Κατανεμημένα Συστήματα
Εξαμηνιαία Εργασία

Ομάδα 6
Βασίλειος Βρεττός - el18126,
Νικόλαος Παγώνας - el18175
Αλέξανδρος Τσάφος - el18211

Σκοπός της εξαμηνιαίας εργασίας ήταν η υλοποίηση ενός συστήματος blockchain στο οποίο θα καταγράφονται δοσοληψίες μεταξύ χρηστών. Την ασφάλεια και εγκυρότητα αυτών των δοσοληψιών την παρέχει το σύστημα validation των δοσοληψιών καθώς και η χρήση αλογρίθμου consensus για την διόρθωση σφαλμάτων.

1 Ζητήματα Υλοποίησης/Εκτέλεσης

1.1 Υλοποίηση

Επιλέξαμε σαν γλώσσα προγραμματισμού την Go, η οποία έχει αποδειχθεί ως καταπληκτική επιλογή στην υλοποίηση προγραμμάτων που εκτελούνται με κατανεμημένο τρόπο. Η επίδοσή της είναι συγκρίσιμη με άλλες low-level γλώσσες προγραμματισμού (όπως η C) και η ευκολία προγραμματισμού είναι αρκετά κοντά σε πιο high-level γλώσσες όπως η Python ή η Javascript. Όλες οι σημαντικότερες βιβλιοθήκες που χρειαστήκαμε (SHA256, RSA key-pairs, JSON encoding, e.t.c.) ήταν ήδη υλοποιημένες.

Για την επικοινωνία των client μεταξύ τους, αξιοποιήσαμε sockets. Τα μηνύματα είναι κωδικοποιημένα με την μορφή JSON.

1.2 Εκτέλεση πειραμάτων/μετρήσεων

Για την εργασία, χρησιμοποιήθηκαν πόροι από την υπηρεσία Okeanos-Knossos. Συνολικά έχουμε 5 ξεχωριστά μηχανήματα, το καθένα με 2 πυρήνες. Ιδανικά, μπορούμε να τρέχουμε εώς 10 nodes χωρίς να υπάρχει "σοβαρό" ζήτημα χρονοδρομολόγησης των διεργασιών/nodes από το λειτουργικό του συστήματος.

Τα πειράματα έγιναν είτε για 5 ή για 10 nodes, για να ελέγξουμε κατά πόσο είναι κλιμακώσιμο το σύστημα που υλοποιήσαμε.

2 Περιγραφή του συστήματος

Κάθε Node χρησιμοποιεί τις εξής δομές:

- **NodeDataList:** Οι πληροφορίες για όλους τους υπόλοιπους κόβμους του δικτύου.
- **GlobalUTXO:** Η στοίβα με Unspent Transactions κάθε κόμβου. Έχουμε κάνει την σύμβαση ότι όλοι γνωρίζουν τα Unspent Transaction Outputs των άλλων.
- **Locks:** Χρησιμοποιούμε 3 locks για τις σημαντικότερες λειτουργίες του συστήματος που έχουν πιθανά race conditions (Broadcasting, Mining και Blockchain Addition)

Ορίζουμε κάποιες έννοιες που θα συναντήσουμε παρακάτω.

- **Signed Transaction:** Δοσοληψία που έχει δημιουργηθεί από node και υπογραφθεί από αυτό. Πρέπει να γίνει validated από τα υπόλοιπα nodes για να μπει σε node.
- **Capacity:** Το πλήθος των Signed Transaction που πρέπει να υπάρχουν σε ένα block ώστε να ξεκινήσει να γίνει mined. Δεν δεχόμαστε άλλο πλήθος.
- **Difficulty:** Ο αριθμός των μηδενικών που πρέπει να ξεκινάει το SHA256 Hash του block που κάνουμε mine ώστε να θεωρηθεί σωστό το nonce που επιλέξαμε.

2.1 Διαχείριση Unspent Transaction Output

Στο σύστημα, δεν υπάρχει "balance" πορτοφολιού. Το ποσό των NBC που έχει ο κάθε χρήστης το βρίσκουμε αθροίζοντας όλα τα Unspent Transaction Outputs του. Ένα Unspent Transaction Output δημιουργείται στην γέννηση ενός ορθού transaction μεταξύ 2 χρηστών. Ο χρήστης που στέλνει χρήματα τοποθετεί στην στοίβα του τα ρέστα αυτής της συναλλαγής ενώ ο χρήστης που λαμβάνει χρήματα τοποθετεί στην στοίβα του το ποσό που έλαβε. Τα transaction outputs αυτά περιέχουν επίσης το ID της δοσοληψίας από την οποία προέκυψαν. Η πληροφορία αυτή είναι κρίσιμη για την επιβεβαίωση των δοσοληψιών καθώς και για την αποφυγή του double spending.

Την κινητικότητα των στοιβών αυτών, την καταγράφουν όλοι οι χρήστες του δικτύου. Αυτό γίνεται μέσω των dictionaries που υπάρχουν στην δομή του node. Μέσω αυτών, καταγράφουμε τα validated, commited και used UTXOs κάθε κόμβου. Valid UTXO είναι όταν έχει περάσει από την φάση του soft-validation, δηλαδή απλά ελέγχουμε την ορθότητα του transaction αλλά δεν την τοποθετούμε σε κάποιο block προς mining. Commited UTXO είναι το UTXO που προήλθε από valid transaction και πήγε προς mine. Used UTXO είναι τα transaction outputs τα οποία έχουν χρησιμοποιηθεί και δεν πρέπει να ξαναυπολογιστούν για την γέννηση οποιουδήποτε transaction. Με αυτό το σύστημα όλοι γνωρίζουν τους διαθέσιμους πόρους που έχει κάποιος άλλος χρήστης στο δίκτυο.

Σημείωση: Αυτό ισχύει MONO επειδή εκτελούνται όλες οι συναλλαγές αφού έχουν συνδεθεί όλοι οι χρήστες και αφού δεν φεύγει κανείς από το δίκτυο.

2.2 Block Mining

Το mining κάθε block γίνεται όταν γεμίσει η λίστα με τα transactions, όπως έχουμε περιγράφει παραπάνω. Υπάρχει μια διαφορά με το πως επιβάλλουμε τον κανόνα του difficulty στο mining. Αντί να ελέγχουμε τα πρώτα DIFFICULTY bits, ελέγχουμε τα πρώτα DIFFICULTY δεκαεξαδικά ψηφία του block hash, άρα τα πρώτα nibbles.

Παρατηρήσαμε ότι με τους αριθμούς difficulty που μας δοθήκαν ώστε να ελέγξουμε την επίδοση του συστήματος, προέκυπταν κάποια προβλήματα στην επικοινωνία. Μία μεγαλύτερη δυσκολία στο mining προσέφερε μεγαλύτερη σταθερότητα και καλύτερα αποτελέσματα, αλλά λίγο πιο αργό block mining time.

2.3 Επαλήθευση Transaction/Block/Blockchain

Αυτά τα 3 αντικείμενα είναι αυτά τα οποία γίνονται validated από κάθε χρήστη και, εν τέλει, προσφέρει ασφάλεια χωρίς την ύπαρξη κάποιου κεντρικού ελεγχτή.

2.3.1 Transaction

Ένα transaction δημιουργείται από έναν χρήστη προς έναν άλλο και συμπεριλαμβάνονται: οι χρήστες που συμμετέχουν, το ποσό που μεταφέρεται, τα UTXOs που χρησιμοποιήθηκαν ως είσοδοι και τα UTXOs που παράγονται ως έξοδοι. Αυτό το transaction γίνεται signed με το private key του αποστολέα και δίνεται στους άλλους χρήστες για validation.

Το validation μιας δοσοληψίας απαιτεί έλεγχο της υπογραφής με χρήση το public key του αποστολέα και την ορθότητα των στοιχείων. Τα UTXOs που χρησιμοποιήθηκαν ως είσοδοι πρέπει να υπάρχουν στην στοίβα του αποστολέα ώστε να επιβεβαιωθεί ότι όντως έχει το απαραίτητο ποσό για την συναλλαγή. Επίσης επιβεβαιώνονται ότι ο αποστολέας και ο παραλήπτης είναι ορθοί και υπάρχουν στο δίκτυο. Αν αποτύχει οποιοσδήποτε από αυτούς τους ελέγχους, η επαλήθευση αποτυγχάνει και ΔΕΝ μπαίνει αυτή η συναλλαγή στο block. Αν πετύχει, μπαίνει στο block και αφαιρούνται τα UTXOs από τον αποστολέα και προστίθεται το ποσό της συναλλαγής στον παραλήπτη.

2.3.2 Block

Ένα block, για να θεωρηθεί έγκυρο, πρέπει να ισχύουν τα εξής:

- Οι δοσοληψίες που περιέχει είναι έγκυρες.
- Η τιμή του hash του είναι σωστή. Για να γίνει αυτός ο έλεγχος, ξαναγίνονται hash τα δεδομένα του και συγκρίνονται με την τιμή αυτή.
- Το index του block δεν πρέπει να είναι μικρότερο από το μήκος της τωρινής αλυσίδας κάθε κόμβου. Αυτό σημαίνει ότι έχουμε λάβει παλιότερο block.
- Το index του block επίσης δεν πρέπει να είναι μεγαλύτερο από το μήκος της τωρινής αλυσίδας. Αυτό σημαίνει ότι έχουμε κάποιο κενό στην αλυσίδα.
- Το πεδίο previousHash είναι όντως ίδιο με το hash του τελευταίου block του τωρινού blockchain κάθε κόμβου.

Στις 2 τελευταίες περιπτώσεις, καλείται η resolve conflict, ένας αλγόριθμος consensus ώστε να λάβουν όλοι οι χρήστες μια κοινή, valid αλυσίδα. Θα εξηγήσουμε τον αλγόριθμο consensus που υλοποιήσαμε παρακάτω.

2.3.3 Blockchain

Για να θεωρείται ένα blockchain έγκυρο, πρέπει να θεωρούνται έγκυρα όλα τα επιμέρους block από τα οποία αποτελείται

2.4 Αλγόριθμος Consensus

Στο ενδεχόμενο που έχουμε είτε κάποια διακλάδωση του blockchain μεταξύ κόμβων ή κάποιο κενό στην αλυσίδα, οι κόμβοι πρέπει να εκτελέσουν την συνάρτηση resolve conflict η οποία εκτελεί αλγόριθμο consensus. Το αποτέλεσμα αυτής της διαδικασίας είναι να λάβουν όλοι μια valid αλυσίδα ώστε να συνεχιστεί η συνέπεια του συστήματος.

Αρχικά, στέλνουν όλοι οι χρήστες μήνυμα το οποίο περιέχει το blockchain τους. Συγκρίνουν τα μεγέθη μέχρι να βρεθεί η μεγαλύτερη σε μήκος αλυσίδα. Αν υπάρχουν πολλές επιλογές, επιλέγεται η αλυσίδα του γηραιότερου κόμβου (καθώς διατρέχουμε από την αρχή την λίστα με τις πληροφορίες των κόμβων), δηλαδή αυτού με το μικρότερο Node ID. Πριν γίνει οποιαδήποτε σύγκριση, ελέγχεται πρώτα το validity των αλυσιδών και προφανώς απορρίπτεται αν βρεθεί μη-έγκυρη.

3 Μετρήσεις - Συμπεράσματα Πειραμάτων

Για την εκτέλεση των πειραμάτων, τρέχουμε όλους τους πιθανούς συνδυασμούς για capacity = {1, 5, 10}, difficulty = {4,5}. Συνολικά λαμβάνουμε αποτελέσματα για 6 διακριτούς συνδυασμούς. Η εκτέλεση γίνεται για 5 και 10 nodes, ξεχωριστά. Ο κάθε χρήστης εκτελεί 100 συναλλαγές ο καθένας, άρα θα έχουμε συνολικά 500 συναλλαγές στο σύστημα για 5 nodes και 1000 για 10 nodes. Μερικές από αυτές τις συναλλαγές δεν "περνάνε" στο blockchain, αφού δεν είναι δυνατή η συναλλαγή λόγω ανεπαρκούς balance στο πορτοφόλι του αποστολέα.

Οι 2 κυριότερες μετρικές που ελέγχουμε είναι το throughput ($\frac{Transactions}{Second}$) του συστήματος και ο μέσος χρόνος εκτέλεσης mining ενός block.

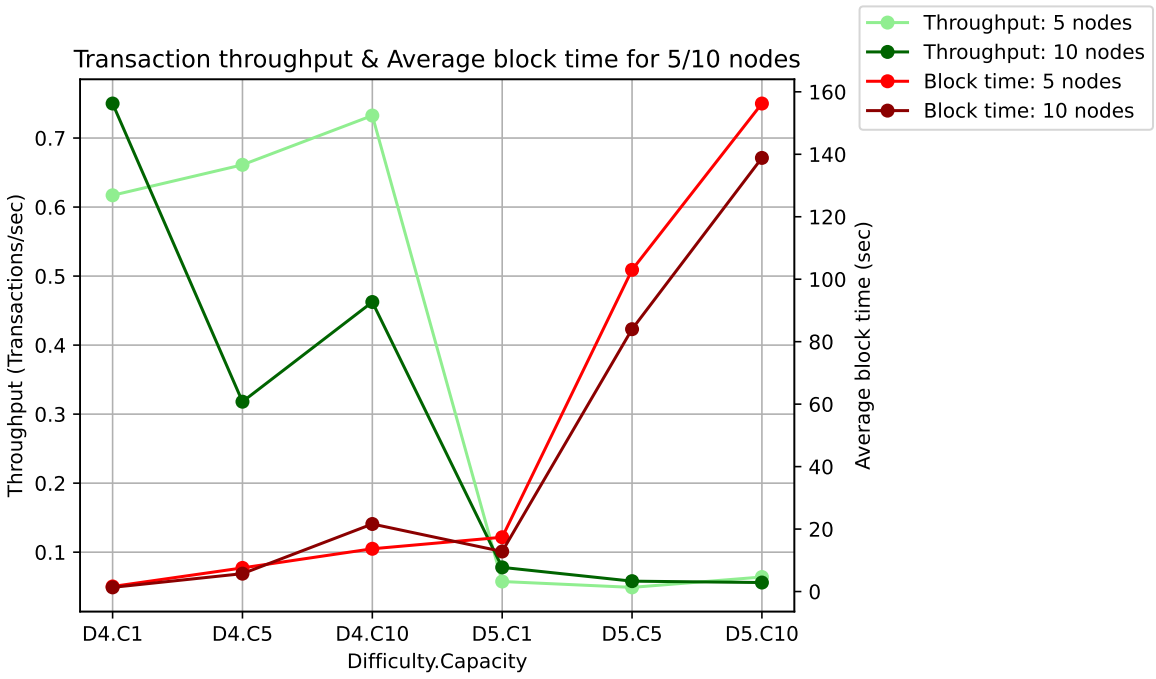

Figure 1: Noobcash Benchmarking - Transaction Throughput - Average Block Time

3.1 Throughput - Median Block Time

Τα 2 χαρακτηριστικά, capacity και difficulty, επηρεάζουν αντίστοιχα το throughput και το μέσο block time του συστήματος. Αυξάνοντας το capacity μειώνουμε το throughput καθώς τα nodes κάνουν mine blocks πιο συχνά αλλά επίσης δημιουργούμε ένα πιο ασφαλές και έγκυρο δίκτυο αφού δημιουργούμε ένα έγκυρο snapshot των transaction πιο συχνά.

Αυξάνοντας το difficulty έχουμε μεγαλύτερο μέσο χρόνο block time. Σε ένα πραγματικό σύστημα, που το Proof-of-Work θα είχε και reward system, θα δημιουργούσε μια πιο σταθερή οικονομία και θα έδινε λόγο σε παραπάνω χρήστες να συμμετάσχουν στο δίκτυο.

3.2 Scalability

Πλέον, διπλασιάζοντας τους χρήστες, η πιθανότητα να βρεθεί ένα block στην διαδικασία του mining είναι αρκετά μεγαλύτερη. Πάλι όμως, η εύρεση του σωστού nonce είναι θέμα τύχης και υπάρχουν κάποιες ακραίες περιπτώσεις (π.χ. 20 λεπτά για να βρεθεί ένα σωστό nonce) που χαλάνε τον Μ.Ο. των μετρήσεων. Αν και το μέσο block time είναι λίγο καλύτερο από τα πειράματα με τους 5 χρήστες, το throughput είναι μικρότερο αφού χρειάζεται περισσότερος χρόνος για να εκτελεστούν και οι 1000 συναλλαγές. Δεν μπορούμε να έχουμε πλήρως γραμμικό speedup αφού η διαδικασία που μπορεί να παραλληλοποιηθεί (block mining) εξαρτάται από τύχη.

3.3 Γενικό συμπεράσμα για την συγκεκριμένη υλοποίηση

Όπως αναφέραμε και παραπάνω, παρατηρούμε μεγαλύτερη σταθερότητα σε μεγαλύτερο difficulty αφού λειτουργεί καλύτερα η επικοινωνία μεταξύ των κόμβων. Δυστυχώς δεν μπορούμε

να έχουμε πλήρη έλεγχο στο πότε να διακόπτουμε την εκτέλεση κώδικα από ένα νήμα αφού η Go δεν είναι πλήρως preemptive. Δεν καταφέραμε να έχουμε τέτοιου είδους επικοινωνία ώστε να λειτουργεί με πλήρη τελειότητα το σύστημα. Αν γραφόταν σε μία ακόμα πιο low-level γλώσσα προγραμματισμού, με καλύτερο έλεγχο του Λειτουργικού από πίσω θα μπορούσε να τρέξει βέλτιστα ακόμα και για μικρότερα difficulties.