κατά την δημιουργία του Blockchain. Τελικά λοιπόν το πεδίο nonce περιέχει τον αριθμό αυτόν που αποτελεί λύση του proof of work (η διαδικασία του mine που περιγράψαμε παραπάνω).
- *pervious_hash:* Το πεδίο αυτό περιέχει το hash του προηγούμενου από block στο Blockchain. Κατά την δημιουργία του Block το πεδίο αυτό αποκτά το hash του τελευταίου Block το οποίο βρίσκεται στο Blockchain εκείνη την χρονική στιγμή και δεν τροποποιείται ποτέ.
- *Hash:* Το πεδίο αυτό αναφέρεται στο hash του συγκεκριμένου block το οποίο για να υπολογισθεί χρησιμοποιεί τα πεδία index , nonce , previous_hash , listOfTransactions και Timestamp. Ο υπολογισμός αυτός γίνεται με την κλίση στην μέθοδο myHash() της συγκεκριμένης κλάσης. Το πεδίο hash επαναυπολογίζεται όταν βρεθεί το Nonce του συγκεκριμένου Block μετά την διαδικασία του mine.
- *to_send():* Η μέθοδος αυτή μετατρέπει το αντικείμενο Block σε dictionairy (λεξικό) έχοντας ένα key για κάθε πεδίο του αντικειμένου αυτού και value την

αντίστοιχη τιμή του συγκεκριμένου πεδίου. Η συνάρτηση αυτή χρησιμοποιείται όταν χρειάζεται να μεταφέρουμε ένα transaction στους υπόλοιπους κόμβους (broadcast_transaction) μέσω API. Μάλιστα καλεί και την μέθοδο to_send του αντικειμένου transaction μετατρέποντας όλα τα transactions που υπάρχουν στην λίστα listOfTransactions σε μορφή κατάλληλη για μεταφορά μέσω API.

**Blockchain**

Η κλάση αυτή αντιπροσωπεύει το Blockchain του συστήματος μας. Κάθε κόμβος κρατάει το δικό του Blockchain. Τα πεδία της κλάσης αυτής είναι:

- *List_of_blocks:* Η λίστα με τα επικυρωμένα blocks τα οποία έχουν εισέλθει στο Blockhain.
- *Difficulty:* Ο αριθμός των μηδενικών από τα οποία πρέπει να αποτελείται το Hash ενός block προκειμένου να ολοκληρωθεί η διαδικασία του mine και το block να είναι έτοιμο να γίνει broadcast στους υπόλοιπους κόμβους. Όσο μεγαλύτερος είναι αυτός ο αριθμός τόσο περισσότερο χρόνο απαιτείται προκειμένου να ολοκληρωθεί η διαδικασία mine του κάθε block.
- *Capacity:* Ο αριθμός των transactions που χωρούν σε ένα block. Προκειμένου ένα block να προχωρήσει στην διαδικασία του mine απαιτείται να είναι γεμάτο.
- *to_send():* Η μέθοδος αυτή μετατρέπει το αντικείμενο blockchain σε dictionairy (λεξικό) έχοντας ως key το sting "blockchain" και ως value την λίστα με τα blocks του συγκεκριμένου blockchain. Για το κάθε block καλέιται η μέθοδος to_send() την οποία επεξηγήσαμε στην κλάση Block. Έτσι πλέον το Blockchain είναι σε μορφή κατάλληλη για μεταφορά μέσω API

**Node**

Η κλάση αυτή είναι ο πυρήνας του συστήματος μας, στην ουσία αντιπροσωπεύει κάθε χρήστη και τις διάφορες ενέργειες που πρέπει να εκτελεί για να αλληλεπιδρά με τους υπόλοιπους κόμβους και να απαντά στα αιτήματα που στέλνονται. Στην κλάση αυτή συνδέονται όλες οι επιμέρους κλάσεις και αποκτούν νόημα όλες οι επιμέρους συναρτήσεις. Συγκεκριμένα ο κάθε χρήστης αποτελείται από τα εξής επιμέρους αντικείμενα. Ένα blockchain το οποίο θα γεμίζει με τα διάφορα blocks. Ένα wallet το οποίο περιέχει πληροφορίες όπως η διεύθυνση και το private key. Και διάφορες άλλες μεταβλητές οι οποίες κρατάνε πληροφορίες όπως τα Unspent Output Transactions (UTXOS) όλων των κόμβων του συστήματος, το τρέχων block το οποίο σταδιακά γεμίζει με transactions, το id του κόμβου και την λίστα ring που περιέχει όλες τις πληροφορίες για όλους τους κόμβους που έχουν εισαχθεί στο σύστημα. Παράλληλα στο node έχουν υλοποιηθεί διάφορες συναρτήσεις οι οποίες υλοποιούν

την λογική εισαγωγής ενός κόμβου στο σύστημα, δημιουργίας transaction, επαλήθευσης transaction, γέμισμα ενός block και στην συνέχεια γέμισμα του blockchain και διάφορα άλλα. Για να γίνει πιο ξεκάθαρη η κάθε συνάρτηση θα περιγράψουμε στην συνέχεια την διαδικασία και την σειρά με την οποία συμβαίνουν τα διάφορα γεγονότα στο σύστημα μας και πως ένας κόμβος ανταποκρίνεται σε αυτά. Αρχικά στο σύστημα μας εισάγεται ο bootstrap κόμβος και δημιουργείται ένα node το οποίο αντιπροσωπεύει τον κόμβο αυτό. Κάθε φορά που κάποιος άλλος κόμβος εισέρχεται στο σύστημα αποκτά και αυτός με την σειρά του ένα αντικείμενο τύπου node και στέλνει αίτημα στον bootstrap κόμβο μεταφέροντας του τα στοιχεία του (wallet_address , ip , port). Ο bootstrap με την σειρά του αποθηκεύει τα στοιχεία του νέου κόμβου στην λίστα ring και δημιουργεί transaction με το οποίο του δίνει 100 NBCs. Καθώς όλοι οι κόμβοι συνδέονται με την σειρά στο σύστημα ο bootstrap συλλέγει τις πληροφορίες του καθενός ενώ παράλληλα δημιουργεί τα αντίστοιχα transactions. Όταν όλοι οι κόμβοι εισαχθούν στο σύστημα ο bootstrap κόμβος συμπληρώνει το blockchain του με τα γεμάτα blocks που δημιουργήθηκαν κατά την διάρκεια εισαγωγής των κόμβων (αφού πρώτα τα έχει επικυρώσει). Στην συνέχεια στέλνει σε όλους τους κόμβους συγκεντρωτικά όλα τα στοιχεία που έχει μαζέψει μέχρι εκείνη την στιγμή. Τα στοιχεία που γίνονται broadcast είναι τα utxos, το blockchain όπως έχει διαμορφωθεί μέχρι εκείνη την στιγμή και η λίστα ring με τα στοιχεία για τον κάθε κόμβο που συμμετέχει στο σύστημα. Οι επιμέρους κόμβοι λαμβάνουν όλα αυτά τα στοιχεία από τον bootstrap και με την σειρά τους κάνουν validate το chain που έλαβαν καλώντας την συνάρτηση valid_chain η οποία ελέγχει το κάθε block ξεχωριστά εκτός από το genesis block. Πλέον όλοι οι κόμβοι είναι έτοιμοι να δεχθούν καινούργια transactions. Όταν ένας κόμβος θέλει να δημιουργήσει ένα transaction καλεί την συνάρτηση create transaction η οποία με την σειρά της δημιουργήσει τα inputs και τα outputs του transaction αυτού, δηλαδή στην ουσία καταγράφει τις αλλαγές που θα επιφέρει στο balance των εμπλεκόμενων nodes το transaction αυτό. Αφου δημιουργηθούν τα inputs και τα outputs το transaction γίνεται broadcast σε όλους τους κόμβους οι οποίοι με την σειρά τους πρέπει να το κάνουν validate. Για να γίνει validate ενα transaction καλείται η συνάρτηση validate_transaction(). Στην συνάρτηση αυτή επαληθεύουμε την υπογραφή και τα αντίστοιχα inputs/outputs (μέσω των utxos) για να εξασφαλίσουμε ότι ο wallet αποστολέας έχει το ποσό amount που μεταφέρει στον παραλήπτη. Καθώς τα transactions γίνονται broadcast και επαληθεύονται από όλους τους επιμέρους κόμβους εισάγονται σε blocks. Όταν ένα block φτάσει στο όριο του capacity του αυτό μπαίνει στην διαδικασία του mining. Κατά την διαδικασία του mining ο κόμβος αναζητά έναν αριθμό Nonce ο οποίος θα έχει ως αποτέλεσμα κάποιοι από τους πρώτους χαρακτήρες του hash του συγκεκριμένου block να είναι “0”. Η σταθερά difficulty όπως έχουμε αναφέρει ξανά ορίζει το πόσοι από αυτούς τους χαρακτήρες πρέπει να είναι “0”. Όταν ολοκληρωθεί η διαδικασία του mining (βρεθεί το κατάλληλο nonce) τότε το block γίνεται broadcast σε όλους τους υπόλοιπους κόμβους. Ο κάθε κόμβος ξεχωριστά λαμβάνει το block που του έγινε broadcast και με την σειρά του το κάνει validate. Για να γίνει το validate του block καλείται η συνάρτηση validate_block_and_add(). Η συνάρτηση αυτή επαληθεύει ότι το πεδίο

current_hash είναι πράγματι σωστό και ότι το πεδίο previous_hash ισούται πράγματι με το hash του προηγούμενου block που βρίσκεται εκείνη την στιγμή στο blockchain. Σε περίπτωση που κάποιος κόμβος λάβει ένα block το οποίο δεν μπορεί να κάνει validate γιατί το πεδίο previous_hash δεν ισούται με το hash του προηγούμενου block καλείται η συνάρτηση resolve_conflict(). Το παραπάνω γεγονός μπορεί να σημαίνει ότι έχει δημιουργηθεί κάποια διακλάδωση, η οποία πρέπει να επιλυθεί. Ο κόμβος ρωτάει τους υπόλοιπους για το μήκος του blockchain και επιλέγει να υιοθετήσει αυτό με το μεγαλύτερο μήκος. Παράλληλα με το blockchain ο κόμβος υιοθετεί και τα utxos του αντίστοιχου αποστολέα του Blockchain ώστε να ενημερώσει κατάλληλα το υπόλοιπο του το οποίο πρέπει να συμβαδίζει με το Blockchain το οποίο έχει υιοθετήσει (θυμίζουμε ότι το υπόλοιπο ενός κόμβου υπολογίζεται ως το άθροισμα των UTXOS τα οποία αναφέρονται στον συγκεκριμένο κόμβο) Η παραπάνω διαδικασία επαναλαμβάνεται κάθε φορά που δημιουργούνται transactions απο τους κόμβους που εμπλέκονται στο σύστημα.

**Rest**

Στο αρχείο αυτό υλοποιούμε το API το οποίο περιέχει τις αναγκαίες λειτουργικότητες έτσι ώστε οι κόμβοι μεταξύ τους να ανταλλάσουν πληροφορίες και να επικοινωνούν. Πιο συγκεκριμένα τα endpoints τα οποία υπάρχουν είναι τα εξής:

- */broadcast/transaction:* το endpoint αυτό χρησιμοποιείται από ένα κόμβο όταν δημιουργήσει ένα transaction για να το κάνει broadcast σε όλους τους υπόλοιπους.

- */add/node:* Όταν ένας κόμβος εισάγεται στο σύστημα καλεί το endpoint αυτό για να καταγραφούν τα στοιχεία του απο τον bootstrap κόμβο.

- */broadcast/ring:* Το endpoint αυτό καλείται απο τον bootstrap κόμβο αφού όλοι οι κόμβοι εισαχθούν στο σύστημα ο ο οποίος μεταφέρεται παράλληλα το blockchain, το ring και τα utxos του bootstrap όπως έχουν δημιουργηθεί μέχρι την δεδομένη στιγμή.

- */broadcast/block:* Όταν ένας κόμβος έχει κάνει mine ένα block χρησιμοποιεί το endpoint αυτό για να το κάνει broadcast σε όλους τους υπόλοιπους.

- */add/transaction:* Το συγκεκριμένο endpoint καλείται απο το cli όταν θέλουμε να δημιουργήσουμε καινούργιο transaction με αποστολέα κάποιον από τους εμπλεκόμενους κόμβους.

- */balance:* Το endpoint αυτό καλείται απο το cli και καλώντας την συνάρτηση wallet_balance() επιστρέφει τον αριθμό των coins που έχει ο κόμβος στο πορτοφόλι του.

- */get/lengths:* Αυτό το endpoint μας επιστρέφει το μήκος του blockchain του αντίστοιχου κόμβου και χρησιμοποιείται σε περίπτωση conflict ώστε να βρεθεί ποιο είναι το μακρύτερο blockchain.

- */getblock:* Μας επιστρέφει το hash και το previous hash του κάθε block του blockchain στο κόμβο που βρισκόμαστε.

- *resolve/conflict:* Μας επιστρέφει το blockchain του κόμβου μαζί με τα utxos του και χρησιμοποιείται προκειμένου να αντιγραφούν τα συγκεκριμένα πεδία του μακρύτερου blockchain ώστε να επιλυθεί το conflict του αντίστοιχου κόμβου.

- */allbalance:* Μας επιστρέφει τα balances των wallets όλων των κόμβων ώστε να βεβαιωθούμε ότι τελικά μετά τα πειράματα το άθροισμα όλων των balances παραμένει ίδιο με το αρχικό του bootstrap.

- */last/block:* Μας επιστρέφει τα transactions που έχουν αποθηκευτεί στο τελευταίο block του blockchain κάτι το οποίο καλείται όταν τρεχτεί η εντολή view από το cli.

- */experiments/<float:time>:* Μας επιστρέφει το throughput και το block time τα οποίο ζητούνται μετά τα πειράματα. Στην αρχή του rest δημιουργείται ο κάθε κόμβος ο οποίος θα εισαχθεί στο σύστημα.

**CLI**

Το CLI δημιουργήθηκε προκειμένου να έχει την δυνατότητα ο χρήστης να επικοινωνεί με το Noobcash σύστημα ζητώντας από αυτό να εκτελέσει κάποιο transaction αλλά και αποκτώντας πληροφορίες σχετικά με την κατάσταση του συστήματος. Για να το επιτύχει αυτό εκτελεί το script python cli.py δίνοντας μία από τις ακόλουθες παραμέτρους:

*--t <recipient_node_id> <amount>:* Με την εντολή αυτή ο χρήστης του αντίστοιχου κόμβου ζητάει από το σύστημα να μεταφέρει <amount> χρήματα στον χρήστη του κόμβου με id = recipient_node_id. Προκειμένου να πραγματοποιηθεί η διαδικασία αυτή πραγματοποιείται από το cli αίτημα στο endpoint */add/transaction* το οποίο στην συνέχεια αναλαμβάνει όλη την διαδικασία δημιουργίας και προώθησης του transaction που περιγράψαμε κατά την επεξήγηση της κλάσης Node.

--view: Με την εντολή αυτή ο χρήστης μπορεί να δει τα δεδομένα του τελευταίου Block που υπάρχει εκείνη την χρονική στιγμή στο Blockchain του κόμβου στον οποίο βρίσκεται. Προκειμένου να πραγματοποιηθεί η διαδικασία αυτή πραγματοποιείται από το cli αίτημα στο endpoint /last/block το οποίο αναλαμβάνει να επιστρέψει το τελευταίο block του Blockchain του συγκεκριμένου κόμβου.

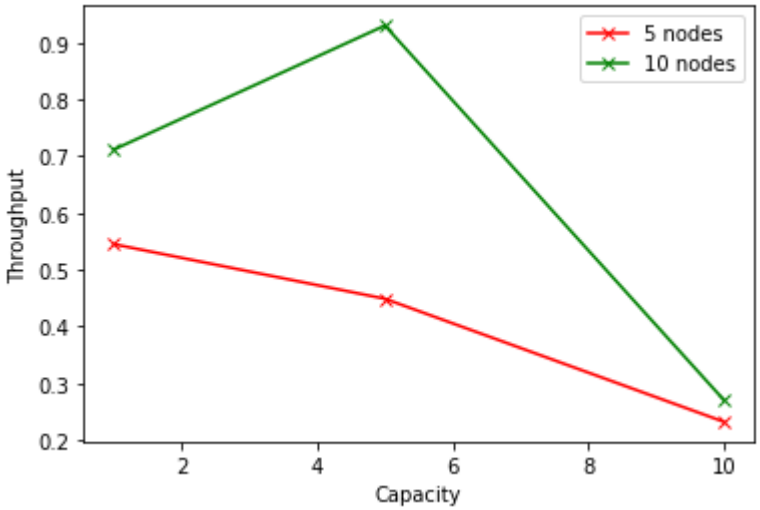
--balance: Με την εντολή αυτή ο χρήστης μπορεί να δει το υπόλοιπο των χρημάτων του εκείνη την χρονική στιγμή. Για να πραγματοποιηθεί αυτό το γίνεται αίτημα στο endpoint /balance. Το endpoint αυτό αθροίζει τα χρήματα των utxos που αναφέρονται στον συγκεκριμένο κόμβο και επιστρέφει στο χρήστη το άθροισμα αυτό το οποίο αποτελεί και το υπόλοιπο των χρημάτων που διαθέτει ο χρήστης.

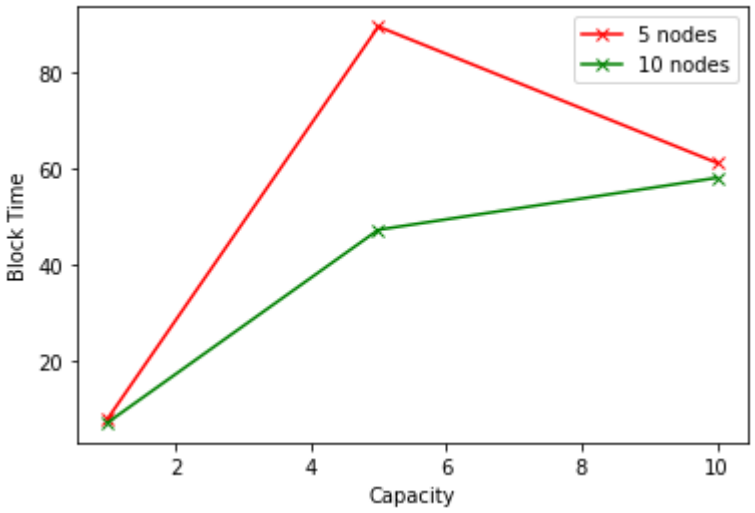
--allbalance: Με την εντολή αυτή ο χρήστης μπορεί να δει το υπόλοιπο των χρημάτων όλων των κόμβων που συμμετέχουν στο σύστημα. Για να πραγματοποιηθεί αυτό το γίνεται αίτημα στο endpoint /allbalance. Το endpoint αυτό αθροίζει για κάθε κόμβο του συστήματος τα χρήματα των utxos που αναφέρονται σε αυτόν και τα επιστρέφει στον χρήστη.

--help: Με την εντολή αυτή ο χρήστης μπορεί να λάβει επεξηγηματικές πληροφορίες για τις παραπάνω εντολές.

**Πειράματα**

Για να εκτελέσουμε τα πειράματα δημιουργούμε σε κάθε VM το script start.py. Στο συγκεκριμένο script διαβάζουμε τα αντίστοιχα αρχεία του κάθε πειράματος και δημιουργούμε τα transactions τα οποία αναγράφονται. Έπειτα καλούμε το endpoint /create/transaction του αντίστοιχου κόμβου ώστε να δημιουργηθούν και να εκτελεστούν τα αντίστοιχα transactions που αναφέρονται στο αρχείο. Στα πειράματα ζητούνται να μετρηθούν τα Throughput και Block Time. Πιο συγκεκριμένα το Throughput ορίζεται ως η ρυθμαπόδοση του συστήματός μας, δηλαδή πόσα transactions εξυπηρετούνται στην μονάδα του χρόνου. Το Block Time ορίζεται ως ο μέσος χρόνος που χρειάζεται ένα block για να γίνει validate. Μετρώντας τα παραπάνω για 5 και 10 κόμβους , difficulty: 4 και capacity: 1,5,10 παίρνουμε τα εξής αποτελέσματα:






Σχολιασμός πειραμάτων:

Παρατηρώντας τις παραπάνω γραφικές παραστάσεις καταλήγουμε στα ακόλουθα συμπεράσματα:

- Παρατηρούμε ότι μεγαλύτερες τιμές block time παρουσιάζει η περίπτωση με τους 5 κόμβους σε σχέση με την περίπτωση των 10 κόμβων για την ίδια τιμή capacity. Κάτι τέτοιο είναι λογικό καθώς το mining γίνεται πιο γρήγορα στην περίπτωση που έχουμε περισσότερους miners. Στην περίπτωση μας και οι 10 κόμβοι είναι miners κάτι που επαληθεύει τον ισχυρισμό μας.
- Επιπλέον όσο αυξάνεται το capacity αυξάνεται και το block time καθώς απαιτούνται περισσότερα transactions προκειμένου να γεμίσει το block και να γίνει mine.
- Παρατηρούμε ότι μεγαλύτερες τιμές throughput παρουσιάζει η περίπτωση με τους 10 κόμβους αντί για 5 για την ίδια τιμή capacity. Κάτι τέτοιο όπως εξηγήσαμε και παραπάνω είναι λογικό αφού όσο περισσότεροι είναι οι κόμβοι τόσο περισσότερα transactions μπορούν να εξυπηρετηθούν και να γίνουν τα block στα οποία βρίσκονται mine.
- Στην περίπτωση του throughput όμως ενώ θα περιμέναμε ότι η αύξηση του capacity θα οδηγήσει στην μείωση του και στις δύο περιπτώσεις κάτι τέτοιο δεν συμβαίνει. Παρότι η αύξηση του capacity οδηγεί και στην αύξηση του χρόνου προκειμένου ένα block να γεμίσει και άρα να πραγματοποιηθεί ένα transaction το μικρό capacity έχουμε έναν αρκετά μεγάλο χρόνο να αφιερώνεται στο mining καθώς για κάθε transaction ο κάθε κόμβος πρέπει να κάνει mine. Κάτι τέτοιο ίσως είναι η αιτία που δεν παρατηρούμε την αναμενόμενη συμπεριφορά σε αυτήν την περίπτωση.

Τέλος να σημειωθεί ότι δεν πραγματοποιήθηκαν μετρήσεις για την περίπτωση όπου difficulty = 5 καθώς ο χρόνος διεξαγωγής τους ήταν πάρα πολύ μεγάλος. Θεωρούμε όμως πως η αύξηση του difficulty θα οδηγήσει σε μεγάλη αύξηση του block time και μείωση του throughput καθώς πλέον γίνεται αρκετά πιο δύσκολο να βρεθεί το κατάλληλο nonce που θα ολοκληρώσει επιτυχώς το mining.