ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εξαμηνιαία Εργασία

Στην συνέχεια ακολουθεί η περιγραφή του συστήματος NoobCash (blockchain) καθώς και η παρουσίαση και η ανάλυση των αποτελεσμάτων των πειραμάτων που έγιναν.

Περιγραφή Συστήματος NoobCash

•**Δημιουργία δικτύου και αρχικά transactions** Το σύστημα υλοποιήθηκε σε Nodejs και η επικοινωνία των κόμβων γίνεται με REST API. Το σύστημα ξεκινάει με τον bootstrap κόμβο (αρχείο bootstrap_node.js) ο οποίος δημιουργεί το genesis block που περιέχει ένα transaction , τα n*100 NBC που δίνονται στον bootstrap. Έπειτα ο bootstrap περιμένει να εισαχθούν όλοι οι κόμβοι στο δίκτυο. Ο κάθε κόμβος (αρχείο networkNode.js) μόλις ξεκινήσει ειδοποιεί τον bootstrap (θεωρούμε ότι τα στοιχεία του bootstrap(ip, port) είναι γνωστά) στέλνοντάς του τα στοιχεία του (ip, port, publicKey). Ο bootstrap αποθηκεύει τα στοιχεία του κάθε κόμβου και στέλνει το uid που είναι ένας αύξων αριθμός μοναδικός για κάθε κόμβο του συστήματος. Μόλις εισέλθει ο τελευταίος κόμβος ο bootstrap γνωστοποιεί τα στοιχεία όλων τον κόμβων σε όλο το δίκτυο και έπειτα εκτελεί n-1 transactions μοιράζοντας 100NBC σε κάθε κόμβο. Από αυτό το σημείο και έπειτα ο bootstrap κόμβος συμπεριφέρεται σαν ένας απλός κόμβος του συστήματος, χωρίς να εκτελεί κάποια επιπλέον λειτουργία. Μόλις ολοκληρωθούν αυτά τα αρχικά transactions ο κάθε κόμβος διαβάζει το αντίστοιχο input file του εκτελεί τα transactions που περιέχονται σε κάθε input file.

•**Transactions** Ο κόμβος που θέλει να εκτελέσει ένα transaction χτυπάει το /transaction/broadcast endpoint του δίνοντας σαν input το uid του κόμβου που θέλει να στείλει NBCs και το amount που θέλει να στείλει. Στη συνέχεια υπογράφει το transaction , δηλώνει τα transaction_inputs και

κάνει broadcast το transaction στο δίκτυο στο /validate transaction endpoint όλων των κόμβων. Ο κάθε κόμβος εκτελεί την validate transaction() και στέλνει θετική ή αρνητική απάντηση. Αν έστω και ένας κόμβος στείλει αρνητική απάντηση το transcaction γίνεται discarded αλλιώς ο κόμβος που έστειλε το transaction, στέλνει σε ολους του κόμβους ότι το transaction έγινε δεκτό από όλους (/add_to_pending endpoint) και το προσθέτουν όλοι στα pending_transactions τους (λίστα από transaction που δεν έχουν προστεθεί σε blocks ακόμα). Επίσης σε αυτή την φάση εφόσον το transaction είναι valid ενημερώνουν όλοι τα wallet τους. Ο κάθε κόμβος μαζι με τα ip, port, publicKey όλων των υπολοίπων κρατάει και τα wallets , δλδ τα UTXOs του κάθε κόμβου, και τα ενημερώνει όποτε ένα transaction γίνεται validated από όλους. Έτσι, τα records των wallets όλων των κόμβων είναι ίδια.

•**Mining** Όταν έχουν συμπληρωθεί capacity (δίνεται σαν όρισμα κατά την εκτέλεση των bootstrap_node.js και networkNode.js) transactions (γίνεται έλεγχος κάθε φορά που προστίθεται ένα transaction στα pending_transactions) ο κόμβος εκτελεί την mine_block(). Αφού βρει το nonce κάνει broadcast το block στο δίκτυο στο /receive_new_block endpoint. Εκεί το block γίνεται validated (validate_block()), προστίθεται στο blockchain, και αφαιρούνται τα transactions που μπήκαν στο block απο τα pending. Εάν έρθει block που πρέπει να προστεθεί σε σημείο της λίστας που υπάρχει ήδη κάποιο block , τότε ο κόμβος το απορρίπτει (κρατάει μόνο αυτό που έφτασε πρώτο). Τέλος, εάν έρθει block το οποίο πρέπει να προστεθεί σε “αδειο” σημείο της λίστας (δλδ αμέσως μετά το τελευταίο block) αλλά το previous block hash δεν συμφωνεί με το current hash του τελευταίου block της λίστας, ο κόμβος εκτελεί την resolve conflict() (consensus) .

•**Consensus** Ο κόμβος ζητά αντίγραφα του blockchain από κάθε κόμβο του δικτύου (/blockchain endpoint) και βρίσκει αυτό με το μεγαλύτερο μήκος. Κρατάει το blockchain που βρήκε και ενημερώνει τα pending

transactions του ανάλογα με το blockchain που κράτησε. Έτσι, τα wallets , blockchains και pending transactions όλων των κόμβων του δικτύου βρίσκονται σε συμφωνία μεταξύ τους.

•**Client** Στο αρχείο index.html παρέχεται ένα πολύ απλο front-end. Ουσιαστικά περιέχει links σε κάποια endpoints που δίνουν πληροφορίες για τα transactions, wallets και blockchains των κόμβων του δικτύου. Συγκεκριμένα: **/view** Προβολή των transactions του τελευταίου block **/new_transaction** Δημιουργία και εκτέλεση νέου transaction **/wallet_balance/uid** Προβολή του υπολοίπου του κόμβου uid(άθροισμα των UTXOs του) **/help** Πληροφορίες για τα παραπάνω endpoints

•**Επιπλέον endpoints-λειτουργίες** Παρέχονται και κάποια επιπλέον endpoints για :

•προβολή ολόκληρου του blockchain κάθε κόμβου (/blockchain) ,

•προβολή του record των wallets κάθε κόμβου (/wallets),

•προβολή των στοιχείων(url, port, publicKey, uid) κάθε κόμβου (/nodes_stats)

Παρουσίαση και Ανάλυση πειραμάτων

Εκτελέστηκαν πειράματα με
capacity : 1, 5, 10 ,
difficulty: 4, 5 , nodes: 5, 10

και πήραμε τα εξής αποτελέσματα : **Για 5 nodes** Difficulty : 4

Capacity : 1 -> Throughput = 0.07 tr/sec - Block Time = 10.418 sec

Difficulty : 4 Capacity : 5 -> Throughput = 0.27 tr/sec - Block Time = 18.762 sec

Difficulty : 4 Capacity : 1 -> Throughput = 0.37 tr/sec - Block Time = 15.608 sec

Difficulty : 5 Capacity : 1 -> Throughput = 0.006 tr/sec - Block Time = 72.815 sec

Difficulty : 5 Capacity : 5 -> Throughput = 0.025 tr/sec - Block Time = 92.404 sec

Difficulty : 5 Capacity : 10 -> Throughput = 0.038 tr/sec - Block Time = 88.705 sec

Για 10 nodes Difficulty : 4 Capacity : 1 -> Throughput = 0.089 tr/sec - Block Time = 11.698 sec

Difficulty : 4 Capacity : 5 -> Throughput = 0.31 tr/sec - Block Time = 12.944 sec

Difficulty : 4 Capacity : 1 -> Throughput = 0.43 tr/sec - Block Time = 16.088 sec

Difficulty : 5 Capacity : 1 -> Throughput = 0.0073 tr/sec - Block Time = 68.906 sec

Difficulty : 5 Capacity : 5 -> Throughput = 0.031 tr/sec - Block Time = 89.263 sec

Difficulty : 5 Capacity : 10 -> Throughput = 0.042 tr/sec - Block Time = 79.665 sec

Παρατηρούμε πως το block time είναι ανεξάρτητο από το capacity και το πλήθος των κόμβων καθώς παραμένει σχετικά σταθερό καθως αλλάζουμε τις παραπάνω παραμέτρους. Ωστόσο, το difficulty καθορίζει το block time καθώς το τελευταίο αυξάνεται σε μεγάλο βαθμό με την αύξηση του difficulty. Τέλος, το throughput εξαρτάται από το capacity (περισσότερα transactions ανά block => γρηγορότερη εξυπηρέτησή τους άρα και καλύτερο throughput) αλλά και από το πλήθος των κόμβων του συστήματος καθώς με περισσότερους κόμβους (miners) πολλά block γίνονται mine σε κάποιους κόμβους και πριν ξεκινήσει το mining σε άλλους. Έτσι αποφεύγεται το mining σε όλους τους κόμβους και αυξάνεται το throughput (αφού δεν χάνεται χρόνος για mining ενός block που έχει είναι ηδη mined και broadcasted στο δίκτυο).
Παρακάτω φαίνονται τα αποτελέσματα και σε διάγραμμα ανάλογα με το πλήθος των κόμβων στο δίκτυο

Throughput
Block Time