



Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Εξαμηνιαία Εργασία **Κατανεμημένα Συστήματα**

Νικόλαος Χάιδος 03118096

Νικόλαος Σπανός 03118822

Πέτρος Χαρίτος 03118863

Σχεδιασμός Συστήματος Noobcash

- *Transactions:*

Κάθε αντικείμενο *transaction* διαθέτει τα *public keys* του αποστολέα και του δέκτη, το ποσό της συναλλαγής, τα *UTXOs* που χρησιμοποιούνται για την εκτέλεση και αυτά που δημιουργούνται, την ψηφιακή υπογραφή (SHA256) και ένα *timestamp*. Παράλληλα, κάθε *transaction* διαθέτει μία Boolean μεταβλητή που υποδεικνύει εάν ανήκει στο *genesis block*, για να αποφεύγεται ο έλεγχος του *transaction*.

- *Blocks:*

Κάθε αντικείμενο *block* διαθέτει το *hash* του προηγούμενου *block*, την λίστα με τα *transactions* που περιέχει, το *nonce*, το *hash* του τρέχοντος *block* και ένα *timestamp*. Παράλληλα, κάθε *block* διαθέτει μία Boolean μεταβλητή που υποδεικνύει εάν είναι το *genesis block*, για να αποφεύγεται ο έλεγχος του και μία μεταβλητή που διατηρεί το μέγεθος του *block*, που χρησιμοποιείται στον υπολογισμό των μετρικών.

- *Blockchain:*

Το *blockchain* αποτελεί μια λίστα των επαληθευμένων *blocks*.

- *Consensus:*

Η διαδικασία *consensus* ξεκινάει όταν ένας κόμβος παραλάβει ένα *block* το οποίο δεν ταιριάζει στην λίστα του (*previous_hash* διαφορετικό από το *hash* του τελευταίου *block*). Η διαδικασία έχει ως σκοπό να επιτευχθεί συμφωνία μεταξύ όλων των κόμβων για την μορφή της αλυσίδας, αποφεύγοντας, έτσι, τις διακλαδώσεις. Ο αλγόριθμος που χρησιμοποιούμε είναι ένας αλγόριθμος δακτυλίου, στον οποίο ο κόμβος που ξεκινάει την διαδικασία στέλνει ένα *token* στον επόμενο κόμβο στο *ring*. Το *token* περιέχει το *id* του εκκινητή κόμβου, το *id* του κόμβου με την μέγιστη αλυσίδα έως εκείνο το σημείο και το μήκος αυτής της αλυσίδας. Κάθε κόμβος που λαμβάνει το *token* το ενημερώνει ανάλογα και το προωθεί. Σε περίπτωση που βρεθούν δύο κόμβοι με ίδιο μήκος αλυσίδας, κερδίζει ο κόμβος με το μικρότερο *id*. Παράλληλα, σε περίπτωση που ο εκκινητής εντοπίσει *token* με διαφορετικό *id* εκκινητή, κερδίζει αυτός με το μικρότερο *id* και το δεύτερο *token* δεν προωθείται. Μόλις το *token* επιστρέψει στον εκκινητή, ενημερώνεται ο νικητής του *consensus*, ο οποίος κάνει *broadcast* την αλυσίδα του σε όλους τους κόμβους. Τέλος, κάθε κόμβος επαληθεύει μόνο τα *blocks* τα οποία είναι διαφορετικά από την αλυσίδα που ήδη διέθετε.

- *Validations:*

Για την σωστή επαλήθευση, διατηρούμε *snapshots* των επαληθευμένων καταστάσεων του κόμβου, ώστε να μπορούν να επαληθευτούν καινούρια *blocks*, αλλά και για να μπορούμε

να επιστρέψουμε σε προηγούμενη έγγυρη κατάσταση, σε περίπτωση που αποτύχει το *validation*.

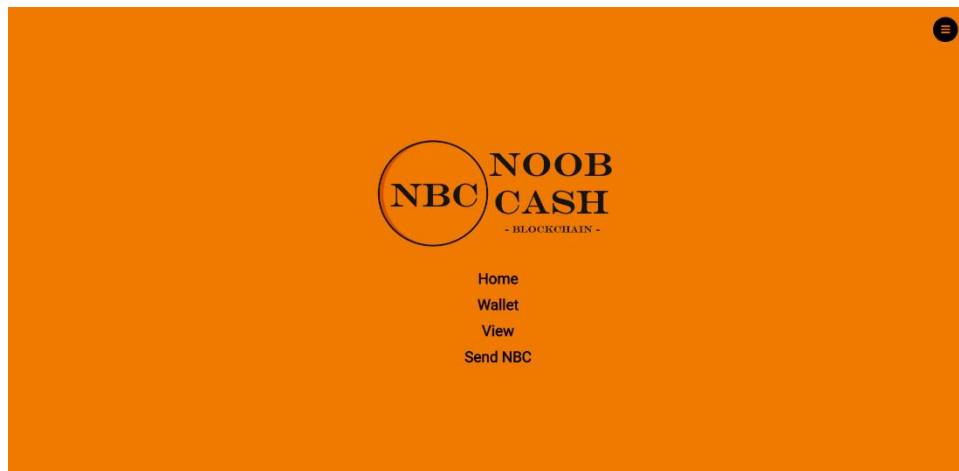
- *Επικοινωνία Κόμβων:*

Η επικοινωνία των κόμβων υλοποιήθηκε δημιουργώντας έναν τοπικό server με Python Flask, ο οποίος λαμβάνει όλα τα αιτήματα και τα τοποθετεί στους αντίστοιχους buffers. Ο server διαθέτει τέσσερις buffers: ληφθέντα *transactions*, ληφθέντα *blocks*, αιτήματα δημιουργίας *transaction* και ληφθέντα *consensus tokens*. Επίσης, όλα τα μηνύματα και δεδομένα αποστέλλονται με χρήση Flask Endpoints. Τέλος, οι buffers εξυπηρετούνται από ένα thread με την κατάλληλη προτεραιότητα.

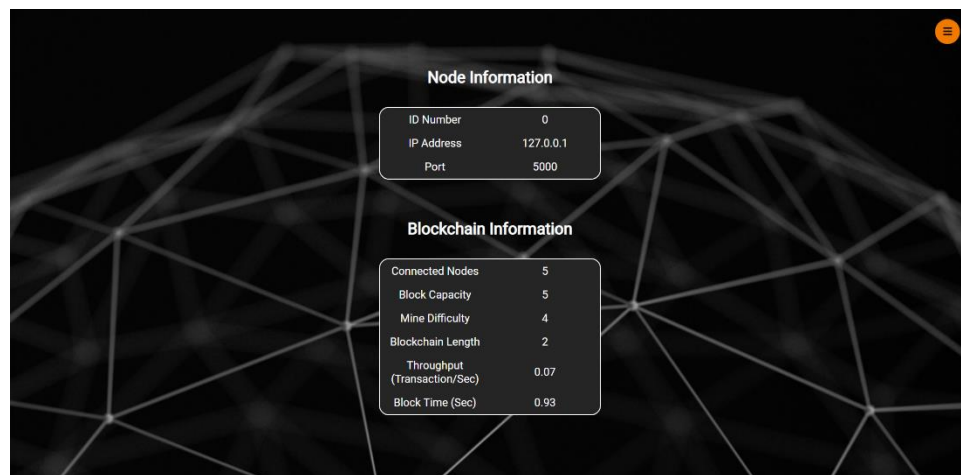
- *Frontend:*

Η υλοποίηση του *frontend* έγινε με χρήση HTML, CSS και Jinja. Εξυπηρετούνται 4 σελίδες:

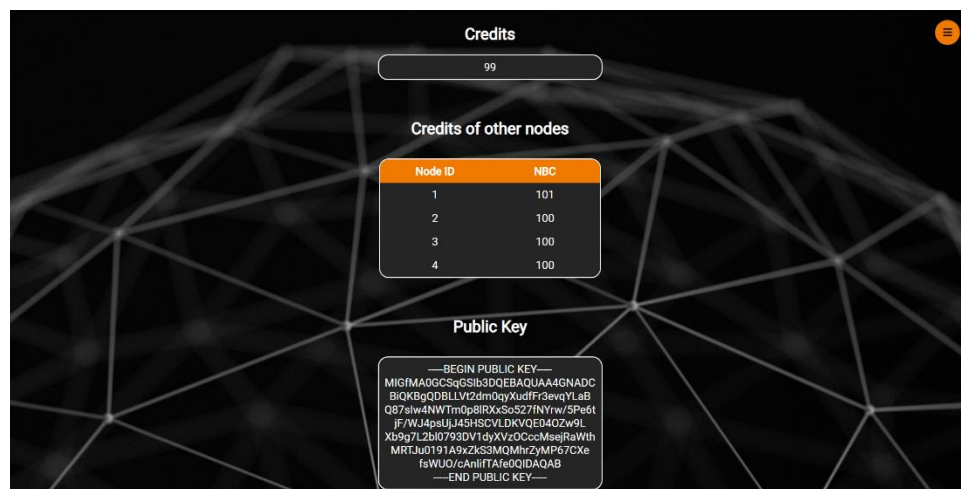
1. Menu Περιήγησης



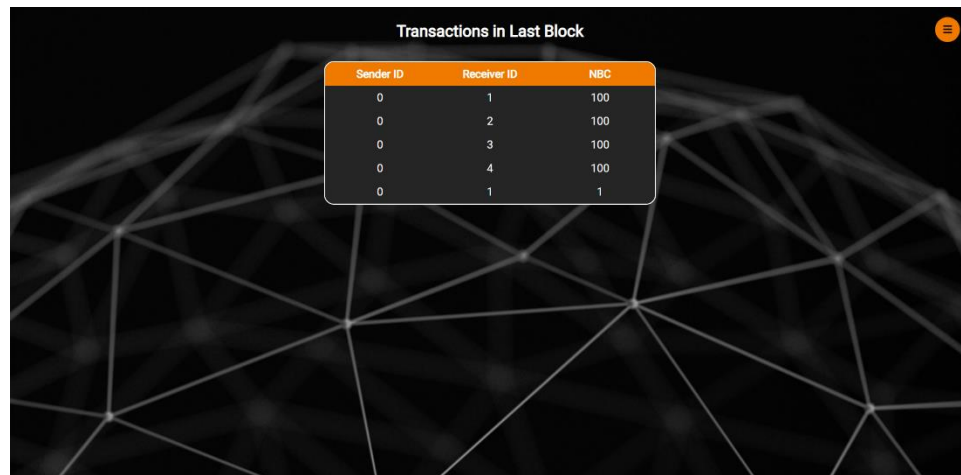
2. **Home:** Η αρχική σελίδα που παρουσιάζει γενικές πληροφορίες για τον κόμβο και το blockchain.



3. **Wallet:** Παρουσιάζει τα υπόλοιπα όλων των κόμβων και το public key του συγκεκριμένου χρήστη.

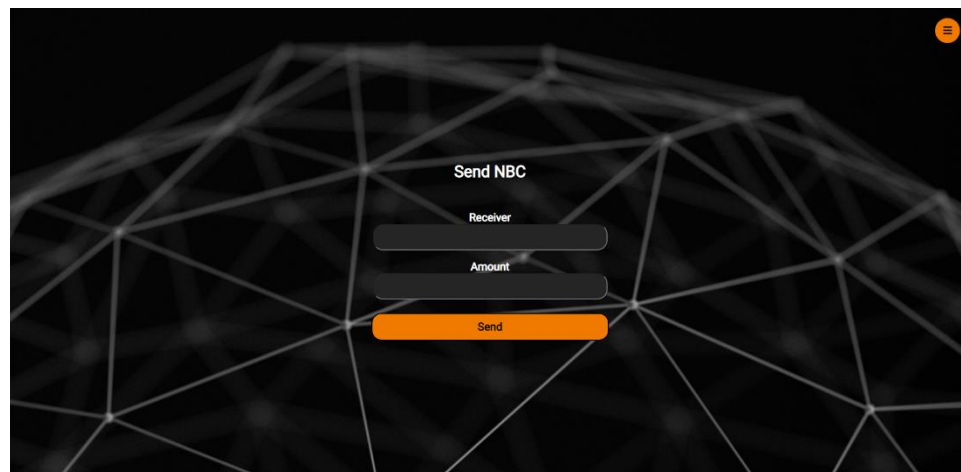


4. **View:** Παρουσιάζει τα *transactions* του τελευταίου επαληθευμένου block.



Sender ID	Receiver ID	NBC
0	1	100
0	2	100
0	3	100
0	4	100
0	1	1

5. **Send:** Υποστηρίζει την αποστολή *NBCs* μεταξύ κόμβων.



Send NBC

Receiver

Amount

Send

Μετρικές Πειραμάτων

5 Nodes:

- Throughput

	Difficulty 4	Difficulty 5
Capacity 1	0.49	0.1
Capacity 5	1.62	0.19
Capacity 10	1.03	0.17

- Mean Block Time

	Difficulty 4	Difficulty 5
Capacity 1	2.02	10.18
Capacity 5	3.02	26.98
Capacity 10	9.72	59.2

10 Nodes:

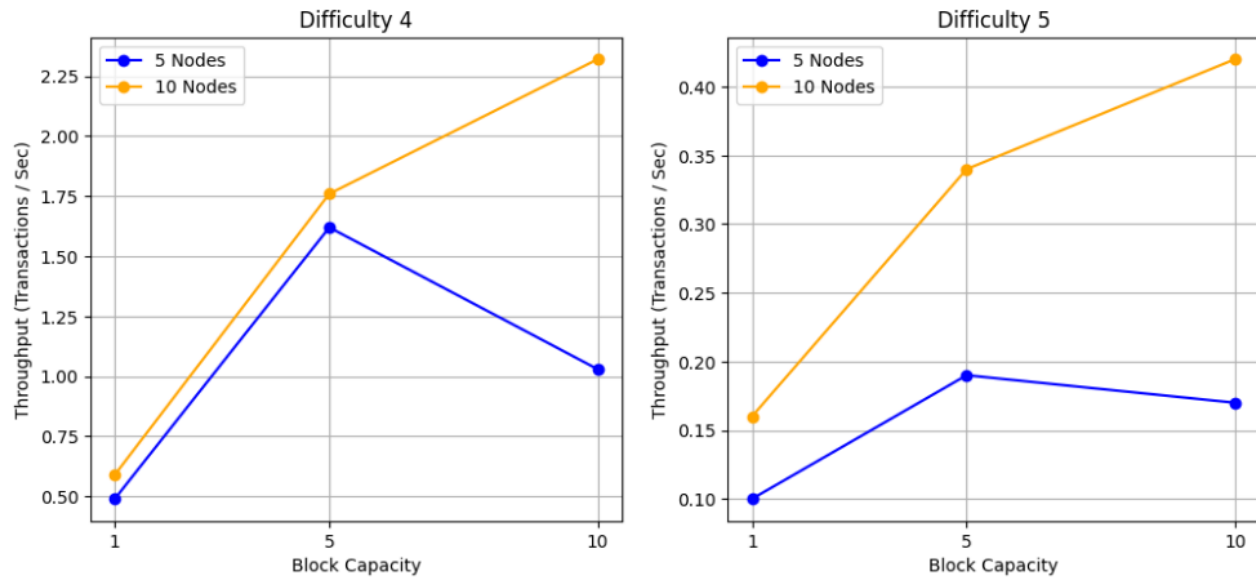
- Throughput

	Difficulty 4	Difficulty 5
Capacity 1	0.59	0.16
Capacity 5	1.76	0.34
Capacity 10	2.32	0.42

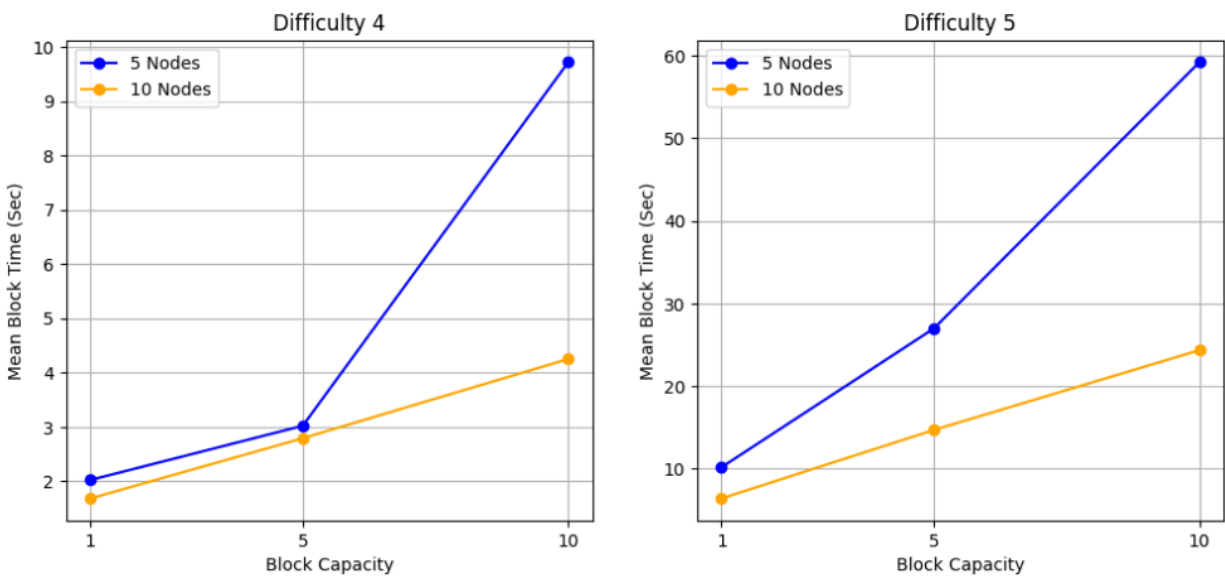
- Mean Block Time

	Difficulty 4	Difficulty 5
Capacity 1	1.68	6.41
Capacity 5	2.79	14.7
Capacity 10	4.25	24.35

Throughput



Mean Block Time



Παρατηρούμε από τα πειράματα ότι με την αύξηση της χωρητικότητας του *block*, το *Mean Block Time* αυξάνεται σε κάθε περίπτωση, καθώς παίρνει παραπάνω χρόνο να συγκεντρωθούν τα *transactions* που χρειάζονται για να συμπληρωθεί το *block*. Όσον αφορά το *Throughput*, τα αποτελέσματα διαφέρουν για 10 κόμβους και για 5 κόμβους, καθώς στους 10 το *Throughput* αυξάνεται σταδιακά με την χωρητικότητα, ενώ στους 5 κόμβους έχουμε βέλτιστο σημείο για χωρητικότητα ίση με 5. Σε όλα τα πειράματα, η αύξηση της δυσκολίας μειώνει την απόδοση

(αύξηση *Mean Block Time* – μείωση *Throughput*), καθώς απαιτείται περισσότερος χρόνος για το mining των *blocks*.