ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ,
ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Κατανεμημένου Συστήματος Εξυπηρετητών

Διπλωματική Εργασία
της
Γεωργίας Σαρρή

Επιβλέπων: Θεοδώρα Βαρβαρίγου,
Καθηγήτρια Ε.Μ.Π

Αθήνα, Ιούνιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ,
ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Κατανεμημένου Συστήματος Εξυπηρετητών

Διπλωματική Εργασία
της
Γεωργίας Σαρρή

Επιβλέπων: Θεοδώρα Βαρβαρίγου,
Καθηγήτρια Ε.Μ.Π

....

Αθήνα, Ιούνιος 2016

.....
Γεωργία Σαρρή
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Ευχαριστίες

Η διπλωματική αυτή εκπονήθηκε στο Εργαστήριο Distributed Knowledge and Media Systems Group του Τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη της Καθηγήτριας Θεοδώρας Βαρβαρίγου.

Θα ήθελα να ευχαριστήσω θερμά την καθηγήτρια κυρία Θεοδώρα Βαρβαρίγου για την ευκαιρία που μου προσέφερε και για την εμπιστοσύνη που μου έδειξε με την ανάθεση της παρούσας διπλωματικής. Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα τον υποψήφιο διδάκτορα Παύλο Κρανά για την πολύτιμη βοήθειά του κατά τη διάρκεια της εργασίας και την υπομονή που έδειξε μέχρι την διεκπεραίωσή της.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς και τους θείους μου για την εμπιστοσύνη και την στήριξη που μου έδωσαν καθόλη την διάρκεια της ακαδημαϊκής μου πορείας. Αλλά και τις δυο πολύ καλές μου φίλες Καλλιόπη και Δήμητρα που δεν έπαψαν στιγμή να βρίσκονται στο πλάι μου όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Listings

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο Διπλωματικής

Η διπλωματική αυτή εργασία αποτελεί μέρος ενός ευρύτερου έργου, του CoherentPaaS , το οποίο έχει ως στόχο την ενσωμάτωση ποικίλων SQL μηχανών καθώς και no-SQL βάσεων δεδομένων σε μια υπηρεσία νέφους PaaS . Όλα τα συστήματα θα προγραμματιστούν να χρησιμοποιούν μια ενιαία γλώσσα προγραμματισμού για τα ερωτήματα προς τις βάσεις ενώ ένα επεκτάσιμο σύστημα διαχείρισης συναλλαγών θα εξασφαλίζει την ολιστική συνοχή ανάμεσα στις βάσεις δεδομένων.

Αντικείμενο της διπλωματικής εργασίας είναι η ανάπτυξη ενός δικτύου εξυπηρετητών χρησιμοποιώντας το μοντέλο master-worker προκειμένου να εξυπηρετούνται τα αιτήματα των πελατών προς κάποια βάση δεδομένων. Η εργασία είχε ως στόχο την εξοικείωση με το συγκεκριμένο μοντέλο συναλλαγών καθώς και με εργαλεία για τον συγχρονισμό και την επικοινωνία μεταξύ των εξυπηρετητών. Για το σκοπό αυτό χρησιμοποιήθηκαν ο Apache Zookeeper για τον συγχρονισμό και το Apache Avro για την επικοινωνία.

1.1.1 Λέξεις Κλειδιά

Cloud Computing, NoSQL βάσεις δεδομένων, υπολογιστικό νέφος, master-worker , client-server , πελάτης-εξυπηρετητής, Apache Zookeeper , Apache Avro , κατανεμημένα συστήματα, συγχρονισμός

1.2 Abstract

This thesis was developed as a part of a larger project, the CoherentPaas, which intends for the integration of a diversity of SQL engines as well as no-SQL data stores in a single cloud PaaS. All these systems will be programmed by using a cloud multi-datastore query language (CloudMdsQL) under a uniform paradigm, and a scalable transactional management system

will provide holistic coherence across data stores. This deliverable analyzes the current state-of-the-art of the three data stores (graph database, key-value and document) with respect to the requirements of this project, and then, for each data store, it identifies and proposes the changes required to adapt it to a cloud PaaS. In particular, a network of servers was developed following the master-worker model to act as a middleware between the clients and the database. The clients send the queries to the servers and they process them and send them to the database. One master is elected, whose job is to assign the right server to a client. Furthermore two extra tools were used; Apache Zookeeper was used in order to maintain the synchronization among the servers and the master and Apache Avro was used to handle the communication between all the participants.

1.2.1 Key Words

Cloud Computing, NoSQL Databases, Distributed Systems, client-server, master-worker, Apache Zookeeper, Apache Avro, Middleware

1.3 Οργάνωση Κειμένου

Το κείμενο της διπλωματικής εργασίας χωρίζεται σε δυο μέρη. Το πρώτο μέρος αποτελείται από δυο κεφάλαια στα οποία γίνεται μια θεωρητική αναφορά στις τεχνολογίες πάνω στις οποίες αναπτύχθηκε το ευρύτερο έργο του οποίου αποτελεί κομμάτι η παρούσα εργασία και οι οποίες αποτελούν σταθμούς για την εξέλιξη και την διαμόρφωση της πληροφορικής. Στο κεφάλαιο δυο μελετάται το υπολογιστικό νέφος, δίνεται ένας ορισμός, μια ιστορική αναδρομή για την εξέλιξή του μέχρι τις μέρες μας και εξετάζονται η αρχιτεκτονική, οι παροχές και τα πλεονεκτήματα και μειονεκτήματά του. Στο κεφάλαιο τρία γίνεται αναφορά στις no-SQL βάσεις δεδομένων, παρουσιάζονται οι βασικοί τύποι αυτών των βάσεων, γίνεται μια σύγκριση με τις SQL βάσεις ενώ εξετάζονται και κάποιες βασικές τους ιδιότητες.

Το δεύτερο μέρος αναφέρεται στις πιο τεχνικές λεπτομέρειες τις διπλωματικής εργασίας. Στο κεφάλαιο τέσσερα δίνεται μια ανάλυση του προβήματος που μελετήθηκε. Στο κεφάλαιο πέντε γίνεται αναφορά στην σχεδίαση του μοντέλου καθώς επίσης αναλύεται και η λειτουργία των δομικών στοιχείων της αρχιτεκτονικής που χρησιμοποιήθηκε ενώ παρέχονται πίνακες και διαγράμματα κλάσεων με τις μεθόδους του καθενός και τη λειτουργικότητά τους. Στο κεφάλαιο έξι αναλύονται τα προγραμματιστικά εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του προτεινόμενου μοντέλου. Τέλος, στο κεφάλαιο επτά γίνεται ο έλεγχος της ορθής λειτουργίας του συστήματος σε πραγματικό χρόνο και εξηγούνται τα αποτελέσματα εκτέλεσης. Επιπλέον δίνεται ένας επίλογος όπου αναφέρονται μελλοντικές επεκτάσεις του συστήματος που υλοποιήθηκε.

Μέρος I

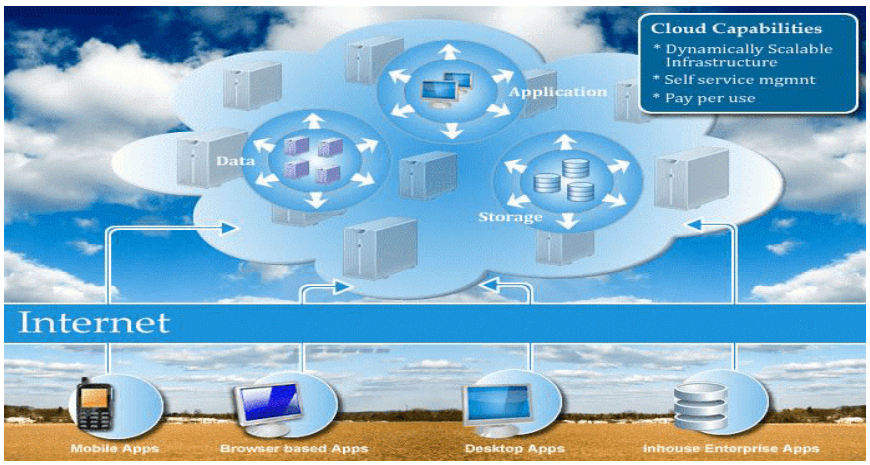
Θεωρητικό Υπόβαθρο

Κεφάλαιο 2

Cloud Computing

2.1 Ορισμός

Το Cloud Computing (υπολογιστικό νέφος) είναι μια έννοια η οποία ήρθε στο προσκήνιο και άρχισε να απασχολεί τον κλάδο της πληροφορικής κυρίως μέσα στην τελευταία δεκαετία. Με βάση τον ορισμό που χρησιμοποιείται από το NIST (The National Institute of Standards and Technology) [12] όταν αναφερόμαστε στο υπολογιστικό νέφος μιλάμε ουσιαστικά για ένα μοντέλο που επιτρέπει ευέλικτη και άμεση δικτυακή πρόσβαση σε έναν κοινό και παραμετροποιήσιμο σύνολο υπολογιστικών πόρων, όπως δίκτυα, εξυπηρετητές, εφαρμογές και υπηρεσίες, οι οποίοι παρέχονται ταχύτατα και με ελάχιστο κόστος διαχείρισης και ελάχιστη αλληλεπίδραση με τον πάροχο της υπηρεσίας. Ουσιαστικά, το «νέφος» είναι η παροχή υπολογιστικών υπηρεσιών μέσω του διαδικτύου.



Σχήμα 2.1: Cloud Computing Overview [18]

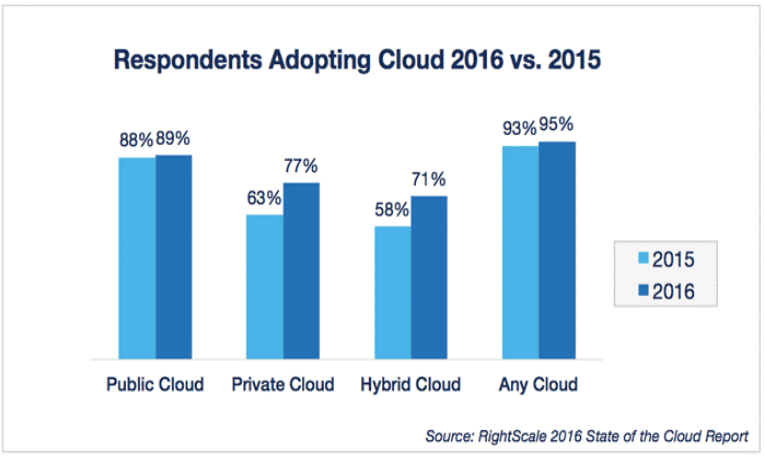
2.2 Εισαγωγή

Ο τομέας της πληροφορικής συνεχώς μεταβάλλεται με όλο και περισσότερες νέες τεχνολογίες να κάνουν την εμφάνιση τους συντελώντας στην ριζική αναδιαμόρφωση του. Ένα τέτοιο μοντέλο προγραμματισμού, το οποίο άρχισε να αναπτύσσεται τα τελευταία χρόνια, είναι αυτό του υπολογιστικού σύννεφου. Σε αυτό το μοντέλο δεδομένα και διάφοροι υπολογισμοί διαχειρίζονται από κέντρα δεδομένων κάποιας τρίτης μονάδας, ανεξάρτητης του δικτύου του χρήστη. Όταν αναφερόμαστε στο υπολογιστικό σύννεφο, αναφερόμαστε στο υλικό, στα συστήματα λογισμικού και στις εφαρμογές που παρέχονται ως υπηρεσίες στο διαδίκτυο. Όταν αποθηκεύουμε για παράδειγμα τις φωτογραφίες μας στο διαδίκτυο αντί στον προσωπικό μας υπολογιστή, όταν χρησιμοποιούμε τις υπηρεσίες του ηλεκτρονικού ταχυδρομείου ή των κοινωνικών δικτύων τότε χρησιμοποιούμε το υπολογιστικό σύννεφο. Άλλα και σε επίπεδα οργανισμών ή εταιρειών όταν χρειάζονται περισσότερη υπολογιστική ισχύ για τις εφαρμογές τους ή περισσότερο χώρο αποθήκευσης μπορούν να επιλέξουν μια λύση μέσω του υπολογιστικού σύννεφου για να καλύψουν αυτές τις ανάγκες. Αυτά είναι μόνο μερικά από τα παραδείγματα χρήσης του «νέφους» από τα οποία μπορεί να γίνει σαφές πως το Cloud Computing γίνεται σιγά σιγά μια όλο και αυξανόμενης ζήτησης υπηρεσία. [19]

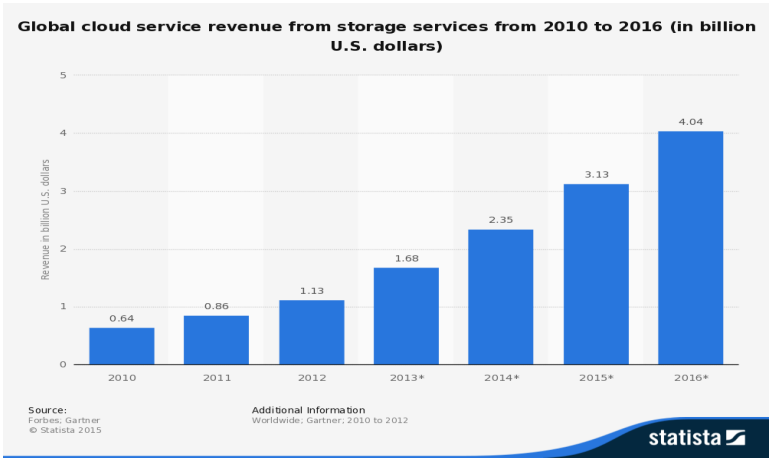
Τα χαρακτηριστικά που κάνουν το υπολογιστικό σύννεφο όλο και πιο δημοφιλή στις μέρες μας είναι αρχικά η ευελιξία που έχει, δηλαδή μπορεί να παρέχει υπηρεσίες ανάλογα με τη ζήτηση που υπάρχει επιτρέποντας στον χρήστη να ρυθμίζει εκείνος το τι χρειάζεται και αποδεσμεύοντας τον με αυτόν τον τρόπο από τον πάροχο. Έπειτα είναι και η ευκολία πρόσβασης στις υπηρεσίες του, όπου πολλές φορές το μόνο που χρειάζεται για να έχει πρόσβαση κάποιος στο «νέφος» είναι ένας περιηγητής δικτύου ενώ τόσο το είδος της συσκευής (είτε είναι κινητό τηλέφωνο είτε είναι ηλεκτρονικός υπολογιστής) όσο και η τοποθεσία δεν δεσμεύουν την παροχή τους. Επιπλέον είναι εύκολη η συντήρηση των υπηρεσιών αυτών αφού ο χρήστης δεν χρειάζεται να εγκαταστήσει κάποιο επιπλέον υλικό στις συσκευές του και δεν χρειάζεται να έχει κάποια επιπλέον τεχνογνωσία. Γενικά μπορούμε να διαπιστώσουμε πως όσο πιο πολύ εξελίσσεται η τεχνολογία και διευκολύνεται η πρόσβαση στο «νέφος» τόσο πιο πολύ μεγάλη απήχηση έχει και προτιμάται έναντι εναλλακτικών μεθόδων.

2.3 Ιστορική Αναδρομή

Όταν σκεφτόμαστε για το Cloud Computing θεωρούμε πως είναι μια τεχνολογία που δημιουργήθηκε μέσα στον 21ο αιώνα. Αυτό όμως δεν είναι αλήθεια, η ιδέα του υπολογιστικού σύννεφου είχε αρχίσει να αναπτύσσεται χρόνια πιο πριν. Η αρχική ιδέα για το υπολογιστικό σύννεφο ξεκίνησε ουσιαστικά το 1950 όπου πολλαπλοί χρήστες μέσω πολύ απλών τερματικών μπορούσαν να έχουν πρόσβαση σε μια κεντρική δομή και να τρέξουν εκεί τις εφαρμογές τους. Η ιδέα



(α′) Adopting Cloud 2015 vs 2016



(β′) Cloud Revenue

Σχήμα 2.2: Cloud Computing Usage And Revenue Statistics

όμως δεν τελεσφόρησε εξαιτίας του κόστους για την απόκτηση και συντήρηση αυτών των κεντρικών μονάδων για κάθε χρήστη καθώς επίσης και λόγω του ότι για τις εφαρμογές εκείνης της εποχής ούτε ιδιαίτερη υπολογιστική ισχύ ούτε μεγάλη μνήμη απαιτούνταν. [7]

Το 1960 δημιουργήθηκε η ιδέα για ένα παγκόσμιο δίκτυο υπολογιστών (J.C.R Licklider–Intergalactic Computer Network) που θα αποτελούσε τον πρόγονο του διαδικτύου και του υπολογιστικού νέφους. Στη ίδια δεκαετία εισάχθηκε και η ιδέα για το μοίρασμα του χρόνου μέσω διαδικασιών που έστελναν αιτήματα για επεξεργασία δεδομένων σε κάποια κεντρική μονάδα. Αυτή η ιδέα αρχικά συνδέθηκε με μεγάλα ονόματα όπως των IBM και DEC .

Μέχρι τις αρχές του 1970 είχαν δημιουργηθεί ολοκληρωμένες πλατφόρμες που υποστήριζαν την ιδέα για το μοίρασμα του χρόνου όπως οι Multics (σε υλικό ΓΕ), Cambridge CTSSκαι οι πρώτες UNIX θύρες (σε υλικό DEC). Παρόλα αυτά ακόμα κυριαρχούσε η ιδέα του μοντέλου του κέντρου δεδομένων όπου οι χρήστες έστελναν τις εργασίες σε κεντρικές μονάδες IBM . Επιπλέον μέσα σε αυτήν την δεκαετία δημιουργήθηκε και η ιδέα των εικονικών μηχανών (Virtual Machines) όπου έγινε δυνατό το σενάριο εκτέλεσης ενός ή και περισσοτέρων λειτουργικών συστημάτων σε απομονωμένο περιβάλλον. Ολόκληροι εικονικοί υπολογιστές μπορούσαν να τρέχουν σε μια υπαρκτή μονάδα που με τη σειρά της μπορούσε να λειτουργεί με εντελώς διαφορετικό λειτουργικό σύστημα. Μέσω των εικονικών μηχανημάτων η ιδέα για ένα κεντρικό σύστημα στο οποίο έχουν όλοι κοινή πρόσβαση εξελίχθηκε επιτρέποντας σε πολλά ξεχωριστά προγραμματιστικά περιβάλλοντα να συνυπάρχουν σε μια μονάδα.

Τη δεκαετία του 1990 οι εταιρείες τηλεπικοινωνιών που αρχικά πρόσφεραν μόνο point-to-point κυκλώματα δεδομένων άρχισαν να προσφέρουν υπηρεσίες

εικονικών ιδιωτικών δικτύων (VPN) οι οποίες είχαν αντίστοιχη ποιότητα και πιο χαμηλό κόστος. Αντί να χτίζουν δομές που να επιτρέπουν στους χρήστες να έχουν ο καθένας την δικιά του σύνδεση, έδωσαν σε κάθε χρήστη πρόσβαση σε μια κοινή δομή. Επιπλέον άρχισαν να ρυθμίζουν την κίνηση σε μια προσπάθεια εξισορρόπησης του φορτίου στους εξυπηρετητές τους, με αυτόν τον τρόπο μπορούσαν να χρησιμοποιούν το bandwidth του δικτύου τους πιο αποτελεσματικά.

Από το 2000 δημιουργήθηκε το Cloud Computing με την έννοια που το γνωρίζουμε σήμερα. Το 2006 η Amazon παρουσίασε το Elastic Compute Cloud , που ήταν η πρώτη φορά που παρουσιάστηκαν στο κοινό υπηρεσίες υποδομής. Το 2008 το Open Nebula της NASA έγινε το πρώτο ανοιχτό λογισμικό για την δημιουργία ιδιωτικών και υβριδικών σύννεφων. Ενώ την ίδια χρονιά άρχισαν οι προσπάθειες για την βελτίωση της ποιότητας των υπηρεσιών στο σύστημα IRMOS χρηματοδοτούμενο από την Ευρωπαϊκή Επιτροπή, όπου οδήγησε στην δημιουργία ενός περιβάλλοντος «νέφους» πραγματικού χρόνου. Εκ τότε είχαμε την παρουσίαση πολλών υποδομών νέφους με την κορύφωση το 2012, όπου η Oracle ανακοίνωσε το Oracle Cloud , την πρώτη υποδομή που πρόσφερε στον χρήστη ένα ολοκληρωμένο σετ παροχής υπολογιστικών υπηρεσιών, όπου συμπεριλάμβαναν παροχή εφαρμογών, πλατφορμών και υποδομών.

2.4 Χαρακτηριστικά

Έχοντας δώσει έναν ορισμό και μια ιστορική αναδρομή για το Cloud Computing μπορούμε να δούμε τα κυριότερα χαρακτηριστικά του που το κάνουν σήμερα τόσο δημοφιλές. Με βάση το NIST (The National Institute of Standards and Technology) [12] το υπολογιστικό σύννεφο έχει τα εξής βασικά χαρακτηριστικά :

- **Κατά παραγγελία υπολογιστικό μοντέλο**: Ο χρήστης μπορεί μόνος του να επιλέξει τους πόρους τους οποίους χρειάζεται και να τους προσαρμόζει ανάλογα με τις απαιτήσεις που έχει κάθε φόρα. Ενώ είναι τελείως αποδεσμευμένος από τον πάροχο των υπηρεσιών.

- **Ευρεία δικτυακή πρόσβαση**: Όλες οι υπηρεσίες είναι διαθέσιμες μέσω του δικτύου και είναι προσβάσιμες από τυποποιημένους μηχανισμούς που προωθούν την χρήση τόσο ελαφριών όσο και πιο βαριών πλατφορμών (π.χ. κινητά τηλέφωνα, ηλεκτρονικούς υπολογιστές, σταθμούς εργασίας κ.λ.π.)

- **Συγκέντρωση πόρων**: όλοι οι πόροι του παρόχου είναι συγκεντρωμένοι ώστε να εξυπηρετούν πολλαπλούς χρήστες με διαφορετικά φυσικά και εικονικά μέσα να ανατίθενται και να ανακατανέμονται δυναμικά, ανάλογα με τις ανάγκες του χρήστη.

- **Ελαστικότητα και Ευχρηστία**: Ίσως από τα σημαντικότερα χαρακτηριστικά του Cloud Computing . Με την άμεση δέσμευση και αποδέσμευση πόρων οι χρήστες μπορούν πολύ εύκολα να καλύπτουν οποιαδήποτε ανάγκη παρουσιάζεται.

Πέρα από αυτά τα χαρακτηριστικά μπορούμε να πούμε ότι το νέφος έχει μερικά ακόμα πλεονεκτήματα [19] :

- **Βασίζεται στο διαδίκτυο**: Όλες οι υπηρεσίες που παρέχονται είναι εκτός της έδρας του χρήστη και φτάνουν σε αυτόν μέσω του διαδικτύου. Δηλαδή, ο χρήστης δεν χρειάζεται να εγκαθιστά στον χώρο του κάποιο υλικοτεχνικό εξοπλισμό.

- **Εύχρηστο**: Δεν χρειάζεται επιπλέον τεχνογνωσία για την χρήση των υπηρεσιών του νέφους. Μάλιστα σε πολλές περιπτώσεις παρέχονται και προκαθορισμένα μοντέλα υπηρεσιών με ήδη ρυθμισμένες παραμέτρους για να διευκολύνουν ακόμα περισσότερο τους χρήστες.

- **Κοστολόγηση με βάση τη χρήση**: Το κόστος ρυθμίζεται ανάλογα με την αυξομειούμενη χρήση των υπηρεσιών που κάνει ο πελάτης. Δεν υπάρχει δηλαδή κάποια σταθερή χρέωση άλλα ο πελάτης χρεώνεται ανάλογα με τους πόρους που δεσμεύει. Η μέθοδος αυτή είναι πιο οικονομική από την αγορά και συντήρηση υλικοτεχνικών υποδομών και βοηθά κυρίως τους νέους χρήστε που έχουν έλλειψη αρχικού κεφαλαίου για ν αποκτήσουν απαραίτητο τεχνικό εξοπλισμό να δημιουργήσουν τις εφαρμογές τους.

- **Κλιμακούμενο**: Οι χρήστες δεν περιορίζονται με στατική δέσμευση πόρων αλλά μπορούν δυναμικά να αυξομειώνουν τη χρήση τους ανάλογα με τις ανάγκες τους.

- **Ανεξαρτησία ως προς τοποθεσία και είδος συσκευής**: οι χρήστες δεν δεσμεύονται από το είδος της συσκευής που έχουν για να χρησιμοποιήσουν τις υπηρεσίες του νέφους αφού υπάρχουν ειδικά διαμορφωμένες πλατφόρμες συμβατές με όλες τις συσκευές, ενώ πολλές φορές το μόνο που χρειάζεται για την πρόσβαση στις υπηρεσίες είναι ένας περιηγητής ιστού. Επιπλέον δεν υπάρχει ούτε γεωγραφική δέσμευση αφού τα πάντα παρέχονται μέσω του διαδικτύου. Μάλιστα οι μεγάλοι πάροχοι έχουν χτίσει κέντρα δεδομένων σε πολλά διαφορετικά σημεία ανά τον κόσμο για να βελτιώσουν την ταχύτητα παροχής των υπηρεσιών τους.

2.5 Αρχιτεκτονική

Τα δομικά στοιχεία που αποτελούν το υπολογιστικό νέφος είναι μια front-end πλατφόρμα, back-end πλατφόρμες (εξυπηρετητές, βάσεις δεδομένων), η παρο-

χές που προσφέρεται και ένα δίκτυο (Internet, Intranet, Intercloud). Όλα αυτά δομούν την αρχιτεκτονική του υπολογιστικού νέφους [18].

2.5.1 Πελάτες-Πλατφόρμες Νέφους

Το front-end κομμάτι του νέφους αποτελείται από πλατφόρμες της οποίες ονομάζουμε πελάτες ή πελάτες νέφους. Αυτοί οι πελάτες αποτελούνται από εξυπηρετητές, μεμονωμένους υπολογιστές ή ολόκληρα δίκτυα, zero clients , συσκευές κινητής τηλεφωνίας και tablets . Αυτοί οι πελάτες επικοινωνούν με τη βάση δεδομένων του νέφους είτε μέσω κάποιας εφαρμογής (middleware), είτε κάποιου περιηγητή δικτύου είτε κάποιας εικονικής συνεδρίας.

2.5.2 Αποθήκη Νέφους

Είναι μια διαδικτυακή αποθήκη όπου αποθηκεύονται τα δεδομένα και είναι προσβάσιμη από πολλούς πελάτες. Υπάρχουν τρεις κύριες υλοποιήσεις, τα public cloud, private cloud, community cloudκαι συνδυασμοί τους γνωστοί και ως , hybrid cloud . Κάθε αποθήκη νέφους πρέπει να είναι agile, ευέλικτη, επεκτάσιμη, ασφαλής και να μπορεί να υποστηρίζει πολλούς πελάτες.

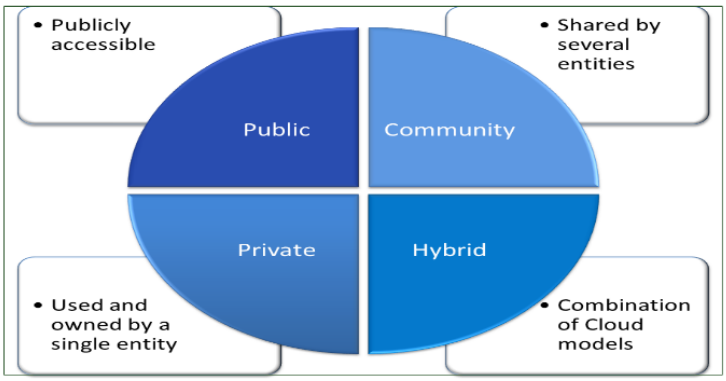
Public cloud

Τα Public cloudsανήκουν και τα διαχειρίζονται ανεξάρτητοι οργανισμοί/εταιρείες. Παρέχουν υπηρεσίες χαμηλού κόστους με μοντέλο Pay-As-You-Go . Όλοι οι πελάτες μοιράζονται τους ίδιους πόρους με περιορισμένες επιλογές προσαρμογής, περιορισμένη προστασία ασφαλείας και διαθεσιμότητα. Όλα αυτά τα διαχειρίζεται και τα διανείμει ο πάροχος. Ένα βασικό πλεονέκτημα του Public cloudείναι ότι είναι πολύ μεγαλύτερο από ένα ιδιωτικό νέφος και μπορεί να προσφέρει πολύ περισσότερες δυνατότητες επέκτασης στους πελάτες του.

Private cloud

Τα Private cloudsδημιουργούνται αποκλειστικά και μόνο έναν πελάατη. Στοχεύουν να κυρίως στο να λύσουν το θέμα της ασφάλειας και προσφέρουν μεγαλύτερο έλεγχο, κάτι στο οποίο στερείται το Public cloud. Υπάρχουν δυο είδη :

- On-premise Private Cloud: είναι τα εσωτερικά υπολογιστικά σύννεφα τα οποία φιλοξενούνται μέσα στο ίδιο το κέντρο δεδομένων του πελάτη. Το μοντέλο αυτό παρέχει πιο τυποποιημένη μορφή διαχείρισης και προστασίας αλλά είναι περιορισμένο όσον αναφορά την επέκταση. Επιπλέον πρέπει να διατίθεται κεφάλαιο για την συντήρηση των υλικών πόρων. Βρίσκει καλύτερη εφαρμογή σε περιπτώσεις όπου χρειάζεται πλήρης έλεγχος στην παραμετρικοποίηση της δομής του νέφους και στην ασφάλεια.



Σχήμα 2.3: Types Of Cloud

- Externally hosted Private Cloud: το είδος αυτό του ιδιωτικού δικτύου φιλοξενείται εξωτερικά του δικτύου του πελάτη μέσω ενός παρόχου νέφους, ο οποίος εγγυάται αποκλειστικότητα στο νέφος που παρέχει και ασφάλεια. Είναι ιδανικό για πελάτες που δεν θέλουν ή δεν μπορούν να διαθέσουν κεφάλαιο και χώρο για τη συντήρηση των υλικοτεχνικών δομών του νέφους.

Community cloud

Τα Community cloudsμοιράζουν την υποδομή τους σε οργανισμούς από κάποια συγκεκριμένη κοινότητα με κοινές προτεραιότητες (παραδείγματος χάριν θέματα ασφάλειας ή εντός της ίδιας δικαιοδοσίας).

Hybrid cloud

Το υβριδικό μοντέλο νέφους αποτελεί σύνθεση από δυο ή και περισσότερα από τα προηγούμενα είδη νέφους τα οποία παραμένουν ως ξεχωριστές οντότητες αλλά συνδέονται μεταξύ τους ώστε να παρέχουν τα πλεονεκτήματα των προηγούμενων μοντέλων.

2.5.3 Παροχές νέφους

Οι παροχές του υπολογιστικού σύννεφου μπορούν να ενταχθούν σε τρεις κατηγορίες :

Software as a Service (SaaS)

Σε αυτό το μοντέλο μια ολόκληρη εφραμογή παρέχεται στον πελάτη ως μια υπηρεσία. Η εφαρμογή τρέχει στο νέφος και πολλαπλοί πελάτες εξυπηρετούνται. Από την πλευρά του πελάτη δεν χρειάζεται κάποια επένδυση σε υλικό ή κάποια απαραίτητη τεχνογνωσία. Ενώ από την πλευρά του παρόχου απαιτείται η συντήρηση της εφαρμογής. Το μοντέλο αυτό είναι κλιμακούμενο και οι

Σχήμα 2.4: Cloud Services

διαχειριστές του συστήματος μπορούν να φορτώσουν το την εφαρμογή σε πολλούς εξυπηρετητές. Σήμερα κύριοι πάροχοι αυτών των υπηρεσιών είναι εταιρείες όπως οι Google, Salesforce, Microsoft, Zoho.

Platform as a Service (Paas)

Εδώ μια ολόκληρη πλατφόρμα, πάνω στην οποία μπορεί να χτιστεί κάποια άλλη εφαρμογή, προσφέρεται ως υπηρεσία. Ο πελάτης έχει την ελευθερία να χτίσει την εφαρμογή του η οποία τρέχει στην υλικοτεχνική υποδομή του παρόχου. Για να καλύψουν τις ανάγκες ητς διαχείρισης και κλιμάκωσης της εφαρμογής οι πάροχοι προσφέρουν προκαθορισμένους τύπους λειτουργικών και εξυπηρετητών όπως η πλατφόρμα LAMP (Linux, Apache, MySql, PHP). Δημοφιλείς υπηρεσίες Paas είναι η Google App Enginekαι η Microsoft Azure .

Infrastructure as a Service(IaaS)

Παρέχει την ικανότητα αποθήκευσης δεδομένων και υπολογιστικές υπηρεσίες ως τυποποιημένες υπηρεσίες πάνω στο δίκτυο. Διατίθενται πόροι όπως εξυπηρετητές, αποθηκευτικά συστήματα, εξοπλισμός δικτύου, κέντρα δεδομένων κ.λ.π. προκειμένου να διαχειρίζονται φόρτο εργασίας. Γνωστή υπηρεσία τέτοιου μοντέλου είναι η Amazon .

2.6 Πλεονεκτήματα και Μειονεκτήματα του Νέφους

Το υπολογιστικό νέφος συγκεντρώνει πολλά πλεονεκτήματα χάρη στα οποία αυξάνεται καθημερινά η ζήτηση των υπηρεσιών του. Μερικά από αυτά είναι :

- **Κλιμάκωση και Προσαρμοζόμενο Κόστος** Με το νέφος οι χρήστες έχουν ουσιαστικά πρόσβαση σε εικονικά άπειρους πόρους προκειμένου να καλύψουν τις ανάγκες τους. Η δέσμευση αυτών των πόρων γίνεται δυναμικά ανάλογα με τη ζήτηση που έχει η εφαρμογή κάθε χρήστη και το κόστος προσαρμόζεται με αυτή τη δέσμευση με ένα μοντέλο τύπου

Pay as you go . Έτσι ένας χρήστης που ξεκινά εκ του μηδενός μπορεί να χτίσει την εφαρμαγή του ουσιαστικά με μηδενικό κόστος και ανάλογα με την ανάπτυξή του να καλύπτει τις ανάγκες του και να πληρώνει αντίστοιχα.

- **Ευελιξία** Δεν υπάρχει περιορισμός στην χρήση του νέφους. Οι χρήστες μπορούν να το χρησιμοποιούν από οποιαδήποτε συσκευή, είτε μιλάμε για ένα ολόκληρο δίκτυο εξυπηρετητών είτε για ένα κινητό τηλέφωνο, και από οποιοδήποτε γεωγραφικό σημείο. Το υπολογιστικό σύννεφο μπορεί να εξυπηρετήσει από έναν απλό χρήστη που θέλει να κρατήσει αντίγραφο των φωτογραφιών του αποθηκευμένο στο διαδίκτυο μέχρι μια μεγάλη εταιρεία που δέχεται καθημερινά στην πλατφόρμα της εκατομμύρια αιτήματα συναλλαγών, όπως παραδείγματος χάριν η Amazon , και έχει ανάγκη από μεγάλη υπολογιστική ισχύ προκειμένου να τα ικανοποιήσει.

- **Αξιοπιστία** Υπάρχει μεγαλύτερη αξιοπιστία από την ύπαρξη τοπικών υλικοτεχνικών δομών , αφού ο πελάτης έχει πρόσβαση σε έναν τεράστιο αριθμό πόρων των οποίο μπορεί να αξιοποιήσει και σε περίπτωση αποτυχίας κάποιου εξυπηρετητή τα δεδομένα και οι υπηρεσίες του μπορούν άμεσα και με καθόλου κόστος να μεταφερθούν στον επόμενο.

- **Εύκολη Διαχείριση** Δεν χρειάζεται καμία τεχνογνωσία ή η εγκατάσταση επιπλέον λογισμικού ή υλικοτεχνικής υποδομής από πλευράς του πελάτη. Η διαχείριση των πόρων απλοποιείται μέσω κεντρικών διαχειριστικών μονάδων ενώ η δέσμευση και αποδέσμευση πόρων γίνεται δυναμικά.

Βέβαια δεν μπορούμε να μιλήσουμε για τα πλεονεκτήματα του νέφους χωρίς να αναφερθούμε στα μειονεκτήματά του.

- **Ασφάλεια και Ιδιωτικότητα** Η ασφάλεια είναι το μεγαλύτερο θέμα ανησυχίας όταν πρόκειται για το νέφος. Με το να συνεργάζεται με εξωτερικούς φορείς ο πελάτης δίνει ευαίσθητα δεδομένα είτε προς επεξεργασία είτε προς φύλαξη στο νέφος. Δεδομένα τα οποία πλέον είναι υπεύθυνος ο πάροχος να διαχειριστεί και να προφυλέξει.

- **Δύσκολη αλλαγή παρόχου** Η αλλαγή παρόχου είναι ιδιαίτερα δύσκολη κυρίως λόγω της δυσκολίας της μεταφοράς μεγάλου όγκου δεδομένων από τον ένα πάροχο στον άλλο. Οπότε ουσιαστικά ο πελάτης δεσμεύεται για συνεργασία μόνο με έναν πάροχο.

- **Μειωμένος έλεγχος** Αφού όλη η υλικοτεχνική δομή του νέφους και η διαχείρισή του γίνεται από τον πάροχο ο έλεγχος του πελάτη περιορίζεται μόνο στις εφαρμογές που τρέχουν πάνω από αυτό το επίπεδο. Οπότε οποιαδήποτε κύρια διαχειριστική ενέργεια, όπως πρόσβαση στον πυρήνα του εξυπηρετητή, αναβάθμιση, διαχείριση του firmware δεν επιτρέπονται.

Κεφάλαιο 3

NoSQL Databases

3.1 Εισαγωγή

Η NoSQL βάση δεδομένων παρέχει έναν μηχανισμό για την αποθήκευση και επεξεργασία δεδομένων που μοντελοποιούνται με διαφορετικό τρόπο από τον κλασσικό τρόπο με πίνακα, της σχεσιακής βάσης δεδομένων. Αυτού του είδους οι βάσεις δεδομένων παρόλο που είχαν σχεδιαστεί και υπήρχαν ήδη από τα τέλη του 1960 άρχισαν να γίνονται πιο δημοφιλείς στις αρχές του 21ου αιώνα όπου μεγάλες εταιρείες όπως η Facebook, Google και Amazon άρχισαν να τις χρησιμοποιούν. Χρησιμοποιούνται όλο και περισσότερο για τη διαχείριση Big Data και σε εφαρμογές πραγματικού χρόνου στο διαδίκτυο [13]. Η ανάπτυξη και οι απαιτήσεις των σύγχρονων εφαρμογών έχουν αλλάξει σε τέτοιο βαθμό όπου οι κλασσικές σχεσιακές βάσεις αδυνατούν να καλύψουν. Οι προγραμματιστές δημιουργούν εφαρμογές που παράγουν μεγάλους όγκων δεδομένων που συνεχώς μεταβάλλονται (δομημένα, ημιδομημένα, αδόμητα και πολυμορφικά δεδομένα). Πλέον δεν είναι αποδοτική η χρήση μοντέλων σχεδιασμού όπως αυτό του καταρράκτη. Απαιτείται να υπάρχει ευελιξία και γρήγορη προσαρμογή στις συνεχώς μεταβαλλόμενες ανάγκες μιας εφαρμογής. Επιπλέον οι σύγχρονες εφαρμογές πρέπει να είναι συνεχώς ενεργές και προσβάσιμες χωρίς περιορισμούς, έχοντας μια κλίμακα που να καλύπτει εκατομμύρια χρηστών. Οι οργανισμοί στρέφονται σε νέες τεχνολογίες για να καλύψουν τις ανάγκες τους όπως αυτή του υπολογιστικού νέφους αντί των τοπικών εξυπηρετητών και αποθηκών. Οι σχεσιακές βάσεις δεδομένων δεν σχεδιάστηκαν για να μπορούν να καλύψουν αυτές τις ανάγκες και να εκμεταλλεύονται τις νέες τεχνολογίες που έχουν αναπτυχθεί για να καλύπτουν την ζήτηση αυξημένου χώρου αποθήκευσης δεδομένων και υπολογιστικής ισχύς. Τα βασικά πλεονεκτήματα τα οποία βοήθησαν στην ανάπτυξη των NoSQL βάσεων είναι η απλότητα στον σχεδιασμό, η υψηλή απόδοση, το ότι μπορούμε να έχουμε εύκολα κλιμάκωση και το ότι δεν χρειαζόμαστε σχήμα. Επιπλέον η δομές δεδομένων στις NoSQL βάσεις είναι διαφορετικές από αυτή της σχεσιακής βάσης δεδομένων με αποτέλεσμα κάποιες λειτουργίες να γίνονται πιο γρήγορα.

3.2 Τύποι βάσεων NoSQL

Υπάρχουν τέσσερεις βασικοί τύποι βάσεων NoSQL .

- **Document databases** : ταιριάζει κάθε κλειδί με μια σύνθετη δεδομένων γνωστή και ως Document . Αυτή η δομή μπορεί να περιέχει πολλά διαφορετικά ζευγάρια κλειδιού-τιμής ή κλειδιού-πίνακα, μπορεί ακόμα να περιέχει και εμφωλευμένα άλλα Document .
- **Graph stores** : χρησιμοποιούνται για την αποθήκευση πληροφορίας για δίκτυα δεδομένων, όπως κοινωνικές δικτυώσεις. Γνωστές τέτοιες βάσεις είναι οι Neo4J και Giraph .
- **Key-value stores** : είναι οι πιο απλές NoSQL βάσεις. Κάθε όνομα αντικείμενο της βάσης αποθηκεύεται ως ένα «κλειδί» μαζί με την τιμή του. Γνωστές τέτοιες βάσεις είναι οι Riak και Berkeley DB .
- **Wide-column stores** : γνωστές τέτοιες βάσεις είναι οι Cassandra και HBase . Αυτού του είδους οι βάσεις είναι ιδανικές για ερωτήματα πάνω σε μεγάλου όγκου συστήματα δεδομένων και αποθηκεύουν στήλες δεδομένων μαζί αντί για σειρές.

Πίνακας 3.1: Χαρακτηριστικά των NoSql βάσεων δεδομένων [14]

Τύπος	**Επίδοση**	**Επεκτασιμότητα**	**Ελαστικότητα**	**Πολυπλοκότητα**	**Λειτουργικότητα**
Document databases	Υψηλή	Ποικίλει	Υψηλή	Χαμηλή	Ποικίλει
Graph stores	Ποικίλει	Ποικίλει	Υψηλή	Υψηλή	Θωρία Γράφων
Key-value stores	Υψηλή	Υψηλή	Υψηλή	Καμία	Ποικίλει
Wide-column stores	Υψηλή	Υψηλή	Μέτρια	Χαμηλή	Ελάχιστη

3.3 Διαφορές noSQL με SQL βάσεις δεδομένων

Παρακάτω δίνονται συνοπτικά οι διαφορές μεταξύ των noSQL και SQL βάσεων δεδομένων [15].

Πίνακας 3.2: NoSQL vs SQL βάσεις δεδομένων

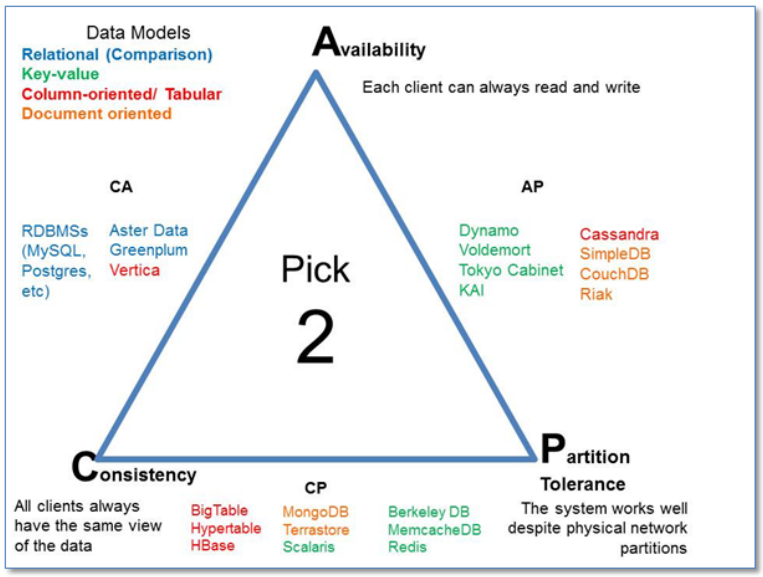
	SQL DATABASES	**NOSQL DATABASES**
Τύπος	Ένας μόνο τύπος με πολύ μικρές παραλλαγές	Πολλών διαφορετικών ειδών
Εξέλιξη	Αναπτύχθηε το 1970 για να λύσει το πρόβλημα αποθήκευσης δεδομένων που είχαν οι πρώτες εφαρμογές εκείνη την εποχή	Αναπτύχθηκε στις αρχές του 2000 για να αντιμετωπίσει τους περιορισμούς των βάσεωνSQL
Παραδείγματα	MySQL, Postgres, Microsoft SQL Server, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
Μοντέλο Αποθήκευσης δεδομένων	Ατομικές εγγραφές αποθηκεύονται σαν σειρές πίνακα όπου κάθε στήλη αντιστοιχεί σε ένα συγκεκριμένο πεδίο της εγγραφής	Ποικίλει ανάλογα με τον τύπο των δεδομένων. Για παράδειγμα tokey-valuemοντέλο μοιάζει με αυτό τωνSQLβάσεων αλλά έχει μόνο δυο στήλες με πιο περίπλοκα δεδομένα. Οι Documentβάσειες δεδομένων δεν χρησιμοποιούν καθόλου το μοντέλο πίνακα και αποθηκεύουν τα δεδομένα τους σε αρχεία τύπουXML, JSON
Σχήμα	Η δομή και οι τύποι δεδομένων φτιάχνονται από πριν. Για να αποθηεκυτεί κάποια νέα πληροφορία πρέπει να γίνει αλλαγή σε ολόκληρη την βάση. Και κατά τη διάρκεια αυτής της αλλαγής η βάση πρέπει να βγει εκτός δικτύου.	Δυναμικά με κανόνες επιβεβαίωσης δεδομένων. Οι εφαρμογές μπορούν να προσθέτουν νέα πεδία κατά την ώρα εκτέλεσης και σε αντίθεση με τις γραμμές στους πίνακες τωνSQLβάσεων μπορούμε να αποθηκεύσουμε μαζί διαφορετικών τύπων δεδομένα.

Κλιμάκωση	Πρέπει οι εξυπηρετητές να είναι αυξανόμενα πιο ισχυροί για να αντμετωπίσουν την αύξηση της ζήτησης. Είναι δυνατό να διαμοιραστεί μια βάση δεδομένων ανάμεσα σε πολλούς εξυπηρετητές αλλά θέλει αρκετές τροποποιήσεις και είναι εύκολο να χαθούν κύρια χαρακτηριστικά όπως τα JOINS	Γίνεται πολύ εύκολα, ο χρήστης αρκεί απλά να προσθέσει περισσότερους απλούς εξυπηρετητές ή με τη χρήση πόρων σε υπολογιστικό σύννεφο. Η βάση αυτόματα διαμοιράζει δεδομένα ανά τους εξυπηρετητές, όταν αυτό είναι απαραίτητο.
Μοντέλο Ανάπτυξης	Και ανοιχτού κώδικα και κλειστού	Ανοιχτού κώδικα
Υποστήριξη συναλλαγών	Ναι, οι αλλαγές μπορούν να ρυθμιστούν ώστε είτε να γίνονται συνολικά είτε καθόλου	Σε συγκεκριμένες περιπτώσεις και σε συγκεκριμένα επίπεδα.
Διαχείριση Δεδομένων	Έχει συγκεκριμένη γώσσα προγραμματισμού για να εκτελεί διάφορα αιτήματα, π.χ. SELECT fields FROM table WHERE...)	Μέσω διαπροσωπειών σε διάφορες γώσσες προγραμματισμού
Συνοχή	Μπορεί να ρυθμιστεί να έχει ισχυρή συνοχή	Εξαρτάται το προϊόν. Μερικές βάσεις προσφέρουν ισχυρή συνοχή (π.χ.MongoDB) ενώ άλλες προσφέρουν ενδεχόμενη συνοχή

3.4 Θεώρημα CAP

Σε ένα κατανεμημένο σύστημα η συνέπεια (Consistency), η διαθεσιμότητα (Availability) και η ανοχή καταμερισμών (Partition Tolerance) είναι ιδιαίτερα σημαντικά. Το 1998 ο Eric Brewer διατύπωσε το θεώρημα CAP το οποίο δηλώνει πως σε οποιοδήποτε κατανεμημένο σύστημα μπορούμε να έχουμε μόνο δυο από τις τρεις αρχές, δηλαδή είτε συνέπεια με ανοχή καταμερισμών, είτε διαθεσιμότητα με συνέπεια, είτε ανοχή καταμερισμών με διαθεσιμότητα κ.τ.λ. Πολλές βάσεις NoSQL προσπαθούν να δώσουν την δυνατότητα στον χρήστη να επιλέξει πώς να ρυθμίσουν την βάση ανάλογα με τις ανάγκες τους. Οι NoSQL βάσεις δεδομένων παρέχουν πολλές επιλογές ώστε ο χρήστης να μπορέσει να ρυθμίσει το σύστημά του με βάση κάποιες συγκεκριμένες προδιαγραφές. Σε αντίθεση με τις σχεσιακές βάσεις οι οποίες παρέχουν προκαθορισμένες ρυθμίσεις. Ένα παράδειγμα χαρακτηριστικού που προσφέρεται προκαθορισμένο στις σχεσιακές βάσεις δεδομένων είναι οι συναλλαγές. Οι περισσότερες NoSQL βάσεις

δεν παρέχουν κάποιο μηχανισμό στήριξης συναλλαγών, που σημαίνει ότι οι χρήστες πρέπει να υλοποιούν τις συναλλαγές από την αρχή. Για παράδειγμα πρέπει να σκέφτονται για κάθε εγγραφή αν είναι ασφαλής για την συναλλαγή, αν θα πρέπει να κατηγοριοποιηθεί ως απαραίτητο για την επιτυχία της συναλλαγής ή αν μπορεί να χαθεί. Πολλές φορές αυτά τα προβλήματα μπορούν να χρησιμοποιώντας εξωτερικούς διαχειριστές συναλλαγών όπως για παράδειγμα ο Apache ZooKeeper



Σχήμα 3.1: CAP Theorem

3.5 Το μοντέλο ACID

Το μοντέλο ACID είναι από τα παλιότερα και πιο σημαντικά κομμάτια της θεωρίας των βάσεων δεδομένων. Θέτει τέσσερεις ιδιότητες τις οποίες πρέπει να έχει κάθε βάση δεδομένων : την ατομικότητα (Atomicity), την συνέπεια (Consistency), την απομόνωση (Isolation) και την αντοχή (Durability). Καμία βάση που δεν έχει αυτά τα τέσσερα χαρακτηριστικά δεν μπορεί να θεωρηθεί αξιόπιστη [1].

Ατομικότητα Η ατομικότητα δηλώνει ότι οποιαδήποτε μεταβολή της βάσης δεδομένων πρέπει να ακολουθεί έναν κανόνα τύπου όλα ή τίποτα, δηλαδή είτε επιτυγχάνει πλήρως είτε δεν επιτυγχάνει καθόλου. Κάθε συναλλαγή πρέπει να είναι ατομική, δηλαδή αν κάποιο μέρος της συναλλαγής αποτύχει τότε αποτυγχάνει όλη η συναλλαγή.

Συνέπεια Η συνέπεια αναφέρεται στο ότι μόνο έγκυρα δεδομένα θα γραφτούν στη βάση. Αν για κάποιο λόγο εκτελεστεί κάποια συναλλαγή η οποία παρα-

βιάζει τον κανόνα συνέπειας της βάσης τότε όλη η συναλλαγή θα γυρίσει πίσω στην κατάσταση όπου δεν παραβίαζε την συνέπεια της βάσης. Από την άλλη όταν ολοκληρώνεται μια συναλλαγή επιτυχώς η βάση μεταβαίνει από τη μια κατάσταση συνέπειας στην άλλη με τα ανανεωμένα δεδομένα της συναλλαγής.

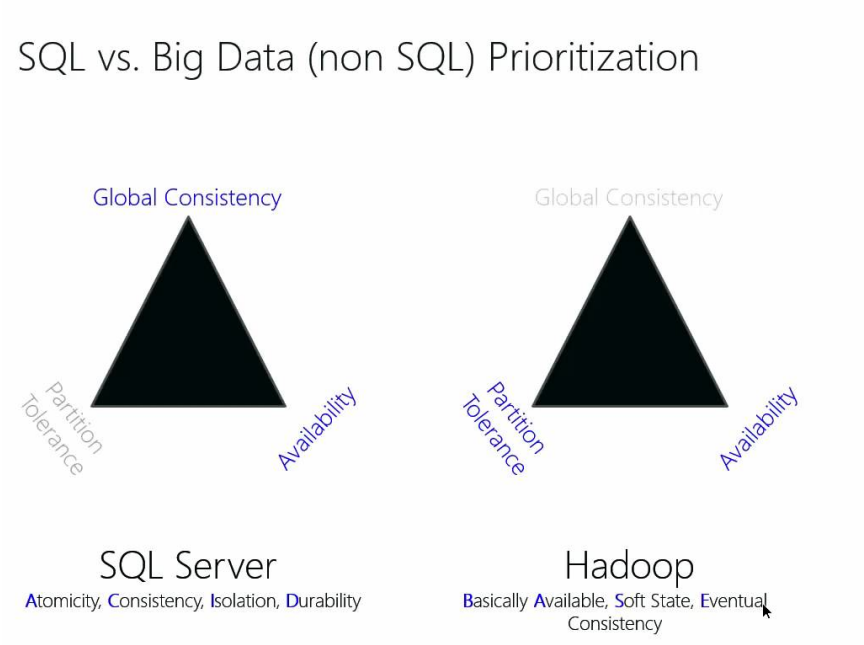
Απομόνωση Η απομόνωση απαιτεί ότι όταν πολλαπλές συναλλαγές προκύπτουν ταυτόχρονα να μην έχουν αντίκτυπο η μια στη εκτέλεση της άλλης. Για παράδειγμα, όταν ο χρήστης Α ξεκινήσει μια συναλλαγή ταυτόχρονα με τον χρήστη Β τότε οι δυο συναλλαγές πρέπει να εκτελεστούν απομονωμένα. Η βάση πρέπει είτε να εκτελέσει πρώτα την συναλλαγή του χρήστη Α πριν εκτελέσει αυτή του Β ή και αντίστροφα. Με αυτόν τον τρόπο αποτρέπεται το να διαβάσει η συναλλαγή του Α μη έγκυρα δεδομένα από την συναλλαγή του Β τα οποία στην πορεία της συναλλαγής μπορεί να αλλάξουν και να μην αποθηκευτούν στη βάση. Προσοχή στο γεγονός ότι η ιδιότητα της απομόνωσης μας εγγυάται μόνο ότι η μια συναλλαγή δεν θα εμπλακεί με την άλλη, δεν μας εγγυάται όμως τη σειρά με την οποία θα εκτελεστούν.

Αντοχή Η αντοχή μας εξασφαλίζει ότι οποιαδήποτε συναλλαγή που έχει αποθηκευτεί στη βάση δε θα χαθεί. Εξασφαλίζεται μέσω backup της βάσης και κράτησης αρχείων με τις συναλλαγές τα οποία διευκολύνουν την επαναφορά των αποθηκεμένων συναλλαγών παρά οποιασδήποτε αποτυχίας της βάσης.

3.6 Το μοντέλο BASE

Οι σχεσιακές βάσεις δεδομένων σχεδιάστηκαν με βάση το μοντέλο ACID . Όμως οι σύγχρονες NoSQL βάσεις δεδομένων λόγω της έλλειψης δομής που έχουν καθιστούν αυτό το μοντέλο πολύ δύσκολο να ακολουθηθεί. Αντίθετα οι αρχές του ACID καθυστερούν πολύ τις λειτουργίες των NoSQLβάσεων. Επομένως έχει αναπτυχθεί ένα πιο ευέλικτο μοντέλο που προσαρμόζεται στις ανάγκες των σύγχρονων βάσεων δεδομένων, το μοντέλο BASE. Οι αρχές αυτού του μοντέλου διευκολύνουν την ευελιξία των NoSQLβάσεων και την διαχείριση και επιμέλεια των αδόμητων δεδομένων τους. Οι αρχές του BASEείναι διαθεσιμότητα (Basic availability), soft state και Eventual Consistency [6] .

Διαθεσιμότητα Οι NoSQLβάσεις προσφέρουν διαθεσιμότητα δεδομένων ακόμα και ύστερα από πολλαπλές αποτυχίες. Αυτό το επιτυγχάνουν μέσω μιας κατανεμημένη προσέγγισης στη διαχείριση της βάσης. Αντί να διατηρούν μια μεγάλη αποθήκη δεδομένων και να συγκεντρώνονται στη διαχείριση και αντοχή σε αποτυχίες αυτής της αποθήκης, οι NoSQLβάσεις διανείμουν τα δεδομένα σε πολλά αποθηκευτικά συστήματα και κρατούν πολλά αντίγραφα. Σε περίπτωση λοιπόν που υπάρχει κάποια αποτυχία σε κάποιο



Σχήμα 3.2: SQL vs NoSQL

κομμάτι της βάσης αυτό δεν σημαίνει απαραίτητα και αποτυχία ολόκληρου του συστήματος.

Soft state Οι BASE βάσεις δεδομένων εγκαταλείπουν την ιδέα της συνέπειας των σχεσιακών βάσεων δεδομένων. Αυτή η ευθύνη μεταφέρεται από την βάση στον προγραμματιστή, ο οποίος είναι υπεύθυνος πλέον να αναπτύξει μεθόδους για τη διατήρηση της συνέπειας στα δεδομένα του.

Eventual Consistency Η μόνο προϋπόθεση που έχουν οι NoSQLβάσεις όσον αναφορά τη συνέπεια είναι ότι κάποια στιγμή τα δεδομένα θα φτάσουν σε ένα στάδιο συνέπειας. Αλλά δεν υπάρχουν εγγυήσεις για το πότε θα συμβεί αυτό. Αυτό αποκλίνει εντελώς από την άμεση συνέπεια που απαιτούσε το μοντέλο ACID , όπου απαγόρευε την εκτέλεση οποιασδήποτε άλλης συναλλαγής μέχρι να ολοκληρωθεί η προηγούμενη.

3.7 Snapshot Isolation

Στις βάσεις δεδομένων και τη διαχείριση συναλλαγών το μοντέλο του Snapshot Isolation εγγυάται ότι όλες οι αναγνώσεις που γίνονται κατά τη διάρκεια της συναλλαγής θα δουν ένα συνεπές Snapshot της βάσης η συναλλαγή θα ολοκληρωθεί επιτυχώς μόνο αν δεν έχουν γίνει αναβαθμίσεις οι οποίες έρχονται σε σύγκρουση με άλλες αναβαθμίσεις που έγιναν από τη στιγμή που δημιουργήθηκε το Snapshot [;]. Το μοντέλο του Snapshot Isolation έχει υιοθετηθεί

από πολλές μεγάλες βάσεις δεδομένων, μεταξύ των οποίων είναι οι SQL Anywhere, InterBase, Firebird, Oracle, PostgreSQL, MongoDB και Microsoft SQL Server . Ο λόγος που προτιμάται είναι ότι επιτρέπει καλύτερη απόδοση από την σειριοποίηση αποφεύγοντας αρκετά από τα προβλήματα που αποφεύγει και η σειριοποίηση. Ουσιαστικά χρησιμοποιείται το μοντέλου του multiversion concurrency control (MVCC) όπου κρατούνται αντίγραφα των τιμών κάθε αντικειμένου. Μέσω του MVCC επιτυγχάνουμε την συνέπεια και βελτιώνουμε την απόδοση του συστήματος δημιουργώντας μια νέα εκδοχή του αντικειμένου κάθε φορά που γίνεται σε αυτό μια εγγραφή και επιτρέποντας στις αναγνώσεις των συναλλαγών να γίνονται πάνω σε αρκετές εκδοχές του αντικειμένου. Στο Snapshot Isolation μια συναλλαγή φαίνεται να λειτουργεί με ένα Snapshot της βάσης δεδομένων, το οποίο περιέχει τα δεδομένα της βάσης όπως ήταν στην αρχή της συναλλαγής. Όταν ολοκληρωθεί η συναλλαγή θα αποθηκεύσει επιτυχώς τα δεδομένα στη βάση μόνο όταν τα δεδομένα τα οποία αλλάζουν δεν είχαν αναβαθμιστεί εξωτερικά αφότου δημιουργήθηκε το Snapshot . Σε περίπτωση τέτοιας προσπάθειας για διπλή εγγραφή η συναλλαγή τερματίζει ανεπιτυχώς. Ένα από τα προβλήματα που παρουσιάζει το μοντέλο του Snapshot Isolation είναι οι ταυτόχρονες αλλαγές δεδομένων. Δηλαδή, έστω ότι εκτελούνται δυο συναλλαγές οι T1 και T2, οι οποίες διαβάζουν ταυτόχρονα αλληλεπικαλυπτόμενα σύνολα δεδομένων τα A1 και A2. Στη συνέχεια ταυτόχρονα αλλάζουν τις τιμές τους σε B1 και B2 αντίστοιχα και τερματίζουν χωρίς η μια να έχει δει την ενημέρωση της άλλης. Αν οι συναλλαγές εκτελούνταν σε ένα σειριοποιήσιμο σύστημα δεν θα υπήρχε πρόβλημα αφού πρώτα θα τερμάτιζε η μια και μετά θα εκτελείτο η άλλη. Όμως, στο Snapshot Isolation αυτού του είδους το πρόβλημα δεν ελέγχεται αφού οι ενημερώσεις γίνονται ταυτόχρονα.

Ένα κλασικό παράδειγμα αυτού του προβλήματος είναι αυτό με τον τραπεζικό λογαριασμό. Έστω οι A1 και A2 είναι τα ποσά δυο λογαριασμών που ανήκουν στο ίδιο άτομο, τον X. Επιτρέπεται η ανάληψη από οποιονδήποτε λογαριασμό υπό την προϋπόθεση ότι το άθροισμα των νέων τιμών θα είναι θετικό ($B1 + B2 \geq 0$). Έστω ότι A1 = A2 = 100 μονάδες. Ο X ξεκινά ταυτόχρονα δυο συναλλαγές, τις T1 και T2, όπου στην T1 κάνει ανάληψη 200 μονάδων από τον πρώτο λογαριασμό και στην T2 κάνει πάλι ανάληψη άλλων 200 μονάδων από τον δεύτερο. Αν η βάση δεδομένων χρησιμοποιούσε σειριοποίηση, θα γινόταν πρώτα η T1, από το συνολικό ποσό θα αφαιρούνταν οι 200 μονάδες και ο συνολικός λογαριασμός θα έμενε με μηδέν διαθέσιμες μονάδες. Έτσι όταν ερχόταν η σειρά της T2 θα αποτύγχανε λόγω του ελέγχου ότι $B1 + B2 \geq 0$. Με το Snapshot Isolation οι T1 και T2 εκτελούνται σε ξεχωριστά Snapshot της βάσης δεδομένων. Κάθε συναλλαγή αφαιρεί 200 μονάδες από τον αντίστοιχο λογαριασμό και επιβεβαιώνει ότι το σύνολο είναι θετικό χρησιμοποιώντας την τιμή που είχε ο λογαριασμός όταν δημιουργήθηκε το Snapshot . Επιπλέον αφού καμία από τις αναβαθμίσεις δεν συγκρούεται με την άλλη και οι δυο ολοκληρώνονται με επιτυχία αφήνοντας τους λογαριασμούς με B1 = B2 = -100 μονάδες και συνολικό ποσό -200 μονάδες.

Αν βασιστεί στον multiversion concurrency control (MVCC) το Snapshot

Isolation επιτρέπει στις συναλλαγές να εκτελούνται χωρίς κάποια ανησυχία για τις ταυτόχρονες διεργασίες και χωρίς να χρειάζεται να επιβεβαιώνει ξανά όλες τις ενέργειες ανάγνωσης μετά το πέρας τις συναλλαγής. Η μόνη πληροφορία που χρειάζεται να αποθηκεύεται είναι μια λίστα με τις αλλαγές που έγιναν κατά τη διάρκεια της συναλλαγής ώστε στο τέλος να ελέγχεται για τυχόν συγκρούσεις πριν το πέρας της συναλλαγής.

Μέρος II

Περιγραφή Μοντέλου Διπλωματικής Εργασίας

Κεφάλαιο 4

Σχεδίαση Μοντέλου

4.1 Ανάλυση Προβλήματος

Η διπλωματική αυτή εργασία αποτελεί μέρος ενός ευρύτερου έργου το οποίο θα παραδοθεί ως υπηρεσία υπολογιστικού σύννεφου και έχει ως στόχο να παρέχει μια ενιαία πλατφόρμα στους χρήστες για συναλλαγές με βάσεις δεδομένων ανεξάρτητα από το είδος της βάση δεδομένων που χρησιμοποιούν. Η αρχιτεκτονική είναι του μοντέλου πελάτη-εξυπηρετητή. Με τους πελάτες να στέλνουν δεδομένα στους εξυπηρετητές και εκείνοι να τα διαχειρίζονται, να κάνουν τις απαραίτητες μετατροπές ανάλογα με τη βάση δεδομένων και να εκτελούν τις αντίστοιχες λειτουργίες.

Μέχρι στιγμής το middleware αυτό είχε υλοποιηθεί με έναν εξυπηρετητή να δέχεται όλα τα αιτήματα από τους πελάτες. Το πρόβλημα με αυτόν τον σχεδιασμό ήταν ότι το σύστημα ήταν πολύ εύκολο να αποτύχει σε περίπτωση που αποτύγχανε ο εξυπηρετητής ενώ δεν ήταν δυνατή η εξυπηρέτηση μεγάλου όγκου πελατών. Όσο πιο πολύ αύξανε ο όγκος των πελατών τόσο πιο δύσκολο ήταν για τον εξυπηρετητή να διαχειριστεί τα δεδομένα.

4.2 Περιγραφή Μοντέλου Διπλωματικής Εργασίας

Προκειμένου να λυθεί το πρόβλημα που αναφέρθηκε στο προηγούμενο κεφάλαιο χρησιμοποιήθηκε ένα μοντέλο σαν και αυτό του master-worker . Στο δίκτυο υπάρχει ένας master ο οποίος μοιράζει το φορτίο στους εξυπηρετητές ανάλογα με κάποιο από τα εξής κριτήρια :

- Συνολικός χρόνος λειτουργίας του εξυπηρετητή: δίνεται προτεραιότητα στους εξυπηρετητές με τον μικρότερο συνολικό χρόνο λειτουργίας
- Μέσος Χρόνος Λειτουργίας : υπολογίζεται ένας μέσος χρόνος λειτουργίας ως το άθροισμα των χρόνων εξυπηρέτησης του κάθε αιτήματος προς

το σύνολο αυτών των αιτημάτων και δίνεται προτεραιότητα στον εξυπηρετητή με τον μικρότερο μέσο χρόνο.

- Αριθμός πελατών που ήδη έχουν ανατεθεί στον εξυπηρετητή: δίνεται προτεραιότητα στον εξυπηρετητή με τους λιγότερους πελάτες
- Τυχαία επιλογή

Ο κάθε πελάτης επικοινωνεί αρχικά με τον master προκειμένου να του ανατεθεί ένας εξυπηρετητής. Ο master επιλέγει τον εξυπηρετητή και του αναθέτει τον πελάτη. Στη συνέχεια όλη η επικοινωνία γίνεται μεταξύ πελάτη-εξυπηρετητή και μόλις τελειώσουν όλες οι διεργασίες ο πελάτης στέλνει ένα μήνυμα στον master για να τον ειδοποιήσει ότι ολοκληρώθηκε η συνεδρία.

Η επικοινωνία μεταξύ master και εξυπηρετητή γίνεται μέσω ενός εργαλείου για τον συγχρονισμό , τον Apache ZooKeeper [1] .Όποτε σηκώνεται ένας εξυπηρετητής επικοινωνεί με τον ZooKeeper , καταγράφει σε αυτόν τα στοιχεία του (ταυτότητα και διεύθυνση IP) και αρχικοποιεί τις μετρήσεις βάση των οποίων γίνεται η επιλογή του από τον master για κάποιο πελάτη (ολικός χρόνος λειτουργίας, μέσος χρόνος λειτουργίας και αριθμός πελατών). Ο ZooKeeper ειδοποιεί τον master για τη δημιουργία του νέου εξυπηρετητή. Επιπλέον, όταν αλλάζει κάποια από τις παραπάνω μεταβλητές ο εξυπηρετητής καταγράφει την νέα της τιμή στον ZooKeeper και εκείνος με τη σειρά του ενημερώνει πάλι τον master για την αλλαγή. Τέλος όταν κλείσει ένας εξυπηρετητής διαγράφονται τα στοιχεία του από τον ZooKeeper και ειδοποιείται ο master .

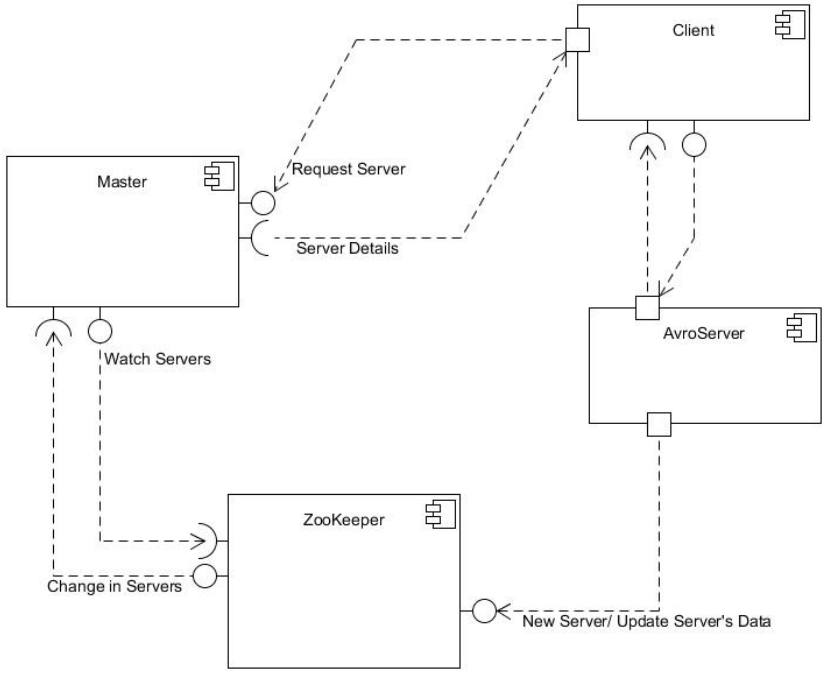
Από τη στιγμή που ο master αναθέσει στον πελάτη έναν εξυπηρετητή αποδεσμεύεται και εμπλέκεται πάλι στο τέλος, όταν ο πελάτης του στείλει ειδοποίηση για το κλείσιμο της σύνδεσης, προκειμένου να αλλάξει κάποια στοιχεία στον ZooKeeper. Η επικοινωνία μεταξύ πελάτη και εξυπηρετητή είναι αυτόνομη. Ο πελάτης στέλνει αιτήματα στον εξυπηρετητή για την βάση, ο εξυπηρετητής τα επεξεργάζεται και απαντά πίσω αν το αίτημα ήταν επιτυχές ή όχι. Σε περίπτωση αποτυχίας από την πλευρά του εξυπηρετητή ενημερώνεται ο πελάτης ενώ σε περίπτωση αποτυχίας από πλευράς του πελάτη ο εξυπηρετητής αφήνει ένα χρονικό περιθώριο για το οποίο η σύνδεση μπορεί να είναι ανενεργή, αν περάσει αυτό το όριο ο εξυπηρετητής καταργεί την σύνδεση και ο πελάτης θα πρέπει να ξαναδημιουργήσει νέα μιλώντας πάλι με τον master για να ξαναπάρει καινούριο εξυπηρετητή.

4.3 Δομικά Στοιχεία

4.3.1 Master

Ο master είναι υπεύθυνος να ρυθμίζει το φορτίο στο δίκτυο. Έχει τη δική του μοναδική ταυτότητα (*masterId*) και διατηρεί έναν ComcurentSkipListMap ,

[1] Περιγραφή του γίνεται σε επόμενο κεφάλαιο



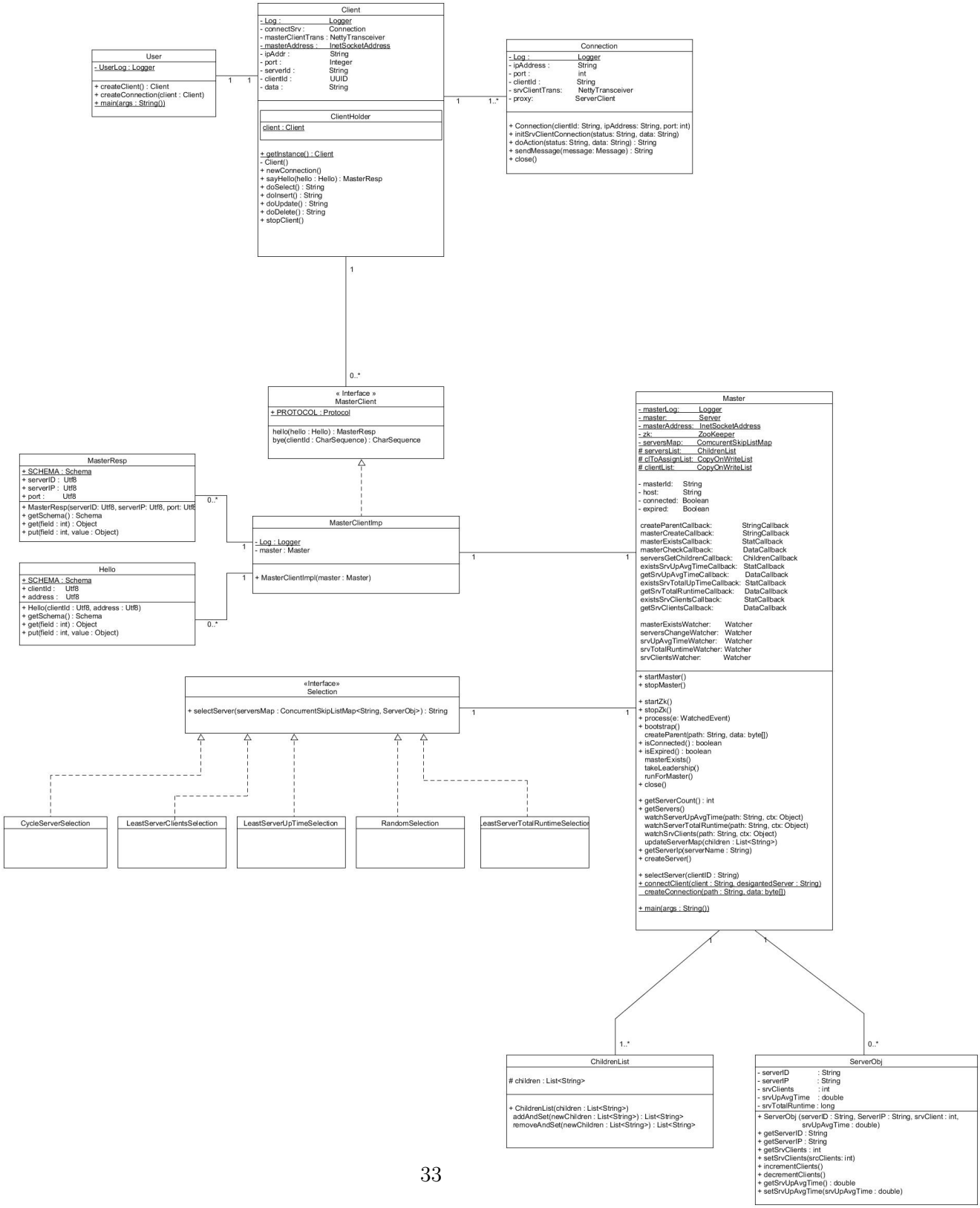
Σχήμα 4.1: Component Diagram

τον *serversMap* , όπου κρατάει στοιχεία για όλους τους ενεργούς εξυπηρετητές του δικτύου και μια CopyOnWriteList , την *clientList* , που διατηρεί τα στοιχεία που του στέλνουν οι πελάτες.

Κάθε φορά που έρχεται ένας νέος πελάτης επικοινωνεί με τον master μέσω πρωτοκόλλου Avro MasterClient. Δέχεται μέσω της μεθόδου *hello(Hello record)* , η οποία υλοποιείται στην MasterClientImpl , τα στοιχεία του πελάτη, τα αποθηκεύει στην *clientList* και επιλέγει τον καταλληλότερο εξυπηρετητή για να αναθέσει. Η επιλογή γίνεται μέσω της μεθόδου *selectServer(serversMap : ConcurrentSkipListMap* $< String, ServerObj >$*)* είτε τυχαία, είτε κοιτώντας τον συνολικό χρόνο που είναι ενεργός ο εξυπηρετητής, είτε κοιτώντας τον μέσο χρόνο λειτουργίας του είτε με βάση τον αριθμό των πελατών που ήδη εξυπηρετεί. Σε περίπτωση που δεν υπάρχει κάποιος ενεργός εξυπηρετητής στο δίκτυο όταν έρθει κάποιος πελάτης ο master τον δημιουργεί.

Επιπλέον ο master επικοινωνεί με τον ZooKeeper για να ενημερώνεται για τις αλλαγές που γίνονται στο δίκτυο από την πλευρά τον εξυπηρετητών. Δηλαδή, αν δημιουργήθηκε κάποιος νέος εξυπηρετητής, αν έκλεισε κάποιος παλιός ή αν υπήρξε κάποια αλλαγή στις μεταβλητές του εξυπηρετητή τις οποίες χρησιμοποιεί ως κριτήριο επιλογής. Σε οποιαδήποτε αλλαγή της κατάστασης του δικτύου ενημερώνει τον *serversMap* με τις νέες πληροφορίες.

Σχήμα 4.2: Master-Client Class Diagram



Πίνακας 4.1: Μεταβλητές του master

Μεταβλητή	**Τύπος**	**Περιγραφή**
masterLog	Logger	Αρχείο όπου γράφονται τα σχόλια κατά την εκτέλεση του master
master	Server	Εξυπηρετητής Avro στον οποίο συνδέεται ο πελάτης
masterAddress	InetSocketAddress	Διεύθυνση IP του master
zk	ZooKeeper	Αντικείμενο την συνεδρία με τον ZooKeeper
serversMap	ComcurentSkipListMap	Αποθηκεύονται τα στοιχεία των εξυπηρετητών του δικτύου
serversList	ChildrenList	Βοηθητική λίστα όπου κρατούνται μόνο οι ταυτότητες των εξυπηρετητών του δικτύου
clToAssignList	CopyOnWriteList	Βοηθητική λίστα όπου κρατούνται οι πελάτες που δεν έχουν ανατεθεί ακόμα σε εξυπηρετητή
clientList	CopyOnWriteList	Βοηθητική λίστα όπου κρατούνται οι όλοι οι πελάτες μέχρι να στείλουν το τελικό μήνυμα αποσύνδεσης
masterId	String	Η μοναδική ταυτότητα του master
host	String	Η διεύθυνση για την επικοινωνία με τον ZooKeeper
connected	Boolean	Βοηθητική μεταβλητή για το αν έχει ολοκληρωθεί η σύνδεση με τον ZooKeeper
expired	Boolean	Βοηθητική μεταβλητή για το αν έχει λήξει η σύνδεση με τον ZooKeeper
createParentCallback	StringCallback	Callback για τη δημιουργία των κόμβων στον ZooKeeper

masterCreateCallback	StringCallback	Callback για τη δημιουργία του κλειδώματος για τον master στον ZooKeeper
masterExistsCallback	StatCallback	Callback για την ύπαρξη του κόμβου /master στον ZooKeeper
masterCheckCallback	DataCallback	Callback για τον έλεγχο ύπαρξης του κόμβου /master στον ZooKeeper
serversGetChildrenCallback	ChildrenCallback	Callback για να πάρει ο master τα στοιχεία του εξυπηρετητή (ταυτότητα και IP) από τον ZooKeeper
existsSrvUpAvgTimeCallback	StatCallback	Callback για τον έλεγχο ύπαρξης του κόμβου που κρατάει την πληροφορία για τον μέσο χρόνο λειτουργίας του εξυπηρετητή στον ZooKeeper
getSrvUpAvgTimeCallback	DataCallback	Callback να πάρει ο master την πληροφορία για τον μέσο χρόνο λειτουργίας του εξυπηρετητή από τον ZooKeeper
existsSrvTotalUpTimeCallback	StatCallback	Callback για τον έλεγχο ύπαρξης του κόμβου που κρατάει την πληροφορία για τον ολικό χρόνο λειτουργίας του εξυπηρετητή στον ZooKeeper
getSrvTotalRuntimeCallback	DataCallback	Callback να πάρει ο master την πληροφορία για τον ολικό χρόνο λειτουργίας του εξυπηρετητή από τον ZooKeeper

existsSrvClientsCallback	StatCallback	Callback για τον έλεγχο ύπαρξης του κόμβου που κρατάει την πληροφορία για τον αριθμό πελατών του εξυπηρετητή στον ZooKeeper
getSrvClientsCallback	DataCallback	Callback να πάρει ο master την πληροφορία για τον αριθμό πελατών του εξυπηρετητή από τον ZooKeeper
masterExistsWatcher	Watcher	Watcher που ενημερώνει τους υποψήφιους masters για αλλαγές στον κόμβο του /master του ZooKeeper
serversChangeWatcher	Watcher	Watcher που ενημερώνει τον master για αλλαγές στον κόμβο των εξυπηρετητών του ZooKeeper
srvUpAvgTimeWatcher	Watcher	Watcher που ενημερώνει τον master για αλλαγές στον κόμβο του μέσου χρόνου λειτουργίας του εξυπηρετητή
srvTotalRuntimeWatcher	Watcher	Watcher που ενημερώνει τον master για αλλαγές στον κόμβο του ολικού χρόνου λειτουργίας του εξυπηρετητή
srvClientsWatcher	Watcher	Watcher που ενημερώνει τον master για αλλαγές στον κόμβο του αριθμού των πελατών του εξυπηρετητή

Πίνακας 4.2: Μέθοδοι του master

Μέθοδοι	**Περιγραφή**
startMaster()	Ξεκινά το Avro κομμάτι του master

stopMaster()	Σταματά το Avro κομμάτι του master
startZk()	Ξεκινά το ZooKeeper κομμάτι του master
stopZk()	Σταματά το ZooKeeper κομμάτι του master
process(e WatchedEvent)	Δημιουργεί την σύνδεση με τον ZooKeeper
bootstrap()	Ξεκινά την διαδικασία για τη δημιουργία των κόμβων στον ZooKeeper
createParent(path String, data byte[])	Δημιουργεί τους κόμβους στον ZooKeeper
isConnected() Boolean	Ελέγχει αν ο master είναι συνδεδεμένος με τον ZooKeeper
isExpired() Boolean	Ελέγχει αν η σύνδεση μεταξύ του master και του ZooKeeper έχει λήξει
masterExists()	Ελέγχει αν υπάρχει το κλείδωμα για τον master στον ZooKeeper
takeLeadership()	Ξεκινά τη διαδικασία για να πάρει τα στοιχεία των εξυπηρετητών από τον ZooKeeper
runForMaster()	Δημιουργεί το κλείδωμα στον ZooKeeper για τη θέση του master
close()	Κλείνει τον master
getServerCount() int	Επιστρέφει τον αριθμό των εξυπηρετητών που έχει αποθηκευμένους ο master
getServers()	Παίρνει τα στοιχεία των εξυπηρετητών από τον ZooKeeper
watchServerUpAvgTime(path String, ctx Object)	Δημιουργεί τον Watcher για τον κόμβο του ZooKeeper που κρατά τον μέσο χρόνο λειτουργίας του εξυπηρετητή
watchServerTotalRuntime(path String, ctx Object)	Δημιουργεί τον Watcher για τον κόμβο του ZooKeeper που κρατά τον ολικό χρόνο λειτουργίας του εξυπηρετητή
watchSrvClients(path String, ctx Object)	Δημιουργεί τον Watcher για τον κόμβο του ZooKeeper που κρατά τον αριθμό πελατών του εξυπηρετητή
updateServerMap(children List $< String >$)	Ανανεώνει τον χάρτη με τους εξυπηρετητές ανάλογα με τα καινούρια στοιχεία που πήρε από τον ZooKeeper
getServerIp(serverName String)	Παίρνει και αποθηκεύει την διεύθυνση IP του εξυπηρετητή από τον ZooKeeper
createServer()	Δημιουργεί έναν καινούριο εξυπηρετητή
selectServer(clientID String)	Επιλέγει τον καταλληλότερο εξυπηρετητή για να αναθέσει σε έναν πελάτη
connectClient(client String, desigantedServer String)	Ξεκινά τη διαδικασία για να ενημερώσει τον ZooKeeper για τη νέα σύνδεση μεταξύ πελάτη και εξυπηρετητή
createConnection(path String, data byte[])	Ενημερώνει τον ZooKeeper για τη νέα σύνδεση μεταξύ πελάτη και εξυπηρετητή
main(args String())	

Πίνακας 4.3: Μέθοδοι της κλάσης MasterClientImpl

Μέθοδοι	**Return Value**	**Περιγραφή**
MasterClientImpl(master : Master)		Κατασκευαστής που αρχικοποιεί το αντικείμενο του master στην κλάση
hello(hello : Hello)	MasterResp	Η μέθοδος που καλεί ο πελάτης για την αρχικοποίηση της σύνδεσης με τον master . Δέχεται ένα αντικείμενο τύπου Hello με τα στοιχεία του πελάτη και γυρίζει ένα αντικείμενο τύπου MasterResp με τα στοιχεία του εξυπηρετητή που επιλέχτηκε
bye(clientId : CharSequence)	CharSequence	Η μέθοδος που καλεί ο πελάτης αφού ολοκληρωθεί η επικοινωνία του με τον εξυπηρετητή για να ενημερώσει τον master . Αφαιρεί τα στοιχεία του πελάτη από την αντίστοιχη λίστα που κρατάει ο master .

Πίνακας 4.4: Selection

Interface: Selection		
Μέθοδος	**Return Value**	**Περιγραφή**
selectServer(serversMap: ConcurrentSkipListMap< $String, ServerObj >$)	String	Δέχεται ως όρισμα τον χάρτη με τους διαθέσιμους εξπηρετητές του master και ανάλογα με την υλοποίηση γυρνάει τον καταλληλότερο εξυπηρετητή για να συνδεθεί ο πελάτης
	Implementations	**Περιγραφή**
	RandomSelection	Επιλέγει τυχαία έναν εξυπηρετητή από τον χάρτη

LeastServerUpTimeSelection	Επιλέγει τον εξυπηρετητή με τον λιγότερο μέσο χρόνο λειτουργίας
LeastServerTotalRuntimeSelection	Επιλέγει τον εξυπηρετητή με τον λιγότερο ολικό χρόνο λειτουργίας
LeastServerClientsSelection	Επιλέγει τον εξυπηρετητή με τον λιγότερο αριθμό πελατών
CycleServerSelection	Επιλέγει κυκλικά όλους τους εξυπηρετητές της λίστας

Πίνακας 4.5: Το αντικείμενο ServerObj

Μεταβλητή	**Τύπος**	**Περιγραφή**
serverID	String	Η ταυτότητα του εξυπηρετητή
serverIP	String	Η διεύθυνση του εξυπηρετητή
srvClients	int	Ο αριθμός των πελατών του εξυπηρετητή
srvUpAvgTime	double	Ο μέσος χρόνος λειτουργίας του εξυπηρετητή
srvTotalRuntime	long	Ο ολικός χρόνος λειτουργίας του εξυπηρετητή
Μέθοδος	**Return Value**	**Περιγραφή**
getServerID	String	Επιστρέφει την ταυτότητα του εξυπηρετητή
getServerIP	String	Επιστρέφει την διεύθυνση του εξυπηρετητή
getSrvClients	int	Επιστρέφει τον αριθμό των πελατών του εξυπηρετητή
setSrvClients(srvClients:int)		Αλλάζει την τιμή των πελατών του εξυπηρετητή σε srvClients

incrementClients()		Αυξάνει κατά ένα την τιμή των πελατών του εξυπηρετητή
decrementClients()		Μειώνει κατά ένα την τιμή των πελατών του εξυπηρετητή
getSrvUpAvgTime()	double	Επιστρέφει τον μέσο χρόνο λειτουργίας του εξυπηρετητή
setSrvUpAvgTime(srvUpAvgTime:double)		Αλλάζει τον μέσο χρόνο λειτουργίας του εξυπηρετητή σε srvUpAvgTime

4.3.2 Εξυπηρετητής

Ο εξυπηρετητής είναι υπεύθυνος να διαχειρίζεται τα αιτήματα προς τη βάση που έρχονται από τους πελάτες. Επιπλέον συνδέεται με τον ZooKeeper , χρησιμοποιώντας τον ως ενδιάμεσο για την επικοινωνία του με τον Master . Διατηρεί έναν ConcurrentHashMap τον *clientsMap* όπου κρατάει τα στοιχεία για τους πελάτες που εξυπηρετεί και δυο Schedulers έναν για να υπολογίζει τον μέσο χρόνο λειτουργίας του και έναν για να βρίσκει και να αφαιρεί από τον *clientsMap* πελάτες των οποίων η σύνδεση έχει περάσει τον επιτρεπτό χρόνο για τον οποίο ήταν ανενεργή.

Με τους πελάτες επικοινωνεί μέσω του Avro πρωτοκόλλου *ServerClient* και η διαχείριση των μηνυμάτων γίνεται μέσω της κλάσης *ServerClientImp* η οποία διατηρεί και μια λίστα με αντικείμενα τα οποία κρατούν την πληροφορία για τον τύπο του αιτήματος και πόσο χρόνο έκανε για να ολοκληρωθεί ώστε να μπορεί ο Scheduler του εξυπηρετητή να υπολογίζει τον μέσο χρόνο λειτουργίας ως το άθροισμα αυτών των χρόνων δια το πλήθος των αιτημάτων . Επιπλέον μέσω κατάλληλων μεθόδων αυτής της κλάσης ο εξυπηρετητής δέχεται τα αιτήματα από τους πελάτες, διαχωρίζει το είδος του αιτήματος και προχωράει αντίστοιχα στην εκτέλεσή του. Ένα αίτημα μπορεί να είναι τύπου :

- **INIT** : χρησιμοποιείται για να ξεχωρίσει τα αιτήματα που αρχικοποιούν την σύνδεση μεταξύ πελάτη και εξυπηρετητή. Για αυτού του τύπου τα αιτήματα ο εξυπηρετητής δημιουργεί ένα καινούριο αντικείμενο πελάτη με τα στοιχεία του, το οποίο αποθηκεύει στον *clientsMap* . Επιπλέον δημιουργεί ένα αντικείμενο για το συγκεκριμένο αίτημα με τον χρόνο που έκανε να ολοκληρωθεί και το αποθηκεύει στην αντίστοιχη λίστα.

- **SELECT** : χρησιμοποιείται για να ξεχωρίσει ο εξυπηρετητής τα αιτήματα προς τη βάση με τις εντολές τύπου select . Επιπλέον δημιουργεί ένα αντικείμενο για το συγκεκριμένο αίτημα με τον χρόνο που έκανε να ολοκληρωθεί και το αποθηκεύει στην αντίστοιχη λίστα.

- **INSERT** : χρησιμοποιείται για να ξεχωρίσει ο εξυπηρετητής τα αιτήματα προς τη βάση με τις εντολές τύπου insert . Επιπλέον δημιουργεί ένα αντικείμενο για το συγκεκριμένο αίτημα με τον χρόνο που έκανε να ολοκληρωθεί και το αποθηκεύει στην αντίστοιχη λίστα.

- **UPDATE** : χρησιμοποιείται για να ξεχωρίσει ο εξυπηρετητής τα αιτήματα προς τη βάση με τις εντολές τύπου update . Επιπλέον δημιουργεί ένα αντικείμενο για το συγκεκριμένο αίτημα με τον χρόνο που έκανε να ολοκληρωθεί και το αποθηκεύει στην αντίστοιχη λίστα.

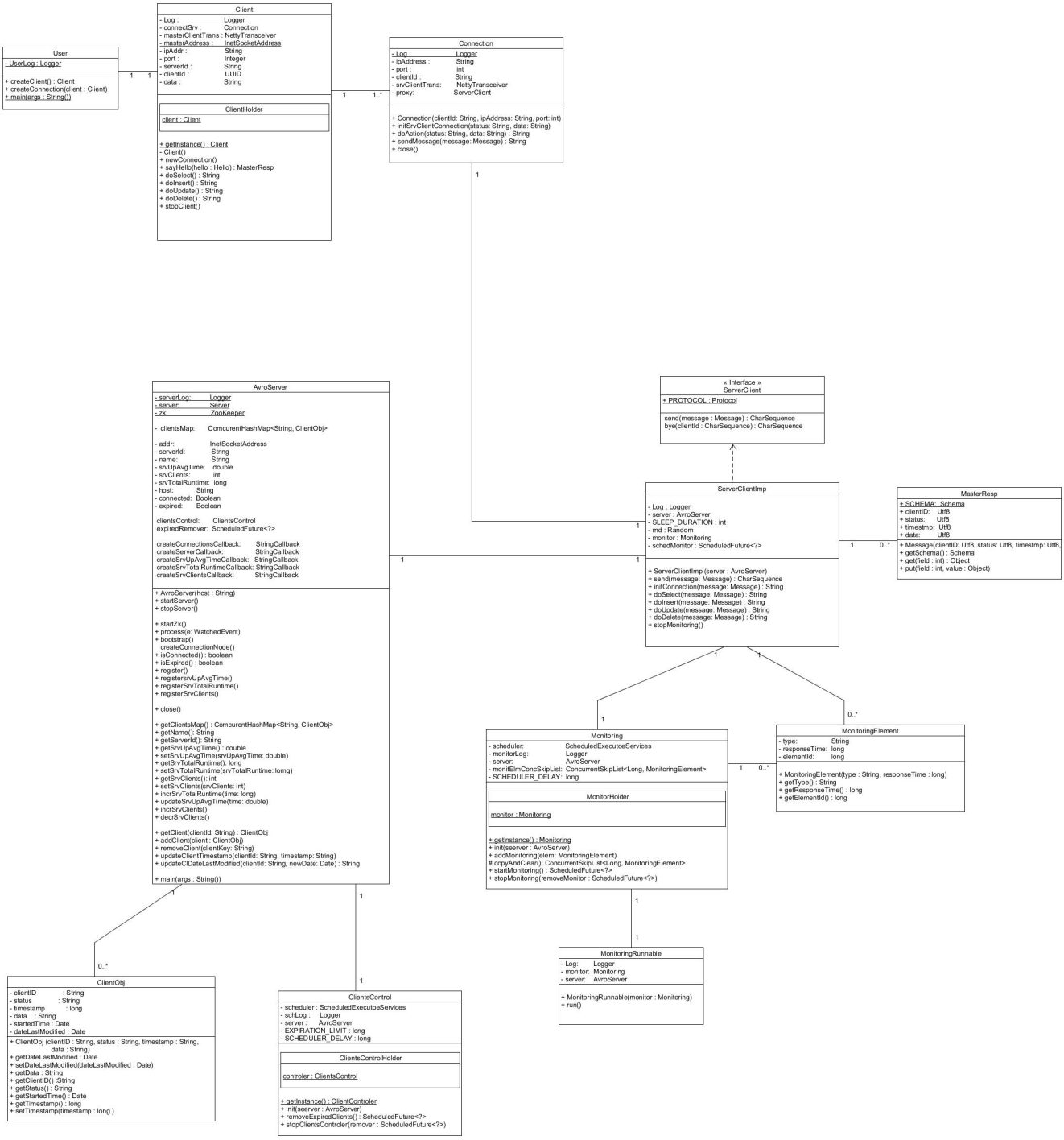
- **DELETE** : χρησιμοποιείται για να ξεχωρίσει ο εξυπηρετητής τα αιτήματα προς τη βάση με τις εντολές τύπου delete . Επιπλέον δημιουργεί ένα αντικείμενο για το συγκεκριμένο αίτημα με τον χρόνο που έκανε να ολοκληρωθεί και το αποθηκεύει στην αντίστοιχη λίστα.

Τέλος όταν διακοπεί η σύνδεση με τον πελάτη ο εξυπηρετητής υπολογίζει τον συνολικό χρόνο λειτουργίας του και αφαιρεί τον πελάτη από τον *clientsMap* ενώ ενημερώνει και τον ZooKeeper για τον νέο αριθμό πελατών και τον ολικό χρόνο λειτουργίας του μέχρι εκείνον τον πελάτη.

Με τον ZooKeeper επικοινωνεί για να γράφει τα στοιχεία που ενδιαφέρουν τον Master ώστε να επιλέγει τον καταλληλότερο κάθε φορά εξυπηρετητή για ανάθεση, δηλαδή γράφει τον αριθμό των πελατών που εξυπηρετεί και τον μέσο και ολικό χρόνο που παρέμενε ενεργός. Προτιμήθηκε να μεσολαβεί ο ZooKeeper αντί να υπάρχει απευθείας επικοινωνία με τον Master ώστε να διασφαλιστεί ο σωστός συγχρονισμός μεταξύ των δυο και να απλοποιηθεί η επικοινωνία, αφού το βάρος για τον έλεγχο του συγχρονισμού μεταφέρεται πλέον από τους Master και εξυπηρετητή στον ZooKeeper. Ο εξυπηρετητής μιλά με τον ZooKeeper σε τακτά χρονικά διαστήματα μέσω του Scheduler ώστε να αναβαθμίζει τον μέσο χρόνο λειτουργίας αλλά και όταν έρχεται ένας πελάτης για να αναβαθμίζει τον αριθμό των πελατών που εξυπηρετεί, τέλος επικοινωνεί όταν φεύγει ένας πελάτης για να μειώνει τον αριθμό των πελατών του και να ανανεώνει τον ολικό χρόνο λειτουργίας του.

Ο εξυπηρετητής διατηρεί δυο Schedulers , τους *ClientsControl* και *Monitoring* . Ο *ClientsControl* ελέγχει ανά τακτά χρονικά διαστήματα τον χρόνο σύνδεσης κάθε πελάτη για τον οποίο έχει μείνει ανενεργή. Αν αυτός ο χρόνος υπερβαίνει το επιτρεπτό διάστημα τότε αφαιρεί τον πελάτη από τη λίστα του εξυπηρετητή. Σε περίπτωση που κάποιος από τους πελάτες που διαγράφτηκαν επιχειρήσει να χρησιμοποιήσει πάλι την ίδια σύνδεση τότε δέχεται πίσω μήνυμα λάθους. Ο *Monitoring* παίρνει ανά τακτά χρονικά διαστήματα τα αντικείμενα της λίστας με τους χρόνους εκτέλεσης κάθε αιτήματος και υπολογίζει έναν μέσο χρόνο λειτουργίας του εξυπηρετητή τον οποίο γράφει στον ZooKeeper . Μόλις τελειώσει τον υπολογισμό του καθαρίζει τη λίστα.

Σχήμα 4.3: Server-Client Class Diagram



Πίνακας 4.6: Μεταβλητές του Εξυπηρετητή

Μεταβλητή	**Τύπος**	**Περιγραφή**
serverLog	Logger	Αρχείο όπου γράφει τα αποτελέσματα ο εξυπηρετητής
server	Server	Αντικείμενο τύπου Server για την σύνδεση με τον πελάτη
zk	ZooKeeper	Αντικείμενο για την σύνδεση με τον Zookeeper
clientsMap	ComcurentHashMap	Αντικείμενο όπου κρατόνται τα στοιχεία των πελατών
addr	InetSocketAddress	Η διεύθυνση του εξυπηρετητή
serverId	String	Η ταυτότητα του εξυπηρετητή
name	String	Η ταυτότητα του εξυπηρετητή με το πρόθεμα "server-" . Χρησιμοποιείται για να φτιάχνεται το path στους κόμβους του ZooKeeper
srvUpAvgTime	double	Ο μέσος χρόνος λειτουργίας του εξυπηρετητή. Υπολογίζεται ως το άθροισμα των χρόνων εκτέλεσης κάποιων αιτημάτων προς το πλήθος τους
srvClients	int	Το πλήθος των πελατών που διαχειρίζεται ο εξυπηρετητής

srvTotalRuntime	long	Ο συνολικός χρόνος λειτουργίας του εξυπηρετητή μέχρι τον νοστο πελάτη. Υπολογίζεται κάθε φορά που ένας πελάτης έκλεισε την σύνδεση ως το άθροισμα των διαφορών μεταξύ του χρόνου όταν έκλεισε η σύνδεση με τον χρόνο που είχε δημιουργηθεί
host	String	Η διεύθυνση για την επικοινωνία με τον ZooKeeper
connected	Boolean	Βοηθητική μεταβλητή για το αν έχει ολοκληρωθεί η σύνδεση με τον ZooKeeper
expired	Boolean	Βοηθητική μεταβλητή για το αν έχει λήξει η σύνδεση με τον ZooKeeper
clientsControl	ClientsControl	Ο Scheduler για την διαγραφή των πελατών που η σύνδεσή τους έχει λήξει
expiredRemover	ScheduledFuture	Βοηθητική μεταβλητή για τον έλεγχο του Scheduler
createConnectionsCallback	StringCallback	Callback για τη δημιουργία κόμβου παιδιού με τη σύνδεση πελάτη εξυπηρετητή στον κόμβο /connections του ZooKeeper
createServerCallback	StringCallback	Callback για τη δημιουργία κόμβου παιδιού με τα στοιχεία του εξυπηρετητή στον κόμβο /servers του ZooKeeper

createSrvUpAvgTimeCallback	StringCallback	Callback για τη δημιουργία κόμβου παιδιού με τον μέσο χρόνο λειτουργίας του εξυπηρετητή στον κόμβο /srvUpTime του ZooKeeper
createSrvTotalRuntimeCallback	StringCallback	Callback για τη δημιουργία κόμβου παιδιού με τον ολικό χρόνο λειτουργίας του εξυπηρετητή στον κόμβο /srvTotalRuntime του ZooKeeper
createSrvClientsCallback	StringCallback	Callback για τη δημιουργία κόμβου παιδιού με το πλήθος των πελατών του εξυπηρετητή στον κόμβο /srvClients του ZooKeeper

Πίνακας 4.7: Μέθοδοι του Εξυπηρετητή

Μέθοδος	**Return Value**	**Περιγραφή**
AvroServer(host: String)		Constructor , ξεκινά τον Scheduler για τον έλεγχο των συνδέσεων
startServer()		Ξεκινά το Avro κομμάτι του εξυπηρετητή
stopServer()		Σταματά το Avro κομμάτι του εξυπηρετητή
startZk()		Ξεκινά το ZooKeeper κομμάτι του εξυπηρετητή
process(e:WatchedEvent)		Δημιουργεί την σύνδεση με τον ZooKeeper

bootstrap()		Callback για τη δημιουργία κόμβου παιδιού με τη σύνδεση πελάτη εξυπηρετητή στον κόμβο /connections του ZooKeeper
createConnectionNode()		Callback για τη δημιουργία κόμβου παιδιού με τα στοιχεία του εξυπηρετητή στον κόμβο /servers του ZooKeeper
isConnected()	Boolean	Ελέγχει αν ο εξυπηρετητής είναι συνδεδεμένος με τον ZooKeeper
isExpired()	Boolean	Ελέγχει αν η σύνδεση μεταξύ του εξυπηρετητή και του ZooKeeper έχει λήξει
register()		Δημιουργεί κόμβο παιδί με τα στοιχεία του εξυπηρετητή στον κόμβο /servers του ZooKeeper
registersrvUpAvgTime()		Αρχικοποιεί τον κόμβο παιδί με τον μέσο χρόνο λειτουργίας του εξυπηρετητή στον κόμβο /srvUpTime του ZooKeeper
registerSrvTotalRuntime()		Αρχικοποιεί τον κόμβο παιδί με τον ολικό χρόνο λειτουργίας του εξυπηρετητή στον κόμβο /srvTotalRuntime του ZooKeeper
registerSrvClients()		Αρχικοποιεί τον κόμβο παιδί με το πλήθος των πελατών του εξυπηρετητή στον κόμβο /srvClients του ZooKeeper

close()		Κλείνει τον εξυπηρετητή
getClientsMap()	ConcurentHashMap< $String, ClientObj >$	Επιστρέφει τον χάρτη με τα στοιχεία όλων των πελατών του εξυπηρετητή
getName()	String	Επιστρέφει την ταυτότητα του εξυπηρετητή μαζί με το πρόθεμα "server-"
getServerId()	String	Επιστρέφει την ταυτότητα του εξυπηρετητή
getSrvUpAvgTime()	double	Επιστρέφει τον μέσο χρόνο λειτουργίας του εξυπηρετητή
setSrvUpAvgTime(srvUpAvgTime: double		Αλλάζει τον μέσο χρόνο λειτουργίας του εξυπηρετητή
getSrvTotalRuntime()	long	Επιστρέφει τον ολικό χρόνο λειτουργίας του εξυπηρετητή
setSrvTotalRuntime(srvTotalRuntime: long)		Αλλάζει τον ολικό χρόνο λειτουργίας του εξυπηρετητή
getSrvClients()	int	Επιστρέφει το πλήθος των πελατών του εξυπηρετητή
setSrvClients(srvClients: int)		Αλλάζει το πλήθος των πελατών του εξυπηρετητή
incrSrvTotalRuntime(time: long)		Αυξάνει τον ολικό χρόνο λειτουργίας του εξυπηρετητή
updateSrvUpAvgTime(time: double)		Αλλάζει τον μέσο χρόνο λειτουργίας του εξυπηρετητή
incrSrvClients()		Αυξάνει το πλήθος των πελατών του εξυπηρετητή
decrSrvClients()		Μειώνει το πλήθος των πελατών του εξυπηρετητή

getClient(clientId: String)	ClientObj	Δέχεται την ταυτότητα ενός πελάτη και επιστρέφει τα στοιχεία του
addClient(client: ClientObj)		Προσθέτει έναν πελάτη στον χάρτη
removeClient(clientKey: String)		Δέχεται την ταυτότητα ενός πελάτη και τον αφαιρεί από τον χάρτη
updateClientTimestamp(clientId:String, timestamp:long)		Αναβαθμίζει τη χρονική σφραγίδα ενός πελάτη με αυτήν του νέου αιτήματος που ήρθε
updateClDateLastModified(clientId:String, newDate:Date)	String	Αλλάζει την τελευταία χρονική στιγμή που ήταν ενεργή η σύνδεση μεταξύ πελάτη και εξυπηρετητή

Πίνακας 4.8: Μέθοδοι της κλάσης ServerClientImpl

Μέθοδος	**Return Value**	**Περιγραφή**
ServerClientImpl(server:AvroServer)		Constructor , ξεκινά τον Scheduler για τον υπολογισμό του μέσου χρόνου λειτουργίας
send(message:Message)	CharSequence	Δέχεται ένα μήνυμα Avro τύπου Message με τα στοιχεία του πελάτη ελέγχει αν το αίτημα που ήρθε είναι κ το τελευταίο χρονικά που έστειλε ο πελάτης και δεν είναι κάποιο προηγούμενης σειράς που λόγω καθυστερήσεων ήρθε αργότερα από τα άλλα αιτήματα και αν όντως είναι το τελευταίο επιλέγει μια από τις παρακάτω μεθόδους

initConnection(message:Message)	String	Δέχεται ένα μήνυμα Avro τύπου Message με τα στοιχεία του πελάτη, δημιουργεί το αντίστοιχο ClientObj και το αποθηκεύει στον χάρτη των πελατών. Δημιουργεί ένα μονιτορινγ αντικείμενο με το χρόνο που έκανε για την αρχικοποίηση και το αποθηκεύει στην αντίστοιχη λίστα του Scheduler . Επιστρέφει στον πελάτη αν το αίτημα ολοκληρώθηκε επιτυχώς ή όχι.
doSelect(message:Message)	String	Δέχεται ένα μήνυμα Avro τύπου Message με τα στοιχεία για το select του πελάτη προς τη βάση, αναβαθμίζει τους χρόνους στο αντίστοιχο ClientObj του χάρτη των πελατών. Δημιουργεί ένα μονιτορινγ αντικείμενο με το χρόνο που έκανε για την εκτέλεση του αιτήματος και το αποθηκεύει στην αντίστοιχη λίστα του Scheduler .Επιστρέφει στον πελάτη αν το αίτημα ολοκληρώθηκε επιτυχώς ή όχι.
doInsert(message:Message)	String	Δέχεται ένα μήνυμα Avro τύπου Message με τα στοιχεία για το insert του πελάτη προς τη βάση, αναβαθμίζει τους χρόνους στο αντίστοιχο ClientObj του χάρτη των πελατών. Δημιουργεί ένα μονιτορινγ αντικείμενο με το χρόνο που έκανε για την εκτέλεση του αιτήματος και το αποθηκεύει στην αντίστοιχη λίστα του Scheduler . Επιστρέφει στον πελάτη αν το αίτημα ολοκληρώθηκε επιτυχώς ή όχι.

doUpdate(message:Message)	String	Δέχεται ένα μήνυμα Avro τύπου Message με τα στοιχεία για το update του πελάτη προς τη βάση, αναβαθμίζει τους χρόνους στο αντίστοιχο ClientObj του χάρτη των πελατών. Δημιουργεί ένα μονιτορινγ αντικείμενο με το χρόνο που έκανε για την εκτέλεση του αιτήματος και το αποθηκεύει στην αντίστοιχη λίστα του Scheduler . Επιστρέφει στον πελάτη αν το αίτημα ολοκληρώθηκε επιτυχώς ή όχι.
doDelete(message:Message)	String	Δέχεται ένα μήνυμα Avro τύπου Message με τα στοιχεία για το delete του πελάτη προς τη βάση, αναβαθμίζει τους χρόνους στο αντίστοιχο ClientObj του χάρτη των πελατών. Δημιουργεί ένα μονιτορινγ αντικείμενο με το χρόνο που έκανε για την εκτέλεση του αιτήματος και το αποθηκεύει στην αντίστοιχη λίστα του Scheduler .Επιστρέφει στον πελάτη αν το αίτημα ολοκληρώθηκε επιτυχώς ή όχι.
stopMonitoring()		Σταματά τον Scheduler

Πίνακας 4.9: Το αντικείμενο ClientObj

Μεταβλητή	**Τύπος**	**Περιγραφή**
clientID	String	Η ταυτότητα του πελάτη
status	String	Το είδος του αιτήματος από τον πελάτη. Μπορεί να πάρει μόνο μια από τις τιμές : INIT,SELECT,INSERT,UPDATE,DELETE

timestamp	long	Ο χρόνος στον οποίο έγινε το αίτημα. Χρησιμεύει για να διατηρείται μια χρονική διάταξη στα αιτήματα που εκτελούνται. Δηλαδή αν έρθει αίτημα από τον πελάτη με timestamp μικρότερο από το προηγούμενο που είχε εξυπηρετήσει, δεν θα εκτελεστεί.
data	String	Δεδομένα που στέλνει ο πελάτης στον εξυπηρετητή
startedTime	Date	Η μεταβλητή αυτή παίρνει μια και μοναδική τιμή και είναι ο χρόνος στον οποίο ξεκίνησε η σύνδεση πελάτη εξυπηρετητή
dateLastModified	Date	Ο χρόνος που ήταν για τελευταία φορά ενεργή η σύνδεση

Μέθοδος	**ReturnValue**	**Περιγραφή**
ClientObj(clientID:String, status:String, timestamp:String, data:String)		Κατασκευαστής
getDateLastModified	Date	Επιστρέφει τον χρόνο που ήταν για τελευταία φορά ενεργή η σύνδεση
setDateLastModified(dateLastModified:Date)		Αλλάζει την τιμή του χρόνου που ήταν για τελευταία φορά ενεργή η σύνδεση σε dateLastModified
getData	String	Επιστρέφει τα δεδομένα που στέλνει ο πελάτης στον εξυπηρετητή
getClientID()	String	Επιστρέφει την ταυτότητα του πελάτη
getStatus()	String	Επιστρέφει το είδος του αιτήματος από τον πελάτη
getStartedTime()	Date	Επιστρέφει τον χρόνο στον οποίο ξεκίνησε η σύνδεση πελάτη εξυπηρετητή
getTimestamp()	long	Επιστρέφει τον χρόνο στον οποίο στον οποίο έγινε το αίτημα
setTimestamp(timestamp:long)		Αλλάζει την τιμή του χρόνου στον οποίο έγινε το αίτημα σε timestamp

Πίνακας 4.10: Παράμετροι του Scheduler ClientControl

Μεταβλητή	**Τύπος**	**Περιγραφή**
scheduler	ScheduledExecutoeServices	Αντικείμενο τύπου Scheduler
schLog	Logger	Αρχείο όπου γράφει τα αποτελέσματα ο Scheduler

server	AvroServer	Αντικείμενο για να κρατάει εικόνα του εξυπηρετητή που ανήκει ο Scheduler
EXPIRATION_LIMIT	long	Επιτρεπτό χρονικό διάστημα να μείνει ανενεργή η σύνδεση
SCHEDULER_DELAY	long	Ο χρόνος ανά τον οποίο τρέχει ο Scheduler

Πίνακας 4.11: Μέθοδοι του Scheduler ClientControl

Μέθοδος	**Return Value**	**Περιγραφή**
getInstance()	ClientControler	Επιστρέφει το singleton του Scheduler
init(server:AvroServer)		Δέχεται ως αντικείμενο τον εξυπηρετητή που καλεί τον Scheduler και αρχικοποιεί τον private εξυπηρετητή που έχει
removeExpiredClients()	ScheduledFuture<? >	Αφαιρεί όλους τους πελάτες των οποίων η σύνδεση έχει παραμείνει ανενεργή πάνω από τον επιτρεπτό χρόνο
stopClientsControler(remover: ScheduledFuture<? >)		Σταματά τον Scheduler

Πίνακας 4.12: Παράμετροι του Scheduler Monitoring

Παράμετρος	**Τύπος**	**Περιγραφή**
scheduler	ScheduledExecutoreServices	Αντικείμενο τύπου Scheduler
monitorLog	Logger	Αρχείο όπου γράφει τα αποτελέσματα ο Scheduler
server	AvroServer	Αντικείμενο για να κρατάει εικόνα του εξυπηρετητή που ανήκει ο Scheduler
monitElmConcSkipList	ConcurrentSkipList < $Long, MonitoringElement >$	Λίστα στην οποία διατηρούνται όλα τα αντικείμενα τύπου Monitring
SCHEDULER_DELAY	long	Ο χρόνος ανά τον οποίο τρέχει ο Scheduler

Πίνακας 4.13: Μέθοδοι του Scheduler Monitoring

Μέθοδος	**Return Value**	**Περιγραφή**
getInstance()	Monitoring	Παίρνει το singleton του Scheduler
init(server:AvroServer)		Δέχεται ως αντικείμενο τον εξυπηρετητή που καλεί τον Scheduler και αρχικοποιεί τον private εξυπηρετητή που έχει
addMonitoring(elem:MonitoringElement)		Προσθέτει αντικείμενα τύπου Monitring στην λίστα
copyAndClear()	ConcurrentSkipList< $Long, MonitoringElement >$	Παίρνει τα αντικείμενα Monitring της λίστας τα αντιγράφει υπολογίζει τον μέσο χρόνο λειτουργίας με βάση το άθροισμα των χρόνων των αντικειμένων προς το πλήθος τους και αδειάζει τη λίστα
startMonitoring()	ScheduledFuture<? >	Ξεκινά τον Scheduler
stopMonitoring(removeMonitor:ScheduledFuture< ? >)		Σταματά τον Scheduler

Πίνακας 4.14: Το αντικείμενο MonitoringElem

Μεταβλητή	**Τύπος**	**Περιγραφή**
type	String	Το είδος του αιτήματος. Μπορεί να πάρει μόνο μια από τις τιμές: INIT,SELECT,INSERT,UPDATE,DELETE
responseTime	long	Ο χρόνος που έκανε για να ολοκληρωθεί το αίτημα
elementId	long	Η ταυτότητα του αντικειμένου

Μέθοδος	**Return Value**	**Περιγραφή**
MonitoringElement(type:String, responseTime:long)		Ο κατασκευαστής
getType()	String	Επιστρέφει το είδος του αιτήματος
getResponseTime()	long	Επιστρέφει το χρόνο ολοκλήρωσης του αιτήματος
getElementId()	long	Επιστρέφει την ταυτότητα του αιτήματος

4.3.3 Πελάτης

Ο πελάτης στέλνει αιτήματα προς τη βάση. Αρχικά συνδέεται με τον master από τον οποίο ζητά έναν εξυπηρετητή. Ο master του απαντά με τα στοιχεία του εξυπηρετητή που επέλεξε και στη συνέχεια ο πελάτης συνδέεται με τον εξυπηρετητή στέλνοντάς του τα αιτήματα προς τη βάση. Η επικοινωνία τόσο με τον master όσο και με τον εξυπηρετητή γίνεται μέσω πρωτοκόλλων Avro .

Ο πελάτης σαν οντότητα έχει σπάσει σε τρεις κλάσεις, τον *User* , τον *Client* και την *Connection* . Ο *User* μπορούμε να πούμε ότι αντιπροσωπεύει το UI . Ουσιαστικά σηκώνει έναν *Client* και του περνάει τα αιτήματα προς την βάση. Ο *Client* δημιουργεί μια σύνδεση με τον master ζητώντας του έναν εξυπηρετητή. Αφού ο master του απαντήσει με τα στοιχεία του καταλληλότερου εξυπηρετητή ο *Client* κλείνει τη σύνδεση με τον master και σηκώνει μια *Connection* η οποία συνδέεται με τον εξυπηρετητή που της πέρασε ο *Client* . Επιπλέον ο *Client* ενημερώνει τον master όταν κλείσει η επικοινωνία του με τον εξυπηρετητή. Η *Connection* διατηρεί την σύνδεση με τον εξυπηρετητή. Το πρώτο μήνυμα που στέλνει η είναι τύπου INIT με τα στοιχεία του πελάτη ώστε να αρχικοποιηθεί η σύνδεση και στη συνέχεια στέλνει στον εξυπηρετητή τα διάφορα αιτήματα προς την βάση.

Πίνακας 4.15: Μέθοδοι του User

Μέθοδος	**Return Value**	**Περιγραφή**
createClient()	Client	Δημιουργεί ένα καινούριο αντικείμενο Client
createConnection(client:Client)		Λέει στον Client να δημιουργήσει νέα σύνδεση
main(args:String())		

Πίνακας 4.16: Παράμετροι του Client

Μεταβλητή	**Τύπος**	**Περιγραφή**
Log	Logger	Αρχείο όπου γράφει τα αποτελέσματα ο Client
connectSrv	Connection	Αντικείμενο τύπου Connection για τη διαχείρηση της σύνδεσης του πελάτη με τον εξυπηρετητή
masterClientTrans	NettyTransceiver	Αντικείμενο τύπου NettyTransceiver για την σύνδεση του πελάτη με τον master
masterAddress	InetSocketAddress	Η διεύθυνση του master

ipAddr	String	Η διεύθυνση IP του εξυπηρετητή την οποία έστειλε ο master
port	Integer	Η θύρα του εξυπηρετητή την οποία έστειλε ο master
serverId	String	Η ταυτότητα του εξυπηρετητή την οποία έστειλε ο master
clientId	UUID	Η ταυτότητα του πελάτη
data	String	Τα δεδομένα που στέλνει ο πελάτης στη βάση

Πίνακας 4.17: Μέθοδοι του Client

Μέθοδος	**Return Value**	**Περιγραφή**
getInstance()	Client	Επιστρέφει το singleton του Client
Client()		Κατασκευαστής
newConnection()		Δημιουργεί νέο αντικείμενο Connection για τη σύνδεση με τον εξυπηρετητή
sayHello(hello:Hello)	MasterResp	Στέλνει τα στοιχεία του πελάτη στον master μέσω του πρωτοκόλλου Avro MasterClient και ζητά έναν εξυπηρετητή
doSelect()	String	Περνάει τα αιτήματα select στο αντικείμενο Connection για να τα στείλει στον εξυπηρετητή και γυρίζει πίσω την απάντηση του εξυπηρετητή για το αν ήταν επιτυχής η εκτέλεση
doInsert()	String	Περνάει τα αιτήματα insert στο αντικείμενο Connection για να τα στείλει στον εξυπηρετητή και γυρίζει πίσω την απάντηση του εξυπηρετητή για το αν ήταν επιτυχής η εκτέλεση
doUpdate()	String	Περνάει τα αιτήματα update στο αντικείμενο Connection για να τα στείλει στον εξυπηρετητή και γυρίζει πίσω την απάντηση του εξυπηρετητή για το αν ήταν επιτυχής η εκτέλεση

doDelete()	String	Περνάει τα αιτήματα delete στο αντικείμενο Connection για να τα στείλει στον εξυπηρετητή και γυρίζει πίσω την απάντηση του εξυπηρετητή για το αν ήταν επιτυχής η εκτέλεση
stopClient()		Σταματάει τον πελάτη και δίνει εντολή να κλείσει η σύνδεση και να σταλεί αντίστοιχο μήνυμα στον master

Πίνακας 4.18: Παράμετροι της Connection

Μεταβλητή	**Τύπος**	**Περιγραφή**
Log	Logger	Αρχείο όπου γράφει τα αποτελέσματα η Connection
ipAddress	String	Η διεύθυνση IP του εξυπηρετητή την οποία έστειλε ο master
port	int	Η θύρα του εξυπηρετητή την οποία έστειλε ο master
clientId	String	Η ταυτότητα του πελάτη
srvClientTrans	NettyTransceiver	Αντικείμενο τύπου NettyTransceiver για την σύνδεση του πελάτη με τον εξυπηρετητή
proxy	ServerClient	Αντικείμενο του πρωτοκόλλου Avro ServerClient για την μεταφορά μηνυμάτων από τον πελάτη στον εξυπηρετητή

Πίνακας 4.19: Μέθοδοι της Connection

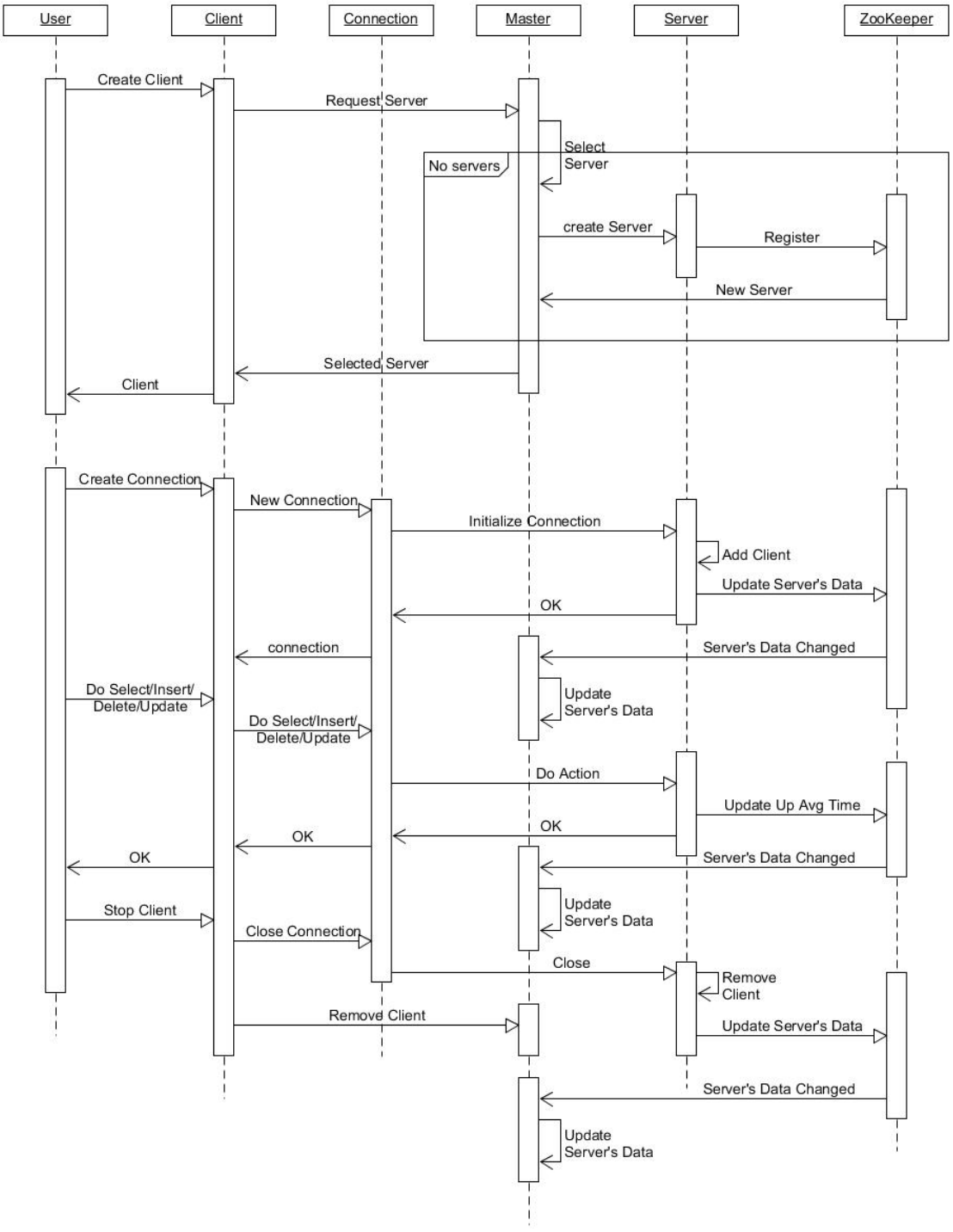
Μέθοδος	**Return Value**	**Περιγραφή**
Connection(clientId:String, ipAddress:String, port:int)		Κατασκευαστής
initSrvClientConnection(status:String, data:String)		Ξεκινάει την σύνδεση με τον εξυπηρετητή στέλνοντας ένα αίτημα τύπου INIT με τα στοιχεία του πελάτη

doAction(status:String, data:String)	String	Δέχεται το είδος του αιτήματος status και τα δεδομένα από τον Client και φτιάχνει το μήνυμα για να στείλει στον εξυπηρετητή
sendMessage(message:Message)	String	Δέχεται το μήνυμα και το προωθεί στον εξυπηρετητή
close()		Κλείνει την σύνδεση με τον εξυπηρετητή

4.4 Περιγραφή Σεναρίου Χρήσης

Αρχικά ο πελάτης ζητά να ξεκινήσει μια συναλλαγή με τη βάση. Τότε ο *User* δημιουργεί έναν *Client* ο οποίος επικοινωνεί με τον master ζητώντας εξυπηρετητή. Ο master ελέγχει αν υπάρχει κάποιος εξυπηρετητής διαθέσιμος. Σε περίπτωση που δεν υπάρχει δημιουργεί αλλά στέλνει μήνυμα λάθους πίσω στον *Client* για να ξανά προσπαθήσει αργότερα. Σε περίπτωση που υπάρχουν διαθέσιμοι εξυπηρετητές, επιλέγει τον καταλληλότερο και στέλνει τα στοιχεία του πίσω στον *Client* . Μόλις λάβει τα στοιχεία αυτά ο *Client* δημιουργεί ένα αντικείμενο τύπου Connection για να συνδεθεί με τον εξυπηρετητή. Η *Connection* στέλνει πρώτα ένα αίτημα αρχικοποίησης για τη σύνδεση με τον εξυπηρετητή. Μόλις ο εξυπηρετητής λάβει αυτό το μήνυμα δημιουργεί ένα αντικείμενο τύπου *ClientObj* το οποίο κρατάει σε έναν χάρτη και ενημερώνει τον ZooKeeper με τα απαραίτητα στοιχεία για τη νέα σύνδεση. Με τη σειρά του ο ZooKeeper ενημερώνει τον master για τα νέα δεδομένα του εξυπηρετητή. Στη συνέχεια στέλνει πίσω στην *Connection* ένα μήνυμα για την επιτυχία ή όχι της διαδικασίας. Σε περίπτωση αποτυχίας η σύνδεση διακόπτεται, αλλιώς η *Connection* προωθεί τα υπόλοιπα αιτήματα προς τη βάση που ζητά ο πελάτης.

Μόλις ολοκληρωθεί η επικοινωνία με τον εξυπηρετητή ο πελάτης κλείνει την σύνδεση. Τότε η *Connection* στέλνει ένα αίτημα τερματισμού στον εξυπηρετητή ο οποίος μόλις το λάβει αφαιρεί το αντίστοιχο αντικείμενο από τον χάρτη του και ενημερώνει τον ZooKeeper με τα νέα δεδομένα και αυτός ενημερώνει τον master . Στη συνέχεια ο *Client* στέλνει ένα αντίστοιχο μήνυμα στον master για τον τερματισμό της σύνδεσης.



Σχήμα 4.4: Sequence Diagram

Κεφάλαιο 5

Υλοποίηση

5.1 Γλώσσα Προγραμματισμού Java

Η Java είναι μια από τις πιο διαδεδομένες γλώσσες αντικειμενοστραφούς προγραμματισμού. Σχεδιάστηκε από την Sun Microsystems και κυκλοφόρησε το 1995 ως στοιχείο του πυρήνα της ομώνυμης πλατφόρμας της Sun . Σχεδιάστηκε ως μια γλώσσα προγραμματισμού γενικού σκοπού με βασικά χαρακτηριστικά την αντικειμενοστρέφεια, την συνέπεια και το ότι βασίζεται σε κλάσεις, ενώ έχει όσο το δυνατόν λιγότερες εξαρτήσεις υλοποίησης. Ένας κώδικας γραμμένος σε Java μπορεί να εκτελεστεί σε οποιαδήποτε πλατφόρμα η οποία την υποστηρίζει χωρίς να χρειάζεται να μεταγλωττιστεί ξανά, ακολουθώντας το μοντέλο "write once, run anywhere" (WORA) . Οι βασικοί στόχοι κατά την σχεδίαση της Java [10] :

- Πρέπει να είναι απλή, αντικειμενοστραφείς και οικεία στον χρήστη.
  Παρουσιάζει ομοιότητες με εντολές της C++
- Πρέπει να είναι σταθερή και ασφαλής.
  Γενικά επειδή αναπτύχθηκε κυρίως για εφαρμογές που θα έτρεχαν σε ετερογενή δίκτυα και επειδή στο διαδίκτυο ελλοχεύουν πολλοί κίνδυνοι, η Java σχεδιάστηκε έτσι ώστε να ελαχιστοποιεί την πιθανότητα προσβολής του συστήματος από κάποιο applet για αυτόν τον σκοπό
- Πρέπει εύκολα να μπορεί να μεταφερθεί από ένα σύστημα σε ένα άλλο ανεξαρτήτου αρχιτεκτονικής [9].
- Πρέπει να έχει υψηλές επιδόσεις.
- Πρέπει να είναι διερμηνεύσιμη (interpreted), νηματική και δυναμική.
  Διερμηνεύσιμη σημαίνει ότι ο μεταγλωτιστής της Java δεν παράγει κάποιο εκτελέσιμο κώδικα αλλά μια μορφή ψευδοκώδικα ο οποίος από μόνος του δεν μπορεί να τρέξει σε καμία μηχανή. Για να δημιουργηθεί εκτελέσιμο

χρειάζεται η χρήση ενός διερμηνέα ο οποίος μετατρέπει τον ψευδοκώδικα σε εκτελέσιμο. Με αυτόν τον τρόπο το πρόγραμμα μπορεί να τρέξει σε οποιοδήποτε μηχάνημα, με οποιοδήποτε λειτουργικό αρκεί να υπάρχει ο διερμηνέας. Νηματική σημαίνει ότι υποστηρίζει τη χρήση πολλών νημάτων, για συστήματα ενός επεξεργαστή η Java χρησιμοποιεί τον δικό της δρομολογητή ενώ σε συστήματα που υποστηρίζουν την χρήση νημάτων η διαχείριση τους ανατίθεται στο λειτουργικό σύστημα.

Οι παραπάνω λόγοι ήταν καθοριστικοί στην επιλογή της Java ως γλώσσας προγραμματισμού για την ανάπτυξη του συστήματος αυτής της διπλωματικής εργασίας. Πιο αναλυτικά, εξαιτίας του αντικειμενοστραφούς χαρακτήρα της παρέχει απλότητα στον σχεδιασμό, διαχωρίζοντας τον κώδικα σε ξεχωριστές οντότητες η οποίες είναι πιο εύκολα διαχειρίσιμες, ενώ μπορούν εύκολα να επαναχρησιμοποιηθούν και σε άλλα κομμάτια του προγράμματος. Επιπλέον, είναι εύκολή η επέκταση χωρίς να επηρεάζονται οι προηγούμενες λειτουργίες του συστήματος. Η εφαρμογή θα αναπτυχθεί ως υπηρεσία υπολογιστικού νέφους οπότε ήταν αναγκαίο χαρακτηριστικό η ασφάλεια που παρέχει έναντι διαφόρων προγραμμάτων κακόβουλου λογισμικού. Επίσης, επειδή η εφαρμογή θα βρίσκεται στο νέφος και θα χρησιμοποιείται από πολλές πλατφόρμες ιδιαίτερης σημασίας είναι και η ανεξαρτησία από τον τύπο της αρχιτεκτονικής που μας παρέχει η Java . Τέλος, ακόμα χαρακτηριστικό για το οποίο προτιμήθηκε είναι ότι έχει αναπτυχθεί κυρίως για διαδικτυακές εφαρμογές οπότε παρέχει πλούσιες βιβλιοθήκες για γρήγορη και ασφαλή διαδικτυακή επικοινωνία την οποία χρειαζόμαστε τόσο για την επικοινωνία ενδοδικτυακά μεταξύ των εξυπηρετητών όσο και για την επικοινωνία μεταξύ πελάτη-εξυπηρετητή. Εναλλακτικά μια άλλη επιλογή που θα μπορούσαν να χρησιμοποιηθούν αντί της Java για την ανάπτυξη του συστήματος θα ήταν η Python η οποία είναι και αυτή μια αντικειμενοστραφής γλώσσα προγραμματισμού που έχει ιδιαίτερη εφαρμογή στην ανάπτυξη διαδικτυακών εφαρμογών.

5.2 Maven

Το Maven Project είναι προϊόν της εταιρίας Απαχε Σοφτωαρε Φουνδατιον [11]. Είναι ένα εργαλείο ανοιχτού κώδικα για τη διαχείριση και ανάπτυξη ενός προγράμματος λογισμικού. Βασίζεται στην ιδέα της ανάπτυξης μέσω μοντέλων αντικειμένων και βοηθάει στο χτίσιμο, την τεκμηρίωση και τον έλεγχο ενός έργου μέσω μιας κεντρικής πληροφορίας. Είναι χτισμένο σε Θαα και επιτρέπει την ανάπτυξη Θαα εφαρμογών.

Ένα από τα κύρια χαρακτηριστικά του είναι η διατήρηση του βασικού κύκλου ζωής της ανάπτυξης ενός προγράμματος λογισμικού σύμφωνα με τις αρχές της βιομηχανίας λογισμικού. Έτσι περιλαμβάνει όλα τα στάδια της ζωής της: από τον συνταχτικό έλεγχο και τη μεταγλώττιση, μέχρι την εγκατάσταση σε έναν εξυπηρετητή και την τεκμηρίωση. Όλες οι πληροφορίες που αφορούν το πρόγραμμα βρίσκονται συγκεντρωμένες σε ένα κεντρικό αρχείο πληροφορίας

με κατάληξη pom, μέσα στο οποίο ορίζονται το όνομα του προγράμματος και η έκδοσή του, οι διάφορες εξωτερικές βιβλιοθήκες που είναι απαραίτητες για να εκτελεστεί το είδος ελέγχου που θα πραγματοποιηθεί πριν την εγκατάσταση. Το Maven επιτρέπει επιπλέον την οργάνωση ολόκληρου του συστήματος σε οντότητες λογισμικού, έχοντας όλα τα χαρακτηριστικά των αντικειμενοστραφή μοντέλων, παρέχοντας δυνατότητες κληρονομικότητας, πολυμορφισμού κτλ.

Στο σύστημα που αναπτύσσουμε κρίθηκε απαραίτητο για την καλύτερη οργάνωση του κώδικα σε προτυποποιημένη ιεραρχική δομή και την ευκολία που παρέχει για την άμεση μεταγλώττισή του.

5.3 Apache Zookeeper

5.3.1 Ορισμός

Ο Apache ZooKeeper έχει σχεδιαστεί από την Apache Software Foundation ώστε να παρέχει υπηρεσίες ανοιχτού κώδικα για τη διαχείριση και τον συγχρονισμό κατανεμημένων συστημάτων. Γράφτηκε σε γλώσσα προγραμματισμού Java και δουλεύει σε όλα τα γνωστά λειτουργικά συστήματα. Αρχικά είχε ξεκινήσει ως μια βοηθητική υπηρεσία ενός άλλου έργου της Apache, το Apache Hadoop, γρήγορα όμως έγινε απαραίτητο εργαλείο σε μεγάλα έργα γνωστών εταιριών όπως η Yahoo! και το eBay, με αποτέλεσμα να κερδίσει μια θέση ανάμεσα στα Top-Level Projects [1] της Apache [5].

5.3.2 Γενικές Πληροφορίες

Χάρη στη ραγδαία ανάπτυξη που γνωρίζουν σήμερα κλάδοι της πληροφορικής όπως αυτοί των Big Data και του Cloud Computing δημιουργείται όλο και μεγαλύτερη ανάγκη για εφαρμογές που αποτελούνται από πολλά συνεργαζόμενα προγράμματα, τα οποία τρέχουν είτε παράλληλα είτε σειριακά σε διαφορετικούς υπολογιστές. Η νέα αυτή τάση ανάπτυξης τέτοιων εφαρμογών έφερε στο προσκήνιο το θέμα για το πώς αυτά τα διαφορετικά προγράμματα θα συγχρονίζονται μεταξύ τους. Είναι ιδιαίτερα δύσκολο και χρονοβόρο για έναν προγραμματιστή να ασχοληθεί παράλληλα με την υλοποίηση της λογικής του συγχρονισμού και την υλοποίηση της λογικής της ίδιας της εφαρμογής του. Αυτό το πρόβλημα ήρθε να λύσει ο ZooKeeper παρέχοντας μια αξιόπιστη υπηρεσία που επιτρέπει στον προγραμματιστή να ασχοληθεί ελάχιστα με το θέμα το συγχρονισμού και να επικεντρωθεί στην ανάπτυξη του λογισμικού του.

Όταν μιλάμε για τον συγχρονισμό διεργασιών μπορούμε να πούμε ότι μιλάμε είτε για τον συντονισμό τους ώστε να δουλεύουν παράλληλα είτε για τον αμοιβαίο αποκλεισμό της μιας με την άλλη ώστε να μπορεί να τρέχει μόνο μια τη φορά. Κλασικό είναι το παράδειγμα του master/worker, όπου υπάρχει

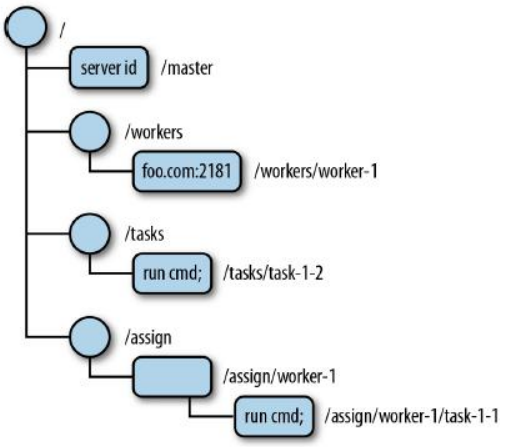
[1] Η Apache χωρίζει τις δραστηριότητες ανάπτυξης λογισμικού σε ξεχωριστές ημιαυτόνομες περιοχές που ονομάζονται Top-Level Projects. Κάθε τέτοιο έργο έχει τη δική του Επιτροπή Διαχείρισης Έργου (Project Management Committee, PCM) [3]

ένας συντονιστής ο master και οι workers οι οποίοι εκτελούν τις διεργασίες. Για τον αμοιβαίο αποκλεισμό μπορούμε να αναφέρουμε την περίπτωση του master,ο οποίος πρέπει να είναι μοναδικός, αλλά υπάρχουν πολλές διεργασίες που ταυτόχρονα προσπαθούν να πάρουν αυτή τη θέση. Ενώ για τον παράλληλο συντονισμό μπορούμε να αναφέρουμε τους workers,οι οποίοι πρέπει να ενημερώνουν τον μαστερ ότι είναι έτοιμοι να εκτελέσουν κάποια διεργασία ώστε να τους την αναθέσει. Επίσης οι workers πρέπει να είναι σε θέση να γνωρίζουν ποια εργασία εκτελούν ακόμα και αν ο master αποτύχει, η διαδικασία αυτή στον συγχρονισμό ονομάζεται διαχείριση μεταδεδομένων . Μέσω της υπηρεσίας του ZooKeeper ο προγραμματιστής μπορεί να υλοποιήσει αυτές τις κλασικές ενέργειες συγχρονισμού, δηλαδή την εκλογή ενός master server,τη διαχείριση ομάδων (π.χ. τωνworkers) καθώς και τη διαχείριση μεταδεδομένων. Επιπλέον μπορεί να χρησιμοποιηθεί για ειδοποίηση σε περίπτωση ολοκλήρωσης κάποιου συμβάντος, για κλειδώματα ή ως μηχανισμός ουράς προτεραιότητας [4].

Υπάρχουν πολλοί αλγόριθμοι οι οποίοι μας επιτρέπουν να υλοποιήσουμε τις παραπάνω τεχνικές, εκεί όμως που ο ZooKeeper υπερτερεί είναι στο ότι απλοποιεί την όλη διαδικασία με το να προσφέρει έναν κοινό χώρο αποθήκευσης με κάποιες επιπλέον ιδιότητες ταξινόμησης όπου όλα τα προγράμματα, τα οποία τρέχουν σε διαφορετικούς υπολογιστές και υπό διαφορετικές συνθήκες δεν θα είχαν κανέναν άλλο κοινό χώρο μνήμης, έχουν πρόσβαση. Με αυτόν τον τρόπο για παράδειγμα, αντί ο master να στέλνει συνεχώς μηνύματα στους γύρω του ότι έχει ήδη εκλεγεί, αρκεί ο κάθε υποψήφιος νέος master να κοιτάει στον ZooKeeper και να βλέπει αν ήδη υπάρχει κάποιος εκλεγμένος ή όχι.

Ο ZooKeeper προσφέρει διεπαφές για κατανεμημένα συστήματα γραμμένα είτε σε C είτε σε Java . Για τον συντονισμό χρησιμοποιεί ιεραρχικά δομημένη ονοματοδοσία που είναι εμπνευσμένη από τη δόμηση των αρχείων [Σχήμα 5.1]. Ενώ τα δεδομένα κρατούνται στη μνήμη και αντιγράφονται σε πολλούς κόμβους για ασφάλεια. Με την επιλογή του να χρησιμοποιεί τη μνήμη ως μέσο συγχρονισμού ο ZooKeeper έχει γρήγορη ανταπόκριση και διαχειρίζεται μεγάλο όγκο φορτίου, γεγονός που βλέπουμε σε πρωτόκολλα συγχρονισμού για πολύ μεγάλα κατανεμημένα συστήματα. Όμως δεν μπορεί να αποθηκεύσει μεγάλο όγκο δεδομένων, δεν είναι χρήσιμος ως μια γενική αποθηκευτική μονάδα. Για αυτόν τον λόγο στις εφαρμογές που τον χρησιμοποιούν πρέπει να υπάρχει ξεκάθαρος διαχωρισμός ανάμεσα στα δεδομένα που έχουν να κάνουν με τον συγχρονισμό από τα υπόλοιπα δεδομένα της εφαρμογής. Οτιδήποτε έχει να κάνει με τον συντονισμό των διεργασιών (για παράδειγμα το όνομα ενός από τους servers και η διεύθυνση IP του) μπορεί να αποθηκευτεί στον ZooKeeper , τα υπόλοιπα (για παράδειγμα φωτογραφίες που θα χρειαζόταν να φέρει ο serverγια μια ιστοσελίδα) πρέπει να αποθηκεύονται σε άλλους καταλληλότερους χώρους αποθήκευσης.

Ένα ερώτημα που μπορεί να δημιουργηθεί είναι γιατί να μη χρησιμοποιήσουμε μία απλή βάση δεδομένων για τον συντονισμό των διεργασιών αντί για τον ZooKeeper ; Η απάντηση είναι γιατί μας προσφέρει πολύ περισσότερα από μια κοινή βάση δεδομένων. Αρχικά μας εξασφαλίζει ότι οι ενημερώσεις για



Σχήμα 5.1: Δέντρο Δεδομένων στον ZooKeeper

αλλαγές στα δεδομένα γίνονται πάντα λαμβάνοντας υπόψιν άλλες αντίστοιχες ενημερώσεις, συμβάντα που μπορούν να δημιουργηθούν και ασύγχρονες απαντήσεις που στέλνονται πίσω στον client. Επιπλέον υπάρχει εγγύηση ότι ο client θα δει πρώτα το συμβάν της αλλαγής σε κάποιο δεδομένο και μετά την ανανεωμένη του εκδοχή. Τέλος είναι σίγουρο ότι η σειρά που θα δει ο client τις αναβαθμίσεις σε δεδομένα είναι η ίδια σειρά με αυτή που είδε τα αντίστοιχα συμβάντα ο ZooKeeper . Γενικεύοντας, μπορούμε να πούμε ότι τα δυνατά σημεία του ZooKeeper είναι ότι εγγυάται την συνοχή, την διάταξη και την ανθεκτικότητα. Επιπλέον επιτρέπει την εύκολη υλοποίηση τον βασικών διαδικασιών συγχρονισμού μέσω απλών διεπαφών. Παρόλα αυτά μπορούμε να παρατηρήσουμε και μερικά αδύνατα σημεία του. Όπως για παράδειγμα το ότι δεν μπορούμε να τον χρησιμοποιήσουμε για την αποθήκευση μεγάλου όγκου δεδομένων. Το βασικότερο, όμως, μειονέκτημά του είναι ότι η παρατήρηση για ένα γεγονός είναι μεμονωμένη, με αποτέλεσμα αν γίνει αλλαγή πριν προλάβει να ανανεωθεί ο παρατηρητής από τον client χάνεται. [17]

5.3.3 Παραδείγματα Χρήσης του ZooKeeper

Παρακάτω μπορούμε να δούμε μερικά παραδείγματα διαφόρων εφαρμογών και οργανισμών που χρησιμοποιούν τον ZooKeeper προκειμένου να κατανοήσουμε καλύτερα τις δυνατότητές του[16].

Έργα Ανοιχτού Λογισμικού

- **AdroitLogic UltraESB:**
  Χρησιμοποιεί τον ZooKeeper σε συνδυασμό με το JMX έναν σκελετό εντολών δομημένο πάνω στον συγχρονισμό του ZooKeeper . Με αυτόν τον τρόπο ελέγχουν καλύτερα το συντονισμό τόσο σε επίπεδο κόμβου όσο και σε επίπεδο ομάδων.

- **Eclipse Gyrex :**
  Το Eclipse Gyrex είναι ένα έργο που παρέχει μια πλατφόρμα για την κατασκευή σύννεφου βασιζόμενου σε Java OSGi. Ο ZooKeeper χρησιμοποιείται ως δομικό στοιχείο του σύννεφου για τη διαχείριση των κόμβων, τον συντονισμό εργασιών ανάμεσα στους workers , για κλειδώματα, ως απλή ουρά και πολλά άλλα.
- **Neo4j:**
  Η Neo4j είναι μια βάση δεδομένων γράφων και χρησιμοποιεί τον ZooKeeper για την εκλογή write-master και τον συντονισμό read-slaves.

Έργα της Apache

- **Apache HBase:**
  Το HBase είναι αποθηκευτική μονάδα η οποία χρησιμοποιείται μαζί με το Hadoop . Χρησιμοποιεί τον ZooKeeper για την εκλογή master στις ομάδες, να κρατάει αρχείο με τους διαθέσιμους servers και να διαχειρίζεται τα μεταδεδομένα.
- **Apache Kafka:**
  Το Kafka είναι υπηρεσία διαχείρισης μηνυμάτων. Χρησιμοποιεί τον ZooKeeper για να εντοπίζει αποτυχίες, να ανακαλύπτει θεματολογίες για ομαδοποιήσεις και να διατηρεί μια κατάσταση παραγωγής/κατανάλωσης αυτών των θεματολογιών.
- **Apache Solr:**
  Χρησιμοποιεί τον ZooKeeper για την εκλογή master , εντοπισμό αποτυχιών και αποθήκευση μεταδεδομένων.

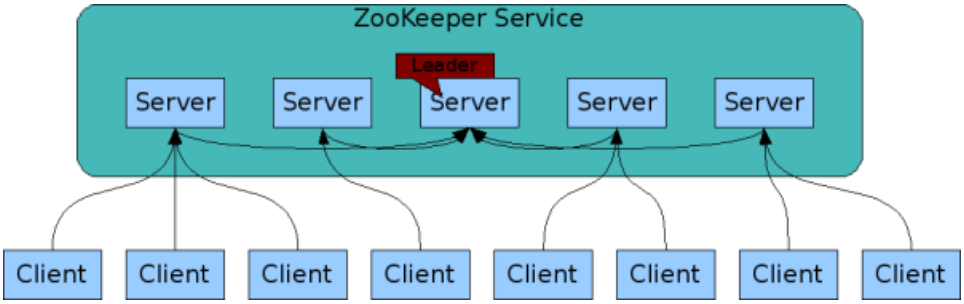
Οργανισμοί

Yahoo!:
Ο ZooKeeper χρησιμοποιείται για μια πληθώρα υπηρεσιών μέσα στη Yahoo!, όπως για εκλογή master , για ρυθμιστικές διεργασίες, για sharding , για κλειδώματα και διαχείριση ομάδων.

Facebook:
Χρησιμοποιεί τον ZooKeeper στο Facebook Messages , μια εφαρμογή η οποία ενσωματώνει διαφορετικά κανάλια επικοινωνίας (email, SMS, Facebook chat, Facebook inbox). Ο ZooKeeper χρησιμοποιείται για sharding και έλεγχο αποτυχιών.

5.3.4 Περιγραφή Λειτουργίας του ZooKeeper

Ο ZooKeeper τρέχει σε μια ομάδα από servers με αποτέλεσμα να υπάρχει αντοχή σε αποτυχίες και καλύτερη απόδοση [Σχήμα 5.2] . Ακολουθείται πάντα το σχήμα master/workers . Με βάση αυτή του τη δομή οι προγραμματιστές



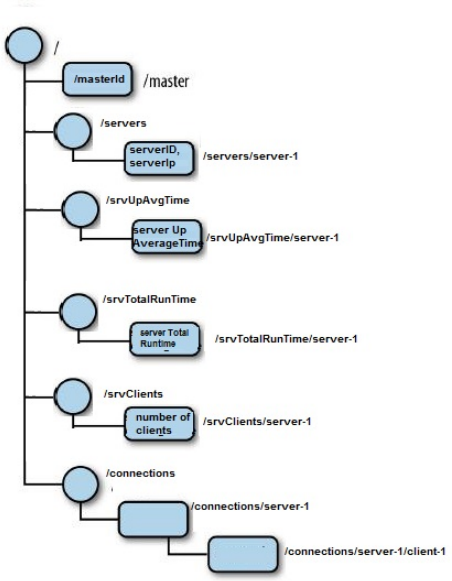
Σχήμα 5.2: Δομή του ZooKeeper

όταν τον χρησιμοποιούν στις εφαρμογές τους, γράφουν ουσιαστικά ομάδες από clients οι οποίοι συνδέονται στους servers και χρησιμοποιούν τις υπηρεσίες που τους παρέχουν. Τα δεδομένα της εφαρμογής κατανέμονται ανάμεσα σε πολλούς κόμβους και ο client μπορεί να συνδεθεί σε οποιονδήποτε από αυτούς, ενώ σε περίπτωση αποτυχίας μπορεί να συνδεθεί σε οποιονδήποτε άλλο από τους διαθέσιμους servers χωρίς τον κίνδυνο να χαθού δεδομένα.

Όσον αναφορά τις εγγραφές τα πάντα περνούν από τον master , έτσι εξασφαλίζεται ότι οι εγγραφές γίνονται γραμμικά. Όταν γίνεται μια εγγραφή η πληροφορία καταγράφεται σε παραπάνω από έναν κόμβους, συμπεριλαμβάνοντας τον server για τον αντίστοιχο client και τον master . Με αυτόν τον τρόπο όλοι οι servers είναι πάντα σε συμφωνία με τον master . Όμως αυτή η μέθοδος δεν ενδείκνυται για λειτουργίες οι οποίες βασίζονται κυρίως σε εγγραφές διότι υπάρχει πολύ μεγάλος φόρτος δεδομένων. Δηλαδή ο ZooKeeper δεν μπορεί να χρησιμοποιηθεί για εφαρμογές που απαιτούν να γίνονται συνεχόμενα πολλές εγγραφές. Αντίθετα είναι ιδανικός σε περιπτώσεις συγχρονισμού ανταλλαγής μνημάτων μεταξύ clients .

Το δυνατό σημείο του ZooKeeper είναι η ανάγνωση δεδομένων, η οποία μπορεί να γίνεται συνεχόμενα αφού όλα τα αιτήματα ανάγνωσης εξυπηρετούνται από έναν συγκεκριμένο server με τον οποίο συνδέεται ο client . Το μόνο μειονέκτημα που μπορεί να υπάρξει είναι ο client να βλέπει παλιά δεδομένα γιατί ο server δεν έχει προλάβει να ενημερωθεί από τον master [8].

Ο ZooKeeper έχει υιοθετήσει μια δομή που μοιάζει με αυτή των συστημάτων αρχείων. Αποτελείται από μικρούς κόμβους δεδομένων τους znodes . Αυτοί οι κόμβοι είναι δομημένοι ιεραρχικά όπως ένα δέντρο. Επιπλέον κάθε znode έχει και έναν πίνακα από bytes για να αποθηκεύει δεδομένα [Σχήμα 1.1]. Κάθε κόμβος περιλαμβάνει δεδομένα και μπορεί να περιλαμβάνει και άλλους κόμβους-παιδιά. Επιπλέον κάθε ένας znode έχει μια συγκεκριμένη ρύθμιση με βάση την οποία συμπεριφέρεται. Υπάρχουν τριών ειδών κόμβοι, οι επίμονοι οι παροδικοί και οι ακολουθιακό. Ένας επίμονος znode μπορεί να διαγραφεί μόνο κάνοντας κλήση διαγραφής. Ενώ, οι παροδικοί znodes διαγράφονται αν αποτύχει ο client ή αν κλείσει τη σύνδεση του με τον ZooKeeper . Οι παροδικοί κόμβοι μπορούν να χρησιμοποιηθούν για κλειδώματα. Για παράδειγμα αν θέλουμε να εκλέξουμε

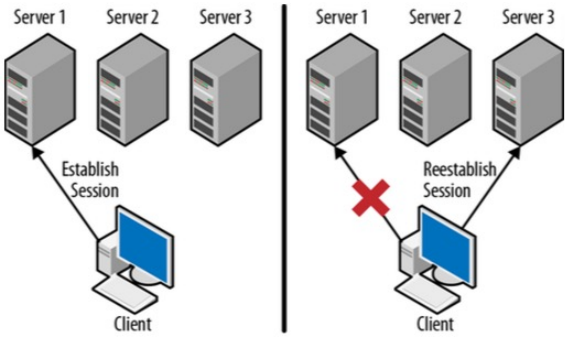


Σχήμα 5.3: Δένδρο Δεδομένων για τη Διπλωματική Εργασία

έναν master δημιουργούμε έναν παροδικοί κόμβο /master και οι υπόλοιποι υποψήφιοι ελέγχουν την ύπαρξη αυτού του κόμβου. Αν για κάποιο λόγο ο εκλεγμένος master αποτύχει τότε ο κόμβος διαγράφεται και τότε μπορεί κάποιος νέος server να εκλεγεί, δημιουργώντας με τη σειρά του νέο κόμβο. Αξίζει να σημειωθεί ότι μόνο σε επίμονους κόμβους μπορούμε να δημιουργήσουμε κόμβους-παιδιά. Τέλος οι ακολουθιακό πέρα από το όνομα που δίνει ο client στον κόμβο προσθέτουν ως μια επιπλέον κατάληξη έναν αριθμό μιας ακολουθίας η οποία επιλέγεται από την ομάδα των servers του ZooKeeper . Αυτοί οι κόμβοι είναι χρήσιμοι, για παράδειγμα, σε περιπτώσεις όπου πολλοί clients θέλουν κάποιο κλείδωμα. Τότε ο κάθε client μπορεί να δημιουργεί έναν ακολουθιακό κόμβο και αυτός με τον μικρότερο αριθμό ακολουθίας να λαμβάνει κάθε φορά το κλείδωμα.

Στο Σχήμα 5.3 μπορούμε να δούμε την δεντρική δομή που ακολουθήθηκε στη διπλωματική εργασία. Το τι αντιπροσωπεύει ο κάθε κόμβος θα εξηγηθεί σε παρακάτω κεφάλαιο.

Ο ZooKeeper παρέχει ένα σύστημα για την παρατήρηση συμβάντων και στέλνει ειδοποίηση πίσω στον client σε περίπτωση αλλαγής. Υπάρχουν διαφορετικών ειδών ειδοποιήσεις ανάλογα με τον τύπο της αλλαγής, για παράδειγμα άλλου τύπου είναι η ειδοποίηση για το αν άλλαξαν τα δεδομένα ενός znode και άλλου τύπου αν διαγράφτηκε κάποιος κόμβος. Όμως οι ειδοποιήσεις είναι γεγονότα που δημιουργούνται μόνο μια φορά και μετά ο client πρέπει να ανανεώσει τον παρατηρητή στον αντίστοιχο znode . Αν πριν την ανανέωση γίνει



Σχήμα 5.4: Μετακίνηση συνεδρίας στον ZooKeeper

κάποια ενημέρωση δεν θα τη δει ο client . Πέρα όμως από αυτό το μειονέκτημα, ο ZooKeeper μας εγγυάται ότι οι ενημερώσεις θα φτάνουν πάντα στον πελάτη σε αντιστοιχία με τις αλλαγές που τις πυροδότησαν.

5.3.5 Χρήση του ZooKeeper στη διπλωματική εργασία

Ο ZooKeeper χρησιμοποιήθηκε στη διπλωματική εργασία αφενός για την εκλογή του master και αφετέρου για να βοηθάει τον εκάστοτε master στην επιλογή του καταλληλότερου serverγια να εξυπηρετήσει κάποιον client .

Δημιουργία συνεδρίας με τον ZooKeeper

Ο πελάτης όταν επικοινωνεί με τον ZooKeeper ουσιαστικά δημιουργεί μια συνεδρία η οποία παραμένει ενεργή για όλη τη διάρκεια της επικοινωνίας. Ακόμα και αν ο εξυπηρετητής που είναι συνδεδεμένος ο πελάτης αποτύχει η συνεδρία θα μετακινηθεί σε κάποιον από τους άλλους εξυπηρετητές του ZooKeeper [Σχήμα: 5.4]. Γενικά όσο η συνεδρία παραμένει ζωντανή ο ZooKeeper θα προσπαθεί να κρατά μια ενεργή σύνδεση μεταξύ του πελάτη και κάποιου από τους εξυπηρετητές του για να τη διατηρεί. Αν ο πελάτης κλείσει τη σύνδεση τότε ο ZooKeeper θα τερματίσει την συνεδρία. Ενώ αν ο ZooKeeper αποφασίσει ότι ο πελάτης έχει ώρα να ανταποκριθεί και υπάρχει κάποιο σφάλμα θα ακυρώσει την συνεδρία και σε περίπτωση που ο πελάτης προσπαθήσει να ξανασυνδεθεί μέσω αυτής της ακυρωμένης συνεδρίας θα του επιστραφεί μήνυμα λάθους και δεν θα επιτραπεί η σύνδεση. Ο κατασκευαστής για αυτήν την συνεδρία είναι της μορφής :

```
ZooKeeper(String connectingString,
     int sessionTimeout,
     Watcher watcher)
```

Όπου:

- connectingString : περιέρχει το όνομα παράχου και την θήρα των εξυπηρετητών του ZooKeeper

- sessionTimeout : είναι ο χρόνος σε milliseconds όπου περιμένει ο ZooKeeper χωρίς να λάβει απάντηση από τον πελάτη μέχρι να αποφασίσει ότι έληξε η συνεδρία.
- watcher είναι ένα αντικείμενο απαραίτητο για να λαμβάνουμε τα γεγονότα που συμβαίνουν κατά τη διάρκεια της συνεδρίας. Αυτά τα γεγονότα τα λαμβάνει η μέθοδος *process(WatchedEvent e)* και είναι
  - SyncConnected : άμα υπάρξει σύνδεση
  - Disconnected : άμα η σύνδεση κλείσει
  - Expired : άμα η σύνδεση λήξει

Τα παραπάνω υλοποιηθηκαν στην διπλωματική μέσω των εξής μεθόδων :

Listing 5.1: Create A Session In ZooKeeper

```
public Master(String host) {
  this.host = host;
}
public void startZk() throws IOException {
  zk = new ZooKeeper(host, 15000, this);
}
public void stopZk() throws IOException, InterruptedException {
  zk.close();
}
public void process(WatchedEvent e) {
  masterLog.info("Processing event: " + e.toString());
  if (e.getType() == Event.EventType.None) {
    switch (e.getState()) {
    case SyncConnected:
      connected = true;
      break;
    case Disconnected:
      connected = false;
      break;
    case Expired:
      expired = true;
      connected = false;
      masterLog.error("Session expired");
      break;
    default:
      break;
    }
  }
}
```

Εκλογή master

Ο master είναι υπεύθυνος να αναθέτει πελάτες στους εξυπηρετητές και να διαχειρίζεται τα μεταδεδομένα του ZooKeeper . Είναι μοναδικός και για την

εκλογή του πρέπει να πάρει το αντίστοιχο κλείδωμα. Για να το πετύχει αυτό η εφαρμογή δημιουργεί στον ZooKeeper έναν εφήμερο κόμβο τον /master . Ουσιαστικά master εκλέγεται όποιος καταφέρει πρώτος να δημιουργήσει τον ομώνυμο κόμβο.

Χρειαζόμαστε δυο στοιχεία για να δημιουργήσουμε τον /master . Αρχικά χρειαζόμαστε τα δεδομένα του κόμβου. Στην διπλωματική εργασία στον /master αποθηκευόταν και το IDτου. Επιπλέον χρειαζόμαστε και μια λίστα με όσους έχουν πρόσβαση στον κόμβο (Access Control List). Επειδή πολύ συχνά ο ZooKeeper τρέχει σε ασφαλές περιβάλλον μας αρκεί ανοικτή λίστα η οποία περιγράφεται μέσω της λέξης κλειδιού `OPEN_ACL_UNSAFE` . Αυτή η λίστα επιτρέπει την πρόσβαση στον κόμβο σε όλους τους χρήστες και όπως δηλώνει και το όνομα της δεν είναι ασφαλές να χρησιμοποιείται σε περιβάλλοντα τα οποία δεν εμπιστευόμαστε.

Για τη δημιουργία του κόμβου χρησιμοποιήσαμε ασύγχρονες κλήσεις. Με αυτόν τον τρόπο επιτρέψαμε να γίνονται πολλές διεργασίες ταυτόχρονα από το ίδιο νήμα αφού η ασύγχρονη κλήση δεν μπλοκάρει μέχρι να έρθει απάντηση στην αίτηση δημιουργίας από τον εξυπηρετητή και επιστρέφει αμέσως. Επιπλέον απλοποιήσαμε αρκετά την υλοποίηση της εφαρμογής αφού δεν υπάρχει ανάγκη να διαχειριστούμε εξαιρέσεις. Επειδή η κλήση δεν περιμένει να δημιουργηθεί ο κόμβος για να επιστρέψει δεν χρειάζεται να ανησυχούμε για InterruptedExceptions , και λόγω του ότι οποιοδήποτε λάθος κατά το αίτημα δημιουργίας κωδικοποιείται μέσα στο callback που χρησιμοποιεί η ασύγχρονη κλήση, δεν χρειάζεται να ανησυχούμε για KeeperExceptions . Για την ασύγχρονη μέθοδο υλοποίησης χρειαζόμαστε επιπλέον δυο παραμέτρους, σε σχέση με τη σύγχρονη: ένα αντικείμενο τύπου callback το οποίο χρησιμεύει για να λάβει το αποτέλεσμα της κλήσης και για να περάσουμε επιπλέον πληροφορία και ένα αντικείμενο το οποίο προσδιορίζεται από τον χρήστη και είναι ουσιαστικά η πληροφορία την οποία περνάμε μέσω του callback .

Για την εκλογή του master αρχικά καλείται η μέθοδος *runForMaster()* η οποία προσπαθεί να δημιουργήσει τον κόμβο /master στον ZooKeeper . Η κλήση για την δημιουργία γίνεται ασύγχρονα και η απάντηση για το αν ήταν επιτυχής ή όχι λαμβάνεται μέσω του *StringCallback masterCreateCallback* , το οποίο περνάει ουσιαστικά έναν κωδικό κατάστασης που μεταφράζεται ως εξής:

- CONNECTIONLOSS : σε περίπτωση που χαθεί η σύνδεση ξανά ελέγχουμε αν υπάρχει ήδη ο κόμβος
- NODEEXISTS : σε περίπτωση που βρούμε τον κόμβο τότε σημαίνει πως κάποιος άλλος εξυπηρετητής έχει πάρει το κλείδωμα του /master . Τότε ελέγχουμε για αλλαγές στον συγκεκριμένο κόμβο, δηλαδή κοιτάμε πότε δε θα υπάρχει πια για να ξανά προσπαθήσουμε να πάρουμε το κλείδωμα.
- OK : δημιουργήθηκε ο κόμβος, ο εξυπηρετητής πήρε το κλείδωμα του /master οπότε προχωράμε στην διαδικασία ανάληψης των καθηκόντων

του.

- Σε οποιοδήποτε άλλη περίπτωση υπήρχε κάποιο πρόβλημα και ο εξυπηρετητής δεν κατάφερε να γίνει ο master .

Listing 5.2: Run For Master Method

```
public void runForMaster() {
  masterLog.info("Running for master...");
  zk.create("/master",
         masterId.getBytes(),
         Ids.OPEN_ACL_UNSAFE,
         CreateMode.EPHEMERAL,
         masterCreateCallback,
         null);
}

StringCallback masterCreateCallback = new StringCallback() {

  public void processResult(int rc, String path, Object ctx,
      String name) {
    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      checkMaster();
      break;
    case OK:
      state = MasterStates.ELECTED;
      takeLeadership();
      break;
    case NODEEXISTS:
      state = MasterStates.NOTELECTED;
      masterExists();
      break;
    default:
      state = MasterStates.NOTELECTED;
       masterLog.error("Something went wrong when running for
           master...",KeeperException.create(Code.get(rc), path
           ));
    }
    masterLog.info("I'm "+ (state == MasterStates.ELECTED ? ""
        : "not ") + "the leader " + masterId);
  }
};
```

Στη συνέχεια υλοποιούμε τη μέθοδο *checkMaster()* για τον έλεγχο ύπαρξης του κόμβου /master . Αυτός ο έλεγχος γίνεται είτε σε περίπτωση που χαθεί η σύνδεση κατά την προσπάθεια δημιουργίας του. Ουσιαστικά ζητάμε τα δεδομένα από τον υπάρχοντα κόμβο. Η απάντηση στην ασύγχρονη κλήση έρχεται μέσω του *DataCallback masterCheckCallback* όπου κατά αντιστοιχία με πριν μπορεί να είναι :

- CONNECTIONLOSS : όταν χαθεί η σύνδεση, όπου ξανά ελέγχουμε για την ύπαρξη τον /master .
- NONODE : όταν δεν βρούμε τον κόμβο, όπου επιχειρούμε να πάρουμε το κλείδωμα.
- OK όταν βρούμε τον κόμβο όπου εφαρμόζουμε έναν watcher στον znode προκειμένου να ενημερωθούμε σε περίπτωση που διαγραφεί.

Listing 5.3: Check For Master Method

```
void checkMaster() {
  zk.getData("/master", false, masterCheckCallback, null);
}

DataCallback masterCheckCallback = new DataCallback() {

  public void processResult(int rc, String path, Object ctx,
      byte[] data,
      Stat stat) {
    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      checkMaster();
      break;
    case NONODE:
      runForMaster();
      break;
    case OK:
      if (masterId.equals(new String(data))) {
        state = MasterStates.ELECTED;
        takeLeadership();
      } else {
        state = MasterStates.NOTELECTED;
        masterExists();
      }
      break;

    default:
      masterLog.error("Error when reading data...",
          KeeperException.create(Code.get(rc), path));
    }
  }
};
```

Τέλος υλοποιούμε τη μέθοδο *masterExists()* η οποία ελέγχει για την ύπαρξη του κόμβου αλλά σε αντίθεση με την προηγούμενη μέθοδο εφαρμόζει και έναν Watcher τον *masterExistsWatcher* στον κόμβο ώστε να ενημερώσει τον εξυπηρετητή σε περίπτωση που διαγραφεί ο κόμβος.

Listing 5.4: Add Watcher To Master

```
void masterExists() {
```

```
  zk.exists("/master", masterExistsWatcher,
      masterExistsCallback, null);
}

StatCallback masterExistsCallback = new StatCallback() {

  public void processResult(int rc, String path, Object ctx,
      Stat stat) {
    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      masterExists();
      break;
    case OK:
      break;
    case NONODE:
      state = MasterStates.RUNNING;
      runForMaster();
      masterLog.info("It seems like the previous master is gone
          ,"
          + "so I am running for master again...");
      break;
    default:
      checkMaster();
      break;
    }
  }
};

Watcher masterExistsWatcher = new Watcher() {
  public void process(WatchedEvent e) {
    if (e.getType() == EventType.NodeDeleted) {
      assert "/master".equals(e.getPath());

      runForMaster();
    }
  }
};
```

Διαχείριση των μεταδεδομένων από τον Master

Χρησιμοποιούμε την ασύγχρονη μέθοδο για τη δημιουργία των αρχείων στον ZooKeeper που θα χρειαστεί ο master . Οι επιπλέον κόμβοι που χρειαζόμαστε για την υλοποίηση του μοντέλου μας είναι :

- /servers : αποθηκεύονται τα στοιχεία (ταυτότητα και διεύθυνση IP) του κάθε διαθέσιμου εξυπηρετητή.
- /srvClients : κρατάει τον αριθμό των πελατών που εξυπηρετεί κάθε εξυπηρετητής.
- /srvTotalRuntime: κρατάει τον ολικό χρόνο για τον οποίο έτρεξε ένας

εξυπηρετητής, δηλαδή από την στιγμή που δημιουργήθηκε μέχρι την στιγμή που έκλεισε.

- /srvUpAvgTime: κρατάει τον μέσο χρόνο για τον οποίο έτρεξε ένας εξυπηρετητής για να εξυπηρετήσει έναν πελάτη, δηλαδή το άθροισμα του χρόνου των ερωτημάτων που έγιναν σε αυτόν τον εξυπηρετητή δια το πλήθος τους.

- /connections : αποθηκεύονται όλες οι διαθέσιμες συνδέσεις μεταξύ εξυπηρετητών και πελατών.

Οι κόμβοι αυτοί δημιουργούνται κάθε φορά που εκλέγεται ένας master και είναι επίμονοι. Παρακάτω δίνεται ο κώδικας για τη δημιουργία τους. Επειδή ο σκοπός είναι απλά να τους δημιουργήσουμε ώστε να μπορέσει στη συνέχεια να γράψει σε αυτούς ο εκάστοτε εξυπηρετητής δεν περνάμε κάποια δεδομένα, αλλά έναν άδειο πίνακα byte .

Listing 5.5: Creating the Metadata

```
public void bootstrap() {
  createParent("/servers", new byte[0]);
  createParent("/srvUpAvgTime", new byte[0]);
  createParent("/srvTotalRuntime", new byte[0]);
  createParent("/srvClients", new byte[0]);
  createParent("/connections", new byte[0]);
}

void createParent(String path, byte[] data) {
  zk.create(path,
      data,
      Ids.OPEN_ACL_UNSAFE,
      CreateMode.PERSISTENT,
      createParentCallback,
      data);
}

StringCallback createParentCallback = new StringCallback() {

  public void processResult(int rc, String path, Object ctx,
      String name) {
    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      createParent(path, (byte[]) ctx);
      break;
    case OK:
      masterLog.info("Parent Created");
      break;
    case NODEEXISTS:
      masterLog.info("Parent already registered: " + path);
      break;
    default:
      masterLog.error("Something went wrong: "
```

```
            + KeeperException.create(Code.get(rc), path));
        break;
      }
    }
};
```

Ο master δημιουργεί έναν Watcher στον κόμβο /servers , τον *serversChangeWatcher* ο οποίος τον ενημερώνει για οποιαδήποτε αλλαγή γίνεται στον αντίστοιχο κόμβο. Δηλαδή κάθε φορά που δημιουργείται κάποιος καινούριος εξυπηρετητής και κάθε φορά που διαγράφεται κάποιος. Όταν σηκώνεται κάποιος νέος εξυπηρετητής όπως θα δούμε και παρακάτω δημιουργεί κάτω από τον επίμονο κόμβο /servers έναν εφήμερο κόμβο παιδί με τα στοιχεία του. Ο serversChangeWatcher πιάνει αυτές τις αλλαγές και ενημερώνει τον master γυρίζοντας του μέσω του αντίστοιχου Callback μια λίστα με όλα τα παιδιά κάτω από τον /servers .

Listing 5.6: Create /servers znode Watcher

```
Watcher serversChangeWatcher = new Watcher() {
  public void process(WatchedEvent e) {
    if (e.getType() == EventType.NodeChildrenChanged) {
      assert "/servers".equals(e.getPath());
      getServers();
    }
  }
};

void getServers() {
  zk.getChildren("/servers", serversChangeWatcher,
      serversGetChildrenCallbck, null);
}

ChildrenCallback serversGetChildrenCallbck = new
    ChildrenCallback() {

  public void processResult(int rc, String path, Object ctx,
      List<String> children) {

    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      getServers();
      break;
    case OK:
      masterLog.info("Successfully got a list of servers: "
          + children.size() + " servers");
      updateServerMap(children);

      for (int i = 0; i < children.size(); i++) {
       watchSrvUpAvgTime("/srvUpAvgTime/"+children.get(i), ctx)
           ;
       watchSrvTotalRuntime("/srvTotalRuntime/"+children.get(i)
           , ctx);
```

```
        watchSrvClients("/srvClients/"+children.get(i), ctx);
      }

      break;
    default:
      masterLog.error("getChildren failed...",
          KeeperException.create(Code.get(rc), path));
    }
  }
};
```

Επιπλέον για κάθε κόμβο παιδί που βλέπει ο master δημιουργεί για αυτόν έναν Watcher για να λαμβάνει τις αντίστοιχες αλλαγές από τους άλλους κόμβους που θα γράφει ο εξυπηρετητής (/srvClients, / srvUpAvgTime, / srvTotalRuntime). Κάθε φορά που ο master λαμβάνει μια ειδοποίηση για κάποια αλλαγή σε κάποιον από τους παραπάνω κόμβους πρέπει να ανανεώνει τον Watcher , διότι έχει την ικανότητα να τον ειδοποιεί μόνο για ένα γεγονός και μετά διακόπτεται η λειτουργία του. Επομένως ο master πρέπει να τον ανανεώνει μετά από κάθε ειδοποίηση αν θέλει να υπάρχει συνεχής παρακολούθηση στους κόμβους.

Παρακάτω παρατίθεται ο κώδικας για τη δημιουργία του Watcher και των αντίστοιχων Callbacks για την παρακολούθηση του /srvUpAvgTime . Κατά αντιστοιχία γίνονται και για τους υπόλοιπους κόμβους.

Listing 5.7: Create /srvUpAvgTime znode Watcher and Callbacks

```
void watchSrvUpAvgTime(String path, Object ctx) {
  zk.exists(path, srvUpAvgTimeWatcher,
      existsSrvUpAvgTimeCallback, ctx);
}

Watcher srvUpAvgTimeWatcher = new Watcher() {

  public void process(WatchedEvent event) {
    if (event.getType() == EventType.NodeDataChanged) {
      assert event.getPath().contains("/srvUpAvgTime/server-");

      masterLog.info("[Master]: srvUpAvgTimeWatcher: Node Data
          Changed..."
              + event.getPath());
      ServerObj obj;
      obj = serversMap.get(event.getPath().replace("/
          srvUpAvgTime/",""));
      zk.getData(
          event.getPath(),
          false,
          getSrvUpAvgTimeDataCallback,
          obj);
    }
  }
};
```

```
StatCallback existsSrvUpAvgTimeCallback = new StatCallback() {
  public void processResult(int rc, String path, Object ctx,
      Stat stat) {
    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      watchSrvUpAvgTime(path, ctx);
      break;
    case OK:
      if (stat != null) {
        //zk.getData(path, false, getSrvUpAvgTimeDataCallback,
            ctx);
        //masterLog.info("[Master]: srvUpAvgTime node is there:
             " + path);
      }
      break;
    case NONODE:
      break;
    default:
      masterLog.error("[Master]: Something went wrong when "
          + "checking if the status node exists: "
          + KeeperException.create(Code.get(rc), path));
      break;
    }
  }
};

DataCallback getSrvUpAvgTimeDataCallback = new DataCallback() {

  public void processResult(int rc, String path, Object ctx,
      byte[] data,
      Stat stat) {
    switch (Code.get(rc)) {
    case CONNECTIONLOSS:
      ServerObj obj;
      obj = serversMap.get(path.replace("/srvUpAvgTime/", ""));
      zk.getData(path, false, getSrvUpAvgTimeDataCallback, obj)
          ;
      return;
    case OK:
      String serverKey = path.replace("/srvUpAvgTime/", "");
      double time = Double.parseDouble(new String(data));
      masterLog.info("[Master]: Server Up Average Time of " +
          serverKey + ": " + time);

      assert (ctx != null);
      serversMap.get(serverKey).setSrvUpAvgTime(time);

      watchSrvUpAvgTime(path, ctx);
      break;
    case NONODE:
      masterLog.warn("[Master]: " + path + " node is gone!");
      return;
    default:
```

```
            masterLog.error("[Master]: Something went wrong here, "
                + KeeperException.create(Code.get(rc), path));

        }
      }
    };
```

Επικοινωνία εξυπηρετητή με ZooKeeper

Οι εξυπηρετητές συνδέονται με τον ZooKeeper προκειμένου να αποθηκεύουν τον ολικό χρόνο για τον οποίο είναι ενεργοί, τον μέσο χρόνο για τον οποίο έτρεχαν αιτήματα από τους πελάτες τους και το πλήθος των πελατών που εξυπηρετούν. Τα στοιχεία αυτά τα διαβάζει στη συνέχεια ο master προκειμένου να κάνει τις αντίστοιχες ενέργειες για καταμερισμό φορτίου στο δίκτυο. Η σύνδεση με τον ZooKeeper είναι αντίστοιχη με αυτή του master . Ο εξυπηρετητής δημιουργεί μια συνεδρία την οποία κρατά για όσο είναι ενεργός. Αφού εδραιωθεί η συνεδρία, δημιουργεί εφήμερους κόμβους παιδιά κάτω από τους αντίστοιχους επίμονους κόμβους που έχει δημιουργήσει ο master (/servers, /srvTotalRuntime, /srvUpAvgTime, /srvClients) , όπου και αποθηκεύει τις αντίστοιχες πληροφορίες. Δηλαδή, στον κόμβο /servers αποθηκεύει την ταυτότητά του και την διεύθυνσή IP του, στον /srvTotalRuntime γράφει τον ολικό χρόνο που ήταν ενεργός, στον /srvTotalRuntime γράφει ανά τακτά χρονικά διαστήματα έναν μέσο χρόνο για τον οποίο εκτελούσε αιτήματα και στον /srvClients γράφει το πλήθος των πελατών που εξυπηρετεί. Αρχικά ο εξυπηρετητής γράφει στον /servers τα στοιχεία του και στη συνέχεια δημιουργεί στους υπόλοιπους κόμβους παιδιά τα οποία αρχικοποιεί με το 0 και στη συνέχεια αναβαθμίζει τα αντίστοιχα δεδομένα. Παρακάτω δίνονται οι μέθοδοι όπου φαίνεται η εγγραφή του εξυπηρετητή στον κόμβο /servers , η αρχικοποίηση του /σρ"λιεντς καθώς και η αναβάθμισή του όταν φτάσει ένας καινούριος πελάτης. Με αντίστοιχο τρόπο υλοποιούνται και οι μέθοδοι για τους άλλους κόμβους.

Listing 5.8: Register Server Methods

```
public void register() {
zk.create("/servers/"+name,
      (addr.getAddress()+":"+server.getPort()).toString().
          getBytes(),
      Ids.OPEN_ACL_UNSAFE,
      CreateMode.EPHEMERAL,
      createServerCallback,
      null);
registersrvUpAvgTime();
registerSrvTotalRuntime();
registerSrvClients();
}

StringCallback createServerCallback = new StringCallback() {
```

```
public void processResult (int rc, String path, Object ctx,
    String name) {
  switch(Code.get(rc)) {
  case CONNECTIONLOSS:
    register();
    break;
  case OK:
    serverLog.info("Registered successfully..."+serverId);
    break;
  case NODEEXISTS:
    serverLog.warn("Already registered..."+serverId);
    break;
  default:
    serverLog.error("Something went wrong"+KeeperException.
        create(Code.get(rc),path));
  }
}
};

public void registerSrvClients() {
  zk.create("/srvClients/"+name,
        "0".getBytes(),
        Ids.OPEN_ACL_UNSAFE,
        CreateMode.EPHEMERAL,
        createSrvClientsCallback,
        null);
}

StringCallback createSrvClientsCallback = new StringCallback()
    {
  public void processResult (int rc, String path, Object ctx,
      String name) {
    switch(Code.get(rc)) {
    case CONNECTIONLOSS:
      registerSrvClients();
      break;
    case OK:
      serverLog.info("Server clients node created successfully
          ..."+serverId);
      break;
    case NODEEXISTS:
      serverLog.warn("Clients node already exists..."+serverId)
          ;
      break;
    default:
      serverLog.error("Something went wrong"+KeeperException.
          create(Code.get(rc),path));
    }
  }
};


public void incrSrvTotalRuntime(long time) throws
    KeeperException, InterruptedException {
```

```
    String path = "/srvTotalRuntime/"+name;
    srvTotalRuntime += time;
    String timeStr = String.valueOf(srvTotalRuntime);
    zk.setData(path, timeStr.getBytes(), zk.exists(path, true).
        getVersion());
  }
```

5.4 Apache Avro

5.4.1 Ορισμός

Το Apache Avro είναι ένα σύστημα σειριοποίησης δεδομένων το οποίο αναπτύχθηκε από την Apache Software Foundation στα πλαίσια ενός άλλου μεγάλου της έργου του, Apache Hadoop . Έχει υλοποιηθεί για τις γλώσσες προγραμματισμού C, C++, Java, PHP, Python, και Ruby . Παρέχει πλούσιες δομές δεδομένων, ευκολία σειριοποίησης δεδομένων σε δυαδική μορφή, απομακρυσμένη κλήση μεθόδων και εύκολη ενσωμάτωση σε δυναμικές γλώσσες προγραμματισμού μέσω της δυνατότητας δυναμικής παραγωγής κώδικα [2].

5.4.2 Λειτουργία του Avro

Χρήση Σχημάτων

Η λειτουργία του Avro βασίζεται σε σχήματα (schemas). Τα σχήματα αυτά ορίζονται ως αρχεία μορφής JSON και μπορούν να αποτελούνται τόσο από πρωταρχικούς [2] όσο και από σύνθετους [3] τύπους δεδομένων [20] .

Ένα σχήμα έχει μια από τις παρακάτω μορφές :

- συμβολοσειρά τύπου JSON
- αντικείμενο τύπου JSON , της μορφής:
  {*"type": "typeName" ...attributes...*}
  όπου το typeName είναι ένας από τους τύπους που υποστηρίζει το Avro .
- πίνακας τύπου JSON , ο οποίος αντιπροσωπεύει μια ένωση από τους τύπους που χρησιμοποιούνται στο σχήμα.

Ένα απλό παράδειγμα σχήματος είναι το εξής:

Listing 5.9: Simple Schema Example

```
{"namespace": "example.avro",
 "type": "record",
 "name": "Hello",
```

[2] null, boolean, int, long, float, double, bytes, string
[3] record, enum, array, map, union

```
 "fields": [
     {"name": "clientID", "type": "string"},
     {"name": "address", "type": ["string", "null"]}
 ]
}
```

Το συγκεκριμένο σχήμα ορίζει μια εγγραφή για ένα αντικείμενο τύπου Hello το οποίο χρησιμοποιήθηκε στην διπλωματική εργασία για την ανταλλαγή μηνυμάτων μεταξύ των master και client . Σε μια εγγραφή το ελάχιστο που επιτρέπεται να ορίζεται είναι ο τύπος της("type": "record"), το όνομά της ("name": "Hello") και τα πεδία της, στο συγκεκριμένο παράδειγμα: το ID του πελάτη και η διεύθυνσή του. Επιπλέον ορίζουμε το όνομα του πακέτου (namespace": "example.avro") το οποίο μαζί με το πεδίο του ονόματος ("name": "Hello") συνθέτουν το πλήρες όνομα του σχήματος.

Τα πεδία της εγγραφής ορίζονται σε έναν πίνακα αντικειμένων, τα οποία χαρακτηρίζονται από ένα όνομα και τον τύπο τους. Παρατηρούμε ότι τα χαρακτηριστικά μιας εγγραφής είναι και αυτά αντικείμενα ενός σχήματος, δηλαδή μπορεί να είναι πρωταρχικοί ή σύνθετοι τύποι. Για παράδειγμα, παραπάνω έχουμε το ID του πελάτη που είναι μια απλή συμβολοσειρά (πρωταρχικός τύπος) και τη διεύθυνσή του, η οποία είναι μια ένωση (σύνθετος τύπος) που αναπαρίσταται ως πίνακας τύπου JSON . Οι ενώσεις είναι σύνθετοι τύποι, οι οποίοι μπορεί να είναι οποιασδήποτε μορφής από αυτές που αναγράφονται στον πίνακα. Για παράδειγμα, η διεύθυνση του πελάτη μπορεί να είναι είτε συμβολοσειρά είτε κενή. Ουσιαστικά έτσι δηλώνονται τα προαιρετικά πεδία.

Πρωταρχικοί Τύποι Σχημάτων

Οι πρωταρχικοί τύποι που υποστηρίζει το Avro είναι :

- null
- boolean
- int
- long
- float
- double
- bytes
- string

Οι τύποι αυτοί δεν έχουν συγκεκριμένα χαρακτηριστικά.

Σύνθετοι Τύποι Σχημάτων

Το Avro υποστηρίζει τους εξής σύνθετους τύπους [2]:

- records : έτσι δηλώνονται οι εγγραφές, αποτελούνται από τα εξής χαρακτηριστικά :
  - name : μια συμβολοσειρά τύπου JSON που δηλώνει το όνομα της εγγραφής
  - namespace : μια συμβολοσειρά τύπου JSON που συμπληρώνει το όνομα της εγγραφής
  - doc : μια προαιρετική συμβολοσειρά τύπου JSON η οποία αποτελεί μια σύντομη περιγραφή για την εγγραφή
  - aliases : ένας προαιρετικός πίνακας τύπου JSON με όλα τα διαφορετικά ονόματα που μπορεί να έχει αυτή η εγγραφή
  - fields : ένας πίνακας τύπου JSON με όλα τα πεδία της εγγραφής. Κάθε πεδίο έχει τα εξής χαρακτηριστικά :
    - name : μια συμβολοσειρά τύπου JSON που δηλώνει το όνομα του πεδίου
    - doc : μια προαιρετική συμβολοσειρά τύπου JSON η οποία αποτελεί μια σύντομη περιγραφή για το πεδίο
    - type : ένα αντικείμενο τύπου JSON το οποίο δηλώνει ένα άλλο σχήμα ή μια συμβολοσειρά τύπου JSON που δηλώνει κάποιον από του ήδη ορισμένους τύπους του Avro .
    - default: μια προκαθορισμένη τιμή για το πεδίο. Είναι προαιρετικό χαρακτηριστικό και οι επιτρεπτές τιμές εξαρτώνται από το τι τύπος είναι το πεδίο, για παράδειγμα ένας ακέραιος δεν μπορεί να έχει ως προκαθορισμένη τιμή έναν δεκαδικό αριθμό.
    - order: είναι προαιρετικό χαρακτηριστικό και καθορίζει τη στοίχιση του πεδίου. Επιτρεπτές τιμές είναι οι "ascending" (the default), "descending"ή "ignore".
    - aliases : ένας προαιρετικός πίνακας τύπου JSON με όλα τα διαφορετικά ονόματα που μπορεί να έχει αυτό το πεδίο.
- enums : έχουν τα εξής πεδία
  - name : μια συμβολοσειρά τύπου JSON που δηλώνει το όνομα της απαρίθμησης.
  - namespace : μια συμβολοσειρά τύπου JSON που συμπληρώνει το όνομα της απαρίθμησης
  - doc : μια προαιρετική συμβολοσειρά τύπου JSON η οποία αποτελεί μια σύντομη περιγραφή

  - aliases : ένας προαιρετικός πίνακας τύπου JSON με όλα τα διαφορετικά ονόματα που μπορεί να έχει αυτή η απαρίθμηση
  - symbols : ένας πίνακας τύπου JSON με όλες τις τιμές της απαρίθμησης. Κάθε τιμή πρέπει να είναι μοναδκή.

- arrays : υποστηρίζουν ένα μόνο χαρακτηριστικό :
  - items : είναι το σχήμα των αντικειμένων του πίνακα.

- maps : Τα κλειδιά του χάρτη είναι συμβολοσειρές και υποστηρίζουν ένα μόνο χαρακτηριστικό :
  - values : είναι το σχήμα των αντικειμένων του χάρτη.

- unions : αναπαρίστανται με πίνακες τύπου JSON , για παράδειγμα [¨νυλλ¨, ¨στρινγ¨] το οποίο δηλώνει ότι το αντικείμενο μπορεί να είναι είτε κενό είτε μια συμβολοσειρά.

- fixed : ο τύπος αυτός χρησιμοποιείται για τιμές που θέλουμε να αποθηκεύσουμε μόνιμα, ανεξαρτήτως του κύκλου ζωής του προγράμματος. Υποστηρίζει τα εξής χαρακτηριστικά :
  - name : μια συμβολοσειρά τύπου JSON που δηλώνει το όνομα του.
  - namespace : μια συμβολοσειρά τύπου JSON που συμπληρώνει το όνομα.
  - aliases : ένας προαιρετικός πίνακας τύπου JSON με όλα τα διαφορετικά ονόματα που μπορεί να έχει ο τύπος
  - size : ένας ακέραιος που δηλώνει το πλήθος των bytes που αναπαριστούν κάθε τιμή.

Χρήση Πρωτοκόλλων

Τα πρωτόκολλα στο Avro χρησιμοποιούνται ώστε να περιγράφονται οι διεπαφές για τον απομακρυσμένο έλεγχο των μεθόδων. Κάθε πρωτόκολλο έχει συγκεκριμένα ορίσματα, τα οποία είναι [2] :

- **namespace :** το όνομα του πακέτου μέσα στο οποίο εντάσσεται το σχήμα
- **protocol :** το όνομα του πρωτοκόλλου
- **doc :** μια περιγραφή για το πρωτόκολλο
- **types :** μια λίστα με τους ορισμούς των ονομαστικών τύπων που χρησιμοποιεί το πρωτόκολλα (records, enums, fixed,errors).
- **messages :** η μορφή των μνημάτων που θα ανταλλαχθούν. Τα οποία έχουν, και αυτά, τα δικά τους ορίσματα :

- **doc :** περιγραφή του μηνύματος
- **request:** μια λίστα από τα σχήματα-μεταβλητές που θα σταλθούν.
- **response :** το σχήμα της απάντησης
- **error schemas:** μια προαιρετική ένωση σχημάτων για τον διαχειρισμό των αποτυχιών.
- **μονόδρομη boolean μεταβλητή**

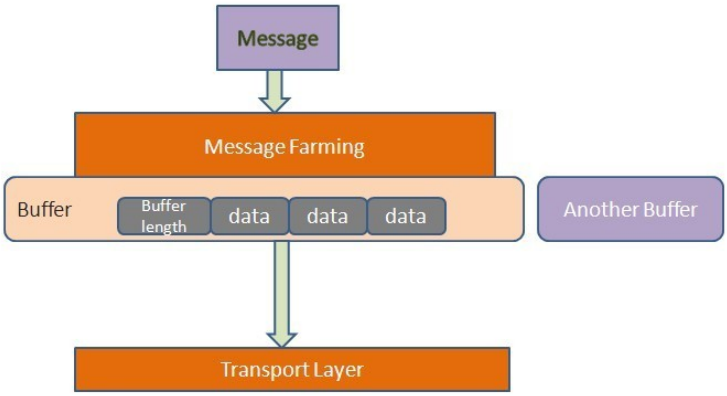
Παρακάτω βλέπουμε το σχήμα που χρησιμοποιήθηκε στη διπλωματική εργασία για να οριστεί το πρωτόκολλο ανταλλαγής μηνυμάτων ανάμεσα στον εξυπηρετητή και τον πελάτη.

Listing 5.10: Avro Schema For Server Client Message Exchange Protocol

```
{"namespace": "org.thesis.project.avro.rpc.protocol",
 "protocol": "ServerClient",
 "doc": "An exchange of data between server and client.",
 "types": [
           {"name": "Message", "type": "record",
            "fields":[{"name": "clientID", "type": "string"},
                      {"name": "status",   "type": "string"},
                      {"name": "timestamp","type": "string"},
                      {"name": "data","type": "string"} ]}],
 "messages": {
              "send":{"request":
                    [{"name":"message","type":"Message"}],
                      "response":"string"},
              "bye":{"request":[{"name":"bye","type":"string
                  "}],
                     "response":"string"}
  }
}
```

Μετάδοση Μηνυμάτων

Τα μηνύματα μπορούν να μεταδοθούν μέσω διαφόρων μηχανισμών, ενώ κατά τη μετάδοση το μήνυμα αναπαρίσταται ως μια αλληλουχία από bytes . Η μετάδοση των μηνυμάτων αποτελείται από δύο στάδια, τη μετάδοση ενός αιτήματος και αποδοχή της αντίστοιχης απάντησης. Ο μηχανισμός της ανταλλαγής μηνυμάτων εξαρτάται από το σύστημα που χρησιμοποιείται. Για παράδειγμα το πρωτόκολλο HTTP υποστηρίζει μηχανισμό για αιτήματα και απαντήσεις. Αλλά κατά τη μετάδοση μηνυμάτων πολλών clients που μιλούν σε ένα socket θα χρειαζόταν να δίνεται σε κάθε μήνυμα μια μοναδική ταυτότητα. Μια μετάδοση μπορεί να χαρακτηριστεί είτε ως stateless είτε ως stateful . Για το πρώτο είδος δεν χρειάζεται να έχει εδραιωθεί κάποια σύνδεση, σε αντίθεση με το δεύτερο είδος όπου η σύνδεση χρησιμεύει για την ανταλλαγή πολλών μηνυμάτων.



Σχήμα 5.5: Avro's Transport Message Layers

Πλαισίωση Μηνυμάτων

Τα μηνύματα πλαισιώνονται ως μια λίστα από buffers . Η πλαισίωση του μηνύματος είναι ένα στάδιο μετά τη δημιουργία του και πριν την μετάδοσή του [Σχήμα 5.5] και γίνεται με σκοπό τη βελτιστοποίηση κάποιων λειτουργιών. Ένα πλαισιωμένο μήνυμα αποτελείται από 4 bytes που δηλώνουν το μήκος του buffer ακολουθούμενο από τόσα bytes με τα δεδομένα του. Ενώ πάντα το μήνυμα τερματίζεται με έναν buffer μηδενικού μεγέθους.

Μέσω της πλαισίωσης οι αναγνώστες ενός μηνύματος μπορούν να λάβουν αποδοτικά διαφορετικούς buffers από διαφορετικές πηγές ενώ αντίστοιχα οι συγγραφείς του μηνύματος μπορούν να αποθηκεύσουν διαφορετικούς buffers σε διαφορετικούς προορισμούς. Συγκεκριμένα μειώνει αισθητά των αριθμό των φορών που αντιγράφονται μεγάλα δυαδικά αντικείμενα. Για παράδειγμα αν μια παράμετρος απομακρυσμένης κλήσης αποτελείται από μεγάλο όγκο δεδομένων (τάξης megabyte), τα δεδομένα αυτά μπορούν να αντιγραφούν κατευθείαν από έναν περιηγητή αρχείων σε ένα socket και στην άλλη άκρη να γραφούν κατευθείαν σε έναν περιηγητή αρχείων χωρίς να χρειαστεί να μπουν στον χώρο του χρήστη.

Χειραψία

Ο σκοπός της χειραψίας είναι να διασφαλιστεί ότι τόσο ο πελάτης όσο και ο εξυπηρετητής έχουν ο ένας τον ορισμό του πρωτοκόλλου του άλλου, ώστε από τη μια πλευρά ο πελάτης να μπορεί ορθά να αποσειριοποιεί τις απαντήσεις και από την άλλη ο εξυπηρετητής να μπορεί να αποσειριοποιεί σωστά τα αιτήματα. Τόσο ο πελάτης όσο και ο εξυπηρετητής πρέπει να κρατούν στην μνήμη τους τα πρωτόκολλα που είδαν πρόσφατα έτσι ώστε μια χειραψία να μπορεί να ολοκληρώνεται χωρίς επιπλέον επιβάρυνση του δικτύου ώστε να μεταδοθεί πάλι ολόκληρο το κείμενο του πρωτοκόλλου.

Τα αιτήματα και οι απαντήσεις απομακρυσμένων κλήσεων μεθόδων δεν επεξεργάζονται μέχρι να ολοκληρωθεί μια χειραψία. Στην stateless μετάδοση, όλα τα αιτήματα και οι απαντήσεις έχουν ως πρόθεμα μια χειραψία. Ενώ στην λατιντεξτ στατεφυλ μετάδοση η χειραψία μπαίνει ως πρόθεμα μέχρι μια επιτυχημένη απάντηση χειραψίας επιστραφεί από τη σύνδεση. Μετά από αυτό όλα τα πακέτα με τα αιτήματα και τις απαντήσεις ανταλλάσσονται χωρίς χειραψία για όσο παραμένει ενεργή η σύνδεση.παρ Το σχήμα για τη χειραψία είναι το εξής :

Listing 5.11: Avro Handshake Schema

```
{
  "type": "record",
  "name": "HandshakeRequest", "namespace":"org.apache.avro.ipc
      ",
  "fields": [
    {"name": "clientHash",
     "type": {"type": "fixed", "name": "MD5", "size": 16}},
    {"name": "clientProtocol", "type": ["null", "string"]},
    {"name": "serverHash", "type": "MD5"},
    {"name": "meta", "type": ["null", {"type": "map", "values":
        "bytes"}]}
  ]
}
{
  "type": "record",
  "name": "HandshakeResponse", "namespace": "org.apache.avro.
     ipc",
  "fields": [
    {"name": "match",
     "type": {"type": "enum", "name": "HandshakeMatch",
              "symbols": ["BOTH", "CLIENT", "NONE"]}},
    {"name": "serverProtocol",
     "type": ["null", "string"]},
    {"name": "serverHash",
     "type": ["null", {"type": "fixed", "name": "MD5", "size":
         16}]},
    {"name": "meta",
     "type": ["null", {"type": "map", "values": "bytes"}]}
  ]
}
```

5.4.3 Χρήση του Avro στην διπλωματική εργασία

Το Avro χρησιμοποιήθηκε στην διπλωματική εργασία για την επικοινωνία μεταξύ του πελάτη με τον master και για την επικοινωνία μεταξύ του πελάτη και του εξυπηρετητή.

Επικοινωνία πελάτη με master

Η επικοινωνία μεταξύ του πελάτη με τον master γίνεται μέσω του λατιντεξτ Μαστερ˝λιεντ πρωτοκόλλου [listing: 5.14]. Ο πελάτης επικοινωνεί με τον master δυο φορές, στην αρχή για να ζητήσει έναν εξυπηρετητή και στο τέλος, μετά το πέρας της επικοινωνίας με τον εξυπηρετητή για να ανακοινώσει την λήξη των διεργασιών του. Η επικοινωνία αυτή γίνεται μέσω μηνυμάτων που είναι ουσιαστικά εγγραφές του Avro .

Ο πελάτης στην αρχή της επικοινωνίας του με τον master στέλνει μια εγγραφή με τον όνομα Hello η οποία περιέχει την ταυτότητά του και τη διεύθυνση IP του μέσω της μεθόδου *hello(Hello record)* . Ο master όταν κληθεί συτή η μέθοδος κρατά τα στοιχεία του πελάτη, επιλέγει τον κατάλληλο εξυπηρετητή και απαντά με μια εγγραφή τύπου MasterResp , η οποία περιέχει την ταυτότητα, την διεύθυνση IP και την θύρα του εξυπηρετητή που επέλεξε.

Listing 5.12: Client's Hello Record

```
{"type": "record",
 "name": "Hello",
 "fields": [
      {"name": "clientID", "type": "string"},
      {"name": "address", "type": "string"}
       ]
}
```

Listing 5.13: Master's MasterResp Record

```
{"type": "record",
 "name": "MasterResp",
 "fields": [
      {"name": "serverID", "type": "string"},
      {"name": "serverIP", "type": "string"},
      {"name": "port",   "type": "string"}
      ]
}
```

Όταν ολοκληρωθούν οι διεργασίες του πελάτη και κλείσει την σύνδεσή του με τον εξυπηρετητή επικοινωνεί και πάλι με τον master για να τον ενημερώσει στέλνοντάς του απλά μια συμβολοσειρά με την ταυτότητά του μέσω της μεθόδου *bye(CharSequence clientId)* και ο master απαντάει πίσω για την επιτυχία της επικοινωνίας ή όχι.

Listing 5.14: Master-Client Avro Protocol

```
{"namespace": "org.thesis.project.avro.rpc.protocol",
 "protocol": "MasterClient",
 "doc": "An exchange of data between master and client.
     Client sends ID and address and Master sends
     server's address",
 "types": [
```

```
      {"name": "Hello", "type": "record",
       "fields": [
            {"name": "clientID", "type": "string"},
            {"name": "address", "type": "string"}"
            ]},
      {"name": "MasterResp", "type": "record",
       "fields": [
            {"name": "serverID", "type": "string"},
            {"name": "serverIP", "type": "string"},
            {"name": "port",   "type": "string"}"
            ]}
       ],
 "messages": {
        "hello": {
              "request": [{"name": "hello", "type": "Hello"}],
              "response": "MasterResp"},
        "bye": {
            "request": [{"name": "bye", "type": "string"}],
            "response": "string"}
      }
}
```

Επικοινωνία πελάτη με εξυπηρετητή

Η επικοινωνία μεταξύ πελάτη και εξυπηρετητή γίνεται μέσω του πρωτοκόλλου ServerClient το οποίο είδαμε σε προηγούμενο κεφάλαιο [listing 5.10]. Ο πελάτης στέλνει σε μια εγγραφή Avro , την Message καλώντας την μέθοδο *send(Message message))* . Η εγγραφή αυτή περιλαμβάνει τα εξής πεδία :

- clientId : είναι η ταυτότητα του πελάτη
- status : είναι ένα χαρακτηριστικό για το είδος του αιτήματος. Παίρνει τις τιμές INIT αν είναι η πρώτη επικοινωνία πελάτη εξυπηρετητή, SELECT , INSERT, UPDATE, DELETE αν το αίτημα προς τη βάση είναι μια από τις ομώνυμες εντολές.
- timestamp : ο χρόνος όταν έγινε το αίτημα από τον πελάτη. Αυτή η μεταβλητή χρησιμεύει ώστε να υπάρχει μια χρονική διάταξη ως προς τα αιτήματα που υλοποιεί ο εξυπηρετητής.
- data : δεδομένα τα οποία μπορεί να θέλει να στείλει ο πελάτης στη βάση.

Listing 5.15: Client's Message Record

```
{"type": "record", "name": "Message",
 "fields": [{"name": "clientID", "type": "string"},
      {"name": "status", "type": "string"},
      {"name": "timestamp", "type": "string"},
      {"name": "data", "type": "string"}]
}
```

Κεφάλαιο 6

Έλεγχος Λειτουργίας

6.1 Εγκατάσταση Περιβάλλοντος εκτέλεσης Java

Η έκδοση της Java η οποία χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία είναι η 1.8 και το αρχείο εγκατάστασης του JRE μπορεί να βρεθεί στην επίσημη ιστοσελίδα της. Ανάλογα με το λειτουργικό σύστημα ακολουθούμε και τις κατάλληλες οδηγίες που δίνονται προκειμένου να ολοκληρωθεί η εγκατάσταση του πακέτου. Μετά την ολοκλήρωση της εγκατάστασης είναι απαραίτητος ο ορισμός των μεταβλητών περιβάλλοντος όπως αυτές ορίζονται στην επίσημη ιστοσελίδα. Συγκεκριμένα :

- JAVA_HOME : Το μονοπάτι στο δίσκο που βρίσκεται αποθηκευμένο το αρχείο του Jdk .
- JRE_HOME : Το μονοπάτι στο δίσκο που βρίσκεται αποθηκευμένο το αρχείο του jre
- PATH : Προστίθεται στη μεταβλητή path του συστήματος το μονοπάτι προς τα εκτελέσιμα αρχεία (java, javac, κ.τ.λ.)
- CLASSPATH : Το μονοπάτι στο δίσκο που βρίσκονται εγκατεστημένες όλες οι βιβλιοθήκες και τα τύπου jar αρχεία που θα χρησιμοποιηθούν από το σύστημα.

6.2 Εγκατάσταση και Λειτουργία του Apache ZooKeeper

Για να εγκαταστήσουμε τον Apache ZooKeeper κατεβάζουμε την αντίστοιχη έκδοσή του από την επίσημη ιστοσελίδα του (http://zookeeper.apache.org) ακολουθώντας τους αντίστοιχους συνδέσμους ανάλογα με το λειτουργικό

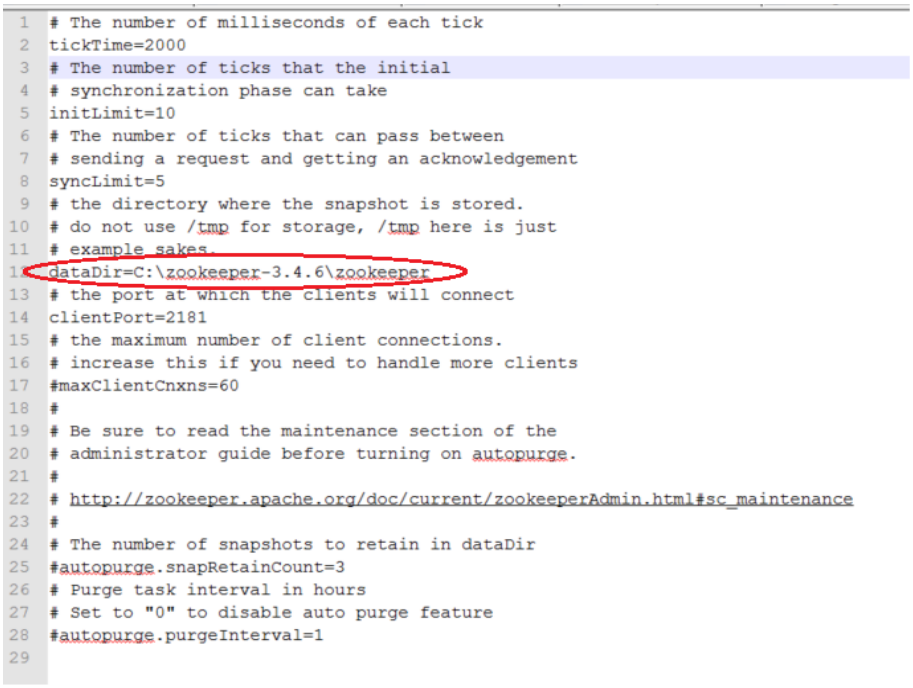
σύστημα στο οποίο πρόκειται να τον εγκαταστήσουμε. Στην διπλωματική εργασία χρησιμοποιήθηκε η έκδοση 3.4.6 του Apache ZooKeeper . Κατεβάζουμε στον υπολογιστή μας και αποσυμπιέζουμε τον φάκελο με την έκδοση που επιθυμούμε. Σε περίπτωση χρήσης Linux, Mac OS X ή οποιοδήποτε άλλου UNIX συστήματος κατεβάζουμε το αντίστοιχο .tar αρχείο το οποίο θα είναι της μορφής zookeeper-3.4.5.tar.gz και το αποσυμπιέζουμε με την εντολή *# tar -xvzf zookeeper-3.4.5.tar.gz* . Ενώ σε περίπτωση χρήσης Windows μπορούμε να χρησιμοποιήσουμε οποιοδήποτε εργαλείο αποσυμπίεσης. Το αρχείο περιλαμβάνει εκτός των άλλων και τους εξής φακέλους που θα χρειαστούμε:

- bin : περιέχει όλα τα scripts που θα χρειαστούμε για την εκκίνηση του Apache ZooKeeper . Τα αρχεία που έχουν κατάληξη .sh είναι για οποιαδήποτε τύπο UNIX συστήματος ενώ αυτά που λήγουν σε.cmd είναι για Windows .
- conf : Έχει όλα τα αρχεία παραμετροποίησης του ZooKeeper
- lib : περιέχει όλα τα εξωτερικά αρχεία τύπου jar που χρειάζονται για να λειτουργήσει σωστά ο Apache ZooKeeper .

Μετά την ολοκλήρωση της εγκατάστασης ορίζουμε την μεταβλητή περιβάλλοντος CLASSPATH του ZooKeeper ώστε να διευκολυνθεί η χρήση του συγκεκριμένου εργαλείου. Τη μεταβλητή αυτή την φτιάχνει το script *zkEnv* που βρίσκεται στον φάκελο bin . Εκτελούμε μια από τις παρακάτω σειρές εντολών ανάλογα σε ποιο λειτουργικό βρισκόμαστε :

UNIX	**Windows**
ZOOBINDIR="$< path\_to\_zook\_file > /bin$" ."$ZOOBINDIR"/zkEnv.sh	set ZOOBINDIR="$< path_to_zook_file > /bin$" call %"$ZOOBINDIR"%.cmd

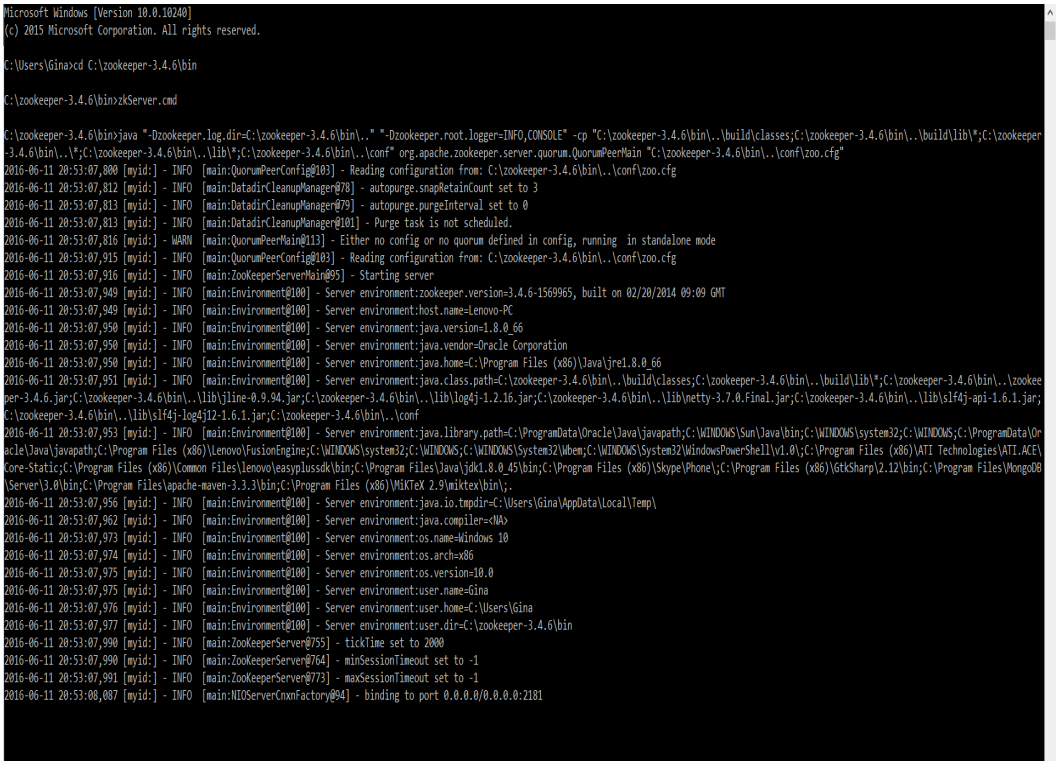
Τέλος φτιάχνουμε έναν φάκελο όπου θα γράφει ο ZooKeeper τα δεδομένα του και ενημερώνουμε τη μεταβλητή dataDir στο αρχείο zoo.cfg με το μονοπάτι τοποθεσίες του φακέλου.



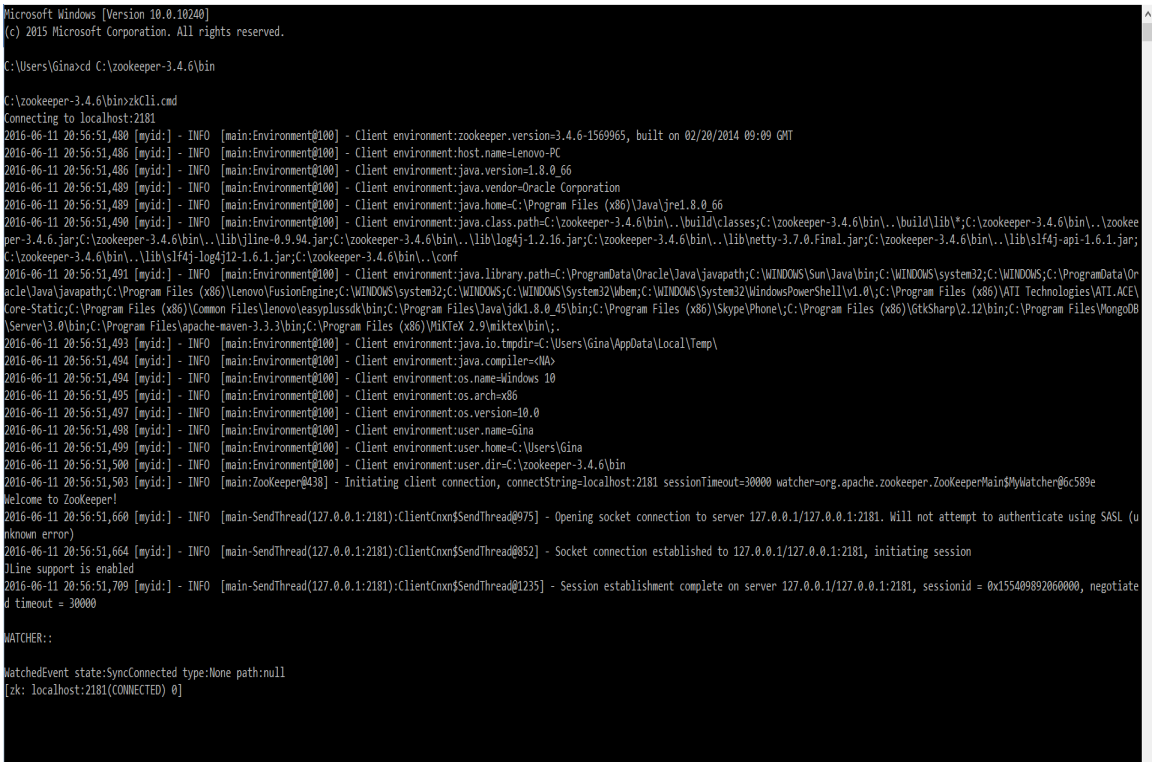
```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=C:\zookeeper-3.4.6\zookeeper
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1

```

Σχήμα 6.1: ZooKeeper Configuration File

Ολοκληρώνοντας τη διαδικασία εγκατάστασης μέσα στο φάκελο bin μπορούμε να βρούμε τα scripts : *zkCli, zkServer* τα οποία χρησιμοποιούμε για να τρέξουμε τον ZooKeeper . Το *zkCli* ανοίγει μια διεπαφή στην οποία μπορούμε να εξοικειωθούμε με τις εντολές του τον ZooKeeper καθώς και να το χρησιμοποιήσουμε αργότερα για έλεγχο ότι όντος έχουν δημιουργηθεί οι απαραίτητοι κόμβοι στο σύστημα. Ενώ ο *zkServer* είναι αυτός που ενεργοποιεί το σύστημα.



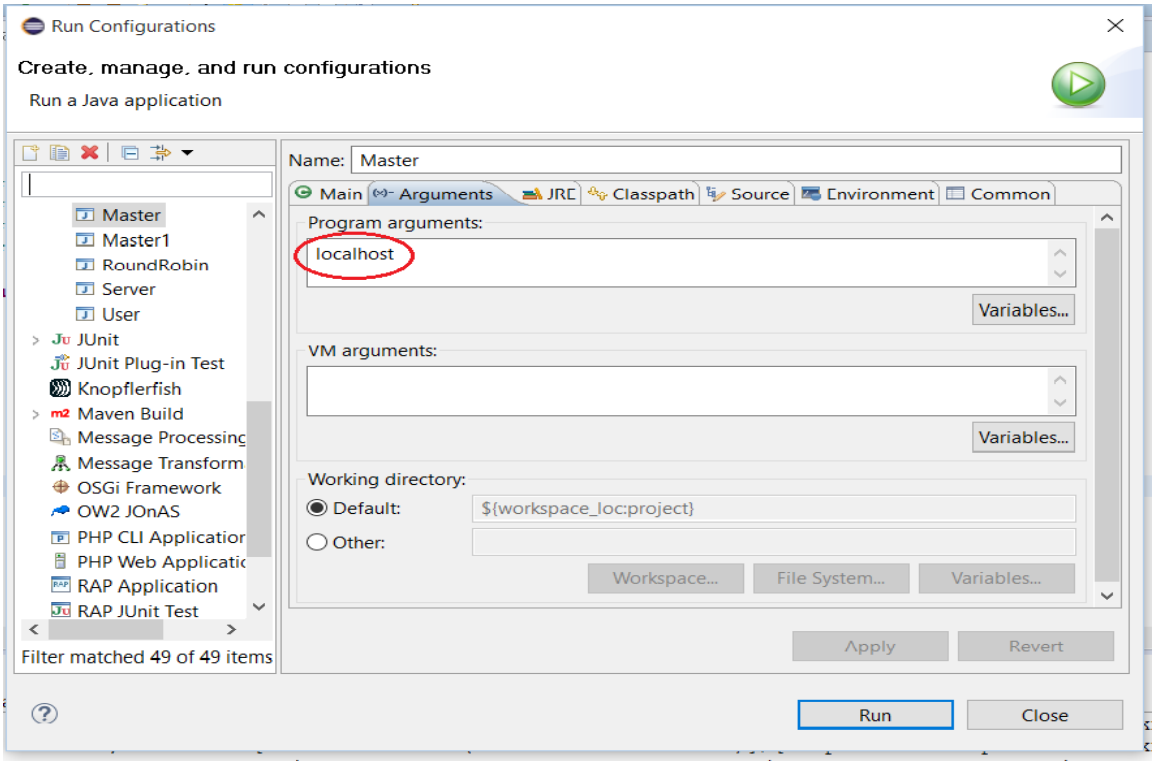
Σχήμα 6.2: Running ZooKeeper Server



Σχήμα 6.3: Running ZooKeeper Client

6.3 Λειτουργία Master

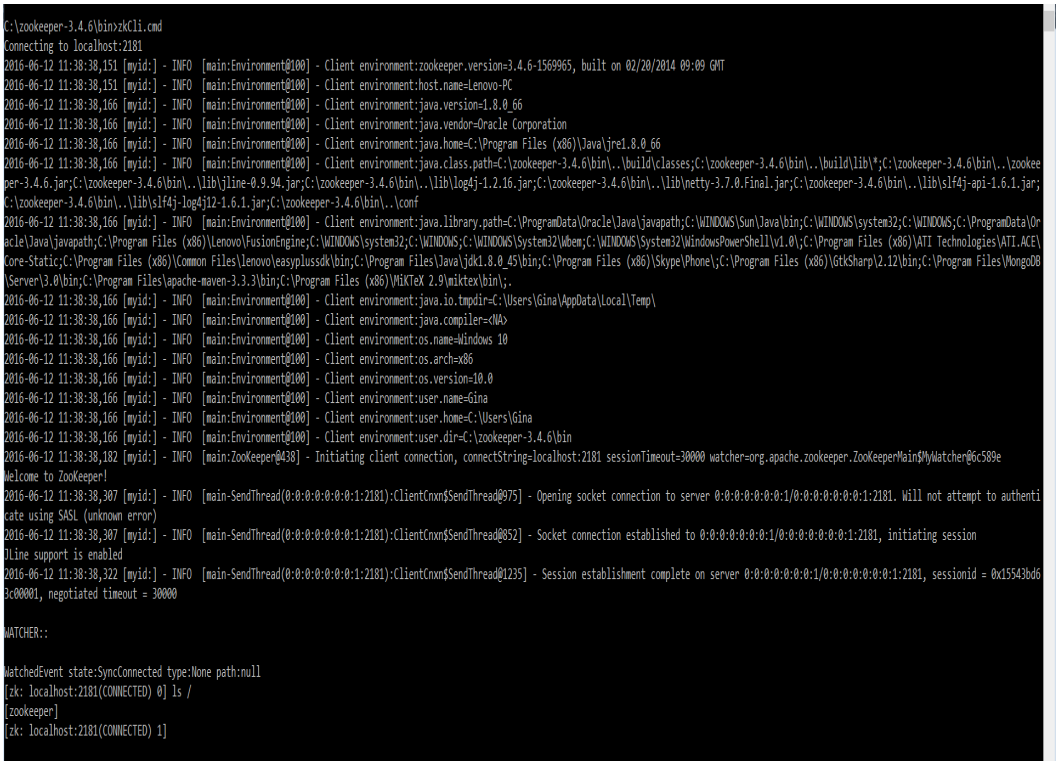
Αφού ενεργοποιήσουμε τον ZooKeeper επόμενο βήμα είναι να σηκώσουμε έναν Master ώστε να διαχειρίζεται τους εξυπηρετητές. Τον τρέχουμε όπως οποιαδήποτε άλλη εφαρμογή Java και παίρνει σαν όρισμα την τοποθεσία που τρέχει ο ZooKeeper .



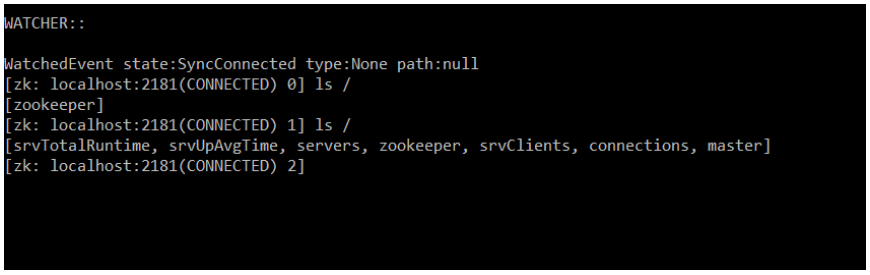
Σχήμα 6.4: Master Configuration

6.3.1 Σύνδεση με ZooKeeper

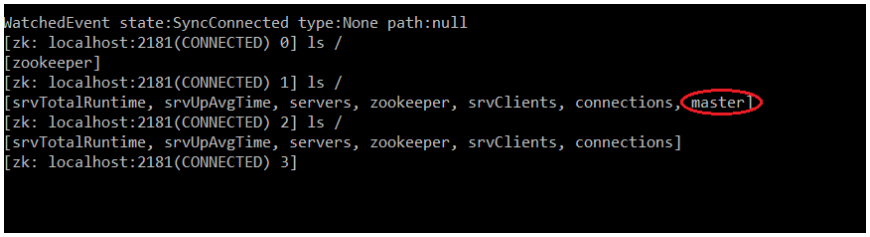
Μόλις ενεργοποιηθεί ο Master συνδέεται με τον ZooKeeper και δημιουργεί τον κόμβο-κλείδωμα (/master) για να δηλώσει την εκλογή του καθώς και τους απαραίτητους κόμβους για τη διαχείριση του συστήματος. Ενώ σε περίπτωση σφάλματος, αν διακοπεί η λειτουργία του παρατηρούμε πως διαγράφεται ο κόμβος κλειδί ενώ διατηρούνται όλοι οι υπόλοιποι ώστε να μη χαθούν δεδομένα. Στις παρακάτω εικόνες με σειρά από πάνω προς τα κάτω φαίνεται πώς είναι η κατάσταση του ZooKeeper πριν συνδεθεί ο Master [Σχήμα: 6.5], τι γίνεται αφού συνδεθεί [Σχήμα: 6.6] και τέλος τι γίνεται αφού αποσυνδεθεί [Σχήμα: 6.7] όπου παρατηρούμε πως έχει σβηστεί μόνο ο κόβος-κλειδί.



Σχήμα 6.5: ZooKeeper State Before Master's Connection



Σχήμα 6.6: ZooKeeper State After Master's Connection



Σχήμα 6.7: ZooKeeper State After Master's Disconnection

Παρακάτω δίνονται αποσπάσματα από το αρχείο Log του Master για τις ενέργειες που κάνει κατά τη σύνδεσή του.

Listing 6.1: Master Connection Log

```
1.  2016-06-12 11:43:15,252INFO -[main]/[org.thesis.project.master.Master@82
    ] - Starting Master-188e2659-850d-4ec2-aae7-b6eb1eda16f6...
2.  2016-06-12 11:43:15,268INFO -[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@975] - Opening socket connection
    to server 0:0:0:0:0:0:0:1/0:0:0:0:0:0:0:1:2181. Will not attempt to
    authenticate using SASL (unknown error)
3.  2016-06-12 11:43:15,268INFO -[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@852] - Socket connection
    established to 0:0:0:0:0:0:0:1/0:0:0:0:0:0:0:1:2181, initiating session
4.  2016-06-12 11:43:15,268DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@892] - Session establishment
    request sent on 0:0:0:0:0:0:0:1/0:0:0:0:0:0:0:1:2181
5.  2016-06-12 11:43:15,283INFO -[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@1235] - Session establishment
    complete on server 0:0:0:0:0:0:0:1/0:0:0:0:0:0:0:1:2181, sessionid = 0
    x15543bd63c00004, negotiated timeout = 15000
6.  2016-06-12 11:43:15,283INFO -[main-EventThread]/[org.thesis.project.
    master.Master@136] - Processing event: WatchedEvent state:SyncConnected
    type:None path:null
7.  2016-06-12 11:43:15,741INFO -[main]/[org.thesis.project.master.Master@85
    ] - Master listens to : 8081
8.  2016-06-12 11:43:15,741INFO -[main]/[org.thesis.project.master.
    Master@232] - Running for master...
9.  2016-06-12 11:43:15,756DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 1,1  replyHeader:: 1,6485,0  request:: '/servers
    ,,v{s{31,s{'world,'anyone}}},0  response:: '/servers
10. 2016-06-12 11:43:15,756DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/srvUpAvgTime serverPath:/
    srvUpAvgTime finished:false header:: 2,1  replyHeader:: 2,6486,0  request
    :: '/srvUpAvgTime,,v{s{31,s{'world,'anyone}}},0  response:: '/
    srvUpAvgTime
11. 2016-06-12 11:43:15,756DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/srvTotalRuntime serverPath:/
    srvTotalRuntime finished:false header:: 3,1  replyHeader:: 3,6487,0
    request:: '/srvTotalRuntime,,v{s{31,s{'world,'anyone}}},0  response:: '/
    srvTotalRuntime
12. 2016-06-12 11:43:15,756DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/srvClients serverPath:/srvClients
    finished:false header:: 4,1  replyHeader:: 4,6488,0  request:: '/
    srvClients,,v{s{31,s{'world,'anyone}}},0  response:: '/srvClients
13. 2016-06-12 11:43:15,756DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/connections serverPath:/
    connections finished:false header:: 5,1  replyHeader:: 5,6489,0  request
```

```
    :: '/connections,,v{s{31,s{'world,'anyone}}},0  response:: '/connections
14. 2016-06-12 11:43:15,772DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/master serverPath:/master finished
    :false header:: 6,1  replyHeader:: 6,6490,0  request:: '/master
    ,#31383865323635392
    d383530642d346563322d616165372d623665623165646131366636,v{s{31,s{'world,'
    anyone}}},1  response:: '/master
15. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$1@189] - Parent Created
16. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$1@189] - Parent Created
17. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$1@189] - Parent Created
18. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$1@189] - Parent Created
19. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$1@189] - Parent Created
20. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master@334] - Going for list of servers
21. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$2@259] - I'm the leader 188e2659-850d-4ec2-aae7-
    b6eb1eda16f6
22. 2016-06-12 11:43:15,772DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00004, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 7,8  replyHeader:: 7,6490,0  request:: '/servers,
    T  response:: v{}
23. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master$7@390] - Successfully got a list of servers: 0 servers
24. 2016-06-12 11:43:15,772INFO -[main-EventThread]/[org.thesis.project.
    master.Master@650] - [Master]: Updating servers' HashMap...
25. 2016-06-12 11:43:20,784DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@717] - Got ping response for
    sessionid: 0x15543bd63c00004 after 0ms
```

Στο Σχήμα: 6.1 φαίνεται η αρχική διαδικασία που ακολουθεί ο Master για τη σύνδεση του με τον ZooKeeper . Αρχικά [γραμμή 1-6] εδραιώνεται η συνεδρία μεταξύ των δύο. Στη συνέχεια [γραμμή 7] ενεργοποιείται το Avro κομμάτι του Master και πιάνει μια θύρα να ακούει για μηνύματα. Έπειτα δημιουργεί τους κόμβους για τη διαχείριση των εξυπηρετητών [γραμμές 9-19], [Σχήμα: 6.6]. Και τέλος παίρνει τη λίστα των διαθέσιμων εξυπηρετητών, που λόγω του ότι ο Master είναι ο πρώτος που δημιουργήθηκε στο συγκεκριμένο παράδειγμα είναι ακόμα κενή.

6.3.2 Σύνδεση με εξυπηρετητές

Στη συνέχεια δίνονται αποσπάσματα από το αρχείο Log του Master για τη διαδικασία σύνδεσης και αποσύνδεσης τριών εξυπηρετητών καθώς και φωτογραφίες με την κατάσταση του ZooKeeper κατά τη διαδικασία αυτή.

Listing 6.2: Master-Servers Connection Log

```
------------------- 1 SERVER CONNECTED -------------------
1   2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeChildrenChanged path:/servers for sessionid 0
    x15543bd63c00009
2   2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 8,8  replyHeader:: 8,6544,0  request:: '/servers,
    T  response:: v{'server-e46e8c60-902d-4c04-8b59-c7ad25521961}
3   2016-06-12 14:01:09,216INFO -[main-EventThread]/[org.thesis.project.
    master.Master$7@390] - Successfully got a list of servers: 1 servers
4   2016-06-12 14:01:09,216INFO -[main-EventThread]/[org.thesis.project.
    master.Master@650] - [Master]: Updating servers' HashMap...
5   2016-06-12 14:01:09,232DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers/server-e46e8c60-902d-4c04
    -8b59-c7ad25521961 serverPath:/servers/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961 finished:false header:: 9,4  replyHeader:: 9,6544,0  request
    :: '/servers/server-e46e8c60-902d-4c04-8b59-c7ad25521961,F  response:: #6
    c6f63616c686f73742f3132372e302e302e313a3630343931,s
    {6541,6541,1465729269201,1465729269201,0,0,0,96057447759740938,25,0,6541}
6   2016-06-12 14:01:09,232DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvUpAvgTime/server-e46e8c60-902d
    -4c04-8b59-c7ad25521961 serverPath:/srvUpAvgTime/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 finished:false header:: 10,3  replyHeader::
    10,6544,0  request:: '/srvUpAvgTime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6542,6542,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6542}
7   2016-06-12 14:01:09,232DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 serverPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 finished:false header:: 11,3  replyHeader::
    11,6544,0  request:: '/srvTotalRuntime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6543,6543,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6543}
8   2016-06-12 14:01:09,232DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvClients/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 serverPath:/srvClients/server-e46e8c60-902d-4c04-8
    b59-c7ad25521961 finished:false header:: 12,3  replyHeader:: 12,6544,0
    request:: '/srvClients/server-e46e8c60-902d-4c04-8b59-c7ad25521961,T
    response:: s
    {6544,6544,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6544}
9   2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@835] - [Master]: --SERVERS-- total 1
```

```
10  2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-e46e8c60-902d-4c04-8b59-c7ad25521961
     IP: localhost/127.0.0.1:60491 Clients: 0.0 Server Up Time0.0

------------------- 2 SERVERS CONNECTED -------------------
11  2016-06-12 14:03:00,901DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 13,8  replyHeader:: 13,6550,0  request:: '/
    servers,T  response:: v{'server-e46e8c60-902d-4c04-8b59-c7ad25521961,'
    server-422b3273-d14c-4761-990f-b8e855ab362c}
12  2016-06-12 14:03:00,901INFO -[main-EventThread]/[org.thesis.project.
    master.Master$7@390] - Successfully got a list of servers: 2 servers
13  2016-06-12 14:03:00,901INFO -[main-EventThread]/[org.thesis.project.
    master.Master@650] - [Master]: Updating servers' HashMap...
14  2016-06-12 14:03:00,901DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers/server-422b3273-d14c
    -4761-990f-b8e855ab362c serverPath:/servers/server-422b3273-d14c-4761-990
    f-b8e855ab362c finished:false header:: 14,4  replyHeader:: 14,6550,0
    request:: '/servers/server-422b3273-d14c-4761-990f-b8e855ab362c,F
    response:: #6c6f63616c686f73742f3132372e302e302e313a3630353138,s
    {6547,6547,1465729380887,1465729380887,0,0,0,96057447759740939,25,0,6547}
15  2016-06-12 14:03:00,901DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvUpAvgTime/server-e46e8c60-902d
    -4c04-8b59-c7ad25521961 serverPath:/srvUpAvgTime/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 finished:false header:: 15,3  replyHeader::
    15,6550,0  request:: '/srvUpAvgTime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6542,6542,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6542}
16  2016-06-12 14:03:00,917INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@835] - [Master]: --SERVERS-- total 2
17  2016-06-12 14:03:00,917INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-422b3273-d14c-4761-990f-b8e855ab362c
     IP: localhost/127.0.0.1:60518 Clients: 0.0 Server Up Time0.0
18  2016-06-12 14:03:00,917DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 serverPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 finished:false header:: 16,3  replyHeader::
    16,6550,0  request:: '/srvTotalRuntime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6543,6543,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6543}
19  2016-06-12 14:03:00,917DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvClients/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 serverPath:/srvClients/server-e46e8c60-902d-4c04-8
    b59-c7ad25521961 finished:false header:: 17,3  replyHeader:: 17,6550,0
    request:: '/srvClients/server-e46e8c60-902d-4c04-8b59-c7ad25521961,T
    response:: s
    {6544,6544,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6544}
20  2016-06-12 14:03:00,917DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
```

```
    x15543bd63c00009, packet:: clientPath:/srvUpAvgTime/server-422b3273-d14c
    -4761-990f-b8e855ab362c serverPath:/srvUpAvgTime/server-422b3273-d14c
    -4761-990f-b8e855ab362c finished:false header:: 18,3  replyHeader::
    18,6550,0  request:: '/srvUpAvgTime/server-422b3273-d14c-4761-990f-
    b8e855ab362c,T  response:: s
    {6548,6548,1465729380887,1465729380887,0,0,0,96057447759740939,1,0,6548}
21  2016-06-12 14:03:00,917DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvTotalRuntime/server-422b3273-
    d14c-4761-990f-b8e855ab362c serverPath:/srvTotalRuntime/server-422b3273-
    d14c-4761-990f-b8e855ab362c finished:false header:: 19,3  replyHeader::
    19,6550,0  request:: '/srvTotalRuntime/server-422b3273-d14c-4761-990f-
    b8e855ab362c,T  response:: s
    {6549,6549,1465729380887,1465729380887,0,0,0,96057447759740939,1,0,6549}
22  2016-06-12 14:03:00,917INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-e46e8c60-902d-4c04-8b59-c7ad25521961
     IP: localhost/127.0.0.1:60491 Clients: 0.0 Server Up Time0.0

------------------- 3 SERVERS CONNECTED -------------------

23  2016-06-12 14:03:11,167DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 21,8  replyHeader:: 21,6556,0  request:: '/
    servers,T  response:: v{'server-e46e8c60-902d-4c04-8b59-c7ad25521961,'
    server-422b3273-d14c-4761-990f-b8e855ab362c,'server-1101feef-3388-4e25
    -833d-0f1998587e18}
24  2016-06-12 14:03:11,167INFO -[main-EventThread]/[org.thesis.project.
    master.Master$7@390] - Successfully got a list of servers: 3 servers
25  2016-06-12 14:03:11,167INFO -[main-EventThread]/[org.thesis.project.
    master.Master@650] - [Master]: Updating servers' HashMap...
26  2016-06-12 14:03:11,167DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers/server-1101feef-3388-4e25
    -833d-0f1998587e18 serverPath:/servers/server-1101feef-3388-4e25-833d-0
    f1998587e18 finished:false header:: 22,4  replyHeader:: 22,6556,0
    request:: '/servers/server-1101feef-3388-4e25-833d-0f1998587e18,F
    response:: #6c6f63616c686f73742f3132372e302e302e313a3630353431,s
    {6553,6553,1465729391152,1465729391152,0,0,0,96057447759740940,25,0,6553}
27  2016-06-12 14:03:11,183INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@835] - [Master]: --SERVERS-- total 3
28  2016-06-12 14:03:11,183DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvUpAvgTime/server-e46e8c60-902d
    -4c04-8b59-c7ad25521961 serverPath:/srvUpAvgTime/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 finished:false header:: 23,3  replyHeader::
    23,6556,0  request:: '/srvUpAvgTime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6542,6542,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6542}
29  2016-06-12 14:03:11,183INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-1101feef-3388-4e25-833d-0f1998587e18
     IP: localhost/127.0.0.1:60541 Clients: 0.0 Server Up Time0.0
30  2016-06-12 14:03:11,183INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-422b3273-d14c-4761-990f-b8e855ab362c
```

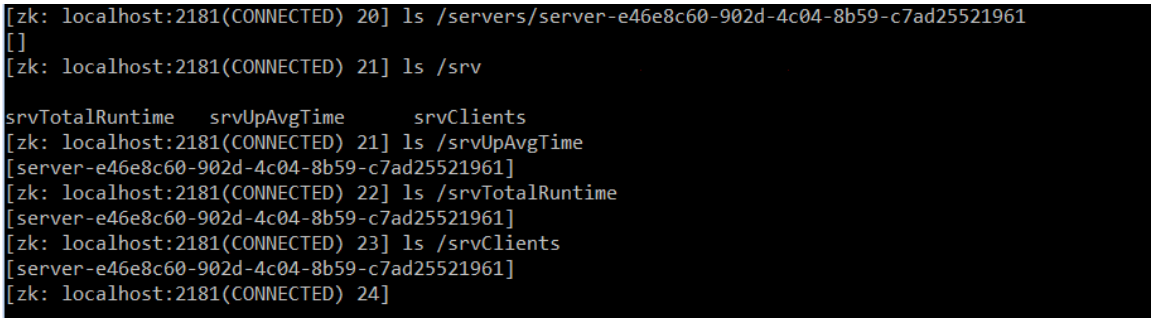
```
     IP: localhost/127.0.0.1:60518 Clients: 0.0 Server Up Time0.0
31   2016-06-12 14:03:11,183DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 serverPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 finished:false header:: 24,3  replyHeader::
    24,6556,0  request:: '/srvTotalRuntime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6543,6543,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6543}
32   2016-06-12 14:03:11,183INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-e46e8c60-902d-4c04-8b59-c7ad25521961
     IP: localhost/127.0.0.1:60491 Clients: 0.0 Server Up Time0.0

------------------- 1 SERVER DISCONNECTED -------------------

33   2016-06-12 14:05:32,012DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 32,8  replyHeader:: 32,6557,0  request:: '/
    servers,T  response:: v{'server-e46e8c60-902d-4c04-8b59-c7ad25521961,'
    server-422b3273-d14c-4761-990f-b8e855ab362c}
34   2016-06-12 14:05:32,012INFO -[main-EventThread]/[org.thesis.project.
    master.Master$7@390] - Successfully got a list of servers: 2 servers
35   2016-06-12 14:05:32,013INFO -[main-EventThread]/[org.thesis.project.
    master.Master@650] - [Master]: Updating servers' HashMap...
36   2016-06-12 14:05:32,015DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvUpAvgTime/server-e46e8c60-902d
    -4c04-8b59-c7ad25521961 serverPath:/srvUpAvgTime/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 finished:false header:: 33,3  replyHeader::
    33,6557,0  request:: '/srvUpAvgTime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6542,6542,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6542}
37   2016-06-12 14:05:32,017DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 serverPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 finished:false header:: 34,3  replyHeader::
    34,6557,0  request:: '/srvTotalRuntime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,T  response:: s
    {6543,6543,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6543}
38   2016-06-12 14:05:32,019DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvClients/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 serverPath:/srvClients/server-e46e8c60-902d-4c04-8
    b59-c7ad25521961 finished:false header:: 35,3  replyHeader:: 35,6557,0
    request:: '/srvClients/server-e46e8c60-902d-4c04-8b59-c7ad25521961,T
    response:: s
    {6544,6544,1465729269201,1465729269201,0,0,0,96057447759740938,1,0,6544}
39   2016-06-12 14:05:32,021DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvUpAvgTime/server-422b3273-d14c
    -4761-990f-b8e855ab362c serverPath:/srvUpAvgTime/server-422b3273-d14c
    -4761-990f-b8e855ab362c finished:false header:: 36,3  replyHeader::
```

```
    36,6557,0  request:: '/srvUpAvgTime/server-422b3273-d14c-4761-990f-
    b8e855ab362c,T  response:: s
    {6548,6548,1465729380887,1465729380887,0,0,0,96057447759740939,1,0,6548}
40  2016-06-12 14:05:32,023DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvTotalRuntime/server-422b3273-
    d14c-4761-990f-b8e855ab362c serverPath:/srvTotalRuntime/server-422b3273-
    d14c-4761-990f-b8e855ab362c finished:false header:: 37,3  replyHeader::
    37,6557,0  request:: '/srvTotalRuntime/server-422b3273-d14c-4761-990f-
    b8e855ab362c,T  response:: s
    {6549,6549,1465729380887,1465729380887,0,0,0,96057447759740939,1,0,6549}1

41  2016-06-12 14:05:32,024DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/srvClients/server-422b3273-d14c
    -4761-990f-b8e855ab362c serverPath:/srvClients/server-422b3273-d14c
    -4761-990f-b8e855ab362c finished:false header:: 38,3  replyHeader::
    38,6557,0  request:: '/srvClients/server-422b3273-d14c-4761-990f-
    b8e855ab362c,T  response:: s
    {6550,6550,1465729380887,1465729380887,0,0,0,96057447759740939,1,0,6550}
------------------- ALL SERVERS DISCONNECTED
    -------------------------------------------------

42  2016-06-12 14:06:46,017DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00009, packet:: clientPath:/servers serverPath:/servers
    finished:false header:: 43,8  replyHeader:: 43,6559,0  request:: '/
    servers,T  response:: v{}
43  2016-06-12 14:06:46,017INFO -[main-EventThread]/[org.thesis.project.
    master.Master$7@390] - Successfully got a list of servers: 0 servers
44  2016-06-12 14:06:46,018INFO -[main-EventThread]/[org.thesis.project.
    master.Master@650] - [Master]: Updating servers' HashMap...
```

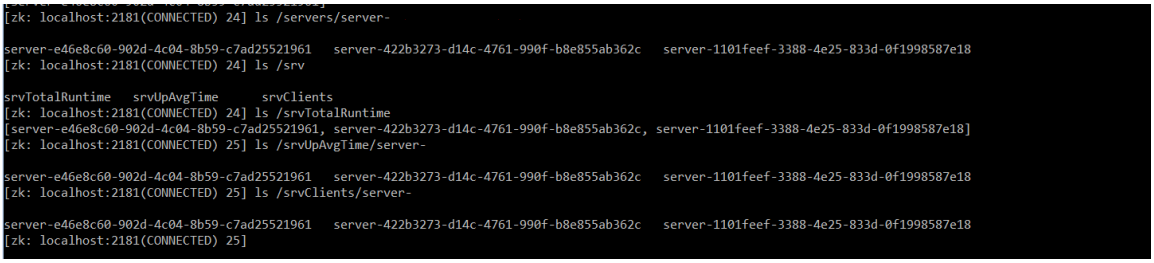
Στις γραμμές 1-2 βλέπουμε ότι ο Master ενημερώνεται από τον ZooKeeper για την αλλαγή που υπήρχε στον κόμβο /servers όταν εγγράφτηκε ένας νέος εξυπηρετητής. Στη συνέχεια ο Master ξεκινά τη διαδικασία ενημέρωσης του χάρτη που κρατάει τα στοιχεία των εξυπηρετητών [γραμμές 3-5]. Δέχεται απάντηση από τον ZooKeeper για την διεύθυνση IP του συγκεκριμένου εξυπηρετητή [γραμμή 6] καθώς επίσης και ενημερώσεις για το τι έγραψε ο εξυπηρετητής στους κόμβους /srvClients, /srvUpAvgTime, /srvTotalRuntime [γραμμές 7-9]. Ενώ στη γραμμή 10 μπορούμε να δούμε τον αναβαθμισμένο χάρτη με τα νέα στοιχεία.

Στα κομμάτια του αρχείου για τους δυο και τρεις εξυπηρετητές τα μηνύματα είναι τα ίδια με τα παραπάνω. Δηλαδή, ένα μήνυμα για την εγγραφή των νέων εξυπηρετητών [γραμμές 14, 29], η εκκίνηση της διαδικασίας ενημέρωσης του χάρτη [γραμμές 15-16, 30-31] με τον Master να λαμβάνει από τον ZooKeeper την ανανεωμένη λίστα εξυπηρετητών, αρχικά με 2 και μετά με 3 εγγραφές. Επειδή η επικοινωνία είναι ασύγχρονη παρατηρούμε πως την ώρα που έχουμε πει στον Master να εμφανίσει την λίστα με τους εξυπηρετητές [γραμμή 17] έρχονται οι ειδοποιήσεις από τον ZooKeeper για την αλλαγή στους υπόλοιπους κόμβους

[γραμμές 18-21] και μετά τυπώνονται τα στοιχεία του νέου εξυπηρετητή [γραμμή 22]. Αντίστοιχα ανακατεμένα είναι και τα μηνύματα για τον τρίτο εξυπηρετητή, αλλά μπορούμε να δούμε τελικά ότι τα στοιχεία και των τριών εξυπηρετητών φαίνονται στις γραμμές 29, 30 και 32.

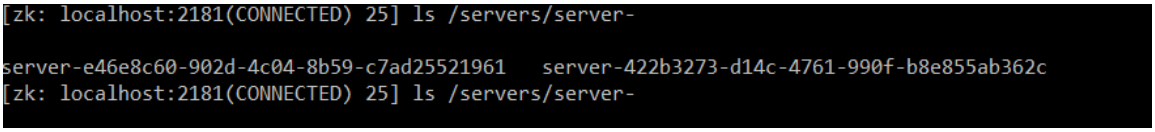


Σχήμα 6.8: One Server Connected

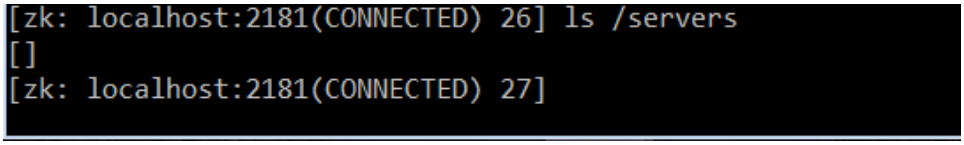


Σχήμα 6.9: Three Servers Connected

Στο επόμενο κομμάτι φαίνεται τι συμβαίνει όταν αποσυνδεθεί ένας εξυπηρετητής. Στη γραμμή 33 ο ZooKeeper ενημερώνει για την αλλαγή τον Master στέλνοντάς του τη νέα λίστα με τους δυο εξυπηρετητές και ο Master στη συνέχεια ενημερώνεται με αντίστοιχο τρόπο όπως είδαμε παραπάνω για τις αλλαγές στους κόμβους ενώ αφαιρεί από τον χάρτη τον εξυπηρετητή που έκλεισε. Το καταλαβαίνουμε ότι όντος αφαιρέθηκε από τον χάρτη γιατί ο Master ζητά ενημερώσεις για τους δυο εξυπηρετητές και δέχεται πίσω 6 μηνύματα, 3 για τον κάθε εξυπηρετητή. Ενώ στο τελευταίο κομμάτι στη γραμμή 42 βλέπουμε ότι έχουν κλείσει όλοι οι εξυπηρετητές και ο ZooKeeper γυρίζει πίσω κενή λίστα.



Σχήμα 6.10: One Server Disonnected



Σχήμα 6.11: Three Servers Disonnected

6.3.3 Σύνδεση με Πελάτη

Παρακάτω δίνεται απόσπασμα του αρχείου Log με το τι συμβαίνει όταν ένας πελάτης επικοινωνήσει με τον Master .

Listing 6.3: Master-Client Connection Log

```
------------------- SERVER IN THE SYSYTEM -------------------

1   2016-06-12 23:29:12,003INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@835] - [Master]: --SERVERS-- total 1
2   2016-06-12 23:29:12,003INFO -[main-EventThread]/[org.thesis.project.
    master.Master$18@841] -  Key: server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
     IP: localhost/127.0.0.1:63024 Clients: 0.0 Server Up Time0.0

------------------- CLIENT CONNECTION -------------------

3   2016-06-12 23:30:22,722INFO -[New I/O server boss #9]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0xddf57d92,
    /127.0.0.1:63044 => /127.0.0.1:8081] OPEN
4   2016-06-12 23:30:22,722INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0xddf57d92,
    /127.0.0.1:63044 => /127.0.0.1:8081] BOUND: /127.0.0.1:8081
5   2016-06-12 23:30:22,722INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0xddf57d92,
    /127.0.0.1:63044 => /127.0.0.1:8081] CONNECTED: /127.0.0.1:63044

------------------- SERVER ASSIGNMENT -------------------

6   2016-06-12 23:30:23,144INFO -[New I/O worker #1]/[o.thesis.project.
    master.MasterClientImpl@23] - [Master]: Method "hello" has been invoked
    ...
7   2016-06-12 23:30:23,151INFO -[New I/O worker #1]/[o.thesis.project.
    master.MasterClientImpl@24] - [Master]: clientID = 4ab9cbe0-baa6-416f-
    acee-849d32ccf27f
8   2016-06-12 23:30:23,151INFO -[New I/O worker #1]/[o.thesis.project.
    master.MasterClientImpl@25] - [Master]: address= org.apache.avro.ipc.
    NettyTransceiver@55a1c291
```

```
9   2016-06-12 23:30:23,152INFO -[New I/O worker #1]/[org.thesis.project.
    master.Master@736] - [Master]: Connection path: /connections/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7/4ab9cbe0-baa6-416f-acee-849d32ccf27f
10  2016-06-12 23:30:23,152INFO -[New I/O worker #1]/[org.thesis.project.
    master.Master@718] - [Master]: -CONNECTION-
11  2016-06-12 23:30:23,152INFO -[New I/O worker #1]/[org.thesis.project.
    master.Master@719] - [Master]: Client-4ab9cbe0-baa6-416f-acee-849
    d32ccf27f connected to Server-server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
     noClients = 0
12  2016-06-12 23:30:23,152DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/connections/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7/4ab9cbe0-baa6-416f-acee-849d32ccf27f serverPath:/
    connections/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7/4ab9cbe0-baa6-416
    f-acee-849d32ccf27f finished:false header:: 13,1  replyHeader:: 13,6657,0
      request:: '/connections/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7/4
    ab9cbe0-baa6-416f-acee-849d32ccf27f,#34616239636265302
    d626161362d343136662d616365652d383439643332636366323766,v{s{31,s{'world,'
    anyone}}},0  response:: '/connections/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7/4ab9cbe0-baa6-416f-acee-849d32ccf27f
13  2016-06-12 23:30:23,152INFO -[main-EventThread]/[org.thesis.project.
    master.Master$17@753] - Client assigned correctly: /connections/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7/4ab9cbe0-baa6-416f-acee-849d32ccf27f

------------------- CLIENT DISCONNECTION -------------------

14  2016-06-12 23:30:23,183INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0xddf57d92,
    /127.0.0.1:63044 :> /127.0.0.1:8081] DISCONNECTED
15  2016-06-12 23:30:23,183INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0xddf57d92,
    /127.0.0.1:63044 :> /127.0.0.1:8081] UNBOUND
16  2016-06-12 23:30:23,183INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0xddf57d92,
    /127.0.0.1:63044 :> /127.0.0.1:8081] CLOSED
17  2016-06-12 23:30:23,183INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@209] - Connection to /127.0.0.1:63044
    disconnected.

------------------- SERVER'S CLIENTS ZNODE UPDATED -------------------

18  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00016
19  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvClients/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
20  2016-06-12 23:30:23,315INFO -[main-EventThread]/[org.thesis.project.
    master.Master$14@573] - [Master]: srvClientsWatcher: Node Data Changed
    .../srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
21  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvClients/server-d2b8d3b9-246f-40
```

```
    cb-ba31-bd5824a483e7 serverPath:/srvClients/server-d2b8d3b9-246f-40cb-
    ba31-bd5824a483e7 finished:false header:: 14,4  replyHeader:: 14,6658,0
    request:: '/srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7,F
    response:: #31,s
    {6656,6658,1465763351972,1465763423299,1,0,0,96057447759740951,1,0,6656}
22  2016-06-12 23:30:23,315INFO -[main-EventThread]/[org.thesis.project.
    master.Master$16@627] - [Master]: Number of clients of server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7: 1
23  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvClients/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 serverPath:/srvClients/server-d2b8d3b9-246f-40cb-
    ba31-bd5824a483e7 finished:false header:: 15,3  replyHeader:: 15,6658,0
    request:: '/srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T
    response:: s
    {6656,6658,1465763351972,1465763423299,1,0,0,96057447759740951,1,0,6656}

------------------- SERVER'S UPAVGTIME ZNODE UPDATED -------------------

24  2016-06-12 23:30:31,106DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
25  2016-06-12 23:30:31,108INFO -[main-EventThread]/[org.thesis.project.
    master.Master$8@418] - [Master]: srvUpAvgTimeWatcher: Node Data Changed
    .../srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
26  2016-06-12 23:30:31,111DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 16,4  replyHeader::
    16,6659,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,F  response:: #31362e30,s
    {6654,6659,1465763351972,1465763431102,1,0,0,96057447759740951,4,0,6654}
27  2016-06-12 23:30:31,111INFO -[main-EventThread]/[org.thesis.project.
    master.Master$10@467] - [Master]: Server Up Average Time of server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7: 16.0
28  2016-06-12 23:30:31,113DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 17,3  replyHeader::
    17,6659,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,T  response:: s
    {6654,6659,1465763351972,1465763431102,1,0,0,96057447759740951,4,0,6654}

------------------- SERVER'S UPAVGTIME ZNODE UPDATED -------------------

29  2016-06-12 23:30:51,107DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
30  2016-06-12 23:30:51,108INFO -[main-EventThread]/[org.thesis.project.
    master.Master$8@418] - [Master]: srvUpAvgTimeWatcher: Node Data Changed
```

```
    .../srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
31  2016-06-12 23:30:51,115DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00016 after 9ms
32  2016-06-12 23:30:51,118DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 18,4  replyHeader::
    18,6660,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,F  response:: #31393232352e30,s
    {6654,6660,1465763351972,1465763451102,2,0,0,96057447759740951,7,0,6654}
33  2016-06-12 23:30:51,118INFO -[main-EventThread]/[org.thesis.project.
    master.Master$10@467] - [Master]: Server Up Average Time of server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7: 19225.0
34  2016-06-12 23:30:51,119DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 19,3  replyHeader::
    19,6660,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,T  response:: s
    {6654,6660,1465763351972,1465763451102,2,0,0,96057447759740951,7,0,6654}

------------------- SERVER'S UPAVGTIME ZNODE UPDATED -------------------

35  2016-06-12 23:31:01,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
36  2016-06-12 23:31:01,106INFO -[main-EventThread]/[org.thesis.project.
    master.Master$8@418] - [Master]: srvUpAvgTimeWatcher: Node Data Changed
    .../srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
37  2016-06-12 23:31:01,107DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00016 after 2ms
38  2016-06-12 23:31:01,109DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 20,4  replyHeader::
    20,6661,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,F  response:: #32363830372e35,s
    {6654,6661,1465763351972,1465763461101,3,0,0,96057447759740951,7,0,6654}
39  2016-06-12 23:31:01,109INFO -[main-EventThread]/[org.thesis.project.
    master.Master$10@467] - [Master]: Server Up Average Time of server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7: 26807.5
40  2016-06-12 23:31:01,110DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 21,3  replyHeader::
    21,6661,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,T  response:: s
```

```
    {6654,6661,1465763351972,1465763461101,3,0,0,96057447759740951,7,0,6654}


---------- SERVER'S CLIENT ZNODE UPDATED / CLIENT FINISHED ----------

41  2016-06-12 23:31:11,175DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvClients/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
42  2016-06-12 23:31:11,175INFO -[main-EventThread]/[org.thesis.project.
    master.Master$14@573] - [Master]: srvClientsWatcher: Node Data Changed
    .../srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
43  2016-06-12 23:31:11,175DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvClients/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 serverPath:/srvClients/server-d2b8d3b9-246f-40cb-
    ba31-bd5824a483e7 finished:false header:: 22,4  replyHeader:: 22,6662,0
    request:: '/srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7,F
    response:: #30,s
    {6656,6662,1465763351972,1465763471159,2,0,0,96057447759740951,1,0,6656}
44  2016-06-12 23:31:11,175INFO -[main-EventThread]/[org.thesis.project.
    master.Master$16@627] - [Master]: Number of clients of server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7: 0
45  2016-06-12 23:31:11,188DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvTotalRuntime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
46  2016-06-12 23:31:11,189DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvClients/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 serverPath:/srvClients/server-d2b8d3b9-246f-40cb-
    ba31-bd5824a483e7 finished:false header:: 23,3  replyHeader:: 23,6663,0
    request:: '/srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T
    response:: s
    {6656,6662,1465763351972,1465763471159,2,0,0,96057447759740951,1,0,6656}

------------- SERVER'S TOTAL RUNTIME ZNODE UPDATED -------------

47  2016-06-12 23:31:11,190INFO -[main-EventThread]/[org.thesis.project.
    master.Master$11@495] - [Master]: SrvTotalRuntimeWatcher: Node Data
    Changed.../srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
48  2016-06-12 23:31:11,192DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvTotalRuntime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7 serverPath:/srvTotalRuntime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7 finished:false header:: 24,4  replyHeader::
    24,6663,0  request:: '/srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,F  response:: #3437383630,s
    {6655,6663,1465763351972,1465763471175,1,0,0,96057447759740951,5,0,6655}
49  2016-06-12 23:31:11,194INFO -[main-EventThread]/[org.thesis.project.
    master.Master$13@544] - [Master]: Total Runtime of server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7: 47860
50  2016-06-12 23:31:11,196DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
```

```
    x15543bd63c00016, packet:: clientPath:/srvTotalRuntime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7 serverPath:/srvTotalRuntime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7 finished:false header:: 25,3  replyHeader::
    25,6663,0  request:: '/srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,T  response:: s
    {6655,6663,1465763351972,1465763471175,1,0,0,96057447759740951,5,0,6655}

---------------- MASTER NOTICE / CLIENT FINISHED ----------------

51  2016-06-12 23:31:11,214INFO -[New I/O server boss #9]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x7145c1f9,
    /127.0.0.1:63085 => /127.0.0.1:8081] OPEN
52  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x7145c1f9,
    /127.0.0.1:63085 => /127.0.0.1:8081] BOUND: /127.0.0.1:8081
53  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x7145c1f9,
    /127.0.0.1:63085 => /127.0.0.1:8081] CONNECTED: /127.0.0.1:63085
54  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.thesis.project.
    master.MasterClientImpl@44] - [Master]: Method "bye" has been invoked...
55  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.thesis.project.
    master.MasterClientImpl@45] - [Master]: clientId = 4ab9cbe0-baa6-416f-
    acee-849d32ccf27f
56  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x7145c1f9,
    /127.0.0.1:63085 :> /127.0.0.1:8081] DISCONNECTED
57  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x7145c1f9,
    /127.0.0.1:63085 :> /127.0.0.1:8081] UNBOUND
58  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x7145c1f9,
    /127.0.0.1:63085 :> /127.0.0.1:8081] CLOSED
59  2016-06-12 23:31:11,214INFO -[New I/O worker #2]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@209] - Connection to /127.0.0.1:63085
    disconnected.

------------------- SERVER'S UPAVGTIME ZNODE UPDATED -------------------

60  2016-06-12 23:31:21,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00016
61  2016-06-12 23:31:21,112INFO -[main-EventThread]/[org.thesis.project.
    master.Master$8@418] - [Master]: srvUpAvgTimeWatcher: Node Data Changed
    .../srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
62  2016-06-12 23:31:21,114DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00016 after 4ms
63  2016-06-12 23:31:21,114DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 26,4  replyHeader::
    26,6664,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
```

```
    bd5824a483e7,F  response:: #34303234342e35,s
    {6654,6664,1465763351972,1465763481105,4,0,0,96057447759740951,7,0,6654}
64  2016-06-12 23:31:21,114INFO -[main-EventThread]/[org.thesis.project.
    master.Master$10@467] - [Master]: Server Up Average Time of server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7: 40244.5
65  2016-06-12 23:31:21,114DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00016, packet:: clientPath:/srvUpAvgTime/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 serverPath:/srvUpAvgTime/server-d2b8d3b9-246f-40
    cb-ba31-bd5824a483e7 finished:false header:: 27,3  replyHeader::
    27,6664,0  request:: '/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7,T  response:: s
    {6654,6664,1465763351972,1465763481105,4,0,0,96057447759740951,7,0,6654}
```

Στη γραμμή 2 βλέπουμε την ταυτότητα του εξυπηρετητή που βρίσκεται στο σύστημα. Στις γραμμές 3-5 βλέπουμε την σύνδεση του πελάτη με τον Master . Στις γραμμές 7 και 8 βλέπουμε τα στοιχεία που έστειλε ο πελάτης στον Master . Ενώ στις γραμμές 9-12 περιγράφεται η διαδικασία ανάθεσης του εξυπηρετητή στον πελάτη. Ο Master στέλνει στον πελάτη τα στοιχεία του αντίστοιχου εξυπηρετητή ενώ δημιουργεί και έναν κόμβο παιδί κάτω από τον κόμβο /connectios όπου αντιστοιχίζει τον εξυπηρετητή με τον πελάτη. Έπειτα, ο πελάτης αποσυνδέεται από τον Master [γραμμές 14-17] και όπως θα δούμε παρακάτω συνδέεται με τον εξυπηρετητή. Στη συνέχεια φαίνονται οι ενέργειες με τα μηνύματα από τον Zookeeper , που ενημερώνει τον Master για τις αλλαγές που κάνει ο εξυπηρετητής στους αντίστοιχους κόμβους. Παρατηρούμε :

- Δυο αλλαγές στον κόμβο /srvClients , μια όταν συνδέεται ο πελάτης [γραμμές 18-23] όπου φαίνεται πως ο εξυπηρετητής έχει έναν πελάτη [γραμμή 22] και μια όταν αποσυνδέεται ο πελάτης [γραμμές 41-46] όπου φαίνεται πως ο εξυπηρετητής δεν έχει πλέον κανέναν πελάτη [γραμμή 44].

- Τέσσερεις αλλαγές στον κόμβο /srvUpAvgTime με τον μέσο χρόνο λειτουργίας του εξυπηρετητή [γραμμές 24-28, 29-34, 35-40, 60-65]. Γίνονται τέσσερα αιτήματα από τον πελάτη προς τον εξυπηρετητή, οπότε μετρώντας τον χρόνο εκτέλεσης κάθε αιτήματος και αθροίζοντάς το για να υπολογίσουμε τον μέσο χρόνο, όντος περιμέναμε τέσσερεις αλλαγές. Και επειδή ο scheduler ο οποίος κρατά και μετρά αυτούς τους χρόνους δρα ανεξάρτητα από το πότε συνδέεται ο κάθε πελάτης παρατηρούμε πως η τελευταία ανανέωση γίνεται αφού έχει τερματίσει η σύνδεση με τον συγκεκριμένο πελάτη.

- Μια αλλαγή στον κόμβο /srvTotalRuntime όπου κρατά τον ολικό χρόνο λειτουργίας του εξυπηρετητή [γραμμές 47-50], ο οποίος ενημερώνεται κάθε φορά που αποσυνδέεται ένας πελάτης.

Τέλος αφού ο πελάτης έχει ολοκληρώσει τις διεργασίες του με τον εξυπηρετητή ενημερώνει τον Master για τη λήξη της σύνδεσης [γραμμές 51-59].

6.3.4 Επιλογή εξυπηρετητή

Παρακάτω δίνεται απόσπασμα των αρχείων Log για την επιλογή εξυπηρετητή από τον Master .

Τυχαία Επιλογή

Listing 6.4: Random Selection Log

```
------------------- TOTAL SERVERS -------------------

2016-06-13 22:08:52,496INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@843] - [Master]: --SERVERS-- total 3
2016-06-13 22:08:52,511INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-5a5f8616-eb68-4a78-9edb-78a2bc1f118a IP:
    localhost/127.0.0.1:64196 Clients: 0.0 Server Up Time0.0
2016-06-13 22:08:52,511INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-95f67489-55a0-4b0a-9c4d-96fafaf5a2fd IP:
    localhost/127.0.0.1:64218 Clients: 0.0 Server Up Time0.0
2016-06-13 22:08:52,511INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-b53ab8b0-446a-442c-b647-fe6e198e89d7 IP:
    localhost/127.0.0.1:64240 Clients: 0.0 Server Up Time0.0

------------------- CONNECT 1ST CLIENT -------------------

2016-06-13 22:09:01,484INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:09:01,484INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@727] - [Master]: Client-ce7f9f77-212b-4cda-b717-384b75e95646
    connected to Server-server-b53ab8b0-446a-442c-b647-fe6e198e89d7 noClients
     = 0

------------------- CONNECT 2ND CLIENT -------------------

2016-06-13 22:09:07,431INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:09:07,431INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@727] - [Master]: Client-55a13be1-6b65-4da5-a942-009ecec43b12
    connected to Server-server-b53ab8b0-446a-442c-b647-fe6e198e89d7 noClients
     = 1

------------------- CONNECT 3RD CLIENT -------------------

2016-06-13 22:09:10,337INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:09:10,337INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@727] - [Master]: Client-ac820179-0f04-495b-9da3-f482c065a5bf
    connected to Server-server-b53ab8b0-446a-442c-b647-fe6e198e89d7 noClients
     = 2

------------------- CONNECT 4TH CLIENT -------------------

2016-06-13 22:09:14,590INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:09:14,591INFO -[New I/O worker #4]/[org.thesis.project.master.
```

```
    Master@727] - [Master]: Client-e1766195-4346-4a2e-bf13-76ba17ccfa28
    connected to Server-server-95f67489-55a0-4b0a-9c4d-96fafaf5a2fd noClients
     = 0

------------------- CONNECT 5TH CLIENT -------------------

2016-06-13 22:09:18,839INFO -[New I/O worker #5]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:09:18,839INFO -[New I/O worker #5]/[org.thesis.project.master.
    Master@727] - [Master]: Client-1ca2d96f-e0eb-4cac-a1ee-1ca1e2617ea1
    connected to Server-server-95f67489-55a0-4b0a-9c4d-96fafaf5a2fd noClients
     = 1

------------------- CONNECT 6TH CLIENT -------------------

2016-06-13 22:09:25,442INFO -[New I/O worker #7]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:09:25,442INFO -[New I/O worker #7]/[org.thesis.project.master.
    Master@727] - [Master]: Client-f069bf19-c5fe-4443-b35a-3682fcffe103
    connected to Server-server-b53ab8b0-446a-442c-b647-fe6e198e89d7 noClients
     = 2
```

Κυκλική Επιλογή

Listing 6.5: Cycle Selection Log

```
------------------- TOTAL SERVERS -------------------

2016-06-13 22:16:22,324INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@843] - [Master]: --SERVERS-- total 3
2016-06-13 22:16:22,324INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-109fe01f-fbc2-4a8c-9f7a-408799601b9d IP:
    localhost/127.0.0.1:64732 Clients: 0.0 Server Up Time0.0
2016-06-13 22:16:22,324INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-15644feb-95b7-44a5-8b7c-17e59c42d6d4 IP:
    localhost/127.0.0.1:64710 Clients: 0.0 Server Up Time0.0
2016-06-13 22:16:22,324INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-ed6336c1-3332-4571-bb63-9da0d4d8e9dc IP:
    localhost/127.0.0.1:64754 Clients: 0.0 Server Up Time0.0

------------------- CONNECT 1ST CLIENT -------------------

2016-06-13 22:16:29,433INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:16:29,433INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@727] - [Master]: Client-1caea975-878a-467a-95a9-703e14ff7d31
    connected to Server-server-109fe01f-fbc2-4a8c-9f7a-408799601b9d noClients
     = 0

------------------- CONNECT 2ND CLIENT -------------------

2016-06-13 22:16:34,981INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:16:34,981INFO -[New I/O worker #2]/[org.thesis.project.master.
```

```
    Master@727] - [Master]: Client-f61817e8-f695-4356-9e2d-4d0fd15177af
    connected to Server-server-15644feb-95b7-44a5-8b7c-17e59c42d6d4 noClients
     = 0

------------------- CONNECT 3RD CLIENT -------------------

2016-06-13 22:16:37,556INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:16:37,556INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@727] - [Master]: Client-339430bb-0854-424c-b767-ff5c9487305c
    connected to Server-server-ed6336c1-3332-4571-bb63-9da0d4d8e9dc noClients
     = 0

------------------- CONNECT 4TH CLIENT -------------------

2016-06-13 22:16:43,992INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:16:43,992INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@727] - [Master]: Client-252f581a-01d5-499c-83bf-c354b6dcac47
    connected to Server-server-109fe01f-fbc2-4a8c-9f7a-408799601b9d noClients
     = 1

------------------- CONNECT 5TH CLIENT -------------------

2016-06-13 22:16:49,974INFO -[New I/O worker #5]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:16:49,975INFO -[New I/O worker #5]/[org.thesis.project.master.
    Master@727] - [Master]: Client-5400176c-d14a-4479-b812-b78497631d77
    connected to Server-server-15644feb-95b7-44a5-8b7c-17e59c42d6d4 noClients
     = 1

------------------- CONNECT 6TH CLIENT -------------------

2016-06-13 22:16:57,266INFO -[New I/O worker #6]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:16:57,266INFO -[New I/O worker #6]/[org.thesis.project.master.
    Master@727] - [Master]: Client-43eab668-d3ce-4eda-ab64-9b8e99dafcbd
    connected to Server-server-ed6336c1-3332-4571-bb63-9da0d4d8e9dc noClients
     = 1
```

Επιλογή Εξυπηρετητή με τους λιγότερους πελάτες

Listing 6.6: Least Clients Log

```
------------------- TOTAL SERVERS -------------------

2016-06-13 22:20:53,299INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@843] - [Master]: --SERVERS-- total 3
2016-06-13 22:20:53,299INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-6d757878-e486-4e93-ae6c-f70e9708d9f5 IP:
    localhost/127.0.0.1:65184 Clients: 0.0 Server Up Time0.0
2016-06-13 22:20:53,299INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-bf6fe22d-db8f-4c8a-9fe1-b804583a5263 IP:
    localhost/127.0.0.1:65162 Clients: 0.0 Server Up Time0.0
```

```
2016-06-13 22:20:53,314INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-ce76777f-40de-4c78-9c54-63926451f2d4 IP:
    localhost/127.0.0.1:65140 Clients: 0.0 Server Up Time0.0

------------------- CONNECT 1ST CLIENT -------------------

2016-06-13 22:20:59,816INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:20:59,816INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@727] - [Master]: Client-ab2b7b8c-c282-4f71-aa9a-80f02c9c403e
    connected to Server-server-6d757878-e486-4e93-ae6c-f70e9708d9f5 noClients
     = 0

------------------- CONNECT 2ND CLIENT -------------------

2016-06-13 22:20:59,973INFO -[main-EventThread]/[org.thesis.project.master.
    Master$16@631] - [Master]: Number of clients of server-6d757878-e486-4e93
    -ae6c-f70e9708d9f5: 1

2016-06-13 22:21:05,715INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:21:05,716INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@727] - [Master]: Client-3f2cc6d9-b5d5-48e0-b159-b7e9a7d98f43
    connected to Server-server-bf6fe22d-db8f-4c8a-9fe1-b804583a5263 noClients
     = 0


------------------- CONNECT 3RD CLIENT -------------------

2016-06-13 22:21:05,869INFO -[main-EventThread]/[org.thesis.project.master.
    Master$16@631] - [Master]: Number of clients of server-bf6fe22d-db8f-4c8a
    -9fe1-b804583a5263: 1

2016-06-13 22:21:10,026INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:21:10,026INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@727] - [Master]: Client-a82f4fe7-9624-4ef4-9d06-e66d75e69436
    connected to Server-server-ce76777f-40de-4c78-9c54-63926451f2d4 noClients
     = 0

------------------- CONNECT 4TH CLIENT -------------------

2016-06-13 22:21:10,184INFO -[main-EventThread]/[org.thesis.project.master.
    Master$16@631] - [Master]: Number of clients of server-ce76777f-40de-4c78
    -9c54-63926451f2d4: 1

016-06-13 22:21:14,902INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:21:14,903INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@727] - [Master]: Client-2aab7fe8-dffb-4fa2-b3fc-547b48b2aae0
    connected to Server-server-6d757878-e486-4e93-ae6c-f70e9708d9f5 noClients
     = 1

------------------- CONNECT 5TH CLIENT -------------------
```

```
2016-06-13 22:21:14,987INFO -[main-EventThread]/[org.thesis.project.master.
    Master$16@631] - [Master]: Number of clients of server-6d757878-e486-4e93
    -ae6c-f70e9708d9f5: 2
2016-06-13 22:21:15,875INFO -[main-EventThread]/[org.thesis.project.master.
    Master$16@631] - [Master]: Number of clients of server-bf6fe22d-db8f-4c8a
    -9fe1-b804583a5263: 0

2016-06-13 22:21:19,962INFO -[New I/O worker #6]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:21:19,963INFO -[New I/O worker #6]/[org.thesis.project.master.
    Master@727] - [Master]: Client-9b195070-66b1-4746-ac48-def7a644cce1
    connected to Server-server-bf6fe22d-db8f-4c8a-9fe1-b804583a5263 noClients
     = 0

------------------- CONNECT 6TH CLIENT -------------------

2016-06-13 22:21:20,055INFO -[main-EventThread]/[org.thesis.project.master.
    Master$16@631] - [Master]: Number of clients of server-bf6fe22d-db8f-4c8a
    -9fe1-b804583a5263: 1

2016-06-13 22:21:27,063INFO -[New I/O worker #7]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:21:27,063INFO -[New I/O worker #7]/[org.thesis.project.master.
    Master@727] - [Master]: Client-98ef785f-35fa-4217-8e78-654a7221bf1e
    connected to Server-server-bf6fe22d-db8f-4c8a-9fe1-b804583a5263 noClients
     = 1
```

Επιλογή Εξυπηρετητή με τον λιγότερο μέσο χρόνο λειτουργίας

Listing 6.7: Least Average Time Log

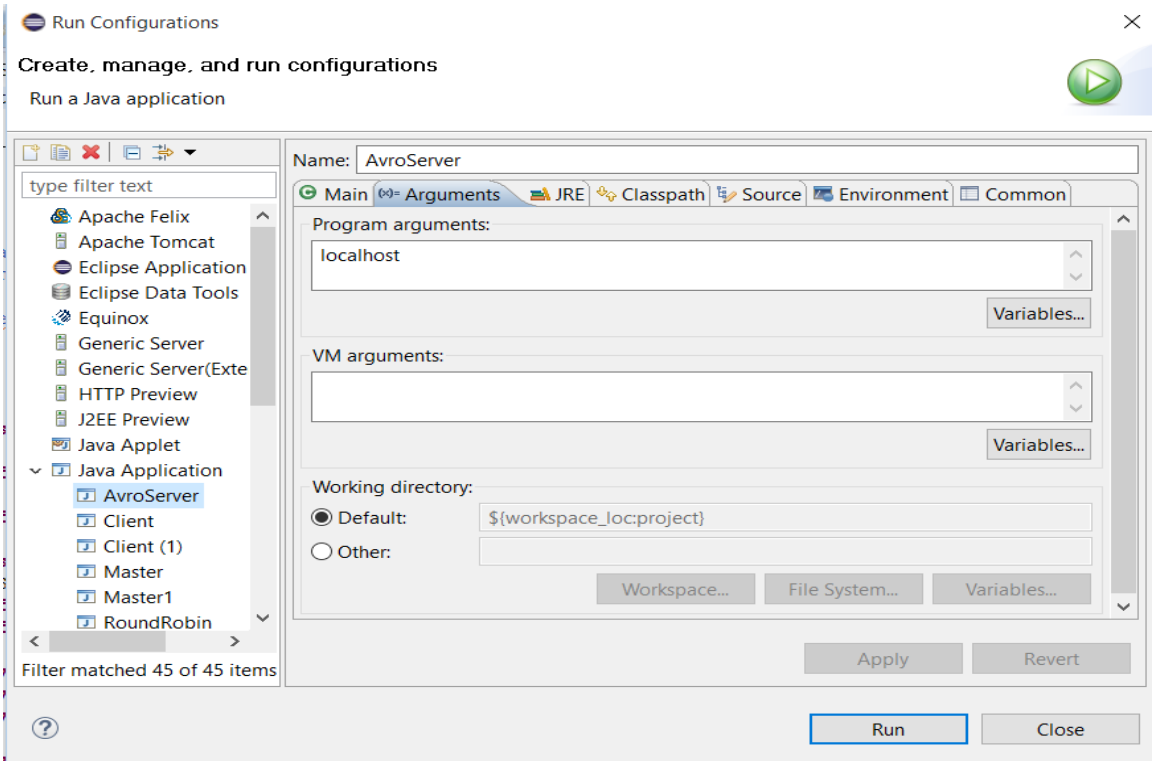
```
------------------- TOTAL SERVERS -------------------

2016-06-13 22:30:25,427INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@843] - [Master]: --SERVERS-- total 3
2016-06-13 22:30:25,427INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-91a1a6dd-83a8-4278-b842-7e6007ac7010 IP:
    localhost/127.0.0.1:49741 Clients: 0.0 Server Up Time0.0
2016-06-13 22:30:25,427INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-a6aaf050-eda5-4ac0-a45d-4543d54c541d IP:
    localhost/127.0.0.1:49763 Clients: 0.0 Server Up Time0.0
2016-06-13 22:30:25,427INFO -[main-EventThread]/[org.thesis.project.master.
    Master$18@849] -  Key: server-d1101515-2471-4bee-ae35-7e625982201b IP:
    localhost/127.0.0.1:49785 Clients: 0.0 Server Up Time0.0

------------------- CONNECT 1ST CLIENT -------------------

2016-06-13 22:30:37,287INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:30:37,287INFO -[New I/O worker #1]/[org.thesis.project.master.
    Master@727] - [Master]: Client-4ec05be0-01e9-4357-b1b2-a030cea32c65
    connected to Server-server-91a1a6dd-83a8-4278-b842-7e6007ac7010 noClients
```

```
     = 0

------------------- CONNECT 2ND CLIENT -------------------

2016-06-13 22:30:39,246INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-91a1a6dd-83a8
    -4278-b842-7e6007ac7010: 16.0

2016-06-13 22:30:43,962INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:30:43,962INFO -[New I/O worker #2]/[org.thesis.project.master.
    Master@727] - [Master]: Client-b8f1c54f-d1c7-43b3-acac-5b3a7373e9c4
    connected to Server-server-a6aaf050-eda5-4ac0-a45d-4543d54c541d noClients
     = 0

------------------- CONNECT 3RD CLIENT -------------------

2016-06-13 22:30:46,829INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-a6aaf050-eda5
    -4ac0-a45d-4543d54c541d: 19.0
2016-06-13 22:30:49,241INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-91a1a6dd-83a8
    -4278-b842-7e6007ac7010: 10897.0

2016-06-13 22:30:51,764INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:30:51,765INFO -[New I/O worker #3]/[org.thesis.project.master.
    Master@727] - [Master]: Client-842023fd-ba23-4663-b54d-7041d652e63c
    connected to Server-server-d1101515-2471-4bee-ae35-7e625982201b noClients
     = 0

------------------- CONNECT 4TH CLIENT -------------------

2016-06-13 22:30:54,543INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-d1101515
    -2471-4bee-ae35-7e625982201b: 20.0
2016-06-13 22:30:59,239INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-91a1a6dd-83a8
    -4278-b842-7e6007ac7010: 15643.0

2016-06-13 22:30:59,330INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:30:59,330INFO -[New I/O worker #4]/[org.thesis.project.master.
    Master@727] - [Master]: Client-3090f4ad-9a01-41ec-a4d9-47b054482d13
    connected to Server-server-a6aaf050-eda5-4ac0-a45d-4543d54c541d noClients
     = 1

------------------- CONNECT 5TH CLIENT -------------------

2016-06-13 22:31:04,536INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-d1101515
    -2471-4bee-ae35-7e625982201b: 5899.0
2016-06-13 22:31:06,828INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-a6aaf050-eda5
```

```
    -4ac0-a45d-4543d54c541d: 6033.5

2016-06-13 22:31:07,022INFO -[New I/O worker #5]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:31:07,022INFO -[New I/O worker #5]/[org.thesis.project.master.
    Master@727] - [Master]: Client-d60af34c-2f9b-47bb-a069-35e17c5d405e
    connected to Server-server-d1101515-2471-4bee-ae35-7e625982201b noClients
     = 1

------------------- CONNECT 6TH CLIENT -------------------

2016-06-13 22:31:09,236INFO -[main-EventThread]/[org.thesis.project.master.
    Master$10@471] - [Master]: Server Up Average Time of server-91a1a6dd-83a8
    -4278-b842-7e6007ac7010: 31056.0

2016-06-13 22:31:14,470INFO -[New I/O worker #6]/[org.thesis.project.master.
    Master@726] - [Master]: -CONNECTION-
2016-06-13 22:31:14,470INFO -[New I/O worker #6]/[org.thesis.project.master.
    Master@727] - [Master]: Client-0345a5c2-d5b3-4a54-8c92-d43c0498d8a8
    connected to Server-server-d1101515-2471-4bee-ae35-7e625982201b noClients
     = 2
```

6.4 Λειτουργία Εξυπηρετητή

Ο εξυπηρετητής τρέχει όπως οποιαδήποτε άλλη εφαρμογή Java με όρισμα την διεύθυνση που βρίσκεται ο Zookeeper .



Σχήμα 6.12: Server Configuration

6.4.1 Σύνδεση με Zookeeper

Παρακάτω δίνεται απόσπασμα από το αρχείο Log του εξυπηρετητή με τις ενέργειες του κατά τη σύνδεσή του με τον Zookeeper .

Listing 6.8: Server Connection Log

```
------------------- CONNECT TO ZOOKEEPER -------------------

1   2016-06-12 14:01:08,563INFO -[main]/[org.apache.zookeeper.ZooKeeper@438]
     - Initiating client connection, connectString=localhost sessionTimeout
    =15000 watcher=org.thesis.project.server.AvroServer@6b71769e
2   2016-06-12 14:01:08,578DEBUG-[main]/[org.apache.zookeeper.ClientCnxn@102
    ] - zookeeper.disableAutoWatchReset is false
3   2016-06-12 14:01:08,703INFO -[main]/[org.thesis.project.server.
    AvroServer@76] - Server is starting...
4   2016-06-12 14:01:08,716INFO -[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@975] - Opening socket connection to
    server 127.0.0.1/127.0.0.1:2181. Will not attempt to authenticate using
    SASL (unknown error)
5   2016-06-12 14:01:08,716INFO -[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@852] - Socket connection established to
    127.0.0.1/127.0.0.1:2181, initiating session
```

```
6   2016-06-12 14:01:08,716DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@892] - Session establishment request
    sent on 127.0.0.1/127.0.0.1:2181
7   2016-06-12 14:01:08,748INFO -[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@1235] - Session establishment complete
    on server 127.0.0.1/127.0.0.1:2181, sessionid = 0x15543bd63c0000a,
    negotiated timeout = 15000
8   2016-06-12 14:01:08,748INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:None path:
    null, localhost

------------------- AVRO PART --------------------

9   2016-06-12 14:01:09,201INFO -[main]/[org.thesis.project.server.
    AvroServer@79] - Server listens to: 60491

------------------- INITIALIZE ZOOKEEPER ZNODES -------------------

10  2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000a, packet:: clientPath:/connections/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 serverPath:/connections/server-e46e8c60-902d-4c04-8
    b59-c7ad25521961 finished:false header:: 1,1  replyHeader:: 1,6540,0
    request:: '/connections/server-e46e8c60-902d-4c04-8b59-c7ad25521961,#6
    c6f63616c686f73742f3132372e302e302e313a3630343931,v{s{31,s{'world,'anyone
    }}},0  response:: '/connections/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961
11  2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000a, packet:: clientPath:/servers/server-e46e8c60-902d-4c04
    -8b59-c7ad25521961 serverPath:/servers/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961 finished:false header:: 2,1  replyHeader:: 2,6541,0  request
    :: '/servers/server-e46e8c60-902d-4c04-8b59-c7ad25521961,#6
    c6f63616c686f73742f3132372e302e302e313a3630343931,v{s{31,s{'world,'anyone
    }}},1  response:: '/servers/server-e46e8c60-902d-4c04-8b59-c7ad25521961
12  2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000a, packet:: clientPath:/srvUpAvgTime/server-e46e8c60-902d
    -4c04-8b59-c7ad25521961 serverPath:/srvUpAvgTime/server-e46e8c60-902d-4
    c04-8b59-c7ad25521961 finished:false header:: 3,1  replyHeader:: 3,6542,0
      request:: '/srvUpAvgTime/server-e46e8c60-902d-4c04-8b59-c7ad25521961
    ,#30,v{s{31,s{'world,'anyone}}},1  response:: '/srvUpAvgTime/server-
    e46e8c60-902d-4c04-8b59-c7ad25521961
13  2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000a, packet:: clientPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 serverPath:/srvTotalRuntime/server-e46e8c60
    -902d-4c04-8b59-c7ad25521961 finished:false header:: 4,1  replyHeader::
    4,6543,0  request:: '/srvTotalRuntime/server-e46e8c60-902d-4c04-8b59-
    c7ad25521961,#30,v{s{31,s{'world,'anyone}}},1  response:: '/
    srvTotalRuntime/server-e46e8c60-902d-4c04-8b59-c7ad25521961
14  2016-06-12 14:01:09,216DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000a, packet:: clientPath:/srvClients/server-e46e8c60-902d-4
```

```
    c04-8b59-c7ad25521961 serverPath:/srvClients/server-e46e8c60-902d-4c04-8
    b59-c7ad25521961 finished:false header:: 5,1  replyHeader:: 5,6544,0
    request:: '/srvClients/server-e46e8c60-902d-4c04-8b59-c7ad25521961,#30,v{
    s{31,s{'world,'anyone}}},1  response:: '/srvClients/server-e46e8c60-902d
    -4c04-8b59-c7ad25521961
15  2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer$1@208] - connections node created...
16  2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer$2@242] - Registered successfully...e46e8c60-902d-4c04-8
    b59-c7ad25521961
17  2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer$3@271] - Server up time node created successfully...
    e46e8c60-902d-4c04-8b59-c7ad25521961
18  2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer$4@298] - Server runtime node created successfully...
    e46e8c60-902d-4c04-8b59-c7ad25521961
19  2016-06-12 14:01:09,232INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer$5@325] - Server clients node created successfully...
    e46e8c60-902d-4c04-8b59-c7ad25521961
```

Στις γραμμές 1-8 βλέπουμε την εδραίωση της συνεδρίας μεταξύ του εξυπηρετητή και του Zookeeper . Στη γραμμή 9 βλέπουμε το Avro κομμάτι του εξυπηρετητή όπου αναλαμβάνει μια θύρα να ακούει για την επικοινωνία του με τους πελάτες. Ενώ στις γραμμές 10-19 βλέπουμε την αρχικοποίηση των επιπλέον κόμβων στον Zookeeper (/srvClients, /srvUpAvgTime, /srvTotalRuntime), όπου δημιουργείται ένα επιπλέον παιδί κάτω από τον αρχικό κόμβο που είχε δημιουργήσει ο Master με το όνομα του εξυπηρετητή και ως δεδομένα τις αντίστοιχες τιμές.

6.4.2 Σύνδεση με πελάτη

Παρακάτω δίνεται απόσπασμα από το αρχείο Log του εξυπηρετητή με τις ενέργειες του κατά τη σύνδεσή του με έναν πελάτη.

Listing 6.9: Server-Client Connection Log

```
------------------- CLIENT CONNECTION -------------------

1   2016-06-12 23:30:23,246INFO -[New I/O server boss #9]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x90f593ef,
    /127.0.0.1:63063 => /127.0.0.1:63024] OPEN
2   2016-06-12 23:30:23,252INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x90f593ef,
    /127.0.0.1:63063 => /127.0.0.1:63024] BOUND: /127.0.0.1:63024
3   2016-06-12 23:30:23,252INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x90f593ef,
    /127.0.0.1:63063 => /127.0.0.1:63024] CONNECTED: /127.0.0.1:63063

------------------- REQUEST: INIT -------------------

4   2016-06-12 23:30:23,283INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@39] - Method "send" has been invoked...
```

```
5   2016-06-12 23:30:23,283INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@40] - clientID: 4ab9cbe0-baa6-416f-acee-849
    d32ccf27f
6   2016-06-12 23:30:23,283INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@41] - status: INIT
7   2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@42] - timestamp: 1465763423230
8   2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@43] - procedure data: Hello world!
9   2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@399] - [Server]: Adding client 4ab9cbe0-baa6-416f-acee
    -849d32ccf27f to map...
10  2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@404] - ==== CLIENTS ====
11  2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@408] -  Key: 4ab9cbe0-baa6-416f-acee-849d32ccf27f IP: 4
    ab9cbe0-baa6-416f-acee-849d32ccf27f Clients: INIT
12  2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@412] - =============
13  2016-06-12 23:30:23,299INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@357] - [Server]: --Clients-- {} 1
14  2016-06-12 23:30:23,299DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 6,3  replyHeader:: 6,6657,0  request:: '/srvClients/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6656,6656,1465763351972,1465763351972,0,0,0,96057447759740951,1,0,6656}
15  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
16  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvClients/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
17  2016-06-12 23:30:23,315INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvClients/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
18  2016-06-12 23:30:23,315DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 7,5  replyHeader:: 7,6658,0  request:: '/srvClients/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#31,0  response:: s
    {6656,6658,1465763351972,1465763423299,1,0,0,96057447759740951,1,0,6656}

------------------- MONITOR ELEMENTS / SCHEDULER [1] -------------------

19  2016-06-12 23:30:23,315INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763423315 with status:
    INIT and duration: 16

------------------- REQUEST: SELECT -------------------
```

```
20  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@39] - Method "send" has been invoked...
21  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@40] - clientID: 4ab9cbe0-baa6-416f-acee-849
    d32ccf27f
22  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@41] - status: SELECT
23  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@42] - timestamp: 1465763423330
24  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@43] - procedure data: Hello world!
25  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@157] - Method "doSelect" starts sleeping for
    19201...
26  2016-06-12 23:30:28,300DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms

------------------- MONITOR ELEMENTS / SCHEDULER [1] -------------------

27  2016-06-12 23:30:31,097INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@26] - [MonitoringRunnable]: Start monitoring...
28  2016-06-12 23:30:31,097INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@37] - [MonitoringRunnable]: Element-1465763423315 with
     status: INIT and duration: 16
29  2016-06-12 23:30:31,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@50] - [MonitoringRunnable]: SrvAvg = 0.0
30  2016-06-12 23:30:31,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@51] - [MonitoringRunnable]: Time Added = 16.0
31  2016-06-12 23:30:31,098INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 16.0
32  2016-06-12 23:30:31,100DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 8,3  replyHeader:: 8,6658,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6654,6654,1465763351972,1465763351972,0,0,0,96057447759740951,1,0,6654}
33  2016-06-12 23:30:31,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
34  2016-06-12 23:30:31,106DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
35  2016-06-12 23:30:31,107INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
36  2016-06-12 23:30:31,109DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 9,5  replyHeader:: 9,6659,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#31362e30,0  response:: s
```

```
    {6654,6659,1465763351972,1465763431102,1,0,0,96057447759740951,4,0,6654}
37  2016-06-12 23:30:31,109INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@350] - [Server]: Path /srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 Updated!!!

------------------- SCHEDULERS -------------------

38  2016-06-12 23:30:31,365INFO -[pool-2-thread-1]/[o.thesis.project.server.
    ClientsControl$1@40] - [ClientsControl]: Expired Clients remover is
    starting...
39  2016-06-12 23:30:36,103DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms
40  2016-06-12 23:30:41,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@26] - [MonitoringRunnable]: Start monitoring...
41  2016-06-12 23:30:41,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms
42  2016-06-12 23:30:41,364INFO -[pool-2-thread-1]/[o.thesis.project.server.
    ClientsControl$1@40] - [ClientsControl]: Expired Clients remover is
    starting...

------------------- SELECT FINISHED -------------------

43  2016-06-12 23:30:42,538INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@159] - Method "doSelect" woke up...
44  2016-06-12 23:30:42,539INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@463] - [Server]: Client-4ab9cbe0-baa6-416f-acee-849
    d32ccf27f last active on :Sun Jun 12 23:30:42 EEST 2016
45  2016-06-12 23:30:42,539INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763442539 with status:
    SELECT and duration: 19209

------------------- REQUEST: UPDATE -------------------

46  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@39] - Method "send" has been invoked...
47  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@40] - clientID: 4ab9cbe0-baa6-416f-acee-849
    d32ccf27f
48  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@41] - status: UPDATE
49  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@42] - timestamp: 1465763442546
50  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@43] - procedure data: Hello world!
51  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@205] - Method "doUpdate" starts sleeping for
    9346...
52  2016-06-12 23:30:46,106DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms

------------------- MONITOR ELEMENTS / SCHEDULER [2] -------------------
```

```
53  2016-06-12 23:30:51,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@26] - [MonitoringRunnable]: Start monitoring...
54  2016-06-12 23:30:51,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@37] - [MonitoringRunnable]: Element-1465763442539 with
     status: SELECT and duration: 19209
55  2016-06-12 23:30:51,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@50] - [MonitoringRunnable]: SrvAvg = 16.0
56  2016-06-12 23:30:51,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@51] - [MonitoringRunnable]: Time Added = 19209.0
57  2016-06-12 23:30:51,099INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 19225.0
58  2016-06-12 23:30:51,101DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 10,3  replyHeader:: 10,6659,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6654,6659,1465763351972,1465763431102,1,0,0,96057447759740951,4,0,6654}
59  2016-06-12 23:30:51,102DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 2ms
60  2016-06-12 23:30:51,106DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
61  2016-06-12 23:30:51,107DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
62  2016-06-12 23:30:51,111DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 11,5  replyHeader:: 11,6660,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#31393232352e30,1  response:: s
    {6654,6660,1465763351972,1465763451102,2,0,0,96057447759740951,7,0,6654}
63  2016-06-12 23:30:51,113INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
64  2016-06-12 23:30:51,114INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@350] - [Server]: Path /srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 Updated!!!

65  2016-06-12 23:30:51,363INFO -[pool-2-thread-1]/[o.thesis.project.server.
    ClientsControl$1@40] - [ClientsControl]: Expired Clients remover is
    starting...

------------------- UPDATE FINISHED -------------------

66  2016-06-12 23:30:51,895INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@207] - Method "doUpdate" woke up...
67  2016-06-12 23:30:51,896INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@463] - [Server]: Client-4ab9cbe0-baa6-416f-acee-849
    d32ccf27f last active on :Sun Jun 12 23:30:51 EEST 2016
```

```
68  2016-06-12 23:30:51,896INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763451896 with status:
    UPDATE and duration: 9349

------------------- REQUEST: INSERT -------------------

69  2016-06-12 23:30:51,902INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@39] - Method "send" has been invoked...
70  2016-06-12 23:30:51,902INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@40] - clientID: 4ab9cbe0-baa6-416f-acee-849
    d32ccf27f
71  2016-06-12 23:30:51,903INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@41] - status: INSERT
72  2016-06-12 23:30:51,903INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@42] - timestamp: 1465763451900
73  2016-06-12 23:30:51,903INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@43] - procedure data: Hello world!
74  2016-06-12 23:30:51,903INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@181] - Method "doInsert" starts sleeping for
    5814...
75  2016-06-12 23:30:56,106DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms

------------------- INSERT FINISHED -------------------

76  2016-06-12 23:30:57,718INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@183] - Method "doInsert" woke up...
77  2016-06-12 23:30:57,719INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@463] - [Server]: Client-4ab9cbe0-baa6-416f-acee-849
    d32ccf27f last active on :Sun Jun 12 23:30:57 EEST 2016


78  2016-06-12 23:30:57,719INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763451896 with status:
    UPDATE and duration: 9349
79  2016-06-12 23:30:57,719INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763457719 with status:
    INSERT and duration: 5816

------------------- REQUEST: DELETE -------------------

80  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@39] - Method "send" has been invoked...
81  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@40] - clientID: 4ab9cbe0-baa6-416f-acee-849
    d32ccf27f
82  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@41] - status: DELETE
83  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@42] - timestamp: 1465763457721
84  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@43] - procedure data: Hello world!
```

```
85  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@230] - Method "doDelete" starts sleeping for
    13434...

------------------- MONITOR ELEMENTS / SCHEDULER [3] -------------------

86  2016-06-12 23:31:01,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@26] - [MonitoringRunnable]: Start monitoring...
87  2016-06-12 23:31:01,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@37] - [MonitoringRunnable]: Element-1465763451896 with
     status: UPDATE and duration: 9349
88  2016-06-12 23:31:01,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@37] - [MonitoringRunnable]: Element-1465763457719 with
     status: INSERT and duration: 5816
89  2016-06-12 23:31:01,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@50] - [MonitoringRunnable]: SrvAvg = 19225.0
90  2016-06-12 23:31:01,098INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@51] - [MonitoringRunnable]: Time Added = 7582.5
91  2016-06-12 23:31:01,099INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 26807.5
92  2016-06-12 23:31:01,100DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 12,3  replyHeader:: 12,6660,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6654,6660,1465763351972,1465763451102,2,0,0,96057447759740951,7,0,6654}
93  2016-06-12 23:31:01,101DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 2ms
94  2016-06-12 23:31:01,104DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
95  2016-06-12 23:31:01,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
96  2016-06-12 23:31:01,105INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
97  2016-06-12 23:31:01,106DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 13,5  replyHeader:: 13,6661,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#32363830372e35,2  response:: s
    {6654,6661,1465763351972,1465763461101,3,0,0,96057447759740951,7,0,6654}
98  2016-06-12 23:31:01,107INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@350] - [Server]: Path /srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 Updated!!!


99  2016-06-12 23:31:01,364INFO -[pool-2-thread-1]/[o.thesis.project.server.
    ClientsControl$1@40] - [ClientsControl]: Expired Clients remover is
```

```
    starting...
100 2016-06-12 23:31:06,103DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms
101 2016-06-12 23:31:11,097INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@26] - [MonitoringRunnable]: Start monitoring...
102 2016-06-12 23:31:11,104DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms

------------------- DELETE FINISHED -------------------

103 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@232] - Method "doDelete" woke up...
104 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@463] - [Server]: Client-4ab9cbe0-baa6-416f-acee-849
    d32ccf27f last active on :Sun Jun 12 23:31:11 EEST 2016
105 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763471159 with status:
    DELETE and duration: 13437

------------------- CLIENT DISCONNECT -------------------

106 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@250] - Method "bye" has been invoked...
107 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@251] - clientID: 4ab9cbe0-baa6-416f-acee-849
    d32ccf27f
108 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@366] - [Server]: --Clients-- {} 0
109 2016-06-12 23:31:11,159DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 14,3  replyHeader:: 14,6661,0  request:: '/srvClients/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6656,6658,1465763351972,1465763423299,1,0,0,96057447759740951,1,0,6656}
110 2016-06-12 23:31:11,175DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
111 2016-06-12 23:31:11,175DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvClients/server-d2b8d3b9-246f
    -40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
112 2016-06-12 23:31:11,175DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 15,5  replyHeader:: 15,6662,0  request:: '/srvClients/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#30,1  response:: s
    {6656,6662,1465763351972,1465763471159,2,0,0,96057447759740951,1,0,6656}
113 2016-06-12 23:31:11,175INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvClients/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
```

```
114 2016-06-12 23:31:11,175DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 16,3  replyHeader:: 16,6662,0  request:: '/srvTotalRuntime/
    server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6655,6655,1465763351972,1465763351972,0,0,0,96057447759740951,1,0,6655}
115 2016-06-12 23:31:11,188DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
116 2016-06-12 23:31:11,188DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvTotalRuntime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
117 2016-06-12 23:31:11,189DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 17,5  replyHeader:: 17,6663,0  request:: '/srvTotalRuntime/
    server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#3437383630,0  response:: s
    {6655,6663,1465763351972,1465763471175,1,0,0,96057447759740951,5,0,6655}
118 2016-06-12 23:31:11,189INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@421] - [Server]: Removing client 4ab9cbe0-baa6-416f-
    acee-849d32ccf27f from map...
119 2016-06-12 23:31:11,190INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@425] - ==== CLIENTS ====
120 2016-06-12 23:31:11,190INFO -[New I/O worker #1]/[org.thesis.project.
    server.AvroServer@433] - =============
121 2016-06-12 23:31:11,191INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
122 2016-06-12 23:31:11,193INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x90f593ef,
    /127.0.0.1:63063 :> /127.0.0.1:63024] DISCONNECTED
123 2016-06-12 23:31:11,193INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x90f593ef,
    /127.0.0.1:63063 :> /127.0.0.1:63024] UNBOUND
124 2016-06-12 23:31:11,193INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@171] - [id: 0x90f593ef,
    /127.0.0.1:63063 :> /127.0.0.1:63024] CLOSED
125 2016-06-12 23:31:11,193INFO -[New I/O worker #1]/[o.a.a.i.
    NettyServer$NettyServerAvroHandler@209] - Connection to /127.0.0.1:63063
    disconnected.


126 2016-06-12 23:31:11,374INFO -[pool-2-thread-1]/[o.thesis.project.server.
    ClientsControl$1@40] - [ClientsControl]: Expired Clients remover is
    starting...
127 2016-06-12 23:31:16,179DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 0ms

------------------- MONITOR ELEMENTS / SCHEDULER [4]-------------------
```

```
128 2016-06-12 23:31:21,103INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@26] - [MonitoringRunnable]: Start monitoring...
129 2016-06-12 23:31:21,103INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@37] - [MonitoringRunnable]: Element-1465763471159 with
     status: DELETE and duration: 13437
130 2016-06-12 23:31:21,103INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@50] - [MonitoringRunnable]: SrvAvg = 26807.5
131 2016-06-12 23:31:21,103INFO -[pool-1-thread-1]/[o.t.project.server.
    MonitoringRunnable@51] - [MonitoringRunnable]: Time Added = 13437.0
132 2016-06-12 23:31:21,103INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 40244.5
133 2016-06-12 23:31:21,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 18,3  replyHeader:: 18,6663,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,T  response:: s
    {6654,6661,1465763351972,1465763461101,3,0,0,96057447759740951,7,0,6654}
134 2016-06-12 23:31:21,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@717] - Got ping response for sessionid:
    0x15543bd63c00017 after 1ms
135 2016-06-12 23:31:21,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid:0
    x15543bd63c00017
136 2016-06-12 23:31:21,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246
    f-40cb-ba31-bd5824a483e7 for sessionid 0x15543bd63c00017
137 2016-06-12 23:31:21,105DEBUG-[main-SendThread(127.0.0.1:2181)]/[o.apache
    .zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c00017, packet:: clientPath:null serverPath:null finished:false
     header:: 19,5  replyHeader:: 19,6664,0  request:: '/srvUpAvgTime/server-
    d2b8d3b9-246f-40cb-ba31-bd5824a483e7,#34303234342e35,3  response:: s
    {6654,6664,1465763351972,1465763481105,4,0,0,96057447759740951,7,0,6654}
138 2016-06-12 23:31:21,112INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-
    bd5824a483e7, localhost
```

Στις γραμμές 1-3 βλέπουμε την εδραίωση της σύνδεσης πελάτη με εξυπηρετητή. Στη συνέχεια ο πελάτης στέλνει τα αιτήματα προς τη βάση. Το πρώτο αίτημα είναι τύπου INIT [γραμμές 4-18] όπου ο εξυπηρετητής δημιουργεί το αντίστοιχο αντικείμενο και κρατά τα στοιχεία του πελάτη [γραμμές 7-9]. Ενώ στη συνέχεια ενημερώνει τον Zookeeper για τον νέο αριθμό πελατών που εξυπηρετεί [γραμμές 14-18]. Στη συνέχεια στέλνονται και τα επόμενα αιτήματα [γραμμές 20-26, 46-52, 69-75, 78-85] όπου σε κάθε αίτημα προκειμένου να προσομοιωθεί ο χρόνος εκτέλεσης του ο εξυπηρετητής «κοιμάται» για ένα τυχαίο χρονικό διάστημα [γραμμή 25]. Κάθε φορά που ολοκληρώνεται ένα αίτημα [γραμμές 43-45, 66-68, 76-79, 103-105] γράφεται σε μια λίστα ο χρόνος ολοκλήρωσής του, ο οποίος παρατηρούμε έχει μια αμελητέα απόκλιση από τον χρόνο που «κοιμήθηκε» ο εξυπηρετητής το οποίο είναι αναμενόμενο γιατί στην

λίστα δεν γράφεται ο χρόνος αυτός αλλά ο ολικός χρόνος από τη στιγμή που έφτασε το αίτημα μέχρι τη στιγμή που ξύπνησε. Στο διάστημα λειτουργίας του εξυπηρετητή τρέχουν και δυο δρομολογητές, σε τακτά χρονικά διαστήματα, ο ένας για να αφαιρεί συνδέσεις που έχουν λήξει [γραμμές 38, 42, 65, 99, 126] και ο άλλος για να υπολογίζει τον μέσο χρόνο λειτουργίας [γραμμές 27, 40,53, 86, 147, 185]. Για τη λειτουργία του πρώτου δρομολογητή θα δούμε σε παρακάτω ενότητα. Για τον δεύτερο δρομολογητή όπως φαίνεται συγκεντρωτικά και στο Σχήμα: 6.13 υπολογίζει τον μέσο χρόνο από τους χρόνους που βρίσκει κάθε φορά στην αντίστοιχη λίστα και τον προσθέτει σε αυτόν που υπήρχε προηγουμένως [π.χ. γραμμές 86-91]. Και στη συνέχεια ενημερώνει τον Zookeeper για την νέα τιμή.

```
19  2016-06-12 23:30:23,315INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763423315 with status:
    INIT and duration: 16

31  2016-06-12 23:30:31,098INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 16.0
```

```
 25  2016-06-12 23:30:23,330INFO -[New I/O worker #1]/[o.thesis.project.
     server.ServerClientImpl@157] - Method "doSelect" starts sleeping for
     19201...

 45  2016-06-12 23:30:42,539INFO -[New I/O worker #1]/[org.thesis.project.
     server.Monitoring@43] - [Monitor]: element-1465763442539 with status:
     SELECT and duration: 19209

 57  2016-06-12 23:30:51,099INFO -[pool-1-thread-1]/[org.thesis.project.
     server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
     -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 19225.0
```

```
51  2016-06-12 23:30:42,547INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@205] - Method "doUpdate" starts sleeping for
    9346...

68  2016-06-12 23:30:51,896INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763451896 with status:
    UPDATE and duration: 9349

74  2016-06-12 23:30:51,903INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@181] - Method "doInsert" starts sleeping for
    5814...

79  2016-06-12 23:30:57,719INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763457719 with status:
    INSERT and duration: 5816

91  2016-06-12 23:31:01,099INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 26807.5
```

```
85  2016-06-12 23:30:57,722INFO -[New I/O worker #1]/[o.thesis.project.
    server.ServerClientImpl@230] - Method "doDelete" starts sleeping for
    13434...

105 2016-06-12 23:31:11,159INFO -[New I/O worker #1]/[org.thesis.project.
    server.Monitoring@43] - [Monitor]: element-1465763471159 with status:
    DELETE and duration: 13437

132 2016-06-12 23:31:21,103INFO -[pool-1-thread-1]/[org.thesis.project.
    server.AvroServer@348] - [Server]: Updating /srvUpAvgTime/server-d2b8d3b9
    -246f-40cb-ba31-bd5824a483e7srvUpAvgTime = 40244.5
```


Σχήμα 6.13: Server's Up Average Time Calculation

Παρακάτω δίνονται στιγμιότυπα από τον Zookeeper κατά τη διάρκεια σύνδεσης του εξυπηρετητή με τον πελάτη.

Πριν την σύνδεση με τον πελάτη

```
[zk: localhost:2181(CONNECTED) 42] get /srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
0
cZxid = 0x1a00
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x1a00
mtime = Sun Jun 12 23:29:11 EEST 2016
pZxid = 0x1a00
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.14: Total Clients Before

```
[zk: localhost:2181(CONNECTED) 41] get /srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
0
cZxid = 0x19ff
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x19ff
mtime = Sun Jun 12 23:29:11 EEST 2016
pZxid = 0x19ff
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.15: Total Runtime Before

```
[zk: localhost:2181(CONNECTED) 43] get /srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
0
cZxid = 0x19fe
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x19fe
mtime = Sun Jun 12 23:29:11 EEST 2016
pZxid = 0x19fe
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.16: Total Average Time Before

Κατά τη διάρκεια της σύνδεσης με τον πελάτη

```
[zk: localhost:2181(CONNECTED) 47] get /srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
1
cZxid = 0x1a00
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x1a02
mtime = Sun Jun 12 23:30:23 EEST 2016
pZxid = 0x1a00
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.17: Total Clients During

```
[zk: localhost:2181(CONNECTED) 46] get /srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
0
cZxid = 0x19ff
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x19ff
mtime = Sun Jun 12 23:29:11 EEST 2016
pZxid = 0x19ff
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.18: Total Runtime During

Μέτρηση στη φάση του INIT

```
zk: localhost:2181(CONNECTED) 48] get /srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
16.0
Zxid = 0x19fe
time = Sun Jun 12 23:29:11 EEST 2016
Zxid = 0x1a03
time = Sun Jun 12 23:30:31 EEST 2016
Zxid = 0x19fe
version = 0
ataVersion = 1
clVersion = 0
phemeralOwner = 0x15543bd63c00017
ataLength = 4
umChildren = 0
zk: localhost:2181(CONNECTED) 49] get /srvUpAvgTime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
6807.5
Zxid = 0x19fe
time = Sun Jun 12 23:29:11 EEST 2016
Zxid = 0x1a05
time = Sun Jun 12 23:31:01 EEST 2016
Zxid = 0x19fe
version = 0
ataVersion = 3
clVersion = 0
phemeralOwner = 0x15543bd63c00017
ataLength = 7
umChildren = 0
```

Σχήμα 6.19: Total Average Time During

Στο τέλος της σύνδεσης με τον πελάτη

```
[zk: localhost:2181(CONNECTED) 42] get /srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
0
cZxid = 0x1a00
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x1a00
mtime = Sun Jun 12 23:29:11 EEST 2016
pZxid = 0x1a00
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.20: Total Clients After

```
[zk: localhost:2181(CONNECTED) 50] get /srvTotalRuntime/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
47860
cZxid = 0x19ff
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x1a07
mtime = Sun Jun 12 23:31:11 EEST 2016
pZxid = 0x19ff
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 5
numChildren = 0
[zk: localhost:2181(CONNECTED) 51] get /srvClients/server-d2b8d3b9-246f-40cb-ba31-bd5824a483e7
0
cZxid = 0x1a00
ctime = Sun Jun 12 23:29:11 EEST 2016
mZxid = 0x1a06
mtime = Sun Jun 12 23:31:11 EEST 2016
pZxid = 0x1a00
cversion = 0
dataVersion = 2
aclVersion = 0
ephemeralOwner = 0x15543bd63c00017
dataLength = 1
numChildren = 0
```

Σχήμα 6.21: Total Runtime After

6.4.3 Λειτουργία ClientsControl Scheduler

Σε περίπτωση που μια σύνδεση ξεπεράσει τον επιτρεπτό χρόνο που είναι αδρανής ο ClientsControl Scheduler αφαιρεί τον πελάτη από την λίστα του και κλείνει την σύνδεση. Παρακάτω δίνεται απόσπασμα από το αρχείο Log όπου φαίνεται η συγκεκριμένη ενέργεια.

Listing 6.10: Expired Connection Server Log

```
1  2016-06-12 23:15:31,900INFO -[pool-2-thread-1]/[o.thesis.project.server.
   ClientsControl$1@40] - [ClientsControl]: Expired Clients remover is
   starting...
2  2016-06-12 23:15:31,900INFO -[pool-2-thread-1]/[org.thesis.project.
   server.AvroServer@421] - [Server]: Removing client e242d373-b435-4970-
```

```
    bdb0-e23fd63458bd from map...
3   2016-06-12 23:15:31,900INFO -[pool-2-thread-1]/[org.thesis.project.
    server.AvroServer@425] - ==== CLIENTS ====
4   2016-06-12 23:15:31,900INFO -[pool-2-thread-1]/[org.thesis.project.
    server.AvroServer@433] - =============
5   2016-06-12 23:15:31,900INFO -[pool-2-thread-1]/[org.thesis.project.
    server.AvroServer@366] - [Server]: --Clients-- {} 0
6   2016-06-12 23:15:31,900DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000e, packet:: clientPath:null serverPath:null finished:false
     header:: 10,3  replyHeader:: 10,6576,0  request:: '/srvClients/server-
    dd7129f1-f8cf-4188-977c-7b6bc959b6b7,T  response:: s
    {6573,6575,1465762312521,1465762321622,1,0,0,96057447759740942,1,0,6573}
7   2016-06-12 23:15:31,900DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@717] - Got ping response for
    sessionid: 0x15543bd63c0000e after 2ms
8   2016-06-12 23:15:31,900DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@741] - Got notification sessionid
    :0x15543bd63c0000e
9   2016-06-12 23:15:31,900DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@763] - Got WatchedEvent state:
    SyncConnected type:NodeDataChanged path:/srvClients/server-dd7129f1-f8cf
    -4188-977c-7b6bc959b6b7 for sessionid 0x15543bd63c0000e
10  2016-06-12 23:15:31,900DEBUG-[main-SendThread(0:0:0:0:0:0:0:1:2181)]/[o.
    apache.zookeeper.ClientCnxn$SendThread@818] - Reading reply sessionid:0
    x15543bd63c0000e, packet:: clientPath:null serverPath:null finished:false
     header:: 11,5  replyHeader:: 11,6577,0  request:: '/srvClients/server-
    dd7129f1-f8cf-4188-977c-7b6bc959b6b7,#30,1  response:: s
    {6573,6577,1465762312521,1465762531900,2,0,0,96057447759740942,1,0,6573}
11  2016-06-12 23:15:31,900INFO -[main-EventThread]/[org.thesis.project.
    server.AvroServer@144] - WatchedEvent state:SyncConnected type:
    NodeDataChanged path:/srvClients/server-dd7129f1-f8cf-4188-977c-7
    b6bc959b6b7, localhost
```

Στις γραμμές 1-5 βλέπουμε να ενεργοποιείται ο δρομολογητής και να αφαιρεί τον συγκεκριμένο πελάτη από την λίστα του εξυπηρετητή. Ενώ στις επόμενες γραμμές φαίνεται να ειδοποιεί ο εξυπηρετητής τον Zookeeper για την μείωση τον πελατών που εξυπηρετεί.

6.5 Λειτουργία πελάτη

Ο πελάτης ενεργοποιείται μέσω της κλάσης User . Παρακάτω δίνεται το αρχείο Log του.

Listing 6.11: Client Log

```
------------------- CONNECT TO MASTER -------------------

1   2016-06-12 23:30:22,663DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@198] - Using Netty bootstrap options: {
    connectTimeoutMillis=60000, tcpNoDelay=true, keepAlive=true}
```

```
2   2016-06-12 23:30:22,674DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@265] - Connecting to localhost/127.0.0.1:8081
3   2016-06-12 23:30:22,691DEBUG-[main]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x95b5b6d4] OPEN
4   2016-06-12 23:30:22,707DEBUG-[New I/O worker #1]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x95b5b6d4,
    /127.0.0.1:63044 => localhost/127.0.0.1:8081] BOUND: /127.0.0.1:63044
5   2016-06-12 23:30:22,707DEBUG-[New I/O worker #1]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x95b5b6d4,
    /127.0.0.1:63044 => localhost/127.0.0.1:8081] CONNECTED: localhost
    /127.0.0.1:8081
6   2016-06-12 23:30:23,019INFO -[main]/[org.thesis.project.client.Client@82
    ] - [Client] : Method 'sayHello' was invoked...

------------------- DISCONNECT FROM MASTER -------------------

7   2016-06-12 23:30:23,183DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@339] - Disconnecting from localhost/127.0.0.1:8081
8   2016-06-12 23:30:23,183DEBUG-[New I/O worker #1]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x95b5b6d4,
    /127.0.0.1:63044 :> localhost/127.0.0.1:8081] DISCONNECTED
9   2016-06-12 23:30:23,183DEBUG-[New I/O worker #1]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x95b5b6d4,
    /127.0.0.1:63044 :> localhost/127.0.0.1:8081] UNBOUND
10  2016-06-12 23:30:23,199DEBUG-[New I/O worker #1]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x95b5b6d4,
    /127.0.0.1:63044 :> localhost/127.0.0.1:8081] CLOSED
11  2016-06-12 23:30:23,199DEBUG-[New I/O worker #1]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@579] - Remote peer localhost
    /127.0.0.1:8081 closed connection.

------------------- CONNECT TO SERVER -------------------

12  2016-06-12 23:30:23,214DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@198] - Using Netty bootstrap options: {
    connectTimeoutMillis=60000, tcpNoDelay=true, keepAlive=true}
13  2016-06-12 23:30:23,214DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@265] - Connecting to /127.0.0.1:63024
14  2016-06-12 23:30:23,214DEBUG-[main]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x7bb5e07b] OPEN
15  2016-06-12 23:30:23,214DEBUG-[New I/O worker #10]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x7bb5e07b,
    /127.0.0.1:63063 => /127.0.0.1:63024] BOUND: /127.0.0.1:63063
16  2016-06-12 23:30:23,214DEBUG-[New I/O worker #10]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x7bb5e07b,
    /127.0.0.1:63063 => /127.0.0.1:63024] CONNECTED: /127.0.0.1:63024

------------------- SEND REQUESTS -------------------

17  2016-06-12 23:30:23,230INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
18  2016-06-12 23:30:23,330INFO -[main]/[org.thesis.project.client.
    Connection@47] - [Connection]: Reply from server: OK
```

```
19  2016-06-12 23:30:23,330ERROR-[main]/[org.thesis.project.client.
    Connection@69] - [Connection]: Doing SELECT action
20  2016-06-12 23:30:23,330INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
21  2016-06-12 23:30:42,539INFO -[main]/[org.thesis.project.client.User@32]
    - [User]: Method "doSelect" was invoked... Result : OK
22  2016-06-12 23:30:42,539ERROR-[main]/[org.thesis.project.client.
    Connection@69] - [Connection]: Doing UPDATE action
23  2016-06-12 23:30:42,546INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
24  2016-06-12 23:30:51,897INFO -[main]/[org.thesis.project.client.User@37]
    - [User]: Method "doUpdate" was invoked... Result : OK
25  2016-06-12 23:30:51,900ERROR-[main]/[org.thesis.project.client.
    Connection@69] - [Connection]: Doing INSERT action
26  2016-06-12 23:30:51,900INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
27  2016-06-12 23:30:57,720INFO -[main]/[org.thesis.project.client.User@40]
    - [User]: Method "doInsert" was invoked... Result : OK
28  2016-06-12 23:30:57,721ERROR-[main]/[org.thesis.project.client.
    Connection@69] - [Connection]: Doing DELETE action
29  2016-06-12 23:30:57,721INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
30  2016-06-12 23:31:11,159INFO -[main]/[org.thesis.project.client.User@43]
    - [User]: Method "doDelete" was invoked... Result : OK
31  2016-06-12 23:31:11,159INFO -[main]/[org.thesis.project.client.
    Client@150] - [Client] : Method 'close' was invoked...
32  2016-06-12 23:31:11,191INFO -[main]/[org.thesis.project.client.
    Connection@111] - [Connection] Close connection with server :OK

------------------- DISCONNECT FROM SERVER -------------------

33  2016-06-12 23:31:11,191DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@339] - Disconnecting from /127.0.0.1:63024
34  2016-06-12 23:31:11,192DEBUG-[New I/O worker #10]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x7bb5e07b,
    /127.0.0.1:63063 :> /127.0.0.1:63024] DISCONNECTED
35  2016-06-12 23:31:11,193DEBUG-[New I/O worker #10]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x7bb5e07b,
    /127.0.0.1:63063 :> /127.0.0.1:63024] UNBOUND
36  2016-06-12 23:31:11,194DEBUG-[New I/O worker #10]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x7bb5e07b,
    /127.0.0.1:63063 :> /127.0.0.1:63024] CLOSED
37  2016-06-12 23:31:11,194DEBUG-[New I/O worker #10]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@579] - Remote peer
    /127.0.0.1:63024 closed connection.

------------------- CONNECT TO MASTER -------------------

38  2016-06-12 23:31:11,214DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@198] - Using Netty bootstrap options: {
    connectTimeoutMillis=60000, tcpNoDelay=true, keepAlive=true}
39  2016-06-12 23:31:11,214DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@265] - Connecting to localhost/127.0.0.1:8081
```

```
40  2016-06-12 23:31:11,214DEBUG-[main]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x8c3c9fd1] OPEN
41  2016-06-12 23:31:11,214DEBUG-[New I/O worker #19]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x8c3c9fd1,
    /127.0.0.1:63085 => localhost/127.0.0.1:8081] BOUND: /127.0.0.1:63085
42  2016-06-12 23:31:11,214DEBUG-[New I/O worker #19]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x8c3c9fd1,
    /127.0.0.1:63085 => localhost/127.0.0.1:8081] CONNECTED: localhost
    /127.0.0.1:8081
43  2016-06-12 23:31:11,214INFO -[main]/[org.thesis.project.client.
    Client@161] - [Client]: Close connection with MasterOK

------------------- DISCONNECT FROM MASTER -------------------

44  2016-06-12 23:31:11,214DEBUG-[main]/[org.apache.avro.ipc.
    NettyTransceiver@339] - Disconnecting from localhost/127.0.0.1:8081
45  2016-06-12 23:31:11,214DEBUG-[New I/O worker #19]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x8c3c9fd1,
    /127.0.0.1:63085 :> localhost/127.0.0.1:8081] DISCONNECTED
46  2016-06-12 23:31:11,214DEBUG-[New I/O worker #19]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x8c3c9fd1,
    /127.0.0.1:63085 :> localhost/127.0.0.1:8081] UNBOUND
47  2016-06-12 23:31:11,214DEBUG-[New I/O worker #19]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@575] - [id: 0x8c3c9fd1,
    /127.0.0.1:63085 :> localhost/127.0.0.1:8081] CLOSED
48  2016-06-12 23:31:11,214DEBUG-[New I/O worker #19]/[o.a.a.i.
    NettyTransceiver$NettyClientAvroHandler@579] - Remote peer localhost
    /127.0.0.1:8081 closed connection.
49
```

6.5.1 Αποτέλεσμα λειτουργίας ClientsControl Scheduler

Παρακάτω δίνεται απόσπασμα από αρχείο Log του πελάτη όταν έχει μείνει α-νενεργός για περισσότερο από το επιτρεπτό χρόνο και ο εξυπηρετητής έκλεισε την σύνδεση.

Listing 6.12: Expired Connection Client Log

```
1   2016-06-13 21:44:39,088 INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
2   2016-06-13 21:44:39,197 INFO -[main]/[org.thesis.project.client.
    Connection@47] - [Connection]: Reply from server: OK
3   2016-06-13 21:44:39,197 ERROR-[main]/[org.thesis.project.client.
    Connection@69] - [Connection]: Doing SELECT action
4   2016-06-13 21:44:39,197 INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
5   2016-06-13 21:44:41,636 INFO -[main]/[org.thesis.project.client.User@32]
     - [User]: Method "doSelect" was invoked... Result : OK
6   2016-06-13 21:48:11,648 ERROR-[main]/[org.thesis.project.client.
    Connection@69] - [Connection]: Doing UPDATE action
7   2016-06-13 21:48:11,649 INFO -[main]/[org.thesis.project.client.
    Connection@98] - [Connection] : Method 'sendMessage' has been invoked...
```

```
8   Exception in thread "main" org.thesis.project.client.exceptions.
    UserExpiredException: Client-ca2f6a61-4187-454d-8b10-a6469be6f1ae:
    Connection expired...
9     at org.thesis.project.client.Client.doUpdate(Client.java:125)
10    at org.thesis.project.client.User.main(User.java:36)
```

Όπως βλέπουμε στην γραμμή 8 ο εξυπηρετητής έστειλε μήνυμα για τη σύνδεση και ο πελάτης δεν μπορεί να συνεχίσει με τα αιτήματα αλλά πρέπει να κλείσει την σύνδεση και να δημιουργήσει νέα.

Κεφάλαιο 7

Επίλογος

7.1 Σύνοψη

Στη διπλωματική εργασία αναπτύχθηκε ένα μοντέλο πελάτη-εξυπηρετητή, με τους εξυπηρετητές να ακολουθούν την αρχιτεκτονική master-worker όπου ένας εξυπηρετητής αναλάμβανε τον ρόλο του master που δέχεται τους πελάτες και τους αναθέτει κάποιον εξυπηρετητή. Στη συνέχεια ο πελάτης αποδεσμεύεται από τον master και επικοινωνεί κατευθείαν με τον εξυπηρετητή στέλνοντάς του τα αιτήματα για την βάση. Για τον συγχρονισμό των οντοτήτων χρησιμοποιήθηκε ως εργαλείο ο Apache Zookeeper στον οποίο οι εξυπηρετητές έγραφαν ό,τι στοιχεία άλλαζαν κατά τη σύνδεση τους με πελάτες και ο Zookeeper αναλάμβανε να ενημερώσει τον master παρέχοντας όλες τις εγγυήσεις που περιμένουμε σε ένα κατανεμημένο σύστημα.

Για την επικοινωνία χρησιμοποιήθηκε το εργαλείο Apache Avro το οποίο διευκόλυνε και απλοποίησε την ανταλλαγή μηνυμάτων μεταξύ όλων των οντοτήτων. Αναπτύχθηκαν δυο πρωτόκολλα Avro , ένα για την επικοινωνία μεταξύ πελάτη και master και ένα για την επικοινωνία μεταξύ πελάτη και εξυπηρετητή. Κατά την επικοινωνία πελάτη-master ο πελάτης στέλνει ένα μήνυμα με τα στοιχεία του και ο master του απαντάει με ένα μήνυμα με τη διεύθυνση και την ταυτότητα του εξυπηρετητή. Στη συνέχεια ο πελάτης συνδέεται με τον εξυπηρετητή και στέλνει σε αυτόν μηνύματα με τα αιτήματα προς την βάση ακολουθώντας την μορφή που ορίζει το δεύτερο πρωτόκολλο και ο εξυπηρετητής απαντάει ανάλογα με το αν ήταν επιτυχής ή όχι η εκτέλεσή του. Για την υλοποίηση της διπλωματικής επιλέχτηκε η γλώσσα προγραμματισμού Java σε συνδυασμό με το Apache Maven για καλύτερη οργάνωση και διαχείριση του κώδικα.

Τέλος, μετά την ολοκλήρωση της υλοποίησης του συστήματος προχωρήσαμε σε εκτέλεση σεναρίων χρήσης από τα οποία παράχθηκαν τα αντίστοιχα αρχεία Log που παρουσιάζονται στο κεφάλαιο 7 για τον έλεγχο ορθής λειτουργίας.

7.2 Μελλοντικές Επεκτάσεις

Το σύστημα των εξυπηρετητών που αναπτύχθηκε στην παρούσα διπλωματική είναι μια λύση η οποία μπορεί να λειτουργήσει αυτόνομα. Παρόλα αυτά δέχεται ακόμα κάποιες επεκτάσεις προκειμένου να μπορέσει να ενσωματωθεί στο CoherentPaaS . Αρχικά μια βελτίωση που επιδέχεται είναι στο γεγονός ότι ο Master τρέχει σε μια συγκεκριμένη διεύθυνση IP και όλοι οι πελάτες ξέρουν στατικά και ρωτούν στην συγκεκριμένη διεύθυνση. Θα μπορούσε να υλοποιηθεί ένα σύστημα όπου η διεύθυνση να μην είναι στατική και σε περίπτωση που άλλαζε ο Master να ενημερωνόντουσαν όλοι οι πελάτες για την νέα του διεύθυνση.

Επιπλέον δεν έχουν ενσωματωθεί ακόμα οι λειτουργίες που θα εκτελεί ο εξυπηρετητής όταν έρχονται τα αιτήματα από τον πελάτη. Μέχρι στιγμής ο πελάτης έστελνε dummies αιτήματα προς τον εξυπηρετητή και εκείνος προσομοίωνε το ν χρόνο εκτέλεσης του αιτήματος με ένα νήμα το οποίο «κοιμόταν» για ένα τυχαίο χρονικό διάστημα. Επομένως σε επόμενη φάση πρέπει να ενσωματωθούν οι λειτουργίες που πρέπει να εκτελούνται ανάλογα με τα αιτήματα που έρχονται από τον πελάτη, ώστε να ολοκληρωθεί το middleware .

Τέλος θα μπορούσε να υλοποιηθεί ένα σύστημα logging που να κρατάει την κατάσταση στην οποία βρίσκεται ο κάθε εξυπηρετητής ώστε σε περίπτωση βλάβης ο πελάτης να μη χρειάζεται να ξαναστέλνει από την αρχή όλα τα αιτήματα, αλλά το σφάλμα να επιλύεται εντός του δικτύου των εξυπηρετητών και να μη βγαίνει προς τα έξω.

Κεφάλαιο 8

Βιβλιογραφία

[1] The acid model for database management systems. http://databases.about.com/od/specificproducts/a/acid.htm.

[2] Apache avro java 1.8.1 api. http://avro.apache.org/docs/current/gettingstartedjava.html.

[3] Apache software foundation - wikipedia. https://en.wikipedia.org/wiki/Apache_Software_Foundation#Projects.

[4] Apache zookeeper - home. https://zookeeper.apache.org/.

[5] Apache zookeeper - wikipedia. https://en.wikipedia.org/wiki/Apache_ZooKeeper.

[6] Base model of database development. http://databases.about.com/od/otherdatabases/a/Abandoning-Acid-In-Favor-Of-Base.htm.

[7] A brief history of cloud computing - thoughts on cloud. http://www.thoughtsoncloud.com/2014/03/a-brief-history-of-cloud-computing/.

[8] Explaining apache zookeeper - stack overflow. http://stackoverflow.com/questions/3662995/explaining-apache-zookeeper.

[9] Java. http://www.medialab.ntua.gr/education/MultimediaTechnology/JavaNotes/files/1.\%20introduction.pdf.

[10] Java (programming language) - wikipedia. https://en.wikipedia.org/wiki/Java_\%28programming_language\%29.

[11] Maven – welcome to apache maven. http://maven.apache.org/.

[12] Nist sp 800-145, the nist definition of cloud computing. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[13] Nosql - wikipedia. https://en.wikipedia.org/wiki/NoSQL.

[14] Nosql databases defined. http://www.planetcassandra.org/what-is-nosql/.

[15] Nosql databases explained. https://www.mongodb.com/nosql-explained.

[16] Poweredby - apache zookeeper. https://cwiki.apache.org/confluence/display/ZOOKEEPER/PoweredBy.

[17] Ted Dunning. Zookeeper - a reliable, scalable distributed coordination system - high scalability.

[18] Torry Harris. Cloud computing overview. http://www.thbs.com/downloads/Cloud-Computing-Overview.pdf.

[19] Qusay F. Hassan. *Demystifying Cloud Computing.* PhD thesis, Mansoura University, Egypt, 2011.

[20] Tim Robertson. Developer blog: Getting started with avro rpc. http://gbif.blogspot.gr/2011/06/getting-started-with-avro-rpc.html.