Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστικων Συστηματων

# Benchmarking Αλγορίθμων Consensus στο Ethereum

Διπλωματικη Εργασια

**κωστουλα βασιλικη**

**Επιβλέπων:** Κοζύρης Νεκτάριος
Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογιας Πληροφορικής και Υπολογιστικων Συστηματων

# Benchmarking Αλγορίθμων Consensus στο Ethereum

Διπλωματικη Εργασια

**κωστουλα βασιλικη**

**Επιβλέπων:** Κοζύρης Νεκτάριος
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14 Ιουλίου 2023.

| *(Υπογραφή)* | *(Υπογραφή)* | *(Υπογραφή)* |
|---|---|---|
| ..... | ..... | ..... |
| Κοζύρης Νεκτάριος | Γκούμας Γεώργιος | Δημήτριος Τσουμάκος |
| Καθηγητής ΕΜΠ | Αναπληρωτής Καθηγητής ΕΜΠ | Αναπληρωτής Καθηγητής ΕΜΠ |

Αθήνα, Ιούλιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογιας Πληροφορικής και Υπολογιστικων Συστηματων

(Υπογραφή)

.....
ΚΩΣΤΟΥΛΑ ΒΑΣΙΛΙΚΗ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογοστών Ε.Μ.Π.

# Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει μια συγκριτική μελέτη τριών αλγορίθμων συναίνεσης στο Ethereum: Proof of Work (PoW), Proof of Stake (PoS) και Byzantine Fault Tolerance (BFT). Οι αλγόριθμοι συναίνεσης διαδραματίζουν κρίσιμο ρόλο στα συστήματα blockchain, επιτρέποντας την επικύρωση των συναλλαγών και τη συμφωνία σχετικά με την κατάσταση του blockchain μεταξύ των συμμετεχόντων κόμβων στο δίκτυο. Δεδομένης της αυξανόμενης δημοτικότητας της τεχνολογίας blockchain και των πιθανών εφαρμογών της σε διάφορους τομείς, είναι απαραίτητο να αξιολογηθεί η απόδοση των διαφόρων αλγορίθμων συναίνεσης για να καθοριστεί ποιος ταιριάζει καλύτερα σε συγκεκριμένες περιπτώσεις χρήσης.

Η παρούσα μελέτη επικεντρώνεται στη συγκριτική αξιολόγηση των αλγορίθμων για τη μέτρηση της καθυστέρησης (latency) και της ρυθμοαπόδοσής (throughput) τους με τη χρήση του Blockbench, ενός framework για την ανάλυση ιδιωτικών blockchain. Τα πειράματα διεξήχθησαν σε δίκτυα με 6 και 12 κόμβους για να αξιολογηθεί η επεκτασιμότητα κάθε αλγορίθμου. Στόχος της μελέτης είναι να παράσχει πληροφορίες σχετικά με τα δυνατά και αδύνατα σημεία κάθε αλγορίθμου συναίνεσης και να ενημερώσει για την επιλογή ενός αλγορίθμου συναίνεσης σε συστήματα blockchain. Τα αποτελέσματα αυτής της μελέτης μπορεί να ενδιαφέρουν τους προγραμματιστές και τους ερευνητές που ασχολούνται με την επεκτασιμότητα και τις επιδόσεις του blockchain.

## Λέξεις Κλειδιά

Ethereum, Ιδιωτικά Blockchain, Αλγόριθμοι συναίνεσης, Proof of Work, Proof of Stake, Byzantine Fault Tolerance, Απόδοση, Καθυστέρηση, Ρυθμαπόδοση, Επεκτασιμότητα

# Abstract

This diploma thesis presents a comparative study of three consensus algorithms in Ethereum: Proof of Work (PoW), Proof of Stake (PoS), and Byzantine Fault Tolerance (BFT). Consensus algorithms play a critical role in blockchain systems by enabling the validation of transactions and the agreement on the state of the blockchain among network participants. Given the increasing popularity of blockchain technology and its potential applications in various domains, it is essential to evaluate the performance of different consensus algorithms to determine which one best fits specific use cases.

This study focuses on benchmarking the algorithms to measure their latency and throughput using Blockbench, a framework for analyzing private blockchains. The experiments were conducted on networks with 6 and 12 nodes to assess the scalability of each algorithm. The study aims to provide insights into the strengths and weaknesses of each consensus algorithm, and to inform the selection of a consensus algorithm in blockchain systems. The results of this study may be of interest to developers and researchers working on blockchain scalability and performance.

## Keywords

Ethereum, Private Blockchain, Consensus Algorithms, Proof of Work, Proof of Stake, Byzantine Fault Tolerance, Performance, Latency, Throughput, Scalability

# Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Νεκτάριο Κοζύρη για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο Υπολογιστικών Συστημάτων. Επίσης ευχαριστώ ιδιαίτερα την Δρ. Κατερίνα Δόκα για την καθοδήγησή και την βοήθεια που μου προσέφερε καθ΄ όλη τη διάρκεια της διπλωματικής. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια και τους φίλους μου οι οποίοι υπήρξαν πυλώνες δύναμης κατά τη διάρκεια αυτού του ταξιδιού.

Αθήνα, Ιούλιος 2023

*Κωστούλα Βασιλική*

# Περιεχόμενα

# Κατάλογος Εικόνων

# Κεφάλαιο 1

# Εισαγωγή

Το 2009, ένα ανώνυμο πρόσωπο ή ομάδα που χρησιμοποιούσε το ψευδώνυμο Satoshi Nakamoto εισήγαγε την τεχνολογία blockchain, η οποία όντας μια επαναστατική τεχνολογία που έχει τη δυνατότητα να προσφέρει σε διάφορους κλάδους κέντρισε έκτοτε το ενδιαφέρον της επιστημονικής κοινοτητα και ιδιαίτερα τους κλάδους της Πληροφορικής και της Οικονομίας. Τα συστήματα blockchain είναι αποκεντρωμένα, διαφανή και ανθεκτικά στις παραβιάσεις, παρέχοντας έτσι ασφαλείς και αξιόπιστες συναλλαγές χωρίς την ανάγκη μεσαζόντων. Η δυνατότητα εκτέλεσης αποκεντρωμένων εφαρμογών, (DApps), σε δίκτυα blockchain είναι ένα βασικό χαρακτηριστικό που οδήγησε στη δημοτικότητα του Ethereum, μιας από τις πιο ευρέως χρησιμοποιούμενες πλατφόρμες blockchain.

Το Ethereum είναι μοναδικό ως προς την ικανότητά του να υποστηρίζει έξυπνα συμβόλαια, τα οποία είναι αυτοεκτελούμενες συμβάσεις με τους όρους της συμφωνίας γραμμένους απευθείας στον κώδικα. Αυτή η δυνατότητα επιτρέπει στους προγραμματιστές να δημιουργούν DApps που μπορούν να λειτουργούν ανεξάρτητα από μια κεντρική αρχή, με διάφορες εφαρμογές σε τομείς όπως η χρηματοδότηση, η διαχείριση του supply chain και η επαλήθευση της ταυτότητας. Ωστόσο, το scalability του Ethereum και άλλων συστημάτων blockchain παραμένει μια σημαντική πρόκληση, καθώς η απόδοση του δικτύου μπορεί να μειωθεί καθώς αυξάνεται ο αριθμός των χρηστών και των συναλλαγών.

Για την αντιμετώπιση αυτής της πρόκλησης, είναι απαραίτητο να αξιολογηθεί η απόδοση των αλγορίθμων συναίνεσης, οι οποίοι είναι υπεύθυνοι για την επικύρωση των συναλλαγών και την επίτευξη συμφωνίας σχετικά με την κατάσταση του blockchain μεταξύ των συμμετεχόντων κόμβων στο δίκτυο.

## 1.1 Αντικείμενο της διπλωματικής

Στόχος της παρούσας διπλωματικής εργασίας είναι η σύγκριση και η συγκριτική αξιολόγηση τριών αλγορίθμων συναίνεσης στο Ethereum: Proof of Work (PoW), Proof of Stake (PoS) και Byzantine Fault Tolerance (BFT).

Διεξάγοντας πειράματα και αναλύοντας την καθυστέρηση και την απόδοση του κάθε αλγορίθμου χρησιμοποιώντας το εργαλείο Blockbench, η παρούσα μελέτη έχει ως στόχο να παράσχει πληροφορίες σχετικά με την απόδοση και την επεκτασιμότητά κάθε αλγορίθμου συναίνεσης σε διαφορετικές συνθήκες δικτύου. Τα αποτελέσματα αυτής της μελέτης μπορούν να ενημερώσουν για την επιλογή αλγορίθμων συναίνεσης για συγκεκριμένες περιπτώσεις χρήσης

καθώς και να συμβάλουν στην ανάπτυξη πιο κλιμακούμενων και αποδοτικών συστημάτων blockchain.

## 1.2 Οργάνωση του τόμου

Η διπλωματική οργανώνεται ως εξής:

Το **Κεφάλαιο 2** αποτελεί το θεωρητικό υπόβαθρο της διπλωματικής και περιλαμβάνει αρχικά την περιγραφή του blockchain του Ethereum, τον ορισμό και τη χρησιμότητα των αλγορίθμων ομοφωνίας καθώς και περιγραφή του Blockbench. Επιπλέον γίνεται εκτενέστερη περιγραφή των χρησιμοποιούμενων αλγορίθμων ομοφωνίας και των βασικών τους στοιχείων.

Στο **Κεφάλαιο 3** παρουσιάζεται αναλυτικά το στήσιμο των δικτύων blockchain καθώς και το απαραίτητο software για την υλοποίηση τόσο των τριών διαφορετικών δικτύων όσο και του εργαλείου που χρησιμοποιείται για το benchmarking. Στο **Κεφάλαιο 4** πραγματοποιείται σύγκριση των επιμέρους δικτύων με τους τρεις διαφορετικούς αλγορίθμους ομοφωνίας και αξιολόγηση της συμπεριφοράς των συστημάτων.

Το **Κεφάλαιο 5** περιλαμβάνει ιδέες για μελλοντικές επεκτάσεις της υλοποίησης ενώ το **Κεφάλαιο 6** αποτελεί τον επίλογο με τα συμπεράσματα της εργασίας.

# Μέρος I

# Θεωρητικό Μέρος

# Κεφάλαιο 2

# Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται αρχικά κάποια βασική θεωρία σχετικα με την τεχνολογία του blockchain και πιο συγκεκριμένα του blockchain του Ethereum. Στη συνέχεια γίνεται ανάλυση των αλγορίθμων consensus που θα χρησιμοποιήσουμε καθώς και της θεωρίας που κρύβεται πίσω τους. Τέλος γίνεται μελέτη του framework του Blockbench.

## 2.1 Ethereum

### 2.1.1 Εισαγωγή στο blockchain

Η τεχνολογία blockchain είναι ένα κατανεμημένο σύστημα αποθήκευσης και μετάδοσης δεδομένων που είναι ανθεκτικό σε παραβιάσεις ενώ παράλληλα λειτουργεί με πλήρη διαφάνεια. Λειτουργει μέσω ενός αποκεντρωμένου δικτύου κόμβων που συντηρούν συλλογικά ένα ψηφιακό βιβλίο συναλλαγών, το οποίο μπορεί να χρησιμοποιηθεί για την επικύρωση και την πιστοποίηση της ακεραιότητας των πραγματοποιούμεων συναλλαγών καθώς και των ψηφιακών περιουσιακών στοιχείων των υπαρχόντων στο εκάστοτε blockchain λογαριασμών.

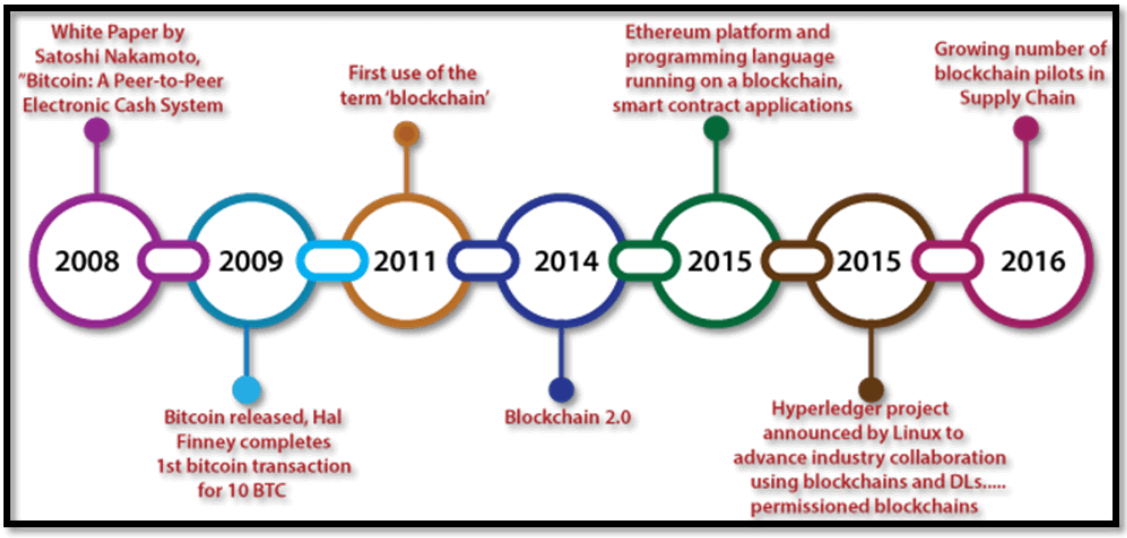
Η έννοια του blockchain συστήθηκε το 2008 από ένα ανώνυμο άτομο ή ομάδα που χρησιμοποιούσε το ψευδώνυμο "Satoshi Nakamoto" σε ένα paper με τίτλο "Bitcoin: A Peer-to-Peer Electronic Cash System". Στη συνέχεια ένα χρόνοα το αργότερα το 2009 δόθηκε ανοιχτά στο διαδίκτυο ο πηγαίος κώδικας του bitcoin και από τη θεωρία προχωρήσαμε στην πράξη.

Το Bitcoin χρησιμοποιεί έναν αλγόριθμο συναίνεσης προοφ-οφ-ωορκ, ο οποίος απαιτεί από τους συμμετέχοντες κόμβους στο δίκτυο να επιλύουν σύνθετα μαθηματικά προβλήμ1ατα για να επικυρώνουν τις συναλλαγές και να τις προσθέτουν στο blockchain.

Το 2015 ξεκίνησε η αλυσίδα μπλοκ Ethereum, εισάγοντας την έννοια των έξυπνων συμβάσεων. Τα έξυπνα συμβόλαια είναι αυτοεκτελούμενες συμβάσεις με τους όρους της συμφωνίας γραμμένους απευθείας στον κώδικα. Αυτή η δυνατότητα επιτρέπει στους προγραμματιστές να δημιουργούν αποκεντρωμένες εφαρμογές (DApps) που μπορούν να λειτουργούν ανεξάρτητα από μια κεντρική αρχή. Το Ethereum χρησιμοποιεί επί του παρόντος έναν αλγόριθμο συναίνεσης proof-of-work, αλλά βρίσκεται σε διαδικασία μετάβασης σε έναν αλγόριθμο προοφ-οφ-στακε, ο οποίος θα βελτιώσει την επεκτασιμότητα και την ενεργειακή αποδοτικότητα.

Έχουν επίσης εμφανιστεί και άλλες πλατφόρμες blockchain, η καθεμία με τα δικά της μοναδικά χαρακτηριστικά και περιπτώσεις χρήσης. Η Ripple, για παράδειγμα, επικεντρώνεται στη διευκόλυνση των διασυνοριακών πληρωμών, ενώ η Hyperledger έχει σχεδιαστεί για χρήση σε επιχειρηματικές εφαρμογές.

Η ασφάλεια και η διαφάνεια της τεχνολογίας blockchain την καθιστούν μια ελκυστική επιλογή για τις επιχειρήσεις και τις κυβερνήσεις που επιδιώκουν τη βελτίωση της αποτελεσματικότητας και τη μείωση του κόστους. Ωστόσο, υπάρχουν ακόμη προκλήσεις που πρέπει να ξεπεραστούν όσον αφορά την επεκτασιμότητα, τη ρύθμιση και τη διαλειτουργικότητα. Η συνεχής έρευνα και ανάπτυξη σε αυτόν τον ταχέως εξελισσόμενο τομέα οδηγεί στη δυνατότητα της τεχνολογίας blockchain να φέρει επανάσταση σε διάφορους κλάδους, από τη χρηματοδότηση και τη διαχείριση της εφοδιαστικής αλυσίδας έως την επαλήθευση της ψηφιακής ταυτότητας και τη διακυβέρνηση.




Εικόνα 2.1: *Blockchain Timeline*

### 2.1.2 Το blockchain του Ethereum

Το Ethereum είναι μια αποκεντρωμένη πλατφόρμα blockchain που επιτρέπει την εκτέλεση smart contracts και την ανάπτυξη αποκεντρωμένων εφαρμογών (DApps). Είναι χτισμένο σε μια αρχιτεκτονική blockchain παρόμοια με το Bitcoin, αλλά διακρίνεται για την προγραμματισιμότητα και την ευελιξία του.

Στον πυρήνα του blockchain του Ethereum βρίσκεται το Ethereum Virtual Machine (EVM), ένα Turing-complete περιβάλλον εκτέλεσης πάνω στο οποίο εκτελούνται τα smart contracts. Το EVM επιτρέπει στους προγραμματιστές να γράφουν κώδικα σε γλώσσες υψηλού επιπέδου, όπως η Solidity, η οποία έχει σχεδιαστεί ειδικά για την ανάπτυξη smart contracts στην πλατφόρμα Ethereum. Τα smart contracts αποτελούν στην πράξη αυτοεκτελούμενες συμφωνίες με προκαθορισμένους κανόνες και όρους. Αυτό σημαίνει ότι μόλις ένα έξυπνο συμβόλαιο γίνει deploy στο blockchain, εκτελεί αυτόματα τις συμφωνημένες ενέργειες χωρίς να χρειάζονται μεσάζοντες ή τρίτα μέρη για να επιβλέπουν ή να επιβάλλουν τους όρους του συμβολαίου. Αυτό εξαλείφει την ανάγκη για μεσάζοντες, μειώνοντας το κόστος, εξορθολογίζοντας τις διαδικασίες και παρέχοντας στους συμμετέχοντες διαφάνεια, καθώς οι όροι του contract κωδικοποιούνται απευθείας στο blockchain, διασφαλίζοντας ότι δεν μπορούν να τροποποιηθούν ή να αλλοιωθούν.

Ένα από τα σημαντικά πλεονεκτήματα του Ethereum είναι η ικανότητά του να υποστηρίζει

αποκεντρωμένες εφαρμογές (DApps). Αξιοποιούν την αποκεντρωμένη φύση του blockchain και τις δυνατότητες των smartcontracts για να προσφέρουν ένα ευρύ φάσμα λειτουργιών. Σε αντίθεση με τις παραδοσιακές εφαρμογές που βασίζονται σε κεντρικούς διακομιστές και μεσάζοντες, οι αποκεντρωμένες εφαρμογές αξιοποιούν το blockchain ως αποκεντρωμένη και ανθεκτική στην παραποίηση βάση δεδομένων. Χρησιμοποιώντας smart contracts, οι DApps μπορούν να αλληλεπιδρούν με το δίκτυο Ethereum και να αποθηκεύουν και να ανακτούν δεδομένα με ασφάλεια. Η διαφάνεια και το αμετάβλητο των διαδικασιων παρέχουν εμπιστοσύνη και ασφάλεια στους χρήστες. Τα παραδείγματα DApps στην πλατφόρμα Ethereum είναι πολυάριθμα και καλύπτουν διάφορους κλάδους και περιπτώσεις χρήσης. Ακολουθούν μερικά αξιοσημείωτα παραδείγματα:

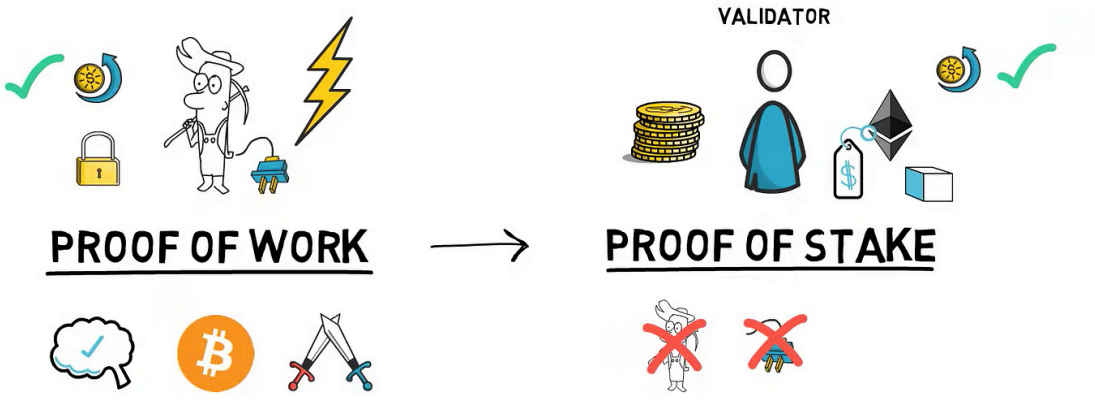
- **Uniswap** Το Uniswap είναι ένα αποκεντρωμένο πρωτόκολλο ανταλλαγής που επιτρέπει στους χρήστες να ανταλλάσσουν κρυπτονομίσματα με βάση τοEthereum απευθείας από τα πορτοφόλια τους, χωρίς την ανάγκη διαμεσολαβητών.

- **Aave** Το Aave είναι μια αποκεντρωμένη πλατφόρμα δανεισμού και δανειοδότησης πουεπιτρέπει στους χρήστες να δανείζουν και να δανείζονται κρυπτονομίσματα αλληλεπιδρώντας με smart contracts. Επιτρέπει στους χρήστες να κερδίζουν τόκους επί των κατατεθειμένων περιουσιακών στοιχείων τους και παρέχει ρευστότητα στους δανειολήπτες.

- **MakerDAO** Το MakerDAO είναι ένας αποκεντρωμένος αυτόνομος οργανισμός που διαχειρίζεται το stablecoin Dai. Χρησιμοποιεί smart contracts για τη διατήρηση της σταθερότητας των τιμών, επιτρέποντας στους χρήστες να παράγουν Dai κλειδώνοντας τα περιουσιακά τους στοιχεία στο Ethereum.

Το Ethereum μέχρι πρότινος χρησιμοποιούσε έναν αλγόριθμος συναίνεσης που ονομάζεται Proof of Work (PoW) για την επικύρωση και την προσθήκη συναλλαγών στο blockchain. Στο μοντέλο PoW, οι εξορύκτες miners ανταγωνίζονται για την επίλυση υπολογιστικά απαιτητικών γρίφων, με τον πρώτο εξορύκτη που θα βρει τη λύση να ανταμείβεται με Ether (ETH). Αυτή η διαδικασία εξασφαλίζει την ασφάλεια και το αμετάβλητο του blockchain, καθώς η αλλαγή μιας παρελθούσας συναλλαγής θα απαιτούσε σημαντική υπολογιστική ισχύ και πόρους.

Ενώ το Ethereum με τη χρήση proof of work αλγορίθμου συναίνερης έχει επιτύχει να φέρει επανάσταση στην τεχνολογία blockchain και να επιτρέψει την ανάπτυξη πολυάριθμων DApps και tokens, αντιμετωπίζει προκλήσεις επεκτασιμότητας. Ο αλγόριθμος συναίνεσης PoW περιορίζει τον αριθμό των συναλλαγών που μπορούν να υποβληθούν σε επεξεργασία ανά δευτερόλεπτο, οδηγώντας σε συμφόρηση του δικτύου σε περιόδους υψηλής δραστηριότητας. Επιπλέον, τα τέλη συναλλαγών (transaction fees) μπορεί να γίνουν απαγορευτικά υψηλά, ιδίως σε περιόδους συμφόρησης του δικτύου.Για την αντιμετώπιση αυτών των προκλήσεων, το Ethereum πρόσφατα υποβλήθηκε σε μια σημαντική αναβάθμιση που αποσκοπεί στην εισαγωγή ενός νέου αλγορίθμου συναίνεσης που ονομάζεται Proof of Stake (PoS). Η λεπτομερής ανάλυση και σύγκριση των αλγορίθμων αυτών καθως και άλλων αλγορίθμων συναινεσης θα πραγματοποιηθεί στη συνέχεια αφου γίνει μια λεπτομερής εισαγωγή στους αλγορίθμους συναίνεσης.




Εικόνα 2.2: *PoW vs PoS*

## 2.2 Αλγόριθμοι συναίνεσης

Οι αλγόριθμοι συναίνεσης είναι μηχανισμοί που χρησιμοποιούνται σε κατανεμημένα συστήματα για την επίτευξη συμφωνίας μεταξύ των συμμετεχόντων σχετικά με την κατάσταση του συστήματος ή την εγκυρότητα ορισμένων συναλλαγών. Αυτοί οι αλγόριθμοι διαδραματίζουν κρίσιμο ρόλο στη διατήρηση της ακεραιότητας, της ασφάλειας και της συνέπειας των κατανεμημένων βιβλίων, όπως τα δίκτυα blockchain.

Οι αλγόριθμοι συναίνεσης μπορούν να εφαρμοστούν σε διάφορους τομείς πέραν της τεχνολογίας blockchain, οπουδήποτε πολλαπλοί συμμετέχοντες πρέπει να συμφωνήσουν σε μια κοινή κατάσταση ή σε ένα σύνολο κανόνων. Για παράδειγμα, χρησιμοποιούνται σε κατανεμημένες βάσεις δεδομένων, ομότιμα δίκτυα και συστήματα Internet of Things (IoT).

Στο πλαίσιο της τεχνολογίας blockchain, οι αλγόριθμοι συναίνεσης είναι απαραίτητοι για την επίτευξη συμφωνίας σχετικά με τη σειρά και την εγκυρότητα των συναλλαγών που καταγράφονται στην blockchain. Επιτρέπουν στα αποκεντρωμένα δίκτυα να επιτυγχάνουν συναίνεση χωρίς να βασίζονται σε μια κεντρική αρχή. Με τη δημιουργία συμφωνίας, οι αλγόριθμοι συναίνεσης αποτρέπουν τη διπλή δαπάνη, την απάτη και διασφαλίζουν το αμετάβλητο του blockchain.

Το πρόβλημα της διπλής δαπάνης αναφέρεται στον κίνδυνο κάποιος χρήστης να ξοδέψει το ίδιο κρυπτονόμισμα ή ψηφιακό περιουσιακό στοιχείο περισσότερες από μία φορές. Το ζήτημα αυτό υφίσταται επειδή τα ψηφιακά περιουσιακά στοιχεία είναι ουσιαστικά ψηφιακά αρχεία οπότε κάποιος κακόβουλος χρήστης θα μπορούσε να δημιουργήσει αντίγραφα. Σε ένα αποκεντρωμένο σύστημα ψηφιακών νομισμάτων όπου δεν υπάρχει κεντρική αρχή για την επικύρωση των συναλλαγών, η αποτροπή της διπλής δαπάνης καθίσταται ζωτικής σημασίας. Οι αλγόριθμοι συναίνεσης διαδραματίζουν βασικό ρόλο στην αντιμετώπιση αυτού του προβλήματος, εξασφαλίζοντας τη συμφωνία μεταξύ των συμμετεχόντων στο δίκτυο σχετικά με την εγκυρότητα και τη σειρά των συναλλαγών. Ακολουθώντας έναν αλγόριθμο συναίνεσης, το δίκτυο μπορεί να καταλήξει σε συναίνεση σχετικά με το σωστό ιστορικό των συναλλαγών. Αυτό αποτρέπει την απάτη, καθιστώντας υπολογιστικά δύσκολο ή οικονομικά ανέφικτο για έναν επιτιθέμενο να χειραγωγήσει το ιστορικό των συναλλαγών και να ξοδέψει τα ίδια περιουσιακά στοιχεία πολλές

φορές. Επιπλέον, οι αλγόριθμοι συναίνεσης συμβάλλουν στο αμετάβλητο του blockchain, διασφαλίζοντας ότι μόλις μια συναλλαγή προστεθεί στην αλυσίδα, γίνεται μόνιμο και αμετάβλητο μέρος του ψηφιακού βιβλίου. Αυτή η αμεταβλητότητα παρέχει διαφάνεια, καθώς καθίσταται πρακτικά αδύνατο να τροποποιηθούν ή να αλλοιωθούν παρελθούσες συναλλαγές χωρίς τη συναίνεση της πλειοψηφίας των συμμετεχόντων στο δίκτυο.

Υπάρχουν διάφορες κατηγορίες αλγορίθμων συναίνεσης, η καθεμία με τα δικά της χαρακτηριστικά και συμβιβασμούς:

- **Proof of Work** Ο PoW είναι ο πιο γνωστός αλγόριθμος συναίνεσης, που χρησιμοποιείται σε κρυπτονομίσματα όπως το Bitcoin και το Ethereum (στην αρχική του έκδοση). Οι εξορύκτες ανταγωνίζονται για την επίλυση πολύπλοκων μαθηματικών γρίφων και ο πρώτος που θα βρει τη λύση ανταμείβεται και προσθέτει ένα νέο μπλοκ στο blockchain. Το PoW είναι γνωστό για την ασφάλειά του, αλλά απαιτεί σημαντική υπολογιστική ισχύ και κατανάλωση ενέργειας.

- **Proof of Stake** Το PoS είναι μια εναλλακτική λύση του PoW που επιλέγει επικυρωτές για τη δημιουργία και επικύρωση block με βάση το ποσό κρυπτονομίσματος που κατέχουν και είναι πρόθυμοι να ¨ποντάρουν' ως εγγύηση. Αυτός ο αλγόριθμος συναίνεσης μειώνει την κατανάλωση ενέργειας σε σύγκριση με τον PoW και επιτρέπει ταχύτερους χρόνους επιβεβαίωσης μπλοκ. Το Ethereum μεταβαίνει από το PoW στο PoS με την αναβάθμιση του Ethereum 2.0.

- **Delegated Proof of Stake** Το DPoS εισάγει ένα σύστημα βασισμένο στην ψηφοφορία, όπου οι κάτοχοι token εκλέγουν έναν περιορισμένο αριθμό αντιπροσώπων για την επικύρωση συναλλαγών και την παραγωγή block για λογαριασμό τους. Το DPoS βελτιώνει την επεκτασιμότητα μειώνοντας τον αριθμό των επικυρωτών, αλλά εισάγει κάποιο επίπεδο συγκεντρωτισμού.

- **Byzantine Fault Tolerance** Οι μηχανισμοί συναίνεσης BFT έχουν σχεδιαστεί για να εξασφαλίζουν συμφωνία μεταξύ των κόμβων σε ένα κατανεμημένο σύστημα, ακόμη και με την παρουσία κακόβουλων ή ελαττωματικών στοιχείων. Οι αλγόριθμοι BFT περιλαμβάνουν συνήθως πολλαπλούς γύρους επικοινωνίας, ψηφοφορίας και επαλήθευσης μεταξύ των κόμβων για την επίτευξη συναίνεσης σχετικά με την κατάσταση του συστήματος και είναι ιδανικοί για ιδιωτικά δίκτυα με ένα σχετικά αξιόπιστο σύνολο κόμβων.

- **Proof of Authority** Η PoA βασίζεται σε ένα σύνολο εγκεκριμένων επικυρωτών, που συνήθως αναγνωρίζονται από τα δημόσια κλειδιά ή τις ταυτότητές τους, οι οποίοι εναλλάσσονται προτείνοντας και επικυρώνοντας block. Το PoA είναι κατάλληλο για private και consortium blockchains, όπου η εμπιστοσύνη εγκαθιδρύεται μεταξύ των συμμετεχόντων.

Αυτά είναι μερικά μόνο παραδείγματα από τους πολλούς διαθέσιμους αλγορίθμους συναίνεσης. Κάθε αλγόριθμος συναίνεσης έχει τα δικά του πλεονεκτήματα, μειονεκτήματα και περιπτώσεις χρήσης. Η επιλογή του αλγορίθμου συναίνεσης εξαρτάται από παράγοντες όπως οι στόχοι του δικτύου, οι απαιτήσεις ασφαλείας, οι ανάγκες κλιμάκωσης και το επιθυμητό επίπεδο αποκέντρωσης. Στα πλαίσια της συγκεκριμένης διπλωματικής θα ασχοληθουμε με τους PoW και

PoS αλγορίθμους σε ένα περιβάλλον ιδιωτικής αλυσίδας Ethereum αλλά και με μία υλοποίηση και ένταξη ενός BFT αλγορίθμου συναίνεσης σε περιβάλλον ιδιωτικής αλυσίδας Ethereum.

### 2.2.1 Proof of Work του Ethereum

Ο Ethash είναι ο αλγόριθμος Proof of Work που χρησιμοποιείται από το Ethereum. Αποτελεί μια τροποποιημένη έκδοση του αλγορίθμου Dagger-Hashimoto και σχεδιάστηκε ειδικά για να είναι memory-hard, πράγμα που σημαίνει ότι η υπολογιστική εργασία που απαιτείται για τη διαδικασία του mining εξαρτάται σε μεγάλο βαθμό από την ποσότητα της μνήμης που είναι διαθέσιμη σε μια συσκευή εξόρυξης. Αυτή η σχεδιαστική επιλογή αποσκοπεί στην επίτευξη μιας πιο ισότιμης διαδικασίας mining, όπου οι miners με εξειδικευμένο υλικό εξόρυξης (ASICs) έχουν μικρότερο πλεονέκτημα έναντι των miners που χρησιμοποιούν υλικό καταναλωτικής ποιότητας, όπως GPUs. Απαιτώντας ένα σημαντικό ποσό μνήμης στη διαδικασία του mining, το Ethash καθιστά πιο δύσκολο για τις ASIC συσκευές, οι οποίες δίνουν προτεραιότητα στην υπολογιστική απόδοση, να αποκτήσουν σημαντικό πλεονέκτημα. Αυτό προωθεί ένα πιο αποκεντρωμένο οικοσύστημα εξόρυξης, όπου οι miners που χρησιμοποιούν GPU καταναλωτικής ποιότητας έχουν μια δίκαιη ευκαιρία να συμμετέχουν στη διαδικασία του mining. Νομοτελειακά, η ιδιότητα memory-hard του Ethash ενισχύει τη δικαιοσύνη της διαδικασίας του mining, διασφαλίζοντας ότι ένα ευρύτερο φάσμα ανθρώπων μπορεί να συμμετάσχει στο δίκτυο του Ethereum.

Ο αλγόριθμος Ethash περιλαμβάνει τα παρακάτω βήματα:

1. **Header Generation-Δημιουργία επικεφαλίδας**

   Για να εξορύξουν ένα νέο block, οι εξορύκτες ξεκινούν με τη δημιουργία μιας επικεφαλίδας block, η οποία περιλαμβάνει τον αριθμό block, τη χρονοσφραγίδα, το nonce και τα metadata. Αυτή η επικεφαλίδα χρησιμεύει ως είσοδος για τη διαδικασία εξόρυξης (mining).

2. **DAG Generation-Δημιουργία κατευθυνόμενου ακυκλικού γράφου**

   Στη συνέχεια, οι miners δημιουργούν ένα κατευθυνόμενο ακυκλικό γράφημα (DAG) χρησιμοποιώντας την επικεφαλίδα του μπλοκ και μια τιμή σπόρου (seed value). Το DAG είναι ένα μεγάλο σύνολο δεδομένων που αποθηκεύεται στη μνήμη της GPU. Δημιουργείται μία φορά για κάθε εποχή εξόρυξης και παραμένει σταθερό κατά τη διάρκεια της εν λόγω εποχής. Ο σκοπός του DAG είναι να παρέχει ένα μοναδικό μείγμα δεδομένων πάνω στο οποίο θα λειτουργήσει η διαδικασία του mining.

3. **Mining Process-Διαδικασία εξόρυξης**

   Η διαδικασία εξόρυξης περιλαμβάνει επανειλημμένο κατακερματισμό της επικεφαλίδας του block μαζί με διαφορετικά nonces μέχρι να βρεθεί ένας κατακερματισμός που να πληροί συγκεκριμένα κριτήρια. Οι miners χρησιμοποιούν τις δυνατότητες παράλληλης επεξεργασίας της GPU τους για τον κατακερματισμό πολλαπλών nonces ταυτόχρονα και αποτελεσματικά. Η απαιτούμενη υπολογιστική εργασία όπως αναφέρθηκε και πριν είναι memory-hard, πράγμα που σημαίνει ότι η διαδικασία κατακερματισμού εξαρτάται

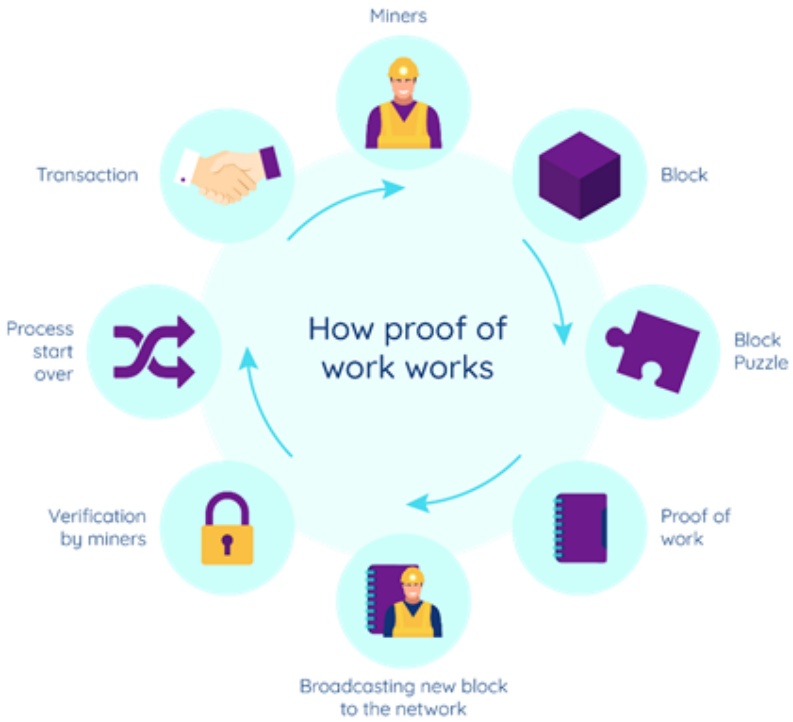
από την ποσότητα μνήμης που διαθέτει η συσκευή που χρησιμοποιείται για τη διαδικασία της εξόρυξης.

4. **Difficulty Adjustment-Ρύθμιση δυσκολίας**

   Ο αλγόριθμος προσαρμογής δυσκολίας του Ethereum εξασφαλίζει ότι ο μέσος χρόνος εξόρυξης ενός block παραμένει γύρω στα 15 δευτερόλεπτα. Εάν αυξηθεί η υπολογιστική ισχύς του δικτύου, με αποτέλεσμα την ταχύτερη εξόρυξη block, τότε η δυσκολία προσαρμόζεται ώστε να αυξηθεί η εργασία που απαιτείται για την εξόρυξη ενός block. Αντίθετως, εάν η υπολογιστική ισχύς του δικτύου μειωθεί, τότε η δυσκολία προσαρμόζεται προς τα κάτω για να εξασφαλιστει ξανά ο μεσος χρόνος εξόρυξης των 15 δευτερολέπτων διατηρηθεί ο χρόνος-στόχος μπλοκ.

5. **Proof of Work Verification**

   Μόλις ένας miner βρει ένα nonce που παράγει ένα hash κάτω από τη δυσκολία-στόχο, μπορεί να μεταδώσει το block στο δίκτυο ως απόδειξη εργασίας (proof-of-work). Στη συνέχεια,οι άλλοι συμμετέχοντες στο δίκτυο κόμβοι μπορούν να επαληθεύσουν την απόδειξη εργασίας, κατακερματίζοντας την επικεφαλίδα του block με το nonce και ελέγχοντας αν το προκύπτον ηαση ικανοποιεί τις απαιτήσεις. Εάν ναι, τότε το block θεωρείται έγκυρο και μπορεί να προστεθεί στο blockchain.




Εικόνα 2.3: *Proof of Work Consensus Mechanism*

### 2.2.2 Proof of Stake του Ethereum

Η μετάβαση από το Ethereum 1.0, δηλαδή το Ethereum με τη χρήση του αλγορίθμου συναίνεσης Proof of Work, στο Ethereum 2.0 έγινε με γνώμονα την επιτακτική ανάγκη να αντιμετωπιστούν βασικές προκλήσεις και να ξεκλειδωθούν νέες δυνατότητες για το μέλλον του δικτύου. Το Ετηερευμ 1.0 αντιμετώπισε περιορισμούς όσον αφορά την επεκτασιμότητα, την κατανάλωση ενέργειας καθώς και τα τέλη συναλλαγών. Για να ξεπεραστούν αυτά τα εμπόδια, ήρθε το Ethereum 2.0 το οποίο υιοθέτησε έναν εναλλακτικό αλγόριθμο συναίνεσης, αυτόν του Proof of Stake (PoS), συνοδευόμενου από σημαντικές αρχιτεκτονικές εξελίξεις. Η μετάβαση αυτή αποσκοπεί στην επίτευξη ανώτερης κλιμακωσιμότητας, ενεργειακής απόδοσης και ασφάλειας, ενώ παράλληλα ανοίγει το δρόμο για την ενσωμάτωση νέων χαρακτηριστικών και λειτουργιών, ικανοποιώντας τις εξελισσόμενες απαιτήσεις των αποκεντρωμένων εφαρμογών και του ευρύτερου οικοσυστήματος blockchain.

Ο αλγόριθμος PoS που χρησιμοποιείται στο Ethereum 2.0 είναι γνωστός ως Beacon Chain, και αποτελεί τη ραχοκοκαλιά του νέου μηχανισμού συναίνεσης. Παρακάτω θα παρατηρήσουμε τα βασικά χαρακτηριστικά του και διαφορές του από τον μηχανισμό συναίνεσης του Ethereum 1.0:

- **Validator Selection-Επιλογή επικυρωτή**

  Αντί για miners, το Ethereum 2.0 βασίζεται σε επικυρωτές, validators, για την ασφάλεια του δικτύου. Οι validators επιλέγονται με βάση το ποντάρισμά τους, δηλαδή το ποσό του κρυπτονομίσματος, στην προκειμένη περίπτωση ETH, που κατέχουν και κλειδώνουν ως εγγύηση. Όσο περισσότερα ETH κατέχει ένας validator, τόσο μεγαλύτερες είναι οι πιθανότητες να επιλεγεί για να προτείνει και να επικυρώσει ένα block.

- **Epochs and Slots-Εποχές και υποδοχές**

  Το Ethereum 2.0 λειτουργεί σε εποχές, οι οποίες είναι χρονικές περίοδοι που αποτελούνται από έναν σταθερό αριθμό υποδοχών. Κάθε υποδοχή έχει μια διάρκεια κατά την οποία ένας τυχαίος validator, ο proposer, μπορεί να προτείνει ένα block. Με τη διαδικασία αυτή στο τέλος κάθε εποχής δημιουργείται μια γραμμική αλυσίδα προτεινόμενων block σε κάθε εποχή.

- **Block Proposals and Attestations-Προτεινόμενα block και Βεβαιώσεις**

  Οι validators προτείνουν ένα block δημιουργώντας μια επικεφαλίδα που περιέχει ένα σύνολο συναλλαγών. Στη συνέχεια μεταδίδουν αυτές τις προτάσεις των blocks στο δίκτυο. Εκτός από τις προτάσεις όμως, οι validators παρέχουν επίσης βεβαιώσεις, οι οποίες στην πράξη είναι ψηφοφορίες σχετικά με την εγκυρότητα των προτεινόμενων block από άλλους validators. Οι βεβαιώσεις αυτές συμβάλλουν στη διαδικασία συναίνεσης και βοηθούν στην οριστικοποίηση των block.

- **Fork choice rule-Κανόνας επιλογής διακλάδωσης**

  Το Ethereum 2.0 χρησιμοποιεί έναν κανόνα επιλογής διακλάδωσης, γνωστό ως κανόνα ¨μακρύτερης αλυσίδας¨, για τον καθορισμό του blockchain. Οι validators σταθμίζουν ποια από τα προτεινόμενα block θεωρούν έγκυρα και χτίζουν πάνω σε αυτά. Η αλυσίδα

με το μεγαλύτερο συσσωρευμένο βάρος, δηλαδή με το μεγαλύτερο συνολικό ποντάρισμα γίνεται μέρος της αλυσίδας. Κανόνας επιλογής διακλάδωσης υπάρχει και στην περίπτωση του Ethereum 1.0, όπου ως κανονική αλυσίδα επιλέγεται η μακρύτερη.

- **Finality and Security-Οριστικότητα και Ασφάλεια**

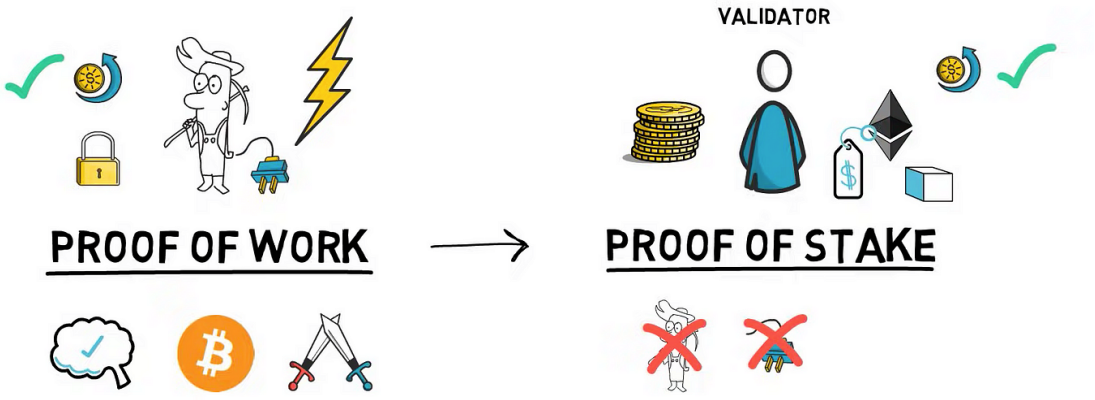
  Το Ethereum 2.0 εισάγει την έννοια της οριστικότητας, η οποία σημαίνει ότι μόλις ένα block προστεθεί στην αλυσίδα με επαρκή αριθμό πιστοποιήσεων, τότε οριστικοποιείται και είνια σημαντικά δύσκολο να ανατραπεί. Αυτό ενισχύει την ασφάλεια του δικτύου, καθώς τα οριστικοποιημένα blocks είναι απρόσβλητα από επιθέσεις, εξασφαλίζοντας το αμετάβλητο των συναλλαγών και των υπολοίπων. Στο Ethereum 1.0, το οποίο βασίζεται στον αλγόριθμο συναίνεσης proof of work, υπάρχει η πιθανότητα αναδιοργάνωσης της αλυσίδας ή διακλαδώσεων. Όταν πολλαπλοί miners βρίσκουν έγκυρα blocks σε παρόμοιες χρονικές στιγμές, μπορεί να προκύψουν προσωρινά forks, έως ότου ένας κλάδος γίνει μεγαλύτερος και γίνει αποδεκτός ως η κανονική αλυσίδα. Ωστόσο, αυτό σημαίνει ότι οι συναλλαγές που περιλαμβάνονται σε συντομότερους κλάδους δεν είναι δυνητικά τελικές και μπορούν να απορριφθούν εάν γίνει αποδεκτή μια μεγαλύτερη αλυσίδα με διαφορετικές συναλλαγές. Στο Ethereum 2.0, όταν συμβαίνει ένα fork, το σύστημα χρησιμοποιεί έναν μηχανισμό που ονομάζεται slashing για να εξασφαλίσει την οριστικότητα. Το slashing είναι μια ποινή που επιβάλλεται στους επικυρωτές που συμπεριφέρονται κακόβουλα ή παραβιάζουν τους κανόνες συναίνεσης. Με τον τρόπο αυτό όταν συμβαίνει μια διακλάδωση, οι επικυρωτές καλούνται να επιλέξουν σε ποιον κλάδο θα χτίσουν. Εάν ένας επικυρωτής συμμετέχει σε αντικρουόμενους κλάδους ή προσπαθεί να υποστηρίξει πολλαπλούς κλάδους, αυτό θεωρείται παραβίαση των κανόνων συναίνεσης. Έτσι οι επικυρωτές έχουν κίνητρο να επιλέγουν την αλυσίδα που έχει το μεγαλύτερο συσσωρευμένο βάρος (με βάση τις συμμετοχές των επικυρωτών) και να χτίζουν πάνω σε αυτήν, καθώς αυτό οδηγεί σε ανταμοιβές. Με την επιβολή ποινών slashing βοηθά το σύστημα να συγκλίνει σε μια ενιαία αλυσίδα, παρέχοντας οριστικότητα στα επιβεβαιωμένα block και τις συναλλαγές. Να τονιστεί ότι στο Ethereum 2.0 τα forks μπορεί να προκύψουν κατα τη διάρκεια του block proposal, πριν δηλαδή οριστικοποιηθεί το block που θα ενταχθεί μόνιμα και μη αναστρέψιμα στην αλυσίδα.

- **Penalties and Rewards-Ποινές και ανταμοιβές**

  Οι validators έχουν κίνητρο να συμπεριφέρονται ειλικρινά μέσω ενός συστήματος ποινών και ανταμοιβών. Εάν ένας επικυρωτής συμπεριφέρεται κακόβουλα ή βγαίνει εκτός σύνδεσης, μπορεί να τιμωρηθεί και να χάσει μέρος της παριουσίας του. Αντίθετα, οι validators που συμμετέχουν ενεργά και ακολουθούν τους κανόνες ανταμείβονται με επιπλέον ETH.

Βάση της περιγραφής που πραγματοποιήθηκε παραπάνω για μερικές από τις έννοιες-κλειδία του proof of stake αλγορίθμου που χρησιμοποιείτει πλέον από το Ethereum μπορούμε να παραθέσουμε τα βήματα που ακολουθούνται στην πράξη όταν πραγματοποιείται μια νέα συναλλαγή στο δίκτυο:

1. **Transaction Propagation-Διάδοση συναλλαγών**




Εικόνα 2.4: *PoW vs PoS*

Όταν ένας χρήστης ξεκινά μια νέα συναλλαγή στο δίκτυο του Ethereum 2.0, μεταδίδεται αρχικά στους κόμβους του δικτύου. Η συναλλαγή περιέχει πληροφορίες όπως ο αποστολέας, ο παραλήπτης, το ποσό και τυχόν πρόσθετα δεδομένα.

2. **Validator Inclusion-Ένταξη επικυρωτή**

   Οι validators λαμβάνουν τη συναλλαγή και τη συμπεριλαμβάνουν στη διαδικασία πρότασης block.

3. **Block Proposal-Προταση block**

   Κατά τη διάρκεια της σειράς τους, οι validators περιλαμβάνουν τη συναλλαγή σε ένα νέο block που προτείνουν. Το block περιέχει έναν κατάλογο συναλλαγών, συμπεριλαμβανομένης της πρόσφατα ληφθείσας, καθώς και άλλα μεταδεδομένα, όπως χρονοσφραγίδες και αναφορές στο προηγούμενο block.

4. **Block Verification-Επαλήθευση block**

   Το προτεινόμενο block μεταδίδεται στη συνέχεια σε άλλους validators για επαλήθευση. Οι επικυρωτές ελέγχουν την εγκυρότητα του block, διασφαλίζοντας ότι όλες οι περιεχόμενες συναλλαγές είναι σωστά διαμορφωμένες και πληρούν τα απαιτούμενα κριτήρια. Επικυρώνουν επίσης ότι ο αποστολέας διαθέτει επαρκή κεφάλαια για την ολοκλήρωση της συναλλαγής.

5. **Consensus Agreement-Συμφωνία συναίνεσης**

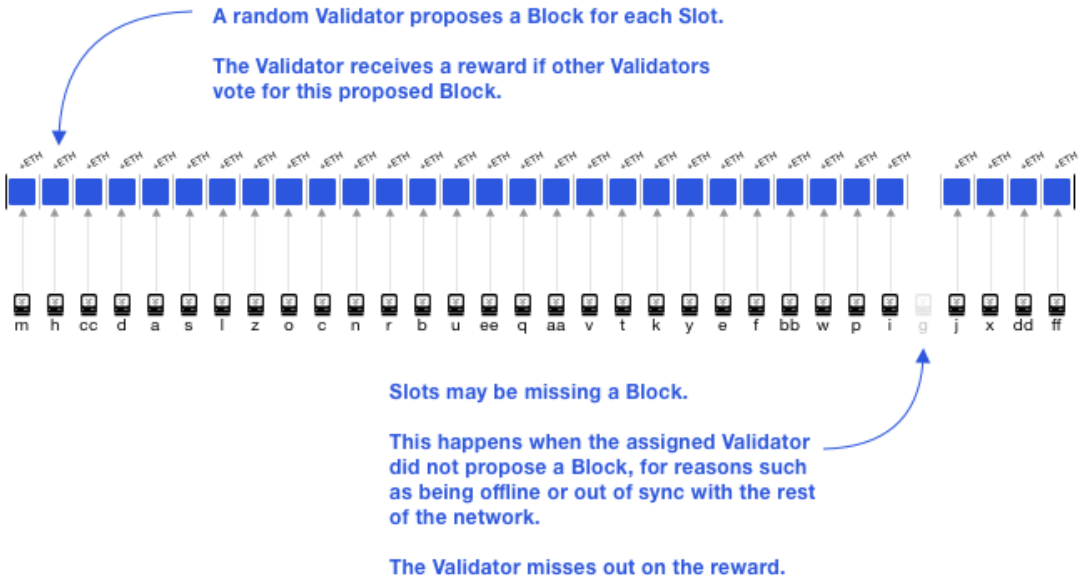
   Οι validators συμφωνούν συλλογικά για την εγκυρότητα του προτεινόμενου block μέσω του μηχανισμού συναίνεσης. Εκτελούν κρυπτογραφικές πράξεις και καταλήγουν σε συναίνεση σχετικά με την κατάσταση της αλυσίδας. Αυτή η συναίνεση επιτυγχάνεται λαμβάνοντας υπόψη τα μερίδια συμμετοχής των validators και τη δύναμη ψήφου τους στη διαδικασία συναίνεσης.

6. **Block Finalization-Οριστικοποίηση block**

Μόλις ένας επαρκής αριθμός validators συμφωνήσει για την εγκυρότητα του block, αυτό θεωρείται οριστικοποιημένο. Η οριστικοποίηση διασφαλίζει ότι το block προστίθεται στην αλυσίδα και γίνεται αμετάβλητο μέρος της ιστορίας του δικτύου. Τα οριστικοποιημένα block δεν μπορούν να τροποποιηθούν, παρέχοντας ασφάλεια και εμπιστοσύνη στο ιστορικό των συναλλαγών.

7. **Transaction Confirmation-Επιβεβαιωση συναλλαγης**

   Με την οριστικοποίηση του block, η συναλλαγή που περιλαμβάνεται σε αυτό θεωρείται επιβεβαιωμένη. Η επιβεβαίωση σημαίνει ότι η συναλλαγή θεωρείται πλέον αποδεκτή και μόνιμη στην αλυσίδα. Οι χρήστες μπορούν να βασίζονται στην επιβεβαίωση για να διασφαλίσουν την ολοκλήρωση και την εγκυρότητα των συναλλαγών τους.




Εικόνα 2.5: *Proof of Stake Consensus Mechanism*

Όπως αναφέρθηκε και παραπάνω, οι validators που συμμετέχουν ενεργά στην διαδικασία που περιγράφηκε παραπάνω και επικυρώνουν τις συναλλαγές ανταμείβονται με πρόσθετο κρυπτονόμισμαμα ενώ αντίθετα, οι validators που συμπεριφέρονται κακόβουλα ή προσπαθούν να συμπεριλάβουν άκυρες συναλλαγές αντιμετωπίζουν κυρώσεις, όπως η απώλεια μέρους των κεφαλαίων που έχουν ποντάρει.

### 2.2.3 NCCU Byzantine Fault Tolerance Consensus Algorithm-Αλγόριθμος Συναίνεσης Ανοχής Βυζαντινών Σφαλμάτων NCCU

Για να γίνει κατανοητή η θεωρία πίσω από τους αλγορίθμους συναίνεσης Byzantine Fault Tolerance (BFT), είναι απαραίτητο να εξεταστεί το πρόβλημα των Βυζαντινών Στρατηγών και η έννοια της ανοχής βυζαντινών σφαλμάτων.

Το Πρόβλημα των Βυζαντινών Στρατηγών είναι ένα θεωρητικό σενάριο που απεικονίζει τις προκλήσεις της επίτευξης συναίνεσης σε ένα κατανεμημένο δίκτυο, όταν ορισμένοι κόμβοι, που

αναφέρονται ως Βυζαντινοί κόμβοι, μπορεί να συμπεριφέρονται κακόβουλα ή να αποτυγχάνουν αυθαίρετα. Σε αυτό το πρόβλημα, μια ομάδα βυζαντινών στρατηγών περικυκλώνει μια εχθρική πόλη και πρέπει να συντονίσει την επίθεση ή την υποχώρησή της. Οι στρατηγοί μπορούν να επικοινωνούν μόνο μέσω μηνυμάτων, αλλά ορισμένοι από αυτούς μπορεί να είναι προδότες που θα στέλνουν αντικρουόμενα ή παραπλανητικά μηνύματα.

Ο στόχος είναι να βρεθεί ένας τρόπος ώστε οι πιστοί στρατηγοί να καταλήξουν σε μια κοινή απόφαση (επίθεση ή υποχώρηση) παρά την παρουσία βυζαντινών στρατηγών. Ωστόσο, λόγω της πιθανότητας προδοτικής συμπεριφοράς και των αποκλίσεων των μηνυμάτων, η επίτευξη συναίνεσης μεταξύ των πιστών στρατηγών καθίσταται δύσκολη.

Η ανεκτικότητα σε βυζαντινά σφάλματα, στο πλαίσιο των κατανεμημένων συστημάτων, αναφέρεται στην ικανότητα ενός δικτύου να επιτυγχάνει συναίνεση ακόμη και όταν ένα υποσύνολο κόμβων, μέχρι ένα ορισμένο όριο, συμπεριφέρεται αυθαίρετα ή κακόβουλα. Εξασφαλίζει δηλαδή ότι το σύστημα παραμένει λειτουργικό και ανθεκτικό, ακόμη και με την παρουσία βυζαντινών σφαλμάτων, όπως αποτυχίες κόμβων, καθυστερήσεις μηνυμάτων ή κακόβουλες ενέργειες.

Ο αλγόριθμος NCCU BFT είναι ένας αλγόριθμος συναίνεσης που έχει σχεδιαστεί για την αντιμετώπιση του προβλήματος των Βυζαντινών Στρατηγών και την επίτευξη ανοχής σε Βυζαντινά σφάλματα σε κατανεμημένα συστήματα. Ο NCCU BFT εξασφαλίζει ότι οι κόμβοι του δικτύου συμφωνούν σε μια ακολουθία αιτήσεων και καταλήγουν σε συναίνεση.

Αρχικά θα ορίσουμε μερικές έννοιες απαραίτητες για την κατανόηση της διαδικασίας και στη συνέχεια θα περιγράψουμε τον μηχανισμό συναίνεσης.

1. **Ορολογία**

   - **Block B:** Ένα block του δικτύου Ethereum.
   - **Ύψος H:** Το ύψος του μπλοκ που υποβάλλεται σε επεξεργασία.
   - **Γύρος R:** Ένας αποτελείται από τα τέσσερα βήματα για επιτευξη συναίνεσης.
   - **Επικυρωτές-Validators:** Κόμβοι που συμμετέχουν στη διαδικασία συναίνεσης.
   - **Προτείνων-Proposer:** Ένας validator που είναι υπεύθυνος για την πρόταση ενός μπλοκ σε έναν γύρο και ο οποίος επιλέγεται με τη βοήθεια Round-Robin scheduling

2. **Ψηφοι**

   - **Prevote:** Οι validators ψηφίζουν για ένα συγκεκριμένο block σε συγκεκριμένο ύψος και γύρο, υποδεικνύοντας την κατάστασή τους.
   - **PreCommitVote:** Παρόμοιο με το Prevote, αλλά συμβαίνει σε ύστερη φάση της διαδικασίας.

3. **Κλειδαρίες-Locksets**

   - **Prevote Lockset:** Μια συλλογή από prevotes για ένα συγκεκριμένο ύψος και γύρο.
   - **PreCommitVote Lockset:** Μια συλλογή από ψήφους precommit για ένα συγκεκριμένο ύψος και γύρο.

- **Απαρτία-Quorum:** Εάν περισσότερα από τα 2/3 των validators ψηφίσουν για το ίδιο μπλοκ μέσα σε ένα lockset, επιτυγχάνεται απαρτία.

4. **Προτάσεις-Proposals**

   - **Block Proposal:** Μια πρόταση που περιέχει ένα νέο block για ψηφοφορία σε συγκεκριμένο ύψος και γύρο.
   - **Voting Instruction:** Μια πρόταση που σηματοδοτεί την κατάσταση απαρτίας, όπου ένας proposer έχει λάβει πάνω από το 1/3 των prevotes για ένα μπλοκ προηγούμενου γύρου.

5. **Βήματα μηχανισμού συναίνεσης**

   - **Βήμα 1: Propose**

     (α′) Ελεγχουμε αν ο validator είναι ο proposer. Εάν όχι, μεταβαίνουμε στο βήμα 2.

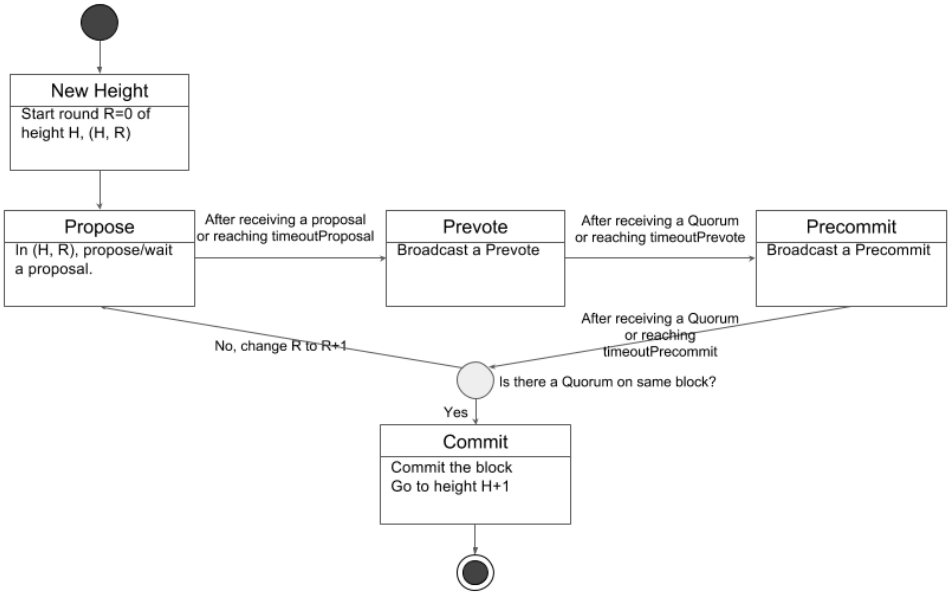
     (β′) Εάν υπάρχει απαρτία για ένα block σε PreVote Lockset από τον προηγούμενο γύρο, προτείνουμε ένα VotingInstruction. Διαφορετικά, δημιουργείται ένα νέο block και γίνεται ένα BlockProposal.

     (γ′) Πηγαίνουμε στο βήμα 3

   - **Βήμα 2: Prevote** Κάθε validator θα πρέπει να μεταδίδει ένα Prevote για ένα block με βάση προηγούμενες ενέργειες, όπως η προηγούμενη ψήφος του για το precommit, η λήψη μιας VotingInstruction ή μιας BlockProposal. Στη συνέχει μεταβαίνει στο βήμα 3.
   - **Βήμα 3: Precommit** Οι validators μεταδίδουν μια ψήφο precommit με βάση τις συλλεγμένες prevotes. Εάν υπάρχει απαρτία για ένα block στο prevote lockset, αποστέλλεται ψήφος precommit για το block. Εάν δεν υπάρχει απαρτία, περιμένουν για περισσότερες ψηφους prevote έως ότου επιτευχθεί ένα χρονικό όριο. Εάν και πάλι δεν υπάρχει απαρτία, αποστέλλεται μια ψήφος precommit χωρίς κάποιο block. Στη συνέχεια μετάβαση στο βήμα 4.
   - **Βήμα 4: Commit** Οι validators ελέγχουν το PrecommitVote Lockset για να καθορίσουν αν μπορούν να προχωρήσουν σε commit. Εάν υπάρχει απαρτία για ένα block στο σύνολο των PrecommitVote Lockset, το block δεσμεύεται και προχωράμε στο επόμενο ύψος. Εάν δεν υπάρχει απαρτία, περιμένουμε για περισσότερες PrecommitVotes μέχρι να επιτευχθεί ένα χρονικό όριο. Εάν και πάλι δεν υπάρχει απαρτία, προχωράμε στον επόμενο γύρο.

6. **Επιλογή Proposer** Για την επιλογή proposer σε κάθε γύρο χρησιμοποιείται η μέθοδος Round-Robin. Προκειται για μια απλή μέθοδο που χρησιμοποιείται για την εκ περιτροπής κατανομή καθηκόντων ή αρμοδιοτήτων σε μια ομάδα συμμετεχόντων. Ακολουθεί μια απλή εξήγηση του τρόπου λειτουργίας της μεθόδου.

   Έστω ότι έχουμε ε μια ομάδα ατόμων που πρέπει να εκτελούν εναλλάξ μια συγκεκριμένη εργασία.




Εικόνα 2.6: *States Diagram of NCCU BFT*

(α′) Σε κάθε άτομο της ομάδας ανατίθεται ένας αριθμός ή μια θέση.

(β′) Η εργασία ανατίθεται αρχικά στο άτομο με τον μικρότερο αριθμό που του έχει ανατεθεί.

(γ′) Το άτομο εκτελεί την εργασία για ένα συγκεκριμένο χρονικό διάστημα ή διάρκεια.

(δ′) Μόλις ολοκληρωθεί το χρονικό διάστημα, η εργασία περνάει στον επόμενο συμμετέχοντα στην ακολουθία.

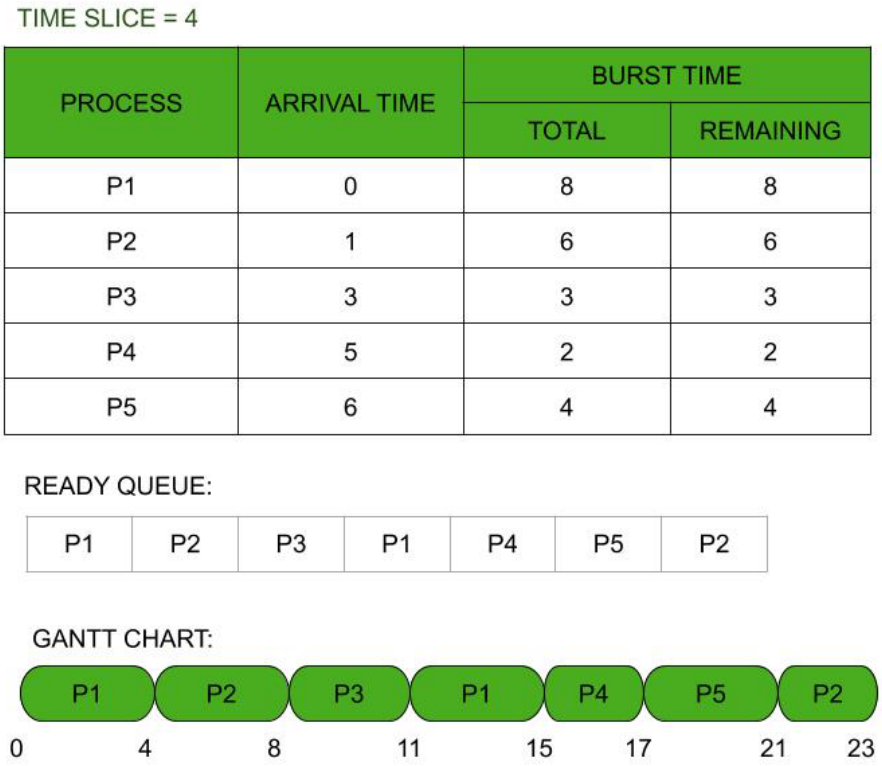
(ε′) Η εναλλαγή συνεχίζεται, με κάθε άτομο να παίρνει τη σειρά του με διαδοχική σειρά.

(ϛ′) Όταν όλοι έχουν κάνει τη σειρά τους, ο κύκλος ξεκινά πάλι από την αρχή, επιστρέφοντας στο πρώτο άτομο.

(ζ′) Αυτή η εναλλαγή επαναλαμβάνεται επ΄ αόριστον όσο η συνθήκη το απαιτεί.

Στην ουσία, η μέθοδος ρουνδ-ροβιν διασφαλίζει ότι κάθε συμμετέχων έχει ίσες ευκαιρίες να εκτελέσει την εργασία και η σειρά ανάθεσης παραμένει σταθερή σε όλη τη διάρκεια της εναλλαγής. Παρέχει μια δίκαιη και ισορροπημένη κατανομή του φόρτου εργασίας ή των ευθυνών μεταξύ των μελών της ομάδας.

Χρησιμοποιώντας μια σειρά από ανταλλαγές μηνυμάτων και επαληθεύσεις, ο NCCU BFT διασφαλίζει ότι όλοι οι ειλικρινείς κόμβοι συμφωνούν σχετικά με την αλληλουχία των αιτημάτων και την εκτέλεσή τους. Ο αλγόριθμος εγγυάται την ασφάλεια, δηλαδή ότι όλοι οι σωστοί κόμβοι συμφωνούν στην ίδια σειρά αιτήσεων, εφόσον ο αριθμός των βυζαντινών κόμβων δεν υπερβαίνει ένα ορισμένο όριο.Ο NCCU BFT παρέχει ανοχή σε σφάλματα Βψζαντινε, επιτρέποντας στο σύστημα να αντέξει την αυθαίρετη συμπεριφορά ή την αποτυχία έως και του ενός τρίτου του συνόλου των κόμβων. Επιτυγχάνει τη συναίνεση μέσω μιας διαδικασίας ψηφοφορίας πολλαπλών γύρων, όπου οι κόμβοι ανταλλάσσουν μηνύματα, επαληθεύουν τις απαντήσεις των άλλων και αποφασίζουν συλλογικά για τη σειρά των λειτουργιών.




Εικόνα 2.7: *Παράδειγμα Round-Robin Scheduling*

## 2.3 Blockbench

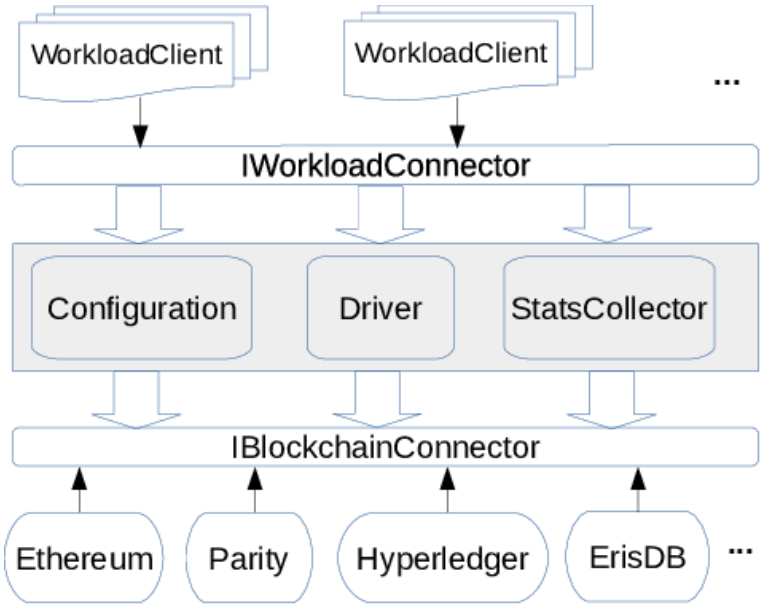
### 2.3.1 Εισαγωή στο Blockbench

Η τεχνολογία blockchain έχει κερδίσει σημαντική προσοχή τα τελευταία χρόνια, με τις ιδιωτικές αλυσίδες να αναδεικνύονται σε δημοφιλή επιλογή για επιχειρήσεις και οργανισμούς που επιθυμούν να αξιοποιήσουν τα οφέλη των συστημάτων κατανεμημένων βιβλίων. Ωστόσο, η αξιολόγηση και η συγκριτική αξιολόγηση των επιδόσεων των ιδιωτικών αυτών αλυσίδων μπορεί να είναι ένα δύσκολο έργο. Σε αυτό το κεφάλαιο, θα εξερευνήσουμε το Blockbench, ένα ισχυρό πλαίσιο που έχει σχεδιαστεί ειδικά για την ανάλυση ιδιωτικών αλυσίδων. Το Blockbench παρέχει στους ερευνητές, τους προγραμματιστές και τους οργανισμούς τα απαραίτητα εργαλεία για την αξιολόγηση των επιδόσεων, της επεκτασιμότητας αλλά και της ευρωστίας των ιδιωτικών πλατφορμών blockchain, διευκολύνωντας την ανάλυση και τη συγκριτική αξιολόγηση ιδιωτικών blockchain. Στόχος του είναι να αντιμετωπίσει την ανάγκη για τυποποιημένες τεχνικές αξιολόγησης επιδόσεων στον τομέα των blockchain. Παρέχοντας μια ευέλικτη και επεκτάσιμη λύση, επιτρέπει στους χρήστες να συγκρίνουν και να αξιολογούν τις επιδόσεις διαφορετικών πλατφορμών blockchain υπό διάφορους φόρτους εργασίας και σενάρια.

### 2.3.2 Σχεδιασμός και Αρχιτεκτονική του Blockbench

Ο σχεδιασμός και η αρχιτεκτονική του Blockbench βασίζονται στις θεμελιώδεις αρχές της αξιολόγησης επιδόσεων και της συγκριτικής αξιολόγησης. Το πλαίσιο είναι προσεκτικά δο-

μημένο ώστε να επιτρέπει στους χρήστες να διεξάγουν διεξοδικές αναλύσεις προσαρμοσμένες στις συγκεκριμένες απαιτήσεις τους. Η αρχιτεκτονική του Blockbench περιλαμβάνει διάφορα βασικά στοιχεία, καθένα από τα οποία εξυπηρετεί έναν συγκεκριμένο σκοπό και συνεργάζεται αρμονικά για να παρέχει ένα ολοκληρωμένο περιβάλλον αξιολόγησης για ιδιωτικά blockchain. Παρακάτω θα μελετήσουμε τα στοιχεία αυτά.




Εικόνα 2.8: *Λειτουργία blockbench*

1. **Δημιουργία φόρτου εργασίας**

   Το Blockbench περιλαμβάνει μια ευέλικτη και προσαρμόσιμη συνιστώσα δημιουργίας φόρτου εργασίας. Αυτό επιτρέπει στους χρήστες να προσομοιώνουν πραγματικό φόρτο εργασίας συναλλαγών, συμπεριλαμβανομένων διαφορετικών τύπων συναλλαγών, συχνοτήτων συναλλαγών και μοτίβων. Η ενότητα παραγωγής φόρτου εργασίας παράγει ρεαλιστικά φορτία εργασίας για τον έλεγχο καταπόνησης του συστήματος blockchain και την αξιολόγηση της απόδοσής του υπό διάφορα σενάρια.

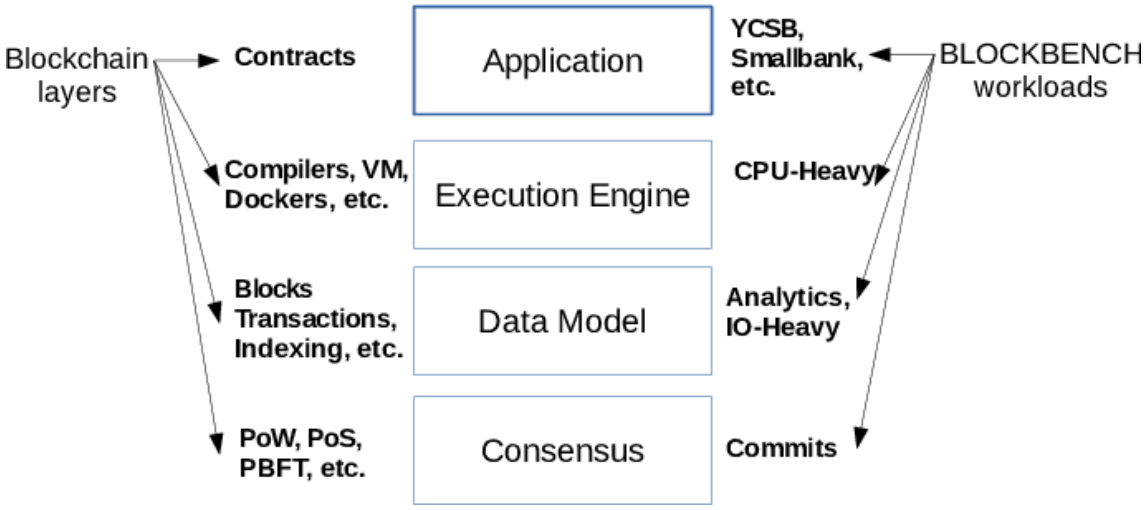
2. **Μοντελοποίηση blockchain**

   Το Blockbench παρέχει ένα ολοκληρωμένο πλαίσιο για τη μοντελοποίηση διαφόρων πτυχών του συστήματος blockchain. Οι χρήστες μπορούν να ορίσουν και να διαμορφώσουν τα χαρακτηριστικά του δικτύου blockchain, όπως ο αριθμός των κόμβων, οι αλγόριθμοι συναίνεσης κλπ. Αυτή η δυνατότητα μοντελοποίησης επιτρέπει στους χρήστες να αναπαραστήσουν με ακρίβεια τη συγκεκριμένη ρύθμιση blockchain και να αξιολογήσουν την απόδοσή της με βάση ρεαλιστικές παραμέτρους.

3. **Συλλογή μετρήσεων επιδόσεων**

   Κατά τη διάρκεια της εκτέλεσης των δοκιμών, το Blockbench συλλέγει ένα ευρύ φάσμα μετρήσεων επιδόσεων για να παρέχει λεπτομερείς πληροφορίες σχετικά με τη συμπεριφορά του συστήματος blockchain. Αυτές οι μετρικές περιλαμβάνουν την απόδοση

συναλλαγών, την καθυστέρηση, τη χρήση πόρων, τον χρόνο διάδοσης block και τον χρόνο μετάβασης συναίνεσης. Με τη συλλογή αυτών των μετρικών, οι χρήστες μπορούν να αποκτήσουν μια ολοκληρωμένη κατανόηση των χαρακτηριστικών απόδοσης του συστήματος και να εντοπίσουν πιθανές περιοχές για βελτιστοποίηση. Η συλλογή μετρήσεων του Blockbench μπορεί να αφορά οποιοδήποτε από τα διαφορετικά επίπεδα ενός ιδιωτικου blockchain οπως φαίνεται και στο παρακάτω διαγραμμα.




Εικόνα 2.9: *Επίπεδα blockchain και οι αντίστοιχοι φόρτοι εργασίας στο blockbench*

Παρακάτω θα πραγματοποιηθεί μια μικρή ανάλυση σχετικά με το τι πραγματεύεται σε κάθε επίπεδο του blockchain.

- **Consensus Layer**
  Ο ρόλος του επιπέδου συναίνεσης είναι να οδηγήσει ολους τους κόμβους του δικτύου στο να συμφωνήσουν σχετικά με το περιεχόμενο της αλυσίδας. μπλοκ.
- **Data Model Layer**
  Αναφέρεται στο επίπεδο εντός του blockchain που είναι υπεύθυνο για τον ορισμό και τη διαχείριση της δομής και της μορφής των δεδομένων που αποθηκεύονται στο blockchain.
- **Execution Layer**
  Αναφέρεται στο επίπεδο εντός του blockchain που χειρίζεται την εκτέλεση έξυπνων συμβολαίων ή αποκεντρωμένων εφαρμογών.
- **Application Layer**
  Αναφέρεται στο επίπεδο εντός του blockchain όπου οι προγραμματιστές μπορούν να δημιουργήσουν και να αναπτύξουν τις συγκεκριμένες εφαρμογές ή περιπτώσεις χρήσης τους πάνω στην υποκείμενη υποδομή blockchain.

Εμείς όπως έχουμε αναφέρει ξανά στα πλαίσια της διπλωματικης αυτής θα διεξάγουμε πειραματισμούς στο Consensus Layer των ιδιωτικών δικτύων που θα στήσουμε καθώς αυτό που μας ενδιάφέρει είναι η απόδοση που έχουν διάφορετικοί αλγόριθμοι συναίνεσης σε ιδιωτικά δίκτυα Ethereum blockchain

Συνδυάζοντας αυτά τα στοιχεία, το Blockbench προσφέρει ένα στιβαρό και προσαρμόσιμο πλαίσιο για την αξιολόγηση των επιδόσεων των ιδιωτικών blockchain. Ο σχεδιασμός και η αρχιτεκτονική του Blockbench διευκολύνουν την ακριβή μοντελοποίηση, τη δημιουργία ρεαλιστικού φόρτου εργασίας, την ολοκληρωμένη συλλογή μετρήσεων επιδόσεων και την αποτελεσματική ανάλυση των αποτελεσμάτων.

### 2.3.3 Μετρικές αξιολόγησης

Κατά την ανάλυση των επιδόσεων ενός blockchain χρησιμοποιώντας διαφορετικά configuration, το Blockbench χρησιμοποιεί στατιστικά στοιχεία εξόδου για να αξιολογήσει το σύστημα με βάση τις παρακάτω μετρήσεις επιδόσεων.

- **Throughput**

  Το throughput μετράται με τον υπολογισμό του αριθμού των επιτυχημένων συναλλαγών που υποβάλλονται σε επεξεργασία ανά δευτερόλεπτο. Με τη διαμόρφωση του φόρτου εργασίας με πολλούς clients αλλά και νήματα ανά client όπου είναι απαραίτητο, ο στόχος είναι να μεγιστοποιηθεί η απόδοση του blockchain και να κορεστούν οι δυνατότητες επεξεργασίας της.

- **Latency-Καθυστέρηση**

  Το latency αναφέρεται στον χρόνο απόκρισης ανά συναλλαγή, υποδεικνύοντας πόσος χρόνος απαιτείται για την επεξεργασία μιας συναλλαγής. Στο Blockbench, ο driver υλοποιεί μια προσέγγιση μπλοκαρισμένης συναλλαγής, όπου περιμένει να ολοκληρωθεί μια συναλλαγή πριν ξεκινήσει η επόμενη. Αυτό επιτρέπει τη μέτρηση της καθυστέρησης με ελεγχόμενο τρόπο.

- **Scalability-Επεκτασιμότητα**

  Η επεκτασιμότητα αξιολογείται με την παρατήρηση των αλλαγών στην απόδοση και την καθυστέρηση καθώς αυξάνεται ο αριθμός των κόμβων και των ταυτόχρονων φόρτων εργασίας. Με τη σταδιακή κλιμάκωση του δικτύου blockchain, το Blockbench μετρά τον τρόπο με τον οποίο επηρεάζεται η απόδοση του συστήματος και παρέχει πληροφορίες σχετικά με την ικανότητά του να διαχειρίζεται μεγαλύτερους φόρτους εργασίας και αυξανόμενα μεγέθη δικτύου.

- **Fault Tolerance-Ανοχή Σφαλμάτων**

  Η ανοχή σφαλμάτων αξιολογεί τον τρόπο με τον οποίο το σύστημα αποδίδει παρουσία αποτυχιών κόμβων. Ενώ τα συστήματα blockchain είναι εγγενώς ανθεκτικά σε βυζαντινές αποτυχίες, δεν είναι δυνατόν να προσομοιωθούν όλες οι πιθανές βυζαντινές συμπεριφορές. Στο Blockbench μπορούν να προσομοιωθούν τρεις τρόποι αποτυχίας: **αποτυχία κατάρρευσης**, όπου ένας κόμβος απλώς σταματά να λειτουργεί, **καθυστέρηση δικτύου**, όπου εισάγονται αυθαίρετες καθυστερήσεις στις ανταλλαγές μηνυμάτων, και **τυχαία απόκριση**, όπου τα μηνύματα που ανταλλάσσονται μεταξύ των κόμβων αλλοιώνονται σκόπιμα.

### 2.3.4 Δοκιμές και αξιολόγηση δικτύων Blockchain

Χρησιμοποιώντας τις μετρικές αξιολόγησης που αναφέρθηκαν παραπάνω, το Blockbench παρέχει μια ολοκληρωμένη ανάλυση των επιδόσεων των blockchain, αξιολογώντας την απόδοση, την καθυστέρηση, την επεκτασιμότητα και την ανοχή σε σφάλματα υπό διάφορες συνθήκες.

Όπως φαίνεται και από την παραπάνω ανάλυση το Blockbench παρέχει ένα ισχυρό πλαίσιο για την αξιολόγηση αλγορίθμων συναίνεσης, επιτρέποντας τη σε βάθος ανάλυση των επιδόσεων και της συμπεριφοράς τους στο επίπεδο συναίνεσης μιας αλυσίδας. Το στρώμα συναίνεσης (consensus layer) αναφέρεται στο συστατικό στοιχείο ενός συστήματος blockchain που είναι υπεύθυνο για την επίτευξη συμφωνίας μεταξύ των συμμετεχόντων στο δίκτυο σχετικά με τη σειρά και την εγκυρότητα των συναλλαγών. Είναι το κρίσιμο τμήμα της αλυσίδας όπου υλοποιούνται αλγόριθμοι συναίνεσης. Εν όψει της διπλωματικής αυτής, θα αξιοποιήθεί το Blockbench για την αξιολογήση των αλγορίθμων συναίνεσης που αναλύθηκαν παραπάνω σε ένα ιδιωτικό Ethereum blockchain. Συγκεκριμένα, θα χρησιμοποιηθεί το έξυπνο συμβόλαιο doNothing που παρέχεται από το Blockbench , το οποίο λειτουργεί ως φορέας για την εκτέλεση συναλλαγών χωρίς παρενέργειες, αφού αποτελεί απλά ένα συμβόλαιο εντος του οποίου δεν πραγματοποιείται καμία ενέργεια. Με την ανάπτυξη αυτού του συμβολαίου εντός του στρώματος συναίνεσης της αλυσίδας, θα αξιολογηθούν οι μετρικές που αναφέρθηκαν παραπάνω.

# Μέρος II

# Πρακτικό Μέρος

# Κεφάλαιο 3

# Υλοποίηση

Η ενότητα αυτή αποσκοπεί στην παροχή μιας ολοκληρωμένης ανάλυσης των πρακτικών πτυχών εφαρμογής της προτεινόμενης έρευνας. Η έμφαση θα δοθεί στη διερεύνηση και αξιολόγηση της απόδοσης και της λειτουργικότητας διαφορετικών διαμορφώσεων βλοκχεηαιν με τη χρήση του Geth Client, του Prysm Client και του framework Blockbench.

Η ανάλυση θα ξεκινήσει με την εξέταση του Geth Client, μιας δημοφιλούς υλοποίησης του Ethereum σε Go language, και των δυνατοτήτων του στη δημιουργία ενός ιδιωτικού δικτύου. Συγκεκριμένα, θα διερευνήσουμε τρεις διαφορετικές διαμορφώσεις: Geth με Proof of Work, Geth με Practical Byzantine Fault Tolerance και Geth με Prysm και Proof of Stake. Κάθε διαμόρφωση παρουσιάζει μοναδικά χαρακτηριστικά και εκτιμήσεις όσον αφορά τους μηχανισμούς συναίνεσης, την επεκτασιμότητα και την απόδοση τους.

Επιπλέον, το κεφάλαιο θα εμβαθύνει στην υλοποίηση και τη χρήση του Blockbench. Θα διερευνήσουμε τη διαδικασία εγκατάστασης και ανάπτυξης του Blockbench, παρέχοντας βήμα προς βήμα οδηγίες για τον τρόπο διαμόρφωσης και αξιοποίησης του για τη δοκιμή και την αξιολόγηση διαφορετικών διαμορφώσεων ιδιωτικών δικτύων.

Μέσω αυτής της σε βάθος διερεύνησης και ανάλυσης το παρόν κεφάλαιο έχει ως στόχο να παρέχει μια πρακτική κατανόηση του τρόπου δημιουργίας και αξιολόγησης διαφορετικών ιδιωτικών δικτύων. Με την εξέταση αυτών των υλοποιήσεων, μπορούμε να συγκεντρώσουμε πολύτιμες πληροφορίες σχετικά με τις επιδόσεις, την επεκτασιμότητα και τη λειτουργικότητα διαφόρων διαμορφώσεων blockchain, συμβάλλοντας στη βαθύτερη κατανόηση της τεχνολογίας ιδιωτικών blockchain και των πιθανών εφαρμογών της.

## 3.1 Geth Client

Ο Geth Client είναι μία από τις πιο ευρέως χρησιμοποιούμενες υλοποιήσεις client του Ethereum, προσφέροντας ένα ισχυρό και πλούσιο σε χαρακτηριστικά περιβάλλον για την αλληλεπίδραση με το blockchain του Ethereum. Παρέχει εκτεταμένη λειτουργικότητα για τη δημιουργία και τη διαχείριση ιδιωτικών δικτύων, επιτρέποντας στους προγραμματιστές και τους ερευνητές να πειραματιστούν με διαφορετικούς αλγόριθμους συναίνεσης και διαμορφώσεις δικτύων.

Αρχικά, το Geth υποστηρίζει τον παραδοσιακό proof of work αλγόριθμο συναίνεσης Ethash , ο οποίος βασίζεται σε υπολογιστικούς γρίφους για την ασφάλεια του δικτύου και την επικύρωση των συναλλαγών αλλά και του Clique ενός proof of authority αλγορίθμου συναίνε-

σης. Επιτρέπει στους χρήστες να διαμορφώνουν διάφορες παραμέτρους που σχετίζονται με το mining, όπως η δυσκολία, τα gas fees και οι ανταμοιβές. Χρησιμοποιώντας το Geth με proof of work, μπορεί κανείς να αποκτήσει γνώσεις σχετικά με τις επιδόσεις, την αποδοτικότητα καθώς και την επεκτασιμότητα ενός ιδιωτικού δικτύου Ethereum.

Επιπλέον, το Geth είναι συμβατό με το Prysm, μια υλοποίηση client του Ethereum 2.0 που ενσωματώνει τον μηχανισμό συναίνεσης proof of stake. Χρησιμοποιώντας το Geth με το Prysm, είναι δυνατή η δημιουργία ενός ιδιωτικού δικτύου που προσομοιώνει την αναβάθμιση σε Ethereum 2.0, παρέχοντας την ευκαιρία να μελετηθεί ο νέος αλγόριθμος συναίνεσης αλλά και ο αντίκτυπος της αλλαγής αυτής στην αποδοτικότητα του δικτύου.

Στη συνέχεια θα ακολουθήσει μια τεχνική αναλυση των τριών αλγορίθμων με τους οποίους θα πειραματιστούμε στα πλαίσια της διπλωματικής αυτής.

### 3.1.1 Geth με PoW

Για το στήσιμο του συγκεκριμένου περιβάλλοντος χρησιμοποιήθηκε κατεξοχήν το documentation του Geth client. Απαραίτητο λογισμικό για την υλοποίηση αυτή είναι η εγκατάσταση στο σύστημα της γλώσσας προγραμματισμού Go καθώς και της επίσημης υλοποίησης του Geth. Παρακάτω θα ακολουθήσει η περιγραφή για το στήσιμο του περιβάλλοντος αυτού με δύο κόμβους. Η επέκταση του δικτύου σε περισσότερους κόμβους πραγματοποιείται με επανάληψη των βημάτων.

- **Βήμα 1: Εγκατασταση του Geth και της Go** Επιβεβαίωση ότι η Go και το Geth είναι εγκατεστημένο στο συστημα. Αυτο μπορεί να γίνει με τη βοήθεια των εντολών:

```
go version
```

```
geth --help
```

- **Βήμα 2: Δημιουργία ενός αρχείου Genesis** Δημιουργούμε ένα αρχείο Genesis, ας πούμε genesis.json, που ορίζει την αρχική διαμόρφωση του ιδιωτικού δικτύου. Αυτό το αρχείο καθορίζει παραμέτρους όπως το αναγνωριστικό δικτύου, τη δυσκολία και τα gas fees. Παρακάτω ακολουθεί παραδειγμα ενός τυπικού Genesis αρχείου.

```
{
    "config": {
      "chainId": 12345,
      "homesteadBlock": 0,
      "eip150Block": 0,
      "eip155Block": 0,
      "eip158Block": 0,
      "byzantiumBlock": 0,
      "constantinopleBlock": 0,
      "petersburgBlock": 0,
      "istanbulBlock": 0,
      "berlinBlock": 0,
      "ethash": {}
    },
    "difficulty": "1",
```

```
    "gasLimit": "8000000",
    "alloc": {
      "7df9a875a174b3bc565e6424a0050ebc1b2d1d82": {"balance":"3000"},
      "f41c74c9ae680c1aa78f42e5647a62f353b7bdde": {"balance":"4000"}
    }
}
```

- **Βήμα 3: Αρχικοποιήση του πρώτου κόμβου** Χρησιμοποιούμε την ακόλουθη εντολή για να αρχικοποιήσουμε τον πρώτο κόμβο με το αρχείο Genesis που δημιουργήθηκε:

```
geth --datadir /path/to/datadir/node1 init /path/to/genesis.json
```

Αντικαταθιστούμε το /path/to/datadir/node1 με την τοποθεσία όπου θέλουμε να αποθηκεύσουμε τα δεδομένα blockchain για τον πρωτο κόμβο και το /path/to/genesis.json με την τοποθεσία του αρχείου Genesis.

- **Βήμα 4: Εκκίνηση του πρώτου κόμβου** Εκτελούμε την παρακάτω εντολή:

```
geth --datadir /path/to/datadir/node1 --networkid <network_id>
--port <port_number> --rpc --rpcport <rpc_port_number>
--rpcapi "eth,net,web3,personal,miner"
--allow-insecure-unlock console
```

Αντικαταθιστούμε το /path/to/datadir/node1 με την τοποθεσία όπου αποθηκεύσατε τα δεδομένα της αλυσίδας. Καθορίζουμε ένα μοναδικό <network_id> για το ιδιωτικό δίκτυο βασιζόμενοι στο Genesis αρχειο που έχουμε ορίζει. Επιλέγουμε ένα διαθέσιμο <port_number> και <rpc_port_number> για το δίκτυο και την επικοινωνία RPC αντίστοιχα. Η σημαία –rpcapi καθορίζει τα API που πρέπει να ενεργοποιηθούν και η σημαία –allow-insecure-unlock επιτρέπει το ξεκλείδωμα λογαριασμών μέσω της κονσόλας.

- **Βήμα 5: Δημιουργία λογαριασμού για τον πρώτο κόμβο** Αν θέλουμε να αλληλεπιδράσουμε με το δίκτυο χρησιμοποιώντας έναν λογαριασμό αρχικά χρησιμοποιούμε την κονσόλα javascript του Geth την οποία ανοίγουμε με την εντολή

```
geth attach /path/to/datadir/node1/geth.ipc
```

και στη συνέχεια δημιουργούμε τον λογαριασμό με χρήση της παρακάτω εντολής εντός της κονσόλας

```
personal.newAccount("<password>")
```

Φροντίζουμε να θυμόμαστε αυτόν τον κωδικό πρόσβασης, καθώς θα χρειαστεί για να ξεκλειδώσουμε το λογαριασμό.

- **Βήμα 6: Επαναληψη των βήματων 3 έως 5 για τον δεύτερο κόμβο** Επαναλαμβάνουμε τα βήματα 3 έως 5. Σε περίπτωση που ο δεύτερος κόμβος δημιουργείτε στον ίδιο server, βεβαιωνόμαστε να επιλέξουμε την καταλληλη τοποθεσία για την αποθήκευση δεδομένων σχετικό με τον κόμβο και βεβαιωνόμαστε οτι χρεισημοποιούμε διαφορετικα ports. Η διαδικασία αυτή θα δημιουργήσει και θα αρχικοποιήσει τον δεύτερο κόμβο και θα σας επιτρέψει να δημιουργήσουμε έναν λογαριασμό κα για αυτόν.

- **Βήμα 7: Σύνδεση των δύο κόμβων** Μέσα από την κονσόλα javascript του Geth του δεύτερου κόμβου, εκτελούμε την ακόλουθη εντολή για να τον συνδέσουμε με τον πρώτο κόμβο:

```
admin.addPeer("<enode_url_of_node1>")
```

Αντικαθιστούμε το <enode_url_of_node1> με το enode URL του πρώτου κόμβου. Μπορούμε να λάβουμε τη διεύθυνση αυτή εκτελώντας παρακάτω εντολή στην κονσόλα javascript του Geth του πρώτου κόμβου.

```
admin.nodeInfo.enode
```

- **Βήμα 8: Εκκίνηση διαδικασίας εξόρυξης** Εκκινούμε τη διαδικασία εξόρυξης και στους δυο κόμβους απλά εκτελώντας στις κονσόλες javascript του Geth και των δύο κόμβων την εντολή:

```
miner.start(1)
```

- **Βήμα 9: Επαλήθευση συνδεσιμότητας κόμβων** Βεβαιωνόμαστε ότι οι δύο κόμβοι είναι συνδεδεμένοι εκτελώντας την εντολή **admin.peers** στην κονσόλα javascript του Geth οποιουδήποτε κόμβου. Θα πρέπει να εμφανίζει πληροφορίες σχετικά με τον συνδεδεμένο ομότιμο.

- **Βήμα 10: Αλληλεπίδραση με το δίκτυο** Στο σημείο αυτό το ιδιωτικό μας δίκτυο είναι ετοιμο. Μπορούμε να αλληλεπιδράσουμε με αυτο, να δοκιμάσουμε να στείλουμε transactions, να δούμε τα επιβεβαιωμένα blocks και τα transactions που εμπεριέχουμε και γενικότερα να πειρματιστούμε.

Παρακάτω ακολουθούν φωτογραφίες από διάφορες φάσεις ενός λειτουργικού δικτύου.

### 3.1.2 Geth με NCCU BFT

Για το στήσιμο του συγκεκριμένου περιβάλλοντος χρησιμοποιήθηκε κατεξοχήν το documentation του public repository FastBFT_ethereum που αποτελεί την προσπάθεια υλοποίησης ενός Geth client με ενσωματομένο bft αλγόριθμο συναίσεσης. Απαραίτητο λογισμικό για την υλοποίηση αυτή είναι η εγκατάσταση στο σύστημα της γλώσσας προγραμματισμού Go καθώς και της υλοποίησης του Geth που βρίσκεται στο repository. Παρακάτω θα ακολουθήσει η περιγραφή για το στήσιμο του περιβάλλοντος αυτού με δύο κόμβους. Η επέκταση του δικτύου σε περισσότερους κόμβους πραγματοποιείται με επανάληψη των βημάτων.

- **Βήμα 1: Εγκατασταση του Geth και της Go** Επιβεβαίωση ότι η Go και το Geth είναι εγκατεστημένο στο συστημα. Αυτο μπορεί να γίνει με τη βοήθεια των εντολών:

```
go version
```

```
geth --help
```

(a) Network Initialization (b) Generating DAG

(c) Mining

Εικόνα 3.1: *Λειτουργικό δίκτυο Ethereum με μηχανισμό συναίνεσης Proof-of-Work*

- **Βήμα 2: Δημιουργία ενός αρχείου Genesis** Δημιουργούμε ένα αρχείο Genesis, ας πούμε genesis.json, που ορίζει την αρχική διαμόρφωση του ιδιωτικού δικτύου. Αυτό το αρχείο καθορίζει παραμέτρους όπως το αναγνωριστικό δικτύου, τον αλγόριθμο συναίνεσης που θα δημιουργηθε και τα gas fees. Παρακάτω ακολουθεί παραδειγμα ενός τυπικού Genesis αρχείου.

```
{
  "config": {
    "chainId": 2234,
    "homesteadBlock": 1,
    "eip150Block": 2,
    "eip150Hash": "0x0000000000000000000000000000
000000000000000000000000000000000000000",
    "eip155Block": 3,
    "eip158Block": 3,
    "ethash":{}
  },
  "nonce": "0x0",
  "timestamp": "0x592d25a6",
  "parentHash": "0x0000000000000000000000000000
000000000000000000000000000000000000000",
  "extraData": "0x0000000000000000000000000000
000000000000000000000000000000000000000",
  "gasLimit": "0x989680",
  "difficulty": "1",
  "mixHash": "0x0000000000000000000000000000
000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
```

```
  "alloc": {
    "0x123463a4B065722E99115D6c222f267d9cABb524": {
      "balance": "23370000000000000000"
    },
    "0x5A473896c4c65D606EBca3a9D2a5cD78fc046FB5": {
      "balance": "23370000000000000000"
    },
    "0x535f21d4328c2480D2b14C037A94FF63141a1d8D": {
      "balance": "23370000000000000000"
    },
    "0x9B4A5bcb65DD2A7f47E647761CA3E2e51408A362": {
      "balance": "23370000000000000000"
    }
  }
}
```

Στη συγκεκριμένη περίπτωση δεν ορίζουμε στο Genesis αρχειο μπρεί να φαινεται οτι ορίζεται ως αλγόριθμος συναίνεσης ο Ethash ωστόσο όπως θα δούμε και παρακάτω θα ενεργοποιήσουμε τον bft μηχανισμό με τη βοήθεια σημαίας. Σημαντικό είναι στο αρχείο που θα δημιουργησουμε να αναθέσουμε την τιμή '1' στο difficulty. Επιπλέον παρατηρούμε ότι στην περιπτωση αυτή είναι απαραίτητο να συμπληρωθεί το πεδίο **alloc** καθώς στην περίπτωση του bft για να πραγματοποιηθούν τα πρώτα transactions χρειάζεται ether για τα fees. Στην πρερίπτωση του proof of work κάτι τέτοιο δεν είναι απαραίτητο καθώς κατα τη διαδικασία του μινινγ περνάνε και κενά blocks τα οποία δίνουν rewards στους miners. Εννοείται πως και στην περίπτωση του Geth με proof of work είναι δυνατή η χρήση του πεδίου **alloc** για την κατανομή ether.

- **Βήμα 3: Αρχικοποίηση του πρώτου κόμβου και εισαγωγή λογαριασμού** Αρχικοποιούμε τον πρώτο κόμβο και στη συνέχεια να εισάγουμε το λογαριασμό που θέλουμε στον πρώτο κόμβο με τη βοήθεια ενός secret.json αρχείου που αποτελεί το private key του λογαριασμού μας. Αυτό συμβαίνει με τη βοήθεια των ακόλουθων εντολών.

```
geth --datadir /path/to/datadir/node1 init /path/to/genesis.json
```

```
geth --datadir /path/to/datadir/node1 import secret.json
--password="somepassword"
```

Αντικαταθιστούμε το /path/to/datadir/node1 με την τοποθεσία όπου θέλουμε να αποθηκεύσουμε τα δεδομένα blockchain για τον πρωτο κόμβο και το /path/to/genesis.json με την τοποθεσία του αρχείου Genesis.

- **Βήμα 4: Δημιουργία του αρχείου static-nodes.json** Δημιουργούμε ένα αρχείο static-nodes.json στην τοποθεσία όπου είναι αποθηκευμένα τα blockchain για τον πρωτο κόμβο. Αυτό το αρχείο θα περιέχει το enode URLS του δεύτερου κόμβου για τη δημιουργία συνδεσιμότητας μεταξύ των δύο κόμβων. Το περιεχόμενο του αρχείου static-nodes.json θα πρέπει να έχει ως εξής:

```
["<enode_url_of_node2>"]
```

Εννοτείται πως η αντιστοιχη διαδικασία για τη συνδεση των κόμβων μπορεί να χρησιμοποιηθεί και στην περίπτωση του proof of work ωστόσο αναφέρεται και ως μια εναλλακτικη μεθοδος για την επιτευξη συνδεσιμότητας των κόμβων. Με την υπαρξη του αρχείου αυτού κάθε φορά που επανακκινεί το δίκτυο επιτυγχάνει εκ νέου σύνδεση των κόμβων.

- **Βήμα 5: Εκκίνηση του πρώτου κόμβου** Εκτελούμε την παρακάτω εντολή:

```
geth --datadir /path/to/datadir/node1
--bft --num-validators <num_of_validators> --node-num <num_of_node>
--networkid <network_id> --port <port_number> --rpc
--rpcport <rpc_port_number> --unlock <num_of_account>
--password="password of account"
```

Αντικαταθιστούμε το /path/to/datadir/node1 με την τοποθεσία όπου αποθηκεύσατε τα δεδομένα της αλυσίδας. Καθορίζουμε ένα μοναδικό <network_id> για το ιδιωτικό δίκτυο βασιζόμενοι στο Genesis αρχειο που έχουμε ορίζει. Επιλέγουμε ένα διαθέσιμο <port_number> και <rpc_port_number> για το δίκτυο και την επικοινωνία RPC αντίστοιχα. Η σημαία –rpcapi καθορίζει τα API που πρέπει να ενεργοποιηθούν και η σημαία –υνλοςκ επιτρέπει το ξεκλείδωμα του λογαριασμού <num_of_account>. Η σημαια –bft είναι αυτή που ορίζει στο δίκτυο τον αλγοριθμο συναίνεσης bft, η –num-validators τον αριθμό των επικυρωτων που θα συμμετάσχουν στη διαδικασία του bft, ενώ η σημαία –node-num την ταυτότητα του κόμβου.

- **Βήμα 6: Επανάληψη των βημάτων 2 έως 5 για το δεύτερο κόμβο**

Επαναλαμβάνουμε τα βήματα 2 έως 5. Σε περίπτωση που ο δεύτερος κόμβος δημιουργείτε στον ίδιο server, βεβαιωνόμαστε να επιλέξουμε την καταλληλη τοποθεσία για την αποθήκευση δεδομένων σχετικό με τον κόμβο και βεβαιωνόμαστε οτι χρεισημοποιούμε διαφορετικα ports.

- **Βήμα 7: Επαλήθευση συνδεσιμότητας κόμβων** Βεβαιωνόμαστε ότι οι δύο κόμβοι είναι συνδεδεμένοι εκτελώντας την εντολή **admin.peers** στην κονσόλα javascript του Geth οποιουδήποτε κόμβου. Θα πρέπει να εμφανίζει πληροφορίες σχετικά με τον συνδεδεμένο ομότιμο. Υπενθυμλιζουμε ότι για να ανόιξουμε την κονσόλα αυτή, εκτελούμε στο τερματικό την παρακάτω εντολή:

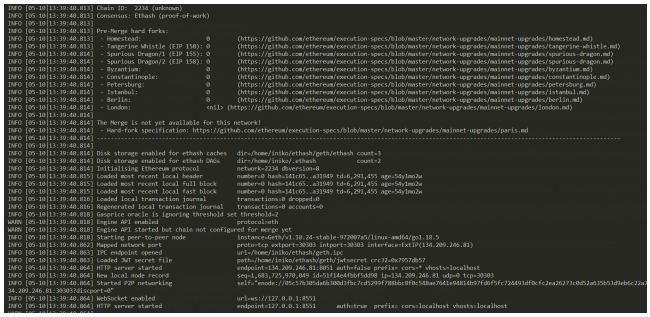
```
geth attach /path/to/datadir/node1/geth.ipc
```

- **Βήμα 8: Εκκίνηση διαδικασίας bft** Η συγκεκριμένη υλοποίηση του bft αλγορίθμου έχει την ακόλουθη ιδιαιτερότητα. Για να εκκινήσει σωστά πρέπει να έχουν σταλεί κάποια transactions μεταξύ των συμμετεχόντων κόμβων και έπειτα να εκκινήσει η διαδικασία για την επίτευξη consensus. Επομένως αφού στείλουμε μερικά transactions εκκινούμε τη διαδικασία από την κονσόλα javascript του Geth του εκάστοτε κόμβου με την εντολή:

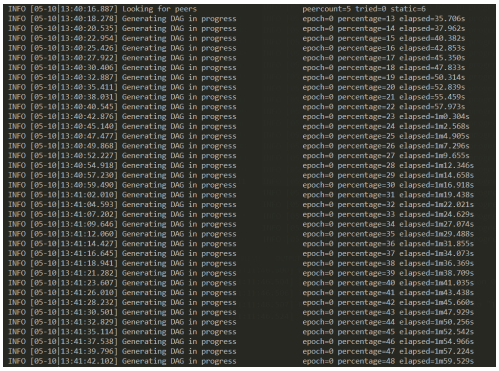
```
miner.start()
```

- **Βήμα 9: Αλληλεπίδραση με το δίκτυο** Στο σημείο αυτό το ιδιωτικό μας δίκτυο είναι ετοιμο. Μπορούμε να αλληλεπιδράσουμε με αυτο, να δοκιμάσουμε να στείλουμε transactions, να δούμε τα επιβεβαιωμένα blocks και τα transactions που εμπεριέχουμε και γενικότερα να πειρματιστούμε. Να σημειωθεί ότι για να πραγματοποιηθεί consensus χρειάζεται να συμμετέχουν στο δίκτυο τουλάχιστον 4 κόμβοι, ενώ συμφωνία επιτυγχάνεται, όταν επιτευχθεί το Quorum, δηλαδή να έχουν ψηφίσει για το ίδιο block τα 2/3 των κόμβων

Παρακάτω ακολουθούν φωτογραφίες από διάφορες φάσεις ενός λειτουργικού δικτύου με 6 κόμβους.



(a) Network Initialization



(b) BFT-consensus

Εικόνα 3.2: *Λειτουργικό δίκτυο Ethereum με μηχανισμό συναίνεσης NCCU-BFT*

### 3.1.3 Geth με PoS

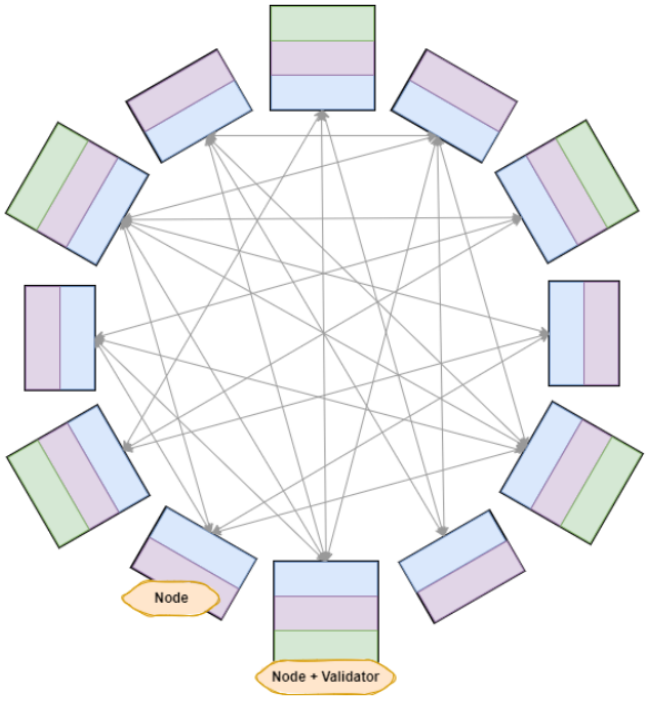
Όπως γνωρίζουμε το Ethereum είναι ένα αποκεντρωμένο δίκτυο κόμβων που επικοινωνούν μέσω ομότιμων συνδέσεων (peer-to-peer conndections). Οι συνδέσεις αυτές δημιουργούνται από υπολογιστές που εκτελούν το εξειδικευμένο λογισμικό, το Geth, ωστόσο στην περίπτωση του Ethereum 2.0 το λογισμικό αυτό δεν είναι αρκετό.

Μέχρι πρότινος, η λειτουργία ενός κόμβου Ethereum σήμαινε απλώς την εγκατάσταση του Geth και την εκτέλεση μιας απλής εντολής για τον συγχρονισμό της αλυσίδας. Μετά τη μετάβαση σε proof-of-stake, η εκτέλεση ενός κόμβου στο Ethereum απαιτεί δύο στοιχεία. Στο Ethereum 2.0 υπάρχουν δύο βασικοί τύποι κόμβων: οι **κόμβοι εκτέλεσης-execution nodes** και οι **κόμβοι φάροι-beacon nodes**. Ωστόσο στην καθομιλουμένη, ένας κόμβος αναφέρεται σε έναν κόμβο εκτέλεσης και έναν κόμβο φάρο που συνεργάζονται. Από τους δύο αυτούς τύπους κόμβων προκύπτει και το λογισμικό που είναι απαραίτητο για το στήσιμο ενός ¨ολοκληρωμένου κόμβου¨.

- **execution layer software** που είναι υπεύθυνο για την επεξεργασία των συναλλαγών και τα έξυπνα συμβόλαια, δηλαδή το Geth
- **consensus layer software** που είναι υπεύθυνο για την εκτέλεση της λογικής proof-of-stake. Στην περιγραφή που θα ακολουθήσει έχουμε επιλέξει να χρησιμοποιοήσουμε την υλοποίηση του **Prysm**, που είναι μια Go υλοποίηση ανοιχτού κώδικα του πρωτοκόλλου

proof-of-stake του Ethereum. Να σημειωθεί εδώ ότι υπάρχουν και άλλες υλοποιήσεις του πρωτοκόλλου αυτού για το Ethereum όπως τα: **Lighthouse** σε Rust, **Nimbus** σε Nim, **Teku** σε Java και **Lodestar** σε Typescript.

Όταν οι χρήστες ποντάρουν τα νομίσματα τους για να συμμετάσχουν στον μηχανισμό συναίνεσης proof-of-stake του Ethereum , χρησιμοποιούν ένα ξεχωριστό κομμάτι λογισμικού που ονομάζεται validator client, το οποίο συνδέεται με τον κόμβο beacon του Prysm. Πρόκειται για ειδικό κομμάτι λογισμικού που διαχειρίζεται τα κλειδιά του validator και καθήκοντα όπως η παραγωγή νέων block και η ψηφοφορία για τα προτεινόμενα block άλλων. Οι validator client συνδέονται στο δίκτυο Ethereum μέσω των κόμβων beacon, οι οποίοι εξαρτώνται από τους κόμβους εκτέλεσης.




Εικόνα 3.3: *Δίκτυο Ethereum με αλγόριθμο συναίνεσης Proof of Stake*

Παρακάτω θα ακολουθήσει η περιγραφή για το στήσιμο του περιβάλλοντος αυτού με δύο κόμβους. Η επέκταση του δικτύου σε περισσότερους κόμβους πραγματοποιείται με επανάληψη των βημάτων.

- **Βήμα 1: Εγκατασταση της Go, του Geth και του Prysm** Επιβεβαίωση ότι η Go, το Geth και το Prysm είναι εγκατεστημένα στο συστημα. Για διευκόλυνση της διαδικασίας προτιμούμε να εγκαταστήσουμε τα Geth και Prysm στην ίδια τοποθεσία. Αυτο μπορεί να γίνει με τη βοήθεια των εντολών:

```
go version
```

```
geth --help
```

```
./beacon-chain help
```

```
./validator help
```

```
./prysmctl --help
```

*Για διευκόλυνση της διαδικασίας προτιμούμε να εγκαταστήσουμε τα Geth και Prysm στην ίδια τοποθεσία, έστω devnet.*

- **Βήμα 2: Configuration files για τη ρύθμιση του Prysm και του Go-Ethereum** Από την πλευρά του **Prysm**, δημιουργούμε ένα αρχείο με όνομα config.yml στο φάκελο devnet που περιέχει τα εξής:

```
CONFIG_NAME: interop
PRESET_BASE: interop

# Genesis
GENESIS_FORK_VERSION: 0x20000089

# Altair
ALTAIR_FORK_EPOCH: 2
ALTAIR_FORK_VERSION: 0x20000090

# Merge
BELLATRIX_FORK_EPOCH: 4
BELLATRIX_FORK_VERSION: 0x20000091
TERMINAL_TOTAL_DIFFICULTY: 50

# Capella
CAPELLA_FORK_VERSION: 0x20000092

# Time parameters
SECONDS_PER_SLOT: 12
SLOTS_PER_EPOCH: 6

# Deposit contract
DEPOSIT_CONTRACT_ADDRESS: 0x4242424242424242424242424242424242424242
```

Η παραπάνω διαμόρφωση περιέχει πληροφορίες σχετικά με τις διαφορετικές εκδόσεις hard-fork που απαιτούνται για την εκτέλεση του Prysm και έχει κάποιες προσαρμοσμένες παραμέτρους για να διευκολύνει την εκτέλεση του δικτύου.

Από την πλευρά του geth, θα χρειαστεί και σε αυτή την περίπτωση να προυπάρχει ένας λογαριασμός ώστε να προχωρήσει η αλυσίδα από τη γένεση της με εξόρυξη μέχρι να φτάσει στη λειτουργία proof-of-stake. Πιο συγκεκριμένα κατα την αρχικοποίηση της η αλυσίδα θα τρέχει με μηχανισμό συναίνεσης proof-of-authority και όταν ξεπεράσει τη δυσκολία 50 θα μπει σε λειτουργία proof-of-stake. Παρακάτω ορίζεται το genesis αρχείο που θα χρησιμοποιηθεί για το δίκτυο μας:

```
{
        "config": {
                "chainId": 32382,
                "homesteadBlock": 0,
                "daoForkSupport": true,
                "eip150Block": 0,
                "eip150Hash":"0x0000000000000000000000000000000000000000
0000000000000000000000000",
                "eip155Block": 0,
                "eip158Block": 0,
                "byzantiumBlock": 0,
                "constantinopleBlock": 0,
                "petersburgBlock": 0,
                "istanbulBlock": 0,
                "muirGlacierBlock": 0,
                "berlinBlock": 0,
                "londonBlock": 0,
                "arrowGlacierBlock": 0,
                "grayGlacierBlock": 0,
                "shanghaiTime": 1681499865,
                "terminalTotalDifficulty": 0,
                "clique": {
                        "period": 5,
                        "epoch": 30000
                }
        },
        "nonce": "0x0",
        "timestamp": "0x6439a649",
        "extraData": "0x0000000000000000000000000000000000000000
0000000000000000000000000123463a4b065722e99115d6c222f267d9c
abb5240000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000",
        "gasLimit": "0x1c9c380",
        "difficulty": "0x1",
        "mixHash": "0x0000000000000000000000000000000000000000000000000000
00000000000000",
        "coinbase": "0x0000000000000000000000000000000000000000",
        "alloc": {
                "123463a4b065722e99115d6c222f267d9cabb524": {
                        "balance": "0x43c33c1937564800000"
```

```
                },
                "14dc79964da2c08b23698b3d3cc7ca32193d9955": {
                        "balance": "0x21e19e0c9bab2400000"
                },
                "15d34aaf54267db7d7c367839aaf71a00a2c6a65": {
                        "balance": "0x21e19e0c9bab2400000"
                },
                "1cbd3b2770909d4e10f157cabc84c7264073c9ec": {
                        "balance": "0x21e19e0c9bab2400000"
                },
                "23618e81e3f5cdf7f54c3d65f7fbc0abf5b21e8f": {
                        "balance": "0x21e19e0c9bab2400000"
                },
                "2546bcd3c84621e976d8185a91a922ae77ecec30": {
                        "balance": "0x21e19e0c9bab2400000"
                },
                "3c44cdddb6a900fa2b585dd299e03d12fa4293bc": {
                        "balance": "0x21e19e0c9bab2400000"
                },
                "4242424242424242424242424242424242424242": {
                        "code": "...",
                        "balance": "0x0"
                }
        },
        "number": "0x0",
        "gasUsed": "0x0",
        "parentHash":
 "0x0000000000000000000000000000000000000000
00000000000000000000",
        "baseFeePerGas": null
}
```

Το παραπάνω αρχείο ρυθμίζει τη διαμόρφωση για το go-ethereum, η οποία εκκινεί ορισμένους λογαριασμούς με ένα υπόλοιπο ETH και αναπτύσσει ένα συμβόλαιο κατάθεσης επικυρωτών στη διεύθυνση 0x4242424242424242424242424242424242424242 24242424242, το οποίο χρησιμοποιείται για νέους επικυρωτές για να καταθέσουν τον απαραίτητο αριθμό κρυπτονομισμάτων και να ενταχθούν στην αλυσίδα proof-of-stake. Ο λογαριασμός με τον οποίο τρέχουμε το geth μέχρι να μεταβούμε στη λειτουργία proof-of-stake, είνα ο 0x123463a4b065722e99115d6c222f267d9cabb524. Επιλέον με στόχο να περιορίσουμε την έκταση που θα λάβει το αρχείο Genesis δεν συμπεριλάβαμε τον κώδικα του συμβολαίου κατάθεσης επικυρωτών. Ολόκληρο το αρχείο genesis μπορεί να βρεθεί στον παρακάτω σύνδεσμο

- **Βήμα 3: Αρχικοποίηση πρώτου κόμβου εκτέλεσης και εισαγωγή λογαριασμού** Αρχικοποιούμε τον πρώτο κόμβο και στη συνέχεια να εισάγουμε το λογαριασμό που θέλουμε στον πρώτο κόμβο με τη βοήθεια ενός secret.json αρχείου που αποτελεί το private key του λογαριασμού μας. Αυτό συμβαίνει με τη βοήθεια των ακόλουθων εντολών.

```
geth --datadir /path/to/datadir/node1 init genesis.json
```

```
geth --datadir /path/to/datadir/node1 account import secret.json
```


```
geth --datadir /path/to/datadir/node1 account import secret.json
```


Αντικαταθιστούμε το /path/to/datadir/node1 με την τοποθεσία όπου θέλουμε να αποθηκεύσουμε τα δεδομένα blockchain για τον πρωτο κόμβο και το /path/to/genesis.json με την τοποθεσία του αρχείου Genesis.

- **Βήμα 4: Εκκίνηση κόμβου εκτέλεσης** Με την παρακάτω εντολή διαμορφώνοντας κατάλληλες τις τοποθεσίες και τα ορίσματα καλούμε την παρακάτω εντολή:

```
geth --http --http.addr "public_IP" --http.api "eth,engine"
--datadir=/path/to/datadir/node1 --allow-insecure-unlock
--unlock="account" --password="your_passwor" --nodiscover console
--syncmode=full --mine --miner.etherbase="account"
```

- **Βήμα 4: Εκκίνηση beacon κόμβου και validator client** Για να τρέξουμε έναν beacon κόμβο και έναν validator client, το Prysm θα χρειαστεί μια κατάσταση γένεσης, η οποία είναι ουσιαστικά κάποια δεδομένα που του λένε το αρχικό σύνολο των επικυρωτών. Θα δημιουργήσουμε μια κατάσταση γένεσης από ένα ντετερμινιστικό σύνολο κλειδιών με την παρακάτω εντολή:

```
./prysmctl testnet generate-genesis --num-validators=32
--output-ssz=genesis.ssz --chain-config-file=config.yml
--override-eth1data=true
```

Επιπλέον θα φροντίσουμε να μεταφέρουμε το αρχείο εξόδου και στο server στον οποίο θα σηκώσουμε το δεύτερο κόμβο. Ένας τρόπος να το κάνουμε αυτό είναι η παρακάτω εντολή:

```
scp /pathto/genesis.ssz user@yoursecondnodeIP:/pathto/genesis.ssz
```

Στη συνέχεια θα σηκώσουμε τον beacon κόμβο και τον validator client με τη βοήθεια των παρακάτω εντολών:

**Beacon Node**

```
./beacon-chain \
--datadir=/beacondata \
--min-sync-peers=1 \
--genesis-state=genesis.ssz \
--bootstrap-node= \
--chain-config-file=config.yml \
--config-file=config.yml \
--chain-id=32382 \
--execution-endpoint=http://public_IP:8551 \
--accept-terms-of-use \
--p2p-host-ip=public_IP \
--jwt-secret=/yourdatadirectory/geth/jwtsecret
```

Από τα παραπάνω ορίσματα το πιο ενδιαφέρον είναι το –execution-endpoint. Είναι απαραίτητο καθώς ορίζει το την πόρτα από την οποία θα επικοινωνούν οι δύο διαφορετικοί κόμβοι του δικτύου μας.

**Validator client**

```
./validator \
--datadir=validatordata \
--accept-terms-of-use \
--interop-num-validators=64 \
--interop-start-index=0 \
--force-clear-db \
--chain-config-file=config.yml \
--config-file=config.yml
```

- **Βήμα 5: Επανάληψη των βημάτων 2 έως 4 για το δεύτερο κόμβο** Πριν ξεκινήσουμε την επανάληψη των βημάτων θα πρέπει να προηγηθούν κάποιες ενέργειες ακόμη. Αρχικά εκτελούμε στον πρώτο κόμβο την παρακάτω εντολή:

```
curl localhost:8080/p2p
```

Η εντολή αυτή θα έχει έξοδο της μορφής:

```
bootnode=[]
self=enr:-MK4.....Iu4A,
/ip4/10.0.0.74/tcp/13000/p2p/16Uiu2HAmJg9Sfy8bX4wyjZNTi8soJrdPt9E9pPzJS
mewN5rLoRM6
0 peers
```

Το κομμάτι της εξόδου που ξεκινάει με /ip4/.. είναι απαραίτητο για την προσθήκη ομότιμου κόμβου στο επίπεδο του consensus οπότε το σημειώνουμε κάπου. Στη συνέχεια σε νέο τερματικό επαναλμβάνουμε τα βήματα 2 και 3 προσμαρμόζοντας εννοείται τα μονοπάτια των αρχείων και τα ορίσματα για την εκκίνηση του δεύτερου κόμβου εκτέλεσης. Έπειτα προαιρετικά και για τη διευκόλυνση μας με τη βοήθεια της παρακάτω εντολής αποθηκεύουμε στη μεταβλητή PEER το string που σημειώσαμε παραπάνω:

```
export PEER=/ip4/10.0.0.74/tcp/13000/p2p/16Uiu2HAmJg9Sfy8bX4wyjZNTi8soJ
rdPt9E9pPzJSmewN5rLoRM6
```

Στη συνέχεια επαναλμβάνουμε το βήμα 4 με τη διαφοροποίηση ότι για την εκκίνηση του beacon κόμβου προσθέτουμε επιπλέον ορίσματα:

```
./beacon-chain \
--datadir=beacondata \
--min-sync-peers=1 \
--genesis-state=genesis.ssz \
--bootstrap-node= \
--chain-config-file=config.yml \
--config-file=config.yml \
--chain-id=32382 \
--execution-endpoint=http://public_IP:8551 \
--accept-terms-of-use \
--p2p-host-ip=public_IP \
--peer=$PEER \
--jwt-secret=/yourdatadirectory/jwtsecret
```

Προσθέτουμε τα απαραίτητα ορίσματα ώστε η δύο διαφορετικοί κόμβοι-φάροι να επικοινωνήσουν μεταξύ τους. Μετά την εκτέλεση των εντολών αυτών αν επανεκτελέσουμε

την εντολή **curl localhost:8080/p2p** θα παρατηρήσουμε ότι οι κόμβοι πλέον επικοινωνούν. Επιπλέον στα logs του beacon node θα παρατηρήσουμε.


Εικόνα 3.4: *Συγχρονισμός beacon nodes*

- **Βήμα 6: Συνδέουμε το execution layer των δύο κόμβων** Αυτό το βήμα πραγματοποιείται όπως και στην περίπτωση του go-ethereum με proof-of-work με εκτέλεση της παρακάτω εντολής στην κονσόλα javascript του Geth ενός από τους δύο συμμετέχοντες κόμβους:

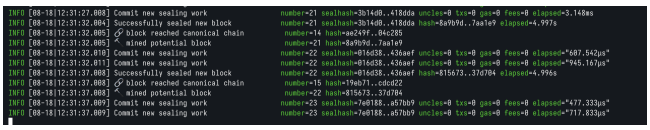
```
admin.addPeer(enodeURL)
```

- **Βήμα 7: Επαλήθευση συνδεσιμότητας των δύο κόμβων** Αυτό το βήμα πραγματοποιείται με τη βοήθεια των παρακάτω εντολών:

```
admin.peers //in javascript console
```

- **Βήμα 8: Αλληλεπίδραση με το δίκτυο** Μετά από μερικά λεπτά το σύστημα φτάνει στη δυσκολία 50 όποτε και μπαίνει στη λειτουργία proof of stake.Μπορούμε να αλληλεπιδράσουμε με αυτο, να δοκιμάσουμε να στείλουμε τρανσαστιονς, να δούμε τα επιβεβαιωμένα βλοςκς και τα τρανσαστιονς που εμπεριέχουμε και γενικότερα να πειραματιστούμε.

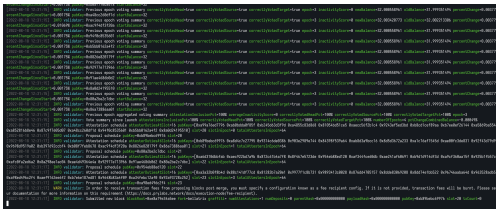
  Παρακάτω παρακολουθούν φωτογραφίες από διάφορες φάσεις του δικτύου.



(a) Network before PoS mode


(b) Entering PoS

(c) Beacon Node



(d) Validator client

Εικόνα 3.5: *Λειτουργικό δίκτυο Ethereum με μηχανισμό συναίνεσης Proof-of-Stake*

## 3.2 Διεξαγωγή Πειραμάτων με τη χρήση του framework Blockbench

Στο σημείο αυτό θα κάνουμε μια γενικότερη ανάλυση της διαμόρφωσης που θα χρειαστεί να κάνουμε στο Blockbench καθώς και του τρόπου με τον οποίο λειτουργεί ενώ στη συνέχεια

θα μιλήσουμε ειδικότερα για κάθε περιβάλλον ξεχωριστά. Η υλοποίηση του Blockbench που χρησιμοποιήθηκε βρίσκεται στο github repository. Επιπλέον να σημειωθεί ότι ο πειραματισμός πραγματοποιήθηκε σε περιβάλλον Linux, σε VM's του Digital Ocean με χαρακτηριστικά 4 GB RAM, 80GB SSD.

Για να πραγματοποιήσουμε πειράμετα στο Consensus layer των ιδιωτικών μας δικτύων με τη βοήθεια του Blockbench θα πρέπει αρχικά να καταλάβουμε πως λειτουργεί. Αυτό που συμβαίνει πίσω από αυτή την υλοποίησης είναι ότι ένα κύριο κόμβος αρχικά σηκώνει το εκάστοτε δίκτυο του Ethereum και έπειτα με τη βοήθεια ενός smart contract στέλνει σε αυτό το δίκτυο transactions και κάνει τις απαραίτητες μετρήσεις. Ας ξεκινήσουμε λοιπον ορίζοντας κάποιες global παραμέτρους που θα χρειαστούν για την επιτυχή πραγματοποίηση της παραπάνω λειτουργίας. Τα παρακάτω βήματα θα πρέπει να πραγματοποιηθούν με ακρίβεια για όλες τους συμμετέχοντες στη διαδικασία servers. Μεγάλη διευκόλυνση της διαδικασία θα ήταν η ύπαρξη ενός Network File System (NFS), το οποίο θα επέτρεπε στους συμμετέχοντες servers να μοιράζονται καταλόγους και αρχεία μεταξύ τους μέσω δικτύου. Η διαδικασία λοιπόν για το στήσιμο του περιβάλλοντος αυτού θα πραγματοποιηθεί εντός του καταλόγου blockbench/benchmark/ethereum/ του repository που δόθηκε παραπάνω και είναι η εξής:

- **Βήμα 1: Διαμόρφωση του αρχείου env.sh** Το αρχείο αυτό περιέχει όλα τα απαραίτητα paths για τη σωστή εκκίνηση του δικτύου μας. Δίπλα σε κάθε παράμετρο φαίνεται σε σχόλιο η χρησιμότητα του.

```
# folder that contains all benchmark scripts
#(this could be on a network share)
ETH_HOME=/your_path/blockbench/benchmark/ethereum
# file that contains ip addresses of servers that
#should be used for setting up the ethereum network
HOSTS=$ETH_HOME/hosts
# file that contains ip addresses of servers
#that should be used for running the benchmark clients
CLIENTS=$ETH_HOME/clients
# folder in which ethereum nodes should store the ethereum data
ETH_DATA=/your_path/eth_data
# folder in which benchmark clients should store their log files
LOG_DIR=/your_path/log_data
# folder that contains the benchmark (client)
#executable (make sure that you have build the client)
#(this could be on a network share)
EXE_HOME=/your_path/blockbench/src/macro/kvstore
# name/type of the benchmark, ycsb is the macro-benchmark
#which contains the driver for measuring consensus layer
BENCHMARK=ycsb
```

- **Βήμα 2: Διαμόρφωση των αρχείων hosts και clients** Εντός των αρχείων αυτών θα πρέπει να βάλουμε στους hosts τις IP των server που θα αποτελέσουν τους κόμβους του εκάστοτε δικτύου Ethereum ενώ στους clients τις IP των server που θα αποτελέσουν τους clients που θα στελνουν τις συναλλαγές στο δίκτυο.

Πέραν των κόμβων που συμμετέχουν στο δίκτυο και των κόμβων που θα αποτελέσουν τους

clients υπάρχει και ένας ακόμη κόμβος που θα εκκίνει τη διαδικασία για την πραγματοποιήση των πειραμάτων.

Στη συνέχεια ακολουθεί περιγραφή για την πραγματοποίηση των πειραμάτων για κάθε διαμόρφωση του δικτύου Ethereum.

### 3.2.1 Διεξαγωγή Πειραμάτων σε ιδιωτικό δίκτυο Ethereum με μηχανισμό συναίνεσης Proof-of-Work

Η διεξαγωγή πειραμάτων στο δίκτυο αυτό είναι η ευκολότερη από άποψη χρόνου, καθώς η υλοποίηση τοθ Blockbench αναπτύχθηκε για το δίκτυο αυτό οπότε η διαδικασία είναι αυτοματοποιημένη. Για τη διεξαγωγή πειραμάτων στο δίκτυο αυτό, αφού πραγματοποιηθεί η διαμόρφωση των αρχείων που αναφέρθηκε παραπάνω, θα χρησιμοποιήσουμε την εκτέλεση της παρακάτω εντολής από τον helper_server:

```
./run-bench.sh <nservers> 1 <nclients> <txrate>
```

Κατα την εκτέλεση της εντολής αυτής αυτό που θα συμβεί είναι να εκκινήσουν nservers για mining, nclients που ο καθένας εκδίδει txrate συναλλαγές ανά δευτερόλεπτο. Κάθε client δρα σε 2 κόμβους. Τα αποτελέσματα αποθηκεύονται στο LOG_DIR που ορίστηκε παραπάνω.

Πιο συγκεκριμένα κατά την εκτέλεση της παραπάνω εντολής αυτα που θα συμβούν είναι:

1. **Αρχικοποίηση**

   - init-all.sh <nservers>: πηγαίνει σε κάθε έναν από τους nservers και καλεί το init.sh <nservers>
   - init.sh <nservers> αρχικοποιεί το γετη σε έναν κόμβο. Κάνει 2 πράγματα:
     - Χρησιμοποιεί το αρχείο ETH_HOME/CustomGenesis_<nservers>.json ως genesis αρχείο (προσοχή! φροντίζουμε να επιλέξουμε το κατάλληλο genesis.
     - Δημιουργία νέου λογαριασμού με κενό κωδικό πρόσβασης

2. **Εκκίνηση mining**

   - start-all.sh <nservers>: Εκκίνηση ενός δίκτυου με nservers miners με τα ακόλουθα βήματα:
     - gather.sh <nservers>: μεταβαίνει σε κάθε έναν από τους κόμβους στους ηοστς και:
       * συλλέγει τις πληροφορίες ομότιμων (του miner) που δημιουργήθηκαν κατά το βήμα αρχικοποίησης. Αυτό γίνεται με τη χρήση enode.sh
       * Προσθήκη των πληροφοριών αυτών σε ένα κοινόχρηστο αρχείο addPeer.txt
     - Πηγαίνει σε κάθε κόμβο στους hosts και καλεί το αρχείο start-mining.sh. Αυτό περιλαμβάνει:
       * Εκκίνηση του geth σε λειτουργία εξόρυξης, με παραμετροποιήσιμες επιλογές
       * Προσθήκη ομότιμων χρησιμοποιώντας το περιεχόμενο του addPeer.txt

- Ύπνος για μια διάρκεια M sec επαρκής για όλους τους miners ώστε να ολοκληρώσουν τη δημιουργία DAG και να συγχρονιστούν στο Blockchain μετά την εξόρυξη μερικών δεκάδων block.

3. **Εκκίνηση των clients**

   - start-multi-clients.sh <nclients> <nservers> όπου η δουλεία του είναι:
     - να εκκινήσει το driver ωστε να αρχίσουν να στέλνονται συναλλαγές από κάθε client στους κόμβους του δικτύου.
     - αφήνει τους clients να τρέξουν για M sec, μια αρκετά μεγάλη διάρκεια για τη συλλογή αρκετών δεδομένων. Στη συνέχεια, τερματίζουν οι διεργασίες των clients

4. **Εκκαθάριση του δικτύου** Στο σημείο αυτό τερματίζουμε του δίκτυο blockchain και διαγράφουμε τα σχετικά δεδομένα. Πραγματοποιείται με το αρχείο **stop-all.sh**

### 3.2.2 Διεξαγωγή Πειραμάτων σε ιδιωτικό δίκτυο Ethereum με μηχανισμό συναίνεσης BFT

Στην περίπτωση αυτή η διαδικασία γίνεται πιο χρονοβόρα καθώς δεν μπορούμε να αυτοματοποιήσουμε όλη τη διαδικασία καθώς χρειάζεται σε κομβικα σημεία της δημιουργίας του ιδιωτικού δικτύου να παρέμβουμε. Πρακτικά η διαφοροποίηση με την παραπάνω διαδικασία είναι ότι στήνουμε το δίκτυο οπως περιγράφη στο section Geth με NCCU BFT και στη συνέχεια φροντίζουμε εντός του αρχείου run_bench.sh να βάλουμε σε σχόλια τα αρχεία init_all.sh και start-all.sh. Στη συνέχεια αφού στήσουμε το ιδιωτικό δίκτυο φροντίζουμε να περιμένουμε μερικά δεύτερόλεπτα (αυτό μπορεί να οριστεί εντός του αρχείου run_bench.sh) και καλούμε το run_bench.sh από τον βοηθητικό κόμβο. Σε αυτή την περίπτωση τα ορίσματα που μας ενδιαφέρει να ορίσουμε σωστά είναι τα <n_clients> και <txnrate>.

### 3.2.3 Διεξαγωγή Πειραμάτων σε ιδιωτικό δίκτυο Ethereum με μηχανισμό συναίνεσης Proof of Stake

Η συγκεκριμένη περίπτωση είναι αντίστοιχη με αυτή του δικτύου Ethereum με NCCU BFT ως μηχανισμό συναίνεσης. Επομένως η διαφοροποίηση είναι ότι στήνουμε το δίκτυο οπως περιγράφη στο section Geth με Proof-of-Stake και στη συνέχεια φροντίζουμε εντός του αρχείου run_bench.sh να βάλουμε σε σχόλια τα αρχεία init_all.sh και start-all.sh. Στη συνέχεια αφού στήσουμε το ιδιωτικό δίκτυο φροντίζουμε να περιμένουμε μερικά δεύτερόλεπτα (αυτό μπορεί να οριστεί εντός του αρχείου run_bench.sh) και καλούμε το run_bench.sh από τον βοηθητικό κόμβο. Σε αυτή την περίπτωση τα ορίσματα που μας ενδιαφέρει να ορίσουμε σωστά είναι τα <n_clients> και <txnrate>.

## 3.3 Επεξεργασία Δεδομένων

Τα αποτελέσματα που προκύπτουν μετά τη διεξαγωγή των παραπάνω πειραμάτων είναι τα εξής. Κάθε κόμβος που συμμετείχε στο δίκτυο Ethereum μετά το πέρας των πειραμάτων έχει

logs της παρακάτω μορφής:

```
Smart contract ycsb deploy ready
Current TIP = 222
In the last 2s, tx count = 0 latency = 0 outstanding request = 3
In the last 2s, tx count = 0 latency = 0 outstanding request = 320
In the last 2s, tx count = 0 latency = 0 outstanding request = 624
polled block 222 : 0 txs
In the last 2s, tx count = 0 latency = 0 outstanding request = 928
polled block 223 : 0 txs
In the last 2s, tx count = 0 latency = 0 outstanding request = 1246
polled block 224 : 134 txs
In the last 2s, tx count = 106 latency = 1028.88 outstanding request = 1446
In the last 2s, tx count = 0 latency = 0 outstanding request = 1750
In the last 2s, tx count = 0 latency = 0 outstanding request = 2054
In the last 2s, tx count = 0 latency = 0 outstanding request = 2374
In the last 2s, tx count = 0 latency = 0 outstanding request = 2678
polled block 225 : 134 txs
In the last 2s, tx count = 0 latency = 0 outstanding request = 2982
polled block 226 : 134 txs
In the last 2s, tx count = 0 latency = 0 outstanding request = 3293
```

Κάποιες βασικές παρατηρήσεις στα παραπάνω αποτελέσματα είναι οι εξής:

- Κάθε φορά που δημιουργείται ένα νέο block, ο driver αναφέρει τον αριθμό των συναλλαγών (txs) που περιλαμβάνονται σε αυτό π.χ. στο block 222 υπήρχαν 0 txs και στο block 224 υπήρχαν 134 txs. Βάση αυτού μπορούμε να υπολογίσουμε το **throuphput** του δικτύου αθροιζοντας τα transactions από όλα τα block που δημιουργήθηκαν κατά τη διεξαγωγή του εκάστοτε πειράματος.

- Με βάση τα txn στα εξορυγμένα block, ο driver καθορίζει τον αριθμό των txn που επεξεργάστηκαν τα τελευταία 2 δευτερόλεπτα. Για αυτά τα txn αναφέρει τη συσσωρευμένη καθυστέρηση (accumulated latency), δηλαδή το άθροισμα των καθυστερήσεων των txn του κόμβου σε δευτερόλεπτα. Για τον προσδιορισμό της καθυστέρησης ενός txn το πρόγραμμα παρακολουθεί πότε υποβλήθηκε ένα txn.Το άθροισμα των καθυστερήσεων των συναλλαγών αναφέρεται ανά client και άνα κόμβο στον οποίο αποστέλλει τις συναλλαγές, οπότε μπορούμε να υπολογίσουμε τη μέση καθυστέρηση συναλλαγών ανά client διαιρώντας με τον αριθμό των συνολικών txn που αφορούν τον συγκεκριμένο client. Στη συνέχεια αφού έχουμε μια μέση καθυστέρηση για κάθε client μπορούμε προσθέτοντας όλες τις μέσες καθυστερήσεις και διαιρώντας με το συνολικό αριθμό των clients να βρούμε το μέσο latency του δικτύου.

Βάση των παρατηρήσεων αυτών στο επόμενο κεφάλαιο θα πραγματοποιηθεί ανάλυση των αποτελεσμάτων των πειραμάτων που διεξήχθησαν στα τρία ιδιωτικά δίκτυα του Ethereum που περιγράφηκαν παραπάνω.

# Κεφάλαιο 4

# Αξιολόγηση

Η αξιολόγηση των αποτελεσμάτων που προκύπτουν από τη συγκριτική αξιολόγηση και την ανάλυση επιδόσεων αποτελεί κρίσιμη πτυχή της αξιολόγησης της αποτελεσματικότητας και της αποδοτικότητας των συστημάτων blockchain. Σε αυτό το κεφάλαιο, παρουσιάζουμε μια ολοκληρωμένη αξιολόγηση των αποτελεσμάτων που λαμβάνονται από πειραματισμούς με τη χρήση του Blockbench στα ιδιωτικά μας δίκτυα που βασίζονται στο Ethereum. Αναλύοντας τα αποτελέσματα, στοχεύουμε να παράσχουμε μια εμπεριστατωμένη κατανόηση των δυνατών σημείων και των περιορισμών κάθε αλγορίθμου συναίνεσης και της καταλληλότητάς τους για συγκεκριμένες περιπτώσεις χρήσης.

Τα πειράματα που διεξήχθησαν αφορούν τα τρία διαφορετικά δίκτυα που αναλύθηκαν στα παραπάνω κεφάλαια σε δυο περιπτώσεις:

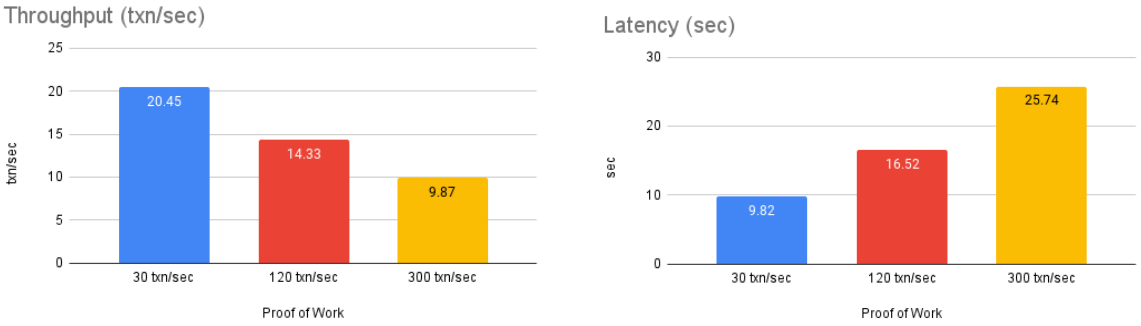
- **δίκτυα 6 κόμβων:** πραγματοποιήσαμε πειράματα με φόρτους εργασίας 30txn/sec, 120txn/sec και 300txn/sec.
- **δίκτυα 12 κόμβων:** πραγματοποιήσαμε πειράματα με φόρτους εργασίας 36txn/sec, 120txn/sec και 300txn/sec, 600txn/sec και 1200txn/sec

Οι φόρτοι εργασίας αυτοί αφορούν όλο το δίκτυο και όχι κάθε κόμβο ξεχωριστά.

## 4.1 Αξιολόγηση Επίδοσης Go-Ethereum με Proof-of-Work

### 4.1.1 Αξιολόγηση επίδοσης δικτύου με 6 κόμβους

Στις παρακάτω γραφικές παραστάσεις φαίνεται η απόδοση του δικτύου Ethereum με 6 κόμβους και τους τρεις διαφορετικούς φόρτους εργασίας που του επιβάλλαμε.




(a) Throughput (b) Latency

Εικόνα 4.1: *Πειραματισμός σε Δίκτυο PoW 6 κόμβων*

Ο PoW είναι ένας σχετικά αργός και απαιτητικός σε πόρους αλγόριθμος συναίνεσης. Απαιτεί από τους κόμβους να επιλύουν πολύπλοκους μαθηματικούς γρίφους, οι οποίοι μπορεί να απαιτούν σημαντική υπολογιστική ισχύ και χρόνο.

- **Workload δικτύου 30txn/sec**
  Για αυτό το σενάριο χαμηλής κίνησης παρατηρούμε μια σχετικά καλή απόδοση του δικτύου από άποψη throughput και latency. Πιο συγκεκριμένα, όσον αφορά το throughput παρατηρούμε ότι το δίκτυο δεν μπορεί να διατηρήσει τις πλήρεις 30 συναλλαγές από δευτερόλεπτο ωστόσο πετυχαίνει ένα καλό ποσοστό. Σχετικά με το latency παρατηρούμε ότι είναι αρκετά χαμηλό τουλάχιστον σε σχέση με αυτό που παρατηρείται στο δημόσιο δίκτυο του Ethereum με Proof of Work ως αλγόριθμο συναίνεσης.

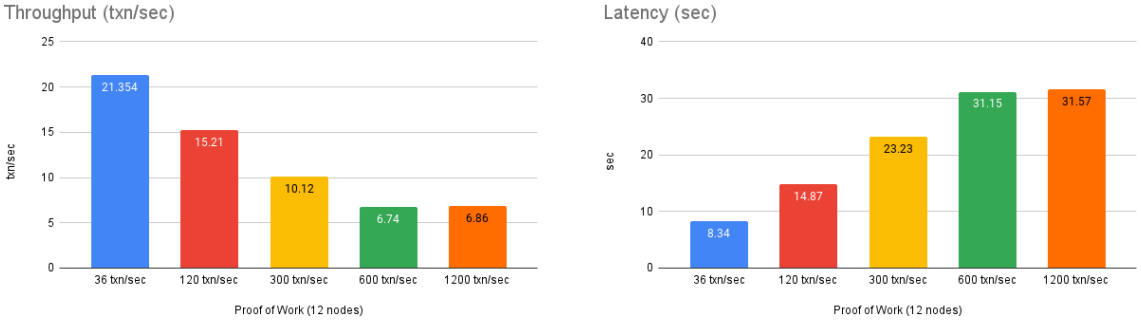
- **Workload δικτύου 120txn/sec**
  Για αυτό το σενάριο υψηλότερη κίνησης κίνησης παρατηρούμε λίγο χειρότερη εικόνα απο αυτή του προηγούμενου φόρτου εργασίας. Πιο συγκεκριμένα, όσον αφορά το throughput παρατηρούμε ότι το δίκτυο πραγματοποιεί λίγο λιγότερες από 20 συναλλαγές ανα δευτερόλεπτο και επομένως ο νέος φόρτος έχει αρχίζει να επηρεάζει την ταχύτητα που επεξεργάζεται το δίκτυο της συναλλαγες και έχει προκαλέσει κάποια σχετική συμφόρηση. Σχετικά με το latency παρατηρούμε ότι έχει ανέβει αρκετά γεγονός που οφείλεται στη συμφόρηση που έχει προκληθεί από τον νέο φόρτο.

- **Workload δικτύου 300txn/sec**
  Στην περίπτωση αυτή το δίκτυο PoW φαίνεται ότι έχει υποστεί σοβαρή συμφόρηση. Το throughput εμφανίζεται σημαντικά μειωμένο σε σχέση με τις δύο προηγούμενες περιπτώσεις και το latency του δικτύου είναι σημαντικά υψηλότερο και από τις δύο προηγούμενες περιπτώσεις. Ένα τέτοιο δίκτυο με αυτό τον αριθμό κόμβων είναι αδύνατο να ανταπεξέλθει σε ένα ένα τέτοιο φόρτο εργασίας.

Ο PoW είναι ένας σχετικά αργός και απαιτητικός σε πόρους αλγόριθμος συναίνεσης. Απαιτεί από τους κόμβους να επιλύουν πολύπλοκους μαθηματικούς γρίφους, οι οποίοι μπορεί να απαιτούν σημαντική υπολογιστική ισχύ και χρόνο. Όσον αφορά το throughput, το δίκτυο PoW του Ethereum υποστηριζε μέχρι να πραγματοποιηθεί η μετάβαση 15-20 συναλλαγές ανά δευτερόλεπτο στο δημόσιο δίκτυο. Σε ένα περιβάλλον ιδιωτικού δικτύου με 6 κόμβους και βελτιστοποιημένες ρυθμίσεις βλέπουμεμε ότι ο αριθμός αυτός θα είναι ελαφρώς υψηλότερος, αλλά φαίνεται απίθανο να μπορέσει να διαχειριστεί σημαντικά μεγαλύτερο αριθμό συναλλαγών.

### 4.1.2 Αξιολόγηση επίδοσης δικτύου με 12 κόμβους

Στις παρακάτω γραφικές παραστάσεις φαίνεται η απόδοση του δικτύου Ethereum με 12 κόμβους και τους πέντε διαφορετικούς φόρτους εργασίας που του επιβάλλαμε.




(a) Throughput

(b) Latency

Εικόνα 4.2: *Πειραματισμός σε Δίκτυο PoW 12 κόμβων*

- **Workload δικτύου 36txn/sec**
  Για αυτό το σενάριο χαμηλής κίνησης παρατηρούμε μια σχετικά καλή απόδοση του δικτύου από άποψη throughput και latency. Πιο συγκεκριμένα, όσον αφορά το throughput παρατηρούμε ότι το δίκτυο μπορεί να πετυχει περίπου 22 συναλλαγές ανά δευτερόλεπτο. Σχετικά με το latency παρατηρούμε ότι είναι αρκετά χαμηλό τουλάχιστον σε σχέση με αυτό που παρατηρείται στο δημόσιο δίκτυο του Ethereum με Proof of Work ως αλγόριθμο συναίνεσης.

- **Workload δικτύου 120txn/sec**
  Το δίκτυο παρατηρούμε πως έχει αρχίσει να δυσκολέυεται με αυτό το φορτίο. Η αύξηση του αριθμού των κόμβων εχει προσφερει μια σχετική βελτίωση σε σχεση με το προηγούμενο πείραμα ωστόσο το δίκτυο περισσότερων κόμβων έχει μια ελαφρώς καλύτερη εικόνα σε σχέση με αυτή του δικτύου των 6 κόμβων

- **Workload δικτύου 300txn/sec**
  Στην περίπτωση αυτή παρατηρούμε επίσης σημαντική επιβάρυνση του δικτύου. Υπάρχει μεγάλη συμφόρηση στο δίκτυο γεγονος που οδηγεί τόσο σε μείωση του throughput όσο και σε σημαντική αύξηση του latency.

- **Workload δικτύου 600txn/sec**
  Το συγκεκριμένο φορτίο βλέπουμε οτι δεν μπορεί να διαχειριστεί αποτελεσματικά από το δίκτυο, γεγονός που οδηγεί σε εξαιρετικά χαμηλή απόδοση και πολύ υψηλή καθυστέρηση.

- **Workload δικτύου 1200txn/sec**
  Το δίκτο πλέον είναι ασύμφορο. Είναι γεγονός ότι ο PoW δεν μπορει να διαχειριστεί αποτελεσματικά αυτούς τους όγκους, με αποτέλεσμα σημαντικά μειωμένο throughput και σημαντικά αυξημένο latency.

### 4.1.3 Παρατηρήσεις για τα Δίκτυα με μηχανισμό συναίνσης PoW

**Scalability-Επεκτασιμότητα στους κόμβους δικτύου** Παρακάτω παρατίθονται κάποιες παρατηρήσεις στα πλαίσια της επεκτασιμότητας του αλγορίθμου όσον αφορά τον αριθμό των κόμβων.

- **Throughput** Με τη συμμετοχή περισσότερων κόμβων στο δίκτυο, αυξάνεται η συνολική υπολογιστική ισχύς του δικτύου. Αυτό θα μπορούσε ενδεχομένως να βελτιώσει την ταχύτητα επικύρωσης συναλλαγών και τη δυνατότητα προσθήκης νέων block στην αλυσίδα. Ωστόσο, στο PoW, η επικύρωση συναλλαγών δεν εκτελείται παράλληλα σε όλους τους κόμβους. Ως εκ τούτου, η αύξηση του αριθμού των κόμβων δεν αυξάνει γραμμικά την απόδοση. Ο διπλασιασμός του αριθμού των κόμβων σε ένα δίκτυο PoW παρατηρούμε οτι μπορεί να βοηθήσει στην ελαφρά αύξηση της χωρητικότητας του δικτύου με την ευρύτερη κατανομή της ισχύος εξόρυξης. Ωστόσο, η βελτίωση δεν είναι σημαντική λόγω των εγγενών περιορισμών του ενΠοΩ.

- **Latency** Συγκριτικά στα δύο είδη πειραμάτων αυτο που παρατηρούμε είναι οτι η καθυστέρηση μπορεί να μειωθεί ελαφρώς σε ένα δίκτυο με περισσότερους κόμβους. Αυτό πιθανότατα συμβαίνει καθώς η ύπαρξη περισσότερων miners αυξάνει τις πιθανότητες γρήγορης επίλυσης του πολύπλοκου μαθηματικού προβλήματος που απαιτείται για την επικύρωση ενός block με αποτέσμα μείωση της καθυστέρησης.

**Scalability-Επεκτασιμότητα στο φόρτο εργασίας** Παρακάτω παρατίθονται κάποιες παρατηρήσεις στα πλαίσια της επεκτασιμότητας του αλγορίθμου όσον αφορά το φόρτο εργασίας που επιβάλλεται στο δίκτυο.

- **Throughput** Καθώς αυξάνεται ο φόρτος εργασίας, πρέπει να διεκπεραιώνονται περισσότερες συναλλαγές σε ένα δεδομένο χρονικό διάστημα. Ωστόσο, τα δίκτυα PoW έχουν περιορισμένη ικανότητα επεξεργασίας συναλλαγών λόγω της χρονοβόρας φύσης της εξόρυξης. Το δίκτυο μπορεί να διαχειριστεί έναν ορισμένο αριθμό συναλλαγών ανά δευτερόλεπτο με βάση παράγοντες όπως το μέγεθος μπλοκ, ο χρόνος μπλοκ και η δυσκολία εξόρυξης. Όταν ο φόρτος εργασίας υπερβαίνει τη χωρητικότητα του δικτύου, το ανεκτέλεστο υπόλοιπο των εκκρεμών συναλλαγών αυξάνεται. Οι miners χρειάζονται περισσότερο χρόνο για να λύσουν πολύπλοκους γρίφους, με αποτέλεσμα να επιβραδύνεται ο ρυθμός δημιουργίας μπλοκ. Ως αποτέλεσμα, η απόδοση μειώνεται και λιγότερες

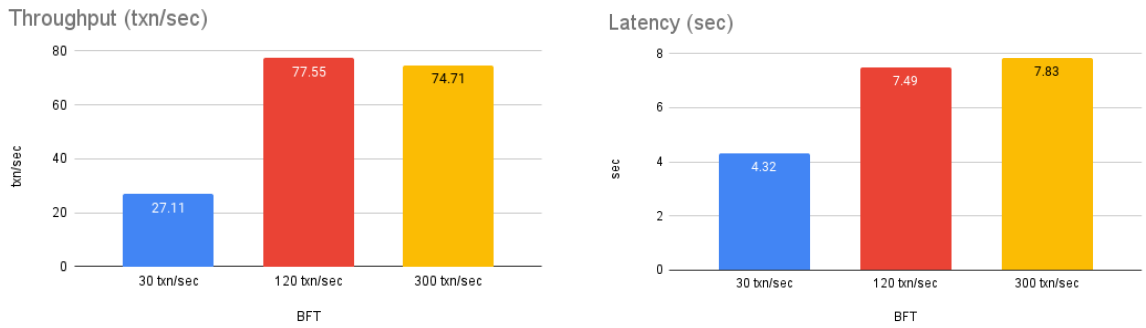
συναλλαγές μπορούν να επιβεβαιωθούν εντός ενός συγκεκριμένου χρονικού πλαισίου. Αυτή η συμφόρηση μπορεί να προκαλέσει καθυστερήσεις στην επεξεργασία των συναλλαγών και να επηρεάσει τη συνολική απόδοση του δικτύου.

- **Latency** Ο υψηλότερος φόρτος εργασίας μπορεί να αυξήσει τους χρόνους επιβεβαίωσης των συναλλαγών, με αποτέλεσμα την αύξηση της καθυστέρησης. Όταν υπάρχουν περισσότερες εκκρεμείς συναλλαγές, οι miners έχουν μεγαλύτερη δεξαμενή για να επιλέξουν κατά τη συναρμολόγηση block. Οι miners ενδέχεται επίσης να χρειάζονται περισσότερο χρόνο για να βρουν μια έγκυρη λύση στο παζλ εξόρυξης λόγω του αυξημένου ανταγωνισμού.Καθώς το δίκτυο γίνεται πιο συμφορημένο, ο μέσος χρόνος που απαιτείται για την επιβεβαίωση και την προσθήκη μιας συναλλαγής στην αλυσίδα αυξάνεται. Η καθυστέρηση μπορεί να γίνει ιδιαίτερα αισθητή όταν ο φόρτος εργασίας ξεπερνά την ικανότητα επεξεργασίας του δικτύο οπως φαίνεται και από τα παραπάνω πειράματα

## 4.2 Αξιολόγηση Επίδοσης Go-Ethereum με NCCU BFT

### 4.2.1 Αξιολόγηση επίδοσης δικτύου με 6 κόμβους

Στις παρακάτω γραφικές παραστάσεις φαίνεται η απόδοση του δικτύου Ethereum με 6 κόμβους και τους τρεις διαφορετικούς φόρτους εργασίας που του επιβάλλαμε.




(a) Throughput (b) Latency

Εικόνα 4.3: *Πειραματισμός σε Δίκτυο NCCU BFT 6 κόμβων*

Ο BFT είναι ένας ταχύτερος και λιγότερο απαιτητικός σε πόρους αλγόριθμος συναίνεσης από τον PoW με υψηλή απόδοση συναλλαγών και χαμηλή καθυστέρηση. Ωστόσο, δεν κλιμακώνεται καλά με τον αριθμό των validators λόγω της υψηλής πολυπλοκότητας της επικοινωνίας.

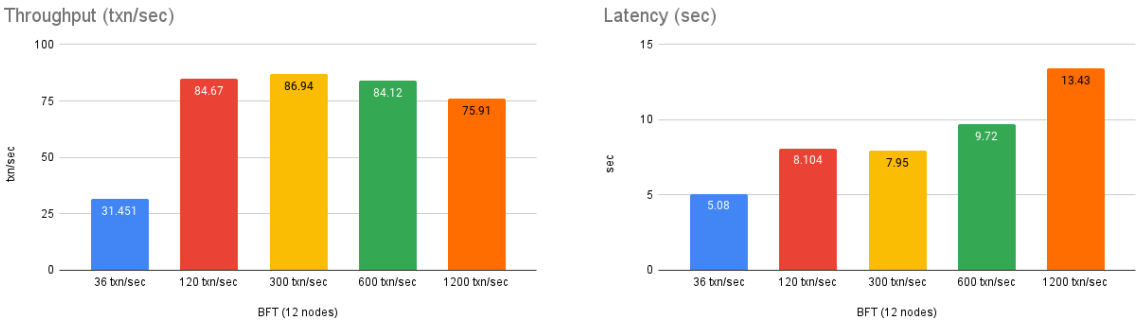
- **Workload δικτύου 30txn/sec**
  Λόγω της αποδοτικής φύσης του αλγορίθμου παρατηρούμε ότι είναι σε θέση να διαχειριστεί αυτό το φορτίο αρκετά εύκολα. Δηλαδή παρατηρούμε ότι το througput είναι κοντα στο 30, ενώ έχει ένα αρκετά χαμηλό latency το οποίο οφείλεται πιθανότατα στο μικρό αριθμό των κόμβων.

- **Workload δικτύου 120txn/sec**
  Το δίκτυο αυτό παρατηρούμε ότί εξακολουθεί να αποδίδει καλά υπό αυτό το φορτίο, διατηρώντας υψηλό throughput και χαμηλό latency. Ειδικότερα όσον αφορά το throughput φαίνεται να ανεβαίνει κοντα στις 50 συναλλαγές το δευτερόλεπτο ενω το latency ανεβαίνει λίγο χωρίς ωστόσο να γίνεται μη αποδοτικό.

- **Workload δικτύου 300txn/sec**
  Ακόμη και ένας αποδοτικός αλγόριθμος τύπου BFT παρατηρούμε ότι έχει κάποιο ανώτατο όριο δειαχείρισης συναλλαγών. Όταν αρχίζει να αυξάνεται σημαντικά το φορτίο φαίνεται ότι δεν μπορεί να διαχειριστεί πολλές επιπλέον συναλλγές ωστόσο σίγουρα μπορεί να διατηρήσει την γενικότερη απόδοση του.

### 4.2.2 Αξιολόγηση επίδοσης δικτύου με 12 κόμβους

Στις παρακάτω γραφικές παραστάσεις φαίνεται η απόδοση του δικτύου Ethereum με 12 κόμβους και τους πέντε διαφορετικούς φόρτους εργασίας που του επιβάλλαμε.




(a) Throughput (b) Latency

Εικόνα 4.4: *Πειραματισμός σε Δίκτυο NCCU BFT 12 κόμβων*

- **Workload δικτύου 36txn/sec**
  Το συγκεκριμενο φορτίο παρατηρούμε ότι είναι εύκολα διαχειρίσιμο για το δίκτυο Ethereum με NCCU BFT. Όπως περιμέναμε λοιπόν το δίκτυο αποδίδει καλά. Με αυξημένο throughput και αρκετά χαμηλό latency.

- **Workload δικτύου 120txn/sec**
  Το δίκτυο μπορεί και στην συγκεκριμένη περίπτωση με σχετική ευκολία να διατηρεί σχετικά υψηλό throughput και χαμηλό latency. Ο συγκεκριμένος φόρτος εργασίας δεν είναι ακόμη ικανός να επηρεάσει σημαντικά το δίκτυο.

- **Workload δικτύου 300txn/sec**
  Στο φορτίο αυτό παρατηρούμε μια σχετικά μικρή βελτίωση στις μετρικές που λάβαμε, χωρίς ωστόσο να υπάρχει σημαντική διαφορά από το προηγούμενο πείραμα.

- **Workload δικτύου 600txn/sec**
  Στη συγκεκριμένο φορτίο παρατηρούμε ότι μάλλον το δίκτυο έχει φτάσει στο ανωτατο

του όριο διαχείρησης συναλλαγών. Παρατηρούμε δηλαδή ότι συντηρεί την απόδοση του προηγούμενου πειράματος.

- **Workload δικτύου 1200txn/sec**
  Ο όγκος εργασίας αυτός παρατηρούμε ότι έχει χειροτερέψει την γενική απόδοση του δικτύου ωστόσο όχι σε καταστροφικό επίπεδο. Πιθανότατα ο συνδυασμός αυτού του φόρτου με την πολυπλοκότητα της επικοινωνιας με περισσότερους κόμβους οδηγούν σε μια σχετική πτώση της γενικής απόδοσης του δικτύου.

### 4.2.3 Παρατηρήσεις για τα Δίκτυα με μηχανισμό συναίνσης NCCU BFT

**Scalability-Επεκτασιμότητα στους κόμβους δικτύου** Παρακάτω παρατίθονται κάποιες παρατηρήσεις στα πλαίσια της επεκτασιμότητας του αλγορίθμου όσον αφορά τον αριθμό των κόμβων.
Στο BFT, η αύξηση των κόμβων μπορεί να επηρεάσει την απόδοση με δύο τρόπους. Από τη μία πλευρά, η ύπαρξη περισσότερων κόμβων στο δίκτυο σημαίνει ότι υπάρχει μεγαλύτερος βαθμός ανοχής σφαλμάτων. Εάν ένας κόμβος αποτύχει ή ενεργήσει κακόβουλα, υπάρχουν περισσότεροι κόμβοι για να πάρουν τη θέση του, ενισχύοντας την ανθεκτικότητα του δικτύου. Από την άλλη πλευρά, το BFT έχει πολυπλοκότητα επικοινωνίας $O(n^2)$, όπου n είναι ο αριθμός των κόμβων. Καθώς το δίκτυο μεγαλώνει, ο αριθμός των μηνυμάτων που χρειάζονται οι κόμβοι για να επιτύχουν συναίνεση αυξάνεται τετραγωνικά, γεγονός που μπορεί να οδηγήσει σε μεγαλύτερη συμφόρηση του δικτύου και μεγαλύτερους χρόνους συναίνεσης, αυξάνοντας έτσι την καθυστέρηση και μειώνοντας την απόδοση. Ωστόσο, δεδομένου του σεναρίου μας με 6 έως 12 κόμβους, ο αντίκτυπος αυτός δεν είναι πολύ σημαντικός, καθώς το μέγεθος του δικτύου εξακολουθεί να είναι σχετικά μικρό.

**Scalability-Επεκτασιμότητα στο φόρτο εργασίας** Παρακάτω παρατίθονται κάποιες παρατηρήσεις στα πλαίσια της επεκτασιμότητας του αλγορίθμου όσον αφορά το φόρτο εργασίας που επιβάλλεται στο δίκτυο.

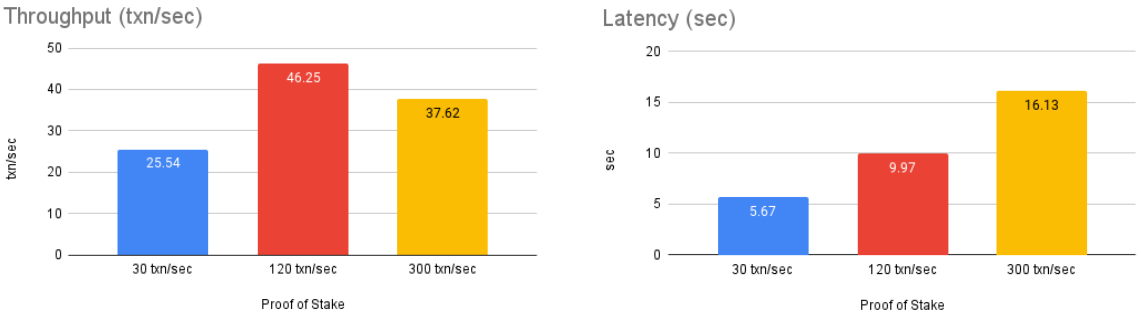
Το NCCU BFT παρουσιάζει ισχυρή κλιμάκωση όταν πρόκειται να χειριστεί αυξημένο φόρτο εργασίας, ιδίως σε μικρότερα δίκτυα. Μπορεί να επεξεργαστεί γρήγορα μεγάλο αριθμό συναλλαγών λόγω του αποτελεσματικού μηχανισμού ψηφοφορίας. Εννοείται ότι με ισχυρά μεγάλο φόρτο εργασίας χάνει την αποδοτικότητα του πιθανότατα λόγω του bandwidth του δικτύου. Ωστόσο ενώ το NCCU BFT αποδίδει καλά υπό υψηλό φόρτο εργασίας σε μικρά δίκτυα, δυσκολεύεται να διατηρήσει αυτή την απόδοση σε μεγαλύτερα δίκτυα.

## 4.3 Αξιολόγηση Επίδοσης Go-Ethereum με Proof-of-Stake

### 4.3.1 Αξιολόγηση επίδοσης δικτύου με 6 κόμβους

Στις παρακάτω γραφικές παραστάσεις φαίνεται η απόδοση του δικτύου Ethereum με 6 κόμβους και τους τρεις διαφορετικούς φόρτους εργασίας που του επιβάλλαμε.

Το PoS του Ethereum, γνωστό ως Ethereum 2.0, έχει σχεδιαστεί για να είναι ταχύτερο και πιο κλιμακούμενο από τον προκάτοχό του PoW. Οι validators επιλέγονται τυχαία για να




(a) Throughput (b) Latency

Εικόνα 4.5: *Πειραματισμός σε Δίκτυο PoS 6 κόμβων*

προτείνουν μπλοκ, γεγονός που χρησιμοποιεί πολύ λιγότερη υπολογιστική ισχύ και συνεπώς αυξάνει τη συνολική αποδοτικότητα.

- **Workload δικτύου 30txn/sec**
  Αυτό το δίκτυο παρατηρούμε ότι χειρίζεται επίσης αρκετά καλά. Δηλαδή παρατηρούμε ότι έχει επίσης πολύ υψηλό throughput κοντά στο 26 ενώ παράλληλα έχει ένα αρκετά χαμηλό latency.

- **Workload δικτύου 120txn/sec**
  Το δίκτυο PoS έχει αρχίσει να παρουσιάζει κάποια επιβάρυνση υπό αυτό το φορτίο. Δηλαδή παρατηρούμε αύξηση του throughput ωστόσο και αύξηση του latency.

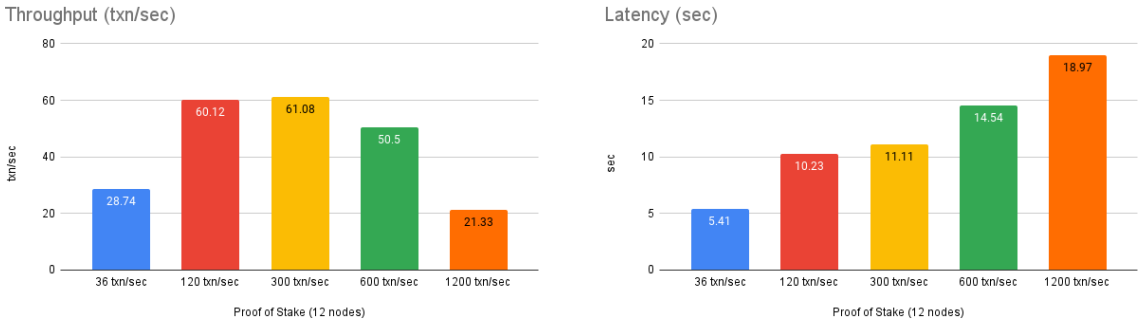
- **Workload δικτύου 300txn/sec**
  Κάτω από ένα τόσο σημαντικό φορτίο, σε συνάρτηση πάντα με τον αριθμό των κόμβων του δικτύου μας, παρατηρούμε ότι και το PoS έχει αρχίσει να δυσκολεύεται αρκετά. Παρατηρείται μείωση στο throughput καθώς και αύξηση στο latency, τα οποία πιθανώς οφείλονται στον αυξημένο ανταγωνισμό μεταξύ των επικυρωτών, τη ιεράρχηση των συναλλαγών και την ανάγκη για πρόσθετη επικύρωση.

### 4.3.2 Αξιολόγηση επίδοσης δικτύου με 12 κόμβους

Στις παρακάτω γραφικές παραστάσεις φαίνεται η απόδοση του δικτύου Ethereum με 12 κόμβους και τους πέντε διαφορετικούς φόρτους εργασίας που του επιβάλλαμε.

- **Workload δικτύου 36txn/sec**
  Το δίκτυο PoS διαπιστώνουμε ότι διαχειρίζεται καλά αυτό το φορτίο. Η ύπαρξη περισσότερων επικυρωτών βελτιώσει ελαφρώς τις επιδόσεις του δικτύου, οδηγώντας σε υψηλότερο throughout και χαμηλότερο latency σε σχέση με ένα μικρότερο δίκτυο.

- **Workload δικτύου 120txn/sec**
  Στη περίπτωση αυτού του φορτίου το δίκτυο συνεχίζει να ανταπεξέρχεται με επιτυχία. Προφανώς δεν πετυχαίνει το 100% των συναλλαγών ωστόσο το throughput αυξήθηκε ενώ το latency μπορει να αυξήθηκε αυτό αλλά σε επιτρεπτά επίπεδα.




(a) Throughput (b) Latency

Εικόνα 4.6: *Πειραματισμός σε Δίκτυο PoS 12 κόμβων*

- **Workload δικτύου 300txn/sec**
  Κάτω από αυτό το φόρτο εργασίας παρουσιάζει μια πολύ μικρή βελτίωση αλλά μάλλον πλησιάζει στα όρια του. Δηλαδή υπάρχει μια μικρή του throughput ενώ το latency παραμένει στα ίδια επίπεδα.

- **Workload δικτύου 600txn/sec**
  Το δίκτυο στην περίπτωση αυτή αρχίζει να παρουσιάζει καθοδική πορεία. Παρουσιάζεται επιδείνωση των throughput και latency.

- **Workload δικτύου 1200txn/sec**
  Παρά τη δυνητική βελτίωση από την ύπαρξη περισσότερων επικυρωτών, η απόδοση και στην περίπτωση αυτή είναι πολύ χαμηλότερη από τον στόχο και η καθυστέρηση σημαντικά υψηλότερη από τα προηγούμενα πειράματα.

### 4.3.3 Παρατηρήσεις για τα Δίκτυα με μηχανισμό συναίνσης PoS

**Scalability-Επεκτασιμότητα στους κόμβους δικτύου** Παρακάτω παρατίθονται κάποιες παρατηρήσεις στα πλαίσια της επεκτασιμότητας του αλγορίθμου όσον αφορά τον αριθμό των κόμβων.

- **Throughput** Η ύπαρξη περισσότερων επικυρωτών σε ένα δίκτυο PoS θα μπορούσε ενδεχομένως να βελτιώσει την απόδοση του δικτύου αυξάνοντας την ικανότητα πρότασης block και επικύρωσης. Ωστόσο, η αύξηση από ότι φαίνεται από τα πειράματα δεν είναι αρκετά σημαντιη.

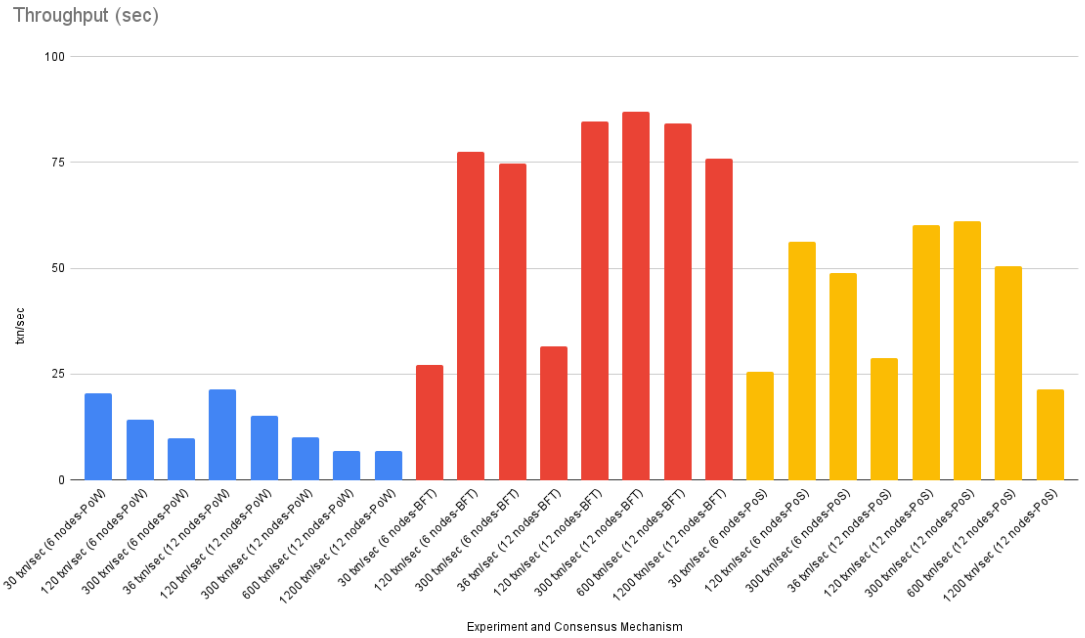
- **Latency** Με περισσότερους επικυρωτές στο δίκτυο, αυξάνονται οι πιθανότητες να επιλεγεί κάποιος για τη δημιουργία ενός νέου block, γεγονός που μπορεί ενδεχομένως να επιταχύνει τη διαδικασία δημιουργίας block και να μειώσει την καθυστέρηση. Ωστόσο, η βελτίωση οπως φαίνεται δεν ειναι ιδιαίτερα σημαντική.

**Scalability-Επεκτασιμότητα στο φόρτο εργασίας** Παρακάτω παρατίθονται κάποιες παρατηρήσεις στα πλαίσια της επεκτασιμότητας του αλγορίθμου όσον αφορά το φόρτο εργασίας που επιβάλλεται στο δίκτυο.

Το PoS επιδεικνύει καλύτερη επεκτασιμότητα όσον αφορά το φόρτο εργασίας. Αντιμετωπίζει καλύτερα τα αυξημένα φορτία συναλλαγών επειδή αποφεύγει την υπολογιστικά δαπανηρή διαδικασία εξόρυξης. Ωστόσο, καθώς αυξάνεται ο αριθμός των συναλλαγών, εξακολουθούν να υπάρχουν σημεία συμφόρησης. Για παράδειγμα, η διαδικασία επιλογής των επικυρωτών και η διαχείριση των στοιχημάτων γίνεται πιο πολύπλοκη με περισσότερες συναλλαγές. Παρόλο που το PoS μπορεί να διαχειριστεί έναν υψηλότερο φόρτο εργασίας εξακολουθεί να αντιμετωπίζει προκλήσεις σε αυξημένους φόρτους εργασίας.

## 4.4 Συγκριτική Αξιολόγηση των τριών δικτύων

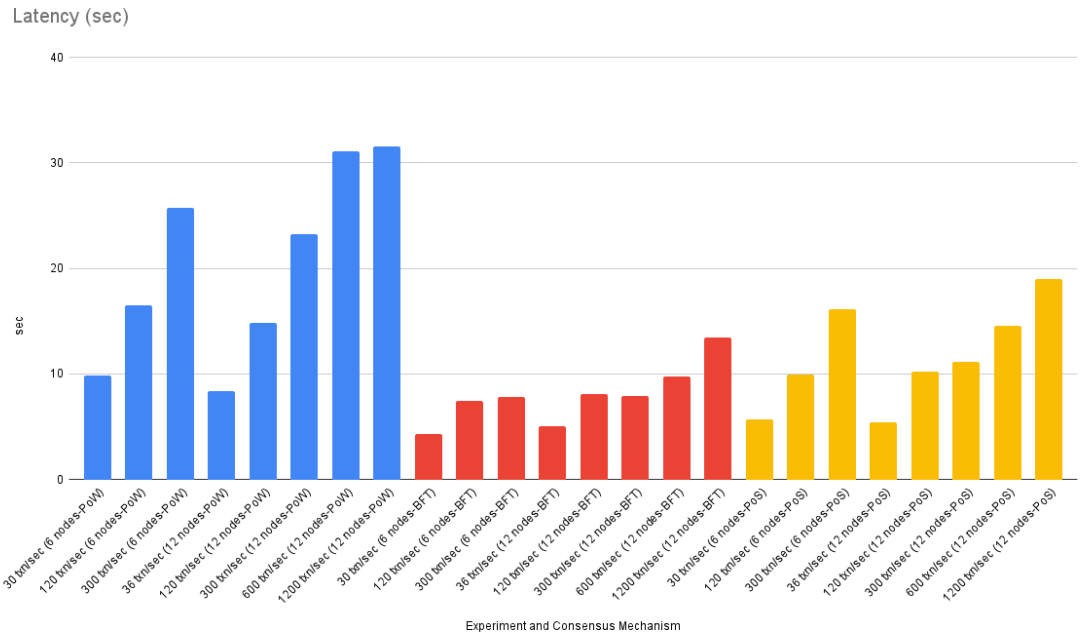
Παρακάτω ακολουθούν δύο συγκεντρωτικές γραφικές παραστάσεις για όλα τα δίκτυα.




Εικόνα 4.7: *Συγκεντρωτικά αποτελέσματα Throughput για όλα τα δίκτυα*

Κατά την εξέταση του **throughput**, το PoW παρουσιάζει γενικά τη χαμηλότερη απόδοση λόγω της διαδικασίας εξόρυξης. Η υπολογιστική πολυπλοκότητα περιορίζει τον αριθμό των συναλλαγών που μπορούν να υποβληθούν σε επεξεργασία ανά δευτερόλεπτο. Από την άλλη πλευρά, το PoS διευκολύνει υψηλότερη απόδοση από το PoW, καθώς δεν εμπλέκει τους εξορύκτες στην επίλυση πολύπλοκων μαθηματικών προβλημάτων. Ωστόσο, σε σχέση με το NCCU BFT εξακολουθεί να υστερεί πιθανότατα λόγω και της διαδικασίας επιλογής επικυρωτών αλλά και διάφορων ελέγχων που γίνονται κατά τη διαδικασία του staking. Το NCCU BFT γενικά υπερτερεί των άλλων δύο όσον αφορά την απόδοση, ιδίως σε μικρότερα δίκτυα. Ωστόσο, καθώς το δίκτυο επεκτείνεται, η απόδοση θα μπορούσε να μειωθεί λόγω της αυξημένης πολυπλοκότητας στη διαδικασία ψηφοφορίας.

Όσον αφορά το **latency**, το Proof of Work υποφέρει περισσότερο λόγω της διαδικασίας εξόρυξης , η οποία απαιτεί την επίλυση πολύπλοκων μαθηματικών προβλημάτων και, ως εκ τούτου, παρατείνει το χρόνο επεξεργασίας των συναλλαγών. Αντίθετα, το PoS συγκριτικά με




Εικόνα 4.8: *Συγκεντρωτικά αποτελέσματα Latency για όλα τα δίκτυα*

το PoW έχει χαμηλότερη καθυστέρηση, καθώς παρακάμπτει την ανάγκη για εντατική υπολογιστική εργασία. Το NCCU BFT προσφέρει τη χαμηλότερη καθυστέρηση μεταξύ των τριών. Αυτό οφείλεται κυρίως στο γεγονός ότι αποφεύγει την ανάγκη εξόρυξης ή στοιχηματισμού, με τις συναλλαγές να επικυρώνονται μέσω μιας διαδικασίας ψηφοφορίας μεταξύ των κόμβων, η οποία μπορεί να εκτελεστεί σχετικά γρήγορα, δεδομένου ότι ο αριθμός των κόμβων δεν είναι υπερβολικά μεγάλος. Ωστόσο παρατηρούμε ότι στις περισσότερες των περιπτώσεων που μελετήσαμε το PoS δεν υστερεί πολύ συγκριτικά με το NCCU BFT οπότε θα έλεγε κανείς ότι οι δύο αυτοί αλγόριθμοι ειναι και οι δύο αρκετά αποδοτικοί στις συγκεκριμένες συνθήκες.

Σχετικά με το **scalability των κόμβων**, το PoW αντιμετωπίζει σημαντικές προκλήσεις. Καθώς το δίκτυο επεκτείνεται, η δυσκολία εξόρυξης αυξάνεται, γεγονός που επιδεινώνεται περαιτέρω από την υψηλή κατανάλωση ενέργειας του αλγορίθμου, καθιστώντας το λιγότερο κατάλληλο για εφαρμογές μεγάλης κλίμακας. Το Proof of Stake είναι συνήθως πιο επεκτάσιμο από το PoW, καθώς απαιτεί λιγότερη υπολογιστική ισχύ. Παρόλα αυτά, καθώς αυξάνεται το μέγεθος του δικτύου, η διαδικασία επιλογής επικυρωτή μπορεί να γίνει πιο περίπλοκη, θέτοντας ενδεχομένως περιορισμούς στην επεκτασιμότητά του. Τέλος το BFT προσφέρει αποτελεσματική επεκτασιμότητα μέχρι ένα ορισμένο όριο. Αν και χειρίζεται αποτελεσματικά μικρά έως μεσαίου μεγέθους δίκτυα, η τετραγωνική αύξηση της πολυπλοκότητας της διαδικασίας ψηφοφορίας με τον αριθμό των κόμβων μπορεί να περιορίσει την απόδοσή του σε πολύ μεγάλα δίκτυα.

Τέλος το **scalability του φόρτου εργασίας** ενώ το PoS και το NCCU BFT επιδεικνύουν γενικά καλύτερη κλιμάκωση σε σύγκριση με το PoW, καθένα από αυτά μπορεί να αντιμετωπίσει πιθανές προκλήσεις καθώς αυξάνεται ο φόρτος εργασίας. Το PoS μπορεί να αντιμετωπίσει πολυπλοκότητες στη διαχείριση των διακυβευμάτων και στην επιλογή των επικυρωτών, ενώ το BFT φαίνεται να έχει την καλύτερη απόδοση. Εν τω μεταξύ, το δίκτυο PoW

είναι σαφές ότι δυσκολεύεται περισσότερο με υψηλό φόρτο εργασίας λόγω της υπολογιστικά δαπανηρής διαδικασίας εξόρυξης.

Συνοψίζοντας, οι PoW, NCCU BFT και PoS παρουσιάζουν το καθένα τα μοναδικά του πλεονεκτήματα και αδυναμίες. Ενώ το BFT παρέχει γενικά την καλύτερη καθυστέρηση και απόδοση, η επεκτασιμότητά του μπορεί να παραπαίει σε πολύ μεγάλα δίκτυα. Το PoS προσφέρει συνήθως καλύτερη γενική απόδοση από το PoW και παρουσιάζει μεγαλύτερη κλιμάκωση, αλλά σίογουρα δεν φαίνεται να είναι τόσο αποτελεσματικός όσο ο BFT σε μικρότερα δίκτυα. Παρόλο που το Proof of Work όπως φαίνεται και από τα πειράματα είναι λιγότερο αποτελεσματικό από τους άλλους δύο αλγορίθμους αξίζει να σημειωθεί πως έχει γνωρίσει ευρεία υιοθέτηση λόγω της στιβαρότητας και της ασφάλειάς που προσφέρει.

# Κεφάλαιο 5

# Μελλοντικές Επεκτάσεις

Καθώς ο τομέας της τεχνολογίας blockchain συνεχίζει να εξελίσσεται ραγδαία, υπάρχουν πολλές προοπτικές για μελλοντική έρευνα και εξερεύνηση στη συγκριτική αξιολόγηση των αλγορίθμων συναίνεσης στα ιδιωτικά δίκτυα Ethereum. Στο παρόν κεφάλαιο παρουσιάζονται πιθανές επεκτάσεις και τομείς εστίασης που μπορούν να επιδιωχθούν για την περαιτέρω βελτίωση της κατανόησης και της αξιολόγησης των αλγορίθμων συναίνεσης σε ιδιωτικά περιβάλλοντα blockchain.

## 5.1 Αξιολόγηση υβριδικών αλγορίθμων συναίνεσης

Μια πολλά υποσχόμενη κατεύθυνση για μελλοντική έρευνα είναι η αξιολόγηση υβριδικών αλγορίθμων συναίνεσης που συνδυάζουν τα πλεονεκτήματα διαφορετικών προσεγγίσεων. Για παράδειγμα, η διερεύνηση των επιδόσεων και των χαρακτηριστικών των αλγορίθμων συναίνεσης που ενσωματώνουν μηχανισμούς Proof of Stake και Proof of Work θα μπορούσε να παράσχει πολύτιμες πληροφορίες σχετικά με τη συνδυασμένη αποδοτικότητα, την ασφάλεια και την επεκτασιμότητά τους. Αυτή η επέκταση θα περιλάμβανε τον σχεδιασμό πειραμάτων και μεθοδολογιών συγκριτικής αξιολόγησης ειδικά προσαρμοσμένων για την αξιολόγηση των επιδόσεων αυτών των υβριδικών μοντέλων.

## 5.2 Ανάλυση προηγμένων πρωτοκόλλων συναίνεσης

Ένας άλλος τομέας ενδιαφέροντος είναι η αξιολόγηση προηγμένων πρωτοκόλλων συναίνεσης πέρα από τα παραδοσιακά PoW, PoS και BFT. Μηχανισμοί συναίνεσης, όπως οι Delegated Proof of Stake και Proof of Authority, θα μπορούσαν να διερευνηθούν για να αξιολογηθούν τα χαρακτηριστικά των επιδόσεών τους σε ιδιωτικά δίκτυα blockchain.

## 5.3 Αξιολόγηση των τεχνικών ενίσχυσης της ιδιωτικότητας

Η προστασία της ιδιωτικότητας αποτελεί κρίσιμο ζήτημα στα συστήματα blockchain, ιδίως στα ιδιωτικά δίκτυα. Οι μελλοντικές επεκτάσεις της παρούσας διπλωματικής εργασίας θα μπορούσαν να επικεντρωθούν στην αξιολόγηση της απόδοσης των αλγορίθμων συναίνεσης σε

συνδυασμό με τεχνικές ενίσχυσης της ιδιωτικότητας, όπως αποδείξεις μηδενικής γνώσης (zero-knowledge proofs), δακτυλιοειδείς υπογραφές (ring signatures) ή εμπιστευτικές συναλλαγές (confidential transactions). Η αξιολόγηση του αντίκτυπου αυτών των τεχνικών στη συνολική απόδοση, την απόδοση και την καθυστέρηση των ιδιωτικών δικτύων θα παρείχε πολύτιμες γνώσεις για τους οργανισμούς που επιδιώκουν να αναπτύξουν ασφαλείς και διατηρώντας την ιδιωτικότητα λύσεις blockchain.

## 5.4 Ανάλυση επεκτασιμότητας για δίκτυα μεγάλης κλίμακας

Καθώς τα δίκτυα blockchain συνεχίζουν να αυξάνονται σε κλίμακα και μέγεθος, η αξιολόγηση της επεκτασιμότητας των αλγορίθμων συναίνεσης γίνεται όλο και πιο σημαντική. Μελλοντικές επεκτάσεις της παρούσας διπλωματικής εργασίας θα μπορούσαν να περιλαμβάνουν τη διεξαγωγή ανάλυσης επεκτασιμότητας σε ιδιωτικά δίκτυα Ετηερευμ με σημαντικά μεγαλύτερο αριθμό κόμβων. Η αξιολόγηση των επιδόσεων, της απόδοσης και της καθυστέρησης των αλγορίθμων συναίνεσης σε σενάρια μεγάλης κλίμακας θα παρείχε σημαντικές πληροφορίες σχετικά με την ικανότητά τους να διαχειρίζονται τις αυξανόμενες απαιτήσεις του δικτύου και τον αυξανόμενο όγκο συναλλαγών.

## 5.5 Σύγκριση με άλλες πλατφόρμες blockchain

Η επέκταση του πεδίου της έρευνας ώστε να συμπεριλάβει και άλλες πλατφόρμες blockchain πέραν του Ethereum θα επέτρεπε τη συγκριτική ανάλυση των αλγορίθμων συναίνεσης. Η αξιολόγηση των επιδόσεων των μηχανισμών συναίνεσης σε δίκτυα που έχουν δημιουργηθεί σε εναλλακτικές πλατφόρμες blockchain, όπως η Hyperledger Fabric, η Corda, η EOS, θα μπορούσε να παράσχει χρήσιμα ευρήματα σχετικά με τα δυνατά και αδύνατα σημεία των διαφόρων πλατφορμών και αλγορίθμων συναίνεσης.

# Μέρος III

# Επίλογος

# Κεφάλαιο 6

# Επίλογος

Στην παρούσα διπλωματική εργασία, διερευνήσαμε και αξιολογήσαμε τρεις βασικούς αλγορίθμους συναίνεσης που είναι ενσωματωμένοι στο Ethereum: Proof of Work, Proof of Stake και Byzantine Fault Tolerance. Στόχος μας ήταν να αξιολογήσουμε και να συγκρίνουμε τις επιδόσεις, την καθυστέρηση, την απόδοση και τα χαρακτηριστικά κλιμάκωσης αυτών των αλγορίθμων συναίνεσης σε ιδιωτικά δίκτυα blockchain.

Μέσω μιας εις βάθος ανάλυσης, παρέχουμε μια ολοκληρωμένη κατανόηση της θεωρίας και των αρχών που διέπουν κάθε αλγόριθμο συναίνεσης. Συζητήσαμε τις τεχνικές ιδιαιτερότητες των PoW, PoS και BFT, τονίζοντας τα δυνατά και αδύνατα σημεία τους. Εξετάζοντας τους υποκείμενους μηχανισμούς και τις σχεδιαστικές επιλογές, αποκτήσαμε γνώσεις σχετικά με τη λειτουργία αυτών των αλγορίθμων συναίνεσης και την καταλληλότητά τους για διαφορετικές περιπτώσεις χρήσης blockchain.

Για να αξιολογήσουμε και να συγκρίνουμε την απόδοση αυτών των αλγορίθμων συναίνεσης, αξιοποιήσαμε το πλαίσιο Blockbench, ένα ισχυρό εργαλείο για την ανάλυση ιδιωτικών blockchain. Το Blockbench μας επέτρεψε να δημιουργήσουμε και να δοκιμάσουμε δίκτυα βασισμένα στο Ethereum, διευκολύνοντας ολοκληρωμένα πειράματα υπό ποικίλους φόρτους εργασίας. Χρησιμοποιώντας αυτό το πλαίσιο, διαμορφώσαμε σχολαστικά και εκτελέσαμε δοκιμές σε τρία διαφορετικά δίκτυα: PoW, PoS και BFT.

Η αξιολόγησή μας επικεντρώθηκε σε βασικές μετρήσεις επιδόσεων, όπως το latency, το throughput και το scalability στο consensus layer των τριών διαφορετικών δικτύων. Μετρήσαμε την ικανότητα των δικτύων να επεξεργάζονται αποτελεσματικά τις συναλλαγές, τους χρόνους απόκρισης για την επιβεβαίωση των συναλλαγών και τις επιδόσεις τους υπό αυξανόμενο φόρτο εργασίας. Μέσω αυστηρού πειραματισμού και ανάλυσης, αποκτήσαμε πολύτιμες γνώσεις σχετικά με τα χαρακτηριστικά απόδοσης κάθε αλγορίθμου συναίνεσης.

Τα αποτελέσματα της αξιολόγησής μας έδειξαν ότι ο PoW, αν και αποδεδειγμένα ασφαλής, παρουσίαζε περιορισμούς όσον αφορά την επεκτασιμότητα και την γενικότερη απόδοση. Το PoS, από την άλλη πλευρά, επέδειξε υποσχόμενα πλεονεκτήματα κλιμάκωσης και γενικότερης απόδοσης, καθιστώντας το μια βιώσιμη εναλλακτική λύση. Ο αλγόριθμος BFT, σχεδιασμένος για να ανέχονται βυζαντινές αποτυχίες, παρουσίασε πολύ καλή απόκριση, ωστόσο γνωρίζοντας τους περιορισμούς που παρουσιάζει στα πλαίσια της επεκτασιμότητας σε σημαντικά μεγάλα δίκτυα καθίσταται κατάλληλος για μικρού βελησνεκούς εφαρμογές που απαιτούν συναίνεση παρουσία κακόβουλων φορέων.

Αξιοποιώντας τις γνώσεις που αποκομίσαμε από τα πειράματα συγκριτικής αξιολόγησης, οι

οργανισμοί και οι προγραμματιστές μπορούν να λαμβάνουν τεκμηριωμένες αποφάσεις κατά την επιλογή του καταλληλότερου αλγορίθμου συναίνεσης για τις συγκεκριμένες απαιτήσεις τους. Είτε θέτουν ως προτεραιότητα την απόδοση ή την επεκτασιμότητα, η αξιολόγησή μας παρέχει πολύτιμη καθοδήγηση για την ευθυγράμμιση της επιλογής του αλγορίθμου συναίνεσης με τα επιθυμητά χαρακτηριστικά απόδοσης.

πιπλέον, η παρούσα διπλωματική άνοιξε το δρόμο για μελλοντική έρευνα και διερεύνηση στη συγκριτική αξιολόγηση αλγορίθμων συναίνεσης στοEthereum. Η μεθοδολογία αξιολόγησης που αναπτύξαμε μπορεί να επεκταθεί για να ενσωματώσει πρόσθετους μηχανισμούς συναίνεσης, υβριδικά μοντέλα και τεχνικές ενίσχυσης της ιδιωτικότητας.

Καθώς το οικοσύστημα blockchain συνεχίζει να εξελίσσεται, η ανάγκη για ακριβή συγκριτική αξιολόγηση και αξιολόγηση των αλγορίθμων συναίνεσης γίνεται όλο και πιο κρίσιμη. Η έρευνά μας συμβάλλει στον αυξανόμενο όγκο γνώσεων σε αυτόν τον τομέα, παρέχοντας μια ολοκληρωμένη ανάλυση των αλγορίθμων PoW, PoS και BFT που είναι ενσωματωμένοι στο Ethereum. Χρησιμεύει ως θεμέλιο για περαιτέρω έρευνες, διασφαλίζοντας τη συνεχή πρόοδο αποτελεσματικών και κλιμακούμενων λύσεων blockchain.

Εν κατακλείδι, η παρούσα διπλωματική εργασία έριξε φως στα χαρακτηριστικά απόδοσης των διαφόρων αλγορίθμων συναίνεσης στα ιδιωτικά δίκτυα του Ethereum. Μέσω της ενσωμάτωσης της θεωρητικής ανάλυσης, του πρακτικού πειραματισμού και της χρήσης του πλαισίου Blockbench, παρέχουμε μια διεξοδική αξιολόγηση των αλγορίθμων PoW, PoS και BFT. Με την κατανόηση των δυνατών σημείων, των αδυναμιών και των συμβιβασμών αυτών των μηχανισμών συναίνεσης, οι οργανισμοί μπορούν να λάβουν τεκμηριωμένες αποφάσεις για την αρχιτεκτονική ισχυρών και αποδοτικών λύσεων blockchain προσαρμοσμένων στις συγκεκριμένες ανάγκες τους.

Ελπίζουμε ότι η παρούσα διπλωματική εργασία συμβάλλει στο αυξανόμενο σώμα γνώσεων στον τομέα της συγκριτικής αξιολόγησης αλγορίθμων συναίνεσης στο Ethereum και χρησιμεύει ως πολύτιμη πηγή για ερευνητές, προγραμματιστές και επαγγελματίες του κλάδου που επιδιώκουν να περιηγηθούν στο πολύπλοκο τοπίο των μηχανισμών συναίνεσης blockchain. Με τη συνεχή έρευνα και τις εξελίξεις, είμαστε βέβαιοι ότι η τεχνολογία blockchain θα συνεχίσει να φέρνει επανάσταση στις βιομηχανίες και να διαμορφώνει το μέλλον των αποκεντρωμένων εφαρμογών και της ψηφιακής εμπιστοσύνης.

# Βιβλιογραφία


[1] Gang Chen Rui Liu Beng Chin Ooi Kian Lee Tan Tien Tuan Anh Dinh, Ji Wang. *BLOCKBENCH: A Framework for Analyzing Private Blockchains. SIGMOD '17: Proceedings of the 2017 ACM International Conference on Management of Data*, σελίδα 1085–1100, 2017.

[2] Miguel Castro και Barbara Liskov. *Practical Byzantine fault tolerance. OSDI '99: Proceedings of the third symposium on Operating systems design and implementation*, σελίδα 278, 1999.

[3] *Ethereum whitepaper*. `https://ethereum.org/en/whitepaper/`. 2023.

[4] *Ethash algorithm*. `https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/mining-algorithms/ethash/`. 2023.

[5] *Ethereum proof of stake*. `https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/`. 2023.

[6] *NCCU BFT consensus algorithm*. `https://github.com/jianwei20/FastBFT_ethereum`. 2023.

[7] *Go-Ethereum Documentation*. `https://geth.ethereum.org/`. 2023.

[8] *Go-Ethereum Official Implementation*. `https://github.com/ethereum/go-ethereum`. 2023.

[9] *Prysm Documentation*. `https://docs.prylabs.network/docs/getting-started`. 2023.

[10] *Prysm Official Implementation*. `https://github.com/prysmaticlabs/prysm`. 2023.

[11] *Running BLOCKBENCH for Ethereum*. `https://medium.com/@mu7eh7/running-blockbench-for-ethereum-6dca3ed44a35`. 2023.

[12] *Official blockbench Implementation*. `https://github.com/ooibc88/blockbench`. 2023.

[13] *Round Robin Selection*. `https://www.geeksforgeeks.org/relation-in-fcfs-and-round-robin-scheduling-algorithm/`. 2023.