Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εξαμηνιαία Εργασία
Κατανεμημένα Συστήματα

Τσιλιμιγκουνάκης Μιχαήλ
el19001

Στεφανής Παναγιώτης
el19096

Σωτηρόπουλος Γιώργος
el19848

9ο Εξάμηνο

Ακαδημαϊκό έτος 2023 – 2024

Αθήνα, Μάρτιος 2024

Περιεχόμενα

GitHub Repository: https://github.com/EEMplekei/BlockChat

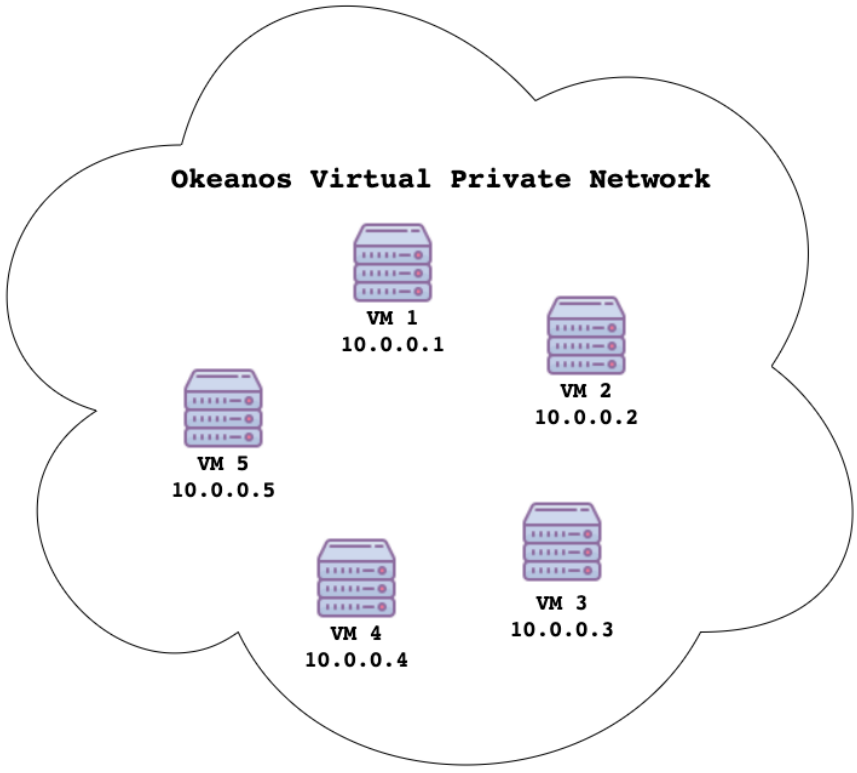
Εισαγωγή

Στα πλαίσια της εξαμηνιαίας εργασίας του μαθήματος υλοποιήσαμε μια πλατφόρμα ανταλλαγής μηνυμάτων αλλά και καταγραφής δοσοληψίων που στηρίζεται στην τεχνολογία blockchain, την πλατφόρμα BlockChat. Για το consensus protocol εφαρμόστηκε ο αλγόριθμος Proof-of-Stake και για το μοντέλο δεδομένων εφαρμόστηκε το account model. (Γενικά το σύστημα BlockChat υλοποιήθηκε σύμφωνα με όλες τις προδιαγραφές της εκφώνησης αλλά και των περεταίρω διευκρινήσεων που δόθηκαν στην πορεία.)

Το Δίκτυο

Ο κάθε node στο δίκτυο αναγνωρίζεται (δικτυακά) μοναδικά από το ζεύγος ip, port. Η επικοινωνία μεταξύ των nodes υλοποιήθηκε με τεχνολογία REST API χρησιμοποιώντας την βιβλιοθήκη FastAPI της Python3.

Χρησιμοποιήσαμε πόρους του Okeanos Knossos έτσι ώστε να δημιουργήσουμε 5 virtual machines όπου επικοινωνούν εσωτερικά με τον εξής τρόπο:



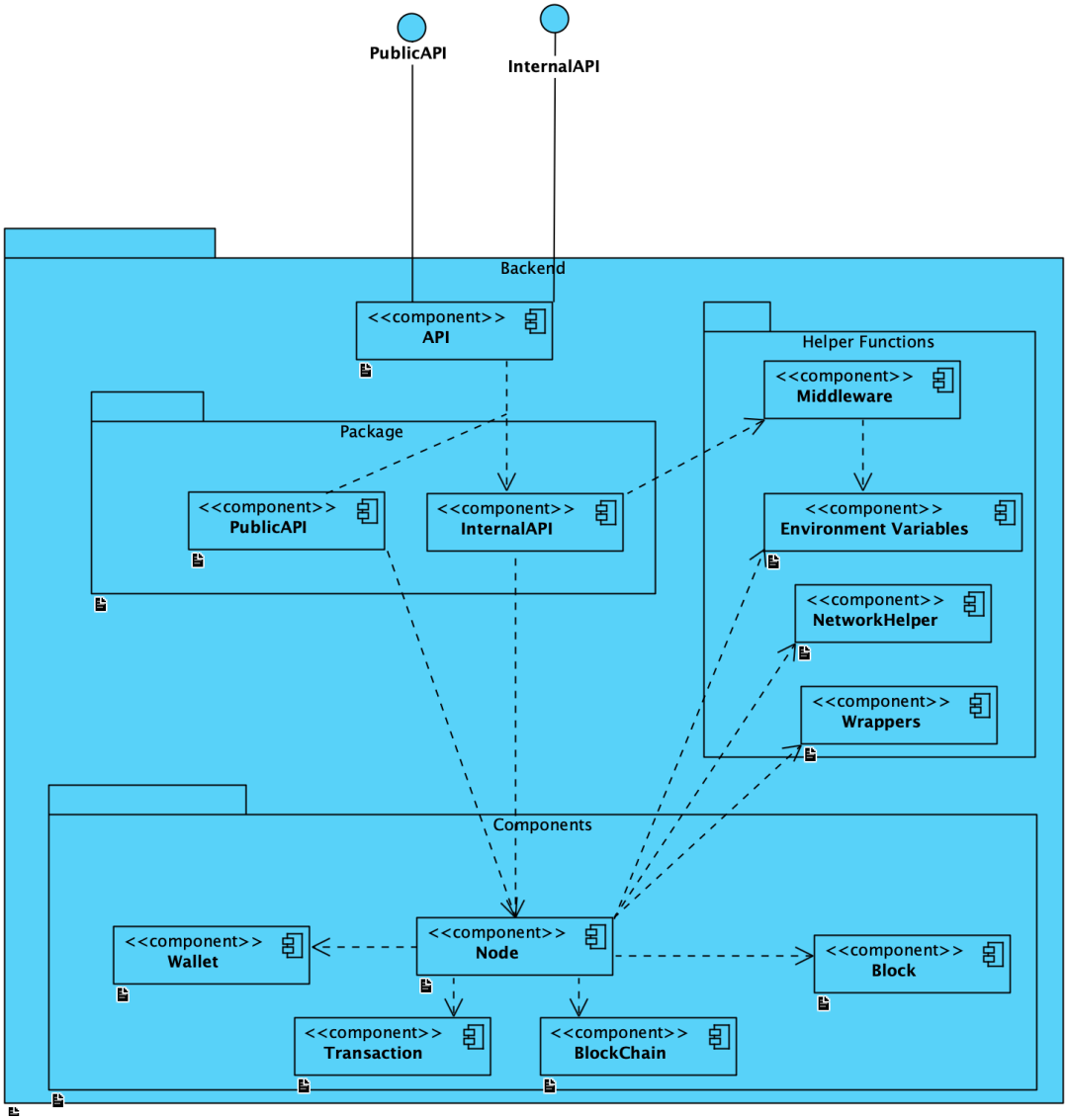
Σε κάθε περίπτωση κάθε Instance του API υλοποιεί ένα node. Στην περίπτωση των 5 nodes έχουμε ένα instance του API να τρέχει στο port 8000 κάθε node, και στην περίπτωση των 10 nodes 2 instances του API σε κάθε node, το ένα στο port 8000 και το άλλο στο port 8001.

GitHub Repository: https://github.com/EEMplekei/BlockChat

Σχεδιασμός Συστήματος

Ο κάθε χρήστης του BlockChat λαμβάνει ένα αντίγραφο του κώδικα του API και του CLI. Τρέχοντας το API ο χρήστης αυτός συνδέεται στο δίκτυο του BlockChat και αποκτά wallet και συμμετέχει σε όλες τις λειτουργίες του blockchain (validation, επαλήθευση transactions τρίτων, λήψη όλου του blockchain κλπ). Μέσω του CLI ο χρήστης έχει την ικανότητα να δημιουργεί νέα transactions, να θέτει το stake του για τον αλγόριθμο Proof-of-Stake αλλά και να βλέπει όλο το blockchain, το τελευταίο block, το balance του και τα incoming messages.

Ο ακριβής σχεδιασμός του συστήματος ενός κόμβου φαίνεται παρακάτω στο Component Diagram που παραθέτουμε, ωστόσο περισσότερη λεπτομέρεια υπάρχει στο αρχείο Visual Paradigm που επισυνάπτεται:



GitHub Repository: https://github.com/EEMplekei/BlockChat

API Endpoints

Τα endpoints του συστήματος μας χωρίζονται σε δύο κατηγορίες, τα Public και τα Internal. Τα public endpoints καλούνται από το CLI και δέχονται requests από οποιαδήποτε διεύθυνση, δηλαδή είναι endpoints τα οποία τα καλεί ένας χρήστης του συστήματος. Τα internal endpoints από την άλλη χρησιμοποιούνται για την επικοινωνία μεταξύ των κόμβων του blockchain και δέχονται κλήσεις μόνο από διευθύνσεις που ανήκουν στο εσωτερικό δίκτυο του blockchain, στην προκειμένη το 10.0.0.0/24 (δεν μπορούν να καλεστούν από χρήστη του συστήματος).

Public API Endpoints

- POST: /api/create_transaction
  Endpoint που περιμένει ένα JSON που περιγράφει ένα transaction, είτε μεταφορά BCCs είτε μήνυμα.

- POST: /api/set_stake
  Endpoint για τον ορισμό του ποσού από τα διαθέσιμα BCCS του εκάστοτε wallet που θα χρησιμοποιηθεί για την συμμετοχή στο Proof-Of-Stake. Το ποσό μένει δεσμευμένο μέχρι να οριστεί νέα stake. Τον unstake υλοποιείται με αποστολή μηδενικού stake σε αυτό το endpoint

- GET: /api/view_last_block
  Endpoint που επιστρέφει το τελευταίο validated block του blockchain. Περιέχει το hash του block, το hash του προηγούμενου block, το timestamp του, τον validator που το δημιούργησε, τα συνολικά fees που αποδόθηκαν στον validator και μια λίστα από transaction που περιείχε. Επιστρέφεται σε μορφή JSON.

- GET: /api/get_balance
  Endpoint που επιστρέφει το διαθέσιμο υπόλοιπο σε BCCs όπως αυτό έχει διαμορφωθεί από τα μέχρι τώρα validated blocks στο blockchain.

- GET: /api/get_temp_balance
  Endpoint που επιστρέφει το προσωρινό υπόλοιπο όπως αυτό έχει διαμορφωθεί από τα μέχρι τώρα transactions που έχει λάβει ο κόμβος, χωρίς απαραίτητα να έχουν μπει ακόμα σε κάποιο block.

- GET: /api/get_chain_length
  Endpoint που επιστρέφει το μήκος της αλυσίδας του blockchain μέχρι την τρέχουσα χρονική στιγμή.

- GET: /api/get_chain
  Endpoint που επιστρέφει μια λίστα με όλα τα blocks του blockchain μέχρι τώρα. Κάθε blocks έχει την μορφή που περιγράφηκε στο endpoint '/view_last_block'

- GET: /api/get_wallet_transaction_list
  Endpoint που επιστρέφει μια λίστα με transactions που αφορούν τον εκάστοτε

κόμβο (είναι είτε παραλήπτης είτε αποστολέας τους).

- GET: /api/ view_ring
  Endpoint που επιστρέφει μια εσωτερική δομή του node (ring) το οποίο κρατάει πληροφορία για κάθε άλλο κόμβο στον blockchain. Η πληροφορία αυτή για κάθε κόμβο περιέχει το address του (public key), την IP και το port του, το ποσό που έχει κάνει stake την τρέχουσα χρονική στιγμή, το υπόλοιπο του και το προσωρινό υπόλοιπο. Επιστρέφεται μια λίστα τέτοιων αντικειμένων.

- GET: /api/get_pending_list_length
  Endpoint που επιστρέφει το πλήθος των transaction που περιμένουν να μπουν σε validated block.

Internal API Endpoints

- POST: /receive_ring
  Endpoint που δέχεται το ring όπως αυτό έχει διαμορφωθεί από τον bootstrap κόμβο αφού εισέλθουν όλοι οι υπόλοιποι. Το ring στέλνεται σαν serialized μορφή του αντίστοιχου python object.

- POST: /receive_blockchain
  Endpoint που δέχεται το blockchain όπως αυτό έχει διαμορφωθεί μέχρι εκείνη την στιγμή. Στέλνεται από τον bootstrap όταν συνδεθούν όλοι οι κόμβοι.

- POST: /receive_transaction
  Endpoint που δέχεται ένα transaction. Το transaction είναι serialized μορφή του αντίστοιχου python object. Στέλνεται από έναν κόμβο στον οποίο δημιουργήθηκε η συναλλαγή από το αντίστοιχο public endpoint, κατά την διαδικασία του broadcast της συναλλαγής αυτής στους υπόλοιπους κόμβους.

- POST: /receive_block
  Endpoint που δέχεται ένα block σε serialized μορφή από το αντίστοιχο object. Στέλνεται από τον validator του κάθε block μετά την δημιουργία του.

- POST: /find_validator
  Endpoint που προτρέπει τον παραλήπτη να εκλέξει validator για το επόμενο block. Καλείται με το που εισέλθουν όλοι οι κόμβοι στο δίκτυο.

- POST: /release_lock
  Endpoint που χρησιμοποιείται για να σημάνει ο validator στους υπόλοιπους κόμβους ότι έστειλε παντού το validated block. Χρησιμοποιείται για τον συγχρονισμό των nodes ώστε να μην έχουμε φαινόμενα όπου ένας κόμβος που δεν είναι ο validator του τρέχοντος block, συνεχίσει να δημιουργήσει το επόμενο πριν λάβει το τρέχον.

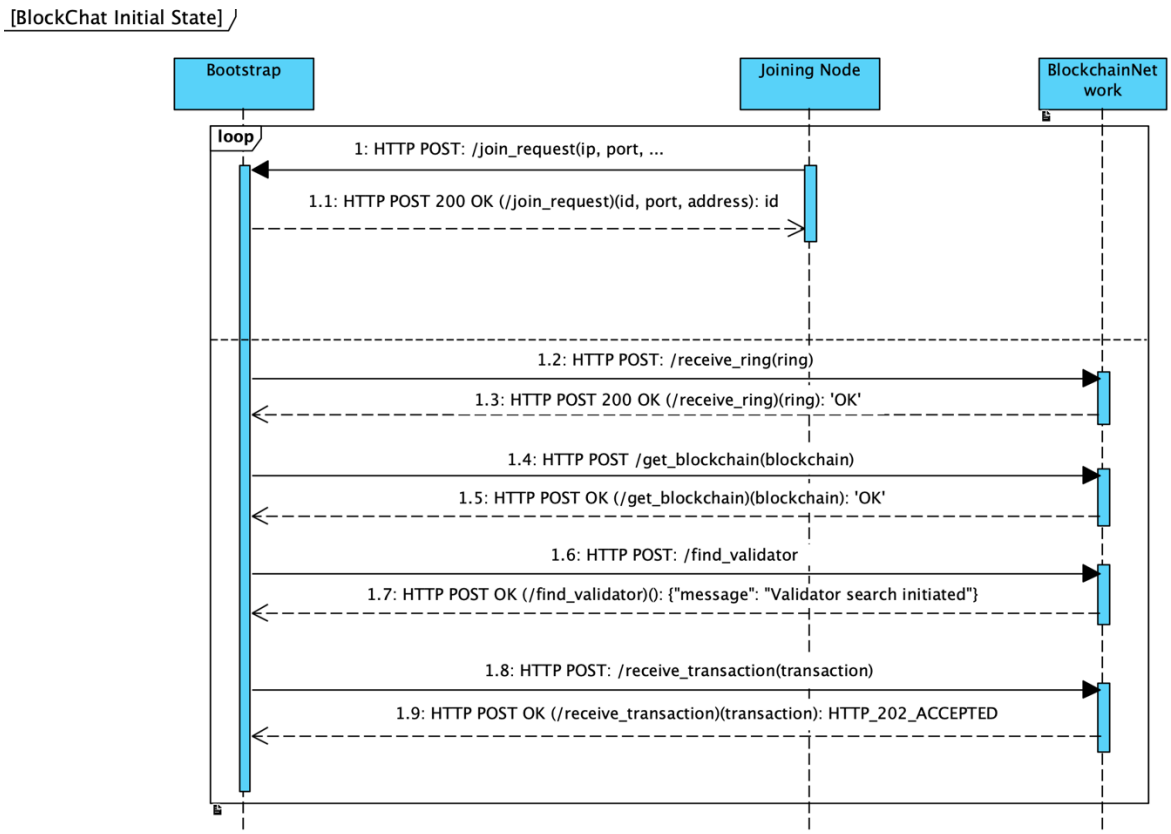
- POST: /join_request
  Endpoint που χρησιμοποιείται από τους κόμβος για να ζητήσουν στον validator να εισέλθουν στο δίκτυο. Δέχεται serialized δεδομένα που περιέχουν την IP, το port και το address του κόμβου που θέλει να εισέλθει.

Περεταίρω πληροφορίες για το API μπορούν να βρεθούν στο SwaggerHub.

Φτάνοντας στο Initial State

Για να φτάσουμε στο initial state όπου όλοι οι κόμβοι έχουν 1000BCC αρκεί να τρέξουμε το API στον κόμβο bootstrap (με διεύθυνση που περιγράφεται στο .env αρχείο, στην προκείμενη *10.0.0.1:8000*) και έπειτα να τρέξουμε διαδοχικά το ένα μετά το άλλο API για τους υπόλοιπους κόμβους.

Αναλυτικά, η επικοινωνία μεταξύ των κόμβων, που συμβαίνει στο παρασκήνιο, προκειμένου να φτάσουμε στο initial state φαίνεται από το παρακάτω sequence diagram:



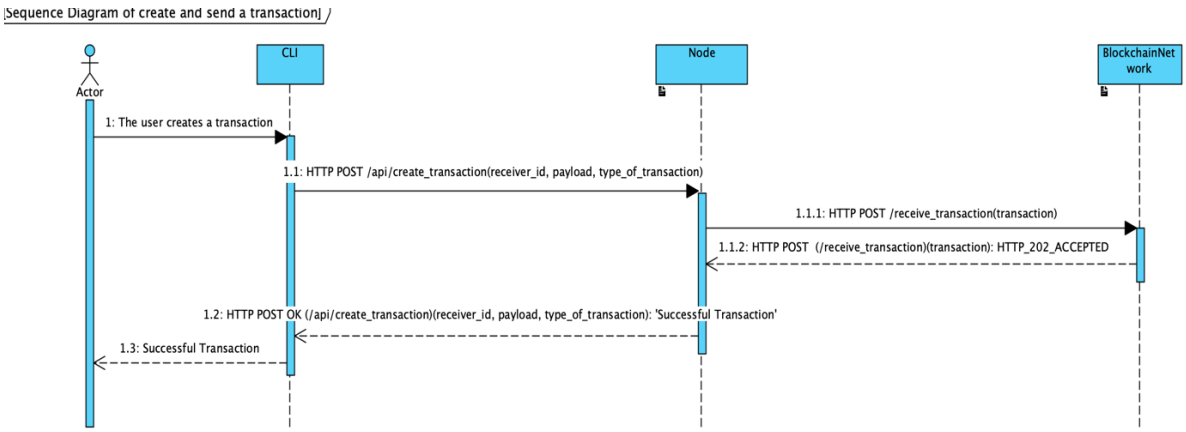
Η διαδικασία ενός transaction

Ένα transaction δημιουργείται μέσω του Command Line Interface (CLI) από έναν χρήστη σε περίπτωση που επιθυμεί να εκτελέσει μια σειρά από ενέργειες. Αυτές οι ενέργειες μπορεί να περιλαμβάνουν την αποστολή ενός μηνύματος, τη μεταφορά νομισμάτων (coins), ή τη συμμετοχή στο consensus protocol δεσμεύοντας νομίσματα για staking [Η διαδικασία του staking υλοποιείται με ειδική κατηγορία transaction *stake*].

Η δημιουργία ενός transaction έχει την εξής διαδικασία:

- Δημιουργία από τον κόμβο αποστολέα
- Broadcast σε όλους τους υπόλοιπους κόμβους
- Validation από τους υπόλοιπους κόμβους, ελέγχοντας τόσο την ταυτότητα του αποστολέα (signature verification) όσο και το διαθέσιμο υπόλοιπο του

Αναλυτικά, η επικοινωνία μεταξύ των κόμβων, που συμβαίνει στο παρασκήνιο, προκειμένου να δημιουργηθεί επιτυχώς ένα transaction φαίνεται στο παρακάτω Sequence Diagram:



Soft State

Κάθε κόμβος του συστήματος έχει μία εσωτερική δομή όπου κρατάει pending transactions τα οποία δεν έχουν εισαχθεί ακόμα σε validated block. Ταυτόχρονα έχει και την πληροφορία του balance και του temp balance.

- balance: Τα υπόλοιπα BCCs ενός χρήστη σύμφωνα με το τελευταίο validated block.
- temp balance: Το balance ενός χρήστη μειωμένο ή αυξημένο κατά το ποσό που υποδεικνύουν τα transactions του pending transactions. Δηλαδή το πραγματικό διαθέσιμο υπόλοιπο την δεδομένη στιγμή.

Μέσω του συνδυασμού της δομής των pending transactions και της πληροφορίας του temp balance επιτυγχάνουμε ένα soft state του current block. Αυτό είναι απαραίτητο καθώς πολλές φορές λόγω παράλληλης δημιουργίας transactions οι εσωτερικές λίστες pending transactions δεν είναι ίδιες (με την ίδια σειρά) μεταξύ των κόμβων. Έτσι, ο validator μπορεί να προσθέσει ένα block στο blockchain το οποίο δεν έχει τις ίδιες πρώτες *k* transactions που έχει ένας άλλος κόμβος (που θα έβαζε αυτός εάν ήταν validator). Αυτό αντιμετωπίζεται με τον εξής τρόπο: κάθε φορά που λαμβάνουν ένα νέο validated block, όλοι οι κόμβοι αφαιρούν από την λίστα pending transactions τις transactions που εμπεριέχονται στο block αυτό. Έτσι, εφόσον υποθέτουμε ότι δεν έχουμε δικτυακά ή βυζαντινά λάθη και transactions δεν χάνονται κατά τα broadcast, όλες οι transactions θα μπουν σε ένα block αργά ή γρήγορα.

Η εκλογή του Validator

Η εκλογή του validator γίνεται με βάση τον αλγόριθμο Proof-of-Stake. Πρακτικά, κάθε χρήστης έχει το δικαίωμα να δεσμεύσει ένα ποσό coins από το διαθέσιμο υπόλοιπο του, το οποίο θα του δώσει ανάλογες πιθανότητες στην λοταρία που θα τρέξει για το ποιος θα είναι ο validator του block (και θα πάρει τα αντίστοιχα fees του block). Για παράδειγμα, αν το συνολικό staking όλων των κόμβων είναι 100

BCCs και ο κόμβος 1 έχει δεσμεύσει για staking 10 BCCs τότε έχει 10% πιθανότητα να είναι validator του block. Στην περίπτωση που κανένας κόμβος δεν έχει δεσμεύσει κάποιο ποσό για staking, τότε η λοταρία επιλέγει τυχαία κάποιο κόμβο από τους υπάρχοντες στο ring. Το πρώτο block μετά το genesis block ανήκει σε αυτή την κατηγορία, αφού κανένας κόμβος δεν έχει μπορέσει να δεσμεύσει stake ακόμα. Τέλος, είναι σημαντικό να αναφέρουμε ότι στην υλοποίηση μας η λοταρία για το ποιος κόμβος θα είναι validator λαμβάνει υπ' όψη τα ποσά των staking του τελευταίου validated block. Το staking ισχύει μέχρι ο κόμβος να θέσει κάτι διαφορετικό ως staking είτε να αποφασίσει να μηδενίσει το staking του.

Πειράματα

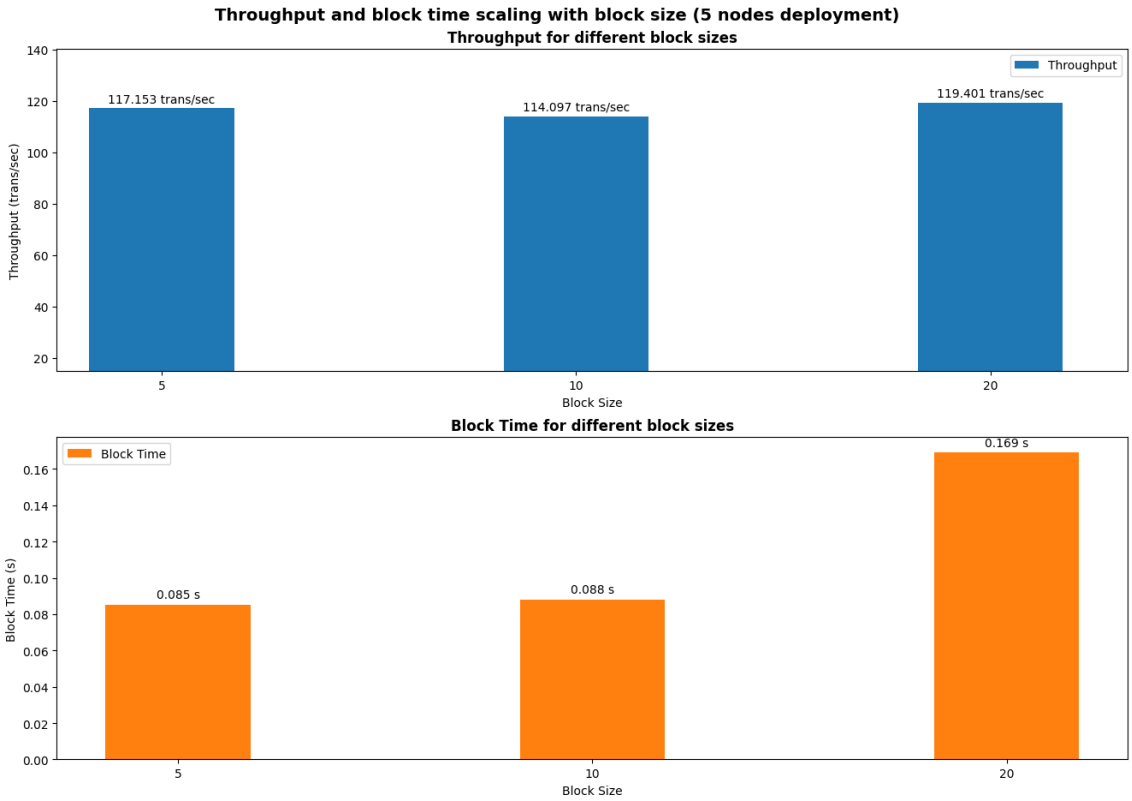
Στα πειράματα θα δούμε πως επηρεάζεται το throughput του συστήματος (transactions ανά δευτερόλεπτο) αλλά και το block time (χρόνος για να δημιουργηθεί block) καθώς αλλάζουμε το μέγεθος του block, δηλαδή το πόσα transaction χωράει. Στην συνέχεια θα δούμε πως η μετρική αυτό συμπεριφέρεται αν μαζί με το μέγεθος του block, διπλασιάσουμε το πλήθος των κόμβων (ώστε να δούμε το scaling του συστήματος). Τα πειράματα αυτά θα εκτελεστούν με σταθερό staking 10 BCCs σε κάθε κόμβο. Το τελευταίο πείραμα θα δοκιμάσει την δικαιοσύνη του συστήματος, αφού ένας κόμβος θα έχει 100 BCCs stake, ενώ οι υπόλοιποι θα παραμείνουν στα 10 BCCs.

Στο πείραμα με τους 5 nodes, οι 5 nodes είναι deployed ένας σε κάθε VM ενώ στο πείραμα με τους 10 nodes το κάθε VM φιλοξενεί 2 nodes σε διαφορετικά ports.

Στις μετρικές έχουμε:

- **throughput** που μετράει πόσα transactions μπορεί να δεχτεί το σύστημα στην μονάδα του χρόνου (δηλαδή να έχουμε επιτυχής ή αποτυχημένη κλήση στο endpoint /api/create_transaction), χωρίς απαραίτητα αυτά να έχουν μπει σε validated block.
- **block time** ή πιο σωστά **digestion block time** που μετράει πόσο χρόνο χρειάστηκε για να δημιουργηθούν όσα block αντιστοιχούν στα transactions που στείλαμε. Αυτό συμβαίνει διότι λόγω υλοποίησης, η δημιουργία block συμβαίνει και μόλις συμπληρωθεί επαρκής αριθμός transactions, αλλά και ανά τακτά χρονικά διαστήματα σε επανάληψη ώστε να αδειάζει η λίστα με τις εκκρεμείς συναλλαγές.

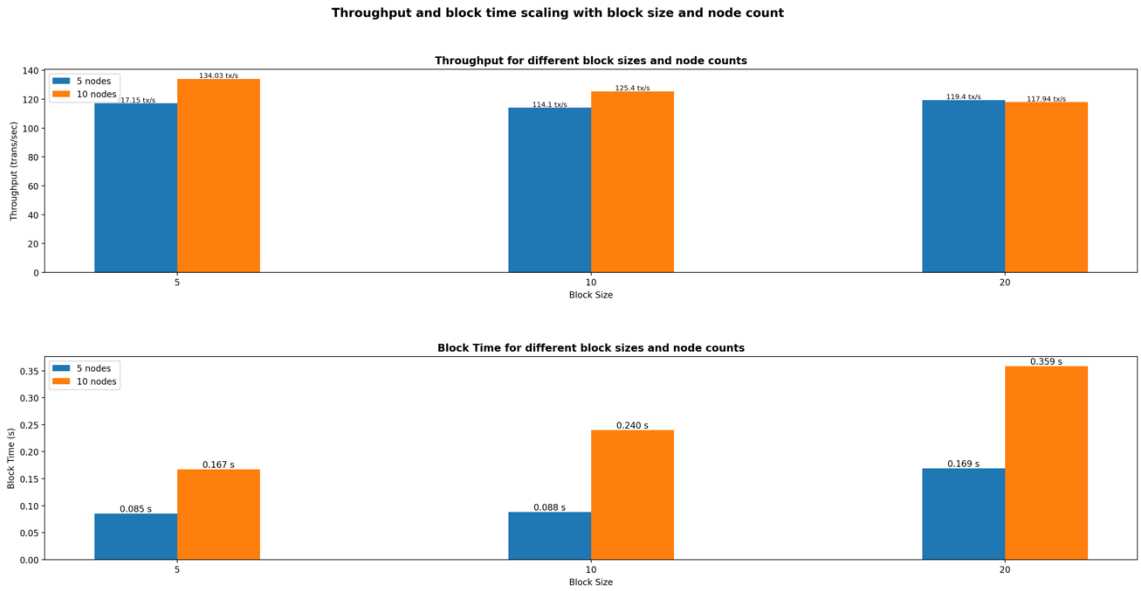
Στο ακόλουθο διάγραμμα παρουσιάζονται στα δύο subplots το throughput και το block time του συστήματος για τα ζητούμενα block sizes (5, 10, 20).



Όπως παρατηρούμε το throughput παραμένει σταθερό. Αυτό συμβαίνει γιατί το μέγεθος του block δεν επηρεάζει κάπου την δημιουργία του transaction, αυτά μπαίνουν σε μία λίστα εκκρεμών transactions και στην συνέχεια μπαίνουν σε block. Το πόσα transactions χωράει το block δεν επηρεάζει την ταχύτητα που μπαίνουν αυτά στην λίστα εκκρεμών transactions.

Στη συνέχεια θα παρουσιάσουμε τα metrics του συστήματος συγκριτικά για deployment 5 και 10 κόμβων. Στις πορτοκαλί bars φαίνονται το throughput και το block time του συστήματος για τα διάφορα block sizes για 10 nodes. Ενώ οι μπλε μπάρες παρουσιάζουν τα προηγούμενα αποτελέσματα για deployment 5 κόμβων για καλύτερη σύγκριση του scaling του συστήματος. Όπως παρατηρούμε από τα γραφήματα το throughput του συστήματος στο πείραμα 10 κόμβων είναι μεγαλύτερος από το αντίστοιχο των 5 κόμβων.

GitHub Repository: https://github.com/EEMplekei/BlockChat



Όπως αναφέραμε και προηγουμένως, το throughput δεν επηρεάζεται από το block size. Βλέπουμε κατά γενικές γραμμές λίγο πιο γρήγορες ταχύτητες για 10 nodes και αυτό οφείλεται στο γεγονός ότι κατά την διάρκεια του benchmarking έχουμε τόσα threads όσα και nodes να στέλνουν transactions στο αντίστοιχο node. Η διαδικασία του transactions broadcast στο backend γίνεται με χρήση ενός ασύγχρονου thread και άρα δεν παρατηρούμε κάποια καθυστέρηση από αυτό (αν αυτό δεν γινόταν τότε θα βλέπαμε γιατί στην μια περίπτωση θα έπρεπε να περιμένουμε σειριακό broadcast σε 10 κόμβους έναντι 5).
Η διαφορά στο block time, που το βλέπουμε να διπλασιάζεται για 10 nodes είναι λόγω του "δικτυακού lock" που έχουμε υλοποιήσει. Όταν ένας κόμβος δεν είναι validator περιμένει εντός της mint_block ώστε ο validator να του πει ότι έστειλε το block παντού (το έκανε broadcast παντού). Προφανώς αυτή η διαδικασία παίρνει διπλάσιο χρόνο αν έχουμε διπλάσιους κόμβους, αφού πρέπει να περιμένουμε όλοι να το βάλουν στο blockchain τους.

3) Επαναλαμβάνουμε το πείραμα (*χρησιμοποιώντας το αρχείο unfair_test.py*) αλλά αυτή τη φορά ένας τυχαίος node του συστήματος θέτει το stake του ίσο με 100, ενώ όλοι οι υπόλοιποι κόμβοι του συστήματος έχουν stake ίσο με 10. Με βάση το Proof-of-Stake πρωτόκολλο αναμένουμε ο κόμβος αυτός να εκλέγεται ως validator περίπου για το 50% των block.
Η αρχική μας υπόθεση επιβεβαιώνεται από το πείραμα, αφού βλέποντας το τελικό blockchain παρατηρούμε ότι ο node, ο οποίος θέτει και το μεγαλύτερο stake είναι κι αυτός που τελικά κληρώνεται τις περισσότερες φορές (λίγο παραπάνω από τις μισές) και συνεπώς κερδίζει τα περισσότερα fees από τις δοσοληψίες.

Συγκεκριμένα σε μία εκτέλεση που για κόμβος με 100 stake επιλέχθηκε ο 2, βλέπουμε ότι σε 51 από τα 77 blocks. Στο τέλος ταυτόχρονα, είχε το μεγαλύτερο balance με 4320, έναντι των άλλων που είχαν:

- node0: 156, node1: 4, node3: 150, node4: 207

GitHub Repository: https://github.com/EEMplekei/BlockChat