Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας

Μελέτη επίδοσης του Blockchain συστήματος Hyperledger Fabric

Διπλωματικη Εργασια

του

ΠΑΝΑΓΙΩΤΗ Δ. ΤΖΩΡΤΖΗ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής

Αθήνα, Οκτώβριος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας

Μελέτη επίδοσης του Blockchain συστήματος Hyperledger Fabric

Διπλωματικη Εργασια

του

ΠΑΝΑΓΙΩΤΗ Δ. ΤΖΩΡΤΖΗ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29 Οκτωβρίου 2020.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Νεκτάριος Κοζύρης	Διονύσιος Πνευματικάτος	Γεώργιος Γκούμας
Καθηγητής	Καθηγητής	Επίκουρος Καθηγητής

Αθήνα, Οκτώβριος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Παναγιώτης Τζώρτζης

29 Οκτωβρίου 2020

Περίληψη

Η τεχνολογία Blockchain τα τελευταία χρόνια έχει δημιουργήσει πολλές δυνατότητες και λειτουργικότητες που στο παρελθόν ήταν αδύνατες. Επιτρέπει σε οντότητες που δεν έχουν εμπιστοσύνη η μία στην άλλη να κάνουν συναλλαγές με έναν σαφή και αδιάβλητο τρόπο. Μετά την πρώτη του εφαρμογή στο πεδίο των κρυπτονομισμάτων, το Blockchain έχει επεκταθεί σε πολλές περιοχές εφαρμογής επαναπροσδιορίζοντας τον τρόπο συναλλαγής και ανταλλαγής πληροφορίας.

Παρά τις αδιαμφισβήτητα σημαντικές του δυνατότητες, τα συστήματα Blockchain μέχρι στιγμής πάσχουν απο σημαντικούς περιορισμούς της απόδοσής τους.

Στόχος της διπλωματικής εργασίας είναι η μέτρηση της απόδοσης μιας δημοφιλής υλοποίησης Blockchain, του Hyperledger Fabric και ο προσδιορισμός των αρχιτεκτονικών παραμέτρων που επηρεάζουν την απόδοσή του.

Λέξεις Κλειδιά

Κατανεμημένα Συστήματα, Blockchain, Hyperledger Fabric, Distributed Ledger Technologies, Consensus

Abstract

Blockchain technology has created many capabilities and functionalities which were impossible in the past. It allows entities that do not trust each other to make transactions in a concrete and safe way. After Blockchain's first use case application in cryptocurrencies, it has evolved to a variety of different use cases, redifining the way that entities and organisations transact with each other and disclose information.

Despite that potential, Blockchain solutions available in the market are affected by serious bottlenecks on performance issues which limits its possible applications The goal of this thesis is to benchmark Hyperledger Fabric's performance, to identify bottlenecks and investigate the parameters that deeply affect performance.

Keywords

Blockchain, Hyperledger Fabric, Distributed Ledger Technologies, Consensus

στα παιδάκια με τα ψυχολογικά που γίνανε παίδαροι

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Νεκτάριο Κοζύρη για την επίβλεψη αυτής της διπλωματικής εργασίας και ειδικά την κ. Κατερίνα Δόκα για την κατανόηση και την καθοδήγησή της καθ΄όλη την διάρκεια. Ειδικό κεφάλαιο θα έπρεπε να αφιερωθεί για τους ανθρώπους εκείνους που όλα αυτά τα χρόνια με στήριξαν, με βοήθησαν και με αγάπησαν. Ανέχτηκαν την γκρίνια μου, με βοήθησαν να ξεπεράσω τις δυσκολίες και με βοήθησαν να διαμορφωθώ σαν άνθρωπος. Η εργασία αυτή θα ήταν πολύ διαφορετική χωρίς αυτούς.

Last but not least, I would like to acknowledge the importance of my cooperation with Daniel Porto, a PhD candidate from Technical University of Lisbon, with whom I had very interesting and useful conversations on my research subject.

Αθήνα, Οκτώβριος 2020

Παναγιώτης Τζώρτζης

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Πρόλογος

Η παρούσα εργασία εκπονήθηκε στο CsLab κατα την ακαδημαική χρονιά 2019-2020. Αποτελεί την λογική ροή της ενασχόλησής μου με τα Κατανεμημένα Συστήματα και μια βαθύτερη και πιο ουσιαστική ματιά στο πώς οργανώνεται ένα τέτοιο σύστημα σε περιβάλλον Production.

Κεφάλαιο 1

Εισαγωγή

Η τεχνολογία Blockchain έχει επαναπροσδιορίσει τις δυνατότητες των υπολογιστικών συστημάτων να ανταλλάσουν πληροφορίες. Πρίν το Blockchain, υπήρχε η έννοια του μεσάζοντα ο οποίος έπαιρνε το ρίσκο και αναλάμβανε να μεσολαβήσει μεταξύ των εμπλεκόμενων μερών σε μία συναλλαγή, λαμβάνοντας και το αντίστοιχο αντίτιμο για την υπηρεσία που παρέχει και το ρίσκο που αναλαμβάνει. Αυτό οδήγησε νομοτελειακά σε μια συγκεντρωτική λειτουργία της οικονομίας, με τους μεσάζοντες να έχουν τεράστια δύναμη. Απο υπολογιστικής άποψης, δημιούργησε δομές με μοναδικά σημεία αποτυχίας.

Η υιοθέτηση του Blockchain δημιουργεί αποκεντρωμένα συστήματα που λειτουργούν σε ένα περιβάλλον χωρίς εμπιστοσύνη, αλλά καταφέρνουν να διατηρούν την ίδια αλυσίδα συναλλαγών. Ουσιαστικά το Blockchain αποτελεί ένα Κατανεμημένο Λογιστικό Βιβλίο, στο οποίο άπαξ και γίνει μια καταχώρηση έχει αποδειχθεί δεν υπάρχει κανένας (σχεδον) τρόπος αυτή η καταχώρηση να αλλάξει.

Για να μπορέσουν οι τεχνολογίες Blockchain να αποκτήσουν ευρεία εφαρμογή είναι σημαντικό να έχουν μεγάλη απόδοση. Η απόδοση μετριέται με διάφορους τρόπους, πιο συνηθισμένος εκ των οποίων είναι η Διεκπεραιωτική Ικανότητα (Througput) η οποία μετριέται σε συναλλαγές ανα δευτερόλεπτο (TPS, Transactions per Second).

Μια απο τις βασικές κατηγοριοποιήσεις των διαφόρων υλοποιήσεων Blockchain έχει να κάνει με το ποιός έχει δικαίωμα συμμετοχής στο δίκτυο. Οι πρώτες υλοποιήσεις του Blockchain που δημιουργήθηκαν, όπως το Bitcoin και το Ethereum, είχαν δημόσιες αλυσίδες και ο οποιοσδήποτε μπορούσε να συμμετέχει στο δίκτυο (public blockchains). Αντιθέτως, οι πιο σύγχρονες υλοποιήσεις (όπως το Hyperledger Fabric) περιορίζουν την πρόσβαση και προσθέτουν επίπεδα ταυτοποίησης για τη συμμμετοχή στο δίκτυο (permissioned blockchains). Η επιλογή της μορφής της αλυσίδας είναι καθοριστική καθώς καθορίζει αρχιτεκτονικές επιλογές που επηρεάζουν άμεσα την απόδοση του δικτύου.

Υπαρχουν διάφορα πρωτόκολλα και αλγόριθμοι με τις οποίες εξασφαλίζεται η συμφωνία μεταξύ των μερών που συμμετέχουν σε κάθε δίκτυο για την έγκριση μιας συναλλαγής. Τα πρωτόκολλα συμφωνίας (Consensus) παίζουν σημαντικό ρόλο στην απόδοση, καθώς η διαδικασία συμφωνίας έιναι μία δαπανηρή διαδικασία απο υπολογιστική σκοπιά.

Τέλος, ένας βασικός παράγοντας που επηρεάζει την λειτουργία του δικτύου είναι οι παράμετροι λειτουργίας του. Παραδείγματα τέτοιων παραμέτρων είναι το πόσες συναλλαγές θα πακετάρονται σε ένα Block, κάθε πότε θα δημιουργείται ένα Block ανεξαρτήτως αριθμού συναλλαγών κλπ.

Γενικότερα, τα δίκτυα Blockchain διακρίνονται για την πολυπλοκότητα λειτουργίας τους και υπάρχουν πολλοί παράμετοι και τεχνικές που επιδέχονται βελτιστοποίησης, χωρίς να υπάρχει όμως πανάκεια, καθώς κάθε σχεδιαστική επιλογή που λαμβάνεται ωφελεί σε κάποια ζητήματα, αλλα δημιουργεί επιπλοκές σε άλλα.

1.1 Αντικείμενο της διπλωματικής

Ένα βασικό ζήτημα που προκύπτει για τα δίκτυα Blockchain είναι η μέτρηση της απόδοσής τους, με σκοπό την ανατροφοδότηση των δεδομένων των μετρήσεων για την εξαγωγή συμπερασμάτων για τις σχεδιαστικές επιλογές που έχουν γίνει τόσο σε επίπεδο δικτύου όσο και σε επίπεδο λογισμικού που εκτελείται στο δίκτυο (Smart Contract).

Αντικείμενο της διπλωματικής είναι η προσομοίωση δικτύων Blockchain πάνω στην πλατφόρμα του Hyperledger Fabric, σε κατανεμημένο περιβάλλον που προσομοιώνει ικανοποιητικά συνθήκες κανονικής λειτουργίας, με διάφορες παραμετροποιήσεις, και η διεξαγωγή μετρήσεων απόδοσης (Benchmarks) με την χρήση του εξειδικευμένου εργαλείου Hyperledger Caliper.

Μέσα απο αυτή την διαδικασία και αξιοποιώντας - ερμηνεύοντας τα αποτελέσματα των μετρήσεών μας, θα βγάλουμε χρήσιμα συμπεράσματα για τις παραμέτρους λειτουργίας του δικτύου, τον βαθμό στον οποίο επηρεάζεται απο κάθε παράμετρο, και μελλοντικά βήματα - βελτιώσεις που μπορούν να γίνουν στην πλατφόρμα του Fabric με σκοπό την βελτίωση της απόδοσής του.

Η διαδικασία της μέτρησης της απόδοσης είναι πολύ βασικό κομμάτι της διαδικασίας σχεδιασμού τόσο της ίδιας της πλατφόρμας, όσο και του εξειδικευμένου λογισμικού που τρέχει σε αυτήν και καθορίζει την προσαρμογή της σε κάθε σενάριο.

1.2 Related Work

Το πεδίο έρευνας σχετικά με την απόδοση των δικτύων Blockchain είναι σχετικά πρόσφατο, λόγω και του οτι αποτελεί καινούρια τεχνολογία. Παρόλα αυτά το ερευνητικό ενδιαφέρον σε αυτό το κομμάτι είναι υψηλό.

Οι Tien Tuan Anh Dinh et al [1] παραθέτουν το Blockbench, ένα Framework με στόχο την διεξαγωγή benchmarks σε δημοφιλείς blockchain ιδιωτικής αλυσίδας. Οι Saingre et al [2] προτείνουν το BCTMark, ένα γενικής χρήσης Framework για διεξαγωγή benchmarks σε emulated δίκτυα.

Όσον αφορά το θέμα της μέτρησης της απόδοσης του Hyperledger Fabric, οι Thakkar et al [3] διεξήγαγαν μια αναλυτική μελέτη της απόδοσης του Fabric και προτείνανε βελτιστοποιήσεις σε αυτό. Οι Javaid et al [4] επικτεντρώθηκαν στην επίδραση της validation φάσης στην απόδοση και την βελτιστοποίησή της. Οι Wang et al [5] διεξήγαγαν μία ακόμα μελέτη απόδοσης στο Fabric, με σκοπό να προσδιοριστούν τα bottlenecks της απόδοσης. Οι Nguyen et al [6] ασχολήθηκαν με την επίδραση των παραμέτρων του δικτύου στην απόδοση του Fabric και επικεντρώθηκαν στην επίδραση της γεωγραφικής κατανομής των κόμβων.

1.3 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε κεφάλαια. Το πρώτο κεφάλαιο είναι η εισαγωγή που μόλις διαβάσατε. Στο δεύτερο κεφάλαιο γίνεται μία θεωρητική εισαγωγή των εννοιών του Blockchain, του Fault Tolerance, και στην συνέχεια γίνεται μία αναλυτική περιγραφή του Hyperledger Fabric. Στο τρίτο κεφάλαιο παρουσιάζουμε το benchmark engine που χρησιμοποιήσαμε, το Hyperledger Caliper. Στο τέταρτο κεφάλαιο αναφερόμαστε στις επιλογές που κάναμε για την πραγματοποίηση των πειραμάτων, και εξηγούμε αναλυτικά τα στάδια που απαιτούνται για την δημιουργία ενός δικτύου Hyperledger Fabric. Το πέμπτο κεφάλαιο είναι χωρισμένο σε ενότητες, καθε μία εκ των οποίων αποτελεί ένα ξεχωριστό πείραμα. Σε κάθε πείραμα παραθέτουμε τα αποτελέσματα και εξάγουμε συμπεράσματα. Το έκτο κεφάλαιο είναι ο επίλογος, στον οποίο γίνεται μια σύνοψη της ερευνητικής διαδικασίας, και προτείνουμε μελλοντικές επεκτάσεις αυτής.

Μέρος I

Θεωρητικό Μέρος

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό θα γίνει μια αναλυτική παρουσίαση τόσο των γενικότερων εννοιών του Blockchain, όσο και των εργαλείων που χρησιμοποιήθηκαν στο πλαίσιο αυτής της εργασίας για την υλοποίηση του δικτύου και την μέτρηση της απόδοσής του.

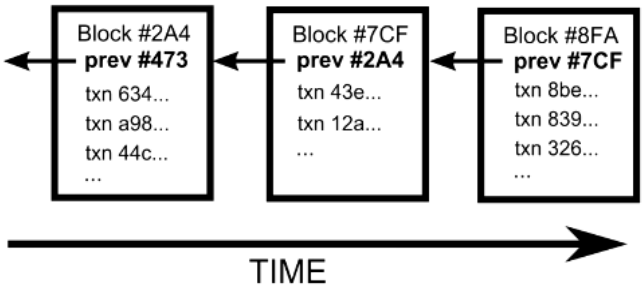
2.1 Blockchain

2.1.1 Το Blockchain απο μια αφαιρετική σκοπιά

Η τεχνολογία του Blockchain είναι ένα κοινόχρηστο, κατανεμημένο λογιστικό βιβλίο (ledger) στο οποίο καταγράφονται συναλλαγές μεταξύ οντοτήτων. Το βιβλίο αυτό είναι στη διάθεση πολλών διαφορετικών κόμβων του δικτύου, οι οποίοι δεν εμπιστεύονται ο ένας τον άλλον. Κάθε κόμβος διατηρεί το ίδιο αντίγραφο του βιβλίου με τους υπόλοιπους, το οποίο συνήθως κρατείται με την μορφή μιας αλυσίδας απο blocks.

Το κάθε block γενικά αποτελείται απο τα εξής δομικά στοιχεία:

- Μια αλληλουχία συναλλαγών, το περιεχόμενο των οποίων καθορίζεται απο την εφαρμογή που υλοποιείται
- Την επικεφαλίδα του block η οποία περιέχει την χρονοσφραγίδα δημιουργίας του και λοιπά μεταδεδομένα, τον κατακερματισμό των συναλλαγών και έναν δείκτη στον κατακερματισμό του προηγούμενου block.



Εικόνα 2.1: *Το εσωτερικό του blockchain*

Η ιδέα οτι κάθε block περιέχει τον κατακερματισμό του προηγούμενού του είναι πολύ σημαντική γιατι με αυτήν την απλή τεχνική σε συνδιασμό με έναν ισχυρό και αξιόπιστο αλγόριθμο συμφωνίας εξασφαλίζεται η ακεραιότητα της αλυσίδας. Συγκεκριμένα:

- Αν κάποιος αλλάξει το περιεχόμενο μιας συναλλαγής που βρίσκεται στην αλυσίδα, τότε θα μεταβληθεί το αποτέλεσμα κατακερματισμού του block στο οποίο βρίσκεται η συναλλαγή. Αυτό θα έχει σαν αποτέλεσμα τα blocks που είναι ύστερα απο αυτό στην αλυσίδα, να είναι πλέον άκυρα, γιατί ο δείκτης στο αποτέλεσμα κατακερματισμού του προηγούμενου block θα είναι πλέον άκυρος, καθώς μια απο τις συναλλαγές που δώθηκαν σαν είσοδος στην συνάρτηση κατακερματισμού είναι πλέον διαφορετική. Αποτέλεσμα αυτού είναι να σπάσει στα δύο η αλυσίδα.

Αυτό το δομικό στοιχείο του blockchain υπάρχει σε κάθε υλοποίηση, και μπορεί σε συνδιασμό με την χρήση ελέγχου ροής των συναλλαγών και ενός ισχυρού αλγόριθμου συμφωνίας να καταστήσει την αλλοίωση της αλυσίδας υπολογιστικά αδύνατη.

Με την χρήση του blockchain δημιουργείται η δυνατότητα λειτουργίες που μέχρι προηγουμένως απαιτούσαν εμπιστοσύνη, και σαν αποτέλεσμα ήταν αναγκαία η διαμεσολάβηση ενός τρίτου μέρους στην συναλλαγή (trusted third party το οποίο αναλάμβανε το ρίσκο και πληρωνόταν για αυτό, να πραγματοποιηθούν με βέβαιο τρόπο σε ένα περιβάλλον που δεν υπάρχει κανενός είδους εμπιστοσύνη.

Η δομή του blockchain απο μόνη της δεν εξασφαλίζει την αδιαβλητότητα του συστήματος. Θεωρητικά θα μπορούσε κανείς να υπολογίσει όλα τα επόμενα αποτελέσματα κατακερματισμού και να πείσει το δίκτυο οτι η δικιά του, αλλοιωμένη έκδοση της αλυσίδας, είναι δεκτή. Μπορούμε όμως, ανάλογα με το περιβάλλον στο οποίο λειτουργεί το δίκτυο, να συνδιάσουμε την δομή του blockchain με μία κατάλληλη στρατηγική που θα πρέπει να ακολουθείται για να επιτευχθεί συναίνεση μεταξύ των κόμβων για να θεωρηθεί μια συναλλαγή ως έγκυρη, με αποτέλεσμα να καταγραφεί στο ledger.

2.1.2 Δημόσιες και Περιορισμένης πρόσβασης αλυσίδες

Ένας βασικός διαχωρισμός μεταξύ των διαφόρων υλοποιήσεων του blockchain είναι το ποιός έχει πρόσβαση στο δίκτυο, ποιός μπορεί δηλαδή να δεί τα δεδομένα και να υποβάλλει συναλλαγές, καθώς και να συμμετάσχει στην διαδικασία της συμφωνίας. Ανάλογα λοιπόν με το αν η αλυσίδα είναι προσβάσιμη στον καθένα ή όχι, χωρίζουμε τις αλυσίδες σε δύο μεγάλες κατηγορίες:

- Τις δημόσιες αλυσίδες
- Τις περιορισμένης πρόσβασης αλυσίδες

Οι δημόσιες αλυσίδες (public blockchains) έχουν το χαρακτηριστικό οτι οποιοσδήποτε θέλει μπορεί να συμμετάσχει, να δεί τα δεδομένα, να πραγματοποιήσει συναλλαγές, να συμμετέχει στην διαδικασία συμφωνίας (consensus) της αλυσίδας. Γνωστές υλοποιήσεις που επιτρέπουν την πρόσβαση στον οποιονδήποτε είναι το Bitcoin [7] και το Ethereum [8].

Το γεγονός οτι μπορεί ο οποιοσδήποτε να συμμετέχει στο δίκτυο έχει σαν αποτέλεσμα να μην υπάρχει καμία απολύτως εμπιστοσύνη μεταξύ των συμμετέχοντων, αφου κανενός η ταυτότητα δεν είναι επιβεβαιωμένη. Αυτό σημαίνει οτι για να εξασφαλιστεί η αδιαβλητότητα του δικτύου πρέπει να υλοποιηθεί κάποιο αυστηρό, και συνήθως δαπανηρό, πρωτόκολλο συμφωνίας, όπως το Proof of Work που υλοποιεί το Bitcoin. Το εν λόγω πρωτόκολλο είναι

σημαντικό να επιλύει το πρόβλημα των Βυζαντινών Στρατηγών που θα παρουσιαστεί παρακάτω, αφου είναι σχεδόν σίγουρο οτι θα υπάρξουν κόμβοι που λειτουργούν δόλια.

Απο την άλλη, οι περιορισμένης πρόσβασης αλυσίδες προσθέτουν επίπεδα ασφαλείας στο δίκτυο και απαιτούν επιβεβαίωση της ταυτότητας του κάθε κόμβου που συμμετέχει. Για αυτό τον λόγο τα δίκτυα αυτά λειτουργούν σε ένα περιβάλλον μερικής εμπιστοσύνης, πράγμα που κάνει την διαδικασία συμφωνίας πιο απλή και λιγότερο χρονοβόρα σε σχέση με τα αυστηρά πρωτόκολλα στις δημόσιες αλυσίδες. Σε αντίθεση με τις δημόσιες αλυσίδες, δεν είναι απαραίτητο το πρωτόκολλο να επιλύει το πρόβλημα των βυζαντινών στρατηγών, καθώς το γεγονός οτι υπάρχει επαλήθευση για την συμμετοχή στο δίκτυο προσθέτει σημαντική αξιοπιστία.

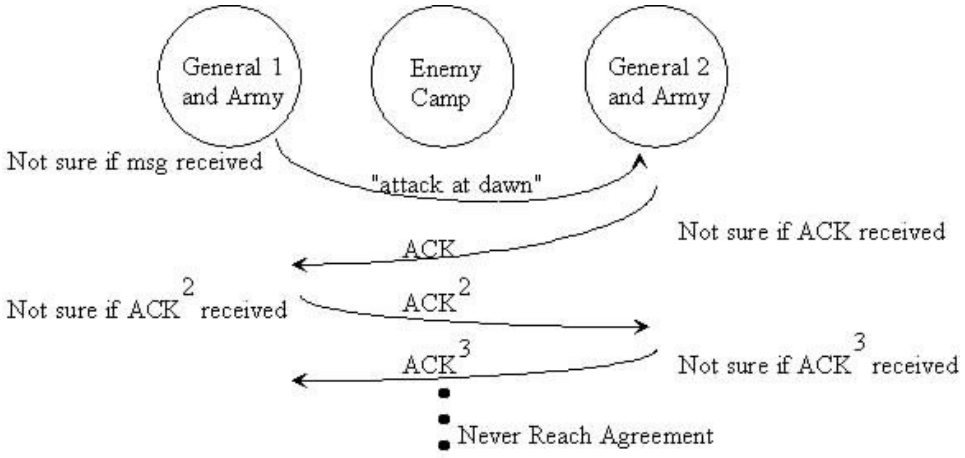
2.1.3 Το πρόβλημα των βυζαντινών στρατηγών

Το πρόβλημα των βυζαντινών στρατηγών είναι ένα απο τα πιό σημαντικά υπολογιστικά προβλήματα και απασχολεί τον κλάδο των κατανεμημένων συστημάτων, καθώς μοντελοποιεί και εξηγεί τις διάφορες καταστάσεις και λόγους αποτυχίας ενός συστήματος. Περιγράφει μια κατάσταση όπου οι εμπλεκόμενες οντότητες πρέπει να συμφωνήσουν σε μια κοινή απόφαση για να αποφευχθεί η πλήρης αποτυχία, αλλά κάποια απο τα εμπλεκόμενα μέρη είναι διεφθαρμένα και μπορεί να μεταδίδουν ψευδείς πληροφορίες ή είναι γενικώς αναξιόπιστα.

Για λόγους πληρότητας αξίζει να αναφερθούμε στο πρόβλημα των 2 στρατηγών, που αποτελεί την πρώτη μορφή του προβλήματος των βυζαντινών στρατηγών.

Το προβλημα των 2 στρατηγων

Η πρώτη μορφή του προβλήματος των βυζαντινών στρατηγών δημοσιεύτηκε το 1975 [9] και περιγράφει ένα σενάριο όπου δύο στρατηγοί θέλουν να επιτεθούν σε έναν κοινό εχθρό. Ο ένας στρατηγός αποτελεί τον ηγέτη (leader) ενώ ο άλλος αποτελεί τον ακολουθητή (follower). Ο κάθε στρατός απο μόνος του δεν είναι αρκετός για να νικήσει τον αντίπαλο, άρα για να νικήσουν πρέπει να συνεργαστούν, να αποφασίσουν απο κοινού και να επιτεθούν ταυτόχρονα.



Εικόνα 2.2: *Το πρόβλημα των 2 στρατηγών*

Η επικοινωνία των 2 στρατηγών γίνεται μέσω αγγελιοφόρων, γεγονός που σημαίνει οτι μπορεί ο αγγελιοφόρος να πιαστεί απο τους εχθρούς και το μήνυμα να μην παραδοθεί. Ακόμα

και αν παραδοθεί όμως, ο παραλήπτης στρατηγός πρέπει να επιβεβαιώσει την λήψη του μηνύματος, με το να στείλει μία επιβεβαίωση με αγγελιοφόρο σαν απάντηση. Ομως η χρήση του αγγελιοφόρου έχει σαν αποτέλεσμα να επαναλαμβάνεται το προηγούμενο σενάριο όπου μπορεί να τον πιάσουν οι εχθροί.

Αυτο επεκτείνεται επ΄ άπειρον, καθώς πάντα θα υπάρχει αβεβαιότητα απο τον αποστολέα του τελευταίου μηνύματος, σχετικά με το άν το τελευταίο μήνυμα (ή επιβεβαίωση) παραλήφθηκε.

Το πρόβλημα των 2 στρατηγών έχει αποδειχτεί οτι είναι μή επιλύσιμο.

Το πρόβλημα των βυζαντινών στρατηγών

Το πρόβλημα των βυζαντινών στρατηγών διατυπώθηκε πρώτη φορά το 1982 [10]. Αποτελεί μια γενίκευση του προβλήματος των 2 στρατηγών για περισσότερους απο 2 στρατηγούς. Πέρα απο αυτό, βάζει την παράμετρο ότι πέρα απο την αποτυχία της αποστολής ενός μηνύματος λόγω της σύλληψης του αγγελιοφόρου, μπορεί κάποιος απο τους στρατηγούς να είναι προδότης, με αποτέλεσμα να στέλνει ψεύτικα μηνύματα για να αποπροσανατολίσει τους υπόλοιπους.

Το πρόβλημα μοντελοποιείται ως εξής: Αν έχουμε n στρατηγούς συνολικά εκ των οποίον ο 1 είναι ο διοικητής και οι n-1 υπόλοιποι είναι οι υπασπιστές του, τότε ο διοικητής πρέπει να στείλει μια εντολή στους n-1 υπασπιστές του έτσι ώστε

- Όλοι οι πιστοί υπασπιστές να υπακούσουν την ίδια εντολή
- Άν ο διοικητής είναι πιστός, τότε κάθε πιστός υπασπιστής πρέπει να υπακούσει την εντολή του

Το ενδιαφέρον είναι οτι ακόμα και αν είναι προδότης ο διοικητής, πρέπει οι πιστοί υπασπιστές να συμφωνήσουν σε μία κοινή απόφαση.

Έχει αποδειχτεί οτι το πρόβλημα των βυζαντινών στρατηγών είναι επιλύσιμο όταν έχουμε $3m + 1$ έντιμους κόμβους για m προδότες.

Ενδεικτικά, ενδιαφέρουσες επιλύσεις του προβλήματος των βυζαντινών στρατηγών που χρησιμοποιούνται απο δημοφιλείς υλοποιήσεις blockchain είναι ο Practical Byzantine Fault Tolerance [11], ο BFTSMART [12], καθως και η υλοποίηση του Bitcoin.

2.1.4 Byzantine Fault Tolerant και Crash Fault Tolerant συστήματα

Ένα κατανεμημένο σύστημα πρέπει να είναι σε θέση να αντιμετωπίζει σφάλματα και να καταλήγει σε συμφωνία. Τα σφάλματα που μπορούν να συμβούν σε ένα κατανεμημένο σύστημα μπορούν να ομαδοποιηθούν σε 2 κατηγορίες:

- Σφάλματα σχετικά με το δίκτυο (Crash Faults)
- Βυζαντινά σφάλματα (Byzantine Faults)

Ένα σύστημα το οποίο είναι σε θέση να αντέξει σφάλματα δικτύου (παραδείγματα τέτοιων σφαλμάτων είναι ένας κόμβος να βγεί εκτός λειτουργίας) και να συνεχίσει να λειτουργεί κανονικά και να διατηρεί την προηγούμενη κατάστασή του ονομάζεται Crash Fault Tolerant. Ένα σύστημα το οποίο χαρακτηρίζεται απο Crash Fault Tolerance εγγυάται την λειτουργία

του όσο τουλάχιστον N/2 + 1 κόμβοι είναι λειτουργικοί. Αντιθέτως, δέν λαμβάνει υπ'όψιν την ενδεχόμενη ύπαρξη διεφθαρμένων κόμβων που μπορεί να προσπαθούν να αποπροσανατολίσουν το δίκτυο και δεν εγγυάται σε καμία περίπτωση συμφωνία σε αυτό το σενάριο.

Ένα σύστημα το οποίο είναι σε θέση να αντέξει πέρα απο σφάλματα δικτύου, βυζαντινά σφάλματα, μπορεί δηλαδή να έρθει σε συμφωνία ακόμα και όταν υπάρχει ένας αριθμός απο διεφθαρμένους κόμβους (σύμφωνα με το πρόβλημα των βυζαντινών στρατηγών που προαναφέραμε) ονομαζεται Byzantine Fault Tolerant ή BFT. Λαμβάνει υπόιν την ύπαρξη διεφθαρμένων κόμβων στο δίκτυο και εγγυάται την επίτευξη συμφωνίας όταν έχουμε τουλάχιστον 3m + 1 έντιμους κόμβους για m διεφθαρμένους.

Με μία γρήγορη ματιά μπορεί κανείς να συμπεράνει οτι ένα BFT σύστημα είναι πιό ασφαλές και πρακτικά αδύνατο να παραβιαστεί. Συστήματα όπως το Bitcoin δεν έχουν παραβιαστεί ποτέ απο το 2008 που έχουν βγει στην κυκλοφορία. Παρόλα αυτά, κάθε σχεδιαστική επιλογή έχει κάποιο κόστος. Για να είναι ένα σύστημα BFT αναγκαζόμαστε να χρησιμοποιούμε εξαιρετικά πολύπλοκους και ενεργειακά κοστοβόρους αλγόριθμους που πραγματοποιούν πολλούς ¨περιττούς' υπολογισμούς. Αυτό έχει σημαντικό αντίκτυπο τόσο στο κόστος διαχείρησης του δικτύου, όσο και στην απόδοσή του. Οι τωρινές υλοποιήσεις BFT πάσχουν είτε σε επίπεδο κλιμακωσιμότητας (scalability) όπως ο PBFT, είτε σε επίπεδο απόδοσης όπως ο Proof Of Work που χρησιμοποιείται στο Bitcoin. Αντίθετα, οι CFT υλοποιήσεις όπως οι Kafka, RAFT, Paxos [13] υπερτερούν κατα πολύ σε ταχύτητα και scalability.

Ο βασικός παράγοντας που θα καθορίσει εν τέλει την επιλογή σχετικά με το άν ένα σύστημα πρέπει να είναι BFT ή απλώς CFT έχει να κάνει με το περιβάλλον λειτουργίας. Συγκεκριμένα, μια δημόσια αλυσίδα όπου ο καθένας μπορεί να συμμετάσχει δεν γίνεται να μην υλοποιήσει BFT, καθώς σε αντίθετη περίπτωση είναι πολυ εύκολο να παραβιαστεί το αναλλοίωτο της αλυσίδας. Όταν όμως μια αλυσίδα λειτουργεί σε περιβάλλον μερικής εμπιστοσύνης και αυξημένης ασφάλειας, που είναι η περίπτωση στις περιορισμένης πρόσβασης αλυσίδες, όπου κάθε οντότητα που συμμετέχει στο δίκτυο είναι γνωστή και ταυτοποιημένη, η υλοποίηση BFT δεν είναι απαραίτητη. Οι περισσότερες υλοποιήσεις που λειτουργούν σε περιβάλλον περιορισμένης πρόσβασης επιλέγουν να υλοποιούν μόνο CFT, καθώς η επιβάρυνση του συστήματος για την υλοποίηση BFT είναι σημαντική ενώ το όφελος σχετικά μικρό. Το Hyperledger Fabric που θα παρουσιάσουμε στη συνέχεια και είναι permissioned blockchain δεν υλοποιεί BFT.

2.1.5 Αλγόριθμοι και πρωτόκολλα επίτευξης συμφωνίας

Έχοντας κάνει αναφορά τόσο στη δομή του Blockchain όσο και στα διαφορετικά χαρακτηριστικά του κάθε αλγόριθμου, μπορούμε πλέον να ολοκληρώσουμε το εισαγωγικό μέρος στην λειτουργία του blockchain αναλύοντας τον ρόλο και την λειτουργία του βασικότερου κομματιού της αλυσίδας: τον αλγόριθμο επίτευξης συμφωνίας.

Εφαρμογές των αλγόριθμων συμφωνίας είναι οι εξής:

- απόφαση σχετικά με το άν πρέπει να καταγραφεί μία συναλλαγή στο blockchain
- εκλογή ενός κόμβου σαν ηγέτη (leader) για κάποια κατανεμημένη διεργασία
- συγχρονισμός αντιγράφων (state machine replication)

Κατάρχάς, για την επίτευξη συμφωνίας σε κάθε υλοποίηση πολλά διακριτά μέρη συνεργάζονται. Σημαντικό για την επίτευξη συμφωνίας, πέρα απο το κομμάτι του consensus, είναι οι διαδικασίες που κάνει ένα δίκτυο για την επικύρωση της εγκυρότητας των συναλλαγών (validation), τον τρόπο με τον οποίο αυτές καταγράφονται στο Distributed Ledger κλπ. Γενικά, όλος ο κύκλος ζωής των συναλλαγών συμβάλλει στην επίτευξη συμφωνίας.

Οι αλγόριθμοι συμφωνίας είναι αλγόριθμοι οι οποίοι είναι υπεύθυνοι για την επίτευξη συμφωνίας σε μία τιμή ανάμεσα σε κατανεμημένες διεργασίες ή συστήματα. Είναι σχεδιασμένοι για να πετυχαίνουν τον ρόλο τους με αξιοπιστία σε δίκτυα παρά τις όποιες αποτυχίες των κόμβων, και, κάποιοι εξ αυτών, ακόμα και όταν το δίκτυο έχει μη αξιόπιστους κόμβους (όπως στα σενάρια που προαναφέρθηκαν στο προηγούμενο κεφάλαιο). Για να το πετύχουν αυτό, οι αλγόριθμοι συμφωνίας είναι απαραίτητο να λαμβάνουν υπ΄όψιν οτι καποιες διεργασίες και κόμβοι κάποια στιγμή δεν θα είναι διαθέσιμοι, και οτι κάποια πακέτα επικοινωνίας θα χαθούν. Ένας αλγόριθμος συμφωνίας για να δουλέψει σωστά πρέπει να έχει αντοχή στα σφάλματα (fault tolerance) [14].

Ένας αλγόριθμος συμφωνίας πρέπει να έχει τις εξής ιδιότητες:

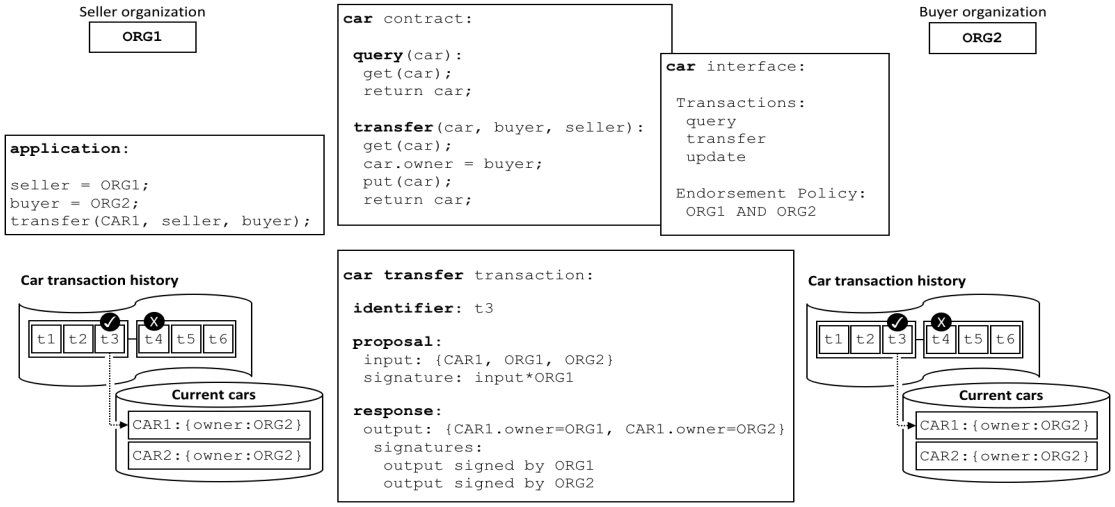
- Τερματισμός (termination): Όλοι οι έντιμοι κόμβοι κάποια στιγμή θα ολοκληρώσουν την διαδικασία υπολογισμού και θα καταλήξουν σε κάποια απόφαση
- Ακεραιότητα (integrity): Άν όλες οι έντιμες διεργασίες προτείναν μία τιμή, τότε κάθε έντιμη διεργασία πρέπει να συμφωνήσει με αυτήν την τιμή
- Συμφωνία (agreement): Όλοι οι έντιμοι κόμβοι του δικτύου πρέπει να συμφωνήσουν στην ίδια τιμή.

Ανάλογα με την υλοποίηση της αλυσίδας, οι αλγόριθμοι consensus εμπλέκονται σε διάφορα στάδια και συνεισφέρουν στην διατήρηση της συνοχής (consistency) του δικτύου.

2.1.6 Έξυπνα Συμβόλαια

Η πρώτη εφαρμογή του Blockchain είχε να κάνει με ένα αποκεντρωμένο σύστημα χρηματικών συναλλαγών μεταξύ οντότητων που δεν εμπιστεύονταν η μία την άλλη, χωρίς την χρήση μεσάζοντων. Με αυτόν τον τρόπο εξαλείφθηκαν τα ζητήματα κόστους, ιδιωτικότητας και ασφάλειας που διεγείρονταν με την χρήση μεσάζοντων. Η εφαρμογή του Blockchain στο πεδίο των χρηματικών συναλλαγών ήταν μια σπουδαία καινοτομία, τόσο σημαντική που αρχικά η τεχνολογία ουσιαστικά ταυτίστηκε με την περάτωση χρηματικών συναλλαγών. Παρόλα αυτά, το Blockchain μπορεί να εφαρμοστεί σε πάρα πολλές περιπτώσεις που δεν έχουν να κάνουν με αυτό το σενάριο. Η επέκταση της λειτουργικότητας του Blockchain έγινε με την προσθήκη έξυπνων συμβολαίων στον κώδικά του.

Ένα έξυπνο συμβόλαιο (Smart Contract) είναι ένα πρόγραμμα το οποίο τρέχει στο Blockchain [15]. Ο κώδικας ενός έξυνου συμβολαίου είναι συγκεκριμένος και δέν μπορεί να αλλάξει άπαξ και προστεθεί στην αλυσίδα. Μια καλή αναλογία για το έξυπνο συμβόλαιο είναι οτι το έξυπνο συμβόλαιο λειτουργεί σαν μεσάζοντας μεταξύ οντοτήτων που δεν εμπιστεύονται η μία την άλλη για να εκτελεστούν προκαθορισμένες και προσυμφωνημένες λειτουργίες [16].



Εικόνα 2.3: *Διάγραμμα έξυπνου συμβολαίου στο Hyperleder Fabric*

Η απαγκίστρωση του Blockchain απο τις χρηματικές συναλλαγές μέσω της χρήσης έξυπνων συμβολαίων ουσιαστικά επιτρέπει την χρήση της λειτουργικότητας και των πλεονεκτημάτων του Blockchain σε πολλά πεδία εφαρμογής. Το έξυπνο συμβόλαιο εμφολεύει όλη την λογική μιας εφαρμογής (business logic).

Τα έξυπνα συμβόλαια μπορούν να αναπτυχθούν σε διάφορες πλατφόρμες, κυριότερες εκ των οποίων είναι το Bitcoin, το Ethereum και το Hyperledger Fabric. Στην περίπτωση του Bitcoin, παρέχεται μια χαμηλού επιπέδου γλώσσα για την ανάπτυξη έξυπνων συμβολαίων που έχει πολλούς περιορισμούς. Το Ethereum απο την άλλη παρέχει μια Turing-Complete γλώσσα που ονομάζεται Solidity, ενώ υποστηρίζει και άλλες γλώσσες προγραμματισμού. Οι δυο προαναφερθείσες πλατφόρμες πάσχουν σε επίπεδο απόδοσης και κλιμακωσιμότητας. Η λύση του Hyperledger Fabric σε ότι αφορά τα έξυπνα συμβόλαια ονομάζεται chaincode και αποτελείται απο τον κώδικα που τρέχει στην αλυσίδα, την βάση δεδομένων κατάστασης (state database) και την διαδικασία επικύρωσης (endorsement).

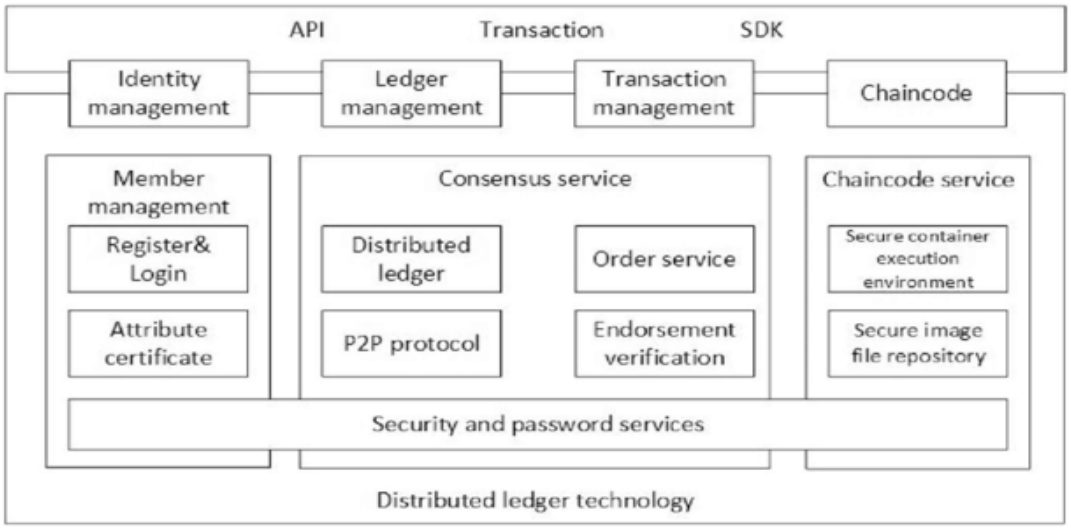
Στο πλαίσιο της παρούσας διπλωματικής δεν μεταβάλλουμε το έξυπνο συμβόλαιο. Αντιθέτως, το θεωρούμε μια σταθερά πάνω στην οποία μεταβάλλουμε άλλους παράγοντες του δικτύου (όπως θα παρουσιαστεί παρακάτω).

2.2 Η υλοποίηση του Blockchain στην πλατφόρμα του Hyperledger Fabric

Η τεχνολογία του Blockchain γνώρισε τεράστια απήχηση με το Bitcoin και στην συνέχεια με την δημιουργία του Ethereum το οποίο πρωτοεισήγαγε την λογική των έξυπνων συμβολαίων στην αλυσίδα. Οι δύο αυτές υλοποιήσεις λειτουργούν σε ένα περιβάλλον δημόσιο, χωρίς ταυτοποίηση των χρηστών ή έλεγχο πρόσβασης. Η προσαρμογή της τεχνολογίας του Blockchain σε πιο συγκεκριμένες, διαφορετικών απαιτήσεων καταστάσεις απαιτεί την χρήση περιορισμών στο ποιός έχει πρόσβαση στο δίκτυο. Το Hyperledger Fabric είναι μια υλοποίηση Blockchain περιορισμένης πρόσβασης.

2.2.1 Τι ειναι το Hyperledger Fabric

Το Hyperledger Fabric είναι μία ανοιχτού κώδικα υλοποίηση της τεχνολογίας κατανεμημένου βιβλίου (Distributed Ledger Technology, DLT). Το Fabric προσφέρει πλήρως παραμετροποιημένη και αρθρωτή αρχιτεκτονική, ώντας έτσι εύκολα προσαρμόσιμο σε πολλά διαφορετικά σενάρια εφαρμογής. Είναι το πρώτο DLT πυ δίνει την δυνατότητα να γράφει κανείς κατανεμημένες εφαρμογές (Distributed Applications, Dpps) σε συνηθισμένες γλώσσες προγραμματισμού, μειώνοντας σημαντικά με αυτόν τον τρόπο το κόστος ανάπτυξης κατανεμημένων εφαρμογών, αφού χρησιμοποιούνται ήδη γνωστές τεχνολογίες. Παράλληλα, δίνει την εντύπωση οτι η εκτέλεση της εφαρμογής γίνεται σε ένα ενιαίο κατανεμημένο υπολογιστή. Αυτή η ιδιότητα είναι που κάνει το Fabric να ξεχωρίζει σε σχέση με άλλες υλοποιήσεις, και για αυτό οι δημιουργοί του το ορίζουν ως "Το πρώτο κατανεμημένο λειτουργικό σύστημα για περιορισμένης πρόσβασης αλυσίδες" [17].



Εικόνα 2.4: *Η αρθρωτή αρχιτεκτονική του Hyperleder Fabric*

Το Fabric λειτουργεί σε περιβάλλον περιορισμένης πρόσβασης, όπου οι συμμετέχοντες είναι γνωστοί και ταυτοποιημένοι. Με αυτόν τον τρόπο μπορεί οι διάφοροι συμμετέχοντες να μην εμπιστεύονται πλήρως ο ένας τον άλλον, μπορούν όμως να λειτουργούν σε ένα περιβάλλον μερικής εμπιστοσύνης. Αυτό, σε πολλές περιπτώσεις, επιτρέπει την χρήση Crash Fault Tolerant αλγορίθμων επίτευξης συμφωνίας, οι οποίοι κατα κανόνα είναι λιγότερο απαιτητικοί σε υπολογιστικούς πόρους (και κατα συνέπεια, έχουν μικρότερο κόστος λειτουργίας). Επίσης αξίζει να αναφερθεί το γεγονός οτι το Fabric απο μόνο του δεν έχει κάποιο κρυπτονόμισμα (όπως πχ τα Bitcoin, Ethereum).

Το γεγονός οτι η αρχιτεκτονική του Fabric είναι αρθρωτή και οτι το Fabric είναι λογισμικό ανοιχτού κώδικα διευκολύνει την προσαρμογή του στις απαιτήσεις της εφαρμογής. Για παράδειγμα, κάποιοι παράγοντες του δικτύου που μπορούν εύκολα να αλλάξουν είναι οι εξής:

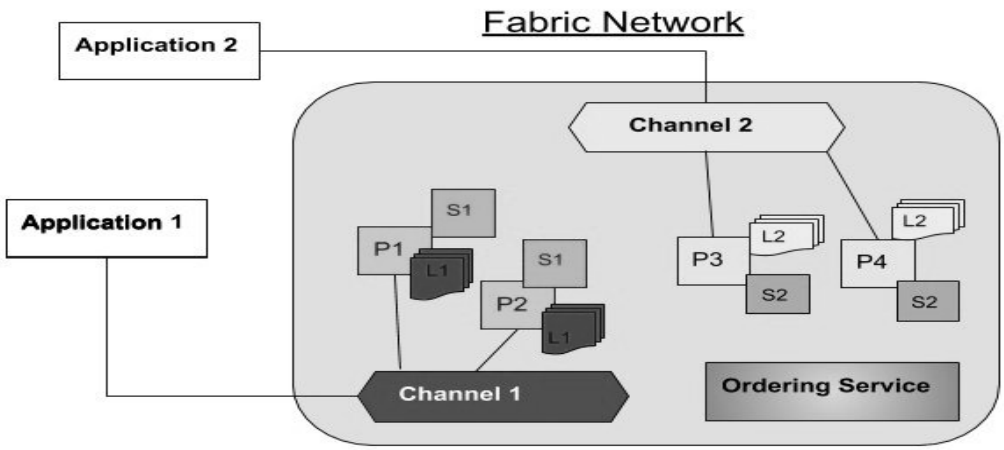
- Ο αλγόριθμος συμφωνίας που χρησιμοποιείται απο το Ordering Service
- Το είδος της βάσης δεδομένων που χρησιμοποιείται για την αποθήκευση της κατάστασης του συστήματος (Wold State Database)

- Το κομμάτι του δικτύου που ασχολείται με την κατανομή ψηφιακών κρυπτογραφικών ταυτότητων στις διάφορες οντότητες που συμμετέχουν στο δίκτυο
- Η πολιτική που ακολουθεί το δίκτυο για την επικύρωση και καταγραφή συναλλαγών

Εμπιστευτικότητα και ιδιωτικά δεδομένα

Στις διάφορες υλοποιήσεις Blockchain δημόσιας αλυσίδας υπήρχε το πρόβλημα των ιδιωτικών δεδομένων και της εμπιστευτικότητας. Με την χρήση αλγορίθμων συμφωνίας όπως ο Proof of Work, κάθε συναλλαγή εκτελούταν απο όλους τους κόμβους του συστήματος και ήταν γνωστή σε όλους, όπως και ο κώδικας του έξυπνου συμβολαίου που θα εκτελεστεί. Λύσεις που δώθηκαν σε αυτό το πρόβλημα ήταν η κρυπτογράφηση των δεδομένων και η χρήση αποδείξεων μηδενικής γνώσης (Zero Knowledge Proofs [18]), λύσεις όμως που δεν έλυναν πλήρως το πρόβλημα και δημιουργούσαν διάφορους προβληματισμούς.

Το Fabric χρησιμοποιεί ένα στοιχείο της αρχιτεκτονικής του, τα κανάλια (channels) για να λύσει αυτό το πρόβλημα. Ουσιαστικά οι συμμετέχοντες στο δίκτυο δημιουργούνε κανάλια επικοινωνίας μεταξύ τους και διατηρείται ξεχωριστή αλυσίδα για κάθε κανάλι. Κάθε κόμβος μπορεί να δεί τις αλυσίδες που σχετίζονται με τα κανάλια στα οποία συμμετέχει, ενώ δεν έχει πρόσβαση σε πληροφορίες που δεν τον αφορούν άμεσα.

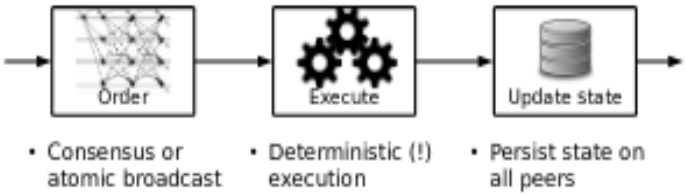


Εικόνα 2.5: *Η χρήση καναλιών για απομόνωση των αλυσίδων*

Αρχιτεκτονική κύκλου ζωής των συναλλαγών

Στις περισσότερες υλοποιήσεις Blockchain το προτόκολλο συμφωνίας πρώτα ορίζει μία σειρά μεταξύ των συναλλαγών (order) και στην συνέχεια τις στέλνει σε όλους τους κόμβους, οι οποίοι τις εκτελούν με την σειρά (execute). Όλοι οι κόμβοι εκτελούν τις συναλλαγές, οι οποίες πρέπει να είναι ντετερμινιστικές, να βγάζουν δηλαδή πάντα το ίδιο αποτέλεσμα, πράγμα που είναι σχεδόν αδύνατο να εξασφαλιστεί αν τα έξυπνα συμβόλαια που καλούνται με αυτές τις συναλλαγές γραφτούν με κάποια γενικού σκοπού γλώσσα προγραμματισμού. Αντιθέτως, εξ αιτίας της αρχιτεκτονικής order-execute και της απαίτησης για ντετερμινιστικές συναλλαγές, η

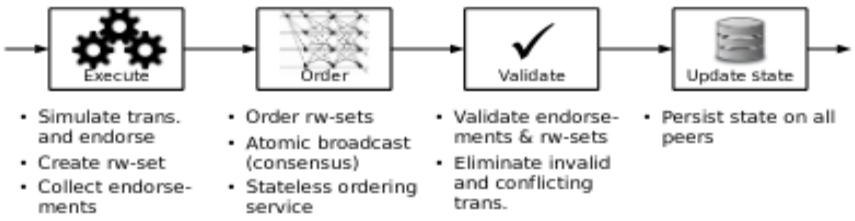
δημιουργία των έξυπνων συμβολαίων γίνεται με ειδικές, συγκεκριμένου σκοπού γλώσσες που είναι μεν αρκετά εκφραστικές για την χρήση που προορίζονται, αλλα έχουν περιορισμούς. Το μοντέλο order-execute απαιτεί σειριακή εκτέλεση των συναλλαγών σε όλους τους κόμβους, πράγμα που επηρεάζει αρνητικά την απόδοση του συστήματος.



Εικόνα 2.6: *Η αρχιτεκτονική order-execute*

Η υλοποίηση του κύκλου ζωής συναλλαγών στο Fabric διαφοροποιείται αρκετά σε σχέση με άλλες υλοποιήσεις. Αντί για το παραδοσιακό μοντέλο order-execute επιλέγει το μοντέλο execute-order-validate. Διαχωρίζεται ο κύκλος ζωής μιας συναλλαγής σε τρεις φάσεις:

- Εκτέλεση της συναλλαγής και έλεγχος της εγκυρότητάς της
- Χρήση ενός αλγόριθμου συμφωνίας για να συμφωνηθεί μία σειρά για τις συναλλαγές
- Επικύρωση της συναλλαγής



Εικόνα 2.7: *Η αρχιτεκτονική execute-order-validate*

Το καινοτόμο είναι οτι το Fabric εκτελεί τις συναλλαγές πριν καθορίσει την τελική τους χρονική σειρά. Αναλυτική περιγραφή του μοντέλου αυτού θα υπάρξει σε παρακάτω ενότητα, μαζί με την επεξήγηση της ροής μιας συναλλαγής στο Fabric.

2.2.2 Δομικά στοιχεία του δικτύου

Ένα δίκτυο στο Hyperledger Fabric αποτελείται απο πολλούς διαφορετικούς κόμβους με εξειδικευμένες αρμοδιότητες ο καθένας. Κάθε κόμβος του Fabric ζεί απομονομένος μέσα σε ένα περιβάλλον εικονικοποίησης ((docker container) και επικοινωνεί με τους υπόλοιπους με την χρήση του πρωτοκόλλου gRPC [19].

Clients

Οι clients (πελάτες) είναι εξωτερικά στοιχεία του δικτύου. Αντικατοπτρίζουν τον τελικό χρήστη της εφαρμογής και είναι υπεύθυνοι για την δημιουργία συναλλαγών. Επικοινωνούν τόσο με τους peers όσο και με τους orderers.

Peers

Οι peers αποτελούν ένα απο τα βασικότερα κομμάτια του δικτύου. Οι peers είναι οι κόμβοι του δικτύου που κρατάνε αντίγραφο της αλυσίδας και του Έξυπνου συμβολαίου που εκτελείται απο κάθε κανάλι του δικτύου. Οι peers χωρίζονται σε δύο κατηγορίες: τους endorsers (endorsing peers) και τους commiters (commiting peers), ανάλογα με τον διακριτό ρόλο που έχουν στο δίκτυο. Συγκεκριμένα:

- Οι endorsers είναι peers που έχουν οριστεί απο την πολιτική του δικτύου ως κόμβοι που εκτελούν προσομοιώσεις συναλλαγών κατα την διαδικασία εκτέλεσης μιας συναλλαγής, απαντώντας σε αιτήματα endorsement requests. Οι endorsers ουσιαστικά υλοποιούν το στάδιο execute της αρχιτεκτονικής execute-order-validate.
- Οι commiters είναι υπεύθυνοι για την επαλήθευση των blocks και των συναλλαγών που αυτά περιέχουν και είναι αυτοί που εν τέλει καταγράφουν και προσθέτουν το block στην αλυσίδα. Οι commiters ουσιαστικά υλοποιούν το στάδιο validate της αρχιτεκτονικής execute-order-validate.

Όλοι οι peers έχουν τον ρόλο του commiter, όμως συγκεκριμένοι peers έχουν επιπροσθέτως τον ρόλο του endorser. Ο διακριτός τους ρόλος θα γίνει ξεκάθαρος στην παρακάτω ενότητα, που θα περιγραφεί εκτενώς ο κύκλος ζωής μίας συναλλαγής.

Orderers

Οι Orderers απαρτίζουν το Ordering Service που είναι υπεύθυνο για την συμφωνία στην σειρά με την οποία οι συναλλαγές θα καταγραφούν στην αλυσίδα. Υπάρχουν διάφορες υλοποιήσεις για το Ordering Service, καθώς το Fabric επιτρέπει την δημιουργία τόσο ενός κεντρικού Ordering Service με μόνο έναν Orderer (πράγμα χρήσιμο μόνο σε περιβάλλοντα ανάπτυξης εφαρμογών και όχι σε κανονικές συνθήκες λειτουργίας), όσο και κατανεμημένων και αποκεντρωμένων Ordering Services που περιλαμβάνουν πολλούς Orderers οι οποίοι χρησιμοποιούν κάποιο πρωτόκολλο συμφωνίας για να επικοινωνήσουν μεταξύ τους. Αυτή η τακτική εξασφαλίζει οτι δεν θα υπάρχει ένα κεντρικό σημείο αποτυχίας για το σύστημα, δηλαδή ένας κόμβος που αν πάψει να αποκρίνεται, το σύστημα θα καταρρεύσει.

Certificate Authority (CA)

Ο κόμβος Certificate Authority (CA) διατηρεί τις ταυτότητες όλων τον κόμβων του δικτύου ((clients, peers, orderers) και είναι υπεύθυνος για την αντιστοίχηση ψηφιακών πιστοποιητικών σε κάθε κόμβο που θα χρησιμοποιηθούν για την ταυτοποίησή τους κατα την λειτουργία του δικτύου. Το Fabric χρησιμοποιεί κρυπτογραφικά πιστοποιητικά X509 για την

υλοποίηση της ταυτοποίησης και της αυθεντικοποίησης των κόμβων του δικτύου. Το Fabric παρέχει δικιά του υλοποίηση του CA, παρόλα αυτά, σύμφωνα με την αρθρωτή αρχιτεκτονική του, μπορεί να χρησιμοποιηθεί και διαφορετική υλοποίηση.

Το Fabric χρησιμοποιεί δύο διαφορετικές τεχνικές για την αρχικοποίηση των κρυπτογραφικών πιστοποιητικών του κάθε κόμβου του δικτύου. Στην πρώτη, όλα δημιουργούνται πρίν την δημιουργία του δικτύου απο έναν CA και κατανέμονται σε κάθε κόμβο. Στην δεύτερη, δημιουργείται πρώτα ένας CA ο οποίος στην συνέχεια, για κάθε οντότητα του δικτύου (πχ peer) που έρχεται η ώρα να δημιουργηθεί, δυναμικά δημιουργεί τα πιστοποιητικά του.

Chaincode

Η εκτέλεση του έξυπνου συμβολαίου (ή αλλιώς chaincode, σύμφωνα με την ορολογία του Fabric) συμβαίνει σε απομονομένο περιβάλλον απο το περιβάλλον του endorsing peer. Κάθε chaincode τρέχει σε διαφορετικό Docker container, γεγονός που απομονώνει τα έξυπνα συμβόλαια τόσο απο τους peers, όσο και μεταξύ τους. Το container του chaincode επικοινωνεί με τον peer με μυνήματα gRPC.

Εκτός απο το chaincode που αντιπροσωπεύει το έξυπνο συμβόλαιο σε επίπεδο εφαρμογής, μέσα στον peer τρέχει και chaincode συστήματος, το οποίο υλοποιεί πολλές λειτουργικότητες που είναι απαραίτητες για την λειτουργία του δικτύου.

Οργανισμοί

Οι οργανισμοί αποτελούν τις οντότητες στις οποίες ¨ανήκουν' οι peers. Χρησιμεύουν για την προσαρμογή του δικτύου σε σενάρια του πραγματικού κόσμου. Κάθε πεερ ανήκει σε έναν οργανισμό.

Πολιτική Endorsement

Η πολιτική Endorsement καθορίζει το ποιοί και πόσοι απο το σύνολο των endorsing peers πρέπει να εκτελέσουν προσομοίωση της συναλλαγής που προτείνει ο πελάτης (client) ώστε αυτή να γίνει αποδεκτή απο το δίκτυο. Μία έγκυρη συναλλαγή οφείλει να έχει συνδιασμό απαντήσεων (endorsement responses) τέτοιο ώστε να ικανοποιείται η πολιτική Endorsement. Σε διαφορετική περίπτωση, η συναλλαγή κρίνεται άκυρη. Η πολιτική endorsement δεν αποτελεί χαρακτηριστικό του δικτύου, αλλά του chaincode στο οποίο αναφέρεται. Αυτό σημαίνει οτι σε ένα δίκτυο με πολλά κανάλια που υπάρχουν πολλά chaincodes μπορούν να υπάρχουν διαφορετικές πολιτικές.

Η πολιτική Endorsement μπορεί να απαιτεί συγκεκριμένους endorsers, ένα μίνιμουμ ποσοστό απο αυτούς, τουλάχιστον ένα endorsement απο κάθε οργανισμό, και άλλα.

Consortium

Ένα Consortium αποτελείται απο τουλάχιστον δύο οργανισμούς του δικτύου οι οποίοι πρέπει να επικοινωνούν μεταξύ τους και να διεξαγουν συναλλαγές. Αυτοί είναι οι οργανισμοί που εντάσσονται σε κανάλια και παρέχουν peers στο δίκτυο. Η ύπαρξη των Consortium

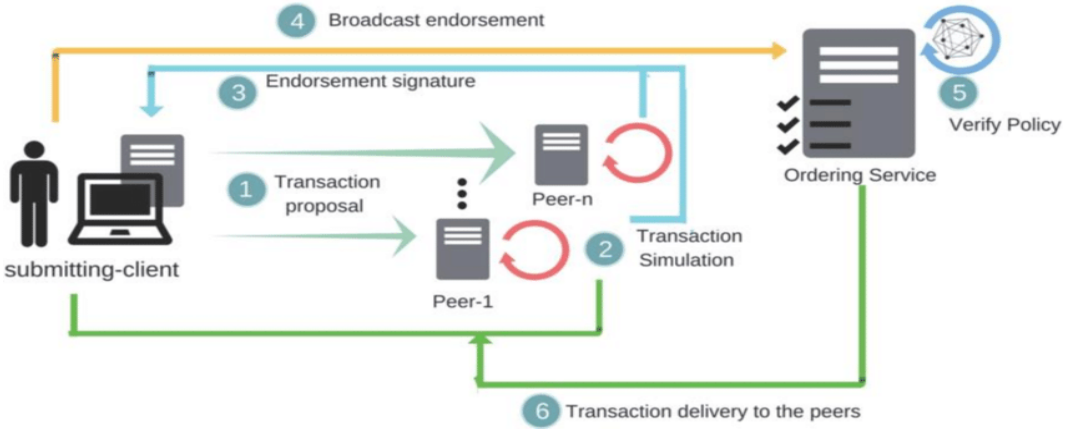
βοηθάει στην δημιουργία πολύπλοκων δίκτυων που διατηρείται μια συνοχή και συνδέονται μεταξύ τους μόνο οι οργανισμοί που έχουν άμεση σχέση ο ένας με τον άλλον.

Ledger

Στο Fabric το Ledger αποτελείται απο δύο διακριτά πράγματα: το blockchain, στο οποίο καταγράφονται όλες οι συναλλαγές, και το world state, που αποτελεί μια βάση δεδομένων που περιέχει τις τωρινές τιμές των καταχωρήσεων που υπάρχουν στην αλυσίδα με μορφή key-value ζευγάρια, επιταχύνοντας έτσι την πρόσβαση στα δεδομένα. Το blockchain δηλαδή περιέχει χρονολογικά όλες τις συναλλαγές που έχουν γίνει, ενώ το world state μας ενημερώνει για την τωρινή τιμή της κάθε καταχώρησης, μετά την εκτέλεση όλων των συναλλαγών της αλυσίδας. Με αυτόν τον τρόπο επιταχύνεται η πρόσβαση στα δεδομένα, ειδικά για την απλή ανάγνωση τιμών.

2.2.3 Transaction Flow

Η ροή μιας συναλλαγής στο Hyperledger Fabric είναι απόρροια των αρχιτεκτονικών επιλογών που έγιναν κατα την σχεδίαση της πλατφόρμας, και ιδιαίτερα της επιλογής του μοντέλου execute-order-validate, στο οποίο αναφερθήκαμε και προηγουμένως. Το transaction flow διαφέρει ανάλογα με τον τύπο συναλλαγής, αν δηλαδή απλώς ζητάμε την ανάγνωση μιας τιμής απο την αλυσίδα (query) ή ζητάμε να καταγραφεί κάτι στην αλυσίδα (invoke). Θεωρώντας ένα σενάριο στο οποίο μία εφαρμογή - πελάτης θέλει να κάνει μία συναλλαγή με το δίκτυο για να καταγραφεί μία τιμή στην αλυσίδα, παρακάτω θα γίνει μία αναλυτική περιγραφή των σταδίων ζωής και ροής της συναλλαγής.



Εικόνα 2.8: *Ο κύκλος ζωής μίας συναλλαγής στο fabric*

- Σε πρώτη φάση, η εφαρμογή - πελάτης δημιουργεί μία πρόταση συναλλαγής (transaction proposal) χρησιμοποιώντας συνήθως το Software Development Kit (SDK) του Fabric. Μία πρόταση συναλλαγής αποτελείται απο το κρυπτογραφικό υλικό του πελάτη (ψηφιακά πιστοποιητικά, υπογραφές κλπ), καθώς και την μέθοδο του chaincode που επιθυμεί να εκτελέσει, μαζί με τις παραμέτρους που πρέπει να δωθούν σαν είσοδο στην μέθοδο του έξυπνου συμβολαίου. Ο πελάτης πρέπει να στείλει την πρόταση συναλλαγής σε τόσους

endorsing peers, όσους απαιτούνται για την ικανοποίηση της πολιτικής endorsement του chaincode με το οποίο θέλει να αλληλεπιδράσει.

- Κάθε endorsing peer που λαμβάνει το transaction proposal επιβεβαιώνει οτι:

  1. είναι σε μία αποδεκτή μορφή (για παράδειγμα, οτι δεν λείπουν απαραίτητα στοιχεία για την περάτωση της συναλλαγής)
  2. δεν έχει κατατεθεί ήδη στο παρελθόν η ίδια συναλλαγή (προσφέροντας ασφάλεια απέναντι σε replay attacks
  3. η υπογραφή του πελάτη είναι έγκυρη
  4. ο πελάτης έχει την δικαιοδοσία να πραγματοποιήσει την ενέργεια που επιθυμεί

  Εάν η συναλλαγή πληροί όλα τα κριτήρια, ο endorser προχωράει στην περάτωσή της. Συγκεκριμένα, εκτελεί μια προσομοίωση της εκτέλεσης της συναλλαγής, εκτελώντας την μέθοδο του chaincode που ζήτησε ο πελάτης, δίνοντας σαν είσοδο τα δεδομένα που περιέχονται στη συναλλαγη. Σε αυτό το σημείο δεν καταγράφεται τίποτα στην αλυσίδα. Σε αυτό το σημείο υλοποιείται το execute κομμάτι της αρχιτεκτονικής execute-order-validate. Το αποτέλεσμα της προσομοίωσης, μαζί με τα αρχικά δεδομένα, δημιουργούν ένα read-write set το οποίο και επιστρέφεται στον πελάτη ως απάντηση πρότασης (proposal response).

- Ο πελάτης, για κάθε απάντηση πρότασης που λαμβάνει απο τους endorsers, επικυρώνει την ψηφιακή υπογραφή του endorser και ελέγχει ότι κάθε απάντηση που λαμβάνει περιέχει το ίδιο read-write set. Όταν συλλέξει αρκετές απαντήσεις για να ικανοποιήσει την πολιτική endorsement, τις συμπεριλαμβάνει σε μία συναλλαγή, και τις στέλνει στο ordering service.

- Σε αυτό το σημείο υλοποιείται το order κομμάτι της αρχιτεκτονικής execute-order-validate. Το ordering service δέχεται συναλλαγές απο όλα τα κανάλια το δικτύου. Όταν το ordering service λάβει μία συναλλαγή, καθορίζει μια χρονολογική σειρά σε σχέση με τις υπόλοιπες συναλλαγές του καναλιού στο οποίο ανήκει. Σε αυτό το σημείο εμπλέκονται αλγόριθμοι συμφωνίας, καθώς συνήθως το ordering service αποτελείται απο πολλούς ordering service nodes, οι οποίοι κρατούν αντίγραφα της σειράς των συναλλαγών (transaction log, με σκοπό να μην υπάρχει μοναδικό σημείο αποτυχίας στο δίκτυο. Όταν καθοριστεί η χρονική σειρά, ο orderer προσθέτει την συναλλαγή στο transaction log. Όταν το μέγεθος του transaction log ξεπεράσει μία καθορισμένη τιμή (Batch Size) ή όταν περάσει συγκεκριμένος χρόνος απο την δημιουργία του προηγούμενου block (batch timeout), ο orderer δημιουργεί ένα block το οποίο και παραδίδει σε όλους τους peers του δικτύου.

- Όταν ένας commiter λάβει ένα block απο το ordering service, για κάθε συναλλαγή ελέγχει:

  1. εαν υπάρχουν αρκετές απαντήσεις πρότασης συναλλαγής ώστε να ικανοποιείται η πολιτική endorsement

2. εάν έχουν υπάρξει αλλαγές στην τωρινή world state, σε σχέση με την έκδοση των μεταβλητών της world state που εμπεριέχονται στο read set. Έτσι, οι συναλλαγές χωρίζονται σε έγκυρες και άκυρες.

Σε αυτό το στάδιο ουσιαστικά υλοποιείται το validate κομμάτι της αρχιτεκτονικής execute-order-validate.

- Οι έγκυρες συναλλαγές καταγράφονται στο ledger. Ο commiter στέλνει απάντηση στον πελάτη σχετικά με την αποδοχή η όχι της συναλλαγής.

Σε περίπτωση που η συναλλαγή δεν αφορούσε την καταγραφή τιμής στην αλυσίδα, αλλα επρόκειτω για απλή ερώτηση τιμής (query), το transaction flow θα ήταν μέχρι το δεύτερο βήμα. Είναι προφανές πως δεν θα υπήρχε λόγος να σταλεί η συναλλαγή στον orderer σε αυτήν την περίπτωση, καθώς δεν υπάρχει λόγος να καταγραφεί στην αλυσίδα.

2.2.4 Ordering Protocols

Το Fabric, με σκοπό το ordering service να μην έχει ένα μοναδικό σημείο αποτυχίας, αναπτύσσει πολλαπλούς orderers στο δίκτυο, έτσι ώστε να επιτευχθεί Crash Fault Tolerance. Οι orderers μιλάνε μεταξύ τους και χρησιμοποιούν πρωτόκολλα συμφωνίας, ώστε να διατηρούν το ίδιο αντίγραφο της λίστας συναλλαγών, με την ίδια χρονική σειρά.

Το fabric προσφέρει 2 υλοποιήσεις replicated transaction log: την υλοποίηση RAFT και την υλοποίηση Kafka. Υπάρχει επίσης και η υλοποίηση Solo, η οποία όμως δεν προφέρει Crash Fault Tolerance στο δίκτυο. Παρακάτω θα γίνει μία συνοπτική παρουσίαση των διαφορετικών υλοποιήσεων.

Solo

Η υλοποίηση Solo αποτελεί μία υλοποίηση rdering Service που αποτελείται απο έναν μοναδικο Ordering Service Node (OSN) και συνεπώς δεν προσφέρει Crash Fault Tolerance. Επιτελεί κανονικά τις διεργασίες που οφείλει να επιτελεί ένα Ordering Service (δημιουργία βλοχκς, διαχείρηση καναλιών κλπ) αλλά δεν επιτρέπει την ύπαρξη πολλαπλών κόμβων που συνεισφέρουν στο ordering των συναλλαγών.

Όταν χρησιμοποιείται solo ordering service ουσιαστικά έχουμε single server system, οπότε δεν υπάρχει πρόβλημα με την επίτευξη consensus σε αυτό το δίκτυο.

RAFT

Ο Raft [20] είναι ένα πρωτόκολλο συμφωνίας που επιτυγχάνει Crash Fault Tolerance. Είναι ένας αλγόριθμος που αναπτύχθηκε με σκοπό να αντικαταστήσει τον PAXOS που είναι μεν πολύ αποδοτικός, αλλα εξαιρετικά πολύπλοκος στην κατανόησή του. Ο RAFT αποτελεί έναν αλγόριθμο που στηρίζεται στο μοντέλο leader-follower.

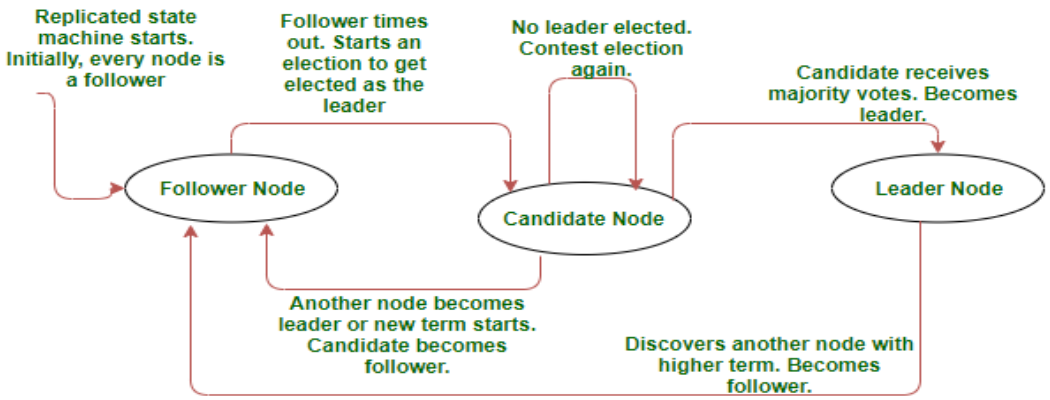
Στον Raft ένας κόμβος μπορεί να βρίσκεται σε τρείς καταστάσεις:

- leader
- candidate

- follower

Ο μόνος κόμβος που αλληλεπιδρά με τον πελάτη είναι ο leader κόμβος. Σε περίπτωση που κάποιος follower δεχτεί αίτηση απο πελάτη, την προωθεί στον leader. Ένας candidate κόμβος ζητάει ψήφους για να γίνει leader.

Στην παρακάτω εικόνα φαίνονται ξεκάθαρα οι διάφορες μεταβάσεις των κόμβων μεταξύ των διαφορετικών καταστάσεων:



Εικόνα 2.9: *Οι μεταβάσεις των κόμβων μεταξύ των διάφορων ρόλων του RAFT*

Για να διατηρήσει ο leader την κατάστασή του, στέλνει ένα σήμα heartbeat στους followers. Κάθε follower έχει σαν παράμετρο ένα timeout, το οποίο είναι ο χρόνος που περιμένει για το heartbeat του leader. Άν περάσει αυτό το χρονικό διάστημα χωρίς να δεχτεί heartbeat, ο κόμβος γίνεται candidate, ψηφίζει τον εαυτό του για νέο leader και στέλνει αιτήσεις στους υπόλοιπους για να κερδίσει την πλειοψηφία του δικτύου και να εκλεγεί ο επόμενος leader.

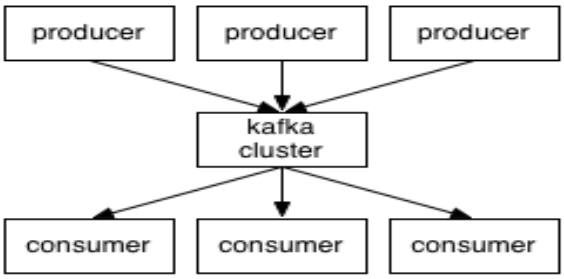
Υπάρχουν πολλά διαφορετικά σενάρια στο πώς μπορεί να εξελιχθεί αυτή η ψηφοφορία, αλλά στο πλαίσιο αυτής της εργασίας μας ενδιαφέρει κυρίως μία αφαιρετική προσέγγιση σχετικά με τον τρόπο λειτουργίας του RAFT, οπότε δεν θα επεκταθούμε παραπάνω σε αυτό το κομμάτι.

Στην συνέχεια, άπαξ και καθοριστεί leader στο δίκτυο, αυτός αναλαμβάνει να δέχεται καινα απαντάει στις αιτήσεις, και οι υπόλοιποι follower κομβοι αντιγράφουν το transaction log απο τον leader. Με αυτόν τον τρόπο, και εφόσον μπορεί να επιτευχθεί πλειοψηφία στους υπάρχοντες κόμβους του δικτύου (να μην υπάρχει ισοπαλία κατα την ψηφοφορία), το δίκτυό μας αποκτά Crash Fault Tolerance.

KAFKA

Ο Kafka [21] είναι ένα σύστημα χειρισμού μηνυμάτων, που χρησιμοποιεί το μοντέλο publish-subscribe. Σε αυτό το μοντέλο, οι καταναλωτές (consumers) εγγράφονται σε ενα θέμα (topic) για να λαμβάνουν μηνύματα, τα οποία δημοσιεύονται απο τους Δημοσιευτές (publishers). Όταν ένα topic γίνεται πολύ μεγάλο, χωρίζεται σε κομμάτια (partitions). Ο Kafka εγγυάται οτι τα μηνύματα μέσα στα partitions έχουν σωστή χρονική σειρά.

Για να επιτευχθεί Crash Fault Tolerance, τα partitions αντιγράφονται μεταξύ πολλών kafka brokers, έτσι ώστε άν ένας broker χαλάσει, το δίκτυο να είναι αποκρίσιμο. Αυτή η διαδικασία υλοποιείται με ένα μοντέλο leader-follower, κατα το οποίο ο leader διαχειρίζεται



Εικόνα 2.10: *Το μοντέλο publish-subscribe του Kafka*

ένα partition και οι followers το αντιγράφουν. Σε περίπτωση που ο leader πάψει να είναι αποκρίσιμος, ένας follower αναλαμβάνει τον ρόλο του.

Για την διαχείρηση των partition και την γνωστοποίηση στους consumers της πληροφορίας σχετικά με το ποιός είναι ο leader στον οποίον πρέπει να μιλήσουν, χρησιμοποιείται η υπηρεσία Zookeeper. Το Zookeeper αποτελεί μία κατανεμημένη βάση key-value που χρησιμεύει για την αποθήκευση μεταδεδομένων. Στην υλοποίηση του kafka, χρησιμεύει για να ενημερώνει το δίκτυο για τις αλλαγές που συμβαίνουν.

Ανεπίσημες υλοποιήσεις

Πέρα απο τις επίσημες υλοποιήσεις του RAFT και του KAFKA, δεδομένου του οτι το fabric είναι ανοιχτού κώδικα και έχει αρθρωτή αρχιτεκτονική, έχουν δημιουργηθεί απο την κοινότητα διάφορες υλοποιήσεις ordering service. Αυτές επικεντρώνονται στην απόδοση χαρακτηριστικών Byzantine Fault Tolerance στο σύστημα, δεδομένου του οτι αυτό είναι που λείπει απο τις επίσημες υλοποιήσεις. Σε αυτό το πλαίσιο, έχουν αναπτυχθεί οι παρακάτω bft υλοποιήσεις:

- PBTF
- BFT-SMART
- HoneyBadger BFT [22]

Κεφάλαιο 3

Το Hyperledger Caliper και η χρήση του σαν Benchmark Engine

Στο κεφάλαιο αυτό θα ασχοληθούμε με το Hyperledger Caliper, ένα εργαλείο που έχει σαν στόχο να τυποποιήσει την διαδικασία benchmarking διαφόρων υλοποιήσεων Blockchain.

3.1 Εισαγωγή

Στον κλάδο των benchmarks και του performance evaluation υπάρχουν διαδικασίες - εργαλεία τα οποία τυποποιούν και προτυποποιούν την διαδικασία ελέγχου της απόδοσης ενός συστήματος. Η τυποποίηση αυτή είναι αναγκαία ώστε να μπορούν να συγκριθούν τα αποτελέσματα της κάθε μελέτης, και να εξαχθούν σημαντικά συμπεράσματα. Σε περίπτωση που η διαδικασία του performance evaluation - benchmarking δεν είναι τυποποιημένη, η υλοποίηση της αποτελεί μια παραπάνω παράμετρο που μπορεί δυνητικά να επηρεάσει τα αποτελέσματα μιας μελέτης, και πρέπει να ληφθεί υπ'όψιν κατα την ερμηνεία των αποτελεσμάτων και την ανάλυσή τους.

Τα δίκτυα blockchain - distributed ledger δεδομένου οτι αποτελούν μια σχετικά νέα τεχνολογία η οποία βρίσκεται ακόμα σε αρχικά στάδια ανάπτυξης, δεν είχαν κάποιο κοινό, τυποποιημένο εργαλείο μέτρησης της απόδοσης. Το κενό αυτό καλείται να λύσει το Hyperledger Caliper.

Το Hyperledger Caliper αποτελεί ένα εργαλείο μέτρησης απόδοσης δικτύων blockchain, το οποίο υποστηρίζει διαφορετικές υλοποιήσεις, και επιτρέπει στον χρήστη να γράψει σενάρια ελέγχου τα οποία μπορούν να προσαρμοστούν και να εκτελεστούν ομοιόμορφα σε διάφορες υλοποιήσεις.

Στο πλαίσιο της εργασίας αυτής χρησιμοποιήσαμε την έκδοση 0.3.2 του caliper, η οποία υποστηρίζει τις εξής υλοποιήσεις blockchain:

- Hyperledger Fabric
- Ethereum
- Hyperledger Sawtooth
- Hyperledger Besu

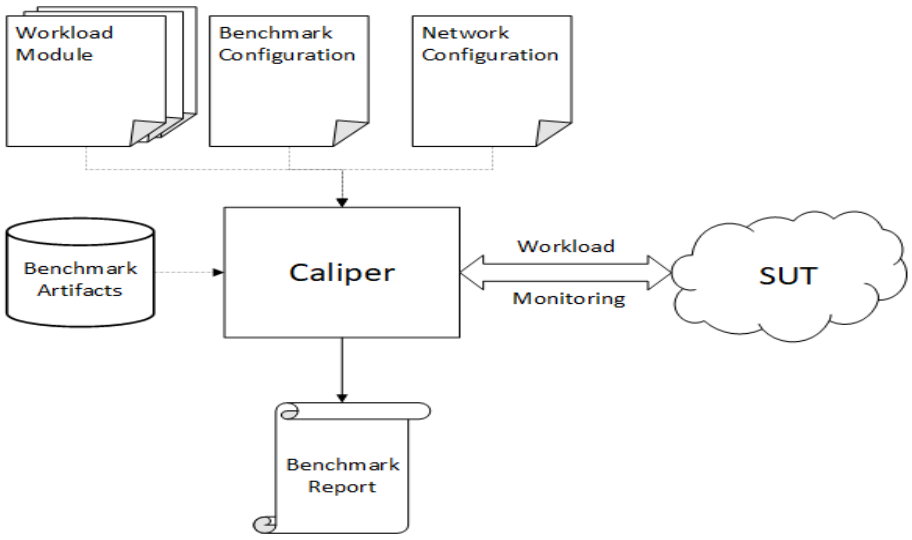
- Hyperledger Burrow
- Hyperledger Iroha
- FISCO BCOS

Το caliper διευκολύνει πάρα πολύ την διαδικασία μέτρησης της απόδοσης ενός blockchain συστήματος, καθώς ο χρήστης δεν χρειάζεται να ασχοληθεί με την διαδικασία αποστολής συναλλαγών, ούτε με την διαδικασία καταγραφής - ελέγχου των μηνυμάτων του δικτύου. Επίσης, προσφέρει πολλές επιλογές παραμετροποίησης σχετικά με τον ρυθμό αποστολής συναλλαγών, καθώς και πολλές μετρικές απόδοσης. Ενδεικτικά, το caliper υποστηρίζει τις εξής μετρικές:

- Ρυθμός διεκπεραίωσης συναλλαγών (Transaction throughput)
- Καθυστέρηση μεταφοράς συναλλαγών (transaction latency)
- Κατανάλωση πόρων συστήματος (επεξεργαστής, μνήμη, δίκτυο κλπ)

3.2 Αρχιτεκτονική

Απο μια αφαιρετική σκοπιά, το caliper είναι ένα εργαλείο που δημιουργεί έναν όγκο εργασίας (workload) σε ένα σύστημα και καταγράφει τις απαντήσεις.



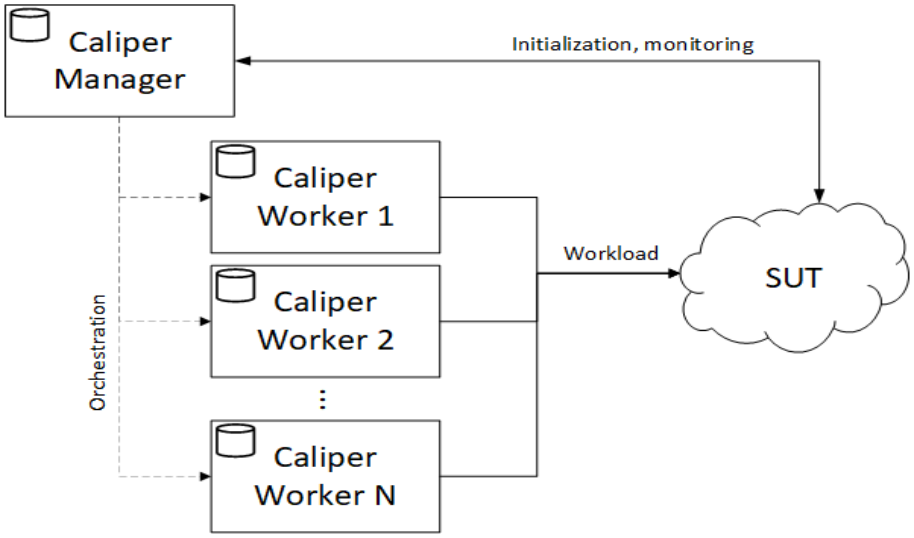
Εικόνα 3.1: *Η γενική λειτουργία του Hyperledger Caliper*

Στην πράξη, πέρα απο την δημιουργία του όγκου εργασίας, το caliper μπορεί, κατα περίσταση, να δημιουργεί το δίκτυο, τα κανάλια καθώς και να εγκαθιστά και να αρχικοποιεί το έξυπνο συμβόλαιο.

Η διαδικασία δημιουργίας όγκου εργασίας είναι μία απαιτητική διαδικασία που δεσμεύει πολλούς υπολογιστικούς πόρους στο σύστημα στο οποίο τρέχει. Αυτό έχει σαν αποτέλεσμα πολλές φορές να επηρεάζονται τα αποτελέσματα του benchmark, καθώς δεν μπορεί το σύστημα που δημιουργεί συναλλαγές να ανταποκριθεί στον ρυθμό με τον οποίο αυτές καταναλώνονται απο το SUT, με αποτέλεσμα να δημιουργείται bottleneck και οι μετρήσεις να μήν είναι ακριβείς.

Για να αντιμετωπίσει αυτό το πρόβλημα, το caliper διαχωρίζει τις εργασίες και χρησιμοποιεί διαφορετικού τύπου διεργασίες, με άλλες ευθύνες η καθε μία.

Το caliper αποτελείται απο δύο ειδών διεργασίες: μία master διεργασία και πολλαπλές worker διεργασίες.



Εικόνα 3.2: *Η αλληλεπίδραση master - worker διεργασιών*

Η master διεργασία είναι υπεύθυνη για την δημιουργία και αρχικοποίηση του SUT, εφόσον ο χρήστης το επιθυμεί, και λειτουργεί σαν ένας συντονιστής του benchmark. Συγκεκριμένα, υποστηρίζει την δημιουργία worker διεργασιών, συντονίζει τους γύρους εκτέλεσης του τέστ, και είναι υπεύθυνη για την συλλογη στοιχείων συναλλαγών και την δημιουργία της αναφοράς στο τέλος του benchmark.

Οι worker διεργασίες είναι υπεύθυνες για την δημιουργία του όγκου εργασίας (workload). Συντονίζονται απο την master διεργασία και αγνοούν η μία την ύπαρξη της άλλης. Αυτό έχει σαν αποτέλεσμα την ευκολότερη κλιμακωσιμότητα του caliper, καθώς υπάρχει η δυνατότητα κατανομής των workers σε διαφορετικά μηχανήματα, με αποτέλεσμα να είναι πιο δύσκολος ο κορεσμός του συστήματος.

3.2.1 Η διεργασία master

Όπως αναφέρθηκε, η διεργασία master είναι υπεύθυνη για τον συντονισμό του benchmark. Το caliper χωρίζει ένα benchmark σε πέντε στάδια, τα οποία εκτελεί η master διεργασία:

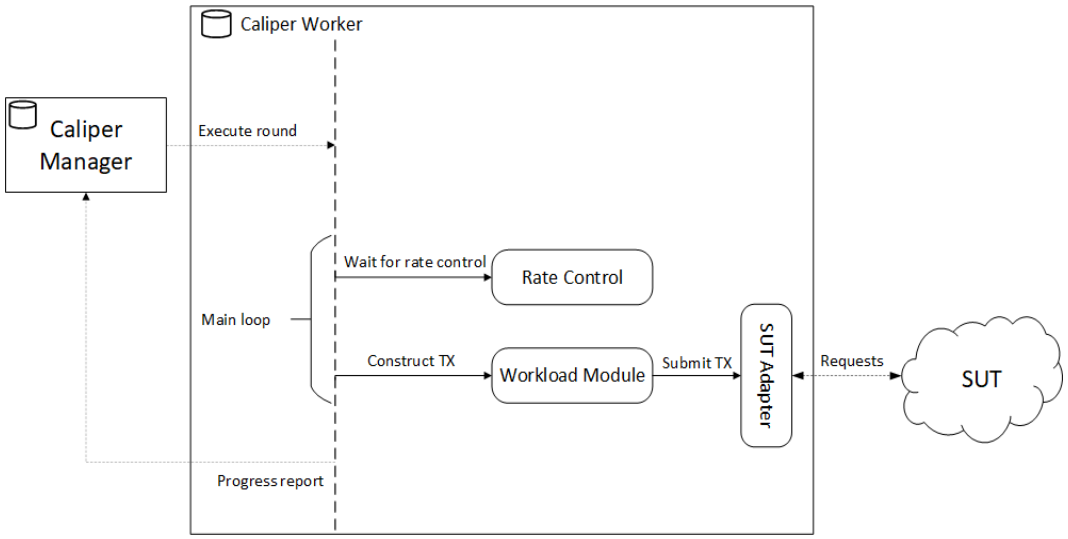
- Στάδιο 1: Εκτέλεση διαδικασίας εκκίνησης. Η διαδικασία εκκίνησης γράφεται με την μορφή script και προσδιορίζεται στο αρχείο παραμέτρων δικτύου (θα αναφερθούμε σε αυτό παρακάτω).
- Στάδιο 2: Αρχικοποίηση του SUT. Σε αυτό το στάδιο το caliper, ανάλογα με την υλοποίηση του SUT στην οποία στοχεύουμε το benchmark, αναλαμβάνει διαδικασίες όπως δημιουργία καναλιών (στην περίπτωση που το SUT είναι Hyperledger Fabric), εγγραφή χρηστών κλπ
- Στάδιο 3: Εγκατάσταση και αρχικοποίηση έξυπνων συμβολαίων.

- Στάδιο 4: Εκτέλεση του τέστ. Εδώ ξεκινάει η διαδικασία εκτέλεσης ενός benchmark. Εδώ, η master διεργασία συντονίζει τους workers δίνοντας κατάλληλες εντολές, ώστε να δημιουργηθεί το ζητούμενο workload. Μετά το πέρας του benchmark, η master διεργασία συγκεντρώνει τα στοιχεία και σχηματίζει την αναφορά.

- Στάδιο 5: Εκτέλεση διαδικασίας τερματισμού. Αντίστοιχα με την διαδικασία εκκίνησης, εκτελείται ένα script που αναλαμβάνει συνήθως να διαγράψει το δίκτυο που έχει δημιουργηθεί.

Κάθε ένα απο αυτά τα στάδια είναι προαιρετικά. Στο πλαίσο της εργασίας αυτής, η δημιουργία των δικτύων γίνεται με δικές μας διαδικασίες, οπότε το μόνο στάδιο του caliper που χρησιμοποιούμε είναι το στάδιο 4.

3.2.2 Οι διεργασίες worker



Εικόνα 3.3: *Το εσωτερικό ενός worker*

Οι διεργασίες worker είναι υπεύθυνες για την δημιουργία του όγκου εργασίας, σε συνεργασία με την διεργασία master. Ο τρόπος λειτουργίας τους είναι ο εξής:

- Η διεργασία worker ξυπνάει όταν λάβει μήνυμα απο την master να εκτελέσει τον επόμενο γύρο

- Ενεργοποιείται ο rate controller, που ουσιαστικά αποτελεί μία λούπα χρονοκαθυστέρησης, που ανα τακτά χρονικά διαστήματα απελευθερώνει μία συναλλαγή.

- Όταν επιτραπεί μία συναλλαγή απο τον rate controller, η διαδικασία worker παραδίδει τον έλεγχο στο workload module, ο ρόλος του οποίου αναλυτικά περιγράφεται σε επόμενη ενότητα. Επιγραμματικά, το workload module δημιουργεί την συναλλαγή και την στέλνει στο SUT.

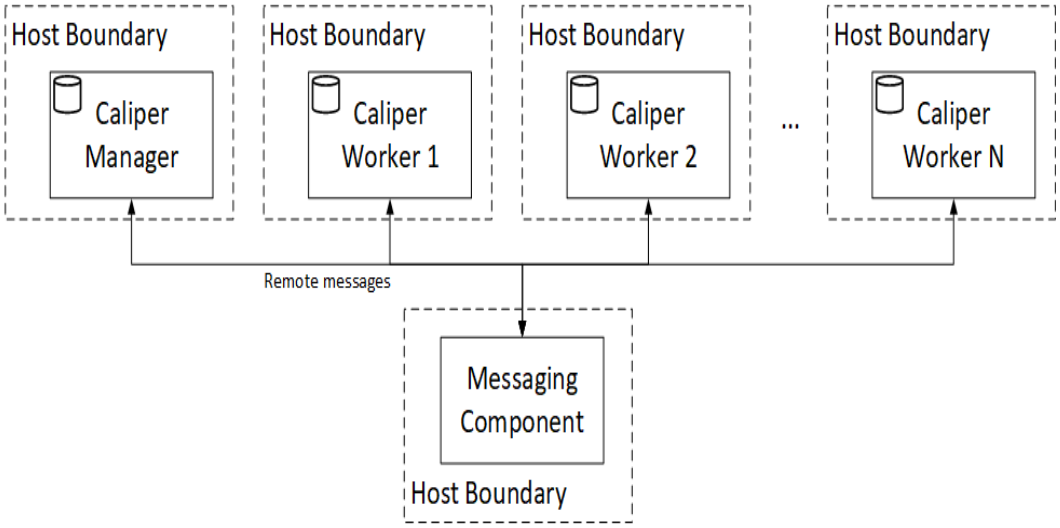
- Τα στοιχεία για κάθε συναλλαγή (καθυστέρηση, κατάσταση επιτυχίας κλπ) στέλνονται στην διεργασία master για να συγκεντρωθούν και να επεξεργαστούν.

- Όταν τελειώσει ο γύρος, ο worker ενημερώνει τον master για την ολοκλήρωση της εργασίας του.

3.2.3 Μοντέλα κατανομής διεργασιών εργατών (workers)

Το caliper υποστηρίζει διαφορετικά μοντέλα κατανομής των workers, καθώς και διαφορετικά μοντέλα διαδιεργασιακής επικοινωνίας.

Οι workers μπορούν να δημιουργηθούν είτε αυτόματα απο την master διεργασία, είτε χειροκίνητα. Στην περίπτωση που δημιουργούνται αυτόματα, περιορίζονται σε έναν host μόνο, και συγκεκριμένα στον ίδιο host που τρέχει η master διεργασία. Σε περίπτωση που δημιουργηθούν χειροκίνητα, μπορούν να κατανεμηθούν σε οποιοδήποτε αριθμό μηχανημάτων επιθυμούμε.



Εικόνα 3.4: *Η πλήρως αποκεντρωμένη λειτουργία του caliper*

Για την διαδιεργασιακη επικοινωνία, διακρίνουμε 2 περιπτώσεις. Στην πρώτη περίπτωση, άν όλοι οι workers δημιουργηθούν απο την master διεργασία, χρησιμοποιείται το πρωτόκολλο IPC - Interprocess Communication, χρησιμοποιώντας την υλοποίηση του Node.JS για επικοινωνία μεταξύ parent - child διεργασίες. Στην δεύτερη περίπτωση, αντί για επικοινωνία πατέρα - παιδιού υποστηρίζεται η χρήση κάποιου messanging component. Στην υλοποίησή μας χρησιμοποιήσαμε το πρωτόκολλο mqtt.

Συνοψίζοντας, καταλήγουμε στις παρακάτω μορφές κατανομής εργατών:

- Εργάτες δημιουργημένοι αυτόματα απο την master διεργασία στον ίδιο host με αυτήν, που χρησιμοποιούν IPC για την επικοινωνία με την master
- Εργάτες δημιουργημένοι αυτόματα απο την master διεργασία στον ίδιο host με αυτήν, που χρησιμοποιούν κάποιο σύστημα ανταλλαγής μηνυμάτων
- Εργάτες δημιουργημένοι απο τον χρήστη, κατανεμημένοι σε πολλούς hosts, που χρησιμοποιούν κάποιο σύστημα ανταλλαγής μηνυμάτων

Η τρίτη υλοποίηση είναι πολύ σημαντική, καθώς μας επιτρέπει να κάνουμε scale το benchmark όσο θέλουμε. Αυτό έχει σαν αποτέλεσμα να μπορούμε να προσεγγίσουμε σενάρια

πραγματικής λειτουργίας κατα το benchmark, δημιουργώντας την κατανομή που θέλουμε για τους clients, και προσαρμόζοντας το benchmark στις πραγματικές απαιτήσεις του use case το οποίο εξετάζουμε.

3.2.4 Παραμετροποίηση του συστήματος

Για να εξασφαλιστεί η αφαιρετικότητα και να μπορεί το εργαλείο να προσαρμόζεται σε πολλές διαφορετικές καταστάσεις, οι παράμετροι του σεναρίου ελέγχου (benchmark scenario) και του συστήματος υπο εξέταση (System Under Test - SUT) καταγράφονται σε αρχεία παραμετροποίησης με την μορφή YAML κωδικοποίησης και δίνονται σαν είσοδος στο σύστημα.

Παραμετροι benchmark

Ένα benchmark στο caliper αποτελείται απο διαφορετικούς γύρους. Κάθε γύρος μπορεί να οριστεί είτε με συγκεκριμένη χρονική διάρκεια, είτε για συγκεκριμένο αριθμό συναλλαγών. Σε κάθε γύρο ορίζεται η στρατηγική αποστολής συναλλαγών, η διάρκεια, και ο κώδικας που θα τρέξει για την ενεργοποίηση της μεθόδου του έξυπνου συμβολαίου που επιθυμούμε να χρησιμοποιήσουμε.

Το αρχείο παραμετρων benchmark (benchmark configuration file) περιέχει πληροφορίες που περιγράφουν το πώς πρέπει να εκτελεστεί ένα τέστ. Συγκεκριμένα, είναι το σημείο στο οποίο ορίζεται απο τί γύρους θα αποτελείται ένα benchmark και εδώ καθορίζονται οι ρυθμίσεις για κάθε γύρο. Επίσης, εδώ καθορίζεται το πόσες διεργασίες - εργάτες θα χρησιμοποιηθούν για την δημιουργία των συναλλαγών.

Στο αρχείο παραμετρων benchmark επίσης ορίζονται οι μετρικές του συστήματος που θέλουμε να παρακολουθούμε, όπως η χρησιμοποίηση του επεξεργαστή και άλλα.

Το αρχείο παραμετρων benchmark περιέχει όλη την λογική του τεστ. Ουσιαστικά αποτελεί τον ενορχηστρωτή του τέστ, και είναι σε μεγάλο βαθμό ανεξάρτητο του SUT, με αποτέλεσμα να μπορεί κανείς να χρησιμοποιήσει το ίδιο αρχείο βενχημαρκ ονφιγυρατιον φιλε σε διαφορετικές πλατφόρμες, με πολύ μικρές αλλαγές.

Τέλος, το αρχείο αυτό περιέχει μεταδεδομένα σχετικά με το benchmark που θα πραγματοποιηθεί.

Παραμετροι δικτύου

Σε αυτό το αρχείο υπάρχει όλη η πληροφορία σχετικά με τους κόμβους του δικτύου υπο εξέταση (SUT). Αυτό έχει σαν αποτέλεσμα να έχει διαφορετική μορφή για κάθε είδος δικτύου που εξετάζουμε.

Ανεξαρτήτως μορφής, είναι ένα αρχείο που περιγράφει την τοπολογία του SUT, τις διευθύνσεις και τα πιστοποιητικά για όλους τους κόμβους που το αποτελούν και πρέπει να αλληλεπιδρούν με τον client (peers, orderers, certificate authorities), τις ταυτότητες των χρηστών που μπορούν να δημιουργήσουν συναλλαγές (έτσι ώστε να μπορέσει το caliper να τις χρησιμοποιήσει για να δημιουργήσει συναλλαγές), καθώς και πληροφορίες για τα κανάλια που τρέχουν στο δίκτυο, και τα έξυπνα συμβόλαια που έχουν εγκατασταθεί.

Workload Modules

Δεδομένου του οτι το caliper είναι γενικής χρήσης εργαλείο benchmarking, δεν περιλαμβάνει κώδικα για την υλοποίηση του benchmark, παρά μια διεπαφή (interface) για να γράψει ο χρήστης τα δημιουργήσει μόνος του. Τα workload modules περιέχουν κώδικα γραμμένο σε javascript (και συγκεκριμένα τρέχουν στο runtime Node.JS) ο οποίος κάνει export τρείς συναρτήσεις: την init, την run και την end.

Η init τρέχει κατα την αρχικοποίηση του γύρου, και χρησιμεύει για την αρχικοποίηση μεταβλητών κλπ που θα χρησιμοποιηθούν έπειτα για διάφορους υπολογισμούς.

Η run τρέχει κάθε φορά που ο rate controller απελευθερώνει μία συναλλαγή. Εδώ γίνεται η χρήση του Software Development Kit (SDK) του Fabric για να επικοινωνήσει το caliper με το σύστημα. Δύο μέθοδοι είναι σημαντικές, η invokeSmartContract που χρησιμεύει για να δημιουργήσει ένα transaction στο δίκτυο, και η querySmartContract, που χρησιμεύει για να δημιουργήσει ένα query για μία τιμή.

Η end εκτελείται όταν ολοκληρωθεί ο γύρος, και είναι χρήσιμη για τον υπολογισμό στατιστικών κλπ.

Λοιπές παραμετροποιήσεις

Πέρα απο τα προαναφερθέντα αρχεία, άλλα αρχεία που δίδονται σαν είσοδος στο caliper για την πραγματοποίηση ενός τέστ μπορεί να είναι τα εξής:

- Πακέτα τρίτων που χρησιμοποιούνται απο τα workload modules
- Κρυπτογραφικά πιστοποιητικά που είναι απαραίτητα για την αλληλεπίδραση με το σύστημα
- Ρυθμίσεις εκτέλεσης
- Κώδικας του εξυπνου συμβολαίου, σε περίπτωση που θέλουμε το caliper να το εγκαταστήσει για εμάς

3.3 Rate controllers

Ο ρυθμός με τον οποίο στέλνουμε συναλλαγές στο σύστημα είναι πολύ βασική παράμετρος που καθορίζει την σωστή υλοποίηση ενός benchmark. Σε κάποιες περιπτώσεις θέλουμε να δούμε πώς αντιδράει το σύστημα σε μία σταθερά υψηλή ροή συναλλαγών, σε άλλες απαιτούμε ανατροφοδότηση και προσαρμογή του ρυθμού. Κάθε συμπέρασμα που θέλουμε να βγάλουμε και κάθε κατάσταση στην οποία θέλουμε να δοκιμάσουμε το σύστημά μας απαιτεί ξεχωριστή αντιμετώπιση ως προς τον ρυθμό αποστολής συναλλαγών.

Το caliper τυποποιεί αυτήν την παράμετρο με τους rate controllers. Απο αρχιτεκτονική σκοπιά, οι rate controllers είναι δομές οι οποίες απελευθερώνουν ανα τακτά χρονικά διαστήματα την εκτέλεση μίας συναλλαγής. Οι rate controllers καθορίζονται στο αρχείο παραμέτρων benchmark και είναι διαφορετικοί για κάθε γύρο του benchmark. Πέρα απο τις τυποποιημένες

και υποστηριζόμενες υλοποιήσεις, ο χρήστης μπορεί να υλοποιήσει τον δικό του rate controller, υλοποιώντας την διεπαφή που απαιτεί το σύστημα. Παρακάτω θα παραθέσουμε κάποιους βασικούς rate controllers που το caliper υποστηρίζει out-of-the-box:

fixed-rate controller

Αποτελεί τον πιό βασικό controller. Στέλνει συναλλαγές με ένα σταθερό ρυθμό, ο οποίος ορίζεται ως transactions per second (TPS)

fixed-backlog controller

Ο συγκεκριμένος controller είναι πολύ σημαντικός. Σε αντίθεση με τον fixed-rate, έχει σκοπό να διατηρεί μία συνεχόμενη ροή συναλλαγών προς το σύστημα. Ανατροφοδοτείται δυναμικά απο τις απαντήσεις του SUT και προσαρμόζει τον ρυθμό με τον οποίο στέλνει συναλλαγές, έτσι ώστε πάντα να περιμένει απάντηση απο N συναλλαγές. Αυτό έχει ως αποτέλεσμα να παίρνουμε το μέγιστο δυνατό throughput απο ένα σύστημα, με βάση τον φόρτο που ορίσαμε.

Ο αριθμός συναλλαγών που ανα πάσα στιγμή πρέπει να μην έχουν ολοκληρωθεί (δηλαδη να έχουν σταλεί, αλλα να μήν έχει επιβεβαιωθεί η επικύρωσή τους) ορίζεται απο την παράμετρο unfinished per client.

Στο πλαίσο αυτής της εργασίας χρησιμοποιείται ευρέως ο fixed-backlog controller.

fixed-feedback-rate controller

Αποτελεί μία παραλλαγή του fixed-rate, με την διαφορά οτι άν οι συναλλαγές για τις οποίες δέν έχει λάβει απάντηση ξεπεράσουν μία καθορισμένη τιμή, αυτός θα σταματήσει να στέλνει καινούριες.

Σε αντίθεση με τον fixed-backlog, δεν προσαρμόζει τον ρυθμό με τον οποίο στέλνει συναλλαγές, απλώς σταματάει για ένα χρονικό διάστημα να στέλνει, ώστε να μήν υπερφορτώσει το σύστημα.

linear-rate controller

Ο linear-rate δέχεται δύο παραμέτρους, την startingTps και την finishingTps. Σκοπός του είναι να αυξομοιώνει τον ρυθμό αποστολής συναλλαγών μεταξύ αυτών των δύο τιμών, ώστε να βρεθούν οι τιμές που επηρεάζουν την απόδοση του συστήματος.

composite-rate controller

Ο composite-rate επιτρέπει τον συνδιασμό ήδη υλοποιημένων controllers σε έναν γύρο. Συγκεκριμένα, επιτρέπει στον χρήστη να ορίσει διαφορετικούς controllers οι οποίοι θα εναλλάσονται κατα την διάρκεια ενός γύρου. Οι διαφορετικοί rate controlelrs εντός του composite-rate εναλάσσονται σύμφωνα με την αναλογία που ορίζει ο χρήστης, η οποία δίνεται με την μορφή βαρών (weights). Ο composite rate είναι αρκετά χρήσιμος, καθώς μπορεί κανείς με αυτόν τον τρόπο να μοντελοποιήσει περίπλοκα σενάρια, όπως κατανομές poisson, που δέν είναι ήδη υλοποιημένα.

zero-rate controller

Ο συγκεκριμένος controller απλώς σταματάει για την διάρκεια του γύρου την δημιουργία συναλλαγών. Μπορεί να συνδιαστεί με τον composite-rate για να δημιουργήσει κατανομές συναλλαγών.

Μέρος II

Πρακτικό Μέρος

Κεφάλαιο 4

Παρουσίαση του Πειραματικού Περιβάλλοντος

Στο κεφάλαιο αυτό θα παρουσιαστούν στοιχεία που χρησιμοποιήσαμε για την δημιουργία των δικτύων στα οποία στην συνέχεια κάναμε benchmarks.

4.1 Εισαγωγή

Το Hyperledger Fabric είναι μια πλατφόρμα της οποίας η απόδοση επηρεάζεται απο πάρα πολλές παραμέτρους. Για να μπορέσουμε να βγάλουμε συμπεράσματα για κάποιες εξ αυτών, πρέπει σε πρώτη φάση να κάνουμε κάποιες παραδοχές που θα είναι κοινές για τα πειράματα που θα πραγματοποιήσουμε. Αυτές οι παραδοχές αφορούν κυρίως παραμέτρους των οποίων την απόδοση δεν θα ερευνήσουμε πειραματικά, όπως πχ το smart contract (chaincode) που θα χρησιμοποιήσουμε.

Πρίν περάσουμε λοιπόν στην περιγραφή - εκτέλεση των πειραμάτων, πρέπει να αναφερθούμε σε σχεδιαστικές επιλογές που κάναμε και στα χαρακτηριστικά των δικτύων που δημιουργούμε για να μετρήσουμε την απόδοσή τους.

4.2 Υπολογιστικοί πόροι - τοπολογία δικτύου

Για την πραγματοποίηση των πειραμάτων μας διατέθηκαν υπολογιστικοί πόροι απο την πλατφόρμα okeanos-knossos. Χρησιμοποιήσαμε αυτούς τους πόρους για να δημιουργήσουμε εικονικές μηχανές (VMs) τις οποίες διασυνδέσαμε μεταξύ τους σε ένα τοπικό δίκτυο. Για πρόσβαση στο διαδίκτυο, χρησιμοποιήσαμε μία εικονική μηχανή η οποία λειτουργεί ως NAT Router, έχει public IP και μέσω αυτής αποκτούν όλες πρόσβαση στο διαδίκτυο.

Η κατανομή των πόρων ανα εικονική μηχανή έγινε ανάλογα με το τί υπηρεσία τρέχει η κάθε μηχανή. Στον παρακάτω πίνακα παρουσιάζουμε τα χαρακτηριστικά μηχανής, ανάλογα με την υπηρεσία του fabric που αυτή εξυπηρετει.

Δοκιμάζοντας διαφορετικές τοπολογίες, καταλήξαμε οτι η απόδοση είναι καλύτερη όταν κάθε μηχανή τρέχει το πολύ ένα instance του fabric, είτε αυτό είναι peer είτε orderer. Ο λόγος είναι το ότι οι peers σαν διεργασίες είναι πολύ απαιτητικές απο άποψη επεξεργαστικής ισχύος, ενώ οι orderers ανταλάσσουν πολλά μηνύματα, με αποτέλεσμα να έχουν μεγάλη χρήση bandwidth. Όταν αλληλοπλέκονται στο ίδιο μηχάνημα, αυτό έχει σαν αποτέλεσμα να καταναλώνει μεγάλο μέρος του εύρους ζώνης δικτύου ο orderer, με αποτέλεσμα ο peer να

Service	CPUs	RAM
Fabric Peer	8	16GB
Fabric Orderer	8	16GB
Caliper Master	4	8GB
Caliper Workers	16	16GB
NAT Router	4	8GB

Πίνακας 4.1: *Κατανομή πόρων ανα υπηρεσία στις εικονικές μηχανές*

μην δεχεται γρήγορα τις συναλλαγές τις οποίες πρέπει να εξυπηρετήσει, και να δημιουργείται περεταίρω καθυστέρηση στο δίκτυο. Τ

Όσον αφορά τους workers του caliper, υπάρχουν κάποιοι περιορισμοί. Αρχικά, είναι προγράμματα γραμμένα σε Node.JS το οποίο είναι single-threaded. Αυτό σημαίνει οτι δεν μπορεί ένας worker να χρησιμοποιήσει παραπάνω απο έναν επεξεργαστή. Άν ο αριθμός των workers σε ένα σύστημα ξεπεράσει τον αριθμό των επεξεργαστών, αυτοί ανταγωνίζονται μεταξύ τους για τους πόρους, και οδηγούμαστε σε αχρείαστα context switches τα οποία ρίχνουν την απόδοση του συστήματος. Επίσης, δεδομένου του οτι σε ένα λειτουργικό σύστημα πολλές διεργασίες εκτελούνται στο παρασκήνιο, επιλέξαμε να αναπτύσσουμε αρκετά λιγότερους workers απο τον αριθμό πυρήνων σε έναν host. Συγκεκριμένα, στα πειράματά μας χρησιμοποιούμε 4 workers ανα host, όπου όπως αναφέρθηκε, ο κάθε host διαθέτει 16 πυρήνες.

Συνοψίζοντας, εκτός και αν αναφέρεται ρητώς διαφορετικά, στα πειράματά μας αναπτύσσουμε τον κάθε peer, orderer σε διαφορετικό host, και 4 workers ανα host.

4.3 Container Orchestration

Το docker είναι μία πλατφόρμα που επιτρέπει την εκτέλεση εφαρμογών σε μικρά και ελαφριά απομονωμένα περιβάλλοντα εκτέλεσης, τα containers. Οι εφαρμογές που τρέχουν σε containers στον ίδιο host μοιράζονται τον ίδιο kernel, αλλα λειτουργούν απομονωμένα η μία απο την άλλη. Τα containers θυμίζουν την λογική των εικονικών μηχανών, αλλά σε αντίθεση με μία εικονική μηχανή, δεν μπορούνε να 'ζήσουν' απο μόνα τους, γιατί χρησιμοποιούν τον kernel του host μηχανήματος.

Κάθε node του fabric τρέχει μέσα σε ένα docker container. Όταν όλοι οι κόμβοι του δικτύου τρέχουν στον ίδιο host μπορούν να μιλήσουν μεταξύ τους εύκολα, καθώς υπάρχει άμεση διασύνδεση μεταξύ τους. Όταν όμως αναπτύσσουμε το fabric σε κατανεμημένο περιβάλλον, που είναι και το πραγματικό use case, χρειαζόμαστε έναν τρόπο αυτοί οι containers να νομίζουν οτι βρίσκονται στο ίδιο μηχάνημα, οτι είναι κομμάτι ενός cluster. Η διαδικασία αυτή ονομάζεται container orchestration.

Για το container orchestration δύο είναι οι βασικές επιλογές: το Docker Swarm και το Kubernetes. Στην υλοποίησή μας επιλέξαμε να κάνουμε χρήση του Docker Swarm.

Το Docker Swarm αποτελεί ένα Container Orchestration εργαλείο που κάνει εύκολη την διαχείρηση containers που αναπτύσσονται σε πολλές μηχανές. Σε συνδιασμο με το εργαλείο docker-compose, απλοποιεί σε πολύ μεγάλο βαθμό την χρονοβόρα και επίπονη διαδικασία της δημιουργίας ενός cluster σε πολλούς hosts.

Το Docker σαν πλατφόρμα έχει εξειδικευμένες επιλογές δικτύωσης. Στο πλαίσιο της εργασίας αυτής χρησιμοποιήσαμε overlay docker networks. Τα overlay δίκτυα ¨κάθονται' πάνω απο τα διαφορετικά φυσικά δίκτυα των hosts, επιτρέποντας στα containers που συνδέονται σε αυτά να επικοινωνούν άμεσα και με ασφάλεια μεταξύ τους.

Στο πλαίσιο αυτής της εργασίας, δημιουργούμε ένα swarm στο οποίο εντάσσουμε όλες τις εικονικές μηχανές. Στη συνέχεια, δημιουργούμε ένα overlay δίκτυο. Κάθε container που κάνουμε deploy θα συνδέεται σε αυτό το overlay δίκτυο.

4.4 Chaincode

Η απόδοση του chaincode δεν είναι μία παράμετρος που θα μας απασχολήσει στο πλαίσιο αυτής της μελέτης. Υπο αυτό το πρίσμα, μπορούμε να χρησιμοποιήσουμε οποιοδήποτε chaincode θέλουμε.

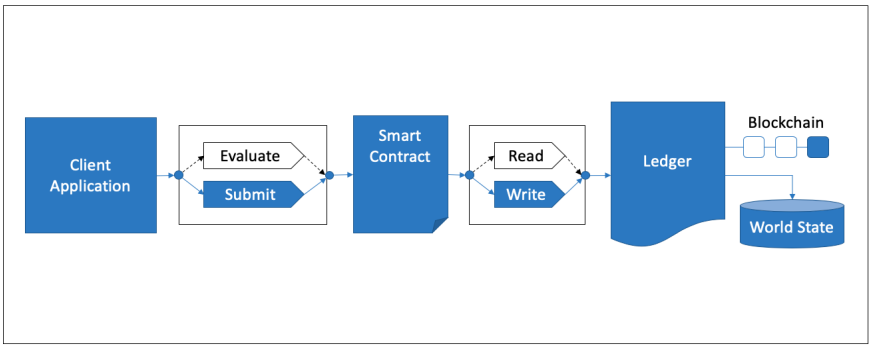
Το chaincode στην πράξη εμπεριέχει το business logic της κατανεμημένης εφαρμογής που υλοποιεί το σύστημά μας. Υπάρχουν διάφορα παραδείγματα chaincode στο επίσημο αποθετήριο κώδικα του hyperledger fabric. Εμείς διαλέξαμε το fixed-asset chaincode, το οποίο υποστηρίζει δυναμική αλλαγή του μεγέθους συναλλαγής, το οποίο αποτελεί μία απο τις παραμέτρους των οποίων την απόδοση ερευνάμε. Το fixed-asset έχει χρησιμοποιηθεί και για τα επίσημα performance evaluations του Hyperledger Fabric. Η επιλογή αυτή δεν είναι δεσμευτική και τα αποτελέσματά μας είναι ανεξάρτητα αυτής.

Το fixed-asset εμπεριέχει μεθόδους που μοντελοποιούν διάφορα σενάρια χρήσης. Στο πλαίσιο αυτής της μελέτης, θα χρησιμοποιήσουμε κυρίως τις παρακάτω:

- createAsset: μία μέθοδος που δημιουργεί μία καταχώρηση στο world state για ένα ασσετ. Η έννοια του asset είναι αφαιρετική, θα μπορούσε να είναι οτιδήποτε. Η συναλλαγή καταγράφεται στο blockchain. Στην γλώσσα του fabric, αποτελεί invoke του smart contract.

- empty-contract: μια μέθοδος αντίστοιχη με την createAsset, με την διαφορά οτι στο blockchain δεν καταγράφεται κάποιο νέο asset, αλλά καταγράφεται η επίκλησή της.

- getAsset: μία μέθοδος με την οποία ζητάμε το περιεχόμενο ενός asset απο το world state. Η επίκληση αυτής της μεθόδου δεν καταγράφει κάτι στο blockchain



Εικόνα 4.1: *Η διαδρομή μιας createAsset και μιας emptyContract συναλλαγής*

4.5 Στρατηγική πειραμάτων

Τα πειράματά μας εκτελούνται σε ένα ςλουδ περιβάλλον εικονικοποίησης. Στο περιβάλλον αυτό τρέχουν παράλληλα πολλά διαφορετικά πράγματα. Παρατηρήσαμε οτι αυτό είχε συνέπειες στα πειραματικά μας αποτελέσματα. Συγκεκριμένα, το ίδιο πείραμα άν εκτελεστεί σε διαφορετικές χρονικές στιγμές, με ακριβώς τις ίδιες παραμέτρους εισόδου, μπορεί να βγάλει διαφορετικά αποτελέσματα, δημιουργώντας ένα στατιστικό λάθος.

Ένας τρόπος να εξαλειφθεί αυτό θα ήταν να στήναμε το δίκτυο σε φυσικά μηχανήματα διασυνδεδεμένα μεταξύ τους αντί για εικονικές μηχανές. Δεδομένου του οτι πόροι για ένα τέτοιο εγχείρημα είναι δύσκολο να βρεθούν, επιλέξαμε την εξής στρατηγική για την (όσο το δυνατόν) εξάλειψη του στατιστικού λάθους:

Εκτελούμε κάθε πείραμα πολλές φορές (περίπου 10), συλλέγουμε τα αποτελέσματα όλων των πειραμάτων, και παίρνουμε το median αυτών των τιμών.

Κεφάλαιο 5

Πειραματικά Αποτελεσματα

Στο κεφάλαιο αυτό θα περιγράψουμε σενάρια πειραμάτων, θα παραθέσουμε τα αποτελέσματα και θα ακολουθήσει εξαγωγή συμπερασμάτων ερμηνεύοντας τα αποτελέσματα αυτά.

5.1 Εισαγωγή

Σε κάθε ένα απο τα παρακάτω πειράματα, εκτός και αν αναφέρεται ρητώς διαφορετικά, η δημιουργία workload γίνεται απο 12 caliper workers που είναι κατανεμημένοι σε 3 εικονικές μηχανές (4 workers στην κάθε μία). Αν δεν αναφέρεται κάτι συγκεκριμένο, ο rate controller που χρησιμοποιείται είναι ο fixed-backlog με backlog: 50 transactions. Τέλος, το endorsement policy που χρησιμοποιείται είναι το πιο απλό, 1-of-any, που δέχεται μία συναλλαγή αν έχει ένα endorsement απο οποιονδήποτε peer. Η έκδοση του Fabric που χρησιμοποιήσαμε είναι η 1.4.8, και η έκδοση του caliper η 0.3.2.

Σε κάθε πείραμα θα προσδιορίζονται οι παραδοχές που έγιναν και θα αιτιολογούνται. Στη συνέχεια, θα περιγράφεται η πειραματική διάταξη (πέρα απο αυτά που ήδη έχουν αναφερθεί) και οι διάφορες παράμετροι. Στην συνέχεια, θα γίνει παράθεση των αποτελεσμάτων. Τέλος, θα γίνει ερμηνεία αυτών, εξήγηση και εξαγωγή συμπερασμάτων.

Οι βασικές παράμετροι αξιολόγησης του συστήματος είναι οι εξής:

- throughput: συναλλαγές που εξυπηρετούνται στην μονάδα του χρόνου
- average latency: μέσος χρόνος καθυστέρησης για την ολοκλήρωση μίας συναλλαγής

Fabric Version	1.4.8
Caliper Version	0.3.2
Caliper Worker Distribution	Remote
Number of workers	12
Endorsement Policy	1-of-any
Rate Controller	fixed-backlog
Unfinished_per_client	50
Test duration	5min

Πίνακας 5.1: *Σταθερές παράμετροι στα πειράματα*

5.2 Μελέτη της επίδρασης του ρυθμού δημιουργίας block στην απόδοση του συστήματος

Σε αυτό το πείραμα θα μελετήσουμε την επίδραση του μεγέθους του block στο throughput και στο latency του συστήματος.

Το ordering service του Hyperledger Fabric χρησιμοποιεί δύο παραμέτρους για να αποφασίσει πότε ένα σύνολο συναλλαγών θα αποτελέσει ένα block και θα σταλεί στους peers για επιβεβαίωση: το απόλυτο μέγεθος του block σε συναλλαγές και το batch timeout, το μέγιστο χρονικό διάστημα που θα παραμείνει μία συναλλαγή στο transaction log πρίν μπεί σε block. Όταν κάποια απο τις δύο συνθήκες εκπληρωθεί, δημιουργείται ένα block.

Στο πλαίσιο αυτής της μελέτης θα προσπαθήσουμε να μελετήσουμε την επίδραση της κάθε παραμέτρου ξεχωριστά, και στην συνέχεια να βρούμε το συνδιασμό των παραμέτρων που οδηγούν σε μέγιστο throughput το σύστημα.

Αξίζει να αναφερθούμε στο γεγονός ότι η πλήρης αποσύμπλεξη αυτών των παραμέτρων δεν είναι εύκολη, καθώς χρειάζονται και οι δύο για την εύρυθμη λειτουργία του συστήματος. Αυτό γίνεται εύκολα αντιληπτό, γιατί αν βάλουμε μία τεράστια τιμή στο Batch Timeout και δεν ολοκληρωθεί το προσδιορισμένο block size, οι συναλλαγές που θα έχουν ξεμείνει στο transaction log του orderer δεν θα εξυπηρετηθούν μέχρι να έρθουν άλλες και να συμπληρωθεί το μέγεθος block που ορίσαμε. Αυτό θα μειώσει την απόδοση του συστήματος, και θα οδηγήσει σε ακύρωση μελλοντικών συναλλαγών οι οποίες θα θέλουν να μεταβάλλουν τιμές του world state οι οποίες όμως θα πρέπει να είχαν ήδη αλλάξει απο συναλλαγές που ποτέ δεν ολοκληρώθηκαν στην ώρα τους.

5.2.1 Η επίδραση του μεγέθους block

Για να μελετήσουμε την επίδραση του block size στήνουμε ένα δίκτυο Hyperledger Fabric με τα εξής χαρακτηριστικά:

Αριθμός Οργανισμών	2
Αριθμός Peers ανά οργανισμό	2
Orderer	Solo
Μέγεθος Συναλλαγής σε bytes	10
Timeout Συναλλαγής	45 sec

Πίνακας 5.2: *Χαρακτηριστικά δικτύου Fabric στο πείραμα 5.2*

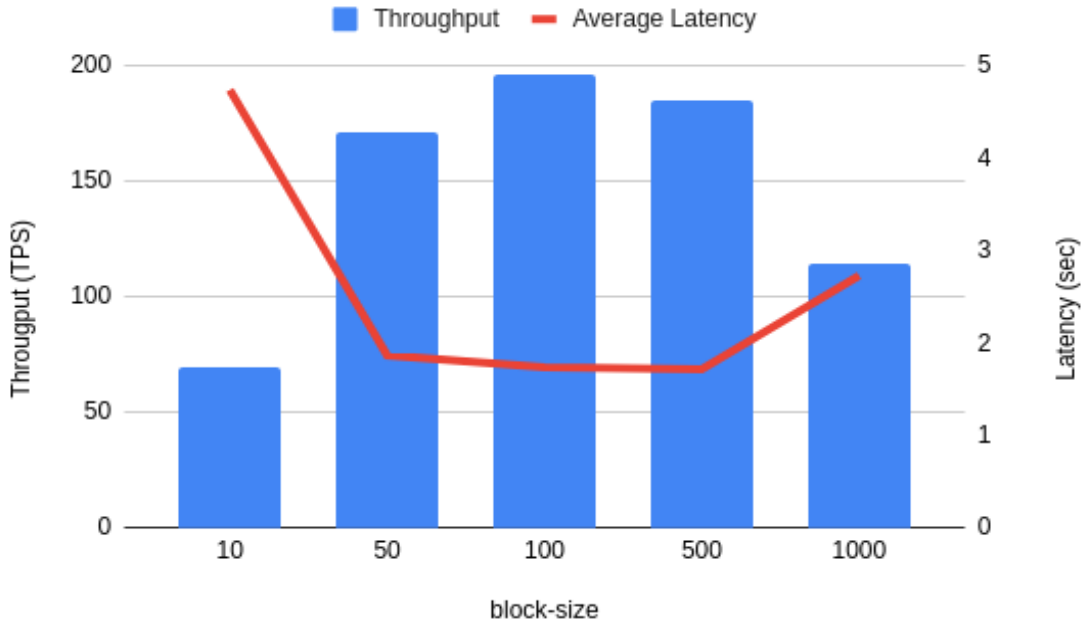
Ο λόγος που επιλέξαμε να βάλουμε Solo Orderer αντί για κάποια production-ready υλοποίηση είναι οτι δεν επηρεάζει τις μετρήσεις μας, καθώς το είδος του orderer δεν επηρεάζει την δημιουργία blocks, αλλά τον τρόπο με τον οποίον διαχειρίζονται το transaction log. Η χρήση RAFT, Kafka σε αυτό το πείραμα θα προσέθετε ένα επίπεδο πολυπλοκότητας που είναι περιττό για τα συμπεράσματα που θέλουμε να βγάλουμε.

Στο πλαίσιο αυτής της μελέτης δοκιμάσαμε την απόδοση του δικτύου με block size 10, 50, 100, 500, 1000. Η παράμετρος Batch Timeout τέθηκε στην τιμή των 2 δευτερολέπτων για τα πειράματα με 10, 50, 100 block-size, ώστε να μην δημιουργούνται τεράστιες καθυστερήσεις στο σύστημα σε περίπτωση αστοχίας και μη συμπλήρωσης του προαπαιτούμενου μεγέθους

block. Στα πειράματα με 500, 1000 block-size θέσαμε το Batch Timeout στα 3 δευτερόλεπτα, γιατί υπήρχε η περίπτωση στα 2 δευτερόλεπτα να κόβονται συνεχώς μικρότερα blocks απο το αναμενόμενο. Τα αποτελέσματα που πήραμε είναι τα παρακάτω:

Block Size	Throughput (TPS)	Average Latency (sec)
10	69.75	4.74
50	171.35	1.87
100	196.3	1.74
500	184.8	1.72
1000	114.65	2.73

Πίνακας 5.3: *Αποτελέσματα πειράματος 5.2.1*



Σχήμα 5.1: *Επίδραση του block size στην απόδοση*

Σχολιασμός αποτελεσμάτων

Παρατηρούμε οτι αυξάνοντας το block-size παρατηρείται αρχικά αύξηση του throughput και μείωση του latency, αλλα ξεπερνώντας ένα όριο η απόδοση μειώνεται πάλι. Αυτό είναι αναμενόμενο καθώς η διαδικασία επαλήθευσης του block απο τους peers είναι απαιτητική, χρονοβόρα και δύσκολη, και κυρίως δεν μπορεί να γίνει παράλληλα. Αυτό έχει σαν αποτέλεσμα, όταν το block ξεπεράσει ένα συγκεκριμένο μέγεθος, η ουρά επαλήθευσης να είναι πάρα πολύ μεγάλη και να δεσμεύει τους πόρους του peer για αρκετό χρονικό διάστημα, καθιστώντας τον peer μη αποκρίσιμο σε άλλες αιτήσεις.

Με βάση τα αποτελέσματά μας συμπαιρένουμε οτι το block-size 100 μας δίνει τα βέλτιστα αποτελέσματα, τόσο απο πλευράς throughput όσο και απο πλευράς latency.

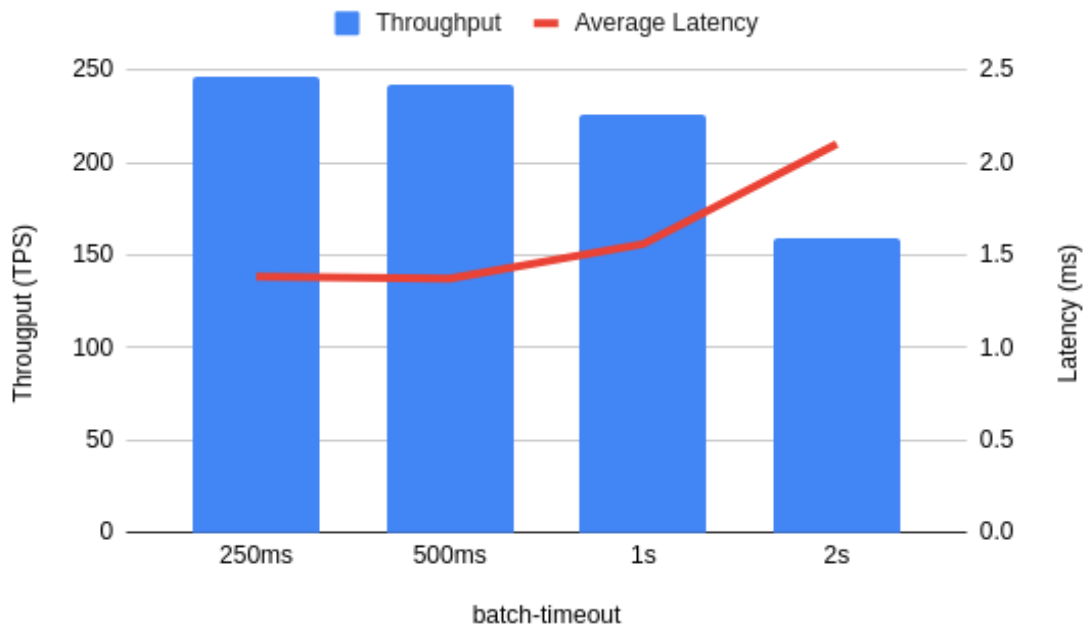
5.2.2 Η επίδραση του Batch Timeout

Για την μελέτη της επίδρασης του Batch Timeout στήσαμε ένα ίδιο δίκτυο με αυτό του προηγούμενου πειράματος, με την διαφορά ότι θέσαμε την παράμετρο block-size σε 10000, μία τεράστια τιμή, αναιρώντας με αυτόν τον τρόπο την επίδραση της συγκεκριμένης παραμέτρου. Στο πλαίσο αυτού του πειράματος, τα blocks δημιουργούνται μόνο όταν λήξει ο χρόνος που ορίσαμε στο Batch Timeout.

Εκτελούμε πειράματα με τιμές Batch Timeout 250ms, 500ms, 1sec, 2sec αντίστοιχα και λαμβάνουμε τα παρακάτω αποτελέσματα:

Batch Timeout	Throughput (TPS)	Average Latency (sec)
250ms	246.1	1.385
500ms	242.25	1.375
1s	225.65	1.56
2s	158.85	2.1

Πίνακας 5.4: *Αποτελέσματα πειράματος 5.2.2*



Σχήμα 5.2: *Επίδραση του batch timeout στην απόδοση*

Σχολιασμός αποτελεσμάτων

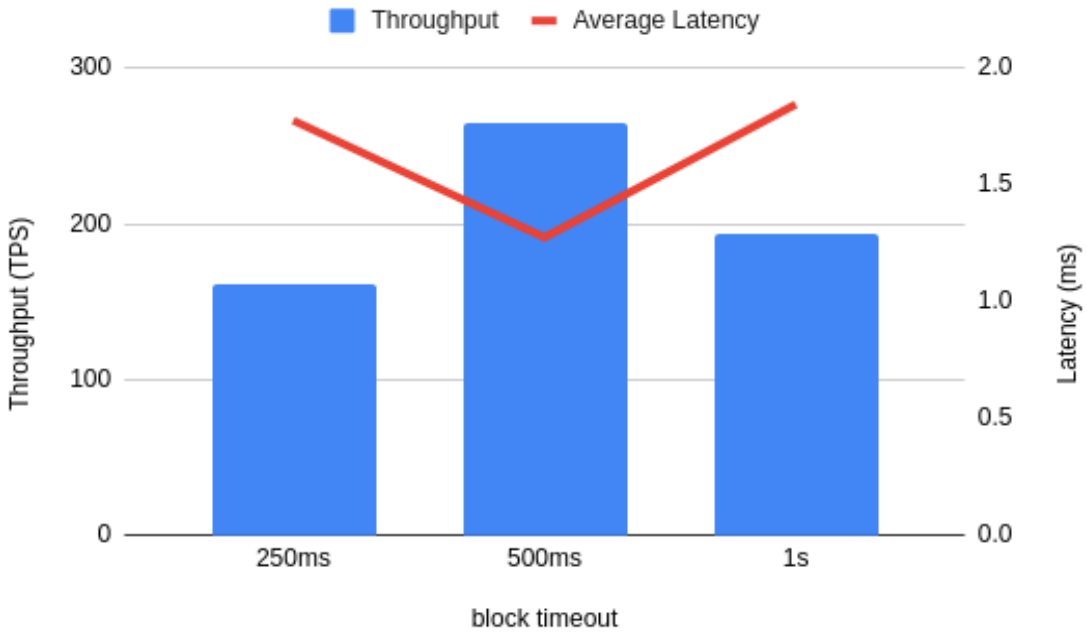
Στο σενάριο αυτό το μέγεθος του block καθορίζεται αποκλειστικά απο το πόσο συχνά κόβεται κάποιο block. Όσο πιο πολύ καθυστερήσουμε την δημιουργία ενός block, τόσο περισσότερες συναλλαγές θα περιλαμβάνει, με αποτέλεσμα να υπερφορτώνονται οι peers στην διαδικασία επαλήθευσης και να μειώνεται η απόδοση του συστήματος.

5.2.3 Συνδιασμός των δύο παραμέτρων και βέλτιστα αποτελέσματα

Σε αυτό το στάδιο έχοντας πλέον καταλήξει οτι για block-size μεγαλύτερο του 100 το σύστημά μας αποδίδει χειρότερα, δοκιμάζουμε διαφορετικές τιμές batch-timeout, καταγράφοντας την απόδοση του συστήματος. Συγκεκριμένα, δοκιμάσαμε τον συνδιασμό του block-size 100 με τα batch-timeout 250ms, 500ms, 1s αντίστοιχα και καταλήξαμε στα παρακάτω αποτελέσματα:

Block Size	Batch Timeout	Throughput (TPS)	Average Latency (sec)
100	250ms	161	1.775
100	500ms	264.6	1.275
100	1s	193.3	1.845

Πίνακας 5.5: *Αποτελέσματα πειράματος 5.2.3*



Σχήμα 5.3: *Συνδιαστική επίδραση των παραμέτρων στην απόδοση*

Σχολιασμός αποτελεσμάτων

Δεδομένου του ότι οι παράμετροι block-size και batch-timeout συνεργάζονται για να εξασφαλίζουν μία σταθερή ροή block στο σύστημα, παρατηρούμε οτι η χρήση του συνδιασμού batch-timeout 500ms και block-size εξασφαλίζει οτι τα blocks θα έχουν μέγιστο μέγεθος το optimal μέγεθος που παρατηρήθηκε στο πείραμα 5.1.1, και σε περίπτωση που κάποια συναλλαγή ¨ξεμείνει' σε ένα block δεν θα παραμείνει ανολοκλήρωτη για πολλή ώρα, προσθέτοντας καθυστέρηση στο σύστημα. Μειώνοντας το timeout στα 250ms τα blocks κόβονται πολύ γρήγορα, με αποτέλεσμα το μέγεθός τους να είναι συνήθως αρκετά μικρότερο απο 100 και

να πέφτει η απόδοση. Μεγαλώνοντας το timeout, σε περίπτωση που ένα σύνολο συναλλαγών ¨ξεμείνουν' σε ανολοκλήρωτο block, πρέπει να περιμένουμε τουλάχιστον 1 δευτερόλεπτο, γεγονός που οδηγεί σε περεταίρω καθυστερήσεις.

5.3 Η επίδραση του μεγέθους της συναλλαγής στην απόδοση του συστήματος

Σε αυτό το πείραμα θα μελετήσουμε την επίδραση του μεγέθους της συναλλαγής στο throughput και στο latency του συστήματος.

Η συναλλαγή στο fabric είναι μία γενική - αφαιρετική έννοια. Διαφορετικά business logic έχουν αρκετά διαφορετικές συναλλαγές, και μία βασική παράμετρος που επηρεάζει την απόδοση είναι το μέγεθος συναλλαγής.

Για να μελετήσουμε την επίδραση του block size στήνουμε ένα δίκτυο Hyperledger Fabric με τα εξής χαρακτηριστικά:

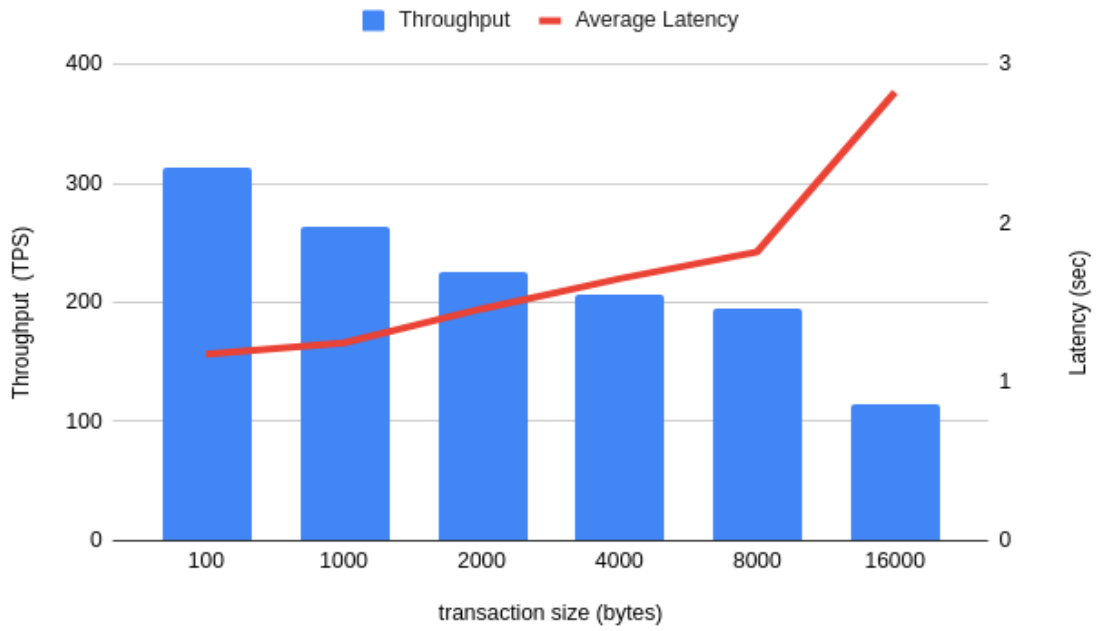
Αριθμός Οργανισμών	2
Αριθμός Peers ανά οργανισμό	2
Orderer	Solo
Timeout Συναλλαγής	45 sec
Block size	100
Batch Timeout	500ms

Πίνακας 5.6: *Χαρακτηριστικά δικτύου Fabric στο πείραμα 5.3*

Απο τα προηγούμενα πειράματα χρησιμοποιούμε τον συνδιασμό παραμέτρων block-size 100, batch-timeout 500ms, δεδομένου οτι είχε τα βέλτιστα αποτελέσματα. Στην συνέχεια, τρέχουμε benchmarks με μεταβλητό μέγεθος συναλλαγής, και παίρνουμε τα παρακάτω αποτελέσματα:

Transaction Size (bytes)	Throughput (TPS)	Average Latency (sec)
100	313.55	1.175
1000	263.8	1.245
2000	225.45	1.46
4000	207.2	1.65
8000	194.55	1.82
16000	114.05	2.285

Πίνακας 5.7: *Αποτελέσματα πειράματος 5.3*



Σχήμα 5.4: *Επίδραση του μεγέθους συναλλαγής στην απόδοση*

Σχολιασμός αποτελεσμάτων

Τα αποτελέσματα του συγκεκριμένου πειράματος είναι αναμενόμενα. Όταν αυξάνεται το μέγεθος της συναλλαγής αυξάνεται και η χρησιμοποίηση του δικτύου. Επίσης, σπαταλάται πολύ περισσότερος χρόνος στην αποστολή των συναλλαγών μεταξύ των διαφόρων συμμετέχοντων στο δίκτυο, και αυτό πολλαπλασιάζεται ανάλογα με το μέγεθος του block. Αυτό έχει σαν αποτέλεσμα την πολύ γρήγορη κατανάλωση του διαθεσιμου bandwidth και την συνολική πτώση της απόδοσης. Χαρακτηριστικό είναι οτι για μεγέθη συναλλαγών 32000 bytes και πάνω το σύστημα δεν κατάφερε να ανταποκριθεί θετικά σε ούτε μία συναλλαγή.

5.4 Η επίδραση της προσθήκης peers στην απόδοση

Σε αυτό το πείραμα θα μελετήσουμε το πώς ακριβώς επηρεάζεται η απόδοση απο την αύξηση του αριθμού των διαθέσιμων endorsers του δικτύου.

Για να μελετήσουμε την επίδραση του αριθμού των endorsers στήνουμε ένα δίκτυο Hyperledger Fabric με τα εξής χαρακτηριστικά:

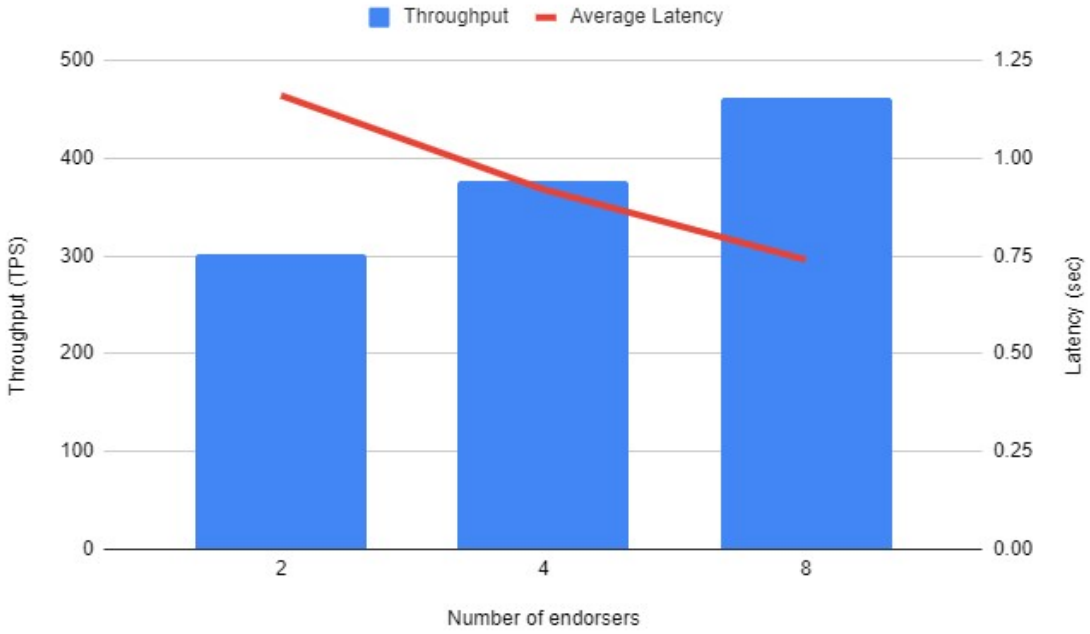
Αριθμός Οργανισμών	2
Orderer	Solo
Timeout Συναλλαγής	45 sec
Block size	100
Batch Timeout	500ms

Πίνακας 5.8: *Χαρακτηριστικά δικτύου Fabric στο πείραμα 5.4*

Σε ένα δίκτυο με αυτά τα χαρακτηριστικά, τρέχουμε πειράματα μεταβάλλοντας τον αριθμό των endorsers. Δοκιμάσαμε την απόδοση του δικτύου για 2,4,8 endorsers και λάβαμε τα παρακάτω αποτελέσματα:

Αριθμός Endorsing Peers	Throughput (TPS)	Average Latency (sec)
2	301.95	1.16
4	376.9	0.92
8	462	0.74

Πίνακας 5.9: *Αποτελέσματα πειράματος 5.4*



Σχήμα 5.5: *Επίδραση του αριθμού endorsing peers στην απόδοση*

Σχολιασμός αποτελεσμάτων

Παρατηρείται οτι η αύξηση του αριθμού των διαθέσιμων endorsers οδηγεί σε αύξηση της απόδοσης του δικτύου. Αυτό είναι αναμενόμενο, καθώς για μία απλή πολιτική endorsement όπως η 1-of-any που χρησιμοποιούμε, η οποία στην ουσία ζητάει ένα οποιοδήποτε endorsement για να γίνει δεκτή μία συναλλαγή, αξιοποιείται η ύπαρξη πολλών endorsers για την παράλληλη εκτέλεση του σταδίου execute της ζωής των συναλλαγών, οδηγώντας σε σημαντική αύξηση της απόδοσης. Προφανώς, για να γίνει αντιληπτή αυτή η διαφορά απόδοσης απαιτείται load-balancing μεταξύ των διαθέσιμων endorsers απο την πλευρά του πελάτη ο οποίος στέλνει συναλλαγές. Το caliper χρησιμοποιεί load-balancing και για αυτό παρατηρούμε και αυτήν την σημαντική διαφορά στην απόδοση. Η παρατήρηση αυτή θα αναλυθεί εκτενέστερα στην επόμενη πειραματική διάταξη.

5.5 Η επίδραση του load-balancing στην απόδοση

Κατα την διάρκεια της δημιουργίας των τοπολογιών για την εκτέλεση του πειράματος 5.4 δοκιμάσαμε δύο διαφορετικούς τρόπους προσθήκης περισσότερων endorsing peers στο δίκτυο. Ο πρώτος τρόπος ήταν το scale up του κάθε οργανισμού με περισσότερους peers, όπως και κάναμε στην προηγούμενη ενότητα. Ο δεύτερος ήταν η προσθήκη παραπάνω οργανισμών στο δίκτυο, οι οποίοι περιείχαν έναν συγκεκριμένο αριθμό απο peers.

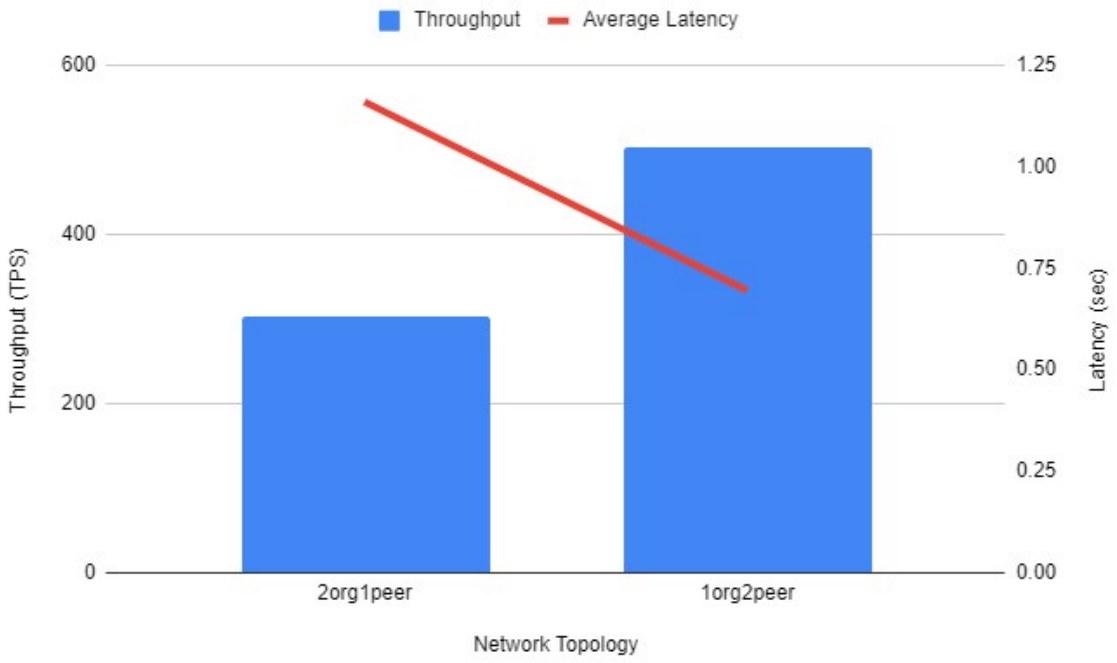
Σε μία πειραματική διάταξη για 2 peers, τρέξαμε πειράματα για την περίπτωση που είχαμε:

- 2 οργανισμούς, που ο καθένας περιείχε έναν peer (2org1peer)
- 1 οργανισμό που περιείχε 2 peers (1org2peer)

Τα αποτελέσματα που πήραμε ήταν τα εξής:

Κατανομή Endorsing Peers	Throughput (TPS)	Average Latency (sec)
2org1peer	301.95	1.16
1org2peer	502.8	0.695

Πίνακας 5.10: *Αποτελέσματα πειράματος 5.5*



Σχήμα 5.6: *Επίδραση της κατανομής των peers στην απόδοση*

Σχολιασμός αποτελεσμάτων

Τα αποτελέσματα αυτά σε πρώτη φάση φαίνονται παράδοξα, καθώς η πολιτική 1-of-any απαιτεί ένα οποιοδήποτε endorsement για να είναι αποδεκτή μία συναλλαγή, οπότε θεωρητικά και οι δύο τοπολογίες θα έπρεπε να έχουν παραπλήσια, αν όχι την ίδια, απόδοση.

Ο λόγος που υπάρχει αυτή η διαφορά της απόδοσης έγκειται στην εσωτερική λειτουργία του load generator που χρησιμοποιούμε, του caliper. Συγκεκριμένα, το caliper, σε περίπτωση που δεν του προσδιορίσει ο χρήστης τί πολιτική endorsement χρησιμοποιεί το δίκτυο, για να ικανοποιήσει κάθε πιθανή πολιτική, ζητάει ένα endorsement απο κάθε οργανισμό, θεωρεί δηλαδή πολιτική N-outOf-N (στην δική μας περίπτωση, οταν έχουμε 2 οργανισμούς, 2-outOf-2. Αυτό έχει σαν αποτέλεσμα, στην περίπτωση του 2org1peer να ζητάει χωρίς λόγο endorsement και απο τους 2 οργανισμούς, ενώ στην πραγματικότητα αρκεί μόνο ένα. Δεδομένου λοιπόν οτι θεωρεί οτι χρειάζεται απάντηση καί απο τους δύο οργανισμούς, αντί να έχει 2 peers που θα χρησιμοποιεί για να κάνει load-balancing, βλέπει μόνο έναν. Αντιθέτως, στην περίπτωση 1org2peer βλέπει 2 endorsing peers, οι οποίοι και αξιοποιούνται για την παραλληλοποίηση του execute κομματιού της συναλλαγής. Παρόμοια αποτελέσματα παρατηρούνται και στην περίπτωση 2org2peer, 1org4peer και πάει λέγοντας.

5.6 Η επίδραση της πολιτικής endorsement στην απόδοση

Σε αυτό το πείραμα θα μελετήσουμε το πώς ακριβώς επηρεάζεται η απόδοση απο την αύξηση του αριθμού των διαθέσιμων endorsers του δικτύου.

Για να μελετήσουμε την επίδραση του endorsement policy στήνουμε ένα δίκτυο Hyperledger Fabric με τα εξής χαρακτηριστικά:

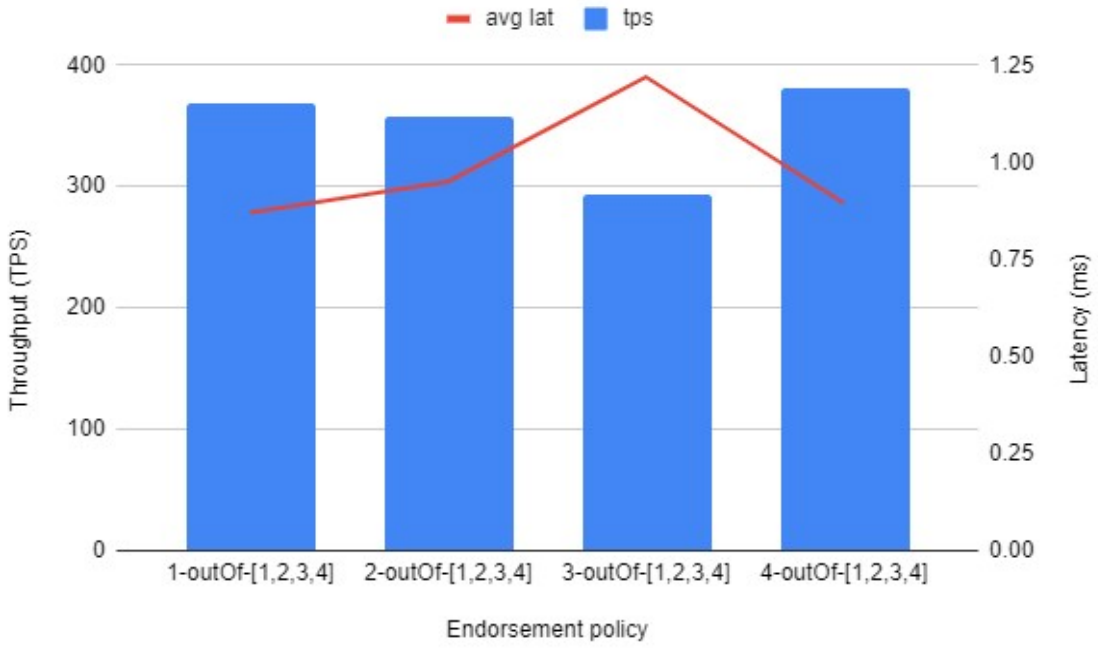
Αριθμός Οργανισμών	4
Peers ανα οργανισμό	2
Orderer	Solo
Timeout Συναλλαγής	45 sec
Block size	100
Batch Timeout	500ms

Πίνακας 5.11: *Χαρακτηριστικά δικτύου Fabric στο πείραμα 5.6*

Σε ένα δίκτυο με αυτά τα χαρακτηριστικά, τρέχουμε πειράματα μεταβάλλοντας την πολιτική endorsement του chaincode. Δοκιμάσαμε την απόδοση του δικτύου για 1-outOf-[1,2,3,4], 2-outOf-[1,2,3,4], 3-outOf-[1,2,3,4], 4-outOf-[1,2,3,4] endorsement policy. Επεξηγηματικά, να αναφέρουμε οτι η n-outOf-n πολιτικές απαιτούν n endorsements απο το σύνολο των peers. Λάβαμε τα παρακάτω αποτελέσματα:

Endorsement Policy	Throughput (TPS)	Average Latency (sec)
1-outOf-[1,2,3,4]	368.8	0.87
2-outOf-[1,2,3,4]	358.2	0.95
3-outOf-[1,2,3,4]	293.9	1.22
4-outOf-[1,2,3,4]	381.25	0.895

Πίνακας 5.12: *Αποτελέσματα πειράματος 5.6*



Σχήμα 5.7: *Επίδραση του endorsement policy στην απόδοση*

Σχολιασμός αποτελεσμάτων

Οι πολιτικές 1-outOf-[1,2,3,4] και 4-outOf-[1,2,3,4] έχουν παραπλήσια απόδοση. Αυτό οφείλεται στο οτι δεν εμπεριέχουν υποπολιτικές (η 1-outOf-[1,2,3,4] μεταφράζεται σε OR(Org1.Peer, Org2.Peer, Org3.Peer, Org4.Peer) και η 4-outOf-[1,2,3,4] σε AND(Org1.Peer, Org2.Peer, Org3.Peer, Org4.Peer)). Αντίθετα, υπάρχει πτώση απόδοσης στην 2-outOf-[1,2,3,4] και ακόμα μεγαλύτερη στην 3-outOf-[1,2,3,4]. Οι δύο τελευταίες πολιτικές εμπεριέχουν πολλές υποπολιτικές. Για παράδειγμα η 2-outOf-[1,2,3,4] μεταφράζεται ως OR((Org1.Peer AND Org2.Peer), (Org1.Peer AND Org3.Peer)) κλπ.

Το κομμάτι του validation της πολιτικής είναι μία δύσκολη διαδικασία γιατί εμπεριέχει πολλές επικυρώσεις κρυπτογραφικών πιστοποιητικών. Η χρήση πολιτικών που είναι πιο ¨πολύπλοκες' αυξάνουν την πολυπλοκότητα της διαδικασίας, γιατί η επικύρωση της πολιτικής απαιτεί πολλά στάδια.

Σε αυτό το πλαίσιο, είναι λογική η πτώση της απόδοσης στις σύνθετες πολιτικές. Η πτώση αυτή γίνεται πιο αισθητή στην 3-outOf-[1,2,3,4] γιατι περιπλέκει ακόμα περισσότερο το στάδιο επικύρωσης.

Μέρος III

Επίλογος

Κεφάλαιο 6

Επίλογος

Σε αυτό το κεφάλαιο θα γίνει μια σύνοψη της διπλωματικής και θα προταθούν μελλοντικές επεκτάσεις.

6.1 Σύνοψη - Συμπεράσματα

Μέσω της εκπόνησης αυτής της εργασίας απέκτησα βαθιά κατανόηση του τρόπου λειτουργίας του Hyperledger Fabric και των πλεονεκτημάτων που έχει σε σχέση με προηγούμενες υλοποιήσεις blockchain στην ανάπτυξη Distributed Applications. Ασχοληθήκαμε εκτενώς με την απόδοση του Fabric στο επίπεδο που αφορά τις παραμέτρους του δικτύου και το πώς η καθεμία επηρεάζει την απόδοση.

Βασικό συμπέρασμα που βγαίνει απο αυτήν την εργασία είναι πως ο βαθμός παραμετροποίησης του Hyperledger Fabric είναι τεράστιος και υπάρχουν πολλές παράμετροι που προσαρμόζουν την απόδοσή του. Δεν υπάρχει κάποιος one-size-fits-all κανόνας για τις τιμές αυτών των παραμέτρων, καθώς είναι άρρηκτα δεμένος με το use case στο οποίο απευθυνόμαστε.

Πέρα απο αυτό, μπορούμε να βγάλουμε το συμπέρασμα ότι, για να εξυπηρετήσουμε μεγαλύτερο αριθμό αιτήσεων, πρέπει αφενός να έχουμε αρκετούς endorsers για να τις απαντήσουν, και αφετέρου να επιλέξουμε μία πολιτική endorsement η οποία να καλύπτει μεν τις ανάγκες του σεναρίου μας, αλλά να μην χαρακτηρίζεται απο περιττή πολυπλοκότητα, καθώς αυτό θα ρίξει την απόδοση του δικτύου σημαντικά. Ένας εύκολος τρόπος να αυξηθεί η απόδοση του δικτύου είναι η προσθήκη παραπάνω endorsing peers, ώστε να παραλληλοποιηθεί η διαδικασία του endorsement.

Επίσης, μεγάλη σημασία έχει το chaincode που τρέχει στο δίκτυο, και πιο συγκεκριμένα το μέγεθος της συναλλαγής που αυτό απαιτεί. Πέρα απο το payload της συναλλαγής, σημαντικό μέγεθος καταλαμβάνουν τα endorsements, τα ψηφιακά πιστοποιητικά κλπ. Είδαμε οτι όσο μεγαλύτερο μέγεθος έχει η συναλλαγή μας, τόσο πιο κακή είναι η απόδοση του δικτύου. Συμπαιρένουμε λοιπόν οτι πρέπει να γίνει προσεκτικός σχεδιασμός του chaincode, με βασικό γνώμονα το μέγεθος της συναλλαγής.

Σημαντική είναι επίσης και η σωστή δημιουργία των εφαρμογών - πελατών που είναι υπεύθυνες για την δημιουργία των συναλλαγών, καθώς μία λανθασμένη επιλογή σε αυτούς αναιρεί τις προσπάθειες βελτιστοποίησης του δικτύου. Για παράδειγμα, αν η εφαρμογή θεωρεί κάποια διαφορετική πολιτική endorsement απο την πραγματική, θα συμπεριφέρεται με βάση αυτό και η διαδικασία δημιουργίας - διεκπεραίωσης συναλλαγών θα καθυστερήσει σημαντικά.

Τέλος, πολύ σημαντικό είναι να γίνει σωστή επιλογή των παραμέτρων block-size και batch timeout με βάση τους διαθέσιμους πόρους που υπάρχουν στο σύστημα. Με βάση την μελέτη μας, αποτελούν τον πλέον σημαντικό παράγοντα που επηρεάζει την απόδοση του συστήματος και η επιλογή τους αποτελεί καθοριστικό παράγοντα της απόδοσης.

Συνοψίζοντας, καταλήγουμε στο συμπέρασμα οτι η δημιουργία ενός αποδοτικού δικτύου Hyperledger Fabric είναι μια σύνθετη διαδικασία, και εμπεριέχει πολλές παραμέτρους που πρέπει κανείς να λάβει υπόψιν για τον ορθό σχεδιασμό του δικτύου.

6.2 Μελλοντικές Επεκτάσεις

Σε αυτό το σημείο θα επιχειρήσουμε να προτείνουμε μελλοντικές επεκτάσεις της εργασίας αυτής ως προς το πεδίο έρευνας.

- Μελέτη της επίδρασης του ordering service στην απόδοση του συστήματος
- Μελέτη της απόδοσης των ανεπίσημων υλοποιήσεων ordering service, με εμβάθυνση κυρίως στην χρήση Byzantine Fault Tolerant υλοποιήσεων και της επίδρασης στην απόδοση του συστήματος
- Μελέτη της επίδρασης της χρήσης τεχνικών προστασίας ιδιωτικών δεδομένων που προσφέρει το Fabric στην απόδοση του συστήματος
- Μελέτη της επίδρασης της προγραμματιστικής υλοποίησης του chaincode και συγκεκριμένα εξαγωγή συμπερασμάτων για βέλτιστες προγραμματιστικές τακτικές για την δημιουργία αποδοτικών έξυπνων συμβολαίων για το Hyperledger Fabric

Βιβλιογραφία

[1] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi και Kian-Lee Tan. *BLOCKBENCH: A Framework for Analyzing Private Blockchains*. *CoRR*, αβς/1703.04057, 2017.

[2] Dimitri Saingre, Thomas Ledoux και Jean Marc Menaud. *BCTMark: a framework for benchmarking blockchain technologies*. *AICCSA 2020 - 17th IEEE/ACS International Conference on Computer Systems and Applications*, σελίδες 1–8, Antalya, Turkey, 2020. IEEE.

[3] P. Thakkar, S. Nathan και B. Viswanathan. *Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform*. *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, σελίδες 264–276, 2018.

[4] Haris Javaid, Chengchen Hu και Gordon Brebner. *Optimizing Validation Phase of Hyperledger Fabric*, 2019.

[5] Canhui Wang και Xiaowen Chu. *Performance Characterization and Bottleneck Analysis of Hyperledger Fabric*, 2020.

[6] Thanh Son Lam Nguyen, Guillaume Jourjon, Maria Potop-Butucaru και Kim Thai. *Impact of network delays on Hyperledger Fabric*. *CoRR*, αβς/1903.08856, 2019.

[7] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. *Cryptography Mailing list at https://metzdowd.com*, 2009.

[8] Gavin Wood. *Ethereum: A secure decentralised generalised transaction ledger*.

[9] E. A. Akkoyunlu, K. Ekanadham και R. V. Huber. *Some Constraints and Tradeoffs in the Design of Network Communications*. *SIGOPS Oper. Syst. Rev.*, 9(5):67–74, 1975.

[10] L. Lamport, R. Shostak και M. Pease. *The Byzantine Generals Problem*. *ACM Trans. Program. Lang. Syst.*, 4:382–401, 1982.

[11] Miguel Castro και Barbara Liskov. *Practical Byzantine Fault Tolerance*. *OSDI*, 1999.

[12] Alysson Bessani, João Sousa και Eduardo Alchieri. *State machine replication for the masses with BFT-SMART*. σελίδες 355–362, 2014.

[13] Leslie Lamport. *The part-time parliament*. 2019.

[14] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn και George Danezis. *Consensus in the Age of Blockchains*, 2017.

[15] Maher Alharby, Amjad Aldweesh και Aadvan Moorsel. *Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research (2018)*, 2019.

[16] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin και F. Wang. *An Overview of Smart Contract: Architecture, Applications, and Future Trends. 2018 IEEE Intelligent Vehicles Symposium (IV)*, σελίδες 108–113, 2018.

[17] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco και Jason Yellick. *Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. CoRR*, αβς/1801.10228, 2018.

[18] Shafi Goldwasser, Silvio Micali και Chales Rackoff. *The knowledge complexity of interactive proof-systems*. 1989.

[19] *gRPC*. `https://en.wikipedia.org/wiki/GRPC`.

[20] Diego Ongaro και John Ousterhout. *In Search of an Understandable Consensus Algorithm. Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, σελίδα 305–320, USA, 2014. USENIX Association.

[21] Jay Kreps. *Kafka : a Distributed Messaging System for Log Processing*. 2011.

[22] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi και Dawn Song. *The Honey Badger of BFT Protocols*. σελίδες 31–42, 2016.