

# Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών  
και Μηχανικών Υπολογιστών

## Κατανεμημένα Συστήματα

9<sup>ο</sup> Εξάμηνο - Ροή Υ

### Αναφορά Εφαρμογής Noobcash

Ναυσικά Αμπατζή - 031 17 198  
*el17198@mail.ntua.gr*

Δημήτριος Δήμος - 031 17 165  
*el17165@mail.ntua.gr*

Νικόλαος Χριστόπουλος - 031 17 065  
*el17065@mail.ntua.gr*

Αθήνα  
Μάρτιος, 2022



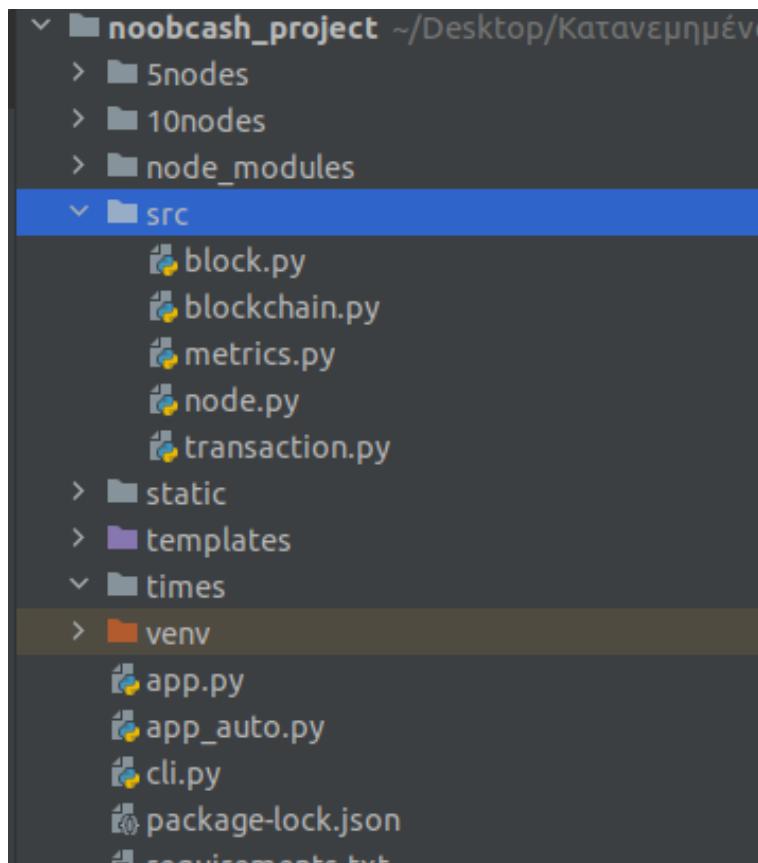
## Εισαγωγή

Ο σκοπός της εργασίας αυτής ήταν η ανάπτυξη ενός απλού συστήματος blockchain, όπου θα καταχράφονται οι διοσδι-ηψίες μεταξύ των συμμετεχόντων και θα εξασφαλίζεται το consensus με χρήση Proof-of-Work. Το blockchain είναι ουσιαστικά μία σειρά καταχωρίσεων που αφορούν συναλλαγές, σε ένα δημόσιο καθάστιχο (ledger). Κάθε καινούρια ομάδα καταχωρίσεων -ένα «block»- συνδέεται με τα προηγούμενα, δημιουργώντας μία «αλυσίδα» καταχωρίσεων, δηλαδή ένα «blockchain».

Τα blocks αυτά συνδέονται μονοσήμαντα μεταξύ τους. Προκύπτουν δε μέσα από μια διαδικασία που ονομάζουμε «proof of work», κατά την οποία επιτυγχάνεται η αλγορίθμική επίλυση ενός «δύσκολου» υπολογιστικού προβλήματος. Κατ' αυτό τον τρόπο, το blockchain λειτουργεί ως ένα αποκεντρωμένο (decentralized) λογιστικό καθολικό, το οποίο είναι κοινό για όλους τους συμμετέχοντες, μιας και όλοι οι εμπλεκόμενοι αποθηκεύουν ένα αντίγραφό του· κάτι που εξασφαλίζει την ασφάλεια και η διαιρέσιμη των συναλλαγών. Η ειδοποιός διαφορά -αναφορικά με την προστασία- προκύπτει από το γεγονός ότι δεν είναι πλέον απαραίτητη η ύπαρξη μιας ενδιάμεσης «έμπιστης» αρχής (πχ. μιας τράπεζας), ενώ η εμπιστοσύνη των συναλλασσομένων μερών βασίζεται σε αλγορίθμική επιβεβαίωση.

## Περιβάλλον Υλοποίησης

Το σύστημά μας αναπτύχθηκε με τη χρήση του framework Flask σε γλώσσα προγραμματισμού Python. Χρησιμοποιήθηκε επίσης HTML,CSS και jQuery για την ανάπτυξη του frontend. Ακολουθούν τα περιεχόμενα του project.



(a) Project Set up

## Εκκίνηση Εφαρμογής

Κατά την εκκίνηση της εφαρμογής αρχικά πρέπει να εισαχθεί ο bootstrap node και στη συνέχεια οι κόμβοι παιδία. Για την εισαγωγή του bootstrap node πρέπει να δώσει η εξής εντολή:

```
python app.py PORT IP NumberOfChildren true
```

με τα κατάλληλα ορίσματα στο PORT, IP και NumberOfChildren. Το όρισμα true δηλώνει ότι ο κόμβος αυτός θα είναι ο bootstrap.

Στη συνέχεια εισάγουμε τόσα παιδιά όσα και το NumberOfChildren με την ίδια εντολή με την παραπάνω, με τη διαφορά ότι αντί για true ορίζουμε no. Σημειώνεται ότι εάν θέλουμε να τρέξουμε τα πειράματα που μας έχουν δώσει μέσα από τα αρχεία 5nodes και 10nodes, δίνουμε τις ίδιες αντολές απλά app\_auto.py αντί για app.py και ορίζουμε στο αρχείο node.py στη συνάρτηση all\_files\_nodes τον φάκελο που θέλουμε.

## Περιγραφή Συστήματος

Το σύστημά μας ορίζεται στα κομμάτια app, node, blockchain, transaction, block, cli και frontend. Ακολουθεί η ανάλυση των διαδικασιών που έχουν υλοποιηθεί σε καθένα από αυτά. Σημειώνεται ότι αρχετές διαδικασίες θα τρέχουν παράλληλα. Για τον σκοπό αυτό έγινε η χρήση των Threads.

### App

Στο app.py έχει στηρθεί το API του συστήματός μας. Έχουν υλοποιηθεί τα endpoints από το backend, το cli και το frontend. Για την εκκίνηση του αφού ο χρήστης δώσει τις εντολές που αναφέρθηκαν παραπάνω καλείται η κλάση node.py και ξεκινάει η λειτουργία του noobcash.

### Node

Αντικείμενα της κλάσης:

Αντικείμενο	Ρόλος
port	Port του Node
ip	IP του Node
children	Αριθμός των nodes (εκτός του bootstrap)
public_key	Δημόσιο Κλειδί
private_key	Ιδιωτικό Κλειδί
ring	Λίστα με όλες τις IP των συνδεδεμένων κόμβων
public_keys	Λίστα με τα public keys των υπόλοιπων κόμβων
unspent_coins	Διαθέσιμα coins του node
transactions_dictionary	Λίστα με λίστες από τα public_keys - unspent coins
buffer	Λίστα με transactions που είναι στην ουρά για εκτέλεση
chain	Το μέχρι τώρα blockchain
all_nodes_chains	Dictionary για αποθήκευση των chains σε περίπτωση conflict
all_nodes_transactions	Dictionary για αποθήκευση όλων των transactions
all_utxos	Dictionary με τα utxos του κάθε node.

- Αρχικά εισέρχεται ο bootstrap node, στον οποίο αρχικοποιούνται τα χρήματα που θα μοιράσει στα παιδιά. Δημιουργεί το genesis block, μέσω της συνάρτησης genesis\_block\_new της κλάσης Transaction. Παράλληλα (χρήση thread) αναμένει την εισαγωγή όλων των παιδιών.
- Κάθε φορά που εισέρχεται ένα παιδί στο σύστημα ενημερώνει τον bootstrap σχετικά με την IP και το δημόσιο κλειδί του.
- Όταν εισέλθουν όλα τα παιδιά ο bootstrap ξεκινάει να τους αναθέτει IDs και τους στέλνει από 100 NBC.

Ακολουθεί η επεξήγηση των συναρτήσεων της κλάσης Node:

- **generate\_wallet:** Δημιουργία ιδιωτικού και δημοσίου κλειδιού για κάθε κόμβο που εισέρχεται στο σύστημα.
- **register\_child:** Προσθήκη της IP του node που εισέρχεται στο ring και του public\_key στη λίστα με τα public\_keys.
- **cont:** Αναμονή μέχρι να εισέλθουν όλα τα παιδιά στο σύστημα (ενεργοποίηση thread). Έπειτα τους διανέμονται από τον bootstrap κόμβο από 100 NBC.
- **create\_transaction:** Συνάρτηση δημιουργίας συναλλαγής. Δέχεται ως ορίσματα το ID του παραλήπτη και το ποσό που του αποστέλεται. Αρχικά ελέγχεται εάν το ποσό που αποστέλεται είναι επαρκές. Εάν επαρκούν δημιουργείται ένα αντικείμενο κλάσης Transaction, ενημερώνονται οι κατάλληλες μεταβλητές, υπογράφεται η συναλλαγή, γίνεται broadcast σε όλο το δίκτυο και προστίθεται στο block.
- **child\_response:** Ο εισερχόμενος κόμβος ενημερώνεται σχετικά με την ύπαρξη του genesis block, αρχικοποιεί ται το transactions\_dictionary με τα συνολικά χρήματα που πρέπει να στείλει ο bootstrap node στα παιδιά και ο κόμβος πλέον είναι έτοιμος να δεχθεί χρήματα.
- **broadcast\_transaction:** Η συναλλαγή που μόλις έχει δημιουργηθεί αποστέλεται σε όλους τους κόμβους εκτός από τον αποστολέα.
- **threadone\_wait:** Η συνάρτηση αυτή τρέχει συνεχώς όταν ο πατέρας στείλει τα αρχικά χρήματα σε όλα τα παιδιά και τα παιδιά τα λάβουν (ενεργοποίηση thread). Σε περίπτωση που δεν έχουμε mine ή consensus και ο buffer δεν είναι άδειος (εκρεμούν συναλλαγές για επικύρωση) δημιουργείται ένα αντικείμενο κλάσης Transaction. Ελέγχεται εάν είναι έγκυρη (μέσω της υπογραφής) και εάν είναι καταφράγεται ο χρόνος της μέχρι να ολοκληρωθεί.
- **wallet\_balance:** Επιστρέφεται το υπόλοιπο των χρημάτων του κόμβου.
- **validate\_transaction:** Επικύρωση της συναλλαγής μέσω της υπογραφής της και του ελέγχου των inputs και outputs.
- **validate\_block:** Εδώ πραγματοποιείται η διαδικασία του mining.
- **resolve\_conflict:** Αυτή η συνάρτηση καλείται όταν ένα κόμβος λάβει ένα block το οποίο δεν μπορεί να κάνει validate γιατί το πεδίο previous\_hash δεν ισούται με το hash του προηγούμενου block. Αυτό 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.
- **all\_files\_nodes** Η συνάρτηση αυτή καλείται όταν ενεργοποιηθεί το thread file\_runs από το αρχείο app.py και τρέχει όλες τις συναλλαγές στο file που ορίζουμε. Χρησιμοποιήθηκε για τις συναλλαγές στα αρχεία 5nodes και 10nodes.

## Transaction

Αντικείμενα της κλάσης:

Αντικείμενο	Ρόλος
sender	ID του αποστολέα της συναλλαγής
receiver	ID του παραλήπτη της συναλλαγής
coins	Το ποσό που αποστέλεται (NBC)
inputs	Λίστα με UTXOs (debits)
outputs	Λίστα με UTXOs (credits)
signature	Η υπογραφή της συναλλαγής
transaction_id	To id της συναλλαγής

Ακολουθεί η επεξήγηση των συναρτήσεων της κλάσης Transaction:

- **sign\_transaction:** Κάθε νέα συναλλαγή υπογράφεται με το ιδιωτικό κλειδί του αποστολέα.
- **verify\_signature:** Επαλήθευση της υπογραφής της συναλλαγής.
- **hashing:** Υλοποίηση hashing.
- **convert\_to\_JSON:** Μετατροπή του transaction σε json μορφή για την ευκολότερη μεταφορά της πάνω από το δίκτυο.

## Blockchain

Αντικείμενα της κλάσης:

Αντικείμενο	Ρόλος
transactions_list	Λίστα με τις συναλλαγές του Blockchain
blocks_list	Λίστα με τα blocks του Blockchain
new_mine_thread	Thread που χρησιμοποιείται για το mining

Ακολουθεί η επεξήγηση των συναρτήσεων της κλάσης Blockchain:

- **genesis\_block\_new:** Συνάρτηση δημιουργίας του genesis block, το οποίο είναι το μόνο που δεν επαληθεύεται.
- **put\_transaction\_inblock:** Έλεγχος εάν υπάρχει χώρος στο τρέχον block για την εισαγγή της νέας συναλλαγής. Ο χώρος καθορίζεται από τη σταθερά CAPACITY στην αρχή του αρχείου blockchain.py. Εάν δεν υπάρχει χώρος ξεκινάει η διαδικασία του mining.
- **domine:** Υλοποίηση της διαδικασία του mining, μόνο όταν το thread new\_mine\_thread ενεργοποιηθεί.
- **convert\_b:** Μετατροπή του block σε json μορφή για την ευκολότερη μεταφορά του πάνω από το δίκτυο.
- **parameters\_copy:** Αντιγραφή του ID και του ring από την κάση Node.

## Block

Αντικείμενα της κλάσης:

Αντικείμενο	Ρόλος
index	Ο αύξων αριθμός του block
timestamp	To timestamp της δημιουργίας του block.
transactions	Οι συναλλαγές που περιέχονται στο block.
nonce	Η λύση του proof of work
previous_hash	To hash του προειγούμενου block στο Blockchain current_hash   To hash του block.

Ακολουθεί η επεξήγηση των συναρτήσεων της κλάσης Block:

- **mine\_block:** Συνάρτηση mining του block, χρησιμοποιώντας τα πρώτα MINING\_DIFFICULTY ψηφία του hash του block. Η σταθερά MINING\_DIFFICULTY ορίζεται στην αρχή του αρχείου block.py.
- **hashing:** Hashing του τρέχοντος Block.
- **convert\_block:** Μετατροπή του block σε json μορφή για την ευκολότερη μεταφορά του πάνω από το δίκτυο.

Για τη χρήση του συστήματος υλοποιήθηκε ένα απλό CLI και μία γραφική διεπαφή. Προσφέρουν τις ίδιες δυνατότητες και τα δύο. Οι δυνατότητες που υλοποιήθηκαν είναι οι εξής:

- Δημιουργία νέου Transaction.
- Προβολή των transactions που περιέχονται στο τελευταίο επικυρωμένο block του noobcash blockchain.
- Εμφάνιση του υπολοίπου του wallet.
- Εμφάνιση οδηγιών σχετικά με τη χρήση του cli και της διεπαφής.

Ακολουθεί η παρουσίασή τους:

## CLI

Για την εκκίνηση του CLI, εκτελούμε στον φάκελο του project την εντολή python cli.py PORT IP με τα στοιχεία PORT και IP του node που θέλουμε να το εκκινήσει.

Ο χρήστης μπορεί να δώσει τις ακόλουθες εντολές:

- **help** για προβολή των διαθέσιμων ενεργειών

```

help

Usage:
$ python app.py HOST PORT NUMBER_OF_CHILDREN_NODES IP           Run as child node, N children nodes, IP
$ python app.py HOST PORT N IP                                Run as bootstrap node, for N children nodes, IP of bootstrap

Available commands:
* `t [recipient_id] [amount]'                                Send 'amount' NBC to 'recipient' node
* `view`                                                       View transactions of the latest block
* `balance`                                                    View balance of each wallet (as of last validated block)
* `help`                                                       Print this help message
* `exit`                                                       Exit client (will not stop server)

Enter a desired action! Type help if want to know the available actions!

```

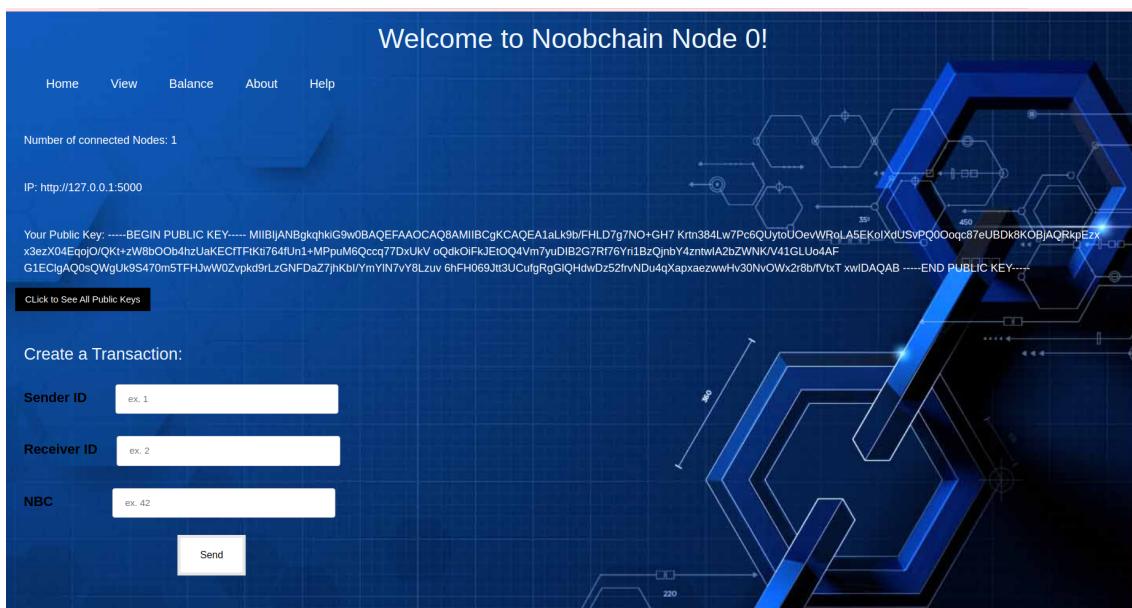
(a) CLI help

- **view:** Προβολή των transactions που περιέχονται στο τελευταίο επικυρωμένο block του noobcash blockchain.
- **balance:** Εμφάνιση του υπολοίπου του wallet.
- **t <recipient\_address> <amount>:** To <recipient\_address> είναι το id του κόμβου που θέλουμε να στείλουμε χρήματα και <amount> τα NBC (πχ. t 2 3).
- **exit:** Έξοδος από το σύστημα.

## Web App

Η Web εφαρμογή διαθέτει τις λειτουργίες από αναφέρθηκαν παραπάνω. Ακολουθεί η παρουσίαση των διαθέσιμων σελίδων:

- Αρχική Σελίδα - Δημιουργία Transaction.



(a) Homepage

- Σελίδα Balance (/balance)



(a) Balance Page

- Σελίδα View (/view): Εάν δεν έχει πραγματοποιηθεί κάποια συναλλαγή ο χρήστης δεν έχει πρόσβαση.



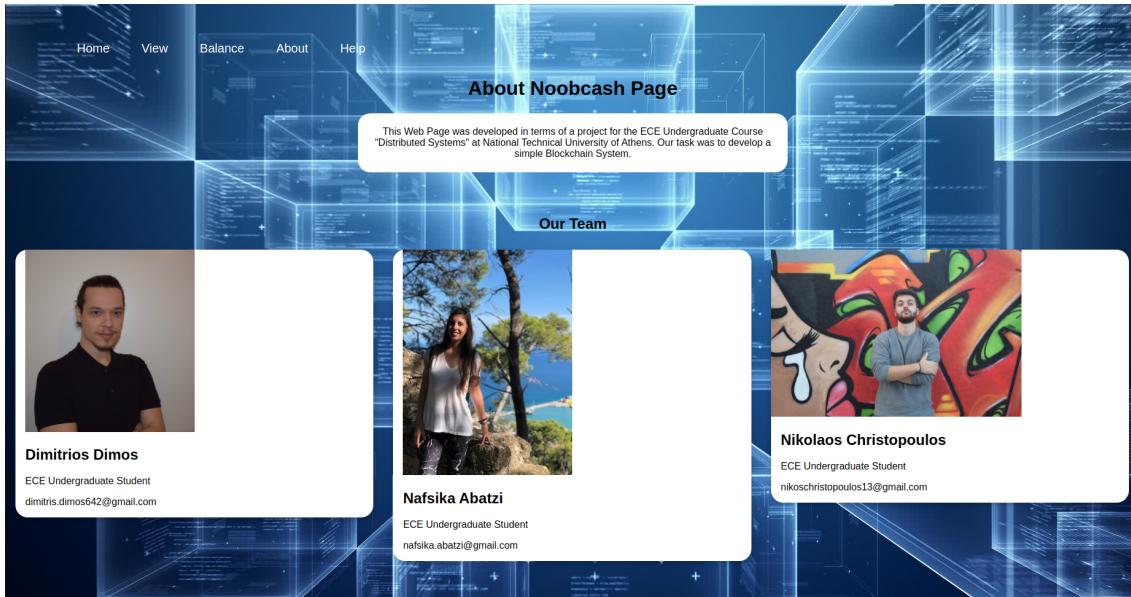
(a) View Page

- Σελίδα Help (/help)



(a) Help Page

- Σελίδα About (/about)



(a) About Page

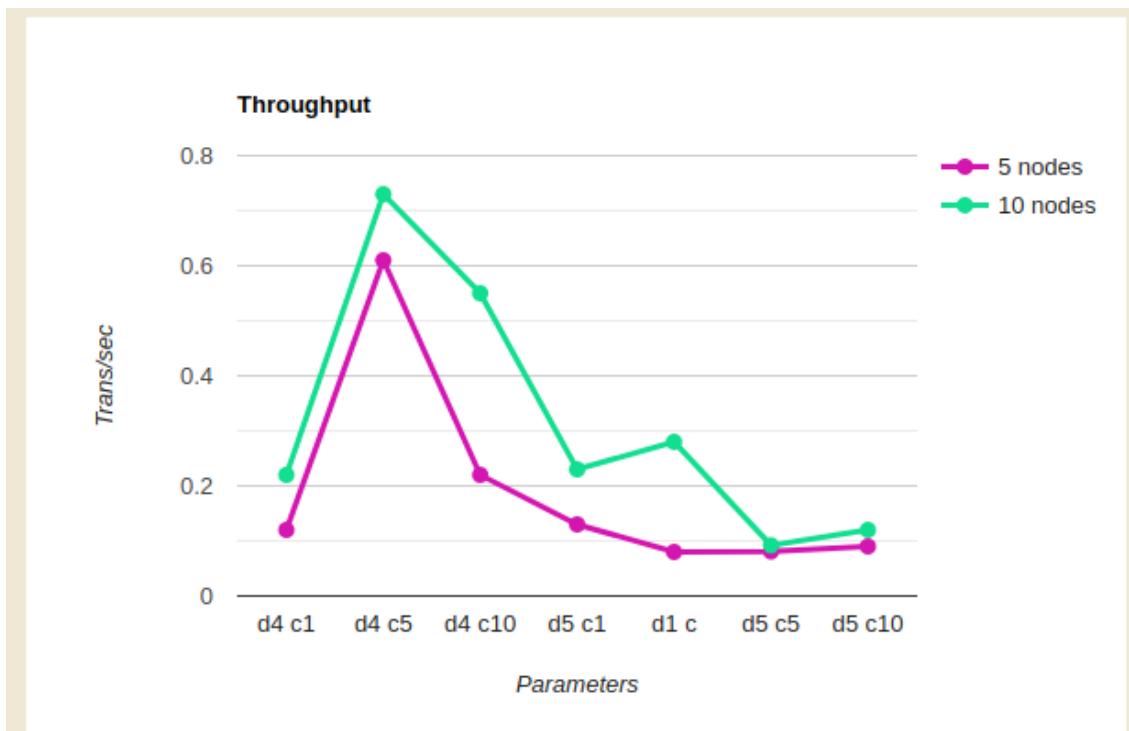
## Πειράματα

Μελετήθηκε η απόδοση και η κλιμακωσμότητα του συστήματός μας σε 5 Virtual Machines που μας δώθηκαν.

## Απόδοση

Στήσαμε ένα noobcash με 5 clients, οι οποίοι διάβαζαν transactions από τον φάκελο 5nodes. Για (a) capacity 1, 5 και 10 και (b) difficulty 4 και 5 καταγράψαμε τα παρακάτω:

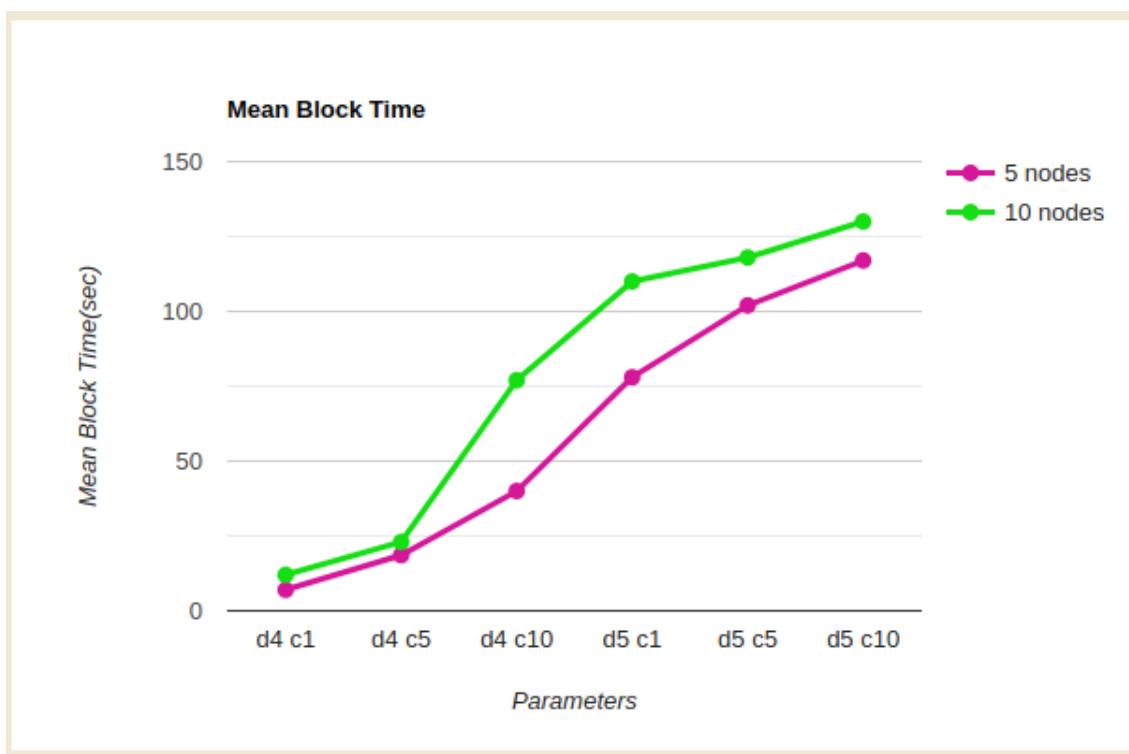
- Throughput (ρυθμαπόδοση) του συστήματός εξυπηρετούνται στην μονάδα του χρόνου.
- Block time, δηλαδή τον μέσο χρόνο που απαιτείται για να προστεθεί ένα νέο block στο blockchain.



(a) Throughput

### Κλιμακωσιμότητα

Για τις ίδιες παραμέτρους τρέξαμε το σύστημα εισάγοντας 10 κόμβους και πήραμε τα εξής αποτελέσματα:



(a) Mean Block Time