Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων Και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων Ηλεκτρονικης Και Συστηματων πληροφορικης

Επιτήρηση Συμφωνιών Επιπέδου Υπηρεσίας με Έμμεσες Απεικονίσεις σε Δίκτυα Αλυσίδας-Κορμού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεωργία-Γρηγορία Ε. Ασημάκη

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2022

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Επιτήρηση Συμφωνιών Επιπέδου Υπηρεσίας με Έμμεσες Απεικονίσεις σε Δίκτυα Αλυσίδας-Κορμού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεωργία-Γρηγορία Ε. Ασημάκη

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 9η Ιουνίου 2022.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Συμεών Παπαβασιλείου	Εμμανουήλ Βαρβαρίγος
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2022

. . . . . .

Γεωργία-Γρηγορία Ε. Ασημάκη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Copyright © Γεωργία-Γρηγορία Ε. Ασημάκη, 2022.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.
Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η υιοθέτηση δικτυακών δομών Αλυσίδας-Κορμού (Blockchains) σε σύγχρονες τεχνολογικές εφαρμογές καθιστά μη απαραίτητη την παρουσία τρίτων σε συναλλακτικές διαδικασίες και λύσεις, καθώς επίσης εξασφαλίζει και την εγγύτητα των υπηρεσιών. Με τη σύσταση των έξυπνων νομικών συμβολαίων δημιουργούνται νέες προοπτικές στον τομέα της διαχείρισης Συμφωνιών Επιπέδου Υπηρεσίας (Service Level Agreements – SLAs). Αυτού του είδους οι συμφωνίες απαιτούν μετρήσεις και εκπληρώσεις διαδικασιών οι οποίες λόγω της φύσης τους πραγματοποιούνται πιο ορθά και με πιο ωφέλιμο τρόπο, για τον πάροχο της υπηρεσίας (Service Provider - SP), αλλά και για τον πελάτη, μέσα σε ένα αποκεντρωμένο δίκτυο όπου όλοι οι συμμετέχοντες συνιστούν στην ίση διαχείριση της τήρησης των SLAs. Στην παρούσα διπλωματική εργασία εξετάζεται η δυνατότητα επίτευξης έμμεσων απεικονίσεων συμφωνιών επιπέδου υπηρεσίας σε Δικτύα Αλυσίδας-Κορμού (Blockchain Networks), με την ταυτόχρονη επιτήρηση και παρακολούθηση των συμφωνιών στα υποκείμενα περιβάλλοντα. Προτείνεται η δυναμική κατασκευή και εκτέλεση έξυπνων συμβολαίων που αντιστοιχίζουν τις εν λόγω συμφωνίες (SLAs) πάνω σε αποκεντρωμένα δικτύα αλυσίδας-κορμού ενώ παρουσιάζεται και η διαδικασία ανίχνευσης μετρικών τους σε συστήματα χειρισμού γεγονότων (Prometheus). Το δημιουργηθέν σύστημα έμμεσων απεικονίσεων επιχειρεί να προσεγγίσει έννοιες αποκεντρωμένης παρακολούθησης SLAs (decentralized SLA monitoring) και εξετάζεται ως προς την απόδοσή του σε πειραματικά Ethereum δίκτυα (Ethereum Testnets). Τέλος, γίνεται επιπλέον μελέτη της απόδοσης της προτεινόμενης λύσης ως προς διαφορετικού τύπου SLAs με άξονα την ευελιξία των επιτρεπόμενων παραβιάσεων SLA απεικονίσεων μέσα σε ορισμένα χρονικά πλαίσια.

Λέξεις κλειδιά

Συμφωνίες Επιπέδου Υπηρεσίας, Δίκτυα Αλυσίδας-Κορμού, Έξυπνα Συμβόλαια, Node.js, Παρακολούθηση Συμφωνιών Επιπέδου Υπηρεσίας (SLA Monitoring Environment)

Abstract

The adoption of Blockchain network structures in modern technological applications makes the presence of third parties in transactional processes and solutions unnecessary, as well as ensures the proximity of services. The introduction of smart legal contracts creates new perspectives in the field of Service Level Agreement management. Such agreements require measurements and fulfillments of procedures which due to their nature are performed more correctly and in a more beneficial way, for the Service Provider (SP), but also for the customer, within a decentralized network where all participants constitute equal management of compliance with en SLAs. In the present Diploma Thesis, the possibility of achieving indirect representations of Service Level Agreements (SLAs) in Blockchain Networks is examined, by simultaneously supervising and monitoring the agreements in the underlying environments. It is proposed to dynamically build and execute smart contracts that correspond to these agreements (SLAs) on decentralized Blockchain networks, while the process of detecting their metrics in event handling systems (Prometheus) is also presented. The created indirect imaging system attempts to approach concepts of decentralized monitoring of SLAs (decentralized SLA monitoring) and is examined for its performance in Ethereum experimental networks (Ethereum Testnets). Finally, an additional study is made of the performance of the proposed solution in terms of different types of SLAs, focusing on the flexibility of the allowed violations of SLA representations within certain time frames.

Keywords

Service Level Agreements, Blockchain Networks, Smart Contracts, Node.js , SLA Monitoring Environment

Ευχαριστίες

Με την εκπόνηση της παρούσας διπλωματικής εργασίας ο κύκλος των σπουδών μου στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου φτάνει στο τέλος του. Οφείλω ένα μεγάλο ευχαριστώ στην επιβλέπουσα καθηγήτρια της εργασίας, κ. Βαρβαρίγου για την εμπιστοσύνη και την ευκαιρία που μου έδωσε να μελετήσω ένα τόσο ενδιαφέρον αντικείμενο. Επιπλέον, θα ήθελα να ευχαριστήσω θερμά το διδακτορικό ερευνητή Νίκο Καψούλη για την πολύτιμη βοήθεια και διαρκή καθοδήγηση που μου προσέφερε μέχρι το πέρας της διπλωματικής εργασίας. Τέλος, θα ήθελα να ευχαριστήσω από καρδίας τόσο την οικογένεια μου, αλλά και τους φίλους μου για την υποστήριξη που μου παρείχαν καθ΄ όλη τη διάρκεια των σπουδών μου.

Γεωργία-Γρηγορία Ε. Ασημάκη
Αθήνα, 9η Ιουνίου 2022

Περιεχόμενα

Περίληψη **7**

Abstract **9**

Ευχαριστίες **10**

Περιεχόμενα **13**

Κατάλογος Σχημάτων **16**

Κατάλογος Πινάκων **17**

1 Εισαγωγή **18**
1.1 Κίνητρα και Στόχοι 18
1.2 Δομή Εργασίας 19

2 Θεωρητικό Υπόβαθρο και σχετικές εργασίες **20**
2.1 Εξέλιξη του Παγκόσμιου Ιστού 20
2.1.1 Web 1.0 20
2.1.2 Web 2.0 21
2.1.3 Web 3.0 22
2.2 Συμφωνίες Επιπέδου Υπηρεσίας 23
2.2.1 Περιεχόμενο SLA 24
2.2.2 Κύκλος ζωής SLA 24
2.3 Blockchain 26
2.3.1 Αρχιτεκτονική Blockchain 26
2.3.2 Τύποι Blockchain 28
2.3.3 Κυριότερα σημεία του Blockchain 28
2.3.4 Smart Contracts 29
2.3.5 Smart contract Platforms 29
2.3.6 Consensus Mechanisms 30
2.3.7 Blockchain Oracles 30

2.4 Τεχνολογίες . . . 31
2.4.1 HTML . . . 31
2.4.2 Javascript . . . 31
2.4.3 Node.js . . . 32
2.4.4 Express . . . 32
2.4.5 JSON . . . 32
2.4.6 Solidity . . . 32
2.4.7 Infura . . . 32
2.4.8 Metamask . . . 33
2.4.9 Prometheus . . . 33
2.4.10 Remix IDE . . . 33
2.4.11 Etherscan . . . 33
2.5 Σχετικές Εργασίες . . . 33

3 Σχεδιασμός και Υλοποίηση Συστήματος 35
3.1 Γενική περιγραφή αρχιτεκτονικής συστήματος . . . 35
3.2 Συμφωνία Επιπέδου Υπηρεσίας . . . 36
3.3 HTML page . . . 38
3.4 Έξυπνο Συμβόλαιο . . . 39
3.4.1 Αλγόριθμος I . . . 40
3.4.2 Αλγόριθμοι II, III και IV . . . 40
3.5 Εφαρμογή . . . 43

4 Επίδειξη Λειτουργικότητας Εφαρμογής 53
4.1 Αλγόριθμος I . . . 54
4.2 Αλγόριθμοι II, III και IV . . . 57

5 Συγκριτικές Μελέτες 64
5.1 Συγκριτική μελέτη μεταξύ πειραματικών Ethereum δικτύων . . . 64
5.2 Συγκριτική μελέτη μεταξύ αλγορίθμων ελέγχου παραβιάσεων SLA μετρικών . . . 67

6 Μελλοντικές επεκτάσεις και Συμπεράσματα 70
6.1 Συμπεράσματα . . . 71

Κατάλογος Σχημάτων

2.1 Web 2.0 “Meme Map” by Tim O’Reilly 2009 21
2.2 Web 2.0 “Meme Map” by Tim O’Reilly 2009 (Μεταφρασμένο) 22
2.3 Στάδια κύκλου ζωής SLA 25
2.4 Παράδειγμα blockchain που αποτελείται από μια συνεχή ακολουθία μπλοκ. [36] . . 26
2.5 Δομή block [36] 27

3.1 Αρχιτεκτονική Συστήματος 35
3.2 SLA Template 37
3.3 Κώδικας HTML με τον οποίο γίνεται η εισαγωγή του SLA στο σύστημα 38
3.4 Σελίδα εισαγωγής SLA με τη μορφή αρχείου 39
3.5 Κώδικας Constructor των Smart Contracts 39
3.6 Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου I 41
3.7 Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου II 42
3.8 Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου III 42
3.9 Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου IV 43
3.10 Ορισμός Express Server και Application 44
3.11 Σύνδεση με Ethereum Testnets και εισαγωγή του λογαριασμού που δημιουργήθηκε στο Metamask 44
3.12 Εισαγωγή δεδομένων του SLA στο αρχείο Node.js μέσω συνάρτησης http post . . 45
3.13 Μεταγλώττιση έξυπνου συμβολαίου μέσω του Remix IDE 46
3.14 Εισαγωγή έξυπνου συμβολαίου 47
3.15 Συνάρτηση deployment για το έξυπνο συμβόλαιο 47
3.16 Συνάρτηση Interval για προσδιορισμό χρονικού παραθύρου ανάμεσα στις ενημερώσεις του Prometheus και ανάμεσα στις κλήσεις της συνάρτησης ελέγχου του έξυπνου συμβολαίου. 48
3.17 Τιμές μετρήσεων διαθεσιμότητας, HTTP σφαλμάτων και χρόνου απόκρισης που λειτουργούν ως αυτοσχέδιο Blockchain Oracle 49
3.18 Συνάρτηση EventEmit με την οποία γίνεται η κλήση της συνάρτησης ελέγχου παραβιάσεων στο έξυπνο συμβόλαιο. 50
3.19 Σύνδεση με Prometheus και ορισμός μετρικής Violations. 50
3.20 Συνάρτηση EventListen μέσω της οποίας ενημερώνεται η μετρική του Prometheus 51

4.1 Αρχική σελίδα Prometheus 53
4.2 Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus. 54
4.3 Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus. 56
4.4 Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus. 56
4.5 Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus. 56
4.6 Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus. 57
4.7 Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus. 57
4.8 Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus. 58
4.9 Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus. 59
4.10 Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus. 59
4.11 Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus. 60
4.12 Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus. 61
4.13 Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus. 61
4.14 Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus. 61
4.15 Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus. 62
4.16 Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus. 63
4.17 Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus. 63

5.1 Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος I, Rinkeby Ethereum testnet 65
5.2 Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος I, Goerli Ethereum testnet 65
5.3 Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος I, Kovan Ethereum testnet 66

5.4 Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος II, Rinkeby Ethereum testnet 68
5.5 Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος III, Rinkeby Ethereum testnet 68
5.6 Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος IV, Rinkeby Ethereum testnet 69

Κατάλογος Πινάκων

3.1 Πληροφορίες του κάθε SLA που δημιουργήθηκε. 37

4.1 Μέσος όρος για την κάθε μετρική που προκύπτει ανά δέκα μετρήσεις 55
4.2 Παραβιάσεις ανά μέτρηση για τη ¨Διαθεσιμότητα' για το SLA με id = 1, (n:δεν υπάρχει παραβίαση,y:υπάρχει παραβίαση) 58
4.3 Παραβιάσεις ανά μέτρηση για τα ¨Σφάλματα HTTP' για το SLA με id = 1, (n:δεν υπάρχει παραβίαση,y:υπάρχει παραβίαση) 60
4.4 Παραβιάσεις ανά μέτρηση για το ¨Χρόνο Απόκρισης' για το SLA με id = 1, (n:δεν υπάρχει παραβίαση,y:υπάρχει παραβίαση) 62

5.1 Αριθμός παραβιάσεων της κάθε μετρικής, αλλά και συνολικά, για το κάθε SLA που χρησιμοποιήθηκε σε εκτέλεση του Αλγορίθμου I. 66
5.2 Txn fees που χρεώθηκε ο καθένας από τους 5 λογαριασμούς που δημιουργήθηκαν στο Metamask για καθένα από τα πειραματικά Ethereum δίκτυα Goerli, Kovan και Rinkeby, ύστερα από την προσθήκη 5 SLAs στο σύστημα με 50 ελέγχους παραβιάσεων, με τη χρήση του Αλγορίθμου I. 67
5.3 Txn fees του Rinkeby πειραματικού Ethereum δικτύου που χρεώθηκε ο καθένας από τους 5 λογαριασμούς που δημιουργήθηκαν στο Metamask για καθέναν από τους 4 αλγορίθμους που χρησιμοποιήθηκαν, ύστερα από την προσθήκη 5 SLAs στο σύστημα με 50 ελέγχους παραβιάσεων. 69

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρα και Στόχοι

Τη σημερινή εποχή, η τεχνολογία των δικτύων παρουσιάζει ολοένα και μεγαλύτερη πολυπλοκότητα, ενώ ταυτόχρονα υπάρχει απαίτηση για υψηλότερα επίπεδα υπηρεσιών και κάλυψη αυξανόμενων απαιτήσεων των καταναλωτών. Όλα τα παραπάνω, σε συνδυασμό με τη ραγδαία ανάπτυξη του ανταγωνισμού ανάμεσα στους παρόχους υπηρεσιών έχουν δημιουργήσει την ανάγκη αναζήτησης νέων τρόπων διαχείρισης των υπηρεσιών τους. Βασική προϋπόθεση επιτυχούς διαχείρισης αποτελεί ο ορισμός των προσδοκιών, δικαιωμάτων και υποχρεώσεων του πελάτη και του παρόχου, θέματα τα οποία καλύπτονται πλήρως μέσω των Συμφωνιών Επιπέδου Υπηρεσίας (Service Level Agreements (SLAs)).

Θεμελιώδες κομμάτι των SLAs είναι η επιτήρηση της ποιότητας των υπηρεσιών (Quality of Service - QoS), μέσω της οποίας μπορεί να υπολογιστεί το ποσοστό συμμόρφωσης της υπηρεσίας με τους όρους της συμφωνίας. Η διαδικασία επιτήρησης, όμως, των όρων και αποζημίωσης είναι περίπλοκη και χρονοβόρα. Στη περίπτωση που θα συμβεί κάποια παραβίαση, οι συνδρομητές και οι πάροχοι των υπηρεσιών χρησιμοποιούν προσωπικό και κεφάλαιο για την εξακρίβωση και την αντιμετώπισή της.

Το κάθε μέλος της συμφωνίας διατηρεί μία δική του βάση δεδομένων με όλο το ιστορικό των μετρικών που χρησιμοποιούνται για την παρακολούθηση της ποιότητας των υπηρεσιών. Το ιστορικό αυτό αποθηκεύεται με τρόπο συγκεντρωτικό, μεμονωμένης προσέγγισης και με ελάχιστη διαφάνεια. Επομένως, δύσκολα μπορεί να επιτευχθεί εμπιστοσύνη ως προς την εγγύτητα των πληροφοριών του. Επιπλέον, ακόμα και στην περίπτωση που οι βάσεις αυτές είναι φερέγγυες, λόγω της τεχνολογίας με την οποία κατασκευάζονται, είναι επιρρεπείς σε απώλειες ή και έκθεση ευαίσθητων δεδομένων.

Η τεχνολογία Blockchain και τα Έξυπνα Συμβόλαια έχουν προοπτικές διατήρησης αμετάβλητης βάσης δεδομένων, αλλά και επίτευξης κατανεμημένης συναίνεσης μεταξύ των μελών χωρίς την ανάγκη επικύρωσης των αποθηκευμένων πληροφοριών από έναν έμπιστο τρίτο. Η παρούσα διπλωματική εργασία πραγματεύεται τη δυνατότητα έμμεσης απεικόνισης των Συμφωνιών Επιπέδου

Υπηρεσίας σε Δικτύα Αλυσίδας-Κορμού (Blockchain Networks), με την ταυτόχρονη επιτήρηση και παρακολούθηση των συμφωνιών στα υποκείμενα περιβάλλοντα. Με αυτό τον τρόπο γίνεται προσπάθεια επίτευξης ασφάλειας, εγγύτητας και αμετάβλητου στη διαδικασία επιτήρησης της ποιότητας των υπηρεσιών.

Η διαδικασία επιτήρησης γίνεται μέσω έξυπνων συμβολαίων τα οποία αναπτύσσονται στο δίκτυο Ethereum και εξυπηρετούν ταυτόχρονα και τον πελάτη, αλλά και τον πάροχο της υπηρεσίας. Τα συμβόλαιο αυτά διαθέτουν όλες τις απαραίτητες πληροφορίες οι οποίες έχουν ήδη συμφωνηθεί προτού γίνει το deployment του smart contract στο δίκτυο. Έτσι, ο έλεγχος για παραβιάσεις αυτοματοποιείται ανάλογα με τους όρους της συμφωνίας. Παράλληλα, ενημερώνεται και ένα σύστημα χειρισμού γεγονότων (Prometheus) μέσα από το οποίο καθίσταται δυνατή η on-chain παρακολούθηση των τυχόν παραβιάσεων του SLA. Ένας επιπλέον στόχος της διπλωματικής εργασίας είναι η μελέτη της απόδοσης της λύσης που παρουσιάζεται, γύρω από την ευελιξία των επιτρεπόμενων παραβιάσεων SLA απεικονίσεων μέσα σε ορισμένα χρονικά πλαίσια. Τέλος, το σύστημα που προτείνεται εξετάζεται και ως προς την απόδοσή του σε Ethereum Testnets.

1.2 Δομή Εργασίας

Η παρούσα διπλωματική εργασία χωρίζεται σε πέντε βασικά μέρη, το θεωρητικό υπόβαθρο των εξεταζόμενων εννοιών, το σχεδιασμό και την υλοποίηση του προτεινόμενου συστήματος, την επίδειξη της λειτουργικότητας της εφαρμογής, τις συγκριτικές μελέτες και τέλος, τις μελλοντικές επεκτάσεις και τα συμπεράσματα σχετικά με την εφαρμογή του δημιουργηθέντος συστήματος. Αναλυτικά, στο Κεφάλαιο 2 αναλύεται το θεωρητικό υπόβαθρο που αντιστοιχεί:

- Στην εξέλιξη του παγκόσμιου Ιστού από την πρωταρχική του μορφή έως τον Σημασιολογικό Ιστό (Web 3.0)
- Στον ορισμό των Συμφωνιών Επιπέδου Υπηρεσίας (SLAs) και πληροφοριών που απαιτούνται για την κατανόησή τους.
- Στην ανάλυση και τον ορισμό της τεχνολογίας των δικτύων Αλυσίδας-Κορμού (Blockchain) μαζί με έννοιες όπως Smart Contracts, Consensus Mechanisms και Blockchain Oracles.
- Στις τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή του συστήματος που παρουσιάζεται στην εργασία
- Σε εργασίες σχετικού περιεχομένου

Στο Κεφάλαιο 3 περιγράφεται διεξοδικά ο τρόπος με τον οποίο σχεδιάστηκε και υλοποιήθηκε η προτεινόμενη εφαρμογή, στο Κεφάλαιο 4 επιδεικνύεται η λειτουργικότητα της εφαρμογής και στο Κεφάλαιο 5 πραγματοποιείται συγκριτική μελέτη της απόδοσής της σε διάφορα πειραματικά Ethereum δίκτυα, αλλά και με διαφορετικά επιπέδα ευελιξίας των επιτρεπόμενων SLA παραβιάσεων. Τέλος, στο Κεφάλαιο 6 γίνεται αναφορά σε πιθανές μελλοντικές επεκτάσεις της εφαρμογής και στα συμπεράσματα που προκύπτουν για υτην εφαρμογή.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο και σχετικές εργασίες

Στην παρούσα ενότητα αναλύονται όλα τα θεωρητικά μέρη από τα οποία αποτελείται η εργασία. Αρχικά, γίνεται λόγος για τον παγκόσμιο ιστό και την εξέλιξή του. Στην πορεία παρουσιάζονται εκτενώς οι Συμφωνίες Επιπέδου Υπηρεσίας και οι μεταβλητές που τις καθορίζουν και τέλος, γίνεται διεξοδική ανάλυση της τεχνολογίας Blockchain και όσων αυτή συνεπάγεται.

2.1 Εξέλιξη του Παγκόσμιου Ιστού

Ο όρος ¨Παγκόσμιος Ιστός' απαιτεί συχνά αποσαφήνιση από τον όρο ¨Διαδίκτυο¨. Το Διαδίκτυο συνδέει υπολογιστές και πόρους σε ολόκληρο τον κόσμο κάνοντας χρήση των πρωτοκόλλων TCP/IP (Transmission Control Protocol/ Internet Protocol). Ο Παγκόσμιος Ιστός (World Wide Web) αναπτύχθηκε από τον Tim Berners-Lee [14] και ορίζεται ως ένα κατανεμημένο ετερογενές συνεργατικό σύστημα πληροφοριών πολυμέσων. Ουσιαστικά ο Ιστός είναι η πρακτική και υπάρχουσα εφαρμογή του πραγματικού κόσμου μιας καθολικής βάσης δεδομένων πληροφοριών. Οι πληροφορίες αυτές συνδέονται με άλλες με σκοπό οι πιο σημαντικές και χρήσιμες να μπορούν να βρεθούν γρήγορα. [19]

2.1.1 Web 1.0

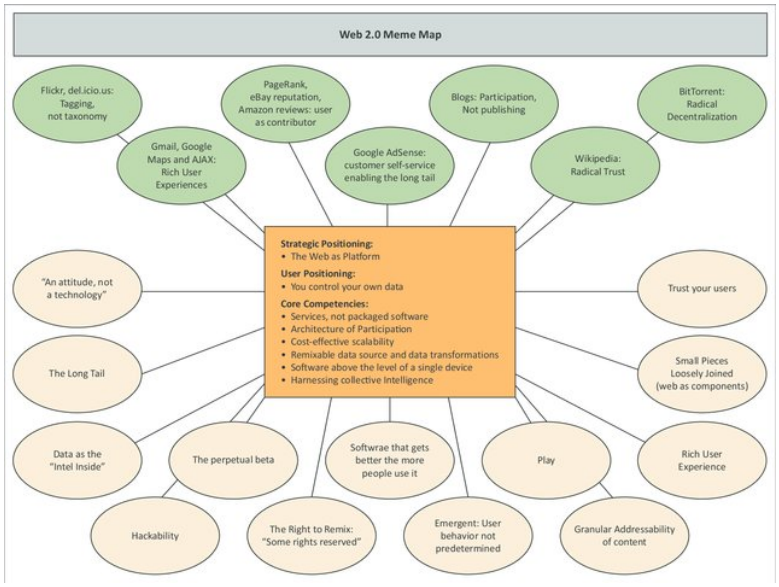
Η σκέψη πίσω από τη δημιουργία του ιστού ήταν η κατασκευή ενός τυπικού χώρου στον οποίο υπάρχει η δυνατότητα διαμοιρασμού δεδομένων. Η πορεία του ιστού ξεκίνησε από τον Web 1.0 ο οποίος ήταν ως επί το πλείστον ένας ιστός με δυνατότητα ανάγνωσης. Οι ιστότοποι περιλάμβαναν στατικές σελίδες HTML που ανανεώνονταν ασυνεπώς και ο πρωταρχικός τους στόχος ήταν να διανείμουν τα δεδομένα για οποιονδήποτε και να δημιουργήσουν μια διαδικτυακή εγγύτητα. Ο ιστός αυτός λειτουργούσε κυρίως μέσω των HTTP, HTML και URL συμβάσεων. [38]

2.1.2 Web 2.0

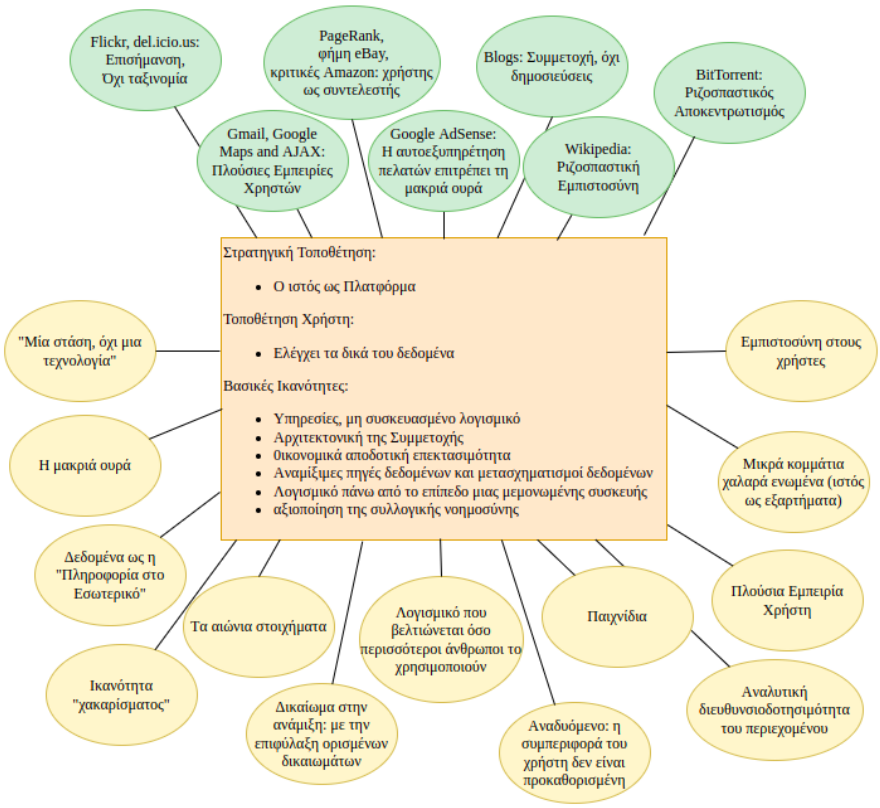
Με σκοπό τη βελτίωση του Ιστού, δεδομένων των μειονεκτημάτων του Web 1.0, κατασκευάστηκε από τον Dale Dougherty, πρωτοπόρο στον ιστό και τον αντιπρόεδρο του O'Reilly το 2004 ένα νέο μοντέλο Ιστού, ο Web 2.0. Το νέο αυτό μοντέλο βασίζεται στα intra links, κάθε σελίδα είναι διαχωρισμένη και περιέχει έναν αριθμό υπερσυνδέσμων (hyperlinks). Οι χρήστες καλούνται να συνδεθούν και στη συνέχεια μέσω αυτών μεταφέρονται από τη μία σελίδα στην άλλη. Με αυτό τον τρόπο ο κάθε χρήστης μπορεί να παραθέτει την κριτική του για την εκάστοτε σελίδα, αλλά και να λάβει πληροφορίες. Ο Web 2.0 δίνει στο χρήστη μία πληθώρα δυνατοτήτων, όπως καταγραφή ιστολογίων, δημοσίευση φωτογραφιών ή βίντεο και σύνδεση με κοινότητες. Έτσι, παρατηρείται αύξηση της αλληλεπίδρασης του χρήστη με τον ιστό, κσθώς ο τελευταίος παρέχει πλέον τον χώρο στον οποίο δίνεται η δυνατότητα ανταλλαγής πληροφοριών, αλλά και μια σειρά από εργαλεία και τεχνολογίες.

Με τον όρο Web 2.0 περιγράφονται ιστότοποι που ενσωματώνουν έντονα το κοινωνικό στοιχείο, αλλά και ενθαρρύνουν περιεχόμενο που δημιουργείται από τους χρήστες. θεωρούν τους χρήστες ως οντότητες πρώτης κατηγορίας με σελίδες προφίλ που συμπεριλαμβάνουν προσωπικές πληροφορίες, αλλά και σχόλια άλλων χρηστών. Δίνεται επίσης η δυνατότητα δημιουργίας συνδέσεων μεταξύ των χρηστών και δημοσίευσης περιεχομένου σε πολλές μορφές. Ακόμα, οι επισκέπτες ενός τέτοιου ιστότοπου μπορούν να κάνουν μετατροπές σε αυτόν, όπως δημοσιεύσεις κριτικών και πρόσθεση πληροφοριών με σκοπό να προβληθούν από μελλοντικούς επισκέπτες. [32]

Για να αναδείξει τις ιδέες που εκπέμπονται από τον πυρήνα του Web 2.0, ο Tim O'Reilly σε ένα συνέδριο που αφορούσε το θέμα [10], παρέπεμψε το διάγραμμα που φαίνεται στα Σχήματα 2.1 και 2.2 (μεταφρασμένο στα ελληνικά).



Σχήμα 2.1: Web 2.0 "Meme Map" by Tim O'Reilly 2009



Σχήμα 2.2: Web 2.0 “Meme Map” by Tim O’Reilly 2009 (Μεταφρασμένο)

2.1.3 Web 3.0

Ο Web 2.0 παρέχει πρόσβαση στους χρήστες σε μία πληθώρα πληροφοριών. Το μειονέκτημά του έγκειται στο γεγονός ότι δεν έχει τη δυνατότητα αυτόματης ταξινόμησής τους και επομένως οι χρήστες πρέπει να αφιερώνουν αρκετό χρόνο στην αναζήτηση των δεδομένων που χρειάζονται. Υπήρξε, λοιπόν, η ανάγκη για έναν Ιστό που θα είχε τη δυνατότητα να οργανώνει αυτόματα τα δεδομένα και να εμφανίζει τα σχετικότερα με την αναζήτηση του χρήστη αποτελέσματα. Λύση στο παραπάνω πρόβλημα έδωσε ο Tim Berners Lee με την ίδρυση του Web 3.0, ο οποίος αναφέρεται και ως Σημασιολογικός Ιστός (Semantic Web). Ο Web 3.0 αποτελεί ένα σύνολο προτύπων που μετατρέπει τον Ιστό σε μια μεγάλη βάση δεδομένων. Πλέον ο Ιστός κατανοεί το νόημα των πληροφοριών και μαζί με τις λέξεις-κλειδιά μπορεί να προσδιορίζει επιπλέον το πλαίσιο στο οποίο οι λέξεις παρουσιάζονται μέσω των μεταξύ τους σχέσεων. Έτσι, καταφέρνει να παρέχει ακριβή αποτελέσματα. Ακόμα, προσδιορίζει τα συνώνυμα των αναζητηθέντων πληροφοριών και εμφανίζει τα δεδομένα που σχετίζονται και με αυτά.

Ο Σημασιολογικός Ιστός εκμεταλλεύεται πράκτορες λογισμικού με σκοπό τη σάρωση και την ερμηνεία πληροφοριών σε ιστοσελίδες. Οι πράκτορες αυτοί είναι προγράμματα που εξερευνούν τον

Ιστό αναζητώντας σχετικές πληροφορίες. Για να μπορέσουν να δράσουν με αυτόν τον τρόπο ο Web 3.0 περιέχει συλλογές πληροφοριών οι οποίες ονομάζονται οντολογίες. Οντολογία θεωρείται ένα αρχείο στο οποίο ορίζονται οι σχέσεις ανάμεσα σε ένα σύνολο όρων. Για παράδειγμα, ο όρος ¨Αδερφός/Αδερφή' χρησιμοποιείται για να ορίσει κάποιον που μοιράζεται τους ίδιους γονείς με το υποκείμενο. Μία οντολογία μπορεί να ορίσει κάθε όρο που βρίσκουμε σε ένα οικογενειακό δέντρο, όπως για παράδειγμα γονέας, παιδί, παππούς κ.ο.κ.. Επιπλέον, μία συλλογή οντολογιών σχηματίζει ένα Περιβάλλον Πλέγματος. Ένας αποτελεσματικός Σημασιολογικός Ιστός είναι απαραίτητο να περιέχει λεπτομερείς και περιεκτικές οντολογίες. Η αρχική ιδέα του Tim Berners Lee βασιζόταν στα metadata. Τα τελευταία αποτελούν πληροφορίες που χρησιμοποιούνται από τους υπολογιστές κατά την αναζήτηση λέξεων-κλειδιών και δεν είναι ορατές στους ανθρώπους. Ο συνδυασμός τους με τις οντολογίες θα είχε τεράστια συνεισφορά στην επιτυχία του Web 3.0. [32]

2.2 Συμφωνίες Επιπέδου Υπηρεσίας

Μια συμφωνία επιπέδου υπηρεσίας (Service Level Agreement - S.L.A.) είναι ένα συμβόλαιο που ορίζει επίσημα τη σχέση μεταξύ ενός παρόχου υπηρεσιών και του πελάτη του. Μία τέτοια συμφωνία χρησιμοποιείται για να προσδιορίσει τις προσδοκίες που οφείλει να έχει ο πελάτης από τον πάροχο, τις υποχρεώσεις και των δύο πλευρών, την απόδοση, τη διαθεσιμότητα τους στόχους ασφαλείας της υπηρεσίας, καθώς και τις διαδικασίες που πρέπει να ακολουθούνται για να διασφαλίζεται η συμμόρφωση με το συμβόλαιο.[34]

Οι μετρικές SLA είναι ένα σύνολο βασικών δεικτών απόδοσης(Key Performance Indicator-KPI) που προορίζονται για μελέτη και παρακολούθηση. Οι βασικοί δείκτες απόδοσης (KPIs) ορίζονται ως ποσοτικοποιήσιμες και στρατηγικές μετρήσεις που αντικατοπτρίζουν τους κρίσιμους παράγοντες επιτυχίας της επιχείρησης.[25]

Οι μετρικές χρησιμοποιούνται για να είναι δυνατός ο εντοπισμός παραβιάσεων των υποσχόμενων επιπέδων των υπηρεσιών ώστε να διαφυλλάσσονται ορθώς τα δικαιώματα και των δύο πλευρών της συμφωνίας. Οι πιο συνήθεις μετρικές περιλαμβάνουν τη Διαθεσιμότητα (Availability), τη Διεκπεραιωτική Ικανότητα (Throughput), το Χρόνο Διακοπής Λειτουργίας (Downtime) και το Χρόνο Απόκρισης (Response Time). Διαφορετικοί οργανισμοί έχουν διαφορετικούς ορισμούς για κρίσιμες παραμέτρους πληροφορικής (Information technology (IT)). Για παράδειγμα, ο ορισμός για τη διαθεσιμότητα μιας υπηρεσίας θα μπορούσε να διατυπωθεί και βασιζόμενος στην υποδομή (συνδέσεις TCP) αλλά και στην ικανότητα πρόσβασης στην εφαρμογή της υπηρεσίας. Συνεπώς, μία κακή επιλογή μετρικών μπορεί να οδηγήσει σε ασάφεια, ανεκπλήρωτες προσδοκίες και προβλήματα κατά την ολοκλήρωση των SLA.

Μπορούμε να χωρίσουμε τις μετρικές των SLA σε τρεις βασικές κατηγορίες. Στην πρώτη ανήκουν τα **αντικείμενα της υπηρεσίας που εξετάζονται**, δηλαδή τα βασικά αντικείμενα της υπηρεσίας, όπως hardware, λογισμικό, δίκτυο κ.λπ.. Πιο σύνθετες μετρικές όπως διαθεσιμότητες από άκρο σε άκρο μπορούν να αναλυθούν σε μικρότερες άμεσες μετρικές που εκχωρούνται σε έναν από αυτούς τους βασικούς τύπους αντικειμένων. Στη δεύτερη κατηγορία ανήκουν οι **διεργασίες ITIL (Information Technology Infrastructure Library)**, όπου οι ευθύνες και οι

διαδικασίες έχουν σαφή ορισμό, πράγμα το οποίο δημιουργεί το κατάλληλο περιβάλλον για πιθανή βελτιστοποίηση των διαδικασιών μέσω των μετρικών. Η τελευταία κατηγορία αφορά τον **βαθμό αυτοματισμού**, δηλαδή κατά πόσο υπάρχει δυνατότητας μέτρησης και κατ΄ επέκταση δυνατότητα αυτοματοποίησης των μετρήσεων. Αυτή η κατηγορία περιλαμβάνει μετρικές που μπορούν να συλλεχθούν εύκολα και διευκολύνει τον εντοπισμό προβληματικών κανόνων SLA σε υπάρχοντα συμβόλαια. [26]

2.2.1 Περιεχόμενο SLA

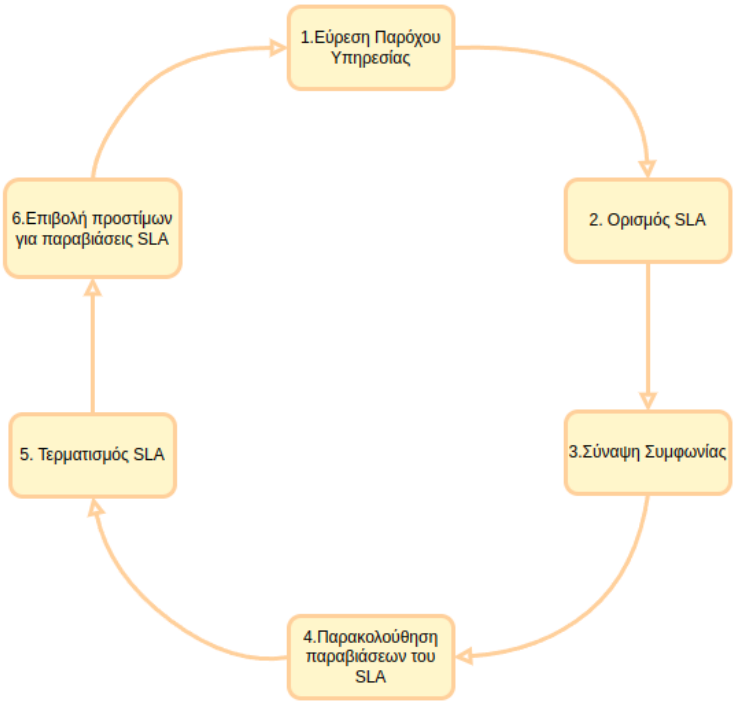
Ένα SLA συνήθως περιέχει τις εξής πληροφορίες [34]:

- Περιγραφή των υπηρεσιών που θα παρέχονται και τα αναμενόμενα επίπεδά τους.
- Διαδικασία που θα ακολουθηθεί σε περίπτωση εμφάνισης προβλημάτων σχετικά με την υπηρεσία.
- Χρονικό πλαίσιο προσδωκώμενης ανταπόκρισης και επίλυσης προβλημάτων.
- Ένα μηχανισμό ελέγχου μαζί με ένα σύνολο μετρικών για την παρακολούθηση της υπηρεσίας.
- Ένδικα μέσα ή κυρώσεις που είναι διαθέσιμα στον καταναλωτή και τον πάροχο σε περίπτωση που δεν πληρούνται οι όροι του συμβολαίου.
- Ρήτρες για συνθήκες υπό τις οποίες δεν ισχύει το επίπεδο υπηρεσίας ή βάσει των οποίων θα θεωρούνταν παράλογο για την ικανοποίηση των απαιτούμενων.
- Περιορισμούς που σχετίζονται με τη συμπεριφορά του πελάτη.

2.2.2 Κύκλος ζωής SLA

Ο τρόπος που διαμορφώνεται ο κύκλος ζωής ενός SLA δεν είναι συγκεκριμένος, αλλά γενικά μπορεί να διαχωριστεί σε έξι επιμέρους στάδια, όπως φαίνεται και στο Σχήμα 2.3. Τα στάδια αυτά επανεκκινούνται κάθε φορά που χρειασθεί να γίνει κάποια αλλαγή στο συμβόλαιο ή απλά ανανέωσή του. [21]

1. *Εύρεση παρόχου της υπηρεσίας (Service Provider - SP)* : Η ενέργεια αυτή γίνεται από τον πελάτη (Customer) ο οποίος οφείλει να γνωρίζει τους απαραίτητους πόρους για την κάλυψη των αναγκών του.
2. *Ορισμός SLA* : Σε αυτό το βήμα διεκπεραιώνονται συνήθως διαπραγματεύσεις μεταξύ του παρόχου και του πελάτη ώστε να ορισθούν οι υπηρεσίες που θα περιλαμβάνει το SLA, τα επιθυμητά επίπεδά τους, αλλά και πολιτικές κυρώσεων .



Σχήμα 2.3: Στάδια κύκλου ζωής SLA

3. *Καθιέρωση συμφωνίας* : Περιλαμβάνει τον ορισμό και την ανάπτυξη ενός προτύπου SLA, οι όροι του οποίου έχουν συμφωνηθεί στο δεύτερο στάδιο. Μόλις το συμβόλαιο υπογραφεί και από τα δύο μέρη, δεσμεύονται και οι δύο από τους συμφωνηθέντες όρους.

4. *Παρακολούθηση παραβίασης SLA* : Η παραβίαση των όρων που ορίζονται στο SLA μπορεί να ανιχνευθεί μόνο μέσω παρακολούθησης των υπηρεσιών. Η μέθοδος που γίνεται η επιτήρηση είναι σημαντικό να παρέχει αξιόπιστες μετρήσεις κάτι το οποίο φέρει αρκετές προκλήσεις από μόνο του.

5. *Τερματισμός SLA* : Τερματισμός του συμβολαίου υφίσταται είτε λόγω λήξεως της ισχύς του είτε λόγω κάποιας παραβίασης των όρων του.

6. *Επιβολή κυρώσεων για παραβίαση SLA* : Σε περίπτωση παραβίασης κάπου συμφωνηθέντα όρου, επικαλούνται και εκτελούνται οι αντίστοιχες ποινικές ρήτρες.

2.3 Blockchain

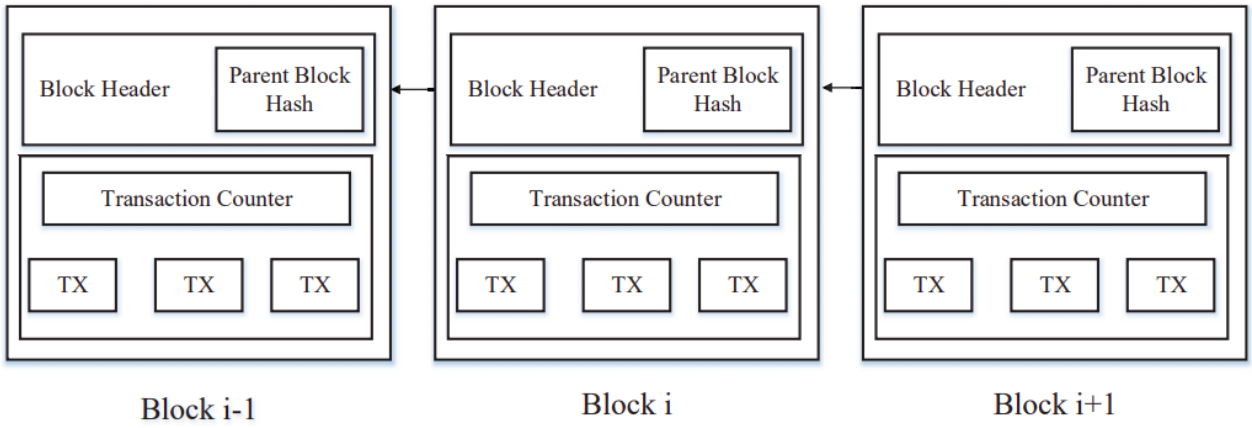
Ένα σημαντικό πρόβλημα που παρατηρείται στα πληροφοριακά συστήματα και ιδιαίτερα σε περιπτώσεις που εμπλέκονται ευαίσθητες πληροφορίες είναι η εμπιστοσύνη. Το πρόβλημα αυτό γίνεται ακόμα πιο περίπλοκο όταν δεν υπάρχει κανένας μηχανισμός επαλήθευσης ή ελέγχου. Έχοντας αυτά τα δεδομένα, ο Satoshi Nakamoto, παρουσίασε το 2008 δύο ριζοσπαστικές έννοιες: το Bitcoin και το Blockchain.

Το Bitcoin [23] αποτελεί ένα εικονικό κρυπτονόμισμα που διατηρεί την αξία του χωρίς υποστήριξη από οποιαδήποτε κεντρική αρχή ή οικονομική οντότητα και διατηρείται συλλογικά και με ασφάλεια από ένα αποκεντρωμένο δίκτυο ομότιμων κόμβων (Peer-to-peer network) που συνθέτουν ένα ελεγχόμενο και επαληθεύσιμο δίκτυο. [28]

Το Blockchain ορίζεται ως μία κατανεμημένη βάση δεδομένων που συντηρεί όλο το ιστορικό των συναλλαγών που έχουν πραγματοποιηθεί στο δίκτυο σε μία λίστα από blocks. Η κατανεμημένη αυτή βάση δεδομένων βρίσκεται υπό μορφή αντιγράφων στην κατοχή όλων των κόμβων του δικτύου. Σημαντικό πλεονέκτημά της τεχνολογίας αυτής είναι ότι παρέχει τη δυνατότητα ασφαλών συναλλαγών και ασφαλούς επικοινωνίας μεταξύ ατόμων μη εμπιστοσύνης χωρίς την ανάγκη αξιόπιστου τρίτου. [11]

2.3.1 Αρχιτεκτονική Blockchain

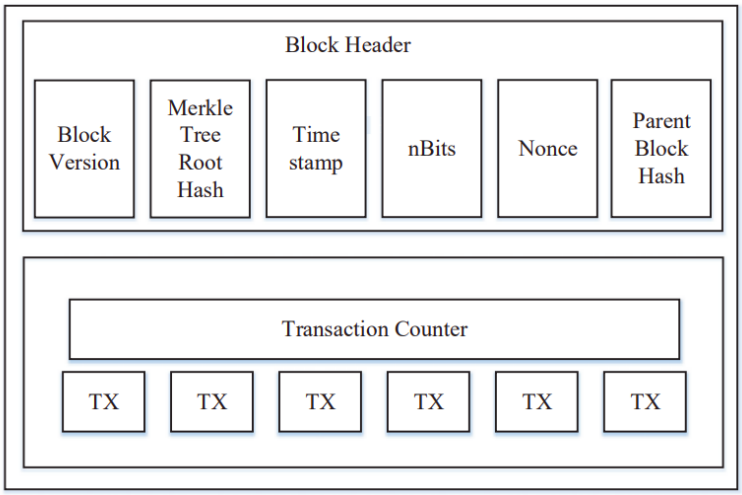
Ως Blockchain ορίζουμε μία σειρά από blocks τα οποία είναι συνδεδεμένα μεταξύ τους μέσω κρυπτογραφικού κατακερματισμού (cryptographic hash). Το κάθε block έχει αποθηκευμένη την τιμή κατακερματισμού του προηγούμενου ώστε να μπορεί να δεικτοδοτεί σε αυτό και έτσι δημιουργείται μία αλυσίδα από block (Chain of Blocks τα οποία έχουν μόνο ένα γονικό block). Το πρώτο block μίας αλυσίδας το οποίο δεν έχει πρόγονο ονομάζεται genesis block. Τα παραπάνω μπορούν να διακριθούν και στο σχήμα 2.4 στο οποίο παρουσιάζεται τμήμα ενός blockchain. [36]



Σχήμα 2.4: Παράδειγμα blockchain που αποτελείται από μια συνεχή ακολουθία μπλοκ. [36]

Στο σχήμα 2.5 φαίνεται η δομή ενός block, το οποίο αποτελείται από δύο τμήματα, την κεφαλίδα (block header) και το σώμα (block body). Το σώμα του μπλοκ αποτελείται από τις συναλλαγές (Transactions) και τον μετρητή συναλλαγών (Counter). Στην κεφαλίδα περιέχονται έξι πληροφορίες: [36]

1. Block Version: ορίζει τους κανόνες επικύρωσης του κάθε block
2. Merkle tree root hash: τιμή κατακερματισμού όλων των συναλλαγών στο block
3. Timestamp: τρέχουσα ώρα
4. nBits: όριο στόχου ενός έγκυρου κατακερματισμού μπλοκ
5. Nonce: Πεδίο τεσσάρων byte που χρησιμοποιείται για τον υπολογισμό κατακερματισμού
6. Κατακερματισμός γονικού μπλοκ (Parent block hash): μια τιμή κατακερματισμού 256-bit που παραπέμπει στο προηγούμενο μπλοκ.



Σχήμα 2.5: Δομή block [36]

Η τρέχουσα κατάσταση ενός blockchain αντιπροσωπεύεται από τις συναλλαγές που δημιουργούνται από τους κόμβους και συγκεντρώνονται σε blocks. Ως συναλλαγή (transaction) σε ένα blockchain δίκτυο θεωρούμε μία μικρή μονάδα εργασίας που είναι αποθηκευμένη σε δημόσια αρχεία. Η εκτέλεση και η αποθήκευση αυτών των εγγραφών στο δίκτυο προϋποθέτει έγκριση της πλειοψηφίας των συμμετεχόντων του δικτύου. Ανά πάσα στιγμή οποιαδήποτε συναλλαγή μπορεί να υποβληθεί σε έλεγχο, αλλά η ενημέρωσή της είναι αδύνατη. [30]

2.3.2 Τύποι Blockchain

Τα δίκτυα blockchain μπορούν να διακριθούν σε δύο γενικές κατηγορίες [29]:

- **Permissionless (ή Public)** : Οποιοσδήποτε κόμβος ή χρήστης μπορεί να ενταχθεί στο σύστημα με ψευδώνυμο τρόπο. Public blockchains είναι το Bitcoin και το Ethereum.

- **Permissioned (ή Private)** : Στην κατηγορία αυτή ανήκουν το Fabric και το Quorum. Εδώ, ο χρήστης και ο κόμβος πρέπει να έχουν εξουσιοδοτηθεί για να ενταχθούν στο σύστημα. Αυτού του είδους τα blockchain είναι πιο κατάλληλα για εταιρικές εφαρμογές και χρησιμοποιούνται ιδιαίτερα στον χρηματοοικονομικό τομέα, καθώς χαρακτηρίζονται από ισχυρό έλεγχο όσον αφορά τα μέλη του και ρύθμιση των δράσεών του.

Τα private blockchain είναι πιο ασφαλή, καθώς δε κάνουν καμία υπόθεση ταυτότητας. Αντίθετα, τα public έχουν ασθενέστερη ασφάλεια, για τον αντίστοιχο λόγο, αλλά έχουν τη δυνατότητα να πετύχουν αρκετά υψηλότερη απόδοση, καθώς μπορούν να χρησιμοποιήσουν πρωτόκολλα συναίνεσης με μεγαλύτερη αποδοτικότητα.

2.3.3 Κυριότερα σημεία του Blockchain

Τα σημαντικότερα σημεία που παρέχει η καινοτομία Blockchain είναι [18] :

- **Αποκέντρωση (Decentralization)**: Το Blockchain είναι μία αποκεντρωμένη και κατανεμημένη εγκατάσταση που εξαρτάται από την αντιστοιχία Peer-to-Peer μεταξύ των κατανεμημένων κόμβων. Η αποκέντρωση που παρέχει το δίκτυο γίνεται ισχυρότερη χρησιμοποιώντας τη δύναμη χειρισμού του κάθε συμμετέχοντα πελάτη. Με αυτόν τον τρόπο μειώνεται η αδράνεια και επιλύεται το ζήτημα ενός μόνο σημείου αποτυχίας.

- **Διαφάνεια (Transparency)**: Στο Blockchain όλοι οι κόμβοι μπορούν να προσεγγίσουν κάθε μία από τις λεπτομέρειες των συναλλαγών που συμβαίνουν σταθερά στο δίκτυο, καθώς επίσης ο καθένας διατηρεί ένα αντίγραφο της βάσης δεδομένων.

- **Αμετάβλητο (Immutability)** : Το πιο σημαντικό χαρακτηριστικό του Blockchain είναι η ικανότητα διασφάλισης της ακεραιότητας των συναλλαγών με την παράδοση αδιάλλακτων αρχείων. Χρησιμοποιεί κατακερματισμό για να συνδέσει κάθε μπλοκ με το προηγούμενο και έτσι διατηρεί την αξιοπιστία του περιεχομένου τους. Επιπλέον, τα μπλοκ που είναι αποθηκευμένα στη βάση δεδομένων δε μπορούν να υποστούν καμία διαφοροποίηση.

- **Ασφάλεια**: Κάνοντας χρήση δημοσίου κλειδιού, η τεχνολογία Blockchain δημιουργεί ένα ασφαλές περιβάλλον απέναντι σε επιθέσεις. Ακόμα, αυξημένη ασφάλεια προσδίδουν και οι μηχανισμοί συναίνεσης που χρησιμοποιεί η συγκεκριμένη τεχνολογία.

- **Ανωνυμία**: Η τεχνολογία Blockchain παρέχει ανωνυμία στον κάθε κόμβο ώστε να εξασφαλίσει την ιδιωτικότητά του.

- **Μείωση Κόστους**: Σε ένα επικεντρωμένο μοντέλο απαιτείται υψηλό επίπεδο και πλήρες πλαίσιο εξοπλισμού και προγραμματισμού για τη συναρμολόγηση του κεντρικού διακομιστή. Μέσω του Blockchain επιτυγχάνεται η μείωση του κόστους που σχετίζεται με την τοποθέτηση και την υποστήριξη τεράστιων κεντρικών διακομιστών καθώς χρησιμοποιεί επεξεργαστική ισχύ από εκπομπές μηχανημάτων.
- **Αυτονομία**

2.3.4 Smart Contracts

Το Smart contract είναι ένα ψηφιακό συμβόλαιο και ορίζεται ως ένα πρόγραμμα υπολογιστή που εκτελείται στο blockchain. Μπορεί να θεωρηθεί ως αξιόπιστος τρίτος μεταξύ των μη έμπιστων συμμετεχόντων σε μία συναλλαγή. Ένα έξυπνο συμβόλαιο δύναται να δημιουργηθεί και να είναι διαθέσιμο για χρήση από οποιονδήποτε κόμβο στο δίκτυο από τη στιγμή που θα πραγματοποιήσει μία συναλλαγή στο blockchain. Μόλις αυτό εισέλθει στο δίκτυο, τότε αυτόματα ο προγραμματιστικός κώδικάς του είναι σταθερός και δε μπορεί να ενημερωθεί. Μόλις το smart contract αναπτυχθεί (γίνει deployed) του ανατίθεται μία διεύθυνση των 160 bit. Με αυτόν τον τρόπο, κάθε φορά που εκτελείται μία συναλλαγή χρησιμοποιώντας μία διεύθυνση εκτελείται και το smart contract που της αντιστοιχεί. Για τη συντήρηση του δικτύου είναι υπεύθυνο ένα δίκτυο κόμβων που ονομάζονται εξορύχοι (miners). Οι miners εκτελούν τα smart contracts, καταλήγουν σε συναίνεση σχετικά με το αποτέλεσμα της εκτέλεσης και ενημερώνουν ανάλογα το blockchain.[11]

2.3.5 Smart contract Platforms

Τα smart contracts έχουν τη δυνατότητα να αναπτυχθούν σε διαφορετικές πλατφόρμες. Η κάθε πλατφόρμα blockchain προσφέρει διαφορετικές δυνατότητες για την ανάπτυξη έξυπνων συμβολαίων. Ορισμένες από τις πιο γνωστές είναι το Ethereum, το Bitcoin και το Hyperledger. Το Ethereum είναι ένα public blockchain, αλλά και κρυπτονόμισμα. Συνεπώς, είναι ικανό να μεταφέρει νομίσματα, αλλά και να υποστηρίξει την κατασκευή και λειτουργία πολύπλοκων εφαρμογών που βασίζονται σε έξυπνα συμβόλαια στο blockchain. Το νόμισμα της πλατφόρμας αυτής ονομάζεται Ether. Το Ethereum έχει δύο βασικούς λογαριασμούς, έναν για το συμβόλαιο και έναν για τους εξωτερικούς χρήστες. Ο πρώτος ελέγχεται από τη λογική του κώδικα του έξυπνου συμβολαίου και απαρτίζεται από αποθηκευτικό χώρο, μία κατάσταση και το χρηματικό υπόλοιπο. Ο λογαριασμός του κάθε χρήστη ελέγχεται μέσω ενός ιδιωτικού κλειδιού που του ανήκει και διατηρεί το υπόλοιπο του λογαριασμού του. Μέσω αυτού του λογαριασμού μπορούν να γίνουν μεταφορές χρημάτων, αλλά και να πραγματοποιηθεί κάποια αναφορά σε smart contract.

Συγκεκριμένα για το Ethereum υπάρχει η εικονική μηχανή Ethereum (Ethereum Virtual Machine - E.V.M.), η οποία εκτελεί τα έξυπνα συμβόλαια. Η EVM ερμηνεύει τον πηγαίο κώδικα των smart contracts ο οποίος έχει αρχικά μεταγλωττιστεί σε μορφή bytecode. Τα έξυπνα συμβόλαια στο Ethereum μπορούν να αναπτυχθούν σε διάφορες Turing-complete γλώσσες όπως είναι η Solidity.[11]

2.3.6 Consensus Mechanisms

Διάδοση δεδομένων διαμέσου των κόμβων ενός blockchain δικτύου συνεπάγεται έλλειψη κεντρικού υπεύθυνου για την επίλυση διαφορών ή τη προστασία από εισβολές. Δηλαδή σε αποκεντρωμένα δίκτυα όπως είναι το blockchain δημιουργείται η ανάγκη για ένα μηχανισμό παρακολούθησης των κεφαλαίων και που ο οποίος θα εγγυάται αδιαμφισβήτητη ανταλλαγή κεφαλαίων για την αποτροπή απάτης. Ο μηχανισμός αυτός ονομάζεται μηχανισμός συναίνεσης και βασίζεται σε δύο γενικές αρχές. Πρώτον, όλοι οι κόμβοι πρέπει να συναινέσουν σε ένα κοινό πρωτόκολλο ενημέρωσης περιεχομένου για τη βάση δεδομένων ώστε να διατηρεί μια σταθερή κατάσταση. Δεύτερον, πρέπει να υπάρχει συγκατάθεση της πλειοψηφίας ώστε να μπορέσει ένα block να γίνει δεκτό στο δίκτυο blockchain. Ένας μηχανισμός συναίνεσης (Consensus Mechanism), λοιπόν, χρησιμοποιείται για τη δημιουργία block τα οποία προστίθενται στην υπάρχουσα βάση δεδομένων για μελλοντική χρήση.

Υπάρχουν αρκετοί μηχανισμοί συναίνεσης, από τους οποίους χρησιμοποιούνται συχνότερα οι Proof of Stake (PoS), Proof of Work (PoW), Practical Byzantine Fault Tolerance (PBFT) και Delegated Byzantine Fault Tolerance (dBFT). Διαχωριστικός παράγοντας αποτελεί ο τρόπος με τον οποίο γίνεται η ανάθεση και η επιβράβευση της επαλήθευσης των συναλλαγών. Το πρωτόκολλο **Proof of Work**, για παράδειγμα, το οποίο αποτελεί ένα από τα πρώτα πρωτόκολλα συναίνεσης που χρησιμοποιήθηκαν και βασίζεται σε υπολογιστικό φορτίο, απαιτεί από τους miners να βρουν μία λύση σε ένα πρόβλημα (mining). Με άλλα λόγια, είναι ένα κομμάτι δεδομένων που είναι δύσκολο (δαπανηρό ή χρονοβόρο) να παραχθεί αλλά εύκολο να επαληθευτεί από άλλους και το οποίο ικανοποιεί ορισμένες απαιτήσεις. Η απόδοση επομένως του μηχανισμού είναι αρκετά χαμηλή και έχει υπολογισθεί ότι δεν είναι κατάλληλη για πολύ μεγάλα οικοσυστήματα. Πολλά κρυπτονομίσματα χρησιμοποιούν μια παραλλαγή του πρωτοκόλλου αυτού. Για να μειωθεί το υψηλό κόστος των πόρων της εξόρυξης, προτάθηκε το πρωτόκολλο **Proof of Stake**, το οποίο εκχωρεί βαθμό δυσκολίας στο πρόβλημα ανάλογο με το μερίδιο που έχει ο κάθε εξορύκτης στο δίκτυο. Ένα τέτοιο πρωτόκολλο μπορεί να παρέχει αυξημένη προστασία από κακόβουλη επίθεση στο δίκτυο, καθώς απαιτεί πολύ υψηλό κόστος επίθεσης και επομένως μειώνει και τα κίνητρα. Ορισμένα πρωτόκολλα συναίνεσης όπως το dBFT και το PBFT βασίζονται στην επικοινωνία μεταξύ διαφορετικών κόμβων και χρησιμοποιούνται κυρίως σε ιδιωτικές αλυσίδες που διαθέτουν κόμβους ελέγχου ταυτότητας. [30]

2.3.7 Blockchain Oracles

Η ιδιότητα που καθιστά την τεχνολογία blockchain εξαιρετικά ασφαλή και αξιόπιστη είναι η απομόνωσή του δικτύου. Το τελευταίο χρειάζεται να σχηματίσει συναίνεση μόνο σε ένα πολύ βασικό σύνολο δυαδικών ερωτήσεων (αληθές/ψευδές) χρησιμοποιώντας δεδομένα που είναι ήδη αποθηκευμένα στη βάση του. Επομένως για να διατηρηθεί η απομόνωσή του, όπως ένας υπολογιστής χωρίς σύνδεση στο Διαδίκτυο, έτσι και ένα blockchain δε περιέχει κάποια ενσωματωμένη λειτουργικότητα που να δύναται να αντλήσει ή να προωθήσει δεδομένα σε οποιοδήποτε εξωτερικό σύστημα. Ωστόσο, η ιδιότητα πρόσβασης σε εξωτερικά δεδομένα είναι ζωτικής σημασίας και

καθορίζει τη σωστή λειτουργικότητα της πλειονότητας των έξυπνων συμβολαίων. Το πρόβλημα αυτό αναφέρεται και ως *Oracle Problem*. Τη λύση σε αυτό, δηλαδή τη γεφύρωση της σύνδεσης μεταξύ του blockchain (on-chain) και του εξωτερικού κόσμου (off-chain) δίνει ένα πρόσθετο και ξεχωριστό κομμάτι υποδομής που ονομάζεται oracle.

Blockchain Oracles ονομάζονται οι υπηρεσίες τρίτων που παρέχουν εξωτερικές πληροφορίες στα smart contracts. Τα Oracles αποτελούν καθοριστικής σημασίας παράγοντες για τα blockchain, καθώς διευρύνουν τις δυνατότητες λειτουργίας των έξυπνων συμβολαίων. Άξιο επισήμανσης είναι ότι τα oracles δεν αποτελούν αυθεντική πηγή δεδομένων, αλλά λειτουργούν ως μηχανές αναζήτησης, επαλήθευσης και αυθεντικοποίησης των εξωτερικών πηγών και κατ΄ επέκταση αναμετάδοσης των πληροφοριών. Τα δεδομένα που μεταδίδουν μπορούν να βρεθούν σε πολλές μορφές, όπως πληροφορίες, μήνυμα επιτυχούς ολοκλήρωσης μιας πληρωμής ή και μετρήσεις που λαμβάνονται από διάφορους αισθητήρες. Για να επιτευχθεί η επικοινωνία που επιτρέπουν τα oracles είναι απαραίτητη η επίκληση έξυπνου συμβολαίου, αλλά και δαπάνη πόρων του δικτύου.

Υπάρχουν αρκετοί τύποι Oracles οι οποίοι ταξινομούνται βάσει των παρακάτω κριτηρίων:

- Αν η **πηγή** (Source) των δεδομένων είναι λογισμικό, υλικό ή άνθρωπος.
- Αν η **κατεύθυνση των πληροφοριών** είναι εισερχόμενη ή εξερχόμενη.
- Αν η **εμπιστοσύνη** είναι συγκεντρωτική (centralized) ή αποκεντρωμένη (decentralized).

Ένα Oracle μπορεί να περιγράφεται από περισσότερους από έναν τύπο. [13]

2.4 Τεχνολογίες

2.4.1 HTML

Η **HTML (Hyper Text Markup Language)** είναι ένα από τα βασικά δομικά στοιχεία του Ιστού, καθώς καθορίζει τη σημασία και τη δομή του περιεχομένου του. Μπορεί επίσης να ενσωματώσει εφαρμογές γραμμένες σε γλώσσες όπως JavaScript που περιγράφει την λειτουργικότητα-/συμπεριφορά μιας ιστοσελίδας και Style Sheets (CSS) που περιγράφει την εμφάνισή/παρουσίασή της για συνολική αναπαράσταση. [4]

2.4.2 Javascript

Η **JavaScript (JS)** είναι μία γενικού σκοπού, υψηλού επιπέδου, αντικειμενοστραφής γλώσσα προγραμματισμού. Χρησιμοποιείται συχνά για τη δημιουργία ιστοσελίδων, αλλά και από πολλά περιβάλλοντα χωρίς πρόγραμμα περιήγησης όπως το Node.js, το Apache CouchDB και το Adobe Acrobat. [6]

2.4.3 Node.js

Το **Node.js** είναι ένα ανοιχτού κώδικα και cross-platform runtime περιβάλλον JavaScript. Είναι ένα δημοφιλές εργαλείο για σχεδόν κάθε είδους έργο. [5]

2.4.4 Express

Το **Express** είναι ένα μινιμαλιστικό και ευέλικτο πλαίσιο εφαρμογών ιστού Node.js που παρέχει ένα ισχυρό σύνολο δυνατοτήτων για εφαρμογές ιστού και για κινητές συσκευές. Παρέχει πλήθος βοηθητικών μεθόδων HTTP και middleware με σκοπό την εύκολη δημιουργία ενός ισχυρού API [3].

2.4.5 JSON

Ένα **JSON (JavaScript Object Notation)** αρχείο ενδείκνυται για κοινή χρήση δεδομένων. Χρσιμοποιεί κείμενο αναγνώσιμο από τον άνθρωπο για την αποθήκευση και τη μετάδοση δεδομένων. Η μορφή του κειμένου καθιστά επιπλέον εύκολο για τις μηχανές να αναλύουν και να δημιουργούν τα δεδομένα.Το JSON βασίζεται σε δύο δομές:

- Μια συλλογή από ζεύγη ονόματος-τιμών. Σε πολλές γλώσσες αυτή η μορφή πραγματοποιείται μέσω των αντικειμένων, εγγραφών, δομών, πίνακα κατακερματισμού, λίστα με πλήκτρα ή συσχετικό πίνακας.
- Μια ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες, αυτό πραγματοποιείται ως πίνακας, διάνυσμα, λίστα ή ακολουθία.

Αυτές είναι καθολικές δομές δεδομένων. Σχεδόν όλες οι σύγχρονες γλώσσες προγραμματισμού τις υποστηρίζουν με τη μία ή την άλλη μορφή. [7]

2.4.6 Solidity

Η **Solidity** είναι μια αντικειμενοστραφής γλώσσα υψηλού επιπέδου που χρησιμοποιείται για την υλοποίηση έξυπνων συμβολαίων και έχει σχεδιαστεί για να στοχεύει την εικονική μηχανή Ethereum (EVM). [8]

2.4.7 Infura

Το **Infura** είναι σουίτα ανάπτυξης blockchain με API υψηλής διαθεσιμότητας και εργαλεία προγραμματιστών. Μέσω αυτού παρέχεται γρήγορη και αξιόπιστη πρόσβαση στα δίκτυα Ethereum και IPFS. [2]

2.4.8 Metamask

Το **Metamask** είναι ένα ψηφιακό πορτοφόλι το οποίο παρέχει τον απλούστερο και ασφαλέστερο τρόπο σύνδεσης με εφαρμογές που βασίζονται σε blockchain. [1]

2.4.9 Prometheus

Ο **Προμηθέας** (Prometheus) είναι ένα εργαλείο παρακολούθησης μετρικών ανοιχτού κώδικα το οποίο διαθέτει πολλαπλούς τρόπους οπτικοποίησης δεδομένων. Εφαρμόζει ένα μοντέλο δεδομένων υψηλών διαστάσεων, όπου οι χρονικές σειρές προσδιορίζονται από ένα όνομα μέτρησης και ένα σύνολο ζευγών κλειδιών-τιμών.[27]

2.4.10 Remix IDE

Το **Remix IDE** είναι μια εφαρμογή ανοιχτού κώδικα που χρησιμοποιείται για ολόκληρη τη διαδικασία ανάπτυξης συμβολαίων καθώς και λειτουργεί ως μία εύκολη προσέγγιση για την εκμάθηση και τη διδασκαλία του Ethereum.[9]

2.4.11 Etherscan

Το **Etherscan** αποτελεί πλατφόρμα εξερεύνησης, αναζήτησης, API και ανάλυσης blockchain για το Ethereum. [16]

2.5 Σχετικές Εργασίες

Η τεχνολογία Blockchain δεν έχει πολύ καιρό που έχει έρθει στο προσκήνιο. Λόγω της καινοτομίας της, λοιπόν, υπάρχει πολύς χώρος ανάπτυξής της. Παρά το μικρό χρονικό διάστημα ωρίμανσής της έχουν γίνει αρκετές προσπάθειες συνεργασίας της με τον τομέα διαχείρισης SLA. Στη συνέχεια, παρουσιάζονται ορισμένες εργασίες που πραγματεύονται θέματα σχετικού περιεχομένου με την παρούσα.

Στο [20] οι Kochovski et al. εισάγουν μία νέα αρχιτεκτονική βασισμένη στο Έξυπνο Συμβόλαιο που παρέχει διαχείριση SLA μεταξύ σχετικών οντοτήτων και παραγόντων σε ένα αποκεντρωμένο υπολογιστικό περιβάλλον (Εικονικές Μηχανές, καταναλωτές υπηρεσιών Cloud και παρόχους Cloud). Κατορθώνουν διαχείριση SLA με χαμηλού κόστους λειτουργίες σε δυναμικές και αποκεντρωμένες ομοσπονδίες Edge-to-Cloud. Οι Brito Gonçalves et al. στο [15] εισηγούνται μια λύση που εξουσιοδοτείται από έξυπνα συμβόλαια με σκοπό την απλοποίηση και αυτοματοποίηση της παρακολούθησης των επιπέδων συμμόρφωσης των SLA και εκτελούν πειράματα χρησιμοποιώντας δεδομένα από πραγματική υποδομή δικτύου στη Βραζιλία. Παρουσιάζουν επιπλέον την αλληλεπίδραση αυτού του συστήματος με ένα αποκεντρωμένο σύστημα αρχείων (IPFS - Interplanetary File System) που θα αποθηκεύει τις αναφορές που δημιουργούνται με κατανεμημένο και διαφανή τρόπο. Στο [12] μελετάται μία προσέγγιση βασισμένη σε blockchain για την αξιολόγηση της

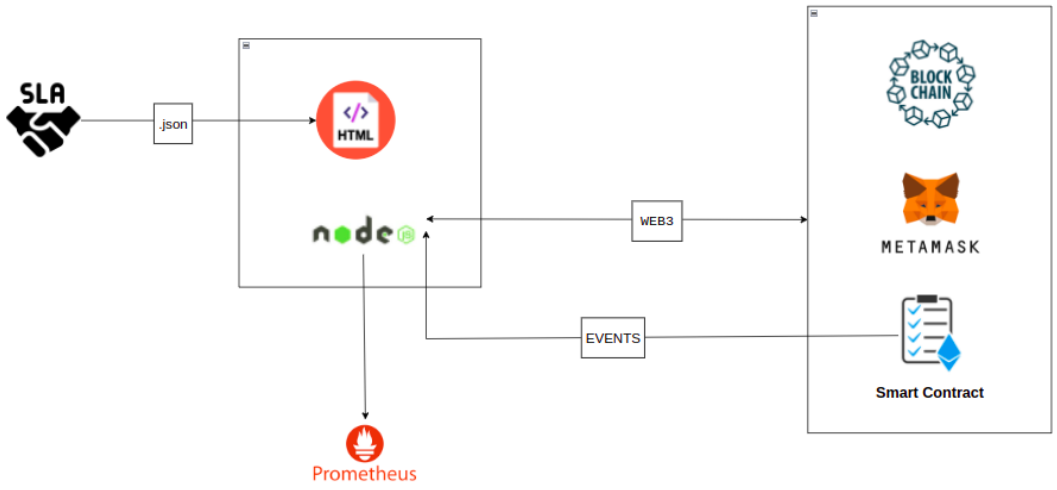
συμμόρφωσης με το SLA και την επιβολή κυρώσεων. Για την επικύρωση της αξιοπιστίας της προτεινόμενης λύσης γίνεται χρήση μίας μεθόδου διαγνωστικής ακρίβειας. Αξιολογείται, επίσης, το Hyperledger Fabric ως προς το κατά πόσο είναι εφικτό να χρησιμοποιηθεί ως υποκείμενη υποδομή blockchain όσον αφορά την καθυστέρηση και τα ποσοστά επιτυχίας/αποτυχίας συναλλαγών. Στα [17] και [31] προτείνεται ο σχεδιασμός και η υλοποίηση Smart Contracts τα οποία απλοποιούν και αυτοματοποιούν τη διαδικασία αποζημίωσης σε περίπτωση παραβίασης SLA, με το δεύτερο να αξιοποιεί την εικονικοποίηση λειτουργιών δικτύου (Network Function Virtualization - NFV) ως απόδειξη της ιδέας (proof-of-concept). Οι Amir Teshome Wonjiga et al. στο [35] προτείνουν ένα SLA το οποίο θα προσφέρεται από τον πάροχο και θα εγγυάται την ακεραιότητα των δεδομένων των καταναλωτών. Ακόμα, μελετάται η επαλήθευση του SLA χρησιμοποιώντας μια μέθοδο ελέγχου ακεραιότητας που βασίζεται σε μία κατανεμημένη βάση δεδομένων. Στο [24] παρουσιάζεται μια ιδέα που βασίζεται σε blockchain και έξυπνα συμβόλαια, η οποία υλοποιεί την παρακολούθηση SLA, καθώς και τις υπηρεσίες χρέωσης cloud με αποκεντρωμένο, διαφανή τρόπο, μειώνοντας την ανάγκη για εμπιστοσύνη του πελάτη στον πάροχο. Στο [33] οι Rafael Brundo Uriarte et al. προτείνουν ένα πλαίσιο για τη διαχείριση δυναμικών SLA με κατανεμημένο τρόπο, βασιζόμενο στον φορμαλισμό των Συμφωνιών Επιπέδου Υπηρεσίας, οι οποίες απεικονίζονται σε έξυπνα συμβόλαια. Το προτεινόμενο πλαίσιο εισάγει την έννοια ενός ανοιχτού κατανεμημένου cloud που προσφέρει διαχειρίσιμες και δυναμικές υπηρεσίες στους πελάτες, δίνοντάς τους τη δυνατότητα να μειώσουν το κόστος και να αυξήσουν την ευελιξία στη διαχείριση πόρων. Οι Kashif Mehboob Khan et al. στο [22] παρουσιάζουν μια κατανεμημένη υποδομή που βασίζεται σε blockchain η οποία αξιοποιεί θεμελιώδεις ιδιότητες δικτύων Αλυσίδας-Κορμού για την επίτευξη αμετάβλητης και αξιόπιστης παρακολούθησης SLA σε υπηρεσίες cloud. Εξετάζονται, ακόμη, διάφοροι παράγοντες του blockchain τα οποία είναι κρίσιμα για την επίτευξη της βέλτιστης απόδοσης, αλλά και διερευνώνται βασικά χαρακτηριστικά αυτών των παραγόντων και το αντίκτυπό που έχουν στη συμπεριφορά του συστήματος για την περαιτέρω κλιμάκωσή του σε περιπτώσεις αυξημένης χρήσης υπηρεσιών. Στο [37] εισηγείται ένα μοντέλο μάρτυρα χρησιμοποιώντας τη θεωρία παιγνίων και τις τεχνικές έξυπνων συμβολαίων με σκοπό την επίλυση του προβλήματος της αξιπιστίας πιθανών παραβιάσεων στο SLA προτού γίνει η καταγραφή τους στο blockchain. Το προτεινόμενο μοντέλο επεκτείνει το υπάρχον μοντέλο υπηρεσίας με έναν νέο ρόλο που ονομάζεται «μάρτυρας» για τον εντοπισμό και την αναφορά παραβιάσεων υπηρεσιών.

Κεφάλαιο 3

Σχεδιασμός και Υλοποίηση Συστήματος

3.1 Γενική περιγραφή αρχιτεκτονικής συστήματος

Σκοπός της παρούσας εργασίας είναι η επιτήρηση on chain ορισμένων μετρικών που παρέχονται στις Συμφωνίες Επιπέδου Υπηρεσίας και η έμμεση απεικόνιση του SLA σε Δίκτυα Αλυσίδας-Κορμού. Συγκεκριμένα, έχουν επιλεγεί οι μετρικές που αφορούν τη διαθεσιμότητα (availability), τα http errors και τον χρόνο απόκρισης (response time). Ακόμα, γίνεται και συγκριτική μελέτη ως προς την απόδοση διαφόρων Ethereum πειραματικών δικτύων, αλλά και τεσσάρων πιθανών αλγορίθμων ελέγχου παραβιάσεων των όρων του SLA.



Σχήμα 3.1: Αρχιτεκτονική Συστήματος

Στο σχήμα 3.1 παρουσιάζεται η αρχιτεκτονική του συστήματος. Κάθε SLA που επιθυμούμε να απεικονίσουμε στο blockchain δίκτυο εισάγεται αρχικά με τη μορφή JSON αρχείου μέσω μίας σελίδας που έχει δημιουργηθεί γιάυτό τον σκοπό με τη βοήθεια της HTML και της Javascript. Στη συνέχεια, συλλέγονται οι πληροφορίες του SLA σε ένα αρχείο Node.js το οποίο δρα ως αποκεντρωμένη εφαρμογή (Decentralized App). Το αρχείο Node.js, όντας συνδεδεμένο με το έξυπνο συμβόλαιο που έχει δημιουργηθεί για τους σκοπούς της εργασίας, φροντίζει να γίνεται αυτόματα deployment για το κάθε SLA στο δίκτυο αλυσίδας-κορμού της επιλογής μας. Το δίκτυο αυτό επιλέχθηκε να είναι το Ethereum το οποίο παρέχεται από τη πλατφόρμα του Infura και για να μπορέσουν να γίνουν οι απαραίτητες συναλλαγές, δημιουργήθηκε λογαριασμός στο Metamask. Συγκεκριμένα, επιλέχθηκαν τα Ethereum Testnets Rinkeby, Goerli και Kovan. Τέλος, για την επιτήρηση του SLA και τις συγκριτικές μελέτες που πραγματοποιούνται, γίνεται χρήση της πλατφόρμας του Prometheus.

Για να επιτευχθούν τα παραπάνω, κατασκευάστηκαν δύο βασικά αρχεία, ένα σε HTML και ένα Node.js, αλλά και πέντε JSON αρχεία που αντιπροσωπεύουν πέντε διαφορετικά SLA. Τα αρχεία που δημιουργήθηκαν, καθώς και η λογική πίσω από αυτά αναλύονται διεξοδικά στις επόμενες ενότητες.

3.2 Συμφωνία Επιπέδου Υπηρεσίας

Η μορφή του SLA επιλέχθηκε να είναι σε αρχείο json που θα περιέχει πληροφορίες που αφορούν:

1. Το ID του SLA.
2. Το όνομα και το ID του πελάτη και του παρόχου της υπηρεσίας.
3. Τις τρεις μετρικές που επιθυμούμε σε αυτή την περίπτωση να παρακολουθήσουμε. Για τη διαθεσιμότητα εισάγουμε το ποσοστό που θέλουμε να ικανοποιείται στο ελάχιστο. Για τα σφάλματα HTTP και το χρόνο απόκρισης εισάγουμε σαν πληροφορία τις μέγιστες τιμές που μπορούν να λάβουν σε μορφή ποσοστού και σε miliseconds αντίστοιχα.

Οι μεταβλητές που ήταν απαραίτητες για τη διαικπεραίωση της εργασίας ήταν η ταυτότητα του SLA και οι τρεις μετρικές που έχουν επιλεγεί. Τα υπόλοιπα δεδομένα έχουν ορισθεί κυρίως για λόγους κατανόησης των στοιχείων που επιβάλλεται να υπάρχουν σε μία συμφωνία και λόγω αυτού δε μας επηρεάζουν οι τιμές τους. Τα παραπάνω μπορούν να διακριθούν και στο σχήμα 3.2 όπου φαίνεται η μορφή του json αρχείου που δημιουργήθηκε, αλλά και στον πίνακα 3.1, οπού αναλύονται οι τιμές που χρησιμοποιήθηκαν για το κάθε SLA.

Στο Σχήμα 3.2 παρέχεται μία απλή περιγραφή των βασικών πληροφοριών που πρέπει να περιέχει ένα SLA. Η επιλογή των μετρικών έγινε βάσει της σπουδαιότητάς τους και τη συχνότητα εμφάνισής τους σε τέτοιες συμφωνίες. Οι υπόλοιπες πληροφορίες αφορούν τα μέλη της συμφωνίας και τη ταυτότητά της, δεδομένα δηλαδή που δε μπορούν να απουσιάζουν από ένα συμβόλαιο. Οι πληροφορίες αυτές εξαρτώνται κυρίως από τις ανάγκες της συμφωνίας και, συγκεκριμένα εδώ, επιλέχθηκε αυτό το μοντέλο καθώς εξυπηρετούσε παράλληλα και τις ανάγκες της εργασίας.

```
{
    "SlaID":"",
    "Provider":
        [
            {
                "name": "",
                "id" : ""
            }
        ],
    "Customer":
        [
            {
                "name": "",
                "id" : ""
            }
        ],
    "availability":"",
    "responseTime":"",
    "httpErrors":""
}
```

Σχήμα 3.2: SLA Template

	SLA 1	SLA 2	SLA 3	SLA 4	SLA 5
SLA id	1	2	3	4	5
Provider name	P1	P2	P3	P4	P5
Provider id	PID1	PID2	PID3	PID4	PID5
Customer name	C1	C2	C3	C4	C5
Customer id	CID1	CID2	CID3	CID4	CID5
Availability	99	98	97	97	98
Response Time	900	1100	1200	1000	1300
HTTP errors	10	29	30	35	20

Πίνακας 3.1: Πληροφορίες του κάθε SLA που δημιουργήθηκε.

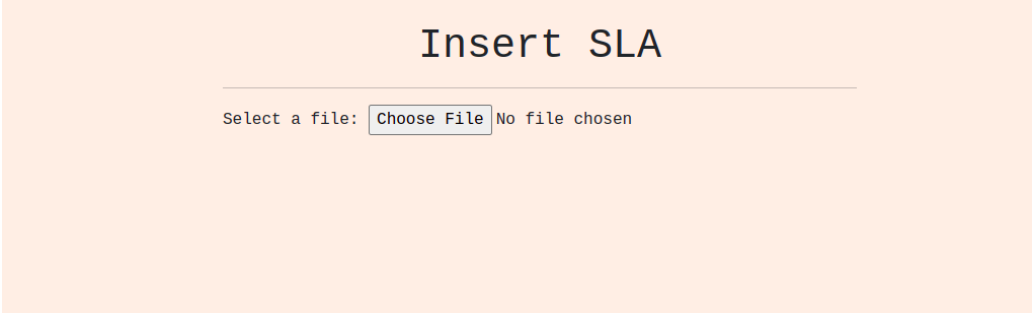
3.3 HTML page

Στην παρούσα ενότητα περιγράφεται η σελίδα στην οποία γίνεται η εισαγωγή του SLA. Η σελίδα αυτή κάνει χρήση της HTML και της Javascript, έχει τη μορφή που φαίνεται στο σχήμα 3.4 και το σημαντικό μέρος του κώδικά της αναλύεται στο σχήμα 3.3.

Με τη βοήθεια των HTML Forms δημιουργήθηκε μια φόρμα που επιτρέπει την εισαγωγή ενός αρχείου. Όπως έχει προαναφερθεί, το αρχείο αυτό θα περιέχει τα δεδομένα που θέλουμε να λάβουμε από το SLA σε μορφή JSON. Σε κάθε είσοδο διαφορετικού JSON file ενεργοποιείται η συνάρτηση 'onChange(event)' στην οποία γίνεται η ανάγνωση του αρχείου και καλείται εν συνεχεία η συνάρτηση 'onReaderLoad(event)' η οποία καλεί με τη σειρά της την 'HttpPost¨. Η τελευταία αποστέλει ένα Post Request στη διεύθυνση 'http://localhost:8080/jsons' με δεδομένα αυτά που έλαβε από το αρχείο του SLA. Η διεύθυνση στην οποία στάλθηκαν τα δεδομένα είναι αυτή στην οποία θα αναζητά το αρχείο NodeJS τις πληροφορίες του SLA.

```
<body>
    <br>
    <div class="container" style="width: 650px;">
        <h1 class="text-center">Insert SLA</h1>
        <hr/>
        <form id="form"> <!-- Δημιουργία φόρμας html για εισαγωγή αρχείου -->
            <label for="myfile">Select a file:</label> <!-- Επιλογή αρχείου (Εδώ θα εισαχθεί το json αρχείο που περιέχει το SLA) -->
            <input type="file" id="myfile" name="myfile"><br>

            <script>
                (function(){

                    function HttpPost(TheURL,data){
                        let xhr = new XMLHttpRequest();
                        xhr.open("POST", TheURL);
                        xhr.setRequestHeader("Accept", "application/json");
                        xhr.setRequestHeader("Content-Type", "application/json");

                        xhr.onreadystatechange = function () {
                        if (xhr.readyState === 4) {
                            console.log(xhr.responseText);
                        }};
                        xhr.send(data);
                    }

                    //Μόλις ενεργοποιηθεί γίνεται ανάγνωση του αρχείου και καλείται η συνάρτηση "onReaderLoad(event)".
                    function onChange(event) {
                        var reader = new FileReader();
                        reader.onload = onReaderLoad;
                    }

                    //Δημιουργεί ένα Post Request στη σελίδα "http://localhost:8080/jsons" όπου γίνεται η ανάγνωση σε επόμενο στάδιο
                    //του json αρχείου από το nodejs αρχείο.
                    function onReaderLoad(event){
                        console.log(event.target.result);
                        HttpPost("http://localhost:8080/jsons",event.target.result);
                    }
                    // Σε κάθε είσοδο νέου αρχείου ενεργοποιεί τη συνάρτηση "onChange(event)".
                    document.getElementById('myfile').addEventListener('change', onChange);

                }());
            </script>
        </form>
    </div>
</body>
```

Σχήμα 3.3: Κώδικας HTML με τον οποίο γίνεται η εισαγωγή του SLA στο σύστημα



Σχήμα 3.4: Σελίδα εισαγωγής SLA με τη μορφή αρχείου

3.4 Έξυπνο Συμβόλαιο

Για την ανάπτυξη των έξυπνων συμβολαίων χρησιμοποιήθηκε η γλώσσα Solidity και για τη μεταγλώττιση και την εξαγωγή των πληροφοριών τους (abi, bytecode) χρησιμοποιήθηκε η εφαρμογή Remix.

```
constructor(string memory _sla,string memory _pn,string memory _pid,string memory _cn, string memory _cid, string memory _a, string memory _rt, string memory _http ) public {
    slaID = _sla;
    Pname = _pn;
    Pid = _pid;
    Cname = _cn;
    Cid = _cid;
    availability = _a;
    responseTime = _rt;
    httpErrors = _http;

}
```

Σχήμα 3.5: Κώδικας Constructor των Smart Contracts

Για τις ανάγκες της εργασίας δημιουργήθηκαν τέσσερα έξυπνα συμβόλαια με βασική τους διαφορά τον αλγόριθμο που ελέγχει για πιθανές παραβιάσεις. Όλα όμως τα συμβόλαια έχουν τον ίδιο κατασκευαστή (constructor), ο οποίος φαίνεται στο σχήμα 3.5. Κάθε φορά που εισάγουμε ένα SLA προωθούμε τα δεδομένα του αυτόματα στον constructor του smart contract μέσω του deployment του συμβολαίου και έτσι γίνεται η απεικόνισή του. Τα δεδομένα αποθηκεύονται σε μεταβλητές του συμβολαίου και χρησιμοποιούνται μετέπειτα για τον έλεγχο παραβιάσεων.

Στις υποενότητες που ακολουθούν αναλύουμε τον τρόπο με τον οποίο έχει υλοποιηθεί καθένα από τα τέσσερα προαναφερθέντα διαφορετικά συμβόλαια.

3.4.1 Αλγόριθμος I

Ο αλγόριθμος που θα περιγραφεί στην παρούσα υποενότητα φαίνεται και σε μορφή κώδικα στο σχήμα 3.6. Συγκεκριμένα, παρουσιάζεται μόνο η συνάρτηση ελέγχου, καθώς είναι η μόνη που διαφέρει σε σύγκριση με τους υπόλοιπους αλγορίθμους.

Για τον έλεγχο των παραβιάσεων των μετρικών κατασκευάστηκε η συνάρτηση "check" η οποία δέχεται τρεις (3) εισόδους, μία για κάθε μετρική που επιθυμούμε να παρακολουθήσουμε. Οι τιμές "OracleAvailability", "OracleHttpErrors" "OracleResponseTime" είναι οι τιμές που προκύπτουν από πραγματικές μετρήσεις των μετρικών της διαθεσιμότητας, των HTTP σφαλμάτων και του χρόνου απόκρισης αντίστοιχα και που λαμβάνονται από κάποιο Oracle. Για τους σκοπούς της άσκησης έχει επιλεγεί οι τιμές αυτές να λαμβάνονται από ένα αυτοσχέδιο Oracle το οποίο περιγράφεται σε επόμενη ενότητα.

Ο αλγόριθμος είναι σχεδιασμένος με τέτοιο τρόπο ώστε να λαμβάνει δέκα μετρήσεις για την κάθε μετρική, να υπολογίζει το μέσο όρο τους και στη συνέχεια ανάλογα με τα όρια που έχουν δοθεί από το SLA να αποφασίζει βάσει του μέσου όρου αν έχει γίνει κάποια παραβίαση. Στην περίπτωση που έχει γίνει, ενεργοποιεί το event που αντιστοιχεί στη μετρική στην οποία παρουσιάστηκε παραβίαση των όρων του συμβολαίου. Πιο αναλυτικά, η μεταβλητή "times", που φαίνεται στον κώδικα του σχήματος 3.6, χρησιμοποιείται σαν μετρητής και ελέγχει πόσες φορές έχει κληθεί η συνάρτηση ελέγχου και άρα έχουν δοθεί νέες μετρήσεις για τις μετρικές. Η επιλογή του αριθμού δέκα σαν όριο έχει γίνει καθαρά λόγω χρονικών περιορισμών στο πλαίσιο της εργασίας. Σε πραγματικές συνθήκες το νούμερο αυτό θα ήταν αρκετά μεγαλύτερο και στην ουσία εξαρτάται από τις επιθυμίες των μελών της συμφωνίας.

Επιπλέον, η συνάρτηση "stringToNumber(str)" έχει δημιουργηθεί για τη μετατροπή μεταβλητών τύπου string σε μορφή αριθμού. Αυτό είναι απαραίτητο για να μπορεί να γίνει η σύγκριση μεταξύ της μετρικής η οποία λαμβάνεται σε μορφή string και των μετρήσεων οι οποίες είναι αριθμητικού τύπου.

3.4.2 Αλγόριθμοι II, III και IV

Ο αλγόριθμοι που αναλύονται στη συνέχεια παρουσιάζονται και σε μορφή κώδικα στα σχήματα 3.7, 3.8, 3.9. Για τον έλεγχο των παραβιάσεων των μετρικών κατασκευάστηκε και σε αυτή την περίπτωση μία συνάρτηση "check" με παραμέτρους τις τιμές OracleAvailability, OracleHttpErrors OracleResponseTime, οι οποίες κατέχουν την ίδια θέση που έχουν και στον Αλγόριθμο I. Η συνάρτηση ελέγχου εδώ, κάθε φορά που καλείται ελέγχει με τον ίδιο τρόπο αν υπήρξε κάποια παράβαση των όρων. Μόλις εντοπισθεί ενημερώνεται ο αντίστοιχος μετρητής (timesA, timesB και timesC) ο οποίος μετράει πόσες φορές έχει γίνει κάποια παραβίαση (στην μετρική της διαθεσιμότητας, των σφαλμάτων HTTP και του χρόνου απόκρισης αντίστοιχα). Η διαφορά των τριών αλγορίθμων εντοπίζεται στον αριθμό των μεμονωμένων παραβιάσεων που χρειάζεται να συγκεντρωθούν ώστε να ενημερωθεί το σύστημα με event πως έγινε παραβίαση. Συγκεκριμένα, ο αλγόριθμος II ενημερώνει με event ύστερα από δύο (2) παραβιάσεις, ο αλγόριθμος III ύστερα από πέντε (5) και ο αλγόριθμος IV ύστερα από δέκα (10). Οι παραβιάσεις και τα events που αναφέρονται αφορούν την

κάθε μετρική ξεχωριστά και αφότου συγκεντρωθούν τα αντίστοιχα ποσά για τον κάθε αλγόριθμο οι μετρητές μηδενίζονται και επαναλαμβάνεται η ίδια διαδικασία.

```
function check(uint32 OracleAvailability,uint32 OracleHttpErrors,uint32 OracleResponseTime) external payable {
    if (times<9) {
        times = times + 1;
        //Υπολογισμός αθροίσματος για την κάθε μετρική
        availabilitySum = availabilitySum + OracleAvailability;
        httpErrorsSum = httpErrorsSum + OracleHttpErrors;
        responseTimeSum = responseTimeSum + OracleResponseTime;
    }
    else{
        //Υπολογισμός αθροίσματος για την κάθε μετρική
        availabilitySum = availabilitySum + OracleAvailability;
        httpErrorsSum = httpErrorsSum + OracleHttpErrors;
        responseTimeSum = responseTimeSum + OracleResponseTime;
        //Υπολογσμός Μέσου Όρου από τις 10 τιμές που ελήφθησαν για τη διαθεσιμότητα
        AMO = availabilitySum/10;
        //Έλεγχος Παραβίασης
        if (AMO<int(stringToNumber(availability))) {
            emit ViolationA(slaID);
        }
        //Μηδενισμός αθροίσματος
        availabilitySum = 0;
        //Υπολογσμός Μέσου Όρου από τις 10 τιμές που ελήφθησαν για τα HTTP ERRORS
        HMO = httpErrorsSum/10;
        //Έλεγχος Παραβίασης
        if (HMO>int(stringToNumber(httpErrors))) {
            emit ViolationB(slaID);
        }
        //Μηδενισμός αθροίσματος
        httpErrorsSum = 0;
        //Υπολογσμός Μέσου Όρου από τις 10 τιμές που ελήφθησαν για το χρόνο απόκρισης
        RTMO = responseTimeSum/10;
        //Έλεγχος Παραβίασης
        if (RTMO>int(stringToNumber(responseTime))) {
            emit ViolationC(slaID);
        }
        //Μηδενισμός αθροίσματος
        responseTimeSum = 0;
        //Μηδενισμός μετρητή
        times = 0;
    }
}
```

Σχήμα 3.6: Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου Ι

```
function check(uint32 OracleAvailability,uint32 OracleHttpErrors,uint32 OracleResponseTime) external payable {
    if (OracleAvailability<int(stringToNumber(availability))) {
        if (timesA==1){
            timesA=0;
            emit ViolationA(slaID);
        }
        else{
            timesA+=1;
        }
    }
    if (OracleHttpErrors>int(stringToNumber(httpErrors))) {
        if (timesB==1){
            timesB=0;
            emit ViolationB(slaID);
        }
        else{
            timesB+=1;
        }
    }
    if (OracleResponseTime>int(stringToNumber(responseTime))) {
        if (timesC==1){
            timesC=0;
            emit ViolationC(slaID);
        }
        else{
            timesC+=1;
        }
    }
}
```

Σχήμα 3.7: Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου II

```
function check(uint32 OracleAvailability,uint32 OracleHttpErrors,uint32 OracleResponseTime) external payable {
    if (OracleAvailability<int(stringToNumber(availability))) {
        if (timesA==4){
            timesA=0;
            emit ViolationA(slaID);
        }
        else{
            timesA+=1;
        }
    }
    if (OracleHttpErrors>int(stringToNumber(httpErrors))) {
        if (timesB==4){
            timesB=0;
            emit ViolationB(slaID);
        }
        else{
            timesB+=1;
        }
    }
    if (OracleResponseTime>int(stringToNumber(responseTime))) {
        if (timesC==4){
            timesC=0;
            emit ViolationC(slaID);
        }
        else{
            timesC+=1;
        }
    }
}
```

Σχήμα 3.8: Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου III

```
function check(uint32 OracleAvailability,uint32 OracleHttpErrors,uint32 OracleResponseTime) external payable {
    if (OracleAvailability<int(stringToNumber(availability))) {
        if (timesA==9){
            timesA=0;
            emit ViolationA(slaID);
        }
        else{
            timesA+=1;
        }
    }
    if (OracleHttpErrors>int(stringToNumber(httpErrors))) {
        if (timesB==9){
            timesB=0;
            emit ViolationB(slaID);
        }
        else{
            timesB+=1;
        }
    }
    if (OracleResponseTime>int(stringToNumber(responseTime))) {
        if (timesC==9){
            timesC=0;
            emit ViolationC(slaID);
        }
        else{
            timesC+=1;
        }
    }
}
```

Σχήμα 3.9: Συνάρτηση Ελέγχου Παραβιάσεων Αλγορίθμου IV

3.5 Εφαρμογή

Στην ενότητα αυτή αναλύονται τα περιεχόμενα του αρχείου Node.js που δημιουργήθηκε με σκοπό την ολοκλήρωση του συστήματος. Σε γενικά πλαίσια, στο αρχείο αυτό περιλαμβάνονται τα παρακάτω:

- Ορισμός πλαισίου ιστού και της εφαρμογής.
- Ορισμός web3 και σύνδεση με το Blockchain δίκτυο που χρησιμοποιείται.
- Εισαγωγή των πληροφοριών που λαμβάνονται από το αρχείο που αντιπροσωπεύει το κάθε SLA.
- Εισαγωγή έξυπνου συμβολαίου που έχει δημιουργηθεί.
- Αυτοματοποίηση του deployment για το κάθε έξυπνο συμβόλαιο που αντιστοιχεί σε κάθε νέο SLA που εισέρχεται στο σύστημα.
- Καθορισμός χρονικών παραθύρων ανάμεσα στους ελέγχους παραβιάσεων SLA, αλλά και στις ενημερώσεις της πλατφόρμας του Prometheus.
- Ορισμός αυτοσχέδιου Oracle για την παροχή μετρήσεων στους ελέγχους παραβιάσεων των έξυπνων συμβολαίων.

- Ορισμός Prometheus.
- Ορισμός μετρικής για τον Prometheus με σκοπό την επιτήρηση των SLA παραβιάσεων.

Ως πλαίσιο ιστού για το Node.js χρησιμοποιήθηκε το Express, όπως φαίνεται και στο Σχήμα 3.10. Για τον ορισμό της εφαρμογής express χρησιμοποιήθηκε το αντικείμενο app. Τέλος, η εφαρμογή ρυθμίστηκε με τέτοιο τρόπο ώστε να ακούει στη θύρα 8080 του localhost και να συνδέεται με το directory που βρίσκονται και τα υπόλοιπα αρχεία του συστήματος.

```
var express = require('express');
const app = express();

app.use(express.static(__dirname));

app.use(express.json()); // for parsing application/json

// for parsing application/x-www-form-urlencoded
app.use(
    express.urlencoded({
      extended: true
    })
);

//Binds and listens for connections on the specified host and port.
app.listen(8080, () => console.log('Server is running on http://localhost:8080'));
```

Σχήμα 3.10: Ορισμός Express Server και Application

Στο σχήμα 3.11 φαίνεται ο τρόπος με τον οποίο γίνεται η σύνδεση με το Ethereum Blockchain και τον λογαριασμό που έχει δημιουργηθεί στο Metamask. Συγκεκριμένα, για την επίτευξη της σύνδεσης με Blockchain node δημιουργήθηκε ένα Project στο Infura στο οποίο παραχωρήθηκε μοναδικό PROJECT ID. Στο σημείο "https://testnet.infura.io/v3/PROJECT ID" που φαίνεται στον κώδικα του σχήματος 3.11, το PROJECT ID είναι αυτό που παρέχεται από την πλατφόρμα του Infura και ο όρος "testnet" αντικαθίσταται κάθε φορά από το αντίστοιχο πειραματικό Ethereum δίκτυο που θέλουμε να χρησιμοποιήσουμε (rinkeby, goerli, kovan).

```
// Configuring the connection to an Ethereum node
const web3 = new Web3(new Web3.providers.HttpProvider("https://testnet.infura.io/v3/PROJECT_ID"));

const signer = web3.eth.accounts.privateKeyToAccount('private key');
web3.eth.accounts.wallet.add(signer);
```

Σχήμα 3.11: Σύνδεση με Ethereum Testnets και εισαγωγή του λογαριασμού που δημιουργήθηκε στο Metamask

Οι δύο επόμενες εντολές του σχήματος 3.11 χρησιμοποιούνται αντίστοιχα για τη σύνδεση με το λογαριασμό του Metamask και τη προσθήκη του λογαριασμού signer στο πορτοφόλι. Η σύνδεση με το Metamask γίνεται μέσω του ιδιωτικού κλειδιού που παρέχεται από την πλατφόρμα και το οποίο αντικαθιστά τον όρο 'private key'. Επειδή στο πειραματικό στάδιο της εργασίας χρησιμοποιούμε πέντε διαφορετικά SLA, υλοποιήθηκαν και πέντε διαφορετικά accounts στο λογαριασμό του Metamask που τους αντιστοιχούν. Επομένως, οι δύο τελευταίες εντολές επαναλαμβάνονται πέντε φορές μία για κάθε account (signer1, signer2, signer3, signer4, signer5).

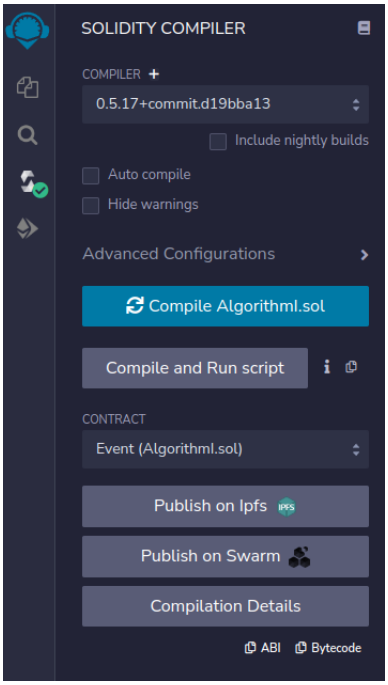
```
//store JSON data
let obj;
let ID='id';
let ProvName='pname';
let ProvID='pid';
let CustName='cname';
let CustID='cid';
let Availability='av';
let RespTime='time';
let HttpErrors='err';

//Stop multiple deployments of the same contract
const ids = [];
var is = 0;

//get data from json input
app.post('/jsons',(req,res)=>{
    is = 0;
    obj = req.body;
    ProvName = obj.Provider[0].name;
    ProvID = obj.Provider[0].id;
    CustName = obj.Customer[0].name;
    CustID = obj.Customer[0].id;
    Availability = obj.availability;
    RespTime = obj.responseTime;
    HttpErrors = obj.httpErrors;
    for (const x in ids){
        if (x == obj.SlaID){
            is = 1;
        }
    }
    if (is==0){
        ID = obj.SlaID;
        ids.push(ID);
        deployment(ID);
    }
});
```

Σχήμα 3.12: Εισαγωγή δεδομένων του SLA στο αρχείο Node.js μέσω συνάρτησης http post

Αναφέρθηκε στην ενότητα 3.3 ότι τα δεδομένα που συλλέγονται από το JSON αρχείο στο οποίο ορίζεται το SLA αποστέλλονται μέσω της μεθόδου http post στη 'http://localhost:8080/jsons' διεύθυνση. Στο σχήμα 3.12 φαίνεται ο τρόπος με τον οποίο λαμβάνονται τα παραπάνω δεδομένα από το Node.js αρχείο.

Στην αρχή του σχήματος 3.12 φαίνεται ότι έχουν δημιουργηθεί εννέα μεταβλητές, οι οποίες χρησιμοποιούνται για την αποθήκευση των τιμών που θα ληφθούν από το JSON αρχείο. Αναλυ-

τικά, στη μεταβλητή "obj" αποθηκεύεται το JSON object που λαμβάνεται μέσω της http post και οι υπόλοιπες μεταβλητές είναι οι πληροφορίες του SLA, οι οποίες έχουν αναλυθεί διεξοδικά στην ενότητα 3.2. Με σκοπό την πιστοποίηση ύπαρξης μοναδικού έξυπνου συμβολαίου για κάθε SLA ID έχουν ορισθεί δύο μεταβλητές. Η μία είναι η 'ids' η οποία αποτελεί μία λίστα στην οποία προστίθεται κάθε SLA ID που έχει αναγνωσθεί μέσω της app.post και η δεύτερη είναι η μεταβλητή 'is¨, η οποία χρησιμοποιείται ως μία boolean μεταβλητή και λαμβάνει τη τιμή ¨1' όταν το ID που διαβάζει η post έχει ήδη αναγνωσθεί και έχει προστεθεί στην 'ids' και τη τιμή ¨0' όταν είναι η πρώτη φορά που η εφαρμογή συναντάει το συγκεκριμένο SLA. Οι πληροφορίες του JSON αρχείου θα ληφθούν από την app.post(), η οποία θα ελέγξει με τον τρόπο που προαναφέραμε εάν είναι η πρώτη φορά που συναντάει το συγκεκριμένο SLA ID και σε περίπτωση θετικής απάντησης θα προχωρήσει σε deployment του συμβολαίου μέσω της συνάρτησης 'deployment(id)' η οποία παρουσιάζεται στο σχήμα 3.15.



Σχήμα 3.13: Μεταγλώττιση έξυπνου συμβολαίου μέσω του Remix IDE

Στο Σχήμα 3.13 παρουσιάζεται η πλατφόρμα του Remix IDE η οποία διευκολύνει τη διαδικασία της μεταγλώττισης του έξυπνου συμβολαίου. Για τη μεταγλώττιση χρησιμοποιείται ο compiler 0.8.7. Έχοντας εισάγει το έξυπνο συμβόλαιο, όπου στη συγκεκριμένη περίπτωση αντιστοιχεί στον Αλγόριθμο I, επιλέγεται το κουμπί "Compile AlgorithmI.sol", το οποίο ξεκινά τη διαδικασία. Μόλις γίνει η μεταγλώττιση και δεν εμφανιστούν σφάλματα, παρέχονται τα περιεχόμενα των abi και bytecode που προκύπτουν. Τα περιεχόμενα αυτά αποθηκεύονται σε μεταβλητές με αντίστοιχο όνομα (abi, bytecode) στο αρχείο Node.js.

Προτού γίνει το deployment και η ανάγνωση των περιεχομένων του JSON αρχείου, ορίζεται το έξυπνο συμβόλαιο που έχει κατασκευασθεί, όπως φαίνεται στο σχήμα 3.14.

```
// Initialize contract
var contr = new web3.eth.Contract(abi);
```

Σχήμα 3.14: Εισαγωγή έξυπνου συμβολαίου

Συμπερασματικά, κάθε φορά που εισέρχεται ένα SLA, μέσω της σελίδας που έχει δημιουργηθεί, στο σύστημα, αποστέλλονται οι πληροφορίες που κατέχει και μόλις αναγνωσθούν από το αρχείο Node.js ενεργοποιείται αυτόματα μία συνάρτηση που εκτελεί το deployment του έξυπνου συμβολαίου στο οποίο αντιστοιχεί. Για να επιτευχθεί το τελευταίο, γίνεται χρήση της συνάρτησης "Contract.deploy(options)" η οποία παρέχεται από τη βιβλιοθήκη web3. Η συνάρτηση παίρνει ως παραμέτρους το bytecode το οποίο το έχουμε λάβει από το Remix IDE και τις πληροφορίες που λαμβάνει ως είσοδο ο κατασκευαστής του έξυπνου συμβολαίου οι οποίες φαίνονται και στο σχήμα 3.5. Εν συνεχεία, καλείται η συνάρτηση "send" με την οποία πραγματοποιείται το deployment μέσω του λογαριασμού που αντιστοιχεί στο id του έξυπνου συμβολαίου. Η εύρεση του αντίστοιχου λογαριασμού γίνεται στην αρχή της συνάρτησης με if / else if statements αναζητώντας ένα ένα για το σωστό id.

```
let signer=signer1;

function deployment(id) {
    if (id==1){
        signer=signer1;
    }
    else if(id==2){
        signer=signer2;
    }
    else if(id==3){
        signer=signer3;
    }
    else if(id==4){
        signer=signer4;
    }
    else if(id==5){
        signer=signer5;
    }

    contr.deploy({
        data: bytecode,
        arguments: [ID,ProvName,ProvID,CustName,CustID,Availability,RespTime,HttpErrors]
    }).send({
        from: signer.address,
        gas: 2000000
    }).then((deployment)=> {
        console.log('Contract was deployed at the following address: ');
        console.log(deployment.options.address);
        Interval(deployment,id);
    }).catch((err) => {
        console.error(err);
    });
}
```

Σχήμα 3.15: Συνάρτηση deployment για το έξυπνο συμβόλαιο

Αφότου γίνει το deployment ορίζεται τι θα ακολουθήσει μέσω της συνάρτησης "then". Επιλέχθηκε, αρχικά, να δημοσιεύεται στην console η διεύθυνση του συμβολαίου που αναπτύχθηκε ώστε να μπορεί να γίνει έλεγχος για τυχόν σφάλματα. Επιπλέον, ύστερα από κάθε deployment καλείται η συνάρτηση Interval(deployment,id) η οποία δέχεται ως παραμέτρους το contract instance που έχει δημιουργηθεί και το id του SLA και παρουσιάζεται στο σχήμα 3.16.

```
let times = [];

async function Interval(deployed,id){
    times[id] = 0;
    if (id==1){
        setInterval(function(){EventEmit(deployed,id,signer1)}, 60*1000);
    }
    else if(id==2){
        setInterval(function(){EventEmit(deployed,id,signer2)}, 60*1000);
    }
    else if(id==3){
        setInterval(function(){EventEmit(deployed,id,signer3)}, 60*1000);
    }
    else if(id==4){
        setInterval(function(){EventEmit(deployed,id,signer4)}, 60*1000);
    }
    else if(id==5){
        setInterval(function(){EventEmit(deployed,id,signer5)}, 60*1000);
    }
    setInterval(function(){EventListen(deployed,id)}, 60*1000);

}
```

Σχήμα 3.16: Συνάρτηση Interval για προσδιορισμό χρονικού παραθύρου ανάμεσα στις ενημερώσεις του Prometheus και ανάμεσα στις κλήσεις της συνάρτησης ελέγχου του έξυπνου συμβολαίου.

Ελέγχεται, αρχικά, και στη συνάρτηση Interval, με τον ίδιο τρόπο, ποιό είναι το account του Metamask που αντιστοιχεί στο εξεταζόμενο SLA id. Μόλις εντοπισθεί καλεί τη συνάρτηση setInterval() η οποία έχει ορισθεί με τέτοιο τρόπο ώστε να καλεί τη συνάρτηση EventEmit() κάθε ένα λεπτό. Τέλος, χρησιμοποιείται πάλι η setInterval() για να καλέσει αυτή τη φορά τη συνάρτηση EventListen() με επαναλήψεις ίδιου χρονικού παραθύρου. Οι συναρτήσεις EventEmit() και EventListen() παρουσιάζονται με τη μορφή κώδικα στα σχήματα 3.18 και 3.20 αντίστοιχα.

Με σκοπό την κατασκευή ενός αυτοσχέδιου Oracle έχουν δημιουργηθεί τρεις πίνακες, οι οποίοι φαίνονται στο σχήμα 3.17. Ο πίνακας numAv αφορά τις μετρήσεις που δίνονται για την διαθεσιμότητα, ο numHTTP αφορά τις μετρήσεις για τα HTTP σφάλματα και ο numRT αφορά τις μετρήσεις για το χρόνο απόκρισης. Συνολικά έχουν προστεθεί πενήντα τιμές στον κάθε πίνακα. Ακόμη, έχει δημιουργηθεί ο πίνακας times ο οποίος αντιστοιχεί κάθε θέση του στον αντίστοιχο αριθμό SLA id. Δηλαδή, στη θέση 1 αντιστοιχίζεται το SLA id = 1, στη θέση 2 το SLA id = 2 κ.ο.κ.. Η κάθε θέση του πίνακα που αντιστοιχεί σε SLA id που έχει εισέλθει στο σύστημα αρχικοποιείται μέσα από τη συνάρτηση Interval σε μηδέν. Ο πίνακας αυτός χρησιμοποιείται για τον υπολογισμό των φορών που έχει κληθεί η συνάρτηση ελέγχου "check()" του έξυπνου συμβολαίου. Επιπλέον, σε κάθε κλήση της δίνονται ως είσοδοι οι μετρήσεις που έχουμε δημιουργήσει και βρίσκονται στους πίνακες numAv, numHTTP και numRT. Επομένως, ο times έχει και ρόλο δείκτη στην ανάλογη θέσση των τριών προαναφερθέντων πινάκων ώστε να λαμβάνονται με τη σωστή σειρά οι μετρήσεις.

```
const numAv = [100,94,99,99,100,99,98,98,100,93,
               92,99,100,96,90,97,99,100,100,100,
               100,100,100,100,100,100,100,100,100,100,
               90,96,99,98,100,100,99,98,99,93,
               95,96,97,98,96,97,98,95,100,93];

const numHTTP = [5,20,15,20,30,40,40,30,45,20,
                 20,25,30,30,31,28,29,35,35,34,
                 35,35,36,37,30,28,35,34,30,32,
                 1,2,3,4,5,6,7,8,1,2,
                 40,39,45,35,33,38,36,30,40,35];

const numRT = [800,1200,1100,700,600,500,1400,1500,1300,1100,
               1200,1150,500,400,300,100,300,600,200,100,
               1200,1300,1250,1210,1270,1290,1100,1300,1250,1240,
               1400,1300,1200,1000,1300,1500,1600,1200,1700,1300,
               800,900,1000,800,1200,1300,700,1000,900,600];
```

Σχήμα 3.17: Τιμές μετρήσεων διαθεσιμότητας, HTTP σφαλμάτων και χρόνου απόκρισης που λειτουργούν ως αυτοσχέδιο Blockchain Oracle

Αναφέρθηκε προηγουμένως, ότι η συνάρτηση Interval καλεί κάθε ένα λεπτό την EventEmit. Η τελευταία δέχεται ως παραμέτρους το contract instance, το οποίο έχει ληφθεί ως αποτέλεσμα του deployment του έξυπνου συμβολαίου, το id του SLA που εξετάζεται και μία μεταβλητή sign που καθορίζει το account που θα πραγματοποιήσει τις πληρωμές. Η εύρεση του signer γίνεται στη συνάρτηση Interval με σκοπό την αποφυγή καθυστερήσεων και πολλαπλών επαναλήψεων της ίδιας διαδικασίας. Ο ρόλος της συνάρτησης EventEmit() είναι η κλήση της συνάρτησης ελέγχου παραβιάσεων "check()" που βρίσκεται στο smart contract. Ταυτόχρονα, της αποστέλλει και ως εισόδους τις τιμές των τριών μετρικών που έχουμε προαναφέρει. Για την επίτευξη των παραπάνω κάνει χρήση της συνάρτησης "myContract.methods.myMethod()" που παρέχεται από τη βιβλιοθήκη web3. Συγκεκριμένα στο πεδίο myContract αντιστοιχίζεται το contract instance που έχει λάβει ως είσοδο και στο πεδίο myMethod η συνάρτηση "check()" μαζί με τις απαραίτητες παραμέτρους. Η εντολή αυτή πραγματοποιείται μέσω της συνάρτησης send η οποία λαμβάνει ως παραμέτρους τον λογαριασμό που αντιστοιχεί στο συγκεκριμένο SLA id και το όριο του gas για τη συναλλαγή αυτή.

Μετά την ολοκλήρωση της παραπάνω διαδικασίας, αυξάνεται κατά ένα το πεδίο στον πίνακα times που αντιστοιχεί στο id του SLA που βρίσκεται υπό επιτήρηση. Μόλις ο αριθμός αυτός φτάσει τα πενήντα, λόγω του ότι η μία φορά αντιστοιχίζεται στον αριθμό μηδέν, σημαίνει ότι δεν υπάρχουν επιπλέον είσοδοι για να γίνει ο έλεγχος και άρα δεν επαναλαμβάνονται κλήσεις της συνάρτησης "check()". Αυτό γίνεται λόγω των πειραματικών στόχων της διπλωματικής εργασίας και σε πραγματικές συνθήκες ο αριθμός αυτός θα ήταν πολύ μεγαλύτερος.

```
function EventEmit(deployed,id,sign){

    console.log('Times of ',id,' are :',times[id]);

    let t = times[id];

    if (t == 50){
        console.log('Completed: ',id);
    }
    else{
        deployed.methods.check(numAv[t],numHTTP[t],numRT[t]).send({
            from: sign.address,
            gas: 2000000
        }).then(()=>{
            console.log('Checking: ', id);
            console.log('Signer: ', sign.address);
            times[id] = times[id] + 1;
        }).catch((err) => {
            console.error(err);
        });
    }
}
```

Σχήμα 3.18: Συνάρτηση EventEmit με την οποία γίνεται η κλήση της συνάρτησης ελέγχου παραβιάσεων στο έξυπνο συμβόλαιο.

```
const register = new client.Registry();

register.setDefaultLabels({
    app:'SLA_APP'
});

client.collectDefaultMetrics({register});

const gauge = new client.Gauge({
    name: 'Violations',
    help: 'Violations_help',
    labelNames: ['SLAid','Violated'] ,
});

register.registerMetric(gauge);

const rand = (low, high) => Math.random() * (high - low) + low;

setInterval(() => {
    gauge.set(rand(0, 15));
}, 1000);

app.get('/metrics', async(req, res) => {
    res.setHeader('Content-Type', register.contentType);
    res.send(await register.metrics());
});
```

Σχήμα 3.19: Σύνδεση με Prometheus και ορισμός μετρικής Violations.

Βασικό κομμάτι της διπλωματικής εργασίας κύριως όσον αφορά τις συγκριτικές μελέτες που θα παρουσιαστούν σε επόμενο κεφάλαιο αποτελεί η πλατφόρμα του Prometheus. Επιλέχθηκε να δημιουργηθεί μία νέα μετρική η οποία θα παρακολουθείται μέσω αυτής της πλατφόρμας και θα αφορά τις παραβιάσεις που έχουν πραγματοποιηθεί. Η σύνδεση με τον Prometheus και ο ορισμός της μετρικής φαίνονται στο σχήμα 3.19.

Αρχικά, γίνεται εισαγωγή του client του Prometheus μέσω της εντολής: "const client = require('prom-client');" και ορίζεται η μεταβλητή register μέσω της οποίας επιλέγονται οι μετρικές που καταχωρούνται. Επιλέχθηκε η χρήση μετρικής Gauge, η οποία αποτελεί έναν μετρητή που έχει τη δυνατότητα να αυξηθεί, αλλά και να μειωθεί. Η μετρική αυτή συνοδεύεται και από δύο labels με ονομασίες Slaid και Violated. Τα labels αυτά, αφορούν το id του SLA στο οποίο αντιστοιχούν οι μετρήσεις και στη μετρική που έχει παραβιαστεί, αντίστοιχα.

```
function EventListen(deployed,id){
    deployed.getPastEvents('ViolationA',{
        fromBlock: 0,
        toBlock: 'latest'
    }).then(function(events){
        gauge.labels(id,'AVAILABILITY').set(events.length);
        console.log('PrometheusA: ',id);
    }).catch((err) => {
        console.error(err);
    });

    deployed.getPastEvents('ViolationB',{
        fromBlock: 0,
        toBlock: 'latest'
    }).then(function(events){
        gauge.labels(id,'HTTP').set(events.length);
        console.log('PrometheusB: ',id);
    }).catch((err) => {
        console.error(err);
    });

    deployed.getPastEvents('ViolationC',{
        fromBlock: 0,
        toBlock: 'latest'
    }).then(function(events){
        gauge.labels(id,'RESPONSE TIME').set(events.length);
        console.log('PrometheusC: ',id);
    }).catch((err) => {
        console.error(err);
    });

}
```

Σχήμα 3.20: Συνάρτηση EventListen μέσω της οποίας ενημερώνεται η μετρική του Prometheus

Η ενημέρωση της μετρικής Violations του Prometheus γίνεται μέσω της συνάρτησης EventListen η οποία καλείται κάθε ένα λεπτό μέσω της Interval. Όπως φαίνεται στο σχήμα 3.20, για κάθε ξεχωριστό event (ViolationA, ViolationB, ViolationC), υπάρχει όμοια αλλά ξεχωριστή διαχείριση. Σε αυτό το σημείο, δίνεται βάση στην εξής λογική: Όσες φορές έχει γίνει κάποιο emit event από το έξυπνο συμβόλαιο, τόσες παραβιάσεις θα έχουν συμβεί. Επομένως, μετρώντας τον αριθμό των emit events υπολογίζουμε και τη ποσότητα των παραβιάσεων. Για την καταμέτρηση αυτή, γίνεται χρήση της συνάρτησης "myContract.getPastEvents(event[, options][, callback])" μέσω της οποίας υπολογίζονται όλες οι κλήσεις του κάθε event που εξετάζεται.

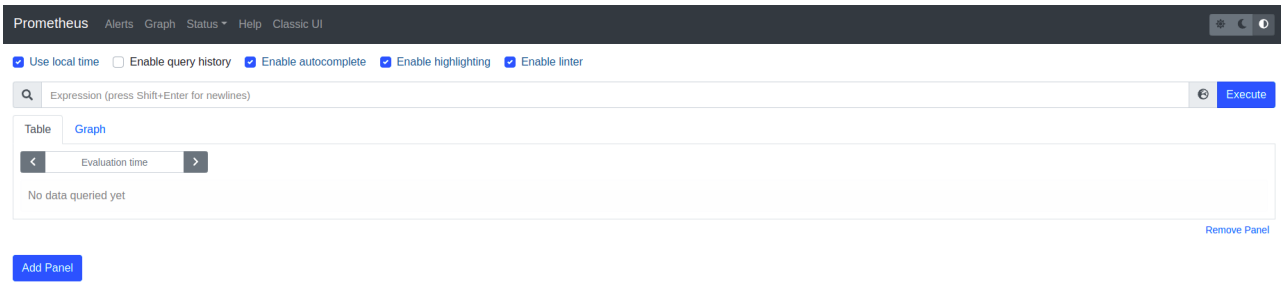
Κεφάλαιο 4

Επίδειξη Λειτουργικότητας Εφαρμογής

Η εφαρμογή αναπτύσσεται και λειτουργεί στην τοπική διεύθυνση του υπολογιστή (localhost). Για την εκτέλεσή της χρειάζεται να βρίσκονται σε λειτουργία το αρχείο Node.js και ο server του Prometheus. Τα παραπάνω χρησιμοποιούν τις θύρες 8080 και 9090, αντίστοιχα. Στη θύρα 8080 εμφανίζεται επομένως το frontend (Σχήμα 3.4), όπου ο χρήστης εισάγει το JSON αρχείο που περιέχει τις πληροφορίες του SLA στο πεδίο "Choose file".

Μόλις γίνει η εισαγωγή του αρχείου, πραγματοποιείται deployment του έξυπνου συμβολαίου με τις πληροφορίες του SLA και ανά ένα λεπτό καλείται η συνάρτηση ελέγχου παραβιάσεων που βρίσκεται στο smart contract, αλλά και ενημερώνεται η μετρική Violations που έχει δημιουργηθεί για την επιτήρηση παραβιάσεων μέσω του Prometheus.

Στο σχήμα 4.1 φαίνεται η αρχική σελίδα του Prometheus. Για την εμφάνιση των αποτελεσμάτων, πληκτρολογείται στο πεδίο Expression η μετρική που βρίσκεται υπό παρακολούθηση. Στη συγκεκριμένη περίπτωση, η μετρική αυτή ονομάζεται Violations.

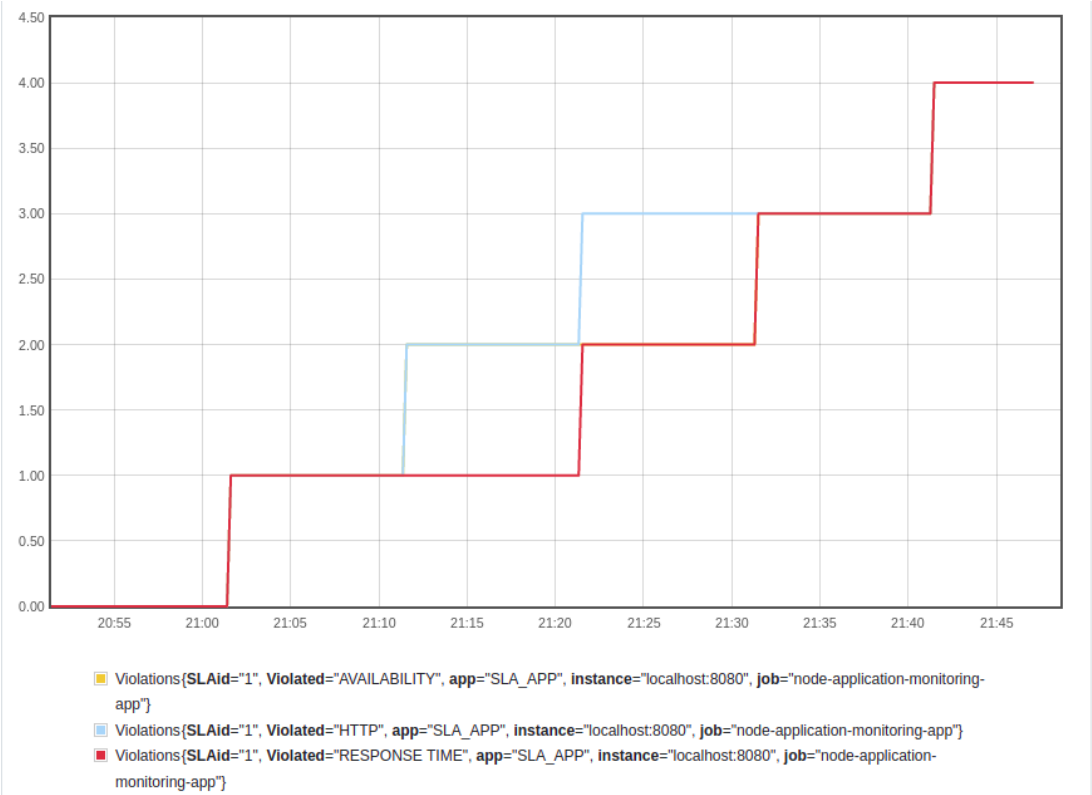


Σχήμα 4.1: Αρχική σελίδα Prometheus

Για την επίδειξη της σωστής λειτουργίας της εφαρμογής, έγινε εκτέλεση του συστήματος με το SLA το οποίο αντιστοιχεί σε id = 1 στο Rinkeby Ethereum Testnet. Παρακάτω παρουσιάζονται αναλυτικά ένα ένα τα αποτελέσματα για τον κάθε αλγόριθμο που χρησιμοποιήθηκε, οι οποίοι αναφέρονται διεξοδικά στις υποενότητες 3.4.1, 3.4.2.

4.1 Αλγόριθμος I

Στο σχήμα 4.2 φαίνεται η συνολική εικόνα των παραβιάσεων που λαμβάνεται από την πλατφόρμα του Prometheus. Στα σχήματα 4.3, 4.4 και 4.5 παρουσιάζεται ξεχωριστά το διάγραμμα που προκύπτει για καθεμία από τις μετρικές ¨διαθεσιμότητα¨, ¨σφάλματα HTTP' και ¨χρόνος απόκρισης¨, αντίστοιχα. Στο συγκεκριμένο σημείο, αξίζει να γίνει υπενθύμιση πάνω στο γεγονός ότι ο Αλγόριθμος I υπολογίζει το μέσο όρο των μετρήσεων που λαμβάνει ανά δέκα τιμές και βάσει του αποτελέσματος αποφασίζει αν έχει γίνει παραβίαση ή όχι.



Σχήμα 4.2: Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus.

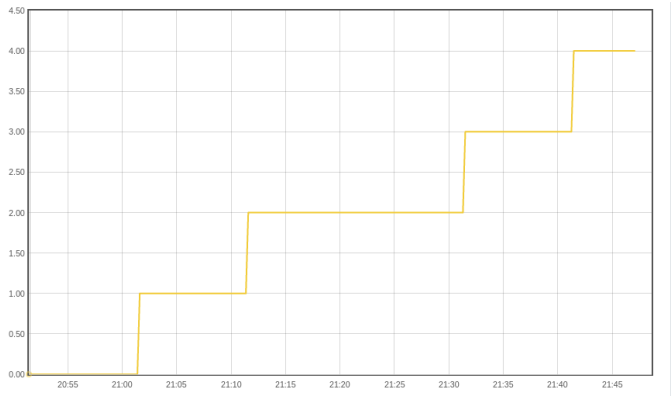
Για τον έλεγχο της λειτουργίας του αλγορίθμου, έχει γίνει υπολογισμός των μέσων όρων της κάθε μετρικής που προκύπτουν ανά δέκα μετρήσεις. Οι τιμές αυτές φαίνονται στον πίνακα 4.1. Στον Πίνακα 3.1 βρίσκονται τα όρια των μετρικών για το κάθε SLA που έχει δημιουργηθεί, όπου στην προκειμένη περίπτωση, δηλαδή για το SLA με id = 1 είναι τα εξής:

- Διαθεσιμότητα: 99 %
- Χρόνος Απόκρισης: 900 ms
- Σφάλματα HTTP: 10 %

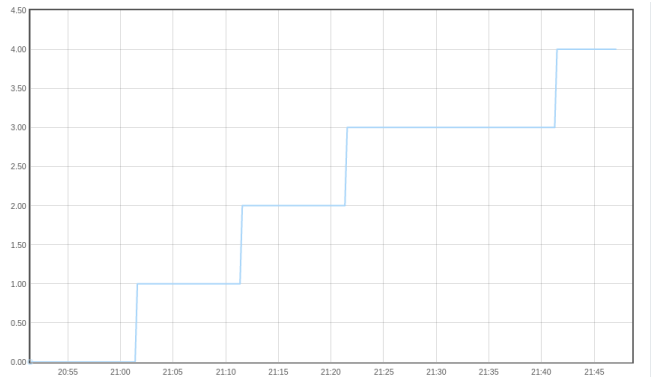
Χρόνος	Διαθεσιμότητα	Σφάλματα HTTP	Χρόνος Απόκρισης
10 λεπτά	98	26	1020
20 λεπτά	97	29	485
30 λεπτά	100	33	1241
40 λεπτά	97	3	1350
50 λεπτά	96	37	920

Πίνακας 4.1: Μέσος όρος για την κάθε μετρική που προκύπτει ανά δέκα μετρήσεις

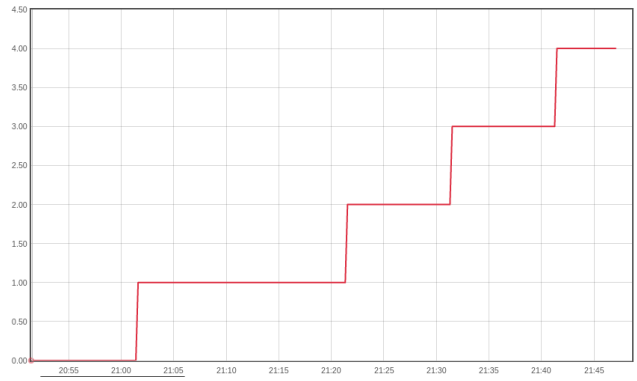
Επομένως, παρατηρώντας τους δύο αυτούς πίνακες μπορούν να διαπιστωθούν οι χρόνοι στους οποίους αναμένονται νέες παραβιάσεις, αλλά και η ποσότητα αυτών. Συγκεκριμένα, οι παραβιάσεις που αναμένονται για τη μετρική της διαθεσιμότητας είναι τέσσερις, όπου η πρώτη γίνεται μετά το πρώτο δεκάλεπτο, όταν δηλαδή έχουν ήδη προωθηθεί δέκα μετρήσεις στη συνάρτηση ελέγχου του έξυπνου συμβολαίου, με την καθεμία να λαμβάνει χώρα ένα λεπτό μετα την προηγούμενη. Όμοια, η δεύτερη παραβίαση αναμένεται στα 20 λεπτά, η τρίτη στα 40 και η τέταρτη στα πενήντα. Σε αυτό το σημείο δε παρέχονται άλλες μετρήσεις οπότε σταματάνε και οι ενημερώσεις. Τα παραπάνω καλύπτονται πλήρως από το σχήμα 4.3. Με όμοιο τρόπο, τα χρονικά σημεία που αναμένουμε παραβιάσεις για τα HTTP σφάλματα είναι στα 10, στα 20, στα 30 και στα 50 λεπτά, τα οποία φαίνονται και στο σχήμα 4.4. Τέλος, για το χρόνο απόκρισης, αναμένονται παραβιάσεις στα 10, 30, 40 και 50 λεπτά (Σχήμα 4.5).



Σχήμα 4.3: Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus.



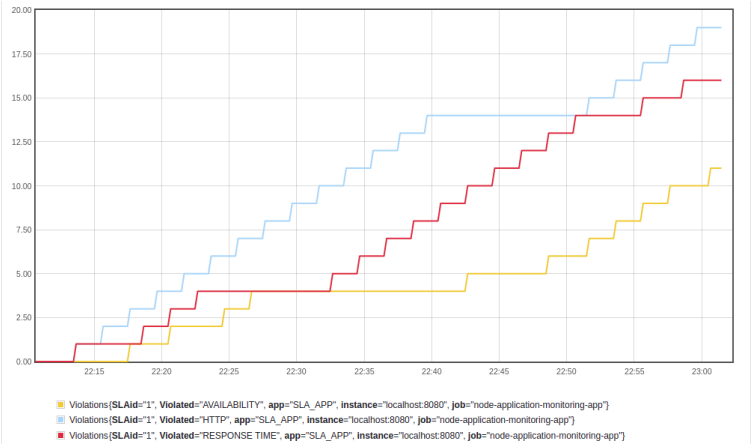
Σχήμα 4.4: Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus.



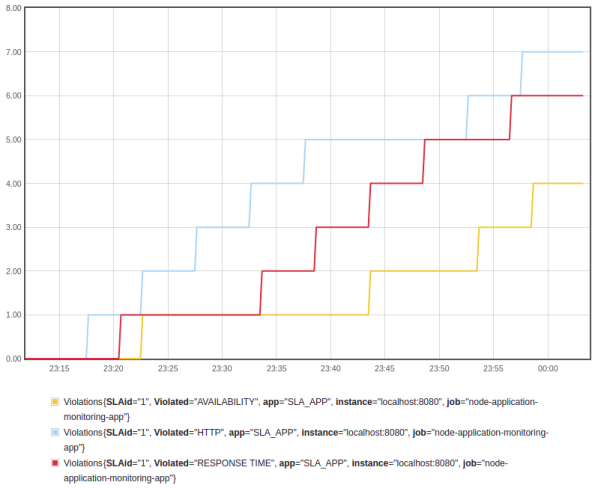
Σχήμα 4.5: Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου I στο SLA με id = 1 στον Prometheus.

4.2 Αλγόριθμοι II, III και IV

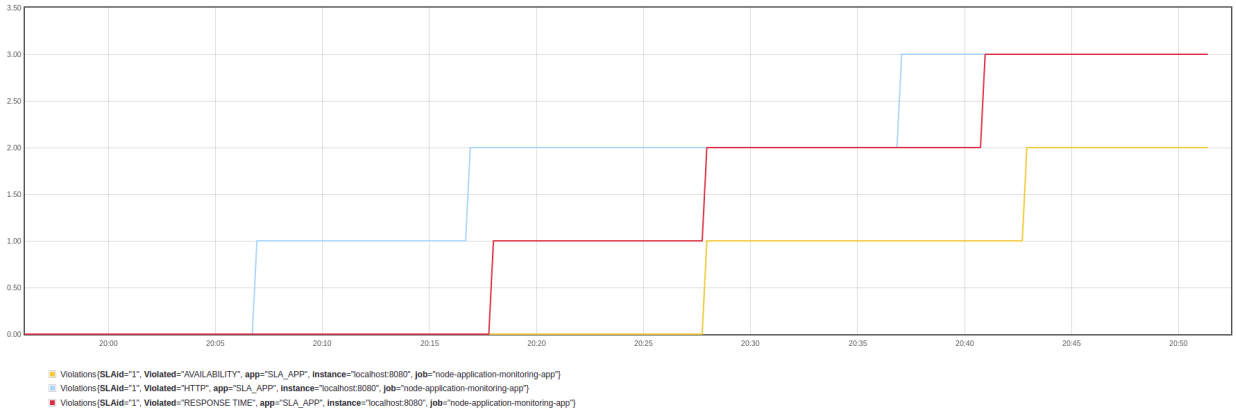
Στα σχήματα 4.6, 4.7 και 4.8 φαίνεται η συνολική εικόνα των παραβιάσεων που λαμβάνεται από την πλατφόρμα του Prometheus για τους Αλγορίθμους II, III και IV, αντίστοιχα. Πιο διεξοδικά, παρουσιάζονται οι παραβιάσεις της μετρικής ¨διαθεσιμότητα' στα σχήματα 4.9, 4.10 και 4.11, της μετρικής ¨σφάλματα HTTP' στα 4.12, 4.13 και 4.14 και της μετρικής ¨χρόνος απόκρισης' στα σχήματα 4.15, 4.16 και 4.17 , για καθέναν από τους αλγορίθμους I, II και IV αντίστοιχα.



Σχήμα 4.6: Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus.



Σχήμα 4.7: Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus.



Σχήμα 4.8: Επιτήρηση παραβιάσεων με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus.

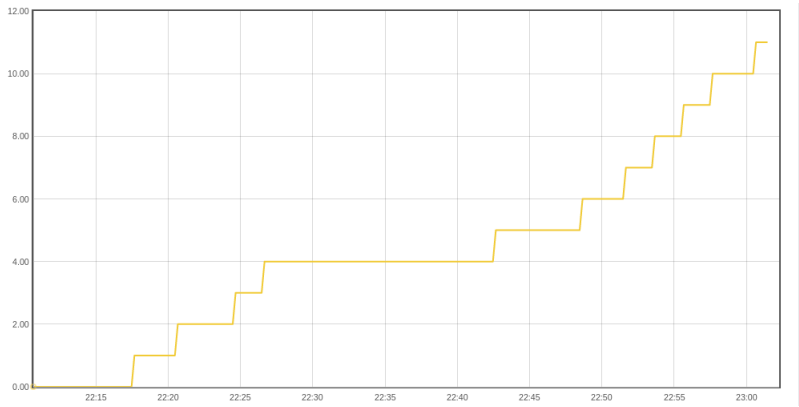
Ακόμα έχουν δημιουργηθεί οι Πίνακες 4.2, 4.3 και 4.4 στους οποίους φαίνεται αν γίνεται παραβίαση των τιμών που ορίζει το SLA (id=1) για κάθε μέτρηση που του δίνεται από το αυτοσχέδιο Oracle που έχει δημιουργηθεί για την εργασία. Οι μετρήσεις φαίνονται στο Σχήμα 3.17 και οι συμβολισμοί y, n δηλώνουν ότι υπάρχει ή δεν υπάρχει παραβίαση, αντίστοιχα. Για παράδειγμα, παρατηρώντας τον πίνακα 4.2 βλέπουμε ότι για την πρώτη μέτρηση (πρώτο λεπτό) η οποία είναι 100%, εφόσον χρειάζεται κάτι μεγαλύτερο του 99% για να μην υπάρξει παραβίαση, έχει ορισθεί με n. Αντίθετα, η δεύτερη μέτρηση (δεύτερο λεπτό) που είναι 94% έχει ορισθεί με y κ.ο.κ.. Έτσι, μετρώντας ανά δύο y, δηλαδή ανά δύο παραβιάσεις μπορούμε να διαπιστώσουμε τα λεπτά στα οποία αναμένουμε να υπάρξει πραγματική παραβίαση για τον Αλγόριθμο II, δηλαδή παραβίαση σύμφωνα με τον αλγόριθμο που έχει ορισθεί στο SLA. Ακολουθούμε την ίδια διαδικασία και για τους υπόλοιπους αλγορίθμους, με τη διαφορά ότι μετράμε ανά 5 και ανά 10 y αντίστοιχα και προκύπτουν τα αποτελέσματα που περιγράφονται παρακάτω και επαληθεύονται από τα Σχήματα 4.9, 4.12, 4.15, 4.10, 4.13, 4.16, 4.11, 4.14 και 4.17.

Χρόνος(λεπτά)	1	2	3	4	5	6	7	8	9	10
+0	n	y	n	n	n	n	y	y	n	y
+10	y	n	n	y	y	y	n	n	n	n
+20	n	n	n	n	n	n	n	n	n	n
+30	y	y	n	y	n	n	n	y	n	y
+40	y	y	y	y	y	y	y	y	n	y

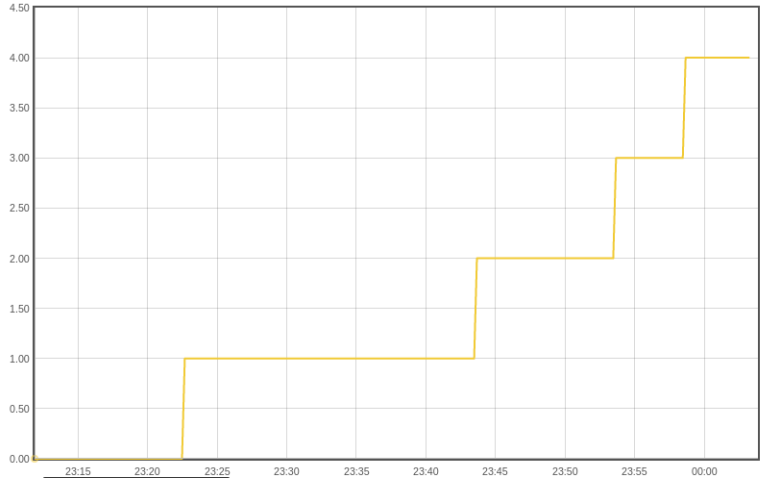
Πίνακας 4.2: Παραβιάσεις ανά μέτρηση για τη ¨Διαθεσιμότητα' για το SLA με id = 1, (n:δεν υπάρχει παραβίαση,y:υπάρχει παραβίαση)

Για τη μετρική ¨**Διαθεσιμότητα**', αναμένονται τα εξής αποτελέσματα, σύμφωνα με τον Πίνακα 4.2:

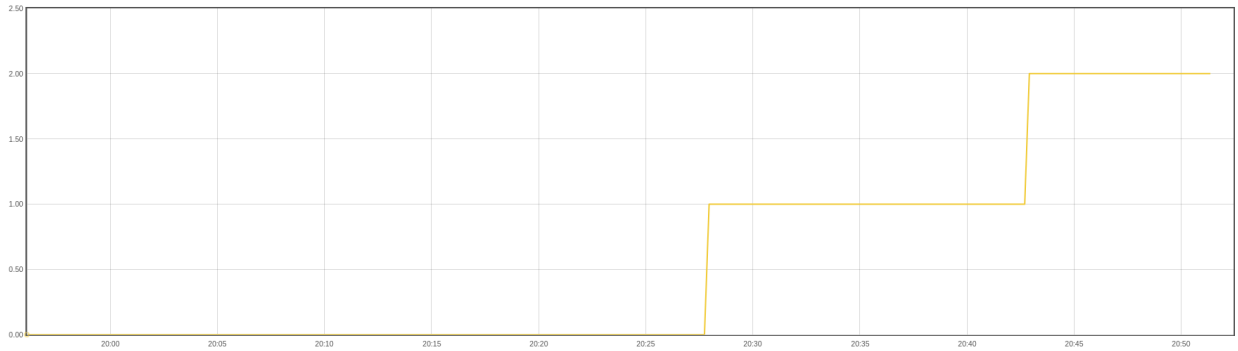
- **Αλγόριθμος II**: Παραβίαση στα 7, 10, 14, 16, 32, 38, 41, 43, 45, 47 και 50 λεπτά. Συνολικά 11 παραβιάσεις.
- **Αλγόριθμος III**: Παραβίαση στα 11, 32, 42 και 47 λεπτά. Συνολικά 4 παραβιάσεις.
- **Αλγόριθμος IV:** Παραβίαση στα 32 και 47 λεπτά. Συνολικά 2 παραβιάσεις.



Σχήμα 4.9: Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus.



Σχήμα 4.10: Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus.



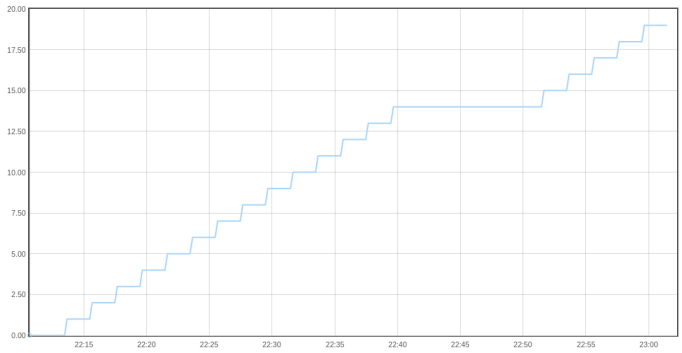
Σχήμα 4.11: Επιτήρηση παραβιάσεων της μετρικής ¨Διαθεσιμότητα¨, με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus.

Για τη μετρική **¨Σφάλματα HTTP'**, αναμένονται τα εξής αποτελέσματα, σύμφωνα με τον Πίνακα 4.3:

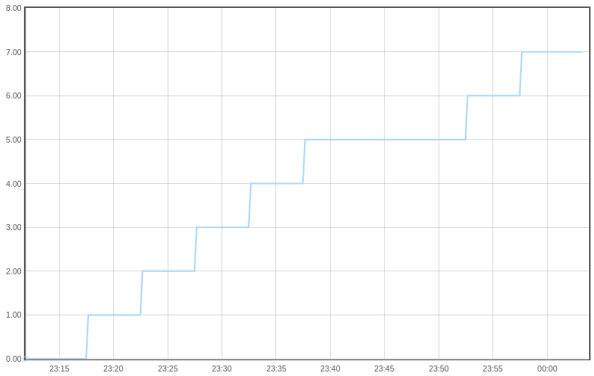
- **Αλγόριθμος II**: Παραβίαση στα 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 41, 43, 45, 47 και 49 λεπτά. Συνολικά 19 παραβιάσεις.
- **Αλγόριθμος III**: Παραβίαση στα 6, 11, 16, 21, 26, 41 και 46 λεπτά. Συνολικά 7 παραβιάσεις.
- **Αλγόριθμος IV:** Παραβίαση στα 11, 21 και 41 λεπτά. Συνολικά 3 παραβιάσεις.

Χρόνος(λεπτά)	1	2	3	4	5	6	7	8	9	10
+0	n	y	y	y	y	y	y	y	y	y
+10	y	y	y	y	y	y	y	y	y	y
+20	y	y	y	y	y	y	y	y	y	y
+30	n	n	n	n	n	n	n	n	n	n
+40	y	y	y	y	y	y	y	y	y	y

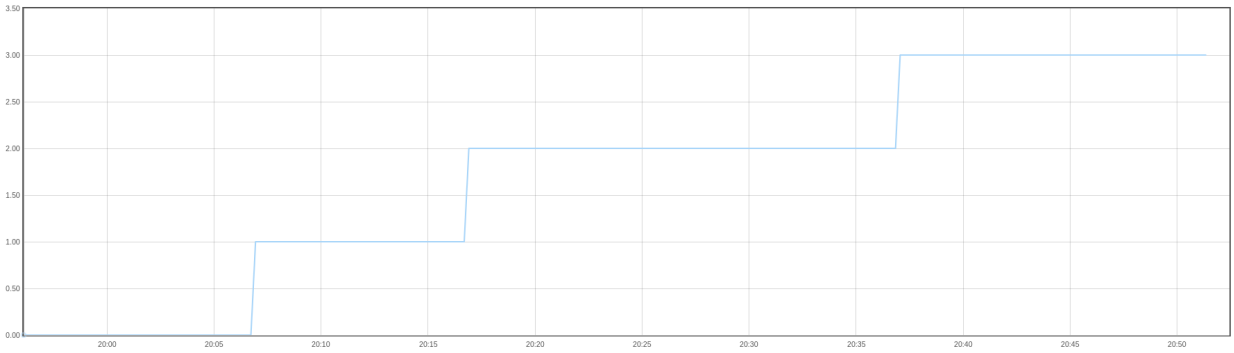
Πίνακας 4.3: Παραβιάσεις ανά μέτρηση για τα ¨Σφάλματα HTTP' για το SLA με id = 1, (n:δεν υπάρχει παραβίαση,y:υπάρχει παραβίαση)



Σχήμα 4.12: Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου II στο SLA με id = 1 στον Prometheus.



Σχήμα 4.13: Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus.



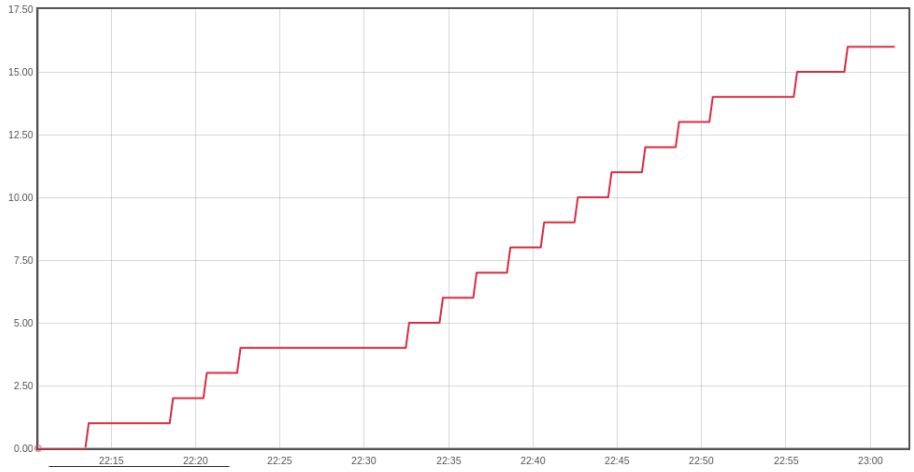
Σχήμα 4.14: Επιτήρηση παραβιάσεων της μετρικής ¨Σφάλματα HTTP¨, με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus.

Για τη μετρική **"Χρόνος Απόκρισης'**, αναμένονται τα εξής αποτελέσματα, σύμφωνα με τον Πίνακα 4.4:

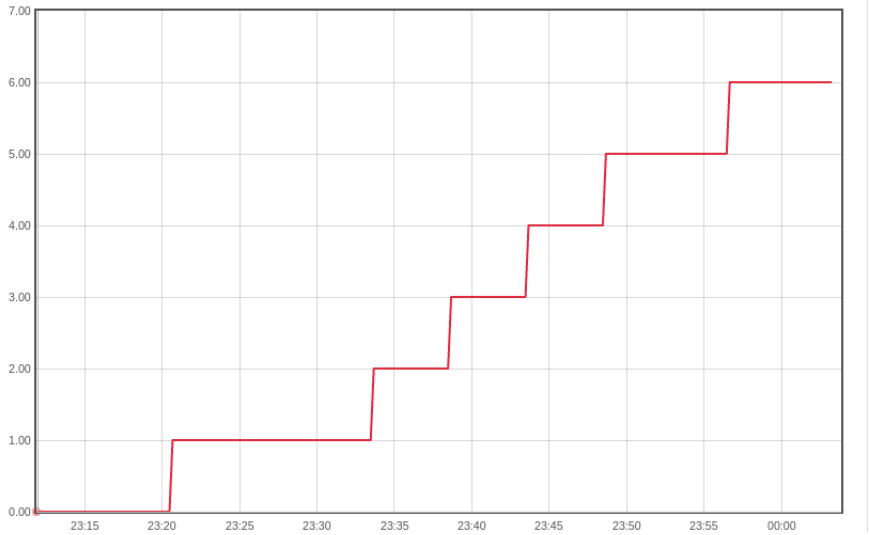
- **Αλγόριθμος ΙΙ**: Παραβίαση στα 3, 8, 10, 12, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 45 και 48 λεπτά. Συνολικά 16 παραβιάσεις.
- **Αλγόριθμος ΙΙΙ**: Παραβίαση στα 9, 22, 27, 32, 37 και 45 λεπτά. Συνολικά 6 παραβιάσεις.
- **Αλγόριθμος IV:** Παραβίαση στα 22, 32 και 45 λεπτά. Συνολικά 3 παραβιάσεις.

Χρόνος(λεπτά)	1	2	3	4	5	6	7	8	9	10
+0	n	y	y	n	n	n	y	y	y	y
+10	y	y	n	n	n	n	n	n	n	n
+20	y	y	y	y	y	y	y	y	y	y
+30	y	y	y	y	y	y	y	y	y	y
+40	n	n	y	n	y	y	n	y	n	n

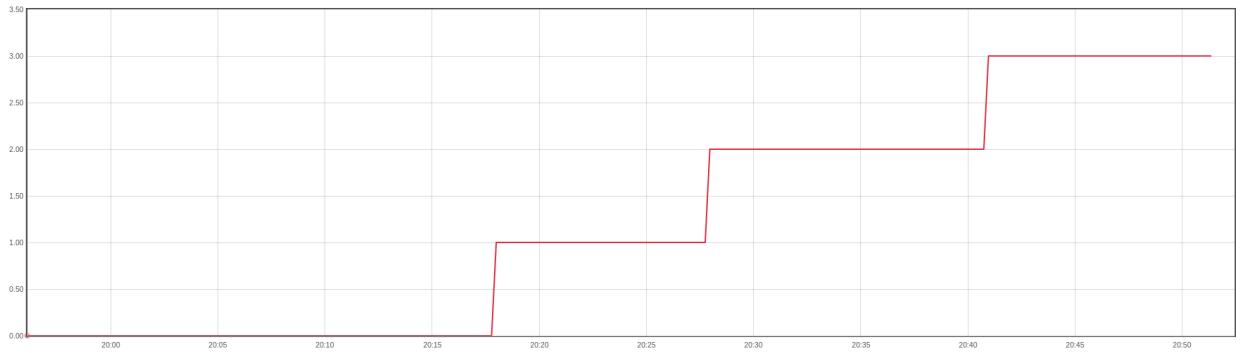
Πίνακας 4.4: Παραβιάσεις ανά μέτρηση για το "Χρόνο Απόκρισης' για το SLA με id = 1, (n:δεν υπάρχει παραβίαση,y:υπάρχει παραβίαση)



Σχήμα 4.15: Επιτήρηση παραβιάσεων της μετρικής "Χρόνος Απόκρισης", με τη χρήση του Αλγορίθμου ΙΙ στο SLA με id = 1 στον Prometheus.



Σχήμα 4.16: Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου III στο SLA με id = 1 στον Prometheus.



Σχήμα 4.17: Επιτήρηση παραβιάσεων της μετρικής ¨Χρόνος Απόκρισης¨, με τη χρήση του Αλγορίθμου IV στο SLA με id = 1 στον Prometheus.

Κεφάλαιο 5

Συγκριτικές Μελέτες

Στο κεφάλαιο αυτό πραγματοποιείται συγκριτική μελέτη ως προς την απόδοση των πειραματικών Ethereum δικτύων, αλλά και την απόδοση των δημιουργηθέντων αλγορίθμων που παρουσιάζονται στις ενότητες 3.4.1 και 3.4.2. Για την πραγματοποίηση της σύγκρισης, δημιουργήθηκαν πέντε SLA, τα περιεχόμενα των οποίων φαίνονται στον Πίνακα 3.1. Τα αντίστοιχα αρχεία προστέθηκαν στο σύστημα σειριακά με χρονική απόσταση δευτερολέπτων. Ύστερα υποβλήθηκαν, σε ίδιο χρόνο, σε 50 ελέγχους, όπου ταυτόχρονα ενημερώνονταν και οι μετρήσεις της κάθε εξεταζόμενης μετρικής από το αυτοσχέδιο Oracle που έχει κατασκευασθεί. Μετά την ολοκλήρωση της κάθε εκτέλεσης υπολογίσθηκαν τα κόστη των συναλλαγών για κάθε λογαριασμό, με τη βοήθεια του Etherscan. Επειδή η διπλωματική εργασία χρησιμοποιεί πειραματικά δίκτυα, τα κόστη που υπολογίσθηκαν και στα οποία βασίζεται η συγκριτική μελέτη, είναι τα Txn taxes, τα οποία αποτελούν αντιπροσωπευτικό κομμάτι των "Value" τιμών που θα εξετάζονταν σε κανονικό δίκτυο.

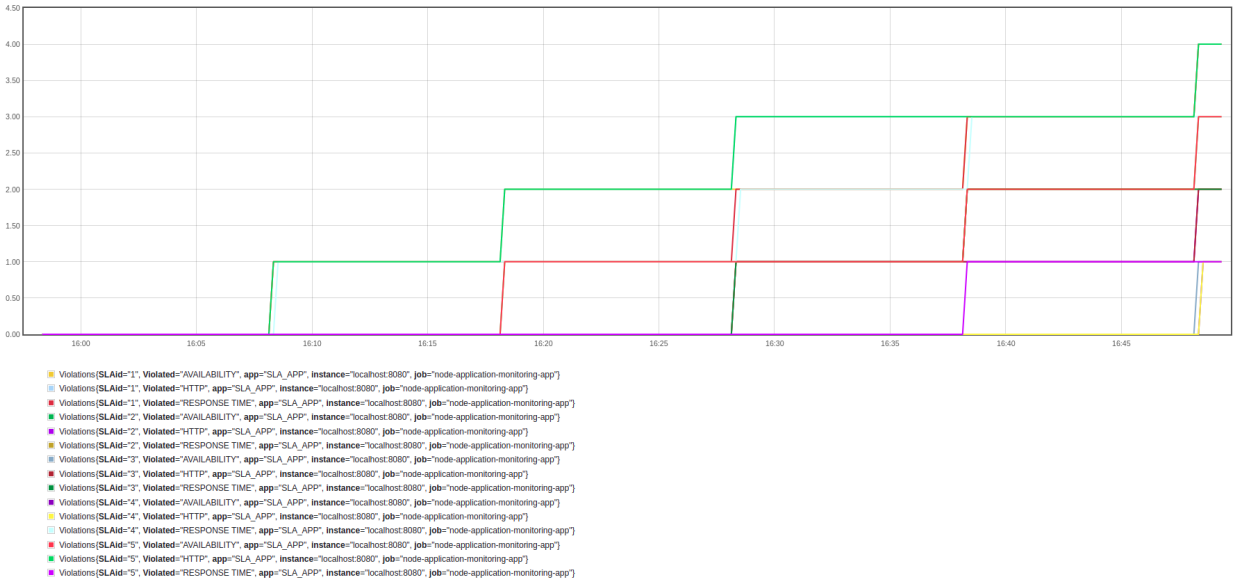
Έχει ορισθεί, οι έλεγχοι παραβιάσεων και οι ενημερώσεις της μετρικής που χρησιμοποιείται στον Prometheus, να επαναλαμβάνονται κάθε ένα λεπτό μέχρι το τέλος του πειράματος. Λόγω, λοιπόν, συγκεκριμένης λογικής του συστήματος, όλα τα αποτελέσματα που προκύπτουν στην πλατφόρμα του Prometheus γίνονται στο ίδιο χρονικό διάστημα, επομένως η συγκριτική μελέτη επικεντρώνεται στο συνολικό κόστος των συναλλαγών που πραγματοποιούνται.

5.1 Συγκριτική μελέτη μεταξύ πειραματικών Ethereum δικτύων

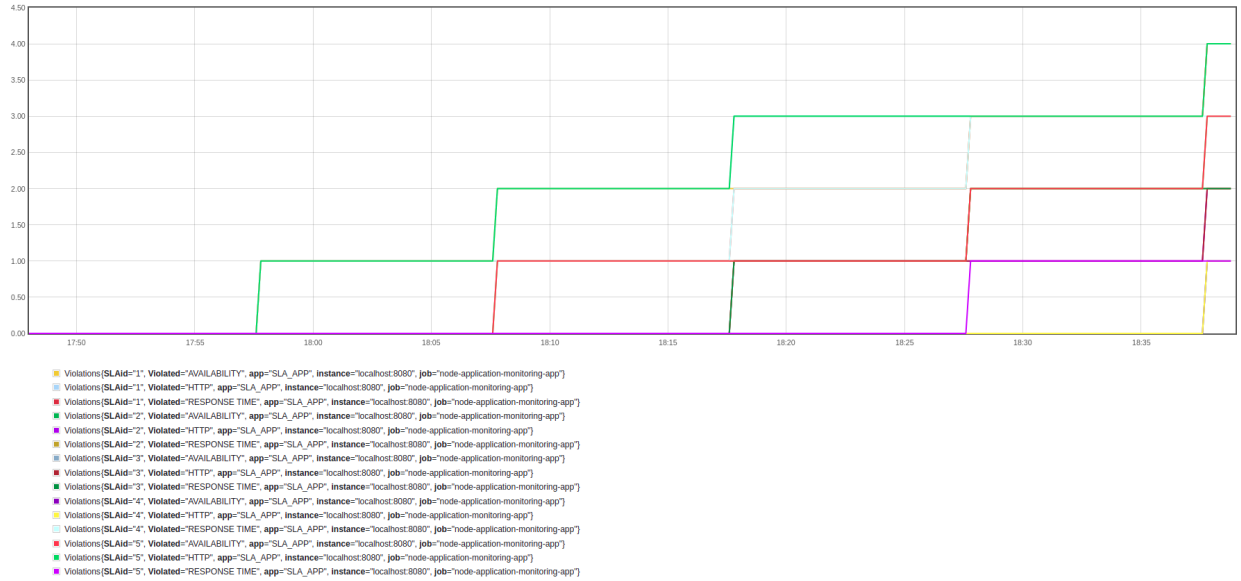
Στην ενότητα αυτή γίνεται σύγκριση του συνολικού κόστους των συναλλαγών που πραγματοποιούνται σε καθένα από τα Ethereum testnets: Goerli, Kovan και Rinkeby. Έγινε ξεχωριστή εκτέλεση του πειράματος για το κάθε πειραματικό δίκτυο και χρησιμοποιήθηκε ο Αλγόριθμος Ι. Τα αποτελέσματα που παρήγαγε η πλατφόρμα του Prometheus φαίνονται στα Σχήματα 5.1, 5.2 και 5.3. Στα σχήματα αυτά, παρουσιάζεται ο τρόπος επιτήρησης των παραβιάσεων, που έχει ορισθεί στη διπλωματική εργασία, για την κάθε εκτέλεση που πραγματοποιήθηκε. Το γεγονός ότι ταυτίζονται τόσο τα επιμέρους χρονικά διαστήματα όσο και ο αριθμός των παραβιάσεων που πραγματοποιήθη-

καν, αποδεικνύει τη σωστή λειτουργία του συστήματος και την δυνατότητα αποφυγής πιθανών καθυστερήσεων των υπό μελέτη δικτύων, που θα αλλοίωναν την εγγύτητα της επιτήρησης.

Σχήμα 5.1: Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος I, Rinkeby Ethereum testnet



Σχήμα 5.2: Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος I, Goerli Ethereum testnet



Σχήμα 5.3: Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος I, Kovan Ethereum testnet

Με σκοπό την καλύτερη κατανόηση των αποτελεσμάτων, δημιουργήθηκε ο Πίνακας 5.1, στον οποίο παρουσιάζονται αναλυτικά οι αναμενόμενες συνολικές παραβιάσεις, αλλά και οι παραβιάσεις ξεχωριστά της κάθε μετρικής, για το κάθε SLA που χρησιμοποιήθηκε. Ακόμα, με τη βοήθεια του Etherscan υπολογίσθηκαν τα συνολικά κόστη που αντιστοιχούν σε κάθε λογαριασμό. Τα κόστη, αυτά, συγκεντρώθηκαν στον Πίνακα 5.2, όπου διακρίνονται ανάλογα με το λογαριασμό και το αντίστοιχο πειραματικό Ethereum δίκτυο.

SLA id	Διαθεσιμότητα	Σφάλματα HTTP	Χρόνος Απόκρισης	Σύνολο
1	4	4	4	12
2	3	2	2	7
3	1	2	2	5
4	1	1	3	5
5	3	4	1	8

Πίνακας 5.1: Αριθμός παραβιάσεων της κάθε μετρικής, αλλά και συνολικά, για το κάθε SLA που χρησιμοποιήθηκε σε εκτέλεση του Αλγορίθμου I.

	Goerli	Kovan	Rinkeby
Account 1	0.009831910030653	0.009831910027529	0.00983191198833
Account 2	0.00980599253058	0.009805992527457	0.009805994483815
Account 3	0.009793587530543	0.009793587527422	0.00979358948265
Account 4	0.009799647530669	0.009799647527439	0.009799649484726
Account 5	0.009816140030717	0.009816140027485	0.009816141986332

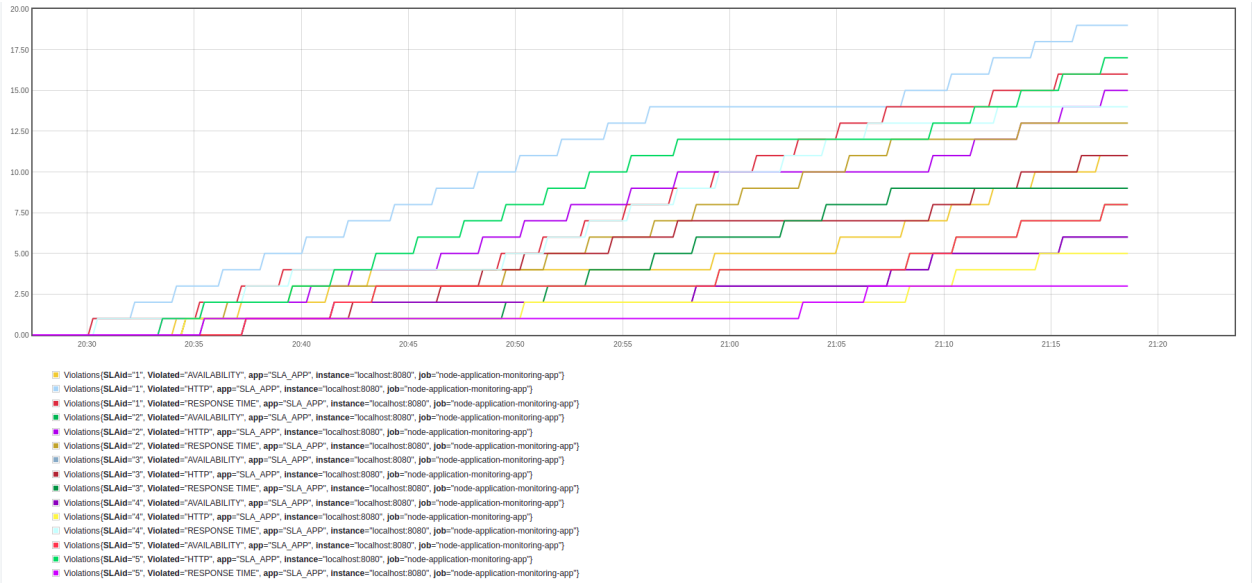
Πίνακας 5.2: Txn fees που χρεώθηκε ο καθένας από τους 5 λογαριασμούς που δημιουργήθηκαν στο Metamask για καθένα από τα πειραματικά Ethereum δίκτυα Goerli, Kovan και Rinkeby, ύστερα από την προσθήκη 5 SLAs στο σύστημα με 50 ελέγχους παραβιάσεων, με τη χρήση του Αλγορίθμου I.

Παρατηρώντας τον πίνακα 5.2 είναι φανερό ότι το Rinkeby δίκτυο είναι το πιο κοστοβόρο με ακόλουθο το Goerli. Επομένως, επιτυγχάνεται καλύτερη απόδοση μέσω του Kovan testnet. Στον πίνακα 5.1, αλλά και στα Σχήματα 5.1, 5.2, 5.3, φαίνονται οι συνολικές παραβιάσεις για το κάθε SLA. Έχει αναφερθεί σε προηγούμενη ενότητα, ότι το SLA με id = i αντιστοιχεί στο account i. Επομένως, συνδυάζοντας τα δεδομένα των πινάκων 5.2 και 5.1 προκύπτει το συμπέρασμα, όπως και αναμένονταν, ότι περισσότερες παραβιάσεις οδηγούν σε μεγαλύτερα κόστη.

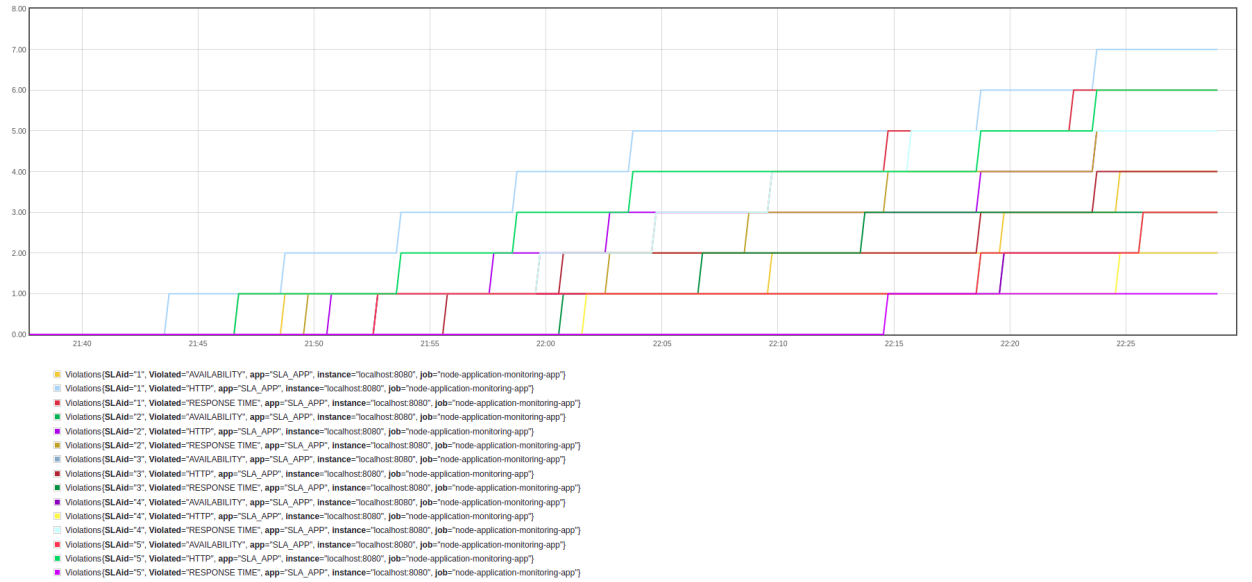
5.2 Συγκριτική μελέτη μεταξύ αλγορίθμων ελέγχου παραβιάσεων SLA μετρικών

Στην ενότητα αυτή συγκρίνονται οι αλγόριθμοι που έχουν αναλυθεί στις ενότητες 3.4.1 και 3.4.2 ως προς την απόδοσή τους. Συγκεκριμένα, έκτελέστηκαν 4 πειράματα με τις ίδιες παραμέτρους, οι οποίες αναφέρονται στην αρχή του κεφαλαίου, στο πειραματικό δίκτυο Rinkeby. Σε κάθε πείραμα χρησιμοποιήθηκε διαφορετικός αλγόριθμος ελέγχου παραβιάσεων στο έξυπνο συμβόλαιο και τα αποτελέσματα που προέκυψαν στη πλατφόρμα του Prometheus φαίνονται στα Σχήματα 5.1, 5.4, 5.5 και 5.6. Στα σχήματα, αυτά, παρουσιάζεται ο τρόπος με τον οποίο χρησιμοποιείται ο Prometheus για την επιτήρηση των παραβιάσεων.

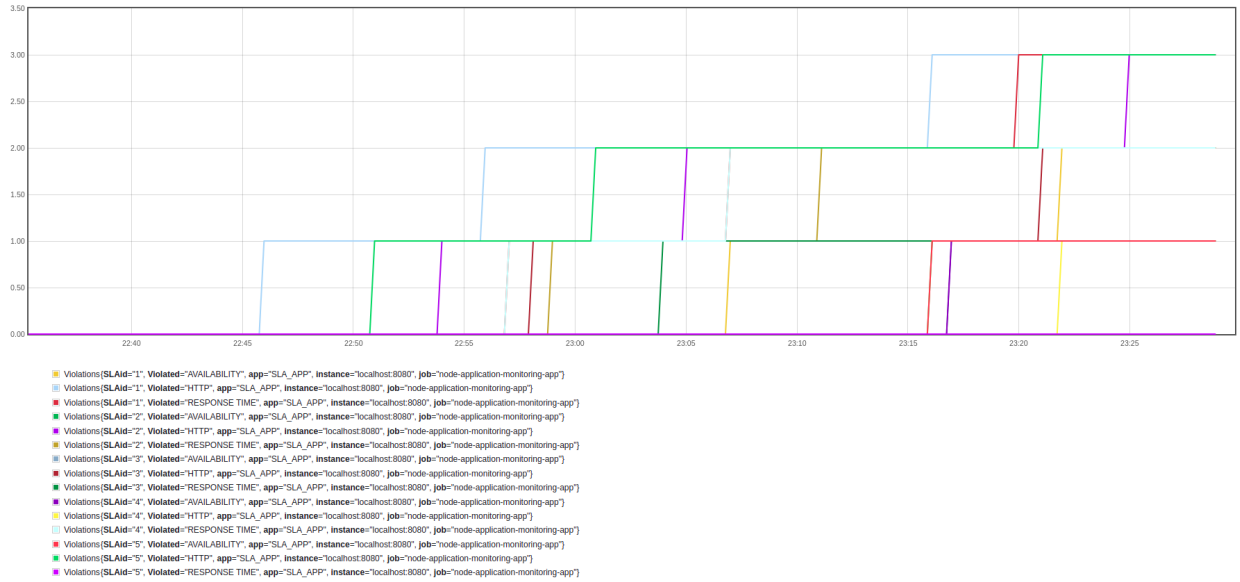
Το πρώτο πράγμα που διακρίνει κανείς από τα αποτελέσματα του Prometheus, είναι το πλήθος των παραβιάσεων στους Αλγορίθμους II και III. Αναμένεται, επομένως, λόγω των αυξημένων συναλλαγών, όπως διαπιστώθηκε στην προηγούμενη υποενότητα, να έχουν τη χειρότερη απόδοση όσον αφορά το συνολικό κόστος. Με τη βοήθεια του Etherscan, υπολογίσθηκαν τα κόστη που αντιστοιχούν σε κάθε λογαριασμό με αντιστοιχία στο κάθε πείραμα με διαφορετικό αλγόριθμο και συγκεντρώθηκαν στον πίνακα 5.3.



Σχήμα 5.4: Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος II, Rinkeby Ethereum testnet



Σχήμα 5.5: Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος III, Rinkeby Ethereum testnet



Σχήμα 5.6: Επιτήρηση παραβιάσεων πέντε SLAs στον Prometheus, Αλγόριθμος IV, Rinkeby Ethereum testnet

	Αλγόριθμος I	Αλγόριθμος II	Αλγόριθμος III	Αλγόριθμος IV
Account 1	0.00983191198833	0.010221605057865	0.00915841505127	0.008796575039946
Account 2	0.009805994483815	0.009571217554574	0.008606332548382	0.008441920038365
Account 3	0.00979358948265	0.009064855051541	0.008382950047095	0.008183905037268
Account 4	0.009799649484726	0.009008435051232	0.008362180047123	0.008158120037243
Account 5	0.009816141986332	0.009020805051447	0.008436515047631	0.008196790037356

Πίνακας 5.3: Txn fees του Rinkeby πειραματικού Ethereum δικτύου που χρεώθηκε ο καθένας από τους 5 λογαριασμούς που δημιουργήθηκαν στο Metamask για καθέναν από τους 4 αλγορίθμους που χρησιμοποιήθηκαν, ύστερα από την προσθήκη 5 SLAs στο σύστημα με 50 ελέγχους παραβιάσεων.

Παρατηρώντας τον Πίνακα 5.3 προκύπτει το συμπέρασμα ότι ο πιο αποδοτικός είναι ο αλγόριθμος IV με επόμενο στη σειρά τον III. Τη χειρότερη απόδοση την έχει ο Αλγόριθμος I και αυτό οφείλεται πιθανόν στην πολυπλοκότητα των υπολογισμών, καθώς πέρα των προσθέσεων τις οποίες αξιοποιούν και οι υπόλοιποι, χρησιμοποιεί και διαιρέσεις για τον υπολογισμό του μέσου όρου. Οι αλγόριθμοι II, III και IV, οι οποίοι διαφέρουν ως προς την ποσότητα των event παραβιάσεων που αποστέλλουν, συμβάλλουν στην επαλήθευση του συμπεράσματος που έγινε όσον αφορά τη ποσότητα των παραβιάσεων, ότι δηλαδή όσες λιγότερες παραβιάσεις λαμβάνουν χώρα, βάσει του αλγορίθμου που χρησιμοποιείται, τόσο καλύτερη είναι και η απόδοση όσον αφορά το συνολικό κόστος.

Κεφάλαιο 6

Μελλοντικές επεκτάσεις και Συμπεράσματα

Η παρούσα διπλωματική εργασία πραγματεύεται την επιτήρηση Συμφωνιών Επιπέδου Υπηρεσίας μέσω έμμεσων απεικονίσεων σε Δίκτυα Αλυσίδας-Κορμού. Το θέμα αυτό δύναται να έχει πληθώρα επεκτάσεων όσον αφορά τα επιμέρους στοιχεία του. Η εργασία αυτή, συγκεντρώνει όλους τους πόρους της στην καθαρή επιτήρηση των παραβιάσεων που γίνονται, και μάλιστα, λόγω της απεικόνισης των SLA στο Blockchain δίκτυο και με τη βοήθεια της πλατφόρμας του Prometheus έχει επιτευχθεί η ζωντανή και αποκεντρωμένη παρακολούθηση αυτών.

Μία πιθανή επέκταση που αξίζει να αναφερθεί, είναι η διαδικασία αποζημίωσης σε περίπτωση παραβίασης των όρων του SLA. Η διαδικασία αυτή είναι αρκετά χρονοβόρα κατά κύριο λόγο και απαιτεί εμπιστοσύνη μεταξύ των μελών της συμφωνίας. Επομένως, υπάρχει η επιθυμία αυτοματοποίησής της, αλλά και εύρεσης τρόπου εκμηδενισμού της ανάγκης για εμπιστοσύνη. Τα παραπάνω θέματα υπήρχαν και στη διαδικασία της επιτήρησης των SLA. Με την ίδια λογική, λοιπόν, ενδείκνυται να κατασκευαστεί και μία μέθοδος αυτόματης αποζημίωσης σύμφωνα με τους προσυμφωνηθέντες όρους του SLA.

Επιπρόσθετα, αναφέρθηκε ότι για την εξαγωγή των μετρήσεων χρησιμοποιήθηκαν τρεις πίνακες, ένας για κάθε μετρική που βρισκόταν υπό επιτήρηση, οι οποίοι δρούσαν σαν ένα αυτοσχέδιο Oracle. Σε πραγματικές συνθήκες, τα δεδομένα αυτά είναι αναγκαίο να ληφθούν από κάποιο Decentralized Oracle το οποίο θα παρέχει τις πραγματικές μετρήσεις με τις οποίες θα γίνεται ο έλεγχος των παραβιάσεων. Ένα Decentralized Oracle είναι μια ομάδα ανεξάρτητων Blockchain Oracles που παρέχουν δεδομένα σε ένα blockchain. Κάθε Decentralized Oracle ανακτά δεδομένα από μια πηγή εκτός αλυσίδας και τα φέρνει στην αλυσίδα. Τα δεδομένα στη συνέχεια συγκεντρώνονται, ώστε το σύστημα να μπορεί να φτάσει σε μια ντετερμινιστική τιμή αλήθειας για αυτό το σημείο δεδομένων. Με αυτόν τον τρόπο παρέχεται λύση στο πρόβλημα των Oracle, δηλαδή στην αδυναμία επικοινωνίας των Blockchain με πηγές πληροφοριών.

Μία επιπλέον επέκταση στην εργασία αυτή, αποτελεί η διαδικασία συνεννόησης και διαπραγματεύσεων μεταξύ των μελών της συμφωνίας, με σκοπό την κατασκευή του SLA. Η μορφή των SLA που χρησιμοποιούνται στην εργασία αυτή είναι προσυμφωνημένη και γίνεται μόνο η απει-

κόνισή της στο blockchain. Προτού κατασκευαστούν αυτά τα συμβόλαια, πραγματοποιείται μία σειρά διαπραγματεύσεων με σκοπό το βέλτιστο και πιο συμφέροντα ορισμό των όρων της συμφωνίας. Οι διαπραγματεύσεις αυτές έχουν το περιθώριο να γίνονται μέσα από ειδικά κατασκευασμένη αποκεντρωμένη εφαρμογή με σκοπό την εξασφάλιση ορθότητας και του αναλλοίωτου για την τελική μορφή των όρων του συμβολαίου, καθώς δε θα εισάγεται στο σύστημα από κάποιο φυσικό πρόσωπο.

Στους όρους του SLA βρίσκεται και ο αλγόριθμος με τον οποίο συμπεραίνεται η ύπαρξη παραβιάσεων για την κάθε μετρική, αλλά και άλλου είδους μετρικές οι οποίες δεν έχουν συμπεριληφθεί στην παρούσα εργασία. Υπάρχει δυνατότητα ορισμού αλγορίθμων που από τη μία δεν αυξάνουν σε μεγάλο βαθμό το συνολικό κόστος των συναλλαγών στο δίκτυο και από την άλλη φροντίζουν το αποτέλεσμα να είναι το ίδιο δίκαιο και για τα δύο μέρη της συμφωνίας. Μία επιπρόσθετη, λοιπόν, πρόταση επέκτασης της εργασίας είναι η αναζήτηση ενός βέλτιστου αλγορίθμου ελέγχου παραβιάσεων όσον αφορά τα συμφέροντα του παρόχου της υπηρεσίας και του πελάτη.

Τέλος, για την πραγματοποίηση της συγκριτικής μελέτης, αλλά και της επίδειξης λειτουργικότητας της εφαρμογής, έχει επιλεγεί να γίνει χρήση πειραματικών Ethereum δικτύων (Rinkeby, Goerli, Kovan). Επομένως, μία ακόμα αναγκαία επέκταση του συστήματος με σκοπό την εκτέλεσή του σε πραγματικό περιβάλλον, είναι η αντικατάσταση αυτών των δικτύων από το κύριο Ethereum δίκτυο ή και από άλλα blockchain δίκτυα. Για παράδειγμα, ένα ιδιωτικό blockchain δίκτυο έχει τη προδιάθεση για εξασφάλιση μεγαλύτερου βαθμού ιδιωτικότητας και ασφάλειας των δεδομένων, αλλά και μείωσης πιθανών καθυστερήσεων λόγω μειωμένης κινητικότητας στο δίκτυο.

6.1 Συμπεράσματα

Μετά το πέρας της παρούσας διπλωματικής εργασίας, αλλά και των επιμέρους συγκριτικών μελετών που πραγματοποιήθηκαν, προκύπτουν τα παρακάτω συμπεράσματα, τα οποία αφορούν τη λειτουργικότητά της.

Όπως προκύπτει και από το Κεφάλαιο 4, το συνολικό σύστημα που κατασκευάστηκε λειτουργεί βάσει των προδιαγραφών που ορίσθηκαν στην εργασία. Επιτυγχάνεται ορθή απεικόνιση των SLA σε Δίκτυα Αλυσίδας-Κορμού χωρίς παράλειψη πληροφοριών. Η επιτήρηση γίνεται σε ζωντανό χρόνο με ακριβή υπολογισμό των παραβιάσεων, χωρίς σφάλματα ή καθυστερήσεις. Επομένως, είναι εύλογα αναμενόμενη η ορθή λειτουργία του συνολικού συστήματος σε μη ερευνητικό περιβάλλον, έχοντας πραγματοποιήσει τις απαραίτητες μετατροπές.

Βασικοί στόχοι της εργασίας αυτής αποτελούν η εξασφάλιση της εγγύτητας των υπηρεσιών και η επίτευξη συναλλακτικών διαδικασιών και λύσεων χωρίς την απαραίτητη παρουσία τρίτων στην επιτήρηση των SLA. Μέσω της απεικόνισης των SLA σε Smart Contracts σε δίκτυο Blockchain κατορθώνεται αποκέντρωση της διαδικασίας της παρακολούθησης των παραβιάσεων της συμφωνίας. Η αποκέντρωση αυτή εκμηδενίζει την ανάγκη ύπαρξης εμπιστοσύνης μεταξύ του παρόχου υπηρεσιών και του πελάτη, αλλά και λόγω του Blockchain δικτύου επιτυγχάνεται ασφάλεια τόσο στις συναλλαγές όσο και στη διαφύλαξη προσωπικών δεδομένων. Συμπερασματικά, η αποκεντρωμένη επιτήρηση SLAs στην οποία στοχεύει η διπλωματική αυτή εργασία, οδηγεί σε αξιόπιστα και

ακριβή αποτελέσματα που δεν απαιτούν επαλήθευση από τρίτο.

Τέλος, όσον αφορά τις συγκριτικές μελέτες που παρουσιάστηκαν στο Κεφάλαιο 5, προέκυψαν τα εξής συμπεράσματα. Αρχικά, διαπιστώθηκε ότι κύριος παράγοντας αύξησης του συνολικού κόστους αποτελεί η πολυπλοκότητα των πράξεων που απαιτούνται. Για παράδειγμα, λιγότερο αποδοτικός αλγόριθμος προέκυψε να είναι ο Ι, ο οποίος ήταν ο μόνος που χρησιμοποιούσε διαιρέσεις. Το γεγονός ότι ο Αλγόριθμος ΙΙ πραγματοποιούσε αρκετά περισσότερες συναλλαγές δεν ήταν αρκετό για να τον οδηγήσει σε χειρότερη απόδοση από τον Ι. Ακόμα, όσον αφορά τα πειραματικά δίκτυα, παρατηρήθηκε ότι επιτυγχάνεται καλύτερη απόδοση με τη χρήση του δικτύου Kovan, χωρίς να κατορθώνει τεράστιες αποκλίσεις από τα υπόλοιπα.

Βιβλιογραφία

[1] *The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask*. https://metamask.io/.

[2] *Ethereum API | IPFS API & Gateway | ETH Nodes as a Service*. https://infura.io/.

[3] *Express 4.x - API Reference*. https://expressjs.com/en/api.html#express.

[4] *HTML: HyperText markup language | MDN*. https://developer.mozilla.org/en-US/docs/Web/HTML.

[5] *Introduction to Node.js*. https://nodejs.dev/learn/introduction-to-nodejs.

[6] *JavaScript | MDN*. https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[7] *JSON*. https://www.json.org/json-en.html.

[8] *Solidity — Solidity 0.8.13 documentation*. https://docs.soliditylang.org/en/v0.8.13/.

[9] *Welcome to Remix's documentation! — Remix - Ethereum IDE 1 documentation*. https://remix-ide.readthedocs.io/en/latest/.

[10] *What Is Web 2.0*. https://oreilly.com{file}.

[11] Alharby, Maher, Amjad Aldweesh, και Aad van Moorsel: *Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research (2018)*. Στο *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*, σελίδες 1–6, Fuzhou, China, Νοέμβριος 2018. IEEE, ISBN 9781728112770. https://ieeexplore.ieee.org/document/8756390/.

[12] Alzubaidi, A., K. Mitra, P. Patel, και E. Solaiman: *A Blockchain-based Approach for Assessing Compliance with SLA-guaranteed IoT Services*. Τεχνική Αναφορά arXiv:2006.15314, arXiv, Ιούνιος 2020. http://arxiv.org/abs/2006.15314, arXiv:2006.15314 [cs] type: article.

[13] Beniiche, Abdeljalil: *A Study of Blockchain Oracles*. Ιούλιος 2020.

[14] Berners-Lee, Tim: *Information Management: A Proposal.* undefined, 1990. https://www.semanticscholar.org/paper/Information-Management%3A-A-Proposal-Berners-Lee/b446cd6a9218bed4beca1027174315e8e8258442.

[15] Brito Gonçalves, João Paulo de, Roberta Lima Gomes, Rodolfo da Silva Villaca, Esteban Municio, και Johann Marquez-Barja: *A Quality of Service Compliance System Empowered by Smart Contracts and Oracles.* Στο *2020 IEEE International Conference on Blockchain (Blockchain)*, σελίδες 532–538, Αύγουστος 2020.

[16] etherscan.io: *Ethereum (ETH) Blockchain Explorer.* http://etherscan.io/.

[17] John Scheid, Eder, Bruno Rodrigues, Lisandro Granville, και Burkhard Stiller: *Enabling Dynamic SLA Compensation Using Blockchain-based Smart Contracts*. Απρίλιος 2019.

[18] Kaur, Rasmeet και Aleem Ali: *Implementation of Blockchain in IoT.* σελίδες 149–161. Μάρτιος 2022.

[19] Khan, Imran: *Internet, world wide web, and the librarian*. Ιανουάριος 2015.

[20] Kochovski, Petar, Vlado Stankovski, Sandi Gec, Francescomaria Faticanti, Marco Savi, Domenico Siracusa, και Seung Woo Kum: *Smart Contracts for Service-Level Agreements in Edge-to-Cloud Computing.* Journal of Grid Computing, Δεκέμβριος 2020.

[21] Maarouf, Adil, Abderrahim Marzouk, και Abdelkrim Haqiq: *Practical modeling of the SLA life cycle in Cloud Computing.* Στο *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, σελίδες 52–58, Marrakech, Morocco, Δεκέμβριος 2015. IEEE, ISBN 9781467387095. http://ieeexplore.ieee.org/document/7489170/.

[22] Mehboob, Kashif, Junaid Arshad, Waheed Iqbal, Sidrah Abdullah, και Hassan Zaib: *Blockchain-enabled real-time SLA monitoring for cloud-hosted services.* Cluster Computing, 25:1–23, Φεβρουάριος 2022.

[23] Nakamoto, Satoshi: *Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.*

[24] Neidhardt, Nils, Carsten Köhler, και Markus Nüttgens: *Cloud Service Billing and Service Level Agreement Monitoring based on Blockchain.* EMISA, 2018.

[25] Nekrashevych, Olena και Yuriy Kovrigo: *Overview of key performance indicators.* Modeling, Control and Information Technologies, σελίδες 108–109, Νοέμβριος 2019.

[26] Paschke, Adrian και Elisabeth Schnappinger-Gerull: *A Categorization Scheme for SLA Metrics.* Ιανουάριος 2006.

[27] Prometheus: *Prometheus - Monitoring system & time series database.* https://prometheus.io/.

[28] Reyna, Ana, Cristian Martín, Jaime Chen, Enrique Soler, και Manuel Díaz: *On blockchain and its integration with IoT. Challenges and opportunities.* Future Generation Computer Systems, 88:173–190, Νοέμβριος 2018, ISSN 0167-739X. https://www.sciencedirect.com/science/article/pii/S0167739X17329205.

[29] Ruan, Pingcheng, Gang Chen, Tien Tuan Anh Dinh, Qian Lin, Dumitrel Loghin, B. Ooi, και Meihui Zhang: *Blockchains and Distributed Databases: a Twin Study.* ArXiv, 2019.

[30] Sabah, Sana, Nada Mahdi, και Israa Majeed: *The road to the blockchain technology: Concept and types.* Periodicals of Engineering and Natural Sciences (PEN), 7:1821–1832, Δεκέμβριος 2019.

[31] Scheid, E. και B. Stiller: *Automatic SLA Compensation based on Smart Contracts April 2018*, 2018. https://www.semanticscholar.org/paper/Automatic-SLA-Compensation-based-on-Smart-Contracts-Scheid-Stiller/5d4ea67e92fd4064d74831bb9ac71e3ea6970437.

[32] SURVEY, PROGRESSION και Padmashri Desai: *PROGRESSION OF WEB 3.0(SEMANTIC WEB) FROM WEB 1.0: A SURVEY*. Απρίλιος 2009.

[33] Uriarte, Rafael Brundo, Rocco De Nicola, και Kyriakos Kritikos: *Towards Distributed SLA Management with Smart Contracts and Blockchain*. Δεκέμβριος 2018.

[34] Verma, D.C.: *Service level agreements on IP networks.* Proceedings of the IEEE, 92(9):1382–1388, Δεκέμβριος 2004, ISSN 1558-2256.

[35] Wonjiga, A. T., S. Peisert, L. Rilling, και C. Morin: *Blockchain as a trusted component in cloud SLA verification*. Δεκέμβριος 2019, ISBN 9781450370448. https://escholarship.org/uc/item/2r60s6b5.

[36] Zheng, Zibin, Shaoan Xie, Hongning Dai, Xiangping Chen, και Huaimin Wang: *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends.* Στο *2017 IEEE International Congress on Big Data (BigData Congress)*, σελίδες 557–564, Ιούνιος 2017.

[37] Zhou, Huan, Xue Ouyang, Z. Ren, Jinshu Su, C. D. Laat, και Zhiming Zhao: *A Blockchain based Witness Model for Trustworthy Cloud Service Level Agreement Enforcement.* IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019.

[38] Önday, Özgür: *Web 6.0: Journey From Web 1.0 To Web 6.0.* Journal of Media & Management, Δεκέμβριος 2019.