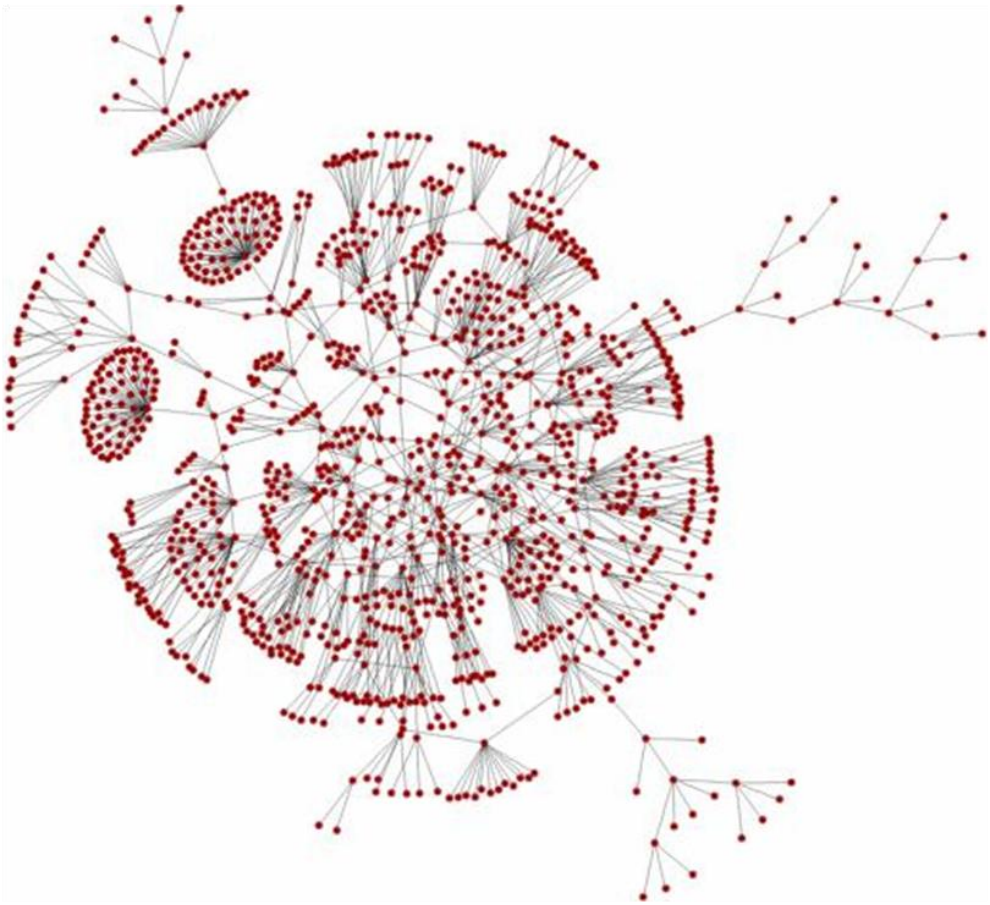
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ: Κατανεμημένα κοινωνικά δίκτυα: διαμοιρασμός βάσεων δεδομένων και εφαρμογών



ΕΠΙΜΕΛΕΙΑ:ΒΑΣΙΛΗΣ ΚΕΦΑΛΛΗΝΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:ΑΝΑΣΤΑΣΙΟΣ ΔΟΥΛΑΜΗΣ

ΧΑΝΙΑ ΝΟΕΜΒΡΙΟΣ 2013

Ευχαριστίες

Η εργασια αυτη δεν θα ειχε γινει πραγματικοτητα χωρις την εμπνευση που δινουν για την ελευθερια στο διαδικτυο ανθρωποι οπως ο Eben Moglen (Freedombox) , ο Ilya Zhitomirskiy (Diaspora Foundation) και ο Aaron Hillel Swart (Rss,reddit,creative commons). Επισης θελω να ευχαριστησω θερμα τον κυριο Αναστασιο Δουλαμη που υπομονετικα ανελαβε να με βοηθησει γνωριζοντας οτι ξεκιναω απο το μηδεν. Θελω να ευχαριστησω την οικογενεια μου και τους φιλους μου που με στηριζανε τα δυο αυτα χρονια.

Περιέχομενα

Περίληψη

Στη σημερινή κοινωνία που το διαδίκτυο αποτελεί ένα από τα κυρίαρχα μέσα ενημέρωσης, επικοινωνίας και ψυχαγωγίας, εμφανίζεται το πρόβλημα της καταπάτησης των προσωπικών δεδομένων των χρηστών από τις μεγάλες πολυεθνικές εταιρίες που τα διαχειρίζονται.

Με την αξιοποίηση της ελεύθερης και γρήγορης διακίνησης πληροφοριών που πλέον προσφέρει το διαδίκτυο, δόθηκε η δυνατότητα σε κάθε χρήστη να επικοινωνεί σε κάθε γωνιά του πλανήτη με άμεσο τρόπο. Για να το πετύχει αυτό είναι απαραίτητο να ενταχθεί στις υπηρεσίες κάποιας εταιρίας και να δημιουργήσει προσωπική σελίδα-profile μέσω της οποίας μπορεί να επικοινωνεί με άτομα της επιλογής του με διαδραστικό τρόπο. Η δομή όμως της τεχνολογίας μέσω της οποίας επιτυγχάνεται αυτή η επικοινωνία έχει αναπτυχθεί έτσι ώστε να επιβάλλεται η γνωστοποίηση της πληροφορίας, πέρα από τα άτομα που θα επιλέξει ο χρήστης, στις εταιρίες παροχής αυτών των υπηρεσιών. Αυτό συμβαίνει λόγω της υποχρεωτικής αποδοχής ασαφών όρων που επιβάλλονται για την εγγραφή στις υπηρεσίες τους. Συνέπεια αυτού αποτελεί η εγκαθίδρυση και νομιμοποίηση μια νέας έκδοσης του ‘’Μεγάλου Αδερφού’’ στη ζωή τους.

Η ανάγκη λοιπόν για μια εναλλακτική επιλογή που θα προσφέρει τις υπηρεσίες που επιζητούν οι χρήστες (ενημέρωση, κοινωνική δικτύωση, διοργάνωση γεγονότων) χωρίς την απαραίτητη εγγραφή σε υπηρεσίες συγκεκριμένων εταιριών και απαλλαγμένοι από τους όρους των, είναι επιβεβλημένη. Αλλιώς, δύσκολα εξασφαλίζεται η ελευθερία και η διαφορετικότητα του ατόμου μες στην κοινωνία και το δικαίωμα αυτού για διαφύλαξη των προσωπικών του δεδομένων.

Οι σύγχρονες τεχνολογίες δίνουν τη δυνατότητα υλοποίησης ενός νέου εναλλακτικού συστήματος, το οποίο θα προστατεύει τους χρήστες του και τα προσωπικά τους δεδομένα. Τα ζητούμενα χαρακτηριστικά ενός τέτοιου συστήματος διαμορφώνονται ως εξής:

- Δυνατότητα επιλογής από το χρήστη της φυσικής τοποθεσίας αποθήκευσης των προσωπικών του δεδομένων.
- Η ποιότητα των υπηρεσιών πρέπει να είναι ανάλογη εκείνων που ως τώρα απολαμβάνει ο χρήστης από τις αντίστοιχες κεντρικοποιημένες υπηρεσίες.

Η λύση έγκειται σε κατανεμημένα και αποκεντρωμένα συστήματα εξυπηρέτησης του διαδικτύου όπου η διακίνηση των ατομικών πληροφοριών των χρηστών θα γίνεται συνδυαστικά με συνεργασία πολύ μεγαλύτερου αριθμού εξυπηρετητών που ο κάθε ένας από αυτούς ανήκει σε κάποιον από τους χρήστες.

Το βασικό εγχείρημα της διπλωματικής αυτής είναι η ανάπτυξη ενός κατανεμημένου, ομότιμα διαμορφωμένου συστήματος κοινωνικής δικτύωσης. Η συνδυαστική χρήση υπάρχουσας ευρέως χρησιμοποιούμενης “ανοιχτής” τεχνολογίας θα βασιστεί σε μια πλατφόρμα βάσης δεδομένων που επιτρέπει τον εύκολο συγχρονισμό των δεδομένων μεταξύ χρηστών καθώς και τη χρήση ενός ανοιχτού πρωτοκόλλου εύκολης διαδικτυακής αναζήτησης και δημιουργίας επαφών.

Λεξεις κλειδια:Κατανεμημενα δικτυα, diaspora, social networks

1ο Κεφάλαιο

1.1 Εισαγωγή

Σήμερα τα Online Social Networks - OSN έχουν αναπτυχθεί με βάση το πρότυπο Web 2.0 (1). Ο όρος Web 2.0 ή Ιστός 2.0 χρησιμοποιείται για να περιγράψει τη νέα γενιά του Παγκόσμιου Ιστού, η οποία βασίζεται στην συνεχώς αυξανομένη δυνατότητα των χρηστών του διαδικτύου να συνεργάζονται και να μοιράζονται πληροφορίες με σύγχρονο και ασύγχρονο τρόπο.

Αυτή η νέα γενιά του Παγκόσμιου Ιστού είναι μια δυναμική διαδικτυακή πλατφόρμα στην οποία μπορούν να αλληλεπιδρούν χρήστες χωρίς εξειδικευμένες γνώσεις σε θέματα υπολογιστών και δικτύων (1). Σύμφωνα με αυτό το πρότυπο, η δυνατότητα επικοινωνίας μεταξύ των χρηστών με τη βοήθεια μιας διαδικτυακής υπηρεσίας, χρησιμοποιείται ολοένα και περισσότερο από τους ανθρώπους. Αυτό έχει οδηγήσει στη δημιουργία πλατφόρμων Online Κοινωνικών Δικτύων (2) όπως, το Facebook , Twitter , το LinkedIn κ.α. Εκατομμύρια χρήστες έχουν δημιουργήσει το προσωπικό τους προφίλ σε αυτές τις διαδικτυακές πλατφόρμες, αναρτώντας σε αυτές ένα μεγάλο αριθμό προσωπικών δεδομένων όπως φωτογραφίες, βίντεο και κείμενα, Όμως όλα αυτά τα δίκτυα βασίζονται στην ύπαρξη κεντρικών διαδικτυακών εξυπηρετητών. Έτσι, τα δεδομένα που αναρτούν οι χρήστες σε αυτές τις πλατφόρμες μπορεί πολύ εύκολα να παραβιαστούν ή να πωληθούν σε άλλες εταιρείες ως αποτέλεσμα κανείς από τους χρήστες, δε θα γνωρίζει που βρίσκονται τα δεδομένα τους και με ποιόν τρόπο έχουν χρησιμοποιηθεί ή θα χρησιμοποιηθούν στο απώτερο μέλλον.

Οπότε μια λύση για τον έλεγχο των δεδομένων των χρηστών από τους ίδιους τους χρήστες σε πλατφόρμες κοινωνικής δικτύωσης είναι η δημιουργία ενός κατανεμημένου κοινωνικού δικτύου χρησιμοποιώντας την υποδομή και την τεχνολογία ενός δικτύου ομότιμων οντοτήτων peer-to- peer, χωρίς την αλλοίωση των δυνατοτήτων που προσφέρει ένα OSN. Μέσω αυτού του τύπου κοινωνικού δικτύου ο χρήστης θα έχει τον πλήρη έλεγχο στα δεδομένα του και θα ορίζει αυτός τα δικαιώματα πρόσβασης σε τρίτους.

1.2 Ανασκόπηση της σημερινής κατάστασης

Ιστοσελίδες όπως το Facebook ,το Twitter, το MySpace κ.α. όπως επίσης και ιστοσελίδες που το περιεχόμενο τους βασίζεται σε κοινότητες και που προσφέρουν δυνατότητες κοινωνικής δικτύωσης (Youtube ,Wikipedia ,Blogspot ,Ask ,About ,eHow ,Answers) έχουν κεντρίσει το ενδιαφέρον εκατομμύρια χρηστών και εκατομμύρια κεφαλαίου από καπιταλιστικές επιχειρήσεις. Συμφώνα με το Compete.com για το 2011 (3) ανάμεσα στις πρώτες σε επισκεψιμότητα ιστοσελίδες παγκοσμίως οι μισές τουλάχιστον αποτελούν σελίδες κοινωνικής δικτύωσης και σελίδες περιεχομένου βασισμένου σε κοινότητες εθελοντών.

Figure 1.1 Οι κοινωνικοτητα στο διαδικτυο εκφραζεται με διαφορα μεσα

Οι υπηρεσίες κοινωνικής δικτύωσης προσφέρουν παρόμοιες λειτουργιές μεταξύ τους όπως : δίκτυα με λίστες φίλων, προσωπικά μηνύματα ,φόρουμ συζήτησης ,δημιουργία κοινοτήτων ,διαχείριση και κοινοποίηση εκδηλώσεων, αρθρογραφία , ερασιτεχνική δημοσιογραφία ,μεταφόρτωση πολυμέσων και άλλα. Με τέτοιες λειτουργίες, οι υπηρεσίες κοινωνικής δικτύωσης δείχνουν το πως το διαδίκτυο συνεχίζει να συνδέει καλύτερα ανθρώπους για κοινωνικούς και επαγγελματικούς σκοπούς.

1.3 Από που προέρχεται αυτή η τάση για κοινωνική δικτύωση;

Από την αρχή, το διαδίκτυο ήταν ένα μέσο για να ενώνει όχι μόνο μηχανήματα αλλά και ανθρώπους. Ηλεκτρονικό ταχυδρομείο, ταχυδρομικές λίστες, το Usenet και πίνακες ανακοινώσεων επέτρεπαν στους ανθρώπους να διασυνδεθούν και να δημιουργήσουν διαδικτυακά κοινωνικά δίκτυα ως επί το πλείστον σχετικά με συγκεκριμένες θεματικές. Αν και αυτές οι ομάδες δεν ορίζουν άμεσα τα κοινωνικά δίκτυα , οι τρόποι με τους οποίους οι άνθρωποι δρούσαν και αντιδρούσαν τα όριζαν εμμέσως.

1.4 Βασικές Υπηρεσίες των Κοινωνικών Δικτύων

Τα Online Κοινωνικά Δίκτυα (Σχ. 1.1), όπως το Facebook και το Twitter , λόγω της καθημερινής αύξησης του αριθμού των νέων χρηστών τους, πέρα από ένα απλό κάλεσμα μεταξύ των χρηστών, έχουν αναπτύξει και ενσωματώσει αρκετές νέες υπηρεσίες (4). Ένα από τα χαρακτηριστικά γνωρίσματα που παρέχει το Facebook και το Twitter είναι το micro-blogging, το οποίο αφορά μία λίστα μικρών καταχωρήσεων κειμένου, μέσω της οποίας οι χρήστες μπορούν να γράφουν και να κοινοποιούν μικρά κείμενα με περιορισμένο αριθμό

χαρακτήρων. Ένα άλλο χαρακτηριστικό του Facebook είναι ο «Τοίχος» μηνυμάτων, ο οποίος είναι μοναδικός για κάθε χρήστη. Ουσιαστικά ο «Τοίχος» του Facebook είναι ένα μέρος για τους φίλους να δημοσιεύουν σημειώσεις στα κοινοποιημένα μεταξύ τους προφίλ. Επίσης, ο κάθε χρήστης ενός κοινωνικού δικτύου όπως το Facebook και Twitter μπορεί να φορτώσει αρχεία δεδομένων και αρχεία πολυμεσικού περιεχομένου. Το mini-feed είναι μια άλλη υπηρεσία των προαναφερόμενων κοινωνικών δικτύων που παρακολουθεί και ενημερώνει τους υπόλοιπους φίλους για την κατάσταση του προφίλ του χρήστη. Επιπρόσθετα ενημερώνει άλλες επιπρόσθετες εφαρμογές που χρησιμοποιεί στο προφίλ του ο χρήστης.

1.4.1 Δημιουργία και Επεξεργασία Προφίλ Χρήστη

Για να συνδεθεί ένας χρήστης στα Online Κοινωνικά Δίκτυα θα πρέπει αρχικά να εγγραφεί σε αυτό, δημιουργώντας έτσι το αρχικό του προφίλ. Εφόσον θα έχει αποκτήσει τα στοιχεία πρόσβασης θα μπορεί συνδεθεί στο δίκτυο και να επεξεργαστεί περεταίρω το προφίλ του. Όπως για παράδειγμα αναρτώντας φωτογραφία του, ή προσθέτοντας προσωπικά δεδομένα κ.α.

Όταν ο χρήστης αποσυνδεθεί από το κοινωνικό δίκτυο, τότε το προφίλ του θα είναι ακόμη διαθέσιμο προς αναζήτηση από τους άλλους χρήστες του ίδιου κοινωνικού δίκτυο. Πολλά από τα OSNs όπως το Facebook, MySpace κ.α. παρέχουν την δυνατότητα στον χρήστη, εάν το επιθυμεί να απενεργοποιήσει την επιλογή αναζήτηση του προφίλ του από άλλους χρήστες του κοινωνικού δικτύου. Αυτό όμως δεν σημαίνει πως και τα δεδομένα που έχει αναρτήσει ο χρήστης στον εξυπηρετητή της εταιρίας, που φιλοξένει το Online Κοινωνικό Δίκτυο είναι ασφαλη και δεν μπορούν να παραβιαστούν.

1.4.2 Προσθήκη και Επεξεργασία Μηνυμάτων

Τα περισσότερα από τα Online Κοινωνικά Δίκτυα παρέχουν στους χρήστες την υπηρεσία αποστολής μηνυμάτων στους φίλους τους, προσθέτοντας στον «Τοίχο» του προφίλ τους ένα ασύγχρονο μήνυμα. Έτσι λοιπόν, όταν ο χρήστης είναι συνδεδεμένος στο κοινωνικό δίκτυο μπορεί να δει το μήνυμα που εστάλει στον προσωπικό του «Τοίχο» άμεσα. Σε περίπτωση όμως, που ο χρήστης δεν είναι συνδεδεμένος στο κοινωνικό δίκτυο τότε, το μήνυμα θα το διαβάσει όταν θα συνδεθεί την επόμενη φορά. Η υπηρεσία αυτή μπορεί να λειτουργήσει σωστά μόνο σε κεντρικοποιημένες υπηρεσίες κοινωνικής δικτύωσης όπως το Facebook, Twitter κ.α.

1.4.3 Αναζήτηση και Κάλεσμα Φίλων

Μια από τις βασικές υπηρεσίες που προσφέρουν τα περισσότερα από τα Online Κοινωνικά Δίκτυα είναι η αναζήτηση φίλων (search friends) και κάλεσμα φίλων (invite friends). Η διαδικασία αυτή επιτυγχάνετε όταν ένας χρήστης τοποθετεί στο προκαθορισμένο πλαίσιο αναζήτησης ως λέξη κλειδί το όνομα ή την διεύθυνση ηλεκτρονικού ταχυδρομείου του φίλου του και στην συνέχεια εκτελέσει την υπηρεσία αναζήτησης του κοινωνικού δικτύου. Μετά την εκτέλεση της υπηρεσίας αναζήτησης και εφόσον το σύστημα θα έχει επιστρέψει αποτελέσματα, δηλαδή θα έχει εντοπίσει κάποιους χρήστες, τότε ο αρχικός χρήστης, που εκτέλεσε την υπηρεσία αναζήτησης, μπορεί να επιλέξει και να καλέσει νέους χρήστες του κοινωνικού δικτύου για να γίνουν «φίλοι» επιλέγοντας τον σύνδεσμο «Invite» από τα αποτελέσματα αναζήτησης. Εδώ αξίζει να αναφέρουμε ότι, μόνο όταν χρήστης που του απεστάλει το μήνυμα «φιλίας» το αποδεχτεί, τότε και μόνο τότε, οι δυο χρήστες μπορούν να θεωρούνται ως «φίλοι» στο κοινωνικό δίκτυο και να ανταλλάζουν δεδομένα μεταξύ τους.

1.4.4 Σύνδεση στο Κοινωνικό Δίκτυο από οπουδήποτε

Τα περισσότερα Online Κοινωνικά Δίκτυα φιλοξενούνται σε διαδικτυακούς εξυπηρετητές (web servers) διάφορων εταιριών, όπου είναι online στο διαδίκτυο συνεχώς. Παρέχοντας την δυνατότητα στους χρήστες του κοινωνικού δικτύου να συνδεθούν από οποιοδήποτε σημείο του κόσμου σε αυτό, αρκεί να έχουν πρόσβαση στο διαδίκτυο με κάποιον υπολογιστή ή smartphone χρησιμοποιώντας μια από τις δημοφιλές εφαρμογές φυλλομετρητών.

1.5 Αποκέντρωση Κοινωνικής Δικτύωσης

Σε ένα Κατανεμημένο Κοινωνικό Δίκτυο P2P, οι παραπάνω κλασικές υπηρεσίες που προσφέρονται σχεδόν από όλα τα κεντρικοποιημένα κοινωνικά δίκτυα, μπορεί να μην υποστηρίζονται σε τέτοιο υψηλό βαθμό που προσφέρονται στα Online Κοινωνικά Δίκτυα. Αυτό οφείλετε στο ότι, σε ένα κατανεμημένο κοινωνικό δίκτυο ο κάθε χρήστης θα έχει τον δικό του εξυπηρετητή στον οποίο θα φιλοξένει το προσωπικό του κοινωνικό δίκτυο. Αυτό σημαίνει ότι, όταν ο χρήστης θα είναι συνδεδεμένος σε αυτό το δίκτυο, τότε θα έχουμε μεγαλύτερη επιτυχία στις παραπάνω υπηρεσίες, διαφορετικά θα πρέπει να μεσολαβήσουν και άλλες τεχνολογίες για την επίτευξη της σωστής λειτουργίας τους.

1.6 Σκοπός και στόχοι της διπλωματικής

(na einai 2-3 selides kai na anaferw to impact.px pws mporei na diasfalistei to aporrhto kai giati afto?dwse 2 paradeigmata)

Σκοπός αυτής της διπλωματικής εργασίας είναι να μελετήσουμε και να συγκρίνουμε τις εναλλακτικές αρχιτεκτονικές, πρωτόκολλα επικοινωνίας και οντολογίες για την υλοποίηση κατανεμημένων κοινωνικών δικτύων, τόσο μεταξύ τους όσο και σε σχέση με τις κεντρικοποιημένες αρχιτεκτονικές με απώτερο στόχο την σχεδίαση και ανάπτυξη ενός υποτυπώδους πρότυπου κατανεμημένου δικτύου εφαρμογών κοινωνικής φύσεως και όχι μόνο. Συγκεκριμένα κάναμε μια προσπάθεια να σχεδιάσουμε και να υλοποιήσουμε μια διαδικτυακή πλατφόρμα ανάπτυξης εφαρμογών που θα φιλοξενείται στον προσωπικό εξυπηρετητή του κάθε χρήστη και όχι σε έναν κεντρικό εξυπηρετητή, χρησιμοποιώντας κατάλληλες τοπολογίες και πρωτόκολλα για την ανταλλαγή δεδομένων με κριτήριο την διάδοση τους και το κατά ποσό είναι ανοιχτής φύσεως.

Θεωρούμε πως στην εποχή μας, την εποχή της πληροφορίας έχει δημιουργηθεί η ανάγκη να αναθεωρηθούν τα όρια που επιτρέπουν οι κοινωνίες στη καταπάτηση των προσωπικών τους δεδομένων. Υπάρχουν δυο τρόποι να επιτευχτεί αυτό. Ο πρώτος είναι ο πολίτικος που όπως αναφέρεται και στην εργασία τα κράτη αναθεωρούν τις νομοθεσίες τους περί προσωπικού απορρήτου προς το χειρότερο. Ο δεύτερος και αυτός για τον οποίο εργαζόμαστε σε αυτή τη διπλωματική είναι ο τεχνικός. Μέχρι στιγμής μια φθηνή, εύχρηστη και ασφαλής λύση για να επικοινωνούν οι άνθρωποι χωρίς μεσάζοντες, η οποία επίσης να μπορεί να επεκταθεί και στις άλλες πτυχές του διαδικτύου δεν έχει προταθεί. Σε αυτή ακριβώς τη προβληματική θα προσπαθήσουμε να απευθυνθούμε με μια τεχνική λύση.

Επίσης προσπαθήσαμε να επικεντρωθούμε στα παρακάτω ερωτήματα ερευνητικού ενδιαφέροντος:

- Πως μπορεί να διασφαλιστεί το απόρρητο της επικοινωνίας μεταξύ των χρηστών στο βαθμό που μας επιτρέπει η σημερινή τεχνολογία;
- Πόσες πτυχές της διαδικτυακής καθημερινότητας μπορεί μια αποκεντρωμένη πλατφόρμα να αντικαταστήσει;
- Πως μπορεί να επιτευχθεί απλότητα χρήσης τέτοια ώστε να είναι δυνατή η σύνδεση στο κατανεμημένο δίκτυο και αρχάριων χρηστών

2ο Κεφάλαιο

2.1 Εισαγώγη

Οι υπηρεσίες κοινωνικής δικτύωσης που στοχεύουν στη δημιουργία on-line κοινοτήτων από ανθρώπους με κοινά ενδιαφέροντα και δραστηριότητες, έχουν γίνει ιδιαιτέρως δημοφιλείς στις μέρες μας. Οι υπηρεσίες αυτές λειτουργούν κυρίως στο διαδίκτυο και προσφέρουν πολλαπλούς τρόπους επικοινωνίας και διάδρασης στους εγγεγραμμένους χρήστες τους, όπου συνήθως προϋποθέτουν τη δημιουργία προσωπικών προφίλ των χρηστών.

Οι χρήστες των υπηρεσιών αυτών μπορούν να δημοσιοποιούν και να μοιράζονται προσωπικές πληροφορίες με άλλες ομάδες χρηστών χωρίς κάποιο περιορισμό. Τέτοιες πληροφορίες είναι για παράδειγμα θέματα σχετικά με τα χόμπι τους, την εργασία τους, τις προτιμήσεις τους, τα αγαπημένα τους πρόσωπα, κ.α. μέσα από το προσωπικό τους προφίλ, αλλά και υπό μορφή μηνυμάτων, φωτογραφιών, βίντεο, κ.α.

Αναμφίβολα οι υπηρεσίες κοινωνικής δικτύωσης αποτελούν μία νέα μορφή επικοινωνίας, ιδιαιτέρως ανάμεσα στους νέους αλλά όχι μόνο. Ταυτόχρονα όμως, οι υπηρεσίες αυτές προσδίδουν και μια καινούργια διάσταση στην έννοια του "προσωπικού χώρου", δημιουργώντας σοβαρές ανησυχίες για παραβίαση της ιδιωτικότητας των χρηστών τους, των οποίων τα προσωπικά δεδομένα δημοσιοποιούνται στο διαδίκτυο με πρωτοφανή τρόπο και ποσότητα.

Επίσης, ακόμη και αν κάποιοι χρήστες επιλέξουν να μην δημοσιοποιήσουν τα προσωπικά τους δεδομένα, μπορεί κάποιοι φίλοι τους να τα δημοσιοποιήσουν, όπως για παράδειγμα φωτογραφίες. Η παραβίαση του προσωπικού απορρήτου γίνεται και όταν οι ίδιοι οι χρήστες δεν σέβονται την ιδιωτικότητα των άλλων δημοσιεύοντας προσωπικά δεδομένα τρίτων χωρίς την συγκατάθεση τους (5).

2.2 Τα κοινωνικά δίκτυα σε αριθμούς

Τα κοινωνικά δίκτυα πλέον έχουν φτάσει στους υπολογιστές του 82 τοις εκατό του παγκοσμίου συνδεδεμένου πληθυσμού που απαριθμεί 1.2 δισεκατομμύρια χρήστες ανά το κόσμο. Η υιοθέτηση των κοινωνικών δικτύων πλέον αντιστοιχεί αναλογικά στην υιοθέτηση γενικότερα του ιντερνέτ δείχνοντας ότι με το που οι νέοι χρήστες συνδέονται στο διαδίκτυο αμέσως συνδέονται και μεταξύ τους μέσω κοινωνικών δικτύων.

Ένα ακόμη σημαντικό στοιχειό που δείχνει τη διείσδυση των κοινωνικών δικτύων στις ζωές εκατομμυρίων ανθρώπων είναι ο χρόνος που αφιερώνουν οι χρήστες σε αυτά. Ποσοστιαία σε σχέση με το χρόνο που οι χρήστες αφιερώνουν συνολικά στο διαδίκτυο , τα κοινωνικά δίκτυα έχουν τριπλασιάσει τα ποσοστά τους τα τελευταία χρόνια. Πιο συγκεκριμένα τον Οκτώβριο του 2011 τα κοινωνικά δίκτυα ήταν το πιο δημοφιλές περιεχόμενο σε παγκόσμιο επίπεδο αποτελώντας το 19% του συνολικού χρόνου χρήσης του διαδικτύου. Το 2007 το ποσοστό αυτό αντιστοιχούσε στο 6 τοις εκατό.

Ο χρόνος χρήσης των κοινωνικών δικτύων έχει κερδίσει έδαφος τα τελευταία αυτά χρόνια "κλέβοντας" χρόνο από παραδοσιακά διαδικτυακά μέσα επικοινωνίας όπως το ηλεκτρονικό ταχυδρομείο και από προγράμματα ανταλλαγής άμεσων μηνυμάτων (chat) , αναδεικνύοντας τη σημαντικότητα τους ως ένα ακόμα κεντρικό κανάλι επικοινωνίας για τους χρήστες.

Γενικότερα η κοινωνική δικτύωση έχει εξελιχτεί τα τελευταία χρόνια σε ένα οργανικό κομμάτι της διαδικτυακής εμπειρίας το οποίο με πολλούς τρόπους αντικατοπτρίζει και προσομοιώνει τις διαδικασίες κοινωνικοποίησης σε πραγματικό επίπεδο "μακριά από το πληκτρολόγιο "(A.F.K) (σχ. 2.1).

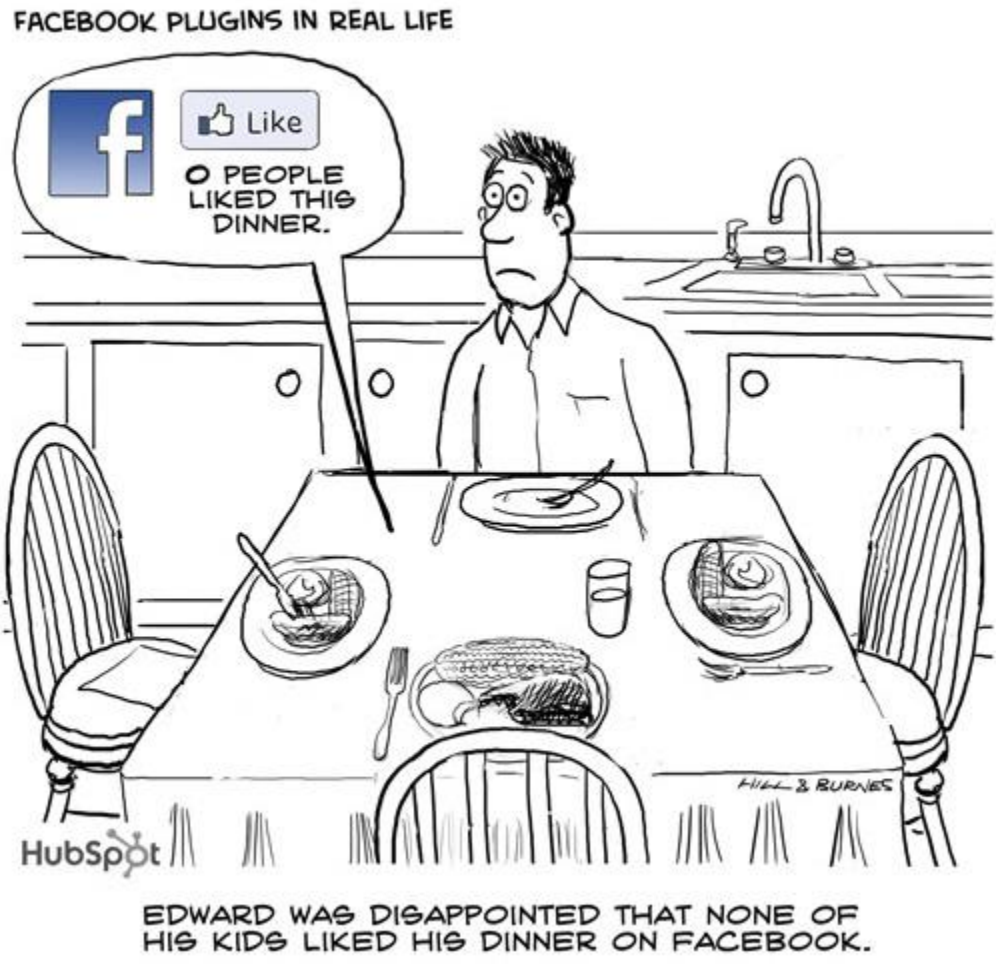


Figure 2.1 Η διαδικτυακη κοινωνικη δικτυωση επηρεαζει τη ζωη μεγαλου ευρους ηλικιων

Η ανάπτυξη της κοινωνικής δικτύωσης είναι ένα παγκόσμιο πολιτισμικό φαινόμενο. Παρά τις πολύ σημαντικές διαφορές στις κυβερνήσεις , τις τηλεπικοινωνιακές υποδομές, τη διαθεσιμότητα της διαδικτυακής πρόσβασης και τις πολιτισμικές πρακτικές ανά τον κόσμο , η κοινωνική δικτύωση αναπτύσσεται σε όλες τις χώρες.

Με μια πιο λεπτομερή ανάλυση των δεδομένων από τη Comscore (6) βλέπουμε ότι η διείσδυση στις οικονομίες των χώρων κυμαίνεται από 53 τοις εκατό στη Κίνα μέχρι και 98 τοις εκατό στις Ηνωμένες Πολιτείες Αμερικής, με 41 από τις 43 οικονομίες που ασχολείται η έρευνα να έχουν πάνω από 83 τοις εκατό διείσδυση.

Ανεξαρτήτως ποσό ανοιχτή ή ποσό κλειστή είναι μια κοινωνία μπορούμε με ασφάλεια να υποθέσουμε ότι πάνω από το μισό συνδεδεμένο πληθυσμό της χρησιμοποιεί μέσα κοινωνικής δικτύωσης , κάνοντας της πρακτική της κοινωνικής δικτύωσης πανταχού παρούσα.

Μια άλλη παρατήρηση που δείχνει τη διαπολιτισμική φύση της συνδεδεμένης κοινωνικής δικτύωσης, έγκειται στο γεγονός ότι στις περιοχές της Λατινικής Αμερικής ,της Ευρώπης , και της Κεντρικής Ανατολικής Αφρικής -τρεις πολύ διαφορετικές πολιτισμικά περιοχές - ο χρόνος που αφιερώνουν οι χρήστες στα κοινωνικά δίκτυα αποτελεί κατ' ελάχιστο το 24% του συνολικού χρόνου ασχολίας με το διαδίκτυο.

Για να κατανοηθεί καλύτερα η κατάσταση της κοινωνικής δικτύωσης σήμερα κάποιος πρέπει να δώσει ιδιαίτερη σημασία στην εταιρία κοινωνικής δικτύωσης Facebook- τη μεγαλύτερη εταιρία παγκοσμίως στο τομέα κοινωνικής δικτύωσης- που επηρεάζει σημαντικά τη γενικότερη κατεύθυνση των κοινωνικών δικτύων. Αυτή τη στιγμή το Facebook είναι η τρίτη μεγαλύτερη σε αξία εταιρία σχετική με το διαδίκτυο και η ιστοσελίδα της απασχολεί το 1/7 του χρόνου που αφιερώνουν η χρήστες στο διαδίκτυο.

2.3 Φιλότοποι (Facebook) και προσωπικά δεδομένα

Το Facebook δεδομένου ότι είναι μια καπιταλιστική επιχείρηση ,έχει ως στόχο το χρηματικό κέρδος. Αυτό το καταφέρνει μέσω εξατομικευμένων στοχευόμενων διαφημίσεων ,κάτι που σημαίνει ότι οι διαφημίσεις προσαρμόζονται στα προσωπικά καταναλωτικά ενδιαφέροντα των χρηστών. Οι υπηρεσίες κοινωνικής δικτύωσης είναι ιδιαίτερα πρόσφορο έδαφος για εξατομικευμένες διαφημίσεις από τη στιγμή που έχουν πρόσβαση σε ένα τεράστιο όγκο δεδομένων που αποτελούνται από προσωπικές επιθυμίες και γενικότερα ενδιαφέροντα των χρηστών ,κάτι που επιτρέπει τη παρακολούθηση αυτών των δεδομένων με σκοπό το κέρδος βρίσκοντας τα προϊόντα που οι χτίστες είναι πιο πιθανό να αγοράσουν. Έτσι εξηγείτε και ότι η μέθοδος των εξατομικευμένων διαφημίσεων είναι η κύρια πηγή εισοδήματος των κερδοσκοπικών εταιριών κοινωνικής δικτύωσης. Το Facebook πιο συγκεκριμένα παρακολουθεί μαζικά τους χρήστες του αποθηκεύοντας, συγκρίνοντας ,αξιολογώντας και πουλώντας τα προσωπικά δεδομένα και τη συμπεριφορά εκατοντάδων εκατομμυρίων χρηστών. Αυτή η μαζική παρακολούθηση των χρηστών είναι εξατομικευμένη και συγκεκριμένη για κάθε χρήστη εξαιτίας της λεπτομερής ανάλυσης των ενδιαφερόντων του χρήστη και έχοντας πρόσβαση στη μεγάλη πλειοψηφία των ιστοσελίδων που επισκέπτεται ο χρήστης (μέσω των κουμπιών 'Like!') που επιτρέπει την ένταξη του χρήστη σε κάποια καταναλωτική κατηγορία και επομένως τη προώθηση διαφημίσεων που θεωρείται βάσει αλγορίθμων ότι θα επηρεάσουν περισσότερο το χρήστη. Ουσιαστικά αυτό που έχει επικρατήσει λοιπόν είναι ότι οι χρήστες επιτρέπουν εν αγνοία τους σε μια κερδοσκοπική επιχείρηση να έχει πρόσβαση στη πλειοψηφία της διαδικτυακής τους εμπειρίας ασχέτως αν βρίσκονται στην ιστοσελίδα της ή όχι ,αποκαλύπτοντας, με αυτό τον τρόπο, ευαίσθητα προσωπικά δεδομένα με σκοπό οι χρήστες να καταναλώνουν περισσότερο.

2.4 Όροι χρήσης υπηρεσιών κοινωνικής δικτύωσης

Η χρήση εξατομικευμένης διαφήμισης και παρακολούθησης για οικονομικό κέρδος είναι εξασφαλισμένη νομικά μέσω των όρων προσωπικών δεδομένων που ο χρήστης υποχρεούται να αποδεχτεί για τη χρήση της υπηρεσίας αυτής. Σε αυτό τον τομέα θα κάνω μια ανάλυση στους όρους χρήσης σχετικά με τα προσωπικά δεδομένα και τη πολιτική διαφημίσεων. Επίσης θα δούμε τη σημασία έχουν τα προσωπικά αυτά δεδομένα στα πλαίσια της πολιτικής εξουσίας και του καπιταλισμού. Κατανοώντας το ρόλο που έχει η διαφήμιση στους όρους χρήσης του Facebook κάποιος μπορεί να καταλάβει πως οι αυτορυθμιζόμενοι νομικοί μηχανισμοί μπορούν να καταλήξουν σε ισχυρή οικονομική εξουσία και κυριαρχία κάποιων εταιριών εις βάρος μεγάλου της πλειοψηφίας του πληθυσμού.

Το ότι το Facebook είναι ένας αυτορυθμιζόμενος νομικός μηχανισμός οφείλεται κυρίως στο ότι η έδρα του σαν εταιρία είναι στο Πάλο Άλτο των ΗΠΑ. Οι όροι χρήσης του Facebook είναι ένα τυπικό παράδειγμα αυτορυθμιζόμενου καθεστώτος για τα προσωπικά δεδομένα , που επιτρέπεται να ορίζει η ίδια η εταιρία το πως θα χρησιμοποιεί και θα επεξεργάζεται τα προσωπικά δεδομένα των χρηστών της. Γενικότερα έρευνες δείχνουν ότι το νομικό σύστημα των ΗΠΑ παρέχει ελάχιστη προστασία στα προσωπικά δεδομένα των πολιτών της και σε σχέση με την Ευρωπαϊκή Ένωση παραμένει πολύ πίσω (7). Οι νόμοι στις ΗΠΑ πιο συγκεκριμένα προστατεύουν μόνο τα προσωπικά δεδομένα που βρίσκονται σε κυβερνητικές βάσεις δεδομένων και όσον αφορά τις επιχειρήσεις και την εμπορική χρήση των προσωπικών δεδομένων οι νόμοι είναι φτιαγμένοι με σκοπό να μεγιστοποιούν τα κέρδη των επιχειρήσεων (8). Στις Ηνωμένες Πολιτείες Αμερικής η κυβέρνηση κάνει απλά παρεμβάσεις στους όρους λειτουργίας των επιχειρήσεων ,ενώ η ελεύθερη αγορά αποτελεί το νομοθετικό σώμα όσον αφορά τη προστασία των προσωπικών δεδομένων. Η ελεύθερη αγορά όμως δε διασφαλίζει τη διαφορετικότητα και τη διασφάλιση της ανεξάρτητης προσωπικής ανάπτυξης του ατόμου, αντιθέτως πολλές φορές αποτελεί τροχοπέδη στην ελευθερία (9).

Συγκεκριμένα στις "Αρχές του Facebook" αναφέρεται: "Κάθε άνθρωπος θεωρείται κάτοχος των δεδομένων του. Πρέπει, λοιπόν, να είναι ελεύθερος να τα μοιράζεται με όποιον θέλει και να τα μεταφέρει όπου θέλει, ακόμη κι αν αυτό σημαίνει ότι θα τα αφαιρέσει από την υπηρεσία Facebook. Κάθε άνθρωπος πρέπει να είναι ελεύθερος να αποφασίζει με ποιον θα μοιράζεται τα δεδομένα του και να ρυθμίζει το απόρρητό του ώστε να προστατεύει αυτές τις επιλογές. Ωστόσο, αυτές οι ρυθμίσεις δεν μπορούν να περιορίσουν το πώς θα χρησιμοποιούν τα δεδομένα σας όσοι τα λαμβάνουν, ειδικά εκτός της υπηρεσίας Facebook." , όπου γίνεται ξεκάθαρο ότι ο χρήστης έχει δικαίωμα να επιλέξει σε ποιους από τους υπόλοιπους χρήστες θα κοινοποιήσει τα προσωπικά του δεδομένα αλλά δεν του επιτρέπεται να συμμετέχει στην απόφαση του ποιες εταιρίες θα έχουν πρόσβαση στα δεδομένα του. Δεν υπάρχουν κάπου ρυθμίσεις απορρήτου που να επιτρέπουν στους χρήστες να απενεργοποιούν τη πρόσβαση των διαφημιστών στα προσωπικά τους δεδομένα. Τα νομικά κείμενα του Facebook επίσης ορίζουν ότι η στοχευόμενη διαφήμιση είναι "απαραίτητη" και "αποδεκτή από τους χρήστες". Το πρόβλημα όμως είναι ότι οι χρήστες δεν ρωτούνται αν θεωρούν τη στοχευόμενη διαφήμιση απαραίτητη και αν συμφωνούν σε αυτή τη μέθοδο διαφήμισης. Το Facebook επίσης λέει ότι "δεν μοιράζεται τις προσωπικές πληροφορίες με διαφημιστές " αλλά η στοχευόμενη διαφήμιση είναι πάντα ενεργοποιημένη και δε δίνεται η δυνατότητα στους χρήστες να την απαλλαχτούν από αυτή. Οι χρήστες πρέπει να συμφωνήσουν με τους όρους

απορρήτου του Facebook για να μπορέσουν να χρησιμοποιήσουν τις υπηρεσίες του. Δεδομένου όμως ότι το Facebook είναι το μεγαλύτερο διαδικτυακό κοινωνικό δίκτυο στο κόσμο και η δεύτερη πιο πολυσύχναστη ιστοσελίδα , είναι σχετικά απίθανο οι νέοι χρήστες να αρνηθούν τους όρους απορρήτου, διαφορετικά θα αποκλειστούν από κοινωνικές διαδικασίες που τους κρατάνε σε επαφή με φίλους και συναδέλφους, όπως επίσης και δυνατότητες να αποκτήσουν νέες επαφές και να συμμετέχουν σε διαφορές κοινότητες.

Όποτε το Facebook αναγκάζει τους χρήστες του για να χρησιμοποιήσουν την πλατφόρμα του να αποδέχονται να χρησιμοποιούνται τα προσωπικά τους δεδομένα με σκοπό το κέρδος. Κάνει την υπόθεση ότι οι χρήστες θέλουν να θυσιάσουν το προσωπικό τους απόρρητο για να μπορούν να χρησιμοποιούν τη πλατφόρμα του (σχ. 2.2). Επίσης θεωρεί ότι για να "βελτιώνεται και να προωθείται η υπηρεσία" το κοινωνικό δίκτυο δε μπορεί αλλιώς παρά να στηθεί με πυλώνα τη διαφήμιση. Τα πιθανά προβλήματα από αυτού του είδους τη διαφήμιση για τους χρήστες δεν αναφέρονται πουθενά.

Figure 2.2 Οροι χρησης Facebook

Εκτός από τα δεδομένα που λαμβάνουν οι διαφημιστικές εταιρίες από το Facebook τους δίνεται η δυνατότητα να χρησιμοποιούν τα λεγόμενα cookies που είναι μικρά προγράμματα που τρέχουν στο περιηγητή του χρήστη και παρακολουθούν τη δραστηριότητα του. Το Facebook δίνει τη δυνατότητα απενεργοποίησης στους χρήστες των cookies έχοντας όμως αυτή τη ρύθμιση σχετικά κρυμμένη κάτι που στη πράξη σημαίνει ότι η πλειοψηφία των χρηστών δεν την μαθαίνουν ποτέ και επακόλουθος δε την απενεργοποιούν. Αυτό αντικατοπτρίζει την επικρατούσα λογική ότι το Facebook δίνει μεγαλύτερη αξία στο χρηματικό κέρδος από ότι στη προστασία του προσωπικού απορρήτου των χρηστών της, που εξηγεί την προσπάθεια η απενεργοποίηση μέρους της εξατομικευμένης διαφήμισης να είναι όσο το δυνατών πιο πολύπλοκη.

Η παρακολούθηση στο Facebook είναι παρακολούθηση καταναλωτών-παραγωγών , οι οποίοι συνεχώς δημιουργούν και μοιράζονται περιεχόμενο φτιαγμένο από τους ίδιους, περιηγούνται σε προφίλ άλλων χρηστών και σε διάφορα δεδομένα, αλληλεπιδρούν με άλλους χρήστες, συμμετέχουν και δημιουργούν κοινότητες και παράγουν πληροφορίες με συνεργατικό τρόπο. Οι διαφημιστικές και όχι μόνο επιχειρήσεις και οι συνεργάτες τους συνεχώς παρακολουθούν και καταγράφουν προσωπικά δεδομένα και συνήθειες των χρηστών και ύστερα αποθηκεύουν , συγχωνεύουν και αναλύουν τα δεδομένα που έχουν συλλέξει. Καταυτό το τρόπο δημιουργούν ένα πολύ λεπτομερές προφίλ της προσωπικότητας του χρήστη. Το Facebook

λοιπόν παρέχει σαν προϊόν τους χρήστες του στις διαφημιστικές επιχειρήσεις. Ανταλλάσσονται χρήματα για πρόσβαση στα προσωπικά δεδομένα του χρήστη που επιτρέπει την οικονομική επιτήρηση του χρήστη. Η επιτήρηση αυτή επιτρέπει στις επιχειρήσεις να δελεάσουν τους χρήστες στο να καταναλώσουν, κατευθύνουν τις ανάγκες τους και τα "θέλω" τους με γνώμονα τα συμφέροντα των επιχειρήσεων που αντιπροσωπεύουν. Σύμφωνα με την ανάλυση του Andrejevic (10): "Η στρατηγική των reality TV , του διαδικτύου και των διαδραστικών πολυμέσων για την αποκόμιση κέρδους δε βασίζεται πλέον τόσο στο να παρακολουθεί κάποιος κάτι, αλλά στο να τον παρακολουθούν. "

2.5 Αναγκαιότητα μιας εναλλακτικής ανεξάρτητης κοινωνικής δικτύωσης

Η παραπάνω ανάλυση δείχνει ότι η εμπιστοσύνη που έχουν οι χρήστες στις μεγάλες εταιρίες όσον αφορά το προσωπικό τους απόρρητο δεν αποδίδει. Οι εταιρίες χρησιμοποιούν αυθαίρετα τα προσωπικά δεδομένα των χρηστών με κερδοσκοπικά κριτήρια κατευθύνοντας τους ανθρώπους σε μια εποχή που όλοι συμμετέχουν στο "μεγάλο αδελφό" οικειοθελώς με μόνο αντάλλαγμα το να μπορούν να επικοινωνούν μεταξύ τους. Οι χρήστες με τον καιρό θεωρούν το προσωπικό τους απόρρητο όλο και λιγότερο σημαντικό , και η εξατομικευμένη παρακολούθηση ατόμων είναι εξέχουσα και άκρως κερδοφόρα επιχειρηματική δραστηριότητα.

Αυτό βεβαία δεν αφορά μόνο το Facebook αλλά και άλλες μεγάλες επιχειρήσεις που έχουν πρόσβαση σε προσωπικές πληροφορίες και συνήθειες. Η Google για παράδειγμα αποκτά πρόσβαση σε όλο και περισσότερους τομείς της ζωής των χρηστών της. Γνωρίζει τις αναζητήσεις των χρηστών μέσω της μηχανής αναζήτησης, γνωρίζει τις επικοινωνίες των χρηστών της μέσω της υπηρεσίας Gmail, γνωρίζει τη τοποθεσία των χρηστών της μέσω του πιο διαδεδομένου λειτουργικού για κινητές συσκευές Android, και πολλά ακόμη. Τα προσωπικά δεδομένα δε προστατεύονται από τα κράτη, αλλά αντιθέτως θεωρούνται ένα πολύ κερδοφόρο εμπόρευμα.

Το πραγματικό όμως πρόβλημα και αυτό που έδωσε τη δυνατότητα ύπαρξης σε αυτές τις επιχειρηματικές δραστηριότητες δεν είναι άλλο από την μέχρι σήμερα αρχιτεκτονική του διαδικτύου. Η συγκεντρωτική αρχιτεκτονική του διαδικτύου με εξυπηρετητές και πελάτες όπου οι πελάτες στέλνουν αιτήματα στους εξυπηρετητές και αυτοί τους απατάνε είναι αυτό που οδήγησε στη λογική ότι για ένα επικοινωνήσει ένας χρήστης με κάποιον άλλο χρήστη είναι αναγκασμένος να συνδεθεί με κάποιον εξυπηρετητή. Αυτό είχε ως αποτέλεσμα ο εξυπηρετητής στον οποίο συνδέονται οι περισσότεροι πελάτες να δημιουργεί ένα μονοπώλιο επιτρέποντας στην εταιρία που τον διαχειρίζεται να τον λειτούργει υπό τους δικούς της κανόνες χωρίς να λαμβάνει υποψία της επιθυμίες και τα συμφέροντα του χρήστη.

Η αρχική λειτουργιά του διαδικτύου ήταν καθαρά αποκεντρωμένη με πλατφόρμες όπως το Usenet , IRC , Finger που επέτρεπαν στους χρήστες να δημοσιεύσουν μικρά κείμενα, να μοιραστούν φωτογραφίες και εικόνες , να στείλουν μηνύματα και αλλά. Ο λόγος που αυτά απέτυχαν ήταν η εμπορικοποίηση του διαδικτύου. Τα μόνα επιχειρηματικά μοντέλα που επέτρεπαν στους επενδυτές να αποκομίσουν κέρδη ήταν τα συγκεντρωτικά μοντέλα, όπου οι κεντρικοί εξυπηρετητές αναλαμβάνουν την εξυπηρέτηση όλων των πελατών τους. Καταυτό

τον τρόπο η χρηματοδότηση για την ανάπτυξη αποκεντρωμένων τεχνολογιών εγκαταλείφθηκε και επικράτησε η ανάπτυξη πλατφορμών που μπορούσαν να κερδοσκοπήσουν οι καπιταλιστές με το επιχειρηματικό μοντέλο της παρακολούθησης των χρηστών και της επιρροής στη συμπεριφορά τους. Το καπιταλιστικό διαδίκτυο φαινομενικά κέρδισε το ανεξάρτητο αυτόνομο και αποκεντρωμένο διαδίκτυο.

Για την αποφυγή των παραπάνω προβλημάτων, την λύση έρχονται να δώσουν τα κατανεμημένα κοινωνικά δίκτυα, όπου η αποθήκευση των δεδομένων γίνεται τοπικά στον εξυπηρετητή του κάθε χρήστη. Έτσι τα δεδομένα αυτά μπορούν να προσπελαστούν από τρίτους μόνο όταν ο χρήστης τους, δηλαδή ο χρήστης που έχει την κυριότητα των δεδομένων, το εγκρίνει.

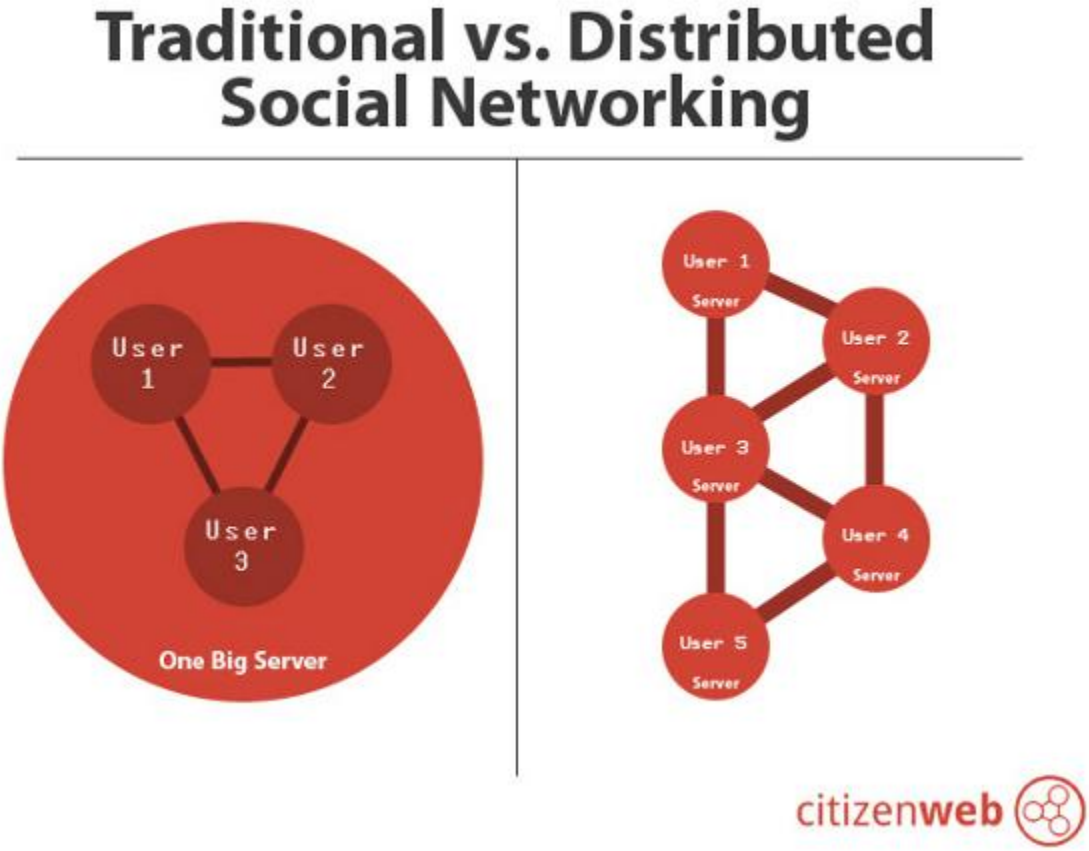


Figure 2.3 Συγκριση κεντρικοποιημενων με κατανεμημενα δικτυα

Κατά τη τελευταία δεκαετία έχουν γίνει πολλές προσπάθειες για ένα αποκεντρωμένο κατανεμημένο ανώνυμο δίκτυο που θα δουλεύει παράλληλα με το υπάρχων διαδίκτυο , από τις οποίες όμως καμία δε κατάφερε να ανταγωνιστεί εμπορικά δίκτυα εταιριών, όπως το Facebook , τη Google κλπ. Οι κυριότερες προσπάθειες βασίζονται σε διάφορα μοντέλα δικτύων υπολογιστών peer-to-peer. Όπως περιγράφει το wikipedia: "Το peer-to-peer είναι ένα δίκτυο που επιτρέπει σε δυο ή περισσότερους υπολογιστές να μοιράζονται τους πόρους τους ισοδύναμα. Όλοι οι κομβόι του δικτύου έχουν ίσα δικαιώματα. Πληροφορίες που βρίσκονται στον ένα κόμβο , ανάλογα με τα δικαιώματα που καθορίζονται, μπορούν να διαβαστούν από όλους τους άλλους και αντίστροφα." κάτι που σημαίνει ότι η ανάγκη για την ύπαρξη μιας μεγάλης εταιρίας που θα φυλάει τα προσωπικά δεδομένα και θα έχει το ρόλο του νομοθέτη στο δίκτυο επικοινωνίας των χρηστών δεν είναι αναγκαία. Οι χρήστες μπορούν να χρησιμοποιούν τις δίκες τους υποδομές (τους προσωπικούς τους υπολογιστές) για τη στέγαση

του δικτύου επικοινωνίας μεταξύ τους και να θέσουν δικούς τους κανόνες για το τρόπο λειτουργίας του δικτύου αυτού. Επίσης, ο χώρος αποθήκευσης των προσωπικών τους δεδομένων μπορεί να είναι επιλογή του χρήστη και η πρόσβαση σε αυτά να είναι δυνατή μόνο μετά από άδεια του χρήστη.

2.6 Επίλογος

Η καταχρηση της εμπιστοσυνης που δειχνουμε στις εταιριες για τα προσωπικα μας δεδομενα ειναι πλεον μη ανεκτη. Η εμπορικοποιηση των προσωπικων μας δεδομενων για χρηματικα οφελη, η ανικανοτητα των κυβερνησεων να επιβαλλουν καποια προστασια στους πολιτες τους και η διαλυση καθε εννοιας ιδιωτικοτητας μας οδηγουν αναγκαστικα σε μια νεα μορφη του διαδικτυου: το Ομοτιμο διαδικτυο.

3ο Κεφάλαιο

3.1 Πρότυπες P2P Web Εφαρμογές Κοινωνικών Δικτύων

Στο τρέχων κεφάλαιο γίνεται μια ανασκόπηση γύρω από τις web εφαρμογές κατανεμημένων κοινωνικών δικτύων που έχουν προταθεί μέχρι σήμερα. Εδώ εστιάζουμε τόσο στην τεχνολογίες που χρησιμοποιούν αυτές οι εφαρμογές, καθώς και αν πληρούν όλες τις δυνατότητες που προσφέρει σήμερα ένα online κοινωνικό δίκτυο OSN.

3.1.1 Το Κατανεμημένο Κοινωνικό Δίκτυο Diaspora

(sxhmata)

Το κοινωνικό δίκτυο Diaspora (σχ.3.1) είναι ελεύθερο και ανοιχτό λογισμικό κατανεμημένης κοινωνικής δικτύωσης που δημιουργήθηκε από τους Dan Grippi, Maxwell Salzberg, Raphael Sofaer και Ilya Zhitomirskiy (11).

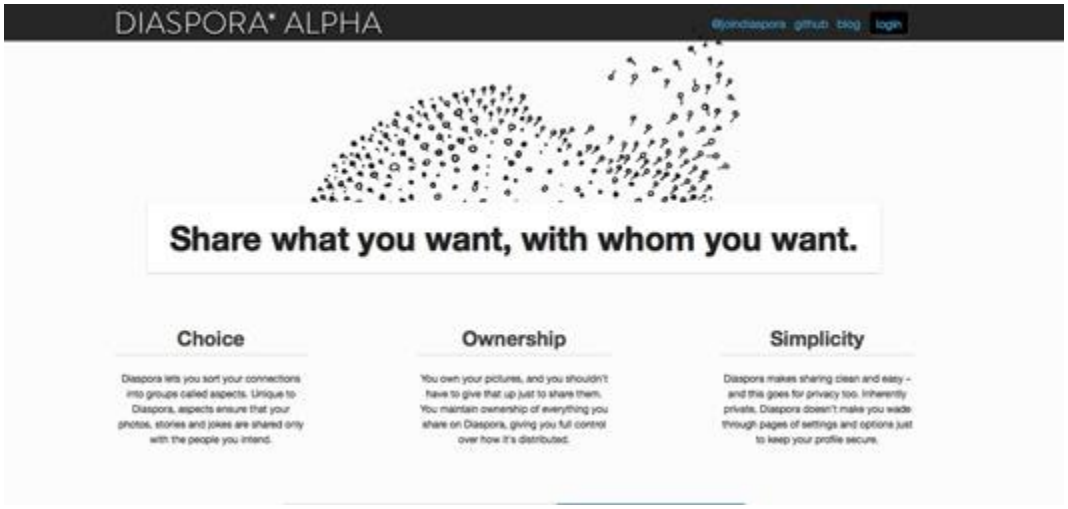


Figure 3.1 Κοινωνικό δίκτυο Diaspora

Το κοινωνικό δίκτυο Diaspora (12) έχει αποκεντρωμένη δομή και αποτελείται από δεκάδες υπολογιστές - διαδικτυακοί εξυπηρετητές που ονομάζονται pods (Σχ. 3.2). Μέσω αυτών των εξυπηρετητών παρέχονται διάφορες διευθύνσεις στο διαδίκτυο όπου ο κάθε χρήστης μπορεί να προβεί σε εγγραφή και στην συνέχεια να συνδεθεί στο κατανεμημένο κοινωνικό δίκτυο Diaspora. Μέχρι σήμερα έχουν προταθεί δυο τρόποι για να εντοπίσουμε τα pods του Diaspora για να συνδεθούμε:

- Ο πρώτος τρόπος είναι με την αποστολή πρόσκλησης μέσω email, όπου μέσω της πρόσκλησης ο χρήστης θα παραπέμπετε σε συγκεκριμένο URL του pod.
- Ο δεύτερος τρόπος είναι να κάνουμε μια αναζήτηση σε έναν κατάλογο των pods όπως στην διεύθυνση www.podupti.me. Μέσω αυτό του τρόπου, μετά από μια σύντομη αξιολόγηση των pods, μπορούμε να επιλέξουμε ένα από τα pods που

θεωρούμε πως είναι ασφαλής και να δημιουργήσουμε έναν λογαριασμό σε αυτό το pod.

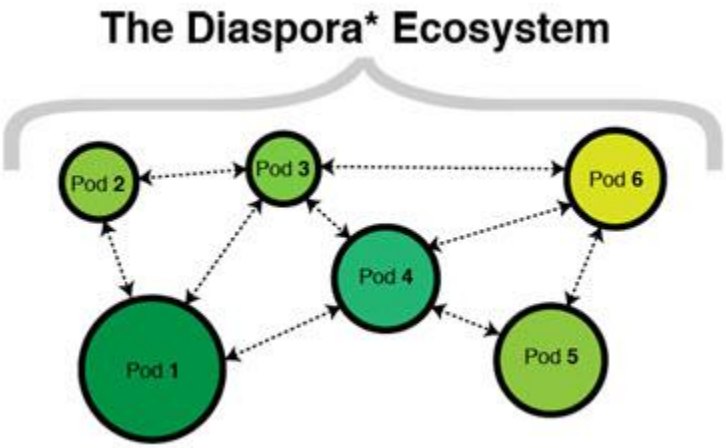


Figure 3.2 Διαδικτυακοί εξυπηρετητές που ονομάζονται pods

Κάθε pod ανάλογα με τη διαχείριση του μπορεί να είναι τόσο αξιόπιστο ή γρήγορο όσο ένα άλλο. Συνήθως αν δεν υπάρχουν προβλήματα μπορούμε να δούμε τα μέλη άλλων pods και να συνδεθούμε μαζί τους, ανταλλάζοντας μηνύματα και πολυμεσικό περιεχόμενο. Το κατανεμημένο κοινωνικό δίκτυο Diaspora πέρα από τις κλασικές υπηρεσίες που προσφέρουν τα κοινωνικά δίκτυα υποστηρίζει επιπλέον τις ακόλουθες υπηρεσίες:

- **Τα Aspects.** Aspects καλούμε την οργάνωση των επαφών μας στο Diaspora (13). Ένα aspect, είναι μια ομάδα επαφών. Μέσω των aspects μπορούμε να οργανώσουμε καλυτέρα τις επαφές μας σε ομάδες με βάση τα κριτήρια που θα ορίσουμε εμείς. Επίσης, εδώ αξίζει να σημειώσουμε ότι οι επαφές (εν προκειμένου οι φίλοι) που έχουν οργανωθεί σε ομάδες δεν γνωρίζουν την ομάδα που μπορούν να ανήκουν σε έναν χρήστη.
- **Θεματικές Ετικέτες (Hash Tags).** Όταν δημιουργούμε ένα μήνυμα μπορούμε να προσθέσουμε λέξεις που αρχίζουν με το σύμβολο #(hash). Μέσω αυτών των ειδικών λέξεων ο εξυπηρετητής ταξινομεί το μήνυμα μας με τέτοιον τρόπο έτσι ώστε να μπορεί να το εντοπίσει κάποιος άλλος φίλος με τα αντίστοιχα ενδιαφέροντα.
- **Ειδική κοινοποίηση.** Στο Diaspora μπορούμε να κοινοποιήσουμε μηνύματα ή φωτογραφίες σε μια ομάδα των επαφών μας (σε μια ομάδα φίλων) ή σε όλους τους φίλους μας ανεξαρτήτως αν ανήκουν σε κάποια ομάδα.
- **Δημοσίευση (Post).** Στο Diaspora η δημοσίευση ενός κειμένου από τον χρήστη μπορεί να αναδημοσιευτεί, κοινοποιηθεί, αναφερθεί και χαρακτηριστεί από άλλο χρήστη, καθώς και από τον δημιουργό του αρχικού δημοσιεύματος. Επιπρόσθετα, ο δημιουργός μιας δημοσίευσης μπορεί και να την αποκρύψει από τους φίλους του. Ένας άλλο ειδικός χαρακτηρισμός των δημοσιεύσεων είναι το φιλτράρισμα τους μέσω της ετικέτας «Not Safe For Work» ορίζοντας την ετικέτα #nsfw σε μια δημοσίευση, ενημερώνοντας έτσι τους άλλους φίλους για την ακαταλληλότητα της δημοσίευσης.
- **Υπηρεσίες διασύνδεσης.** Το Diaspora παρέχει επίσης υπηρεσίες διασύνδεσης με άλλα κοινωνικά δίκτυα, ανακτώντας πληροφορίες για το προφίλ του χρήστη.

- **Προστασίας Προσωπικών Δεδομένων από τον σχεδιασμό (Privacy by design).** Ένα άλλο σημαντικό στοιχείο του κοινωνικού δικτύου Diaspora που αξίζει να αναφέρουμε είναι ότι δεν εκθέτει πληροφορίες του χρήστη σε διαφημιστικές εταιρίες, ή σε ιστοσελίδες που επισκέπτεται ο χρήστης.
- **Κυριότητα των δεδομένων.** Ο κάθε χρήστης που δημιουργεί το δικό του προφίλ στο Diaspora διατηρήσει την πλήρη κυριότητα όλων των πληροφοριών που αναρτά στο προφίλ του συμπεριλαμβανομένων καταλόγων φίλων, μηνυμάτων, φωτογραφιών και λεπτομερειών του προφίλ. Όταν ο χρήστης διαγράφει τον λογαριασμό του τότε όλα τα δεδομένα του προφίλ του διαγράφονται.
- **Μελλοντικές Υπηρεσίες.** Μια από τις βασικότερες μελλοντικές υπηρεσίες που προσβλέπουν να υποστηρίζει το Diaspora είναι η μεταφορά του προφίλ του χρήστη μεταξύ των pods.

Παρακάτω παρουσιάζεται και μια συνολική διεπαφή του frontend του Diaspora (Εικ. 3.3).

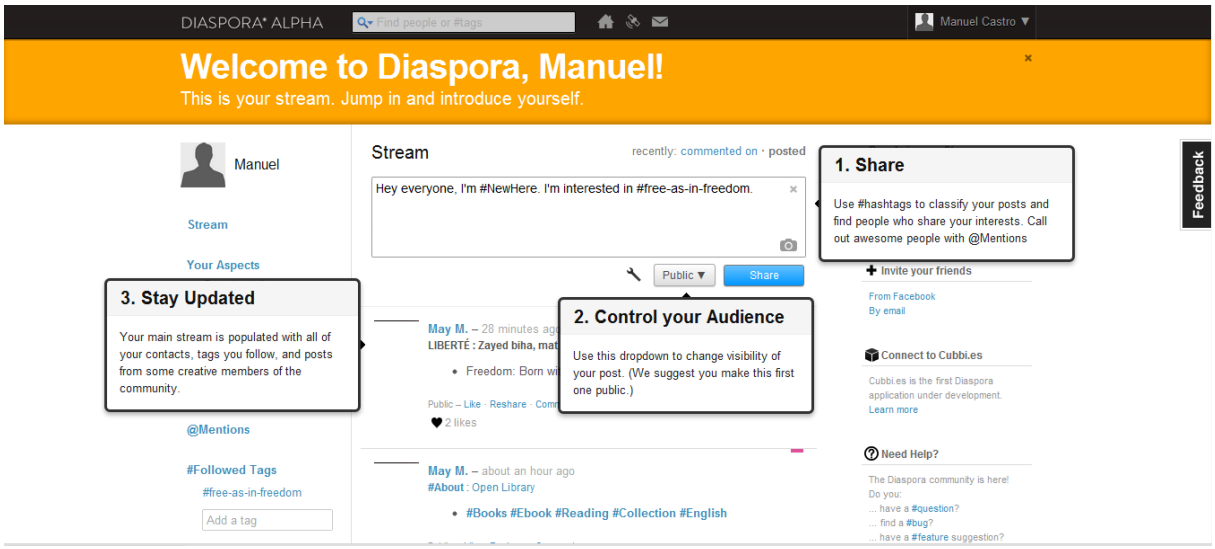


Figure 3.3 Frontend του Diaspora

Ο πηγαίος κώδικας του κοινωνικού δικτύου Diaspora έχει αναπτυχτεί στο Web Framework Ruby on Rails, το οποίο βασίζεται στην αντικειμενοστραφής γλώσσα προγραμματισμού διαδικτύου Ruby. Επιπρόσθετες τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του Diaspora είναι:

Η Haml, η οποία είναι μια γλώσσα σήμανσης που χρησιμοποιείται για απλή περιγραφή του κώδικα HTML οποιουδήποτε εγγράφου διαδίκτυο χωρίς τη χρήση του inline code. Στο Diaspora η HAML χρησιμοποιήθηκε για την δημιουργία πρότυπων (templates). Ωστόσο, Haml έχει την δική της κωδικοποίηση. Το πρότυπο της ιστοσελίδας που έχει αναπτυχτεί με την HAML παράγει δυναμικά την HTML σελίδα που προκύπτει από την εκτέλεση του κώδικα HAML.Το Sass Framework, μέσω του οποίου έχουν αναπτυχτεί τα Διαδοχικά Φύλλα Στυλ (CSS - CascadingStyleSheets) στο Diaspora. Το Sass επιτυγχάνει την συμπίεση των αρχείων CSS και JavaScript μιας ιστοσελίδας που έχει αναπτυχτεί με την γλώσσα προγραμματισμού Ruby. Το CSS είναι μια γλώσσα κωδικοποίησης των στυλ μια

ιστοσελίδας. Συγκεκριμένα το CSS ορίζει και μορφοποιεί τη διάταξη (layout) των HTML εγγράφων.

Τα Backbone.js & Handlebars.js αρχεία JavaScript, τα οποία χρησιμοποιούνται στο Diaspora για την εκτέλεση ρουτινών του κοινωνικού δικτύου. Σχεδόν όλες οι λειτουργίες του Diaspora σε επίπεδο χρήστη (client-side) δρομολογούνται από το Backbone.js, το οποίο επικοινωνεί με τον server σε επίπεδο web services χρησιμοποιώντας το πρωτόκολλο RESTful και την τεχνολογία JSON. Τα δεδομένα εκτέλεσης των web services και η παράγωγη των πρότυπων HTML διαχειρίζονται από το Handlebars.js.

Κατά την διάρκεια προσθήκης ασύγχρονων μηνυμάτων στον «τοίχο» μεταξύ φίλων η διαδικασία που ακολουθείτε από τεχνολογικής πλευράς είναι η εξής:

1. Ο χρήστης προσθέτει το μήνυμα στον τοίχο του ως aspect.
2. Γίνεται έλεγχος ότι το μήνυμα προήρθε από τον πιστοποιημένο χρήστη.
3. Εφόσον ο έλεγχος επιστρέφει αληθές απάντηση, τότε το μήνυμα αποθηκεύεται στο raw_visible_posts πίνακα για αυτόν τον χρήστη. Στην συνέχεια για το μήνυμα αυτό παράγεται ια HTML έκδοση για αποστολή μέσω WebSocket σε άλλους χρήστες.
4. Τέλος, το μήνυμα καταγράφεται σε ένα XML αρχείο, κρυπτογραφημένο και υπογεγραμμένο σε έναν φάκελο με τα παραλαμβανόμενα URLs της μορφής “http://pod.location/receive/users/:id[person_id]”. Η διαδικασία αυτή εκτελείτε στους χρήστες που θα παραλάβουν το μήνυμα.

Κατά την διάρκεια παραλαβής ενός μηνύματος η διαδικασία που ακολουθείτε είναι η εξής:

1. Ο χρήστης λαμβάνει το μήνυμα και αποκρυπτογραφεί την επικεφαλίδα (header) του μηνύματος.
2. Γίνεται έλεγχος της υπογραφής του μηνύματος για να διαπιστωθεί αν ο αποστολέας του μηνύματος είναι ο ίδιος χρήστης που έχει υπογράψει το μήνυμα. Εφόσον αυτό ισχύει τότε το μήνυμα αναγνωρίζεται ως αντικείμενο και αποθηκεύεται στην βάση δεδομένων του χρήστη.
3. Τέλος, στον παραλήπτη αποθηκεύεται το id του μηνύματος και στη συνέχεια το μήνυμα εμφανίζεται στον «τοίχο» του προφίλ του παραλήπτη. Η ίδια διαδικασία εκτελείται και από την πλευρά του αποστολέα.

Το κατανεμημένο κοινωνικό δίκτυο Diaspora έχει αναπτυχθεί σε μεγάλο βαθμό, αν και ακόμη βρίσκεται σε beta έκδοση. Πάρα πολύ χρήστες το έχουν υποστηρίξει όμως υστερεί ακόμη να απαντήσει σε απαιτήσεις όπως, τι θα συμβεί στο μήνυμα όταν ο εξυπηρετητής του χρήστη θα είναι απενεργοποιημένος και κάποιος φίλος του θα του αποστείλει ένα μήνυμα, αν οι φίλοι του αποσυνδεδεμένου χρήστη θα έχουν πρόσβαση στο πολυμεσικό περιεχόμενο του προφίλ του, κ.α. Πλέον με βάση την τελευταία ανακοίνωση της ομάδας ανάπτυξης του Diaspora, ο κώδικας και γενικότερα η συνολική δομή ανάπτυξης του είναι διαθέσιμα στην κοινότητα του.

3.1.2 Το Κατανεμημένο Κοινωνικό Δίκτυο PeerSon

Το κατανεμημένο κοινωνικό δίκτυο PeerSon δημιουργήθηκε και εξακολουθεί να αναπτύσσεται από μια κοινοπραξία επιστημόνων των πανεπιστημίων, KTH Σουηδίας, NTU Σιγκαπούρης και Warsaw Πολωνίας (14). Το PeerSon προσπαθεί να δώσει λύσεις σε ζητήματα που έχουν προκύψει από την χρήση των online κοινωνικών δικτύων, όπως για την ασφάλεια δεδομένων, την ασφαλή επικοινωνία μεταξύ χρηστών, την ανταλλαγή δεδομένων κ.α. Έως σήμερα έχει δημιουργηθεί ένα πρότυπο (prototype) το οποίο αποτελείται από διάφορα μέρη. Ένα μέρος του PeerSon αποτελεί η τεχνολογία peer-to-peer που υποστηρίζει, προσφέροντας έτσι στους χρήστες με κατανεμημένο τρόπο, τις βασικές υπηρεσίες που προσφέρουν και τα online κοινωνία δίκτυα. Ένα άλλο μέρος του PeerSon ασχολείται με την κρυπτογράφηση της πληροφορίας που θα ανταλλάζεται μεταξύ των χρηστών του, δίνοντας έμφαση στην ανταλλαγή δημόσιων και ιδιωτικών κλειδιών κρυπτογράφησης κατά την διάρκεια πιστοποίησης των χρηστών του. Με τον τρόπο αυτό το PeerSon προσφέρει ένα ασφαλές κατανεμημένο κοινωνικό δίκτυο. Βασικός σκοπός του PeerSon είναι οι online χρήστες να μπορούν να ανταλλάζουν άμεσα μεταξύ τους αρχεία, κείμενα και πολυμεσικό περιεχόμενο με ασφάλεια, χωρίς να μπορούν αυτά τα δεδομένα να χρησιμοποιηθούν από τρίτους, μη-πιστοποιημένους χρήστες. Τα αρχεία των χρηστών του θα είναι διαθέσιμα μόνο όταν ο χρήστες θα είναι online, δηλαδή συνδεδεμένοι στο προσωπικό τους κοινωνικό δίκτυο PeerSon.

Μερικές από τις υπηρεσίες που ξεχωρίζουν στο κατανεμημένο κοινωνικό δίκτυο PeerSon είναι:

- **Κοινωνικοί Υπερσύνδεσμοι (Social Links).** Μέσω των social links οι χρήστες μπορούν να συνδεθούν μεταξύ τους και να κατηγοριοποιήσουν σε ομάδες τους χρήστες που έχουν συνδεθεί, ως φίλους, ως συνεργάτες, κ.α. Επίσης, εκτός του ότι μέσω των social links το κατανεμημένο δίκτυο PeerSon επεκτείνετε με ασφάλεια μεταξύ φίλων, μπορεί επίσης να χρησιμοποιήσει τα social links για να επανασυνδεθεί με χρήστες που έχουν χάσει την μεταξύ τους σύνδεση λόγο κάποιου τεχνικού προβλήματος. Έτσι μέσω των κοινωνικών υπερσυνδέσμων καταγράφεται και παρουσιάζεται στους χρήστες του δικτύου, ο κοινωνικός του γράφος, όπως και στην πραγματική ζωή, βοηθώντας έτσι τους χρήστες να επιλέξουν/αναζητήσουν φίλους στην ομάδα που τα ενδιαφέροντα τους συμπίπτουν (14).
- **Προσωπικός Ψηφιακός Χώρος (Digital Personal Space).** Μέσω του ψηφιακού του χώρου ο κάθε χρήστης του PeerSon μπορεί να αποθηκεύει πληροφορίες για το προφίλ του, να ανεβάζει κείμενα και πολυμεσικό περιεχόμενο, να ανταλλάζει μηνύματα με άλλους χρήστες κ.α. Όλα τα παραπάνω μπορεί να τα κοινοποιεί στους φίλους του ή στις επιλεγμένες ομάδες χρηστών που θα έχει δημιουργήσει στο προφίλ του. Επίσης κάθε συμβάν (event) που θα προκύπτει στο προφίλ του χρήστη, μπορεί να κοινοποιείτε παράλληλα και στους φίλους του. Σημαντικό σημείο εδώ είναι ότι όλες οι παραπάνω κοινοποιήσεις του χρήστη θα εκτελούνται μόνο μεταξύ φίλων και μόνο εκείνοι θα μπορούν να έχουν πρόσβαση στο προφίλ του, καθώς και στο πολυμεσικό περιεχόμενο που θα έχει ανεβάσει στο προφίλ του ο χρήστης. Ως προσωπικός ψηφιακός χώρος του χρήστη μπορεί να είναι, είτε ο προσωπικός του υπολογιστής, είτε κάποιος προσωπικός διακομιστής διαδικτύου (14).

- **ΜέσαΕπικοινωνίας (Means of Communication).**Στο κοινωνικό δίκτυο PeerSon έχουν αναπτυχτεί σε μορφή prototype διαφορές υπηρεσίες για την άμεση και έμμεση επικοινωνία μεταξύ των χρηστών του. Έτσι, ένα από τα μέσα που χρησιμοποιείτε για την ασύγχρονη επικοινωνία είναι, ο προσωπικός ψηφιακός χώρος του χρήστη, τον οποίο αναλύσαμε παραπάνω. Ένα άλλο μέσω επικοινωνίας που χρησιμοποιείτε στο PeerSon είναι η υπηρεσία «κοινωνικοί υπερσύνδεσμοι», μέσω της οποίας παρέχετε η δυνατότητα στους χρήστες να βλέπουν τις συνδέσεις (peers) των άλλων χρηστών του δικτύου και να τους καλούν για να γίνουν φίλοι. Για να επιτευχθεί η σύγχρονη επικοινωνία μεταξύ φίλων στο PeerSon προϋπόθεση είναι, ότι και οι δυο φίλοι θα πρέπει να είναι συνδεδεμένοι online (14).

Η αρχιτεκτονική σχεδίαση του PeerSon. Η αρχιτεκτονική σχεδίαση του PeerSon αποτελείτε από δυο ζώνες (2-tieredarchitecture) (14). Η πρώτη ζώνη-tier χρησιμοποιείτε για την υπηρεσία αναζήτησης κόμβων (look-up services), ενώ η δεύτερη ζώνη – tier περιλαμβάνει τις ζεύξεις (peers) και τα στοιχεία των προφίλ των χρηστών του δικτύου. Η υπηρεσία αναζήτηση - look-up service αποθηκεύει μετά-δεδομένα που απαιτούνται για τον εντοπισμό των χρηστών.

Τέτοια δεδομένα είναι, η διεύθυνση IP, πληροφορίες σχετικά με τα αρχεία του χρηστών, καθώς και ειδοποιήσεις των χρηστών. Για να συνδεθεί ένα ομότιμο σύστημα με ένα άλλο ομότιμο σύστημα, αρχικά ζητά από την υπηρεσία αναζήτησης να εντοπίσει όλες τις απαραίτητες πληροφορίες για το ενδιαφερόμενο ομότιμο σύστημα. Στην συνέχεια, εφόσον υπάρχουν διαθέσιμες οι απαραίτητες πληροφορίες από την υπηρεσία αναζήτησης, τότε τα δυο ομότιμα συστήματα – υπολογιστές μπορούν να συνδεθούν απευθείας μεταξύ τους. Εφόσον τα δυο ομότιμα συστήματα έχουν συνδεθεί μπορούν στην συνέχεια να ανταλλάξουν μηνύματα και αρχεία μεταξύ τους, όπως παρουσιάζεται και στο (Σχ. 3.4) (14). Μόλις η διαδικασία της ανταλλαγής δεδομένων ολοκληρωθεί τότε η σύνδεση μεταξύ τους τερματίζεται άμεσα.

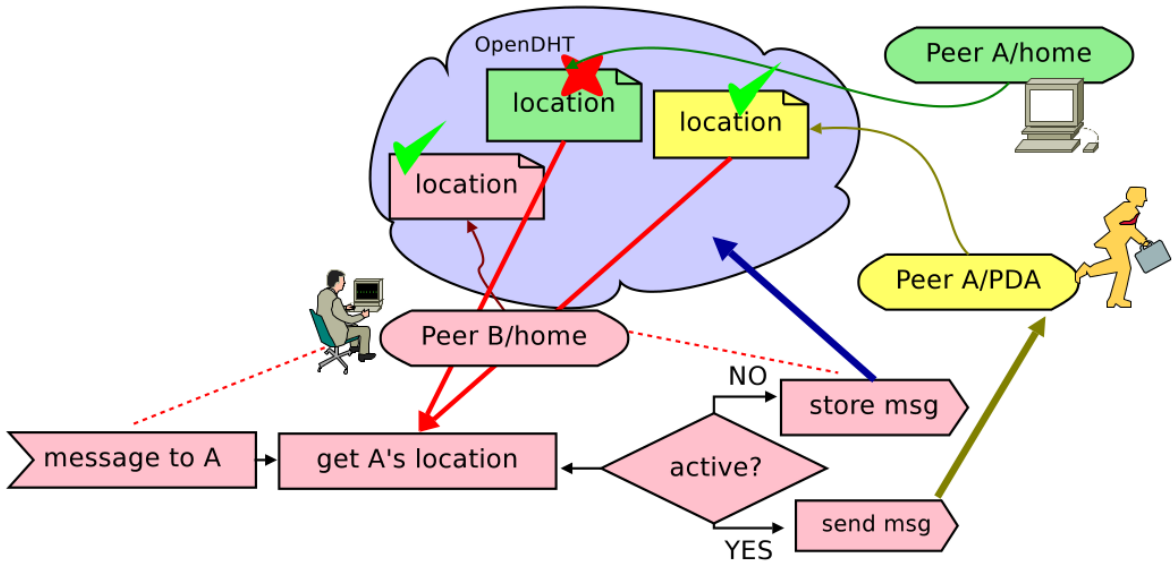


Figure 3.4 Αποστολή μηνύματος από το ομότιμο Β σύστημα στο ομότιμο Α σύστημα

Όπως παρατηρούμε και από το παραπάνω σχήμα το PeerSon χρησιμοποιεί το πρωτόκολλο OpenDHT (14)για την υπηρεσία αναζήτησης. Η ανάλυση του πρωτόκολλου OpenDHT γίνεται παρακάτω στο 4ο Κεφάλαιο. Ένα άλλο σημαντικό σημείο που αξίζει να αναφέρουμε εδώ είναι ότι, το PeerSon, πέρα από το OpenDHT πρωτόκολλο, χρησιμοποιεί και διάφορες άλλες τεχνολογίες για την εξασφάλιση της ασφάλειας στα δεδομένα των χρηστών του. Συγκεκριμένα, για τον κάθε χρήστη του δικτύου παράγει ένα μοναδικό κλειδί, το οποίο το ονομάζει Globally Unique IDs (GUID), μέσω του οποίου πιστοποιείτε ο χρήστης στο δίκτυο. Χρησιμοποιώντας το GUID του κάθε χρήστη ως ένα συμμετρικό κλειδί κρυπτογράφησης το PeerSon κρυπτογραφεί τα δεδομένα που ανταλλάζονται μεταξύ φίλων, παρέχοντας έτσι αυξημένη ασφάλεια στα δεδομένα των χρηστών του. Η διαδικασία αυτή παρουσιάζεται στο (Σχ. 3.5).

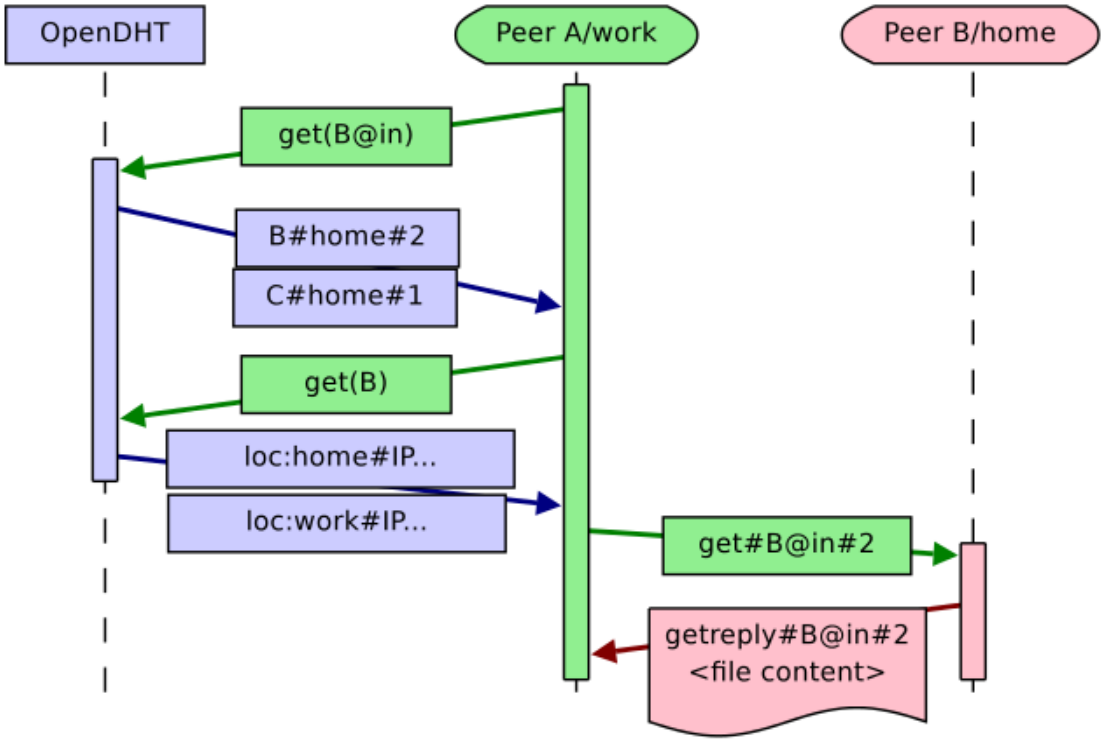


Figure 3.5 Επισκόπηση του συστήματος ανταλλαγής δεδομένων του PeerSon

Συμφώνα και με το παραπάνω σχήμα (Σχ. 3.5) παρατηρούμε ότι μέσω του κατανεμημένου κοινωνικού δικτύου PeerSon οι χρήστες μπορούν να ανταλλάξουν μεταξύ τους, ασύγχρονα/σύγχρονα μηνύματα κειμένου και αρχεία με ασφαλές τρόπο. Όλα τα παραπάνω που αναφέραμε για το PeerSon είναι σε επίπεδο σχεδίασης. Σήμερα η σχεδίαση αυτή εξελίσσεται, ενσωματώνοντας συνεχώς νέες τεχνολογίες, ενώ παράλληλα βρίσκεται σε εξέλιξη η ανάπτυξη του πρότυπου συστήματος PeerSon από την ομάδα της κοινοπραξίας επιστημόνων των πανεπιστημίων, KTH της Σουηδίας, NTU της Σιγκαπούρης και το πανεπιστήμιο του Warsaw της Πολωνίας.

3.1.3 Η Κατανεμημένη Υποδομή του Κοινωνικού Δικτύου PrPl – Private Public

Το PrPl – Private Public (15)κατανεμημένο κοινωνικό δίκτυο, είναι μια πρόταση των ερευνητών και καθηγητών Seok-Won Seong, Jiwon Seo, Matthew Nasielski, Debangsu Sengupta, Sudheendra Hangal, SengKeatTeh, RuvenChu, BenDodson, MonicaS. Lamτων τμημάτων Επιστήμης Υπολογιστών και Ηλεκτρολόγων Μηχανικών του Πανεπιστήμιου του Stanford CA. Η αρχιτεκτονική του PrPl στηρίζεται σε μια αποκεντρωμένη δομή που επιτρέπει στους χρήστες να συμμετέχουν σε online κοινωνική δικτύωση, χωρίς την απώλεια ή

υποκλοπή των δεδομένων τους. Η πρόσωπο-κεντρική αρχιτεκτονική του PrPl προσφέρει στον κάθε χρήστη να χρησιμοποιεί μια υπηρεσία Cloud Butler, η οποία παρέχει ένα ασφαλές εικονικό ψηφιακό χώρο αποθήκευσης για τα προσωπικά στοιχεία και δεδομένα του χρήστη. Επίσης, μέσω της υπηρεσίας αυτής ο κάθε πιστοποιημένος χρήστης του συστήματος μπορεί να κοινοποιήσει δεδομένα του σε άλλους χρήστες με απόλυτη ασφάλεια. Η βασική ιδέα του PrPl είναι, ότι ο κάθε χρήστης έχει τον δικό του χώρο αποθήκευσης, σε δικής του επιλογής διακομιστή. Με τον τρόπο αυτό, ένας διακομιστής μπορεί να είναι ο προσωπικός υπολογιστής του χρήστη, ένας δικτυακός σκληρός δίσκος, ή ακόμη και μια παιχνιδομηχανή με μεγάλο αποθηκευτικό χώρο που συνδέεται στο διαδίκτυο. Η ενοποίηση αυτών των ψηφιακών αποθηκευτικών χώρων του χρήστη επιτυγχάνετε μέσα από το API (ApplicationProgrammingInterface) του PrPlδικτύου. Χρησιμοποιώντας την υπηρεσία Personal-Cloud Butler τα δεδομένα του χρήστη κρυπτογραφούνται από κάποιο πιστοποιητικό και ευρετηριάζονται για κοινοποίηση μόνο στους επιλεγμένους φίλους του. Στο παρακάτω σχήμα (Σχ. 3.6) (15) παρουσιάζεται σχεδιαστικά η αρχιτεκτονική του υποσυστήματος δεδομένων του PrPl.

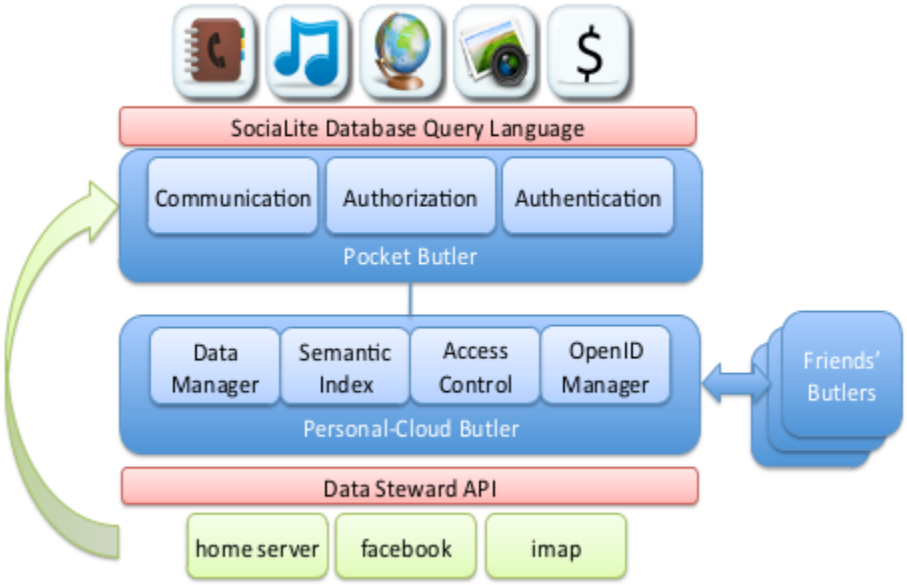


Figure 3.6 Επισκόπηση της αρχιτεκτονικής του υποσυστήματος δεδομένων του PrPl

Ένα άλλο σημαντικό σημείο του PrPl είναι ότι, χρησιμοποιεί το αποκεντρωμένο σύστημα διαχείρισης OpenID πρότυπο, έτσι ώστε οι χρήστες που έχουν καθιερωθεί ως personas από αυτό το σύστημα, θα μπορούν να έχουν πρόσβαση στα δεδομένα του χρήστη. Μερικά από τα χαρακτηριστικά που διακρίναμε στην αρχιτεκτονική δομή, καθώς και στον τρόπο επικοινωνίας των χρηστών μεταξύ τους στο PrPl είναι:

- **Οι Personal-Cloud Butlers.** Η ιδέα του Personal-Cloud Butler είναι στην πραγματικότητα μια προσωπική υπηρεσία για κάθε χρήστη του δικτύου, όπου προσφέρει ασφάλεια στα δεδομένα του χρήστη και οργανώνει και κοινοποιεί τα δεδομένα του κάθε χρήστη με τους φίλους αναλόγως των δικαιωμάτων πρόσβασης που θα έχει ορίσει για κάθε έναν από αυτούς (15). Οι υπηρεσίες που προσφέρει ο Personal-Cloud Butler είναι:

- *Σημασιολογική δεικτοδότηση των προσωπικών δεδομένων.* Παρέχει μια ενοποιημένη μορφή δεικτοδότησης των δεδομένων για την διευκόλυνση της περιήγησηκαι αναζήτηση των προσωπικών πληροφοριών του κάθε χρήστη. Η παρουσίαση των δεδομένων γίνεται σε μια τυποποιημένη μορφή RDF - Resource Description Framework ακλουθώντας συγκεκριμένα και τυποποιημένα πρότυπα οντολογιών, υποστηρίζοντας έτσι και την διαλειτουργικότητα των δεδομένων με άλλες διαδικτυακές εφαρμογές. Βέβαια όλα τα παραπάνω ισχύουν όταν τα δεδομένα του χρήστη είναι διαθέσιμα και προσβάσιμα μέσω του διαδικτύου. Ένα παράδειγμα RDF – Based database είναι το Friend (u) :- (u, a, prpl:Identity) όπου περιγράφει ότι ο u έχει ταυτότητα στην PrPl βάση δεδομένων.

- *Ομόσπονδες σύστημα αποθήκευσης.* Μέσω αυτό του συστήματος ο κάθε χρήστης έχει την δυνατότητα να επιλέξει αυτός τους δικούς του ψηφιακούς χώρους αποθήκευσης των δεδομένων του. Η υπηρεσία του Butler σε κάθε χρήστη παράγει κάποια πιστοποιητικά μέσω τον οποίων, ενεργοποιεί τα συστήματα αποθήκευσης των χρηστών για να επικοινωνήσουν μεταξύ τους, επιτυγχάνοντας έτσι και την άμεση ανταλλαγή δεδομένων μεταξύ φίλων.

- *Αποκεντρωμένη διαχείριση ταυτότητας χρήστη.* Το σύστημά επιτρέπει στους χρήστες να χρησιμοποιούν τους λογαριασμούς personas (λογαριασμός πιστοποιημένου χρήστη στο OpenID), μετατρέποντας το έτσι, ως ένα αποκεντρωμένο σύστημα διαχείρισης ταυτότητας. Μέσω αυτής της σχεδίασης οι ερευνητές του PrPl προτείνανε την επέκταση του συστήματος OpenID, έτσι ώστε να επιτυγχάνετε η αναζήτηση ενός καθορισμένου Butler, χρησιμοποιώντας το προσωπικό του OpenID. Με αυτόν τον τρόπο ο φορέας παροχής OpenID γίνεται ο πυρήνας του συστήματος για την εξακρίβωση της γνησιότητας των προσωπικών υπηρεσιών Butlers του PrPl. Το OpenID είναι ένας δωρεάν και εύκολος στην χρήση τρόπος, για να χρησιμοποιήσουμε μια μοναδική ψηφιακή ταυτότητα στο διαδίκτυο. Μας επιτρέπει έτσι να συνδεόμαστε σε ιστοσελίδες ξεχνώντας τις πολύπλοκες διαδικασίες εγγραφής και χωρίς να χρειάζεται να θυμάστε ένα ακόμη ψευδώνυμο και κωδικό πρόσβασης. Στην πράξη το OpenID είναι μια διεύθυνση διαδικτύου URL το οποίο μπορεί να έχει διάφορες μορφές ανάλογα με τον πάροχο του OpenID που θα επιλέξουμε.

- *Κοινό client proxy.* Για την υποστήριξη single sign-on για εφαρμογές που τρέχουν στη συσκευή του πελάτη, όπως για παράδειγμα ένα smartphone, μια Pocket Butler υπηρεσία χειρίζεται την ταυτοποίηση και επικοινωνία με τα Personal-Cloud Butlers. Επιπρόσθετα, μέσω των απλών διεπαφών χρήστη παρέχεται η δυνατότητα καθορισμού εξουσιοδότησης στα δεδομένα του χρήστη καθώς και η διαχείριση της προσωρινής αποθήκευσης των δεδομένων του, έτσι ώστε να επιτευχθεί ο διαμερισμός των πόρων μεταξύ των πολλαπλών εφαρμογών.

Όπως παρατηρούμε και στο (Σχ. 3.6), το PrPl για την άντληση μεγάλου όγκου δεδομένων από το κατανεμημένο δίκτυο υπηρεσιών των Butlers (από τους εικονικούς ψηφιακούς αποθηκευτικούς χώρους των χρηστών του PrPl) χρησιμοποιείτε μια γλώσσα επερωτήσεων λογικών βάσεων δεδομένων, η οποία ονομάζεται SocialLite Language. Η SocialLite σχεδιάστηκε και υλοποιήθηκε από την ομάδα ερευνητών που σχεδιάζει και αναπτύσσει το κατανεμημένο κοινωνικό δίκτυο PrPl (15). Η Socialite είναι μια επέκταση της Datalog, η οποία είναι μια γλώσσα ερωτήσεων και κανόνων για λογικές βάσεις δεδομένων (deductive databases). Υποστηρίζοντας αναδρομή (recursion) και σύνθεση (composition), η SocialLite είναι μια γλώσσα που μπορεί να «εκφράσει» σε αρκετά ικανοποιητικό επίπεδο ένα γεγονός/ερώτημα. Με τον τρόπο αυτό πολλές κοινωνικές εφαρμογές (social apps) μπορούν εύκολα να γραφτούν με την προσθήκη ενός GUI στο αποτέλεσμα του socialite ερωτήματος.

Ένα σημαντικό σημείο της SocialLite γλώσσας είναι ότι αποκρύπτει την πολυπλοκότητα ενός ερωτήματος στους Butlers των φίλων του χρήστη. Έχει είδη υλοποιηθεί ένα αρχικό πρότυπο της SocialLite γλώσσας καθώς και μερικές πρότυπες εφαρμογές στο PrPl, όπου η πρώτη δοκιμασία τους έγινε σε Cloud Hosting υπηρεσία EC2 της εταιρίας Amazon. Στο σενάριο που εκτελεστικέ από την ομάδα ανάπτυξης του PrPl είχαν δημιουργήσει 100 Butlers, και έλαβαν απάντηση από το socialite ερώτημα σε μερικά δευτερόλεπτα (15).

Τέλος, μια τελευταία τεχνολογία που εντοπίζουμε στο παραπάνω σχήμα (Σχ. 3.6) είναι το Data Steward API. Κάθε συσκευή του χρήστη η οποία ανήκει στο ομόσπονδες σύστημα αποθήκευσης του, τρέχει το λειτουργικό Data Steward API για λογαριασμό του χρήστη. Με τον τρόπο αυτό παρέχει ένα ενιαίο περιβάλλον εργασίας στον Butler και στις PrPl εφαρμογές, αποκρύπτοντας τις λεπτομέρειες σχετικά με το πώς οι πληροφορίες αποθηκεύονται στις συσκευές. Το Data Steward API είναι υπεύθυνο και για την ενημέρωση του Butler σχετικά με τα δεδομένα blobs (binary large objects). Περιοδικά το Data Steward API στέλνει μερικά heartbeats στον Butler ενημερώνοντας τον με τις τελευταίες αλλαγές των blobs, καθώς και με τις τελευταίες αλλαγές των πληροφοριών προσπέλασης τους (15). Ο Butler, στην πραγματικότητα ενημερώνεται για τις τοποθεσίες των blobs, αντιστοιχίζοντας την PrPl URI διεύθυνση σε μια διεύθυνση πηγής δεδομένων URI της μορφής “http://” ή “file://”. Συγκρίνοντας το PrPl με τα Online Κοινωνικά Δίκτυα οι υπηρεσίες (data sources)[26] που προσφέρει είναι:

- ***Τα εργαλεία διαχείρισης του προσωπικού λογαριασμού του χρήστη.*** Ο χρήστης μπορεί να διαχειρίζεται το προσωπικό του ημερολόγιο και τις επαφές του. Διαχείριση των αρχείων σε προσωπικό υπολογιστή του χρήστη. Η υπηρεσία Batler θα πρέπει να εντοπίσει και να αποθηκεύσει όλα τα μεταδιδόμενα των αρχείων του χρήστη στον προσωπικό του υπολογιστή. Τέτοιου είδους αρχεία είναι, φωτογραφίες, έγγραφα, videos κ.α.
- ***Διαχείριση των μηνυμάτων και των επισυναπτομένων αρχείων.*** Πολλοί χρήστες καταφεύγουν στην ταχυδρομική αποστολή δεδομένων στους εαυτούς τους έτσι ώστε τα δεδομένα να μπορούν να τα προσπελαστούν από οπουδήποτε. Ωστόσο, είναι δύσκολο να εντοπίσεις την σωστή έκδοση ενός αρχείου. Οι μετά πληροφορίες σχετικά με τα μηνύματα ηλεκτρονικού ταχυδρομείου, επαφές, και τα συνημμένα, είναι μια σημαντική πηγή στοιχείων για την ενσωμάτωση στο PrPlFederationStoredSystems. Τα μηνύματα ηλεκτρονικού ταχυδρομείου από ηλεκτρονικά καταστήματα όπως το Netflix και Amazon περιέχουν όλο το ιστορικό

αγορών του χρήστη κατά τα προηγούμενα έτη και έως σήμερα. Τέτοιου είδους πληροφορίες μπορούν πολύ εύκολα να ευρετηριαστούν μέσω του PrPl.

- ***Σύνδεση με τα Online κοινωνικά δίκτυα.*** Πολλοί χρήστες έχουν ανεβάσει προσωπικά δεδομένα στα online κοινωνικά δίκτυα, όπως το Facebook, Twitter κ.α. Μέσω του Data Steward API τα δεδομένα αυτά μπορεί να τα αναλύσουμε από τα διάφορα online κοινωνικά δίκτυα και στην συνέχεια να τα οργανώσουμε με βάση το πρότυπο του PrPl. Επίσης εδώ μπούμε να προσθέσουμε απευθείας δεδομένα στο Facebook . Η παραπάνω υπηρεσίες παρουσιάζονται στην (Εικ. 3.7) (15).

Figure 3.7 Η διεπαφή χρήστη του PrPlButler με την εφαρμογή PhotoWall

Αυτό που παρατηρούμε στο PrPl κοινωνικό δίκτυο είναι ότι, ενώ υποστηρίζει μερικώς αυτά που υποστηρίζουν τα Online κοινωνικά δίκτυα, προσεγγίζει το θέμα του κατανεμημένου κοινωνικού δικτύου από μια άλλη οπτική γωνία. Συγκεκριμένα προσφέρει μια ειδική υποδομή κατανεμημένης κοινωνικής δικτύωσης πάνω στην οποία μπορεί να δημιουργηθούν πολλές εφαρμογές κοινωνικής δικτύωσης. Επιπρόσθετα, η σχεδίαση του εστιάζει στην μεταφορά μεγάλου όγκου δεδομένων μεταξύ των χρηστών του.

3.1.4 Το Κατανεμημένο Κοινωνικό Δίκτυο LifeSocial.KOM

Το LifeSocial(16) αναπτύχθηκε από τον KalmanGraffικαι διάφορους φοιτητές στο Εργαστήριο Πολυμέσων Επικοινωνίας (KOM) υπό την επίβλεψη του καθηγητή RalfSteinmetzστο Τεχνολογικό Πανεπιστήμιο του Darmstadt (TechnischeUniversitätDarmstadt) στην Γερμανία. Η ερευνητική ομάδα που ασχολήθηκε σχετικά με τη θεωρία των κατανεμημένων συστημάτων ήταν υπό την επίβλεψη του καθηγητή

Christian Scheideler στο Πανεπιστήμιο του Paderborn στην Γερμανία. Στην πραγματικότητα το LifeSocial.KOM είναι μια p2p υποδομή για ασφαλείς και αποκεντρωμένη online κοινωνική δικτύωση, το οποίο παρέχει την λειτουργικότητα και τις υπηρεσίες που προσφέρουν τα online κοινωνικά δίκτυα σε ένα πλήρως κατανεμημένο και ασφαλή περιβάλλον (16). Αποτελείται από επιμέρους εφαρμογές (plugins) μέσω των οποίων μπορεί να παρατείνει τις υπηρεσίες και την λειτουργικότητα του, παρέχοντας ασφαλή επικοινωνία και ελεγχόμενη πρόσβαση στα δεδομένα των χρηστών. Επίσης, μια επιπλέων υπηρεσία που υποστηρίζει το LifeSocial.KOM είναι η αξιολόγηση και βελτίωση της ποιότητας των υπηρεσιών του, καλύπτοντας έτσι τις ανάγκες των χρηστών που επικοινωνούν μεταξύ τους, καθώς και των παρόχων internet. Η πλατφόρμα λειτουργεί αποκλειστικά χρησιμοποιώντας τους πόρους των συστημάτων των χρηστών του. Με τον τρόπο αυτό μειώνει το λειτουργικό κόστος για έναν πάροχο. Μέχρι σήμερα έχει σχεδιαστεί και υλοποιηθεί μια δοκιμαστική πλατφόρμα αξιολόγησης των επιμέρους εφαρμογών που εξακολουθούν να υλοποιούνται, με έμφαση στην επισήμανση των δυνατοτήτων της P2P διασύνδεσης στον τομέα των online κοινωνικών δικτύων.

Η αρχιτεκτονική δομή του LifeSocial.KOM παρουσιάζεται στο (Σχ. 3.8). Αυτό που παρατηρούμε στο (Σχ. 3.8) είναι, ότι η αρχιτεκτονική δομή του LifeSocial.KOM είναι χωρισμένη σε 3 βασικά μέρη (16):

- ***P2P Platform:*** Overlay and Mechanisms, όπου αναλύονται οι τεχνολογίες και τα πρωτοκόλλα που έχουν χρησιμοποιηθεί για την επίτευξη της p2p επικοινωνίας μεταξύ των συστημάτων των χρηστών.
- ***Plugins and Apps,*** όπου αναλύονται η εφαρμογές (plugins/apps) που έχουν υλοποιηθεί για την ασφαλή επεξεργασία και ανταλλαγή δεδομένων μεταξύ των χρηστών του LifeSocial.KOM.
- ***Graphical User Interface,*** όπου παρουσιάζονται οι επιμέρους διεπαφές χρήστη των εφαρμογών, που χρησιμοποιούνται για την φιλική με τον χρήστη παρουσίαση των δεδομένων.

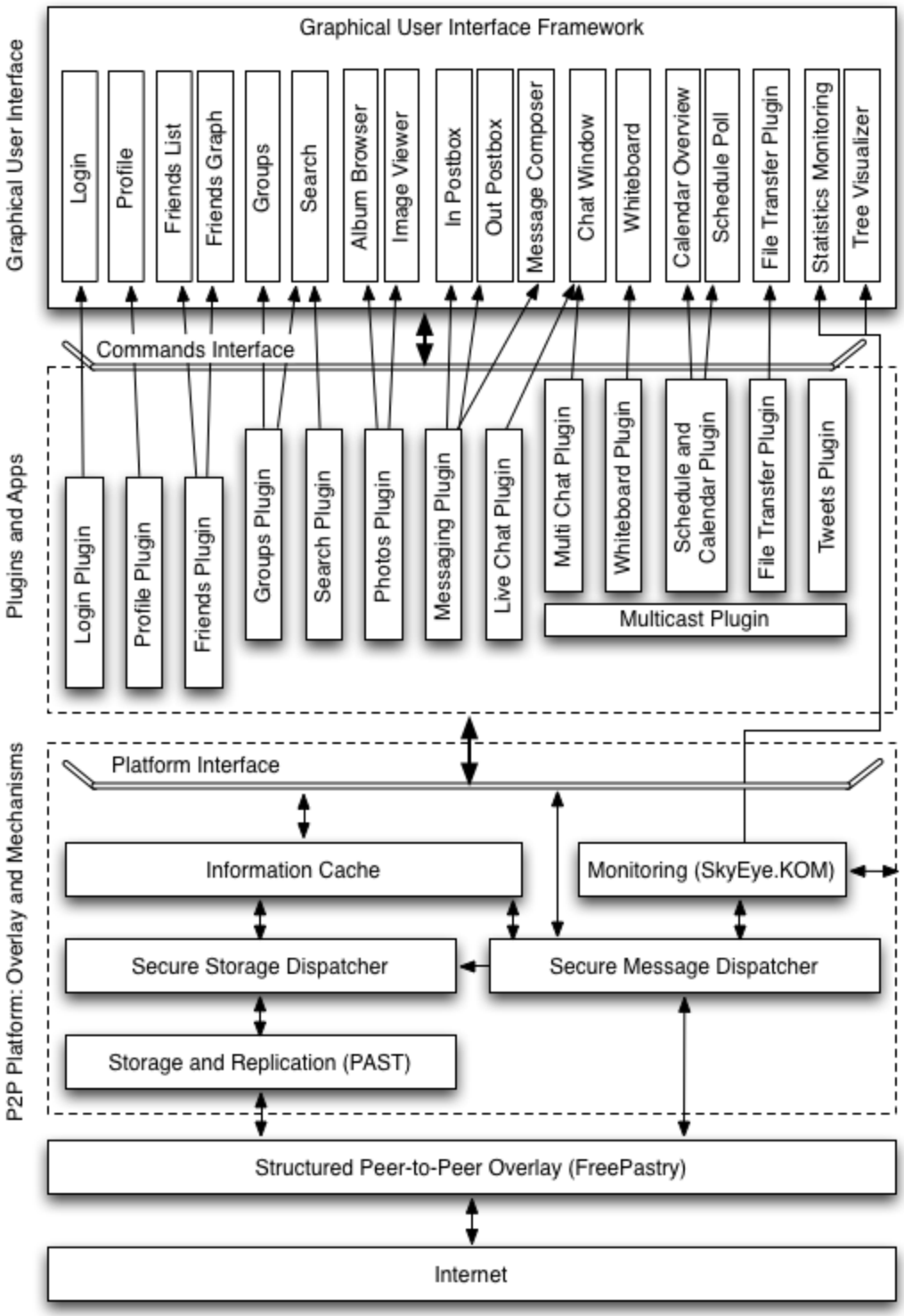


Figure 3.8 Επισκόπηση της αρχιτεκτονικής δομής του LifeSocial.KOM

Στο επίπεδο δικτύου, της αρχιτεκτονικής δομής του LifeSocial.KOM, βρίσκετε μια δομημένη P2P υπηρεσία, η οποία παρέχει τη λειτουργικότητα δρομολόγησης μεταξύ ομότιμων χρησιμοποιώντας το ID του κάθε κόμβου. Η διαδικασία αυτή επιτυγχάνετε μέσω της

λειτουργικότητας του FreePastry (17), το οποίο είναι μια δωρεάν επέκταση το Pastry (18)και παρέχει ένα αξιόπιστο αντικείμενο αποθήκευσης των δεδομένων που ονομάζεται PAST (18).

Το LifeSocial.KOM χρησιμοποιώντας τις υπηρεσίες του FreePastry και του PAST τα αντικείμενα που δημιουργούνται στο LifeSocial.KOM δίκτυο, τα οποία οργανώνονται σε λίστες, είναι αξιόπιστα στα οποία αποθηκεύονται πληροφορίες για τα δεδομένα του χρήστη, όπως για παράδειγμα ο σύνδεσμος μιας φωτογραφίας, αν ανήκει σε κάποια ομάδα φωτογραφιών (photoalbum) κ.α. Η ανάκτηση αυτών των αντικειμένων από το δίκτυο βασίζεται στην ταυτότητα τους όπου χρόνο αναζήτησης/απόκρισης τους δεν ξεπερνά τα δύο δευτερόλεπτα. Η διαδικασία αυτή παρουσιάζεται στο (Σχ. 3.9).

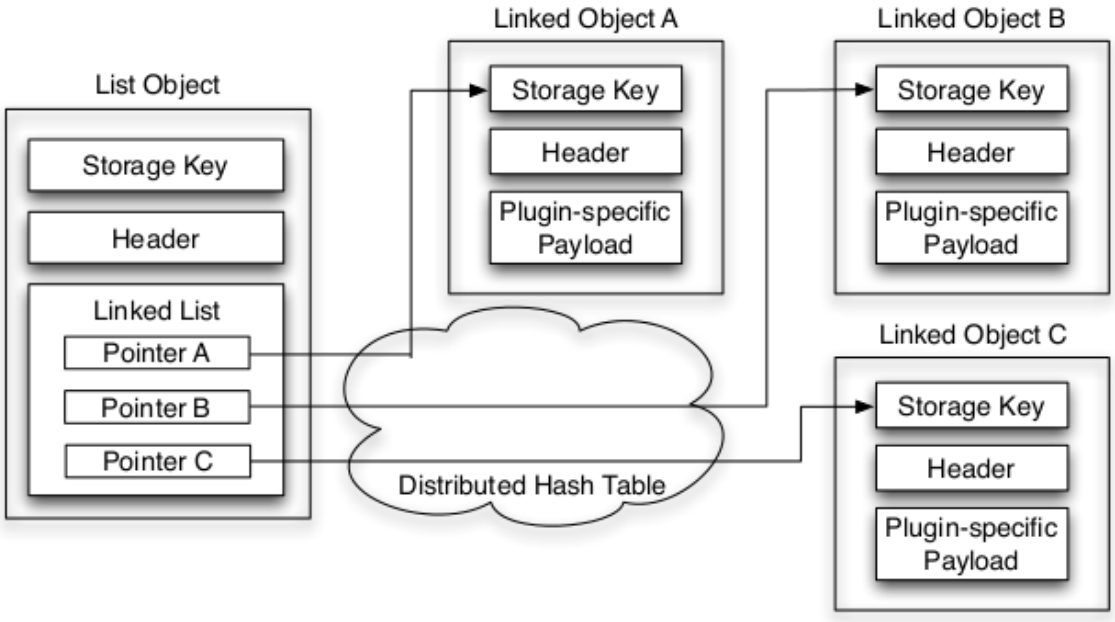


Figure 3.9Η κατανεμημένη διασυνδεδεμένη λίστα αντικειμένων στο LifeSocial.KOM

Οι βασικές εφαρμογές (plugins) που έχουν υλοποιηθεί έως σήμερα για το LifeSocial.KOM είναι οι ακόλουθες:

- **Login:** Εγγραφή / Σύνδεση του χρήστη μέσω των κλειδιών κρυπτογράφησης.
- **Profile:** Παρουσιάζεται μια απλή παρουσίαση των στοιχείων του χρήστη
- **Friends:** Είναι μια λίστα που διατηρεί διασυνδεδεμένα τα προφίλ των φίλων.
- **Messaging:** Ασύγχρονα μηνύματα, όπως το email.
- **Photo:** Μια λίστα photo album που διατηρεί διασυνδέσεις (links) με τις φωτογραφίες του χρήστη.
- **Groups:** Μια λίστα από χρήστες που έχουν συνδεθεί σε μια κοινή ομάδα χρηστών ασπάζοντας τα ίδια ενδιαφέροντα.
- **Tweets:** Μια λίστα από την κατάσταση του προφίλ του χρήστη, καθώς και με τους φίλους followers.
- **(Group) Chat:** Συγχρονισμένα μηνύματα μεταξύ χρηστών.
- **File transfer:** Ανταλλαγή αρχείων μεταξύ χρηστών.

- **Games for two with spectators**: Ένα παράδειγμα αυτής της εφαρμογής είναι το παιχνίδι Tic Tac Toe.
- **Whiteboard:** Μέσω της εφαρμογής αυτής επιτυγχάνετε η συνεργατικότητα μεταξύ των φίλων σχεδιάζοντας από κοινού σε έναν ενιαίο πίνακα.
- **Schedule and Calendar:** Διατήρηση ημερολογίου του κάθε χρήστη.
- **Multicast:** Μπορεί ο κάθε χρήστης να κοινοποιήσει ένα αρχείο σε μια ομάδα χρηστών.

Η διεπαφή χρήστη του front end του LifeSocial.KOM παρουσιάζεται στην (Εικ. 3.10) και στην (Εικ. 3.11).

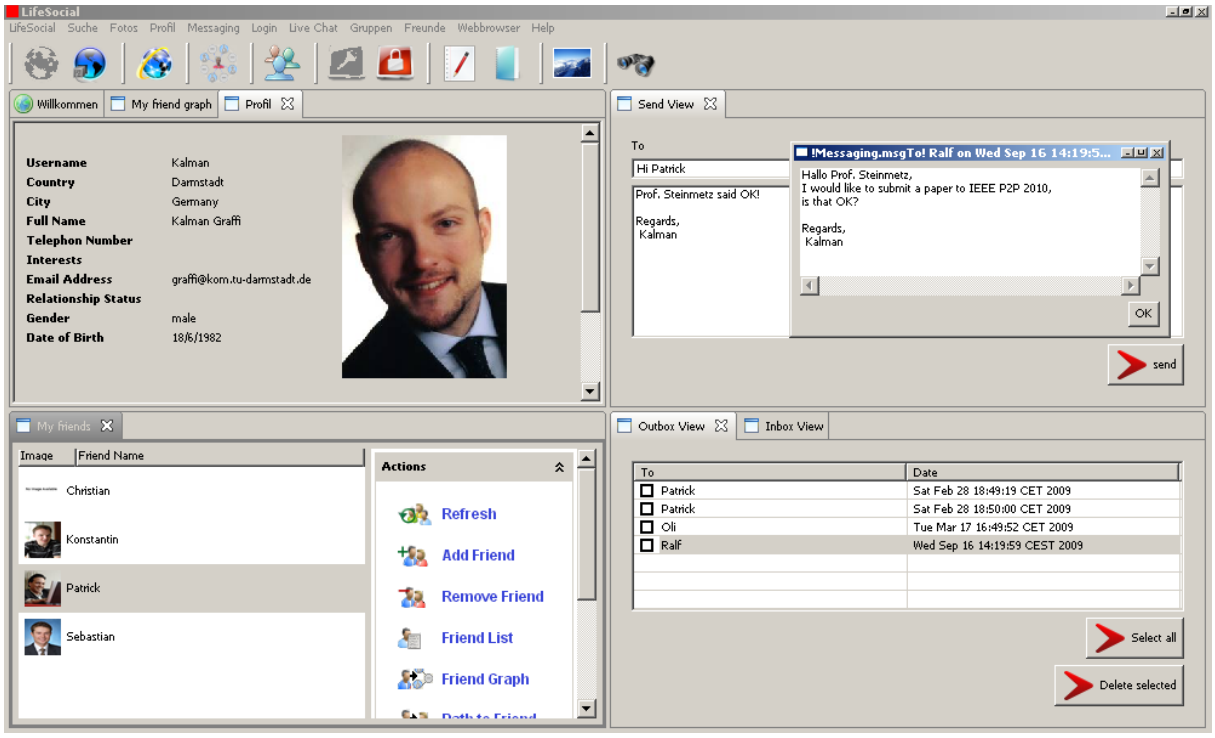


Figure 3.10 Η διεπαφή χρήστη του LifeSocial.KOM του προφίλ, των μηνυμάτων, της λίστας φίλων

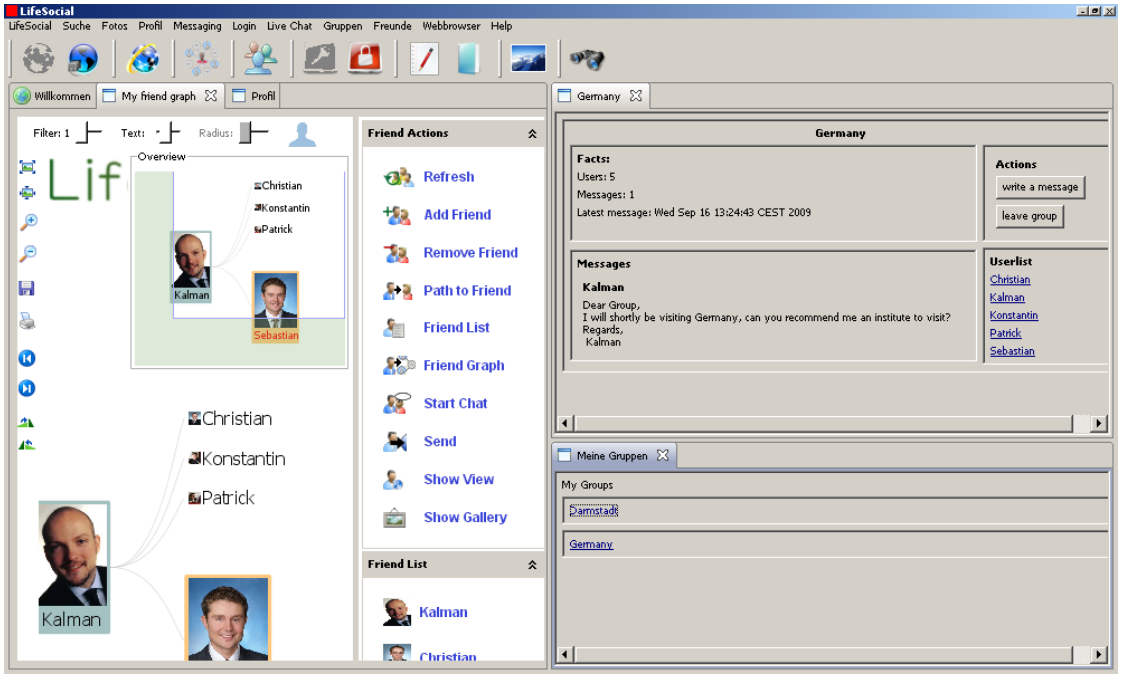


Figure 3.11 Η διεπαφή χρήστη του LifeSocial.KOM των ομάδων χρηστών σε μορφή γράφου

Συγκρίνοντας το LifeSocial.KOM με τα online κοινωνικά δίκτυα παρατηρούμε ότι, ενώ υποστηρίζει όλες τις υπηρεσίες που προσφέρουν τα online κοινωνικά δίκτυα, τα δεδομένα των χρηστών παραμένουν ενεργά όσο ο χρήστης είναι συνδεδεμένος στο δίκτυο από τον δικό του προσωπικό υπολογιστή και δεύτερον οι έως τώρα εφαρμογές του LifeSocial.KOM είναι εφαρμογές desktop και δεν εκτελούνται μέσω κάποιου φυλλομετρητή διαδικτύου.

3.1.5 Η Κατανεμημένη Πλατφόρμα Κοινωνικής Δικτύωσης MyNet

Το MyNet (19)είναι μια πλατφόρμα, η οποία υποστηρίζει ασφαλή P2P υπηρεσίες προσωπικής και κοινωνικής δικτύωσης. Το MyNet έχει υλοποιηθεί στην κορυφή του επιπέδου Unmanaged Internet Architecture(UIA) και επιτρέπει στους απλούς χρήστες να οργανώνουν και να μοιράζονται εύκολα τους πόρους εντός των γειτονικών κοινωνικών τους δικτύων. Μέσω ενός οδηγού-wizard που ονομάζεται MyNetBook, το οποίο είναι ένα από τα εργαλεία του MyNet UI καθοδηγεί τον χρήστη να αποτυπώσει την ταυτότητα του σε μια νέα συσκευή. Έτσι, όταν ο χρήστης έχει αποτυπώσει την δική ταυτότητα και σε άλλες δίκες του συσκευές, τότε οι συσκευές ενώνονται μέσω των υπηρεσιών του MyNet και δημιουργούν ένα προσωπικό δίκτυο, PDC – PersonalDeviceclusters, για τον χρήστη. Πολλά τέτοιου είδους προσωπικά δίκτυα PDCs μπορούν να συνδεθούν μεταξύ τους με την βοήθεια της υπηρεσίας του MyNet Introduction Process, δημιουργώντας ένα κοινωνικό δίκτυο. Κατά την διάρκεια της σύνδεσης μεταξύ των συσκευών στα επίπεδα UIA τους, γίνεται η ανάλλαγη κρυπτογραφημένων δεδομένων δρομολόγησης SIDs(StartpointIdentifiers) και EID (EndpointIdentifiers). Κάθε συσκευή του χρήστη παράγει ένα ιδιωτικό/δημόσιο κλειδί που κρυπτογραφεί τα δεδομένα που ανταλλάζει με άλλες συσκευές. Οι συσκευές του χρήστη

μπορεί να είναι ένα pc, ένα smartphone, μια παιχνιδομηχανή κ.α. τα οποία να υποστηρίζουν την υπηρεσία δικτύωσης.

Οι χρήστες του MyNet έχουν την δυνατότητα να καθορίσουν αυτοί το επίπεδο προσβασιμότητας στους πόρους τους από άλλους χρήστες μέσω των Passlets (πάσο ή εισιτήριο). Το πλαίσιο ασφαλείας του MyNet επιτρέπει την απεριόριστη πρόσβαση του ιδιοκτήτη στις συσκευές του, χωρίς να απαιτείται οποιαδήποτε περαιτέρω διαδικασία για τον ορισμό δικαιωμάτων πρόσβασης. Όσον αφορά τις επαφές και φίλους του χρήστη, μπορούν να χρησιμοποιούν τους πόρους του χρήστη σε τέτοιο επίπεδο αναλόγως με τα δικαιώματα πρόσβασης που τους έχουν χορηγηθεί από τον χρήστη που έχει την κυριότητα των δεδομένων του προφίλ του. Η συνδεσιμότητα και το δίκτυο πλοήγησης γίνονται τόσο απλά όπως επιλέγοντας κάποιο εικονίδιο σε μια οθόνη, ενώ η πολυπλοκότητα των λειτουργιών για τον εντοπισμό υπηρεσιών, δικτύου πρόσβασης και ασφάλειας παραμένουν κρυφά από τον τελικό χρήστη.

Η αρχιτεκτονική του MyNet παρουσιάζεται στο παρακάτω σχήμα (Σχ. 3.12).

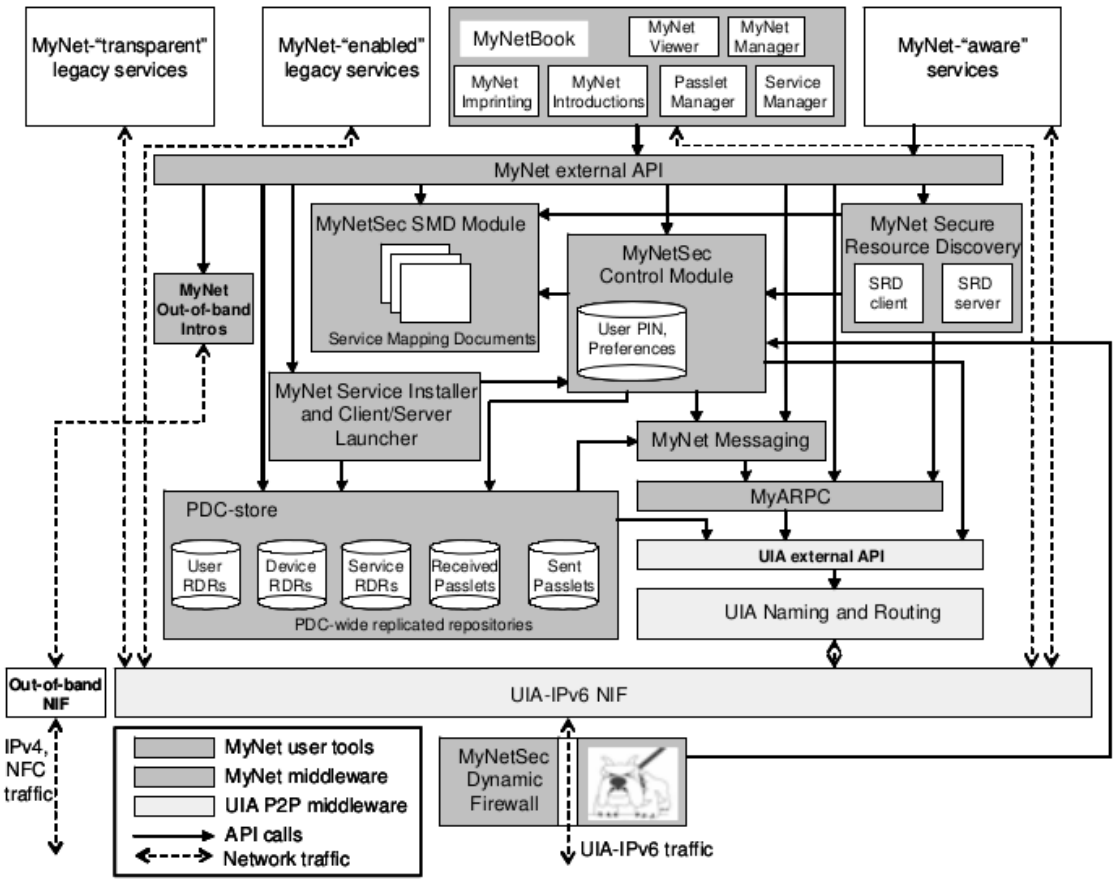


Figure 3.12 Επισκόπηση της αρχιτεκτονικής δομής του MyNet

Η προσθήκη μιας συσκευής ή μιας επαφής στο MyNet PDC εκτελείται από την υπηρεσία Out-of- Band Introductions. Στη συνέχεια εκκινείτε η διαδικασία εντοπισμού συσκευών ομότιμων του MyNet μέσω των μηχανισμών που δεν χρησιμοποιούν UIA-IPv6. Μέσω της υπηρεσίας Out-of- Band επιτυγχάνετε η διαδικασία εισαγωγής της συσκευής στο MyNet PDC χρησιμοποιώντας τις επιμέρους τεχνολογίες των εφαρμογών του MyNet. Μέσω των MyNet – "aware" εφαρμογών οι διεργασίες μπορούν να χρησιμοποιούν την ασύγχρονη απομακρυσμένης διασύνδεσης υπηρεσία RPC- Remote Procedure Call, όπου στην περίπτωση του MyNet ονομάζεται MyARPC. Μέσω αυτής της υπηρεσίας επιτυγχάνετε η ανταλλαγή μηνυμάτων μεταξύ των συσκευών. Το MyNet χρησιμοποιεί την RPC βιβλιοθήκη του UIA, η οποία βασίζεται στην τεχνολογία της SUN RPC. Αντίστοιχες τεχνολογίες του MyARPC είναι οι XML-RPC, SOAP κ.α. Όλα τα μηνύματα του MyARPC μεταδίδονται μόνο μέσω ασφαλές SSL συνδέσεων.

Το MyNet μέσω της υπηρεσίας MyNet Messaging εξασφαλίζει ότι τα μηνύματα που αποστέλλονται προς μόνο μια άλλη συσκευή δεν πρόκειται να χαθούν ακόμη και όταν η συσκευή αυτή είναι απενεργοποιημένη (19). Τα μηνύματα οργανώνονται σε μια ουρά στην συσκευή αποστολέα, όπου για το κάθε μήνυμα που δεν έχει παραληφθεί ορίζεται και μια ημερομηνία διαγραφής του. Συνήθως η διάρκεια διατήρησης των μηνυμάτων αυτών ορίζεται για αρκετές ημέρες. Το MyNetSec Control του MyNet λειτούργει ως ένας δυναμικός τοίχος προστασίας του δικτύου διαχειρίζοντας κατάλληλα την κυκλοφορία των δεδομένων συμφώνα με τις πληροφορίες πρόσβασης που φέρουν τα Passlets.

Η υπηρεσία SMD είναι υπεύθυνη για το parsing των SMD έγγραφων, των υπηρεσιών που είναι εγκατεστημένες σε κάθε συσκευή και για την παροχή πληροφοριών σχετικά με αυτές τις υπηρεσίες για την ανακάλυψη MyNet (SRD) και των μονάδων ασφαλείας. Μια τέτοια SMD έγγραφο παρουσιάζεται στο παρακάτω σχήμα (Σχ. 3.13).

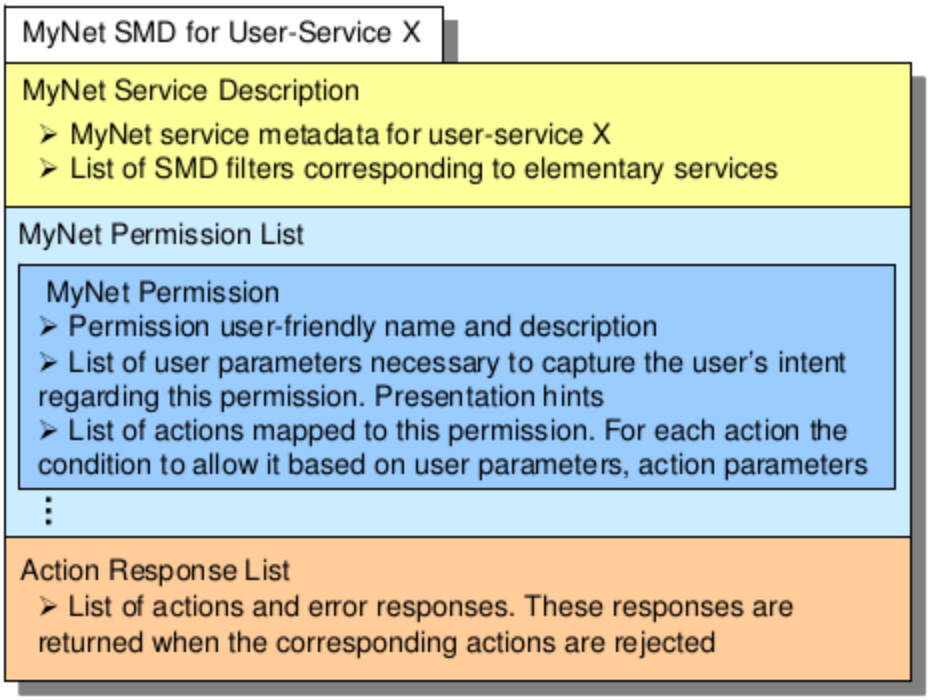


Figure 3.13 Επισκόπηση της δομής του SMD εγγράφου

Οι τεχνολογίες που έχουν χρησιμοποιηθεί για την ανάπτυξη του πρότυπου MyNet είναι: οι γλώσσες προγραμματισμού C, C++, Shell scripts και η Python. Η διεπαφή χρήστη του MyNet παρουσιάζεται στις δυο παρακάτω εικόνες (Εικ. 3.14) και (Εικ. 3.15).

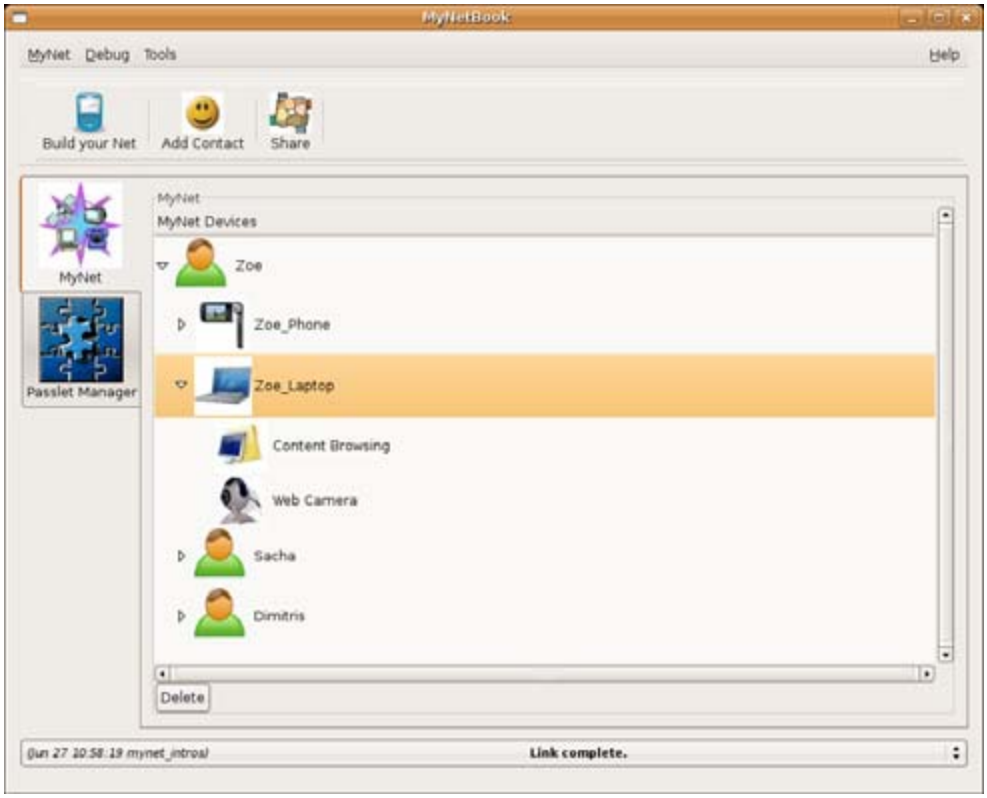


Figure 3.14 Η διεπαφή χρήστη του MyNetViewer

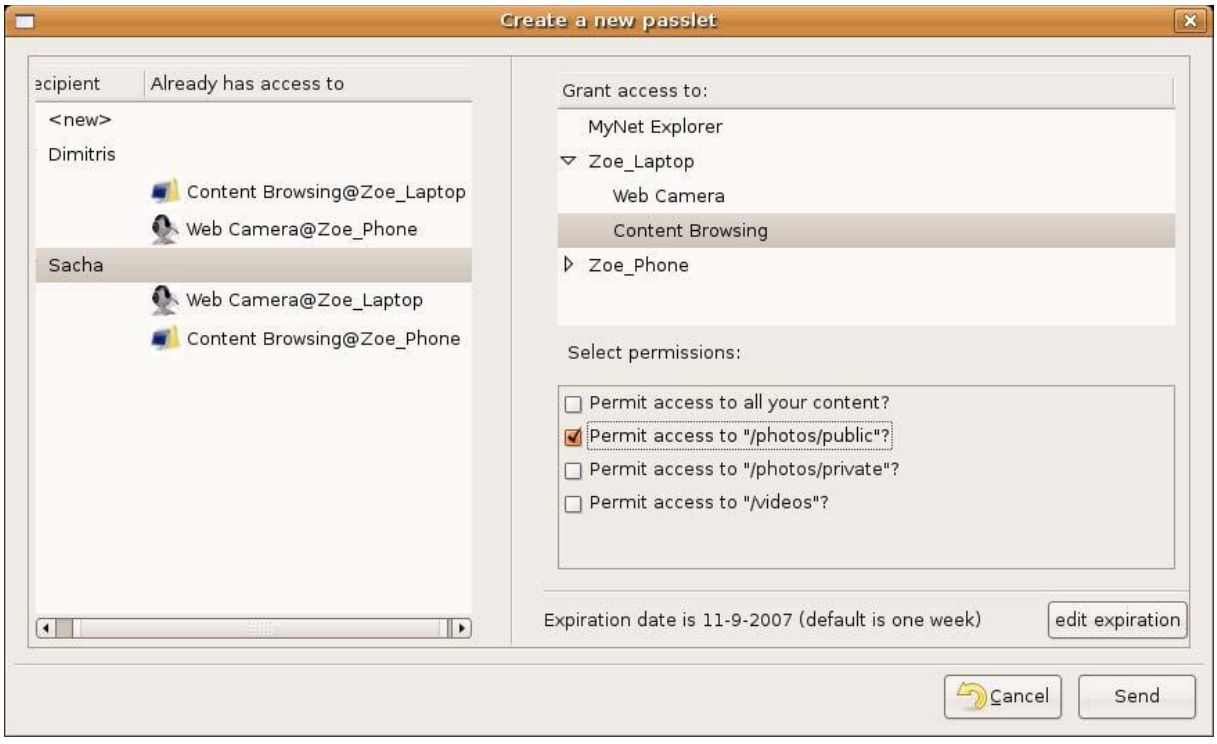


Figure 3.15 Η διεπαφή χρήστη του εργαλείου διαχείρισης MyNetPasslet

Συγκρίνοντας τις υπηρεσίες που προσφέρει το MyNet με τις υπηρεσίες που προσφέρουν τα online κοινωνικά δίκτυα, παρατηρούμε ότι μέσω του MyNet οι χρήστες μπορούν να ανταλλάξουν ασύγχρονα και συγχρονισμένα μηνύματα, καθώς και να μεταφέρουν αρχεία μέσω των desktop εφαρμογών του MyNet και όχι μέσω ενός φυλλομετρητή διαδικτύου. Αυτό αποτελεί έναν περιορισμό στην προσβασιμότητα του χρήστη από οπουδήποτε.

Επίσης, δεν υλοποιεί όλες τις υπηρεσίες που προσφέρει ένα online κοινωνικό δίκτυο, όπως το προφίλ του κάθε χρήστη να είναι online και όταν αυτός αποσυνδέεται, tagging κ.α. Τελευταία παρατηρείται ότι το MyNet έχει αρχίσει να χρησιμοποιείτε ως εφαρμογή σε τηλεοράσεις τελευταίας γενιάς με πρόσβαση στο διαδίκτυο, σε συνδυασμό με μια δεύτερη εφαρμογή που ονομάζεται Falcon από την InfoSys Labs.

3.1.6 Το Κατανεμημένο Κοινωνικό Δίκτυο Peerscape

Το Peerscape (20)είναι ένα πειραματικό peer-to-peer κοινωνικό δίκτυο το οποίο έχει υλοποιηθεί ως μια επέκταση (plugin) τουφυλλομετρητή Firefox 3.x. Το Peerscape έχειαναπτυχθείαπότο Helsinki Institute for Information Technology-HIIT. Υλοποιεί ένα είδος serverless (η μη- διακομιστή) ανάγνωσης-εγγραφής δεδομένων υποστηρίζοντας third-party AJAX εφαρμογές. Στην πραγματικότητα αυτό που επιτυγχάνει το Peerscape, για τους χρήστες που το έχουν εγκαταστήσει ως επέκταση στο Firefox είναι, να αποθηκεύει τοπικά στον υπολογιστή του χρήστη, αντίγραφα των δεδομένων του προφίλ του και των φίλων του, καθώς και των ομάδων που έχει συμμετάσχει ο χρήστης. Επίσης, αποθηκεύει τα αντίγραφα των δεδομένων όπου έχει πλοηγηθεί ο χρήστης. Οι υπολογιστές των χρηστών «φίλων» που αποθηκεύουν τα ίδια δεδομένα δημιουργούν συνδέσεις μεταξύ τους, έτσι ώστε να διατηρούν τα δεδομένα ενημερωμένα με τις τελευταίες αλλαγές που μπορεί να κάνουν οι χρήστες τους.

Το Peerscape χρησιμοποιεί ένα δημόσιο κλειδί κρυπτογράφησης υπογραφών, για την κωδικοποίηση των δεδομένων που ανταλλάζονται μεταξύ των χρηστών, καθώς και για την κωδικοποίηση των συνδέσεων μεταξύ των φίλων και των ομάδων τους. Η πρότυπη έκδοση του Peerscape έχει αναπτυχθεί με την βοήθεια της αντικειμενοστραφής γλώσσας προγραμματισμού Python. Το Peerscape χρησιμοποιεί το REST για να εκτελεί διαδικασίες μέσω της HTTP διεπαφής. Επίσης, για την ενημέρωση των δεδομένων στην διεπαφή πλοήγησης του χρήστη χρησιμοποιεί το JSON.

Το κύριο περιεχόμενο της βάσης δεδομένων του Peerscape είναι οργανωμένο σε κοινά σύνολα βάσεων δεδομένων (datasets). Κάθε σύνολο δεδομένων έχει ένα μοναδικό αναγνωριστικό κλειδί πληροφοριών «ID», το οποίο αποτελείται από 40 δεκαεξαδικά ψηφία. Τα σύνολα δεδομένων μοιάζουν με ιεραρχίες αρχείων, όπου το κάθε σύνολο αποτελείται από μια συλλογή ζευγαριών μεταξύ «κλειδιών - τιμών», όπου τα κλειδιά είναι διαδρομές και οι τιμές δεδομένα (byte-strings). Η κάθε διαδρομή αποτελείται από μια ή περισσότερες μη κενές συμβολοσειρές όπου χρησιμοποιείται ο χαρακτήρας «/» ως διαχωριστής. Η διεπαφή χρήστη του Peerscape παρουσιάζεται στις παρακάτω εικόνες (Εικ. 3.16) και (Εικ. 3.17).

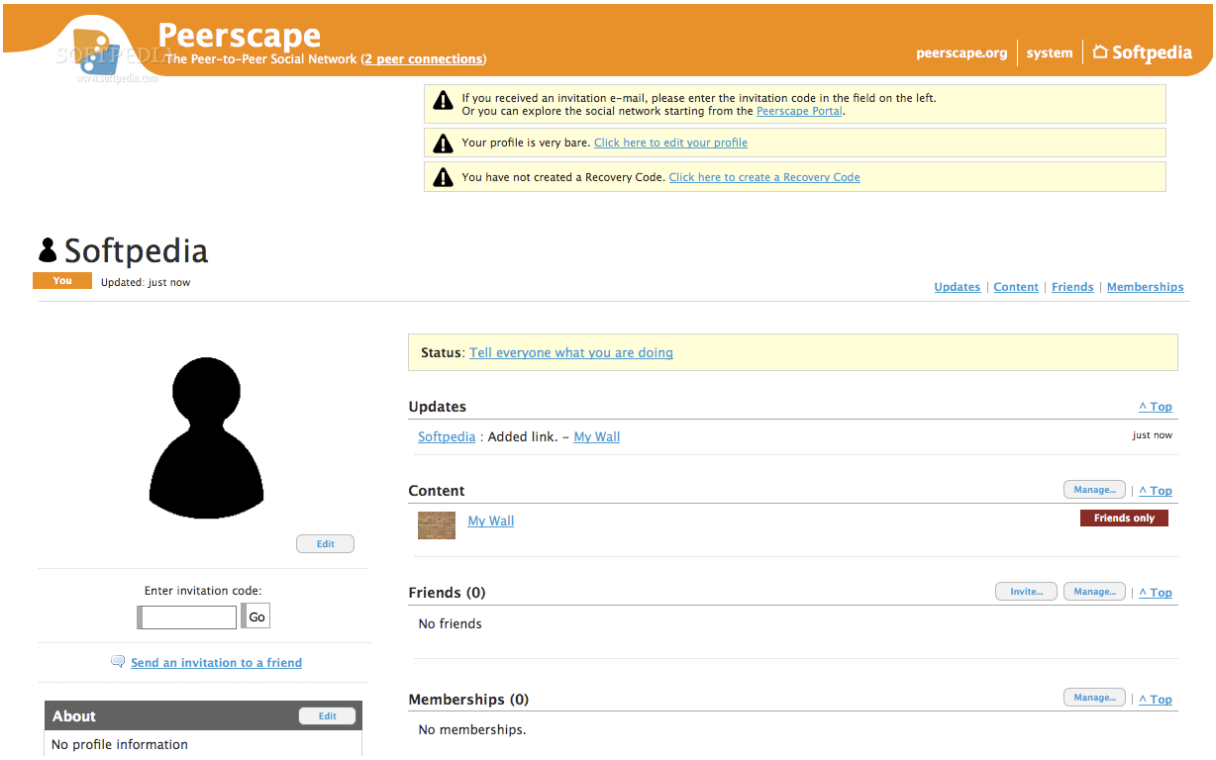


Figure 3.16 Η διεπαφή χρήστη του Peerscape

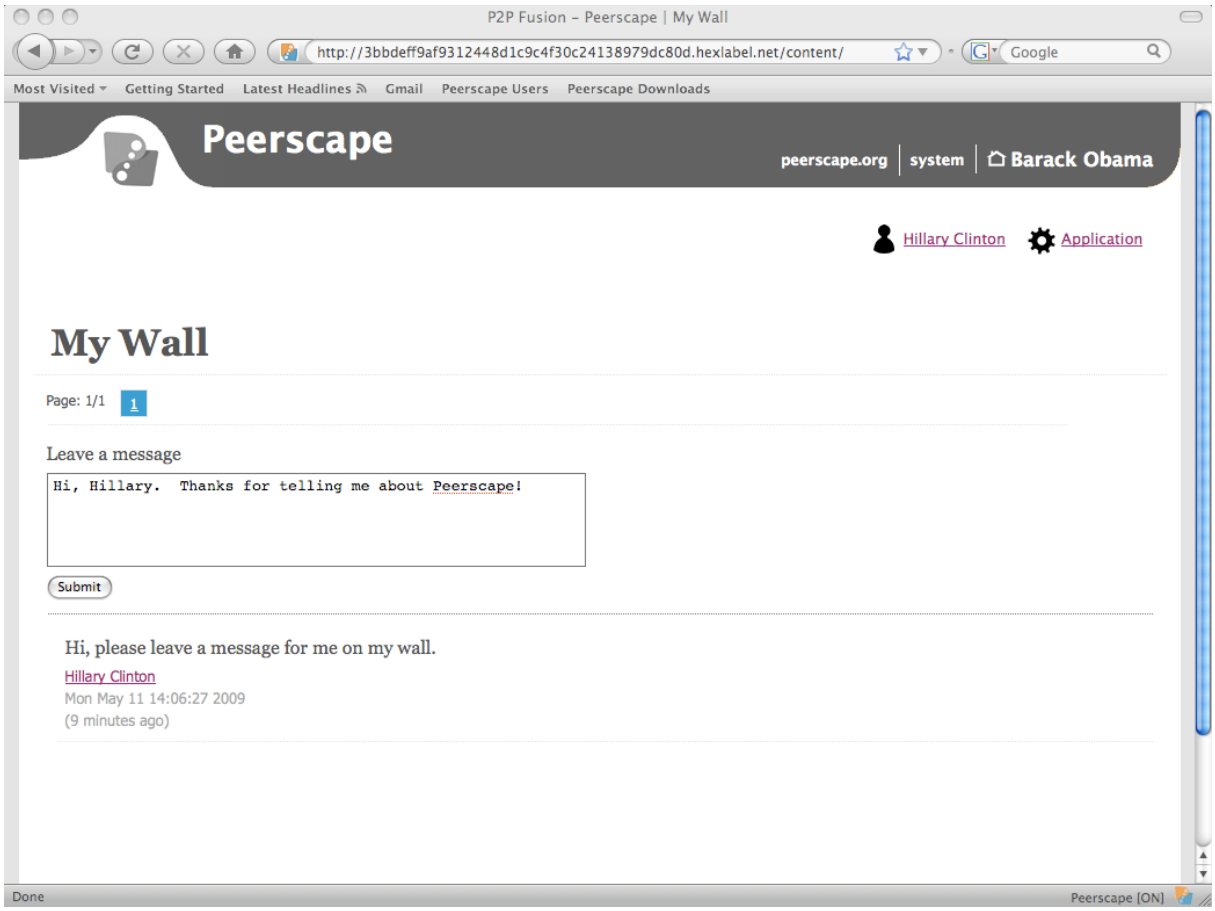


Figure 3.17 Η διεπαφή χρήστη του τοίχου μηνυμάτων του Peerscape

Συγκρίνοντας τις υπηρεσίες που προσφέρει το Peerscape με τις υπηρεσίες των online κοινωνικών δικτύων παρατηρούμε ότι, το Peerscape υποστηρίζει μόνο την ανταλλαγή ασύγχρονων μηνυμάτων και την μεταφορά αρχείων μεταξύ των χρηστών του. Ένα μειονέκτημα του Peerscape είναι ότι δεν εκτελείται από άλλους φυλλομετρητές έκτος του Firefox. Επιπρόσθετα, ένα σημαντικό σημείο που αξίζει να αναφέρουμε για το Peerscape είναι, ότι η ανάπτυξη του έχει σταματήσει από το 2009, όπου πλέον δεν υποστηρίζεται ούτε από την νέα έκδοση του Firefox 15.x

3.2 Βάσειςδεδομένων

Η βάση δεδομένων είναι μια αυστηρά τυποποιημένη συλλογή από σχετικά μεταξύ τους δεδομένα, που παρέχει έναν αυτόματο, κεντρικό και κοινό τρόπο χειρισμού τους. Ένας άλλος ορισμός είναι ότι μια βάση δεδομένων είναι ένα ολοκληρωμένο σύστημα που αποτελείται από δεδομένα (data) και από το κατάλληλο λογισμικό (software), τα οποία χρησιμοποιώντας το υλικό (hardware) βοηθούν στην ενημέρωση και πληροφόρηση των χρηστών (users).

Ένα πρόγραμμα που διαχειρίζεται βάσεις δεδομένων αποκαλείται Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS, DataBase Management System) και με τη βοήθειά του μπορούμε να αποθηκεύσουμε, προσθέσουμε, τροποποιήσουμε, εμφανίσουμε ή και να διαγράψουμε τα αποθηκευμένα δεδομένα. Τα δεδομένα που υπάρχουν στις βάσεις δεδομένων πρέπει να είναι:

Ολοκληρωμένα (Integrated), δηλ. τα δεδομένα πρέπει να είναι αποθηκευμένα σε ομοιόμορφα οργανωμένα σύνολα αρχείων όπου δεν πρέπει να υπάρχει επανάληψη ή πλεονασμός (redundancy) των ίδιων στοιχείων.

Καταμεριζόμενα (Shared), δηλ. να μπορούν περισσότεροι του ενός χρήστες να βλέπουν και να μοιράζονται τα ίδια δεδομένα την ίδια χρονική στιγμή.

Το DBMS, είναι ένα σύνολο από προγράμματα που επιτρέπουν τον χειρισμό των δεδομένων μιας ή περισσοτέρων βάσεων δεδομένων που ανήκουν στο ίδιο σύστημα. Το DBMS περιέχει κάποια εργαλεία γενικής χρήσης για να μπορούμε να δημιουργούμε και να χειριζόμαστε τα δεδομένα. Στα νεώτερα DBMS, όπως είναι η Oracle και η Informix, μπορούμε να έχουμε άμεση πληροφόρηση χωρίς να απαιτείται η παρουσία ενός προγραμματιστή. Τα δεδομένα ενός DBMS μπορούν να χρησιμοποιηθούν σε κάθε μορφής ερώτημα (query) για να πάρουμε τις πληροφορίες που επιθυμούμε.

Οι βάσεις δεδομένων για πολλά χρόνια βασιζόταν στις σχεσιακές βάσεις δεδομένων (relational database systems). Στον κόσμο του υπολογιστική νέφους (cloud computing), ένα από τα πιο σημαντικά μέρη του συστήματος είναι η βάση δεδομένων που πρέπει να χειρίζεται ένα πολύ μεγάλο αριθμό χρηστών. Κατανεμημένοι μηχανισμοί αποθήκευσης άρχισαν να

γίνονται η τυποποιημένη μέθοδος για την αποθήκευση δεδομένων για τη νέα γενιά εφαρμογών από εταιρίες κολοσσούς όπως οι:

- ***Google***
- ***Facebook***
- ***Amazon***
- ***Canonical***
- ***Yahoo***

Οι εταιρίες αυτές χειρίζονται ένα μεγάλο αριθμό από δεδομένα που φτάνει στην τάξη των petabyte. Ο βασικός περιορισμός των σχεσιακών βάσεων δεδομένων αποτελεί δυσκολία για κλιμάκωση σε εφαρμογές όπως η εξόρυξη δεδομένων, το Web 2.0, εφαρμογές του cloud computing και μη γραμμικά σε εκτέλεση ερωτήματα. Παρόλο που οι σχεσιακές βάσεις δεδομένων δίνουν στους χρήστες τη δυνατότητα για ένα βέλτιστο συνδυασμό απλότητας, ευελιξίας, απόδοσης, κλιμακωσιμότητας και συμβατότητας, οι μη σχεσιακές βάσεις δεδομένων παρέχουν στους χρήστες κατανεμημένες και ανοικτού τύπου εφαρμογές που είναι οριζόντια κλιμακώσιμες.

Τα κύρια χαρακτηριστικά των μη σχεσιακών βάσεων δεδομένων είναι ότι δεν έχουν σχήμα, δεν είναι σχεσιακές, έχουν μία απλή Διεπαφή Προγραμματισμού Εφαρμογών (API, Application Programming Interface) και είναι συνεπείς. Διεπαφή Προγραμματισμού Εφαρμογών καλούμε τη διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή και ανταλλαγή δεδομένων.

3.2.1 Σχεσιακές βάσεις δεδομένων

Μία σχεσιακή βάση δεδομένων είναι μία δομή δεδομένων που επιτρέπει στο χρήστη να συνδέσει τις πληροφορίες από διαφορετικούς πίνακες ή διαφορετικούς τύπους πληροφοριών. Ένας πίνακας πρέπει να περιέχει ένα κλειδί για να επιτρέπει να περιγραφεί μοναδικά ένα στιγμιότυπο του πίνακα. Άλλοι πίνακες μπορούν να αναφερθούν σε αυτό το κλειδί για να κάνουν μία σύνδεση μεταξύ των στιγμιότυπων τους και του στιγμιότυπου που δείχνει το κλειδί.

Στις σχεσιακές βάσεις δεδομένων, τα δεδομένα συνδέονται μεταξύ τους με σχέσεις (relations), οι οποίες προκύπτουν από τα κοινά πεδία που υπάρχουν σε διαφορετικά αρχεία. Τα αρχεία αποκαλούνται πίνακες (tables), οι εγγραφές γραμμές (rows) και τα πεδία στήλες (columns). Η ύπαρξη μιας κοινής τιμής στα πεδία δύο αρχείων καθορίζει και μια σχέση μεταξύ των γραμμών διαφορετικών πινάκων. Οι σχεσιακές βάσεις δεδομένων έχουν το πλεονέκτημα ότι είναι λογικά κατανοητές και πολύ ευέλικτες και δεκτικές σε αλλαγές. Το σχεσιακό μοντέλο περιγράφει τη βάση δεδομένων και οργανώνει τις εγγραφές με βάση τις σχέσεις. Για αυτό το λόγο η σχεδίαση μίας βάσης δεδομένων με τη χρήση διαγραμμάτων σχέσεων – οντοτήτων μπορεί να οδηγήσει εύκολα στην υλοποίησή της σε ένα σχεσιακό σύστημα.

Σήμερα, στην αγορά κυκλοφορούν αρκετά Συστήματα Διαχείρισης Βάσεων Δεδομένων βασισμένα στο σχεσιακό μοντέλο, τόσο μεγάλα συστήματα (όπως για παράδειγμα η Oracle, η

Ingress και η Sy–Base), όσο και μικρά συστήματα (όπως για παράδειγμα η Access). Ένας πίνακας μπορεί να χρησιμοποιηθεί τόσο για εγγραφές που περιγράφουν οντότητες, όσο και για εγγραφές που περιγράφουν σχέσεις. Έτσι, οι εγγραφές παρουσιάζονται στις γραμμές του πίνακα, ενώ σε κάθε στήλη του πίνακα υπάρχουν οι τιμές για ένα συγκεκριμένο πεδίο. Συνήθως, όταν γράφουμε πίνακες θεωρούμε ότι οι στήλες είναι διαταγμένες και κατά σύμβαση αναφέρουμε πρώτες τις στήλες που αντιστοιχούν σε κατηγορήματα κλειδιά. Αντίθετα, οι γραμμές του πίνακα συνήθως αναγράφονται μη διαταγμένες ή διαταγμένες με βάση το κλειδί.

Ο αριθμός των στηλών ενός πίνακα σε μία σχεσιακή βάση δεδομένων ονομάζεται βαθμός (arity) και ο αριθμός των γραμμών του ονομάζεται πληθικότητα (cardinality). Έτσι, ένας πίνακας που περιγράφει μία οντότητα με 3 κατηγορήματα θα έχει βαθμό 3. Αντίστοιχα, ένας πίνακας με πληθικότητα 500 σημαίνει ότι έχουν εισαχθεί σε αυτόν 500 εγγραφές. Οι βασικές πράξεις που μπορούν να γίνουν σε μία σχεσιακή βάση δεδομένων είναι οι ίδιες με αυτές που γίνονται σε κάθε βάση δεδομένων, δηλαδή Επιλογή, Εισαγωγή,

Ενημέρωση και Διαγραφή:

- Επιλογή σημαίνει αναζήτηση μίας συγκεκριμένης εγγραφής και η ανεύρεσή της μέσα στον πίνακα (δηλαδή ο εντοπισμός της γραμμής του πίνακα στην οποία βρίσκεται η ζητούμενη εγγραφή).
- Εισαγωγή σημαίνει η προσθήκη μίας νέας εγγραφής στον πίνακα. Η προσθήκη μπορεί να γίνει στο τέλος του πίνακα ή σε συγκεκριμένη θέση του πίνακα (στην περίπτωση που έχουμε διατάξει τον πίνακα).
- Ενημέρωση είναι η μεταβολή κάποιων τιμών για μερικά πεδία μίας εγγραφής (δηλαδή η αλλαγή των δεδομένων για μερικές στήλες σε μία γραμμή).
- Διαγραφή είναι το σβήσιμο μίας γραμμής από τον πίνακα.

3.2.2 Μη σχεσιακές βάσεις δεδομένων

Η νέα γενιά εφαρμογών, όπως το Web 2.0 και οι εφαρμογές κοινωνικής δικτύωσης, απαιτούν τον χειρισμό terabytes ή ακόμα και petabytes δεδομένων. Αυτό επιτυγχάνεται με κατανεμημένο χειρισμό. Αυτός είναι ένας σημαντικός λόγο για τη δύναμη των διαδικτυακών εταιριών, όπως οι Google, Amazon, Salesforce.com, Facebook, Twitterκαι Yahoo!. Οι σχεσιακές βάσεις δεδομένων είναι ακατάλληλες για κατανεμημένο χειρισμό που απαιτεί ένα πολύ μεγάλο αριθμό από εξυπηρετητές.

Υπάρχουν πολλοί λόγοι για τη χρησιμοποίηση του κατανεμημένοι χειρισμού. Από τη μία πλευρά τα προγράμματα πρέπει να είναι κλιμακώσιμα και να έχουν το πλεονέκτημα της χρησιμοποίησης πολλών υπολογιστικών συστημάτων από πολυπύρηνους επεξεργαστές. Από την άλλη πλευρά, οι εξυπηρετητές ιστοσελίδων πρέπει να κατανέμονται σε όλο τον κόσμο, ούτως ώστε να επιτυγχάνουν χαμηλή καθυστέρηση και μικρό ποσοστό αποτυχίας εξυπηρέτησης των πελατών. Τα Συστήματα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων έχουν ακόμα θέση στην αγορά και πρέπει να χρησιμοποιούνται σε αρκετές περιπτώσεις.

Ωστόσο, το Internet έχει αλλάξει απαιτήσεις αποθήκευσης από τα συστήματα βάσεων δεδομένων και οι σχεσιακές βάσεις δεδομένων δε μπορούν να ανταπεξέλθουν το ίδιο καλά με τις μη σχεσιακές. Μία μη σχεσιακή βάση δεδομένων αποθηκεύει απλά την πληροφορία χωρίς μηχανισμούς δόμησής της για να συσχετίσει τα δεδομένα από διαφορετικούς πίνακες. Οι μη σχεσιακές βάσεις δεδομένων, αναφέρονται αλλιώς και ως NoSQL βάσεις δεδομένων και έχουν πολλές μορφές. Όλο το σύστημα της βάσης δεδομένων βασίζεται στο θεωρία CAP που βγαίνει από τα εξής αρχικά:

- ***Consistency (Συνέπεια)***
- ***Availability (Διαθεσιμότητα)***
- ***Partition Tolerance (Ανοχή τμημάτων).***

Η συνέπεια μπορεί να περιγραφεί από την εξής πρόταση: «Όλοι οι πελάτες έχουν πάντα την ίδια όψη για τα δεδομένα». Η διαθεσιμότητα μπορεί να περιγραφεί από την εξής πρόταση: «Κάθε πελάτης μπορεί πάντα να διαβάζει και να γράφει στη βάση δεδομένων».

Τέλος, η ανοχή τμημάτων μπορεί να περιγραφεί από την εξής πρόταση: «Το σύστημα δουλεύει καλά ανεξαρτήτως των φυσικών δικτυακών τμημάτων». Για κάθε σύστημα βάσης δεδομένων, πρέπει να επιλεγούν δύο μόνο από αυτά τα χαρακτηριστικά. Τα υπάρχουσα συστήματα σχεσιακών βάσεων δεδομένων επιλέγουν τη συνέπεια και τη διαθεσιμότητα, αλλά δε μπορούν να έχουν την ανοχή τμημάτων. Οπότε, οι μη σχεσιακές βάσεις δεδομένων μπορούν να έχουν την ανοχή τμημάτων, αλλά πρέπει να θυσιάσουν είτε τη διαθεσιμότητα είτε τη συνέπεια. Για να μπορεί να λειτουργήσει μία μη σχεσιακή βάση δεδομένων παίρνει όσα χρειάζεται για να λειτουργήσει.

Τα σημαντικότερα χαρακτηριστικά των μη σχεσιακών βάσεων δεδομένων:

- είναι οριζόντια και κατακόρυφα κλιμακώσιμα,
- έχουν ζευγάρια κλειδιών-τιμών,
- αποθηκεύονται στη μνήμη και στο δίσκο και
- είναι προσανατολισμένες στο έγγραφο.

Τύποι μη σχεσιακών βάσεων δεδομένων Σε ένα βασικό επίπεδο υπάρχουν τρεις κύριες κατηγορίες για τις μη σχεσιακές βάσεις δεδομένων (21) (22):

- ***Key-value Stores (Αποθήκες κλειδιών-τιμών):*** Τα δεδομένα αποθηκεύονται ως ζευγάρια κλειδιών-τιμών, ούτως ώστε οι τιμές να είναι ευρετηριασμένες από τα αντίστοιχα κλειδιά. Αυτά τα συστήματα μπορούν να αποθηκεύουν δομημένη και μη δομημένη πληροφορία. Ένα παράδειγμα τέτοιου συστήματος είναι το Amazon's SimpleDB.
- ***Column-oriented Databases (Βάσεις δεδομένων προσανατολισμένες στη στήλη):*** Αυτοί οι τύποι των βάσεων δεδομένων περιέχουν μία επεκτάσιμη στήλη από πολύ κοντινά συσχετιζόμενη πληροφορία, παρά σύνολα πληροφοριών σε μία αυστηρή δομή πινάκων από στήλες και γραμμές, όπως μπορούν να βρεθούν στις σχεσιακές βάσεις δεδομένων. Παραδείγματα τέτοιων συστημάτων βάσεων δεδομένων είναι τα BigTable, Cassandra και HBase.
- ***Document-based Stores (Αποθήκες βασισμένες στα έγγραφα):*** Η πληροφορία αποθηκεύεται και οργανώνεται ως μία συλλογή από δεδομένα. Οι χρήστες

επιτρέπεται να προσθέτουν οποιοδήποτε αριθμό πεδίων οποιουδήποτε μεγέθους στο έγγραφο. Τα συστήματα αυτά τείνουν τα αποθηκεύουν τα έγγραφα σύμφωνα με το πρότυπο JSON. Παράδειγμα τέτοιων συστημάτων είναι η CouchDB και η MongoDB.

3.2.2.1 Χαρακτηριστικά μη σχεσιακών βάσεων δεδομένων

Τα σημαντικότερα χαρακτηριστικά των μη σχεσιακών βάσεων δεδομένων είναι τα παρακάτω [1]:

- ***Θεώρημα CAP*** (23)***:*** Τα συστήματα μη σχεσιακών βάσεων δεδομένων επιλέγουν δύο από τις τρεις αρχές του θεωρήματος CAP (Συνέπεια, Διαθεσιμότητα και Ανοχή τμημάτων). Πολλά συστήματα μη σχεσιακών βάσεων δεδομένων έχουν χαλαρώσει τις απαιτήσεις για τη συνέπεια, ούτως ώστε να πετύχουν καλύτερη διαθεσιμότητα και ανοχή τμημάτων. Αυτόοδήγησεσταγνωστάσυστήματα BASE (Basically available, Soft- state, Eventually consistent).
- ***Οριζόντια και κατακόρυφα κλιμακώσιμα:*** Παραδοσιακά οι σχεσιακές βάσεις δεδομένων φιλοξενούνται στον εξυπηρετητή, που μπορεί να κλιμακωθεί προσθέτοντας περισσότερους επεξεργαστές, μνήμη και σκληρούς δίσκους. Οι μη σχεσιακές βάσεις δεδομένων βρίσκονται σε πολλαπλούς εξυπηρετητές που συνήθως χρησιμοποιούν μεθόδους αντιγραφής για να κρατούν τις βάσεις δεδομένων συγχρονισμένες. Οι μη σχεσιακές βάσεις δεδομένων μπορούν να υπάρχουν σε ένα μόνο εξυπηρετητή, αλλά συνήθως σχεδιάζονται για να υπάρχουν σε μία ομάδα εξυπηρετητών .
- ***Ζευγάρια κλειδιών-τιμών:*** Οι σχεσιακές βάσεις δεδομένων προκύπτουν συνήθως από μία συλλογή στηλών σε ένα πίνακα ή/και όψη (τυποποιημένο σχήμα, λειτουργίες σύζευξης). Οι μη σχεσιακές βάσεις δεδομένων αποθηκεύουν τον συνδυασμό του κλειδιού και της τιμής (χωρίς σχήμα).
- ***Στη μνήμη, στο δίσκο:*** Οι σχεσιακές βάσεις δεδομένων σχεδόν πάντα υπάρχουν στον σκληρό δίσκο ή σε μία δικτυακή συσκευή αποθήκευσης. Σύνολα από γραμμές της βάσης δεδομένων στέλνονται στη μνήμη ως μέρος ερωτημάτων SQL ή αποθηκευμένων ρουτινών. Μερικές από τις μη σχεσιακές βάσεις δεδομένων σχεδιάζονται για να αποθηκεύονται στη μνήμη για μεγαλύτερη ταχύτητα και μπορούν να αντιγραφούν στο δίσκο.
- ***Προσανατολισμένες στο έγγραφο:*** Οι προσανατολισμένες στο έγγραφο βάσεις δεδομένων περιέχουν εγγραφές από δεδομένα, πεδία και XML. Οι σχεσιακές βάσεις δεδομένων χαρακτηρίζονται από δεδομένα που οργανώνονται σε πίνακες, γραμμές και στήλες. Τα ερωτήματα ανάκτησης πληροφορίας της SQL συνήθως επιστρέφουν δείκτες σε μία μόνο γραμμή ή σε ένα σύνολο γραμμών που περιέχουν τις στήλες που καθορίζονται από το ερώτημα. Οι μη σχεσιακές βάσεις δεδομένων έχουν λειτουργίες που συχνά γίνονται μέσω κώδικα ή κάποιας διεπαφής.

3.2.2.2 Συστήματα διαχείρισης μη σχεσιακών βάσεων δεδομένων

Τα τελευταία χρόνια έχουν δημιουργηθεί πολλά μη σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων. Τα πιο σημαντικά από αυτά είναι τα:

- ***Amazon SimpleDB***
- ***Apache CouchDB***
- ***MongoDB***
- ***Cassandra***
- ***Hbase***

Τα συστήματα αυτά αναλύονται στις παρακάτω ενότητες, και ακολουθεί μία ανάλυση του μοντέλου δεδομένων και της αρχιτεκτονικής των λογισμικών αυτών. Το σύστημα διαχείρισης Couchdb που είναι και αυτό που πληρεί τις περισσότερες απαιτήσεις του συστήματος θα αναλυθεί σε επόμενο κεφάλαιο.

3.2.2.2.1 Cassandra

Το σύστημα Cassandra (24) είναι ένα πλήρες σύστημα κατανεμημένης βάσης δεδομένων. Το Cassandra προσπαθεί να πάρει το βέλτιστο από δύο άλλα συστήματα μη σχεσιακών βάσεων δεδομένων, τα BigTable και Dynamo για τη δημιουργία ενός νέου κατανεμημένου μηχανισμού διαχείρισης βάσεων δεδομένων. Είναι ανθεκτικό στα λάθη, σταθερό και μπορεί να χρησιμοποιηθεί σε ένα Hadoop (25)cluster για την εκτέλεση MapReduce εργασιών.

Ένα Hadoop cluster είναι ένα cluster υπολογιστών που χρησιμοποιείται για την εκτέλεση MapReduce εργασιών. Όλοι οι υπολογιστές που συμμετέχουν στο συγκεκριμένο cluster είναι προφανώς συνδεδεμένοι μεταξύ τους μέσω δικτύου, και έχουν όλοι τους εγκατεστημένη μια έκδοση του Hadoop. Το Hadoop υποστηρίζει την κατανεμημένη επεξεργασία μεγάλου όγκου δεδομένων και έχει βασιστεί στο Google Map Reduce Frameworkκαι το GoogleFileSystem. Το μοντέλο δεδομένων του Cassandra βασίζεται στις παρακάτω αρχές:

- ***Keyspace:*** ένας χώρος ονομάτων για ένα σύνολο πεδίων.
- ***ColumnFamilies:*** σύνολα από πεδία.
- ***SuperColumns:*** πεδία που μπορούν και τα ίδια να περιέχουν άλλα πεδία.
- ***Πεδία:*** ορίζονται από ένα όνομα, μία τιμή και μία χρονοσφραγίδα.

Συνήθως υπάρχει ένα χώρος ονομάτων για κάθε εφαρμογή που χρησιμοποιεί το Cassandra. Η παραμετροποίηση του χώρου ονομάτων είναι πολύ σημαντική, γιατί είναι ο μηχανισμός που καθορίζει τους παράγοντες αντιγραφής και τη στρατηγική αντιγραφής. Τα σύνολα πεδίων μπορεί κάποιος να τα φανταστεί σαν τους πίνακες στα παραδοσιακά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων, που ορίζονται από στατικά αρχεία παραμετροποίησης και δε μπορούν να αλλάξουν κατά τη διάρκεια εκτέλεσης του cluster.

Νέα δεδομένα προστίθενται με τη μορφή μίας γραμμής, όπως στις σχεσιακές βάσεις δεδομένων, εκτός από το ότι η στήλη μίας γραμμής μπορεί να είναι διαφορετική ανά γραμμή.

Κάθε γραμμή αναγνωρίζεται από ένα μοναδικό κλειδί και περιέχει ένα ή περισσότερα σύνολα πεδίων, που και τα ίδια περιέχουν άλλα πεδία.

Το μοντέλο ερωτημάτων που παρέχει το Cassandra επιτρέπει στον χρήστη να έχει απευθείας πρόσβαση (και για λειτουργίες εγγραφής και ανάγνωσης) σε μία συγκεκριμένη στήλη μίας συγκεκριμένης γραμμής. Επίσης, είναι πιθανή η ανάκτηση όλης της γραμμής.

Επιπλέον, μπορεί ο χρήστης να ορίσει το συντελεστή τμήματος (Slice Predicate) που είναι παρόμοιος με τον συντελεστή στα μαθηματικά, και περιγράφεται από μία ιδιότητα που τα στοιχεία του συνόλου έχουν κοινή. Ο συντελεστής τμήματος είναι χρήσιμος όταν ένα υποσύνολο κάποιων πεδίων χρειάζεται και μπορεί να οριστεί με δύο τρόπους:

- Με μία λίστα ονομάτων πεδίων.
- Με ένα διάστημα τμήματος που περιγράφει το διάστημα, την ταξινόμηση ή/και τον περιορισμό του τμήματος.

Τέλος, αν το Cassandra έχει πλήρη πρόσβαση σε ένα λειτουργικό Hadoop cluster, τότε οι λειτουργίες Hadoop MapReduce μπορούν να χρησιμοποιηθούν για να γίνουν πιο σύνθετοι υπολογισμοί στα δεδομένα που αποθηκεύονται στο Cassandra.

3.2.2.2.1.1 Η αρχιτεκτονική του Cassandra

Το Cassandra cluster είναι μία συλλογή από κόμβους. Επιλέγεται ένας κόμβος – ηγέτης από το ZooKeeper, ενώ οι άλλοι κόμβοι είναι υπεύθυνοι για την αποθήκευση των δεδομένων. Ο κόμβος – ηγέτης αναθέτει τμήματα στους κόμβους αποθήκευσης. Τα δεδομένα χωρίζονται σε κατατμήσεις με έναν μηχανισμό ευρετηριοποίησης. Κάθε κόμβος αναγνωρίζεται με έναν αριθμό που αντιπροσωπεύει τη θέση του στο δακτύλιο που δημιουργείται. Κάθε αντικείμενο δεδομένων ανατίθεται σε έναν κόμβο του δακτυλίου με την ευρετηριοποίηση του και μεταφέρεται κυκλικά στο δακτύλιο σύμφωνα με τη φορά του ρολογιού μέχρι να βρει έναν κόμβο με μεγαλύτερη θέση. Κάθε κόμβος είναι συνυπεύθυνος για οποιοδήποτε αντικείμενο δεδομένων έρθει στο δακτύλιο μεταξύ του κόμβου του ίδιου και του προηγούμενου κόμβου. Ο συνυπεύθυνος κόμβος αποθηκεύει ένα αντίτυπο του αντικειμένου και το αντιγράφει σε διαφορετικούς κόμβους μέσα στο δακτύλιο. Το αντικείμενο αντιγράφεται σε ένα πλήθος κόμβων ανάλογα με τον συντελεστή αντιγραφής N και τη στρατηγική αντιγραφής. Ο συντελεστής αντιγραφής ορίζεται από το χρήστη. Το Cassandra υποστηρίζει τρεις στρατηγικές τοποθέτησης:

- ***Simple Strategy*** (Απλή στρατηγική)
- ***Old Network Topology Strategy*** (Στρατηγική παλιάς τοπολογίας δικτύου)
- ***Network Topology Strategy*** (Στρατηγική τοπολογίας δικτύου)

Ο κόμβος – ηγέτης είναι υπεύθυνος για την ισορρόπηση του φορτίου στους κόμβους, ούτως ώστε κάθε κόμβος να μην είναι υπεύθυνος για παραπάνω από N – 1 αντικείμενα. Οι μεταπληροφορίες αποθηκεύονται και αντιγράφονται στους κόμβους. Μία αίτηση διαβάσματος ή εγγραφής απαιτεί την ανίχνευση του κόμβου στη διαδρομή που έχει το κατάλληλο αντικείμενο. Για τις λειτουργίες εγγραφής, το σύστημα δρομολογεί την ανανέωση

όλων των αντιγράφων της πληροφορίας στους κόμβους που έχει ανατεθεί και περιμένει από έναν ελάχιστο αριθμό κόμβων να επιβεβαιώσουν την επιτυχία της εγγραφής.

3.2.2.2.2 SimpleDB

Το Amazon SimpleDB (26)είναι ένα κλιμακώσιμο και ευέλικτο σύστημα μη σχεσιακών βάσεων δεδομένων, που διευκολύνει την αποθήκευση αντικειμένων πληροφορίας αλλά και την επερώτηση στα αντικείμενα μέσω διαδικτυακών υπηρεσιών. Το Amazon SimpleDB δεν απαιτεί να υπάρχει σχήμα στις βάσεις δεδομένων, διαθέτει αυτόματη ευρετηριοποίηση και παρέχει μία απλή διεπαφή για την αποθήκευση και πρόσβαση στην πληροφορία. Τα δεδομένα είναι προσβάσιμα από το http μέσω των πρωτοκόλλων REST και SOAP. Το SimpleDB αποτελείται από πολλαπλές περιοχές διευθύνσεων και κάθε περιοχή διευθύνσεων αποθηκεύει ένα σύνολο από εγγραφές. Κάθε εγγραφή έχει ένα μοναδικό κλειδί και ένα σύνολο από ιδιότητες που μπορεί να υπάρχουν σε κάθε εγγραφή, αλλά μπορεί και να μην υπάρχουν. Μέσα σε κάθε περιοχή διευθύνσεων, οι εγγραφές ευρετηριοποιούνται αυτόματα για κάθε ιδιότητα. Οι κύριες λειτουργίες είναι εγγραφή/ανάκτηση μίας εγγραφής από κλειδί ή η αναζήτηση ενός συνόλου εγγραφών ανά μία ή πολλών ιδιοτήτων.

3.2.2.2.2.1 Το μοντέλο δεδομένων του SimpleDB

Τα δεδομένα αποθηκεύονται σε περιοχές διευθύνσεων που καθορίζονται από το όνομά τους. Το μέγεθος της περιοχής διευθύνσεων δε μπορεί να ξεπερνά τα 10 GB. Οι περιοχές διευθύνσεων απαρτίζονται από αντικείμενα. Δεν είναι πιθανό κανονικά να υπάρχουν πάνω από 100 περιοχές διευθύνσεων ανά λογαριασμό. Ένα αντικείμενο απαρτίζεται από το όνομα του αντικειμένου που είναι μοναδικό μέσα στην περιοχή διευθύνσεων και από διάφορες ιδιότητες. Μία ιδιότητα απαρτίζεται από το όνομα της ιδιότητας και μία ή περισσότερες τιμές. Οι τιμές των ιδιοτήτων μπορούν να είναι μόνο αλφαριθμητικά. Τα αντικείμενα σε μία περιοχή διευθύνσεων μπορούν να έχουν διαφορετικά ονόματα πεδίων. Το SimpleDB αφήνει το χρήστη να οργανώσει τα δεδομένα σε περιοχές διευθύνσεων που μπορούν να παραλληλιστούν με τους πίνακες στις σχεσιακές βάσεις δεδομένων, με τη διαφορά ότι μία περιοχή διευθύνσεων μπορεί να περιέχει ένα διαφορετικό σύνολο ιδιοτήτων για κάθε αντικείμενο. Όλες οι ιδιότητες είναι πίνακες από bytes με μέγιστο μέγεθος τα 1024 bytes. Κάθε αντικείμενο μπορεί να έχει διαφορετικές τιμές για κάθε ιδιότητα. Λόγω των περιορισμών στο μοντέλο ερωτημάτων, είναι πιθανό να μοντελοποιούνται σχέσεις μεταξύ των αντικειμένων στο SimpleDB χωρίς τη δημιουργία περιττής πληροφορίας. Άρα ο δημιουργός μπορεί να αποκανονικοποιεί τα δεδομένα τους ή να διαχειρίζεται τις σχέσεις στο επίπεδο εφαρμογής. Τα ερωτήματα μπορούν να γίνουν μέσα σε μία περιοχή διευθύνσεων, άρα αν ένας χρήστης θέλει να συνενώσει τα δεδομένα από διαφορετικές περιοχές διευθύνσεων, πρέπει αυτό να γίνει στο επίπεδο της εφαρμογής. Δε θα ήταν φρόνιμο να εισαχθεί όλη η πληροφορία σε μία περιοχή διευθύνσεων, γιατί οι περιοχές διευθύνσεων στο SimpleDB χρησιμοποιούνται για την καλύτερη κατανομή των δεδομένων. Επίσης, η

ταχύτητα εκτέλεσης του ερωτήματος σχετίζεται άμεσα με το μέγεθος κάθε περιοχής διευθύνσεων.

3.2.2.2.2.2 Η αρχιτεκτονική του SimpleDB

Το SimpleDB βασίζεται στο Amazon S3 (Simple Storage Service), στο οποίο χορηγείται στους χρήστες απεριόριστος χώρος σε πολύ χαμηλές χρεώσεις. Τα δεδομένα στο σύστημα S3 αποθηκεύονται σε έναν αριθμό από εξυπηρετητές ή αποθηκευτικές συσκευές στο Amazon, κάτι που κάνει την πλατφόρμα κατανεμημένη και εύκολα κλιμακώσιμη. Το SimpleDB και το S3 είναι επεκτάσεις της αρχιτεκτονικής νέφους υπολογιστών, στην οποία υπολογιστικοί πόροι, λογισμικές εφαρμογές και δεδομένα μοιράζονται διαμέσου του διαδικτύου. Οι λογισμικοί πόροι, όπως οι εφαρμογές και τα δεδομένα, αποθηκεύονται σε κατανεμημένους εξυπηρετητές, ούτως ώστε όταν ένας χρήστης απαιτεί χρήση της εφαρμογής επεξεργασίας κειμένου, το στιγμιότυπο αυτό να παρέχεται στον χρήστη μέσω ενός φυλλομετρητή.

3.2.2.2.3 MongoDB

Το MongoDB (27)είναι ένα αντικειμενοστραφές σύστημα μη σχεσιακών βάσεων δεδομένων που αναπτύχθηκε από την 10gen και είναι ένα λογισμικό ανοικτής αρχιτεκτονικής. Το όνομα MongoDB προέρχεται από τη λέξη «humongous». Η βάση δεδομένων προορίζεται να είναι κλιμακώσιμη και γρήγορη και είναι υλοποιημένη σε C++. Επιπλέον στην αντικειμενοστρέφεια του συστήματος αυτού, το MongoDB μπορεί να χρησιμοποιηθεί για την αποθήκευση και κατάτμηση μεγάλων δυαδικών αρχείων, όπως εικόνων και βίντεο. Το σύστημα είναι σταθερό και ανεκτικό στα λάθη, ενώ παρέχει μία πολύπλοκη γλώσσα ερωτημάτων όπως και μία υλοποίηση του MapReduce.

3.2.2.2.3.1 Το μοντέλο δεδομένων του MongoDB

Το MongoDB αποθηκεύει έγγραφα με τη μορφή των BSON αντικειμένων, που είναι δυαδικά κωδικοποιημένα JSON αντικείμενα. Η μορφοποίηση BSON υποστηρίζει εμφωλευμένες δομές αντικειμένων με ενσωματωμένα αντικείμενα και πίνακες, όπως κάνει και το JSON. Το MongoDB υποστηρίζει αλλαγές στις ιδιότητες, άρα αν μία μόνο ιδιότητα αλλάξει στην εφαρμογή, τότε μόνο αυτή στέλνεται πίσω στη βάση δεδομένων. Κάθε έγγραφο έχει ένα πεδίο ID, που χρησιμοποιείται ως το πρωτεύων κλειδί. Για την υποστήριξη γρήγορων ερωτημάτων, ο σχεδιαστής μπορεί να δημιουργήσει ένα ευρετήριο για ένα πεδίο ενός εγγράφου που θα δεχθεί ερωτήσεις. Το MongoDB υποστηρίζει επίσης, την ευρετηριοποίηση ενσωματωμένων αντικειμένων και πινάκων.

Το MongoDB έχει ένα ειδικό γνώρισμα για τους πίνακες που λέγεται «πολλαπλά κλειδιά». Το γνώρισμα αυτό επιτρέπει τη χρησιμοποίηση ενός πίνακα ως ευρετήριο, που μπορεί για παράδειγμα να περιέχει λέξεις κλειδιά για ένα έγγραφο. Με ένα τέτοιο ευρετήριο, τα έγγραφα μπορούν να αναζητούνται σύμφωνα με τις λέξεις κλειδιά. Τα έγγραφα στο MongoDB μπορούν να οργανωθούν σε συλλογές. Κάθε συλλογή μπορεί να περιέχει οποιοδήποτε τύπο εγγράφων, αλλά τα ερωτήματα και τα ευρετήρια μπορούν να γίνουν μόνο μέσα στις συλλογές. Επειδή το MongoDB έχε ένα περιορισμό ευρετηρίων ανά συλλογή, είναι προτιμότερη η χρησιμοποίηση μίας συλλογής για κάθε τύπο εγγράφου, ούτως ώστε να έχουν τα ερωτήματα καλύτερη απόδοση. Οι σχέσεις στο MongoDB μπορεί να απεικονιστεί μέσω ενσωματωμένων αντικειμένων και πινάκων. Οπότε το μοντέλο δεδομένων πρέπει να έχει τη μορφή δένδρου. Η πρώτη επιλογή υπονοεί ότι μερικά έγγραφα πρέπει να υπάρχουν σε διπλότυπα μέσα στη βάση δεδομένων. Αυτή η λύση πρέπει να χρησιμοποιείται μόνο αν τα διπλότυπα έγγραφα δε χρειάζονται συχνές ανανεώσεις. Η δεύτερη επιλογή είναι η χρησιμοποίηση συζεύξεων στην πλευρά των πελατών που μπορούν να τοποθετηθούν σε μορφή δένδρου. Αυτό απαιτεί περισσότερη δουλειά στο επίπεδο εφαρμογής και αυξάνει την διαδικτυακή επικοινωνία με τη βάση δεδομένων.

3.2.2.2.3.2 Η αρχιτεκτονική του MongoDB

Ένα cluster του MongoDB απαρτίζεται από τρία συστατικά:

- **Κόμβοι Shard**
  Οι κόμβοι shard είναι υπεύθυνοι για την αποθήκευση των πραγματικών δεδομένων. Κάθε τέτοιος κόμβος μπορεί να απαρτίζεται είτε από έναν κόμβο είτε από ένα ζευγάρι κόμβων που έχουν την ίδια πληροφορία. Σε μελλοντικές εκδόσεις του MongoDB, ένας κόμβος shard μπορεί να απαρτίζεται από περισσότερους από δύο κόμβους για καλύτερη απόδοση.
- **Εξυπηρετητές παραμετροποίησης**
  Οι εξυπηρετητές παραμετροποίησης χρησιμοποιούνται για να αποθηκεύουν τις μεταληροφορίες και την πληροφορία δρομολόγησης του cluster του MongoDB και είναι προσβάσιμα μέσω των κόμβων shard και των υπηρεσιών δρομολόγησης
- **Υπηρεσίες δρομολόγησης ή mongos**
  Τα mongos είναι υπεύθυνα για την απόδοση των διεργασιών που αιτούνται οι χρήστες. Ανάλογα με τον τύπο των λειτουργιών, τα mongos στέλνουν αιτήσεις στους απαραίτητους κόμβους shard και συνενώνουν τα αποτελέσματα πριν αυτά επιστρέψουν στον πελάτη. Τα mongos μπορούν να εκτελούνται παράλληλα.

Το MongoDB είναι υλοποιημένο με τη γλώσσα προγραμματισμού C++ και αποτελείται από δύο τύπους υπηρεσιών:

- τον πυρήνα της βάσης δεδομένων που καλείται mongod.
- τις υπηρεσίες mongos.

Το MongoDB για την αποθήκευση χρησιμοποιεί αρχεία που έχουν χαρτογραφηθεί στη μνήμη, και αφήνει στον διαχειριστή εικονικής μνήμης του λειτουργικού συστήματος να

αποφασίσει ποια μέρη της βάσης δεδομένων θα αποθηκευτούν στη μνήμη και ποια θα μείνουν στον σκληρό δίσκο. Το κίνητρο για τη χρησιμοποίηση χαρτογραφημένων στη μνήμη αρχείων είναι η εκμετάλλευση της διαθέσιμης μνήμης όσο γίνεται περισσότερο, με απώτερο σκοπό την καλύτερη απόδοση του συστήματος βάσεων δεδομένων. Σε μερικές περιπτώσεις αυτό μπορεί να εξαλείψει την ανάγκη για ξεχωριστά επίπεδα κρυφής μνήμης στην πλευρά του πελάτη. Σε εξέλιξη βρίσκεται μία εναλλακτική αποθηκευτική μηχανή που θα επιτρέψει τον καλύτερο χειρισμό των εγγραφών/ανακτήσεων. Τα ευρετήρια αποθηκεύονται στο MongoDB ως B-trees όπως και στις περισσότερες βάσεις δεδομένων. Τα έγγραφα στο MongoDB cluster τμηματοποιούνται από τη δική τους συλλογή και από τις επιθυμίες του χρήστη, που καλείται κλειδί shard. Το κλειδί shard είναι παρόμοιο με ένα ευρετήριο και διαθέτει πολλαπλά πεδία. Αυτό το κλειδί shard χρησιμοποιείται στην κατάτμηση ολόκληρης της συλλογή μέσα στα shards. Κάθε shard αποθηκεύει τα ανατεθειμένα σε αυτό έγγραφα, ταξινομημένα σύμφωνα με αυτό το κλειδί. Τα έγγραφα μέσα σε ένα shard είναι οργανωμένα σε κομμάτια. Κάθε κομμάτι περιέχει ένα ταξινομημένο σύνολο εγγράφων από ένα αρχικό κλειδί shard μέχρι ένα συγκεκριμένο τελικό κλειδί shard. Αν ένα κομμάτι γίνει πολύ μεγάλο, τότε θα κατατμηθεί. Τα κομμάτια χρησιμοποιούνται από το MongoDB για την αυτόματη διαχείριση των κλειδιών shards. Αν το μέγεθος ενός shard γίνει μεγάλο, τότε μερικά από τα κομμάτια που περιέχει ανατίθενται σε άλλα shards. Τα κομμάτια χρησιμοποιούνται επίσης, στον επανα- χωρισμό των δεδομένων όταν νέοι κόμβοι προστίθενται ή διαγράφονται. Οι εξυπηρετητές παραμετροποίησης αποθηκεύουν μία εγγραφή σε κάθε κομμάτι σε ένα cluster, που περιέχει ένα αρχικό και ένα τελικό κλειδί του κομματιού και του ανατεθειμένου shard. Η πληροφορία αυτή χρησιμοποιείται από τα mongos για να αποφασίσουν ποιοι κόμβοι shard χρειάζονται σε κάθε αίτηση. Ανάλογα με τον τύπο της λειτουργίας μπορεί είτε ένας shard κόμβος είτε όλοι οι κοντινοί να χρειάζονται για να ικανοποιήσουν την αίτηση αυτή.

3.2.2.2.4 HBase

Το HBase (28) είναι ένα project ανοικτού κώδικα, γραμμένο σε Java και δημιουργημένο από το Apache software foundation. Το HBase λειτουργεί με το Apache's Hadoop Distributed File System (HDFS) ως βασική μονάδα αποθήκευσης. Έχει βελτιστοποιηθεί για υψηλή απόδοση σε πραγματικού χρόνου ερωτήματα που αναφέρονται σε μεγάλη ποσότητα κατανεμημένης πληροφορίας. Το Hbase παρέχει μία RESTful διαδικτυακή υπηρεσία που υποστηρίζει το XML. Παρέχει μία πραγματικού χρόνου, κατανεμημένη και δομημένη βάση δεδομένων πάνω από το κατανεμημένο σύστημα αρχείων Hadoop. Το σύστημα απαρτίζεται από δύο επίπεδα. Πρώτα από όλα το κατανεμημένο σύστημα αρχείων HDFS αποθηκεύει όλα τα δεδομένα και ευθύνεται για την αντιγραφή των δεδομένων σε διπλότυπα, την αποτυχία των κόμβων και την κατανομή των δεδομένων στους κόμβους. Το δεύτερο επίπεδο είναι αυτό που δημιουργήθηκε από το Hbase, όπου κάθε εξυπηρετητής περιοχής είναι υπεύθυνος για μία λίστα εξυπηρετητών, που σημαίνει ότι πρέπει να εγγράφει τις ανανεώσεις και τις εγγραφές σε πίνακες της μνήμης. Επίσης, ο εξυπηρετητής περιοχής λειτουργεί ως μία κρυφή μνήμη για τα

δεδομένα που αποθηκεύονται στο HDFS επίπεδο. Είναι πολύ ενδιαφέρουσα η περιγραφή ενός cluster HBase, επειδή είναι πολύ πιο περίπλοκη από κάθε σύστημα όπου όλοι οι κόμβοι είναι ίσοι.

3.2.2.2.4.1 Το μοντέλο δεδομένων του Hbase

Το μοντέλο δεδομένων του HBase είναι βασισμένο στα παρακάτω συστατικά:

- ***Πίνακες:*** αποτελούνται από γραμμές και στήλες.
- ***Στήλες:*** κάθε στήλη ανήκει σε μία δοθείσα οικογένεια στηλών.
- ***Γραμμές:*** κάθε γραμμή αναγνωρίζεται από το κλειδί της.
- ***Κελιά πινάκων:*** ένα κελί πινάκων είναι η τομή μίας γραμμής και μίας στήλης.

Οι πίνακες δημιουργούνται από τον ορισμό του σχήματος, αλλά ένας νέος πίνακας μπορεί εύκολα να προστεθεί κατά τη διάρκεια κανονικών λειτουργιών. Ο αριθμός των εκδοχών ορίζεται στο στάδιο της δημιουργίας της οικογένειας των στηλών. Το Hbase αποθηκεύει οτιδήποτε ως bytes, που σημαίνει ότι οτιδήποτε μπορεί να χρησιμοποιηθεί ως όνομα σε μία οικογένεια στηλών ή ως κλειδί σε μία γραμμή. Με την HBase, ο χρήστης μίας βάσης δεδομένων διαβάζει δεδομένα χρησιμοποιώντας είτε τις λειτουργίες Get είτε τις Scan. Οι λειτουργίες Get χρησιμοποιούνται για την ανάκτηση όλων των στηλών ή ενός υποσυνόλου από αυτές για μία δοθείσα γραμμή. Οι λειτουργίες Scan χρησιμοποιούνται για την ανάκτηση όλων ή ενός μέρους των γραμμών σε έναν πίνακα. Όπως οι λειτουργίες Get, το υποσύνολο των ζητούμενων στηλών μπορεί να καθοριστεί, αλλά είναι επίσης πιθανό να οριστεί ένας εύρος των γραμμών χρησιμοποιώντας κλειδιά γραμμών για την αρχή και το τέλος του εύρους. Οι πίνακες στο HBase μπορούν να χρησιμοποιηθούν είτε ως είσοδος είτε ως έξοδος στις εργασίες MapReduce του Hadoop, όπου περισσότερο σύνθετοι υπολογισμοί ή ανανεώσεις μπορούν να απαιτούνται. Ο υπολογισμός μπορεί να είναι κατανεμημένος ανάλογα με τον αριθμό των περιοχών που είναι διαθέσιμες στο cluster, το οποίο σημαίνει ότι ο αριθμός των λειτουργιών Map πρέπει πάντα να είναι ίσος με τον αριθμό των περιοχών.

3.2.2.2.4.2 Η αρχιτεκτονική του Hbase

Ένα HBase cluster είναι στην ουσία δύο ξεχωριστά cluster που δουλεύουν μαζί, συχνά στους ίδιους εξυπηρετητές, αλλά όχι απαραίτητα. Το HDFS cluster απαρτίζεται από ένα κόμβο που λειτουργεί ως το σημείο εισόδου στο cluster, δηλαδή είναι ο συγκεκριμένος κόμβος που γνωρίζει που βρίσκονται όλα τα δεδομένα και σε ποιο κόμβο. Το HBase cluster απαρτίζεται από έναν ηγέτη, ένα Zookeeper cluster που λειτουργεί ως το σημείο ελέγχου στο cluster και τους εξυπηρετητές περιοχής που έχουν τα δεδομένα. Οι ηγέτες και οι εξυπηρετητές περιοχής είναι εγγεγραμμένοι στο Zookeeper και αν ένας από αυτούς τερματίσει τη λειτουργία του, τότε επιδιορθώνεται ή αντικαθίσταται από το Zookeeper. Ο ηγέτης κάνει διαχειριστικές λειτουργίες στο παρασκήνιο, όπως το να κρατά το cluster ισορροπημένο και να μετακινεί περιοχές από κόμβους που αποτυγχάνουν.

4ο Κεφαλαιο

4.1 Εισαγωγή

Το λογική του εγχειρήματος που αναπτύσσουμε στη διπλωματική είναι να υλοποιήσουμε ένα κοινωνικό δίκτυο με πλήρως αποκεντρωμένη και ομότιμη μορφή. Παραπάνω έχει αναλυθεί το πώς οι κεντρικές δομές στο διαδίκτυο και ιδιαίτερα στα κοινωνικά δίκτυα όπου ένα μεγάλο μέρος του πληθυσμού εναποθέτει προσωπικές στιγμές και δεδομένα , οδηγούν σε μια μορφή επιτήρησης και εκμετάλλευσης των χρηστών είτε για χρηματικό κέρδος είτε για καταστολή των κοινωνικών αντιδράσεων. Παρακάτω θα αναλυθεί υπό το πρίσμα της ασφάλειας και της ιδιωτικότητας οι διαφορές μεταξύ των κεντρικοποιημένων σε αντίθεση με τις αποκεντρωτικές δομές όσον αφορά τα κοινωνικά δίκτυα. Αρχικά θα γίνει μια ανασκόπηση των διαφορετικών μορφών κατανεμημένων συστημάτων που υπάρχουν ξεκινώντας από τα αποκεντρωτικά δίκτυα που είναι εξαρτημένα από κεντρικούς εξυπηρετητές για το συντονισμό τους και θα καταλήξουμε σε πλήρως αποκεντρωμένα δίκτυα όπως το DHT.

4.2 Peer-to-peer συστήματα

Ο όρος συστήματα ομότιμων οντοτήτων (Peer-to-Peer, p2p) δημιουργήθηκε για να περιγράψει αυτό-οργανώσιμα, κατανεμημένα, ιδεατά δίκτυα για την από κοινού χρήση πόρων και την εκμετάλλευση της τεράστιας ποσότητας που μένει αχρησιμοποίητη στα άκρα του δικτύου. Ο διαμοιρασμός αρχείων είναι η πιο διαδεδομένη και ευρέως ανεπτυγμένη διομότιμη εφαρμογή. Ωστόσο πολλές άλλες προτείνονται και βρίσκονται υπό σχεδιασμό με σκοπό την εκμετάλλευση διαφόρων πόρων όπως η υπολογιστική ισχύς, η μνήμη, το εύρος ζώνης, και άλλοι.

Η αρχιτεκτονική βάσει της οποίας λειτουργούν τα δίκτυα p2p είναι εντελώς διαφορετική από την αρχιτεκτονική πελάτη-εξυπηρετητή (client-server), όπως φαίνεται και στην εικόνα 1, βάσει της οποίας λειτουργεί σήμερα το μεγαλύτερο μέρος του παγκόσμιου διαδικτύου.

Κύριο χαρακτηριστικό αυτής, είναι πως ο πελάτης μόνο μπορεί να χρησιμοποιήσει τους πόρους ενός εξυπηρετητή. Αντίθετα σε ένα p2p δίκτυο, ένας κόμβος του λειτουργεί ταυτόχρονα τόσο ως πελάτης όσο και ως εξυπηρετητής, αφού 'ανταλλάσσει' υπολογιστικούς πόρους με άλλους υπολογιστές. Οι σημαντικές και συχνά υπερβολικές δυνατότητες των συμβατικών προσωπικών υπολογιστών ως προς την υπολογιστική ισχύ, και η εξάπλωση των ευρυζωνικών συνδέσεων τα τελευταία χρόνια έχουν δημιουργήσει τις κατάλληλες προϋποθέσεις για την ανάπτυξη συστημάτων ομότιμων οντοτήτων. Δηλαδή συστήματα που αποτελούνται από υπολογιστές με συγκεκριμένες δυνατότητες και πανόμοιους ρόλους, για αυτό και οι χρήστες τους (peers) ονομάζονται ομότιμοι. Οι peers σχηματίζουν ένα υπερκείμενο δίκτυο (overlay network) με σκοπό την αξιοποίηση των αχρησιμοποίητων πόρων τους, όπως μνήμη υπολογιστική ισχύς, εύρος ζώνης δικτύου πρόσβασης και/ή τον διαμοιρασμό του διαθέσιμου περιεχομένου. Ακόμα πιο ενδιαφέρουσα από τα τεχνικά θεμέλια των συστημάτων είναι οι κοινωνικές προοπτικές. Με διάφορους τρόπους διανέμουν περιεχόμενο, αποτελέσματα και έλεγχο στους χρήστες. Οι σημαντικότερες σύγχρονες εφαρμογές ομότιμων οντοτήτων είναι οι:

- Napster

- BitTorrent

- SETI@Home,

- Gnutella

- KaZaA

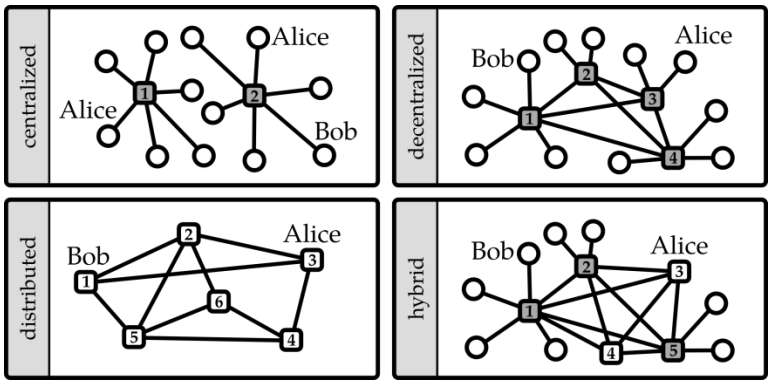
- Skype

- Tribler

4.2.1 Κατηγοριοποίηση Peer-to-Peer συστημάτων

Τα δικτυα χωριζονται σε 4 κατηγοριες οι οποιες απεικονιζονται παρακατω. Επιγραμματικα ειναι :

- Κεντρικοποιημένη
- Αποκεντρωμένη
- Κατανεμημένη
- Υβριδική



4.1Απεικόνιση δομών (Κεντρικοποιημένη, αποκεντρωμένη, κατανεμημένη, υβριδική) (29)

4.2.1.1 Κεντρικοποιημένα peer-to-peer συστήματα

Στις κεντρικοποιημένες αρχιτεκτονικές υπάρχει ένας κεντρικός εξυπηρετητής (Directory Server) στον οποίο απευθύνουν οι κόμβοι τις ερωτήσεις τους για να πληροφορηθούν που βρίσκονται οι επιθυμητές πληροφορίες (π.χ Napster). Μια τέτοια αρχιτεκτονική αν και είναι αρκετά αποδοτική, δεν έχει την ιδιότητα της κλιμάκωσης ενώ έχει ενιαίο σημείο της αποτυχίας (bottleneck).

4.2.1.2 Ιεραρχικά

Οι κόμβοι οργανώνονται σε ιεραρχική δομή όπως γίνεται με τους DNS στο διαδίκτυο. Στα ιεραρχικά peer-to-peer συστήματα εισάγεται ή έννοια των “super-peers” (FastTrack). Η δομή τους μπορεί να είναι κεντρικοποιημένη ή μη.

4.2.1.3 Μη Κεντρικοποιημένα peer-to-peer συστήματα

Μια άλλη κατηγορία αρχιτεκτονικών είναι οι μη – κεντρικοποιημένες όπου οι κόμβοι συγκροτούν το overlay δίκτυο είτε δομημένα ακολουθώντας κανόνες για τον σχηματισμό του δικτύου, είτε αδόμητα όπου δεν υπάρχει ούτε κεντρικό directory ούτε ακριβείς οδηγίες για τον σχηματισμό τοπολογίας του δικτύου και την τοποθέτηση των περιεχομένων.

- **Δομημένα:** Στα δομημένα peer-to-peer συστήματα οι κόμβοι οργανώνονται σε δομημένο γράφο για το σχηματισμό του overlay δικτύου. Στα δεδομένα αντιστοιχίζεται ένα κλειδί και η τοποθέτηση τους στους κόμβους γίνεται με προκαθορισμένο τρόπο έτσι ώστε να διευκολύνεται η αναζήτησή τους και να επιτυγχάνεται η κλιμάκωση. Για να γίνει αυτό χρησιμοποιούνται πίνακες κατακερματισμού (Distributed Hash Tables, DHTs), οι οποίοι αντιστοιχούν κλειδιά σε τιμές. Η τοποθέτηση των αρχείων στα χαλαρά δομημένα συστήματα (Freenet) βασίζεται σε εκτιμήσεις για το που μπορεί να βρεθεί η αναζητούμενη πληροφορία. Στα αυστηρά δομημένα συστήματα τόσο η δόμηση του overlay δικτύου όσο και η τοποθέτηση των αρχείων είναι σαφώς καθορισμένη. Ο εντοπισμός ενός αντικειμένου (δεδομένου) από μια εφαρμογή στα δομημένα συστήματα γίνεται σε μικρό αριθμό βημάτων (network hops), υπό την απαίτηση βέβαια να διατηρείται ένας μικρός πίνακας δρομολόγησης σε κάθε κόμβο. Παραδείγματα τέτοιων συστημάτων αποτελούν τα: Content Addressable Network (CAN), Chord, Tapestry, Pastry, Kademlia και Viceroy.
- **Αδόμητα:** Στα συστήματα αυτά δεν υπάρχει καμιά δομή στο overlay δίκτυο και τα περιεχόμενα τοποθετούνται σε κόμβους στο δίκτυο χωρίς γνώση της τοπολογίας ή άλλης συσχέτισης με αυτό. Τα μη δομημένα συστήματα είναι κατάλληλα σε περιπτώσεις όπου μεγάλο πλήθος κόμβων μετέχει παροδικά στο δίκτυο χωρίς όμως αποδοτικούς μηχανισμούς αναζήτησης, κλιμάκωσης, διαθεσιμότητας. Υποστηρίζουν καλύτερα πολύπλοκες ερωτήσεις σε σχέση με τα δομημένα. Αδόμητα peer-to-peer δίκτυα είναι τα: Napster, Gnutella, FastTrack, KaZaA, BitTorrent, κ.α.

4.2.2 BitTorrent Πρωτόκολλο

Ο προγραμματιστής Brian Cohen σχεδίασε το 2001 ένα πρωτόκολλο για την ανταλλαγή αρχείων μεταξύ ομότιμων χρηστών (peer-to-peer file-sharing protocol) σε όλο τον κόσμο με την ονομασία BitTorrent. Αυτό που το διαφοροποίησε από τα υπόλοιπα P2P δίκτυα είναι πως τα αρχεία δεν βρίσκονται σε ένα κεντρικό διακομιστή αλλά στους σκληρούς δίσκους των χρηστών. Έτσι το BitTorrent εστιάζει στην αποτελεσματική αναπαραγωγή και διανομή δεδομένων με πολύ μεγάλες ταχύτητες χωρίς να απαιτείται μεγάλο και δαπανηρό εύρος ζώνης (bandwidth) σε μια κεντρική τοποθεσία. Ως αποτέλεσμα αυτού του χαρακτηριστικού, το BitTorrent έγινε εξαιρετικά δημοφιλές σύστημα με εκατομμύρια χρήστες και χιλιάδες torrent files παγκοσμίως. Ανά πάσα στιγμή το BitTorrent έχει κατά μέσο όρο περισσότερους ενεργούς χρήστες από το YouTube και το Facebook συνδυασμένα ενώ κατέχει το 53% της συνολικής P2P κίνησης (Ye, Zhang, Li, Su, 2010).

4.2.2.1 Τα μέρη που απαρτίζουν τα BitTorrent

Τα βασικά μέρη τα οποία αποτελούν το σύστημα αυτό είναι τα εξής:

- ***Το σμήνος των BitTorrent (BitTorrent Swarm):*** αποτελείται από τους ομότιμους χρήστες όπου συμμετέχουν στην διαδικασία ανεβάσματος/κατεβάσματος των δεδομένων ακολουθώντας το BitTorrent protocol.
- ***Το .torrent file:*** περιέχει μεταδεδομένα (metadata) σχετικά με το αρχείο που κατεβάζουμε και πληροφορίες όπως το όνομα, τα κομμάτια (chunks, pieces) που το αποτελούν καθώς και το μέγεθος τους και σημαντικότερα περιλαμβάνει την διεύθυνση IP των trackers που συντονίζουν την επικοινωνία μεταξύ των ατόμων που συμμετέχουν στο σμήνος που ασχολείται με το συγκεκριμένο torrent file.
- ***Torrent tracker:*** είναι ο server που καταγράφει τους συμμετέχοντες του κάθε σμήνους σε σχέση με κάθε torrent file. Δεν έχει άμεση επαφή με τα δεδομένα που συναλλάσσονται, ούτε αντίγραφα των αρχείων αλλά γνωρίζει τι κατέχει το κάθε μέλος και την πορεία της συναλλαγής του, καθώς όλοι οι συμμετέχοντες έχουν την υποχρέωση να τον ενημερώνουν περιοδικά για την κατάσταση τους. Με άλλα λόγια διαμεσολαβεί στην επικοινωνία μεταξύ των συνδεδεμένων προγραμμάτων-ατόμων. Οι trackers μπορεί να είναι είτε δημόσιοι (public) είτε ιδιωτικοί (private) ή αλλιώς BitTorrent Darknets (Zhang , Dhungel, Wu, Liu , Ross, 2010). Οι δημόσιοι έχουν ελεύθερη πρόσβαση, ενώ στους ιδιωτικούς έχουν πρόσβαση μόνο τα εγγεγραμμένα μέλη, ενώ η εγγραφή συνήθως γίνεται εφικτή μόνο μετά από πρόσκληση ήδη εγγεγραμμένου μέλους.
- ***BitTorrent Portal:*** είναι ο sever στον οποίο ανακοινώνονται τα torrent file ώστε να μπορούμε να αναζητήσουμε αυτό που ψάχνουμε. Στα ιδιωτικά BitTorrent συνήθως ο ιδιοκτήτης έχει διπλό ρόλο, είναι ο tracker και το portal ταυτόχρονα.
- ***BitTorrent Client ή Peer:*** η έννοια του πελάτη συμπεριλαμβάνει συνήθως δυο έννοιες. Αναφέρεται είτε στο άτομο-υπολογιστής που συμμετέχει σε ένα σμήνος είτε στο πρόγραμμα όπου υλοποιεί το BitTorrent protocol και επιτρέπει την P2P σύνδεση. Παραδείγματα τέτοιων πελατών-προγραμμάτων είναι το μTorrent και το Vuze. Οι συμμετέχοντες χωρίζονται σε δύο κατηγορίες: στους leechers και τους seeders (λόγω έλλειψης ικανοποιητικής μετάφρασης θα συνεχίσουμε να αναφερόμαστε σε αυτούς τους όρους με την αγγλική ορολογία). Seeders αποκαλούνται όσοι κατέχουν ολοκληρωτικά τα αρχεία στα οποία αναφέρετε το εν λόγω torrent file ενώ leechers είναι οι πελάτες όπου δεν το έχουν ακόμη ολοκληρωμένο και το λαμβάνουν από τους υπόλοιπους πελάτες. Αυτόματα μόλις ένας leecher αποκτήσει το 100% των αρχείων μετατρέπεται σε seeder και ενημερώνοντας τον tracker γίνεται διαθέσιμος ως seeder στα υπόλοιπα μέλη του σμήνους.

4.2.2.2 Τρόπος λειτουργίας

4.2.2.2.1 Αρχικό στάδιο

Το πρώτο βήμα στην όλη διαδικασία είναι η δημιουργία ενός torrent file πάνω στα αρχεία που θέλουμε να διακινήσουμε και έπειτα η ανακοίνωση του σε ένα portal. Όπως αναφέραμε το αρχείο αυτό θα περιέχει πληροφορίες όπως όνομα, το είδος του αρχείου, αριθμός κομματιών που το αποτελούν, μέγεθος κομματιών και συνολικό μέγεθος, ημερομηνία που

ανέβηκε και από ποιόν, τρέχων αριθμό seeders και leechers - αριθμός που ενημερώνεται ανά τακτά χρονικά διαστήματα και γενικότερες πληροφορίες. Ο ενδιαφερόμενος για αυτό το αρχείο μπορεί με την χρήση ενός κατάλληλου προγράμματος- πελάτη να ανοίξει αυτό το αρχείο το οποίο θα τον οδηγήσει σε ένα tracker που διαχειρίζεται το συγκεκριμένο torrent file και έχει καταχωρημένες τις IP διευθύνσεις των υπόλοιπων .

Αυτό συνήθως γίνεται με ένα HTTP Get αίτημα. Το νέο μέλος στέλνει ένα μήνυμα (announce started) στον tracker ουσιαστικά δηλώνοντας του την παρουσία του και την επιθυμία του να μάθει τις διευθύνσεις των υπόλοιπων μελών. Ο tracker του απαντάει στέλλοντας του ένα μήνυμα με τις διευθύνσεις κάποιων τυχαία επιλεγμένων μελών (40-200 συνήθως) οι οποίοι θα αποτελούν την γειτονία του. Όταν κάποιος θέλει να αποσυνδεθεί πρέπει να στείλει ένα ανάλογο μήνυμα εξόδου (announce stopped) ώστε να δηλώσει την έξοδο του. Καθώς οι ενημερώσεις του tracker γίνονται πολύ συχνά, όταν κάποιο μέλος χάσει πολλούς γείτονες του (συνήθως φτάσει τους 20) ξαναστέλνει μια αίτηση ώστε να λάβει νέους γείτονες και να συνεχίσει ανεπηρέαστος το κατέβασμα του.

4.2.2.2.2 Σύνδεση στο σμήνος

Για να επικοινωνήσουν δύο μέλη του σμήνους κάνουν χρήση του peer wire protocol το οποίο λειτουργεί πάνω από το TCP . Τόσο τα μηνύματα επικοινωνίας, όσο και η μεταφορά των δεδομένων έχουν την μορφή μιας συνεχούς αμφίδρομης ροής προκαθορισμένων σε μέγεθος μηνυμάτων. Εφόσον εδραιωθεί η TCP σύνδεση μεταξύ των μελών, αποστέλλεται ένα μήνυμα χειραψίας που περιέχει την ταυτότητα του χρήστη, αν ανταποκριθεί με ένα παρόμοιο μήνυμα ο αρχικός αποδέκτης μπορούμε να το θεωρήσουμε έναρξη της επικοινωνίας μας, διαφορετικά το κανάλι μας θεωρείται κλειστό. Στην συνέχεια ο καθένας στέλνει πληροφόρηση σχετική με τα κομμάτια που κατέχει μέσω ενός bitfield μηνύματος. Το μήνυμα αυτό αποτελείται από μια σειρά bits όπου το κάθε bit αντιστοιχίζεται στον δείκτη των κομματιών μας. Όταν αποκτήσουμε κάποιο κομμάτι, αφού πρώτα ελεγχτεί η ορθότητα του μέσω της διαδικασίας SHA-1 hash, ενημερώνουμε με ένα μήνυμα have τους γύρω μας για την νέα απόκτηση μας επιτυγχάνοντας έμμεσα μια διαρκής ενημέρωση σε επίπεδο σμήνους.

Το κάθε μέλος ενός σμήνους μπορεί να βρίσκεται στις δύο παρακάτω ενδεχόμενες καταστάσεις για κάθε του σύνδεση: ενδιαφερόμενος (interested) και μπλοκαρισμένος (chocked). Εάν κάποιος βρίσκεται στην κατάσταση μπλοκαρισμένος, δεν μπορεί να κατεβάσει μέχρι να του δοθεί η ξανά η ελευθερία κάτι που συνήθως γίνεται μέσω αύξησης των αρχείων που μοιράζεται με τον σμήνος, δηλαδή με uploading περισσότερων αρχείων. Ο αλγόριθμος chocking είναι αυτός που εξασφαλίζει αυτή την συμπεριφορά. Περιοδικά, συνήθως ανά 10 δευτερόλεπτα, ένας leecher , διαλέγει να κάνει unchoke τους n leechers από τους οποίους κατεβάζει περισσότερο (συνήθως 4) στα τελευταία 20 δευτερόλεπτα και να τους στείλει κομμάτια, ενώ όλοι οι υπόλοιποι γείτονές του περνούν σε κατάσταση μπλοκαρίσματος. Δηλώνοντας ενδιαφερόμενος, το μέλος που θέλει να κατεβάσει κάποιο κομμάτι εκδηλώνει το ενδιαφέρον του στο μέλος που κατέχει το κομμάτι, ενώ μόλις το λάβει θα πρέπει να ενημερώσει για την αλλαγή αυτή της κατάστασης του στέλνοντας ένα μήνυμαπως δεν ενδιαφέρεται άλλο (not interested). Ο μηχανισμός αυτός των choked/unchecked και interested/ not interested καταστάσεων δίνει στο BitTorent protocol μια “αίσθηση δικαίου” (29). Τα ιδιωτικά BitTorrent Darknets καταγράφουν τις κινήσεις των εγγεγραμμένων μελών τους και τους υποχρεώνουν να διατηρούν μια αναλογία ανεβάσματος/κατεβάσματος κοντά στο 1/1 έτσι ώστε να υπάρχουν πάντα αρχεία σε επάρκεια. Για να πετύχουν τέτοιες αναλογίες συνήθως ακολουθούνται πολιτικές κινήτρων ή “τιμωρίας”. Η γενικότερη πολιτική που ακολουθείται από τα torrents , είτε δημόσια, είτε

ιδιωτικά είναι η τακτική Tit-for-Tat, "δούναι και λαβείν", η οποία εξασφαλίζει ότι ο κάθε leecher θα παρέχει κομμάτια στους leechers από τους οποίους κατεβάζει περισσότερο και δίνει προτεραιότητα με αυτόν τον τρόπο στα μέλη που παρέχουν τα περισσότερα αρχεία. Περιστασιακά, στους δημόσιους trackers κυρίως, δίνεται η δυνατότητα και σε μέλη χωρίς αρχεία να συμμετάσχουν στο σμήνος και να περάσουν σε κατάσταση unchoked, δηλαδή τους παρέχεται η ελευθερία κατεβάσματος, παρόλο που οι ίδιοι δεν έχουν κάτι να μοιραστούν με την υπόλοιπη ομάδα. Η τεχνική αυτή έχει συνήθως αναλογία 1 προς 3 διαδικασίες unchoke και καλείται optimistic unchoking, καθώς ενέχει την ελπίδα ότι τα νέα αυτά μέλη σύντομα θα ανεβάσουν αρχεία που θα μοιραστούν με το σμήνος. Μια ακόμα τεχνική που ακολουθείτε από τον χρήστη που κατεβάζει, μόλις ξεμπλοκαριστεί από τους γείτονές του, είναι η τεχνική της rarest first policy όπου ελέγχει την δημοτικότητα των κομματιών που υπάρχουν διαθέσιμα και επιλέγει να κατεβάσει πρώτα αυτό που είναι πιο σπάνιο ώστε να μην δημιουργηθούν προβλήματα εύρεσης συγκεκριμένων κομματιών.

4.2.2.3 Επεκτάσεις

Καθώς τα BitTorrents πολύ σύντομα έγιναν ευρέως γνωστά για την διακίνηση υλικού όπως ταινίες και σειρές, μουσική κλπ. αρχεία για το οποία δεν είχαν αποκτηθεί οι ανάλογες άδειες κατοχής και διακίνησης, οι trackers έγιναν στόχοι μηνύσεων και πολλοί κινήθηκαν δικαστικά εναντίων τους. Έτσι το BitTorrent protocol εξελίχθηκε και προσπάθησε να δημιουργήσει μηχανισμούς αντιμετώπισης. Έτσι έχουμε την εμφάνιση των DHT (Distributed Hash Table), οι οποίοι παρέχουν την δυνατότητα στον χρήστη των BitTorrent να πληροφορηθεί τις IP των υπόλοιπων χρηστών χωρίς να επικοινωνήσει με τον tracker (30). Αυτό το αποκεντροποιημένο σύστημα λειτουργεί όπως ένας Hash table, με ένα ζευγάρι τιμών και κλειδιού.

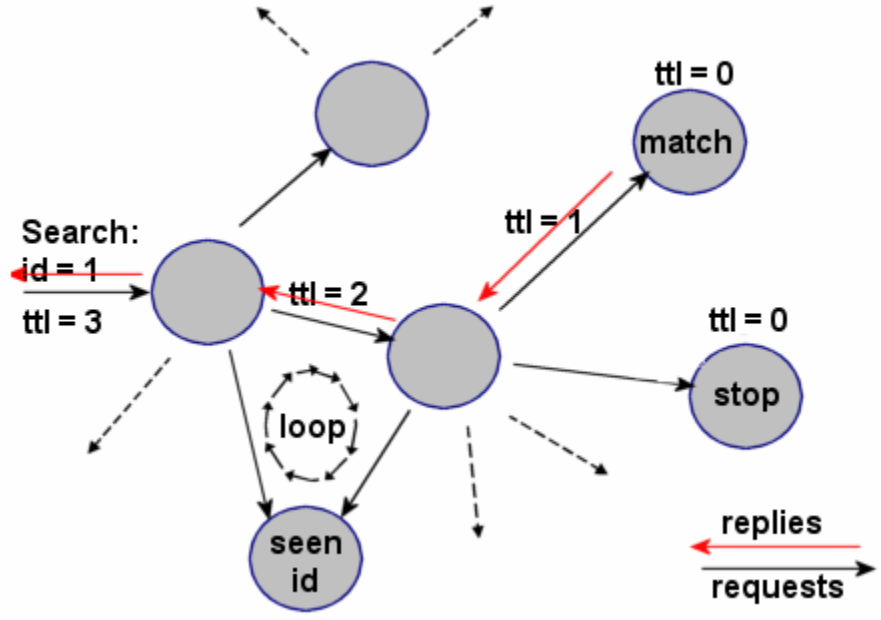


Figure 4.2 Παράδειγμα διάδοσης ερωτήματος σε αδόμητο peer-to-peer δίκτυο

Ένας ακόμη τρόπος ήταν μέσω αιτημάτων PEX (Peer Exchange). Κάποιος που θέλει να κατεβάσει, στέλνει ένα αίτημα PEX σε ένα του γείτονα του, το οποίο ζητάει τις IP

διευθύνσεις της δικιάς του γειτονιάς. Αν αυτός ο γείτονας που το λαμβάνει υποστηρίζει το PEX πρωτόκολλο, τότε του απαντάει με τις ανάλογες διευθύνσεις. Με αυτόν τον τρόπο με την μορφή "κουτσομπολιού" (gossiping protocol), με λίγα αιτήματα μπορεί κάποιος να πληροφορηθεί τις διευθύνσεις μεγάλου μέρους του σμήνους. Με την εισαγωγή αυτών των δύο τρόπων ενημέρωσης του χρήστη, χωρίς να είναι απαραίτητο να απευθυνθεί στον tracker, και κυρίως με την εισαγωγή του DHT πίνακα, το σύστημα των BitTorrent από κεντροποιημένο, πήρε μια υβριδική μορφή, η οποία έφερε μεγάλη αλλαγή στην απόδοση του.

4.3 Κεντρικοποιημένα συστήματα

Αν και από τη φύση του το διαδίκτυο είναι κατανεμημένο και αποκεντρωτικό η πραγματική του χρήση έχει επικεντρωθεί σε λίγες εταιρίες οι εξυπηρετητές των οποίων αναλαμβάνουν και το μεγαλύτερο μέρος της διαδικτυακής κίνησης. Αυτό επιτυγχάνεται κυρίως μέσα από τις δωρεάν υπηρεσίες που προσφέρουν πάντα με αντάλλαγμα τη πώληση των προσωπικών δεδομένων των χρηστών σε διαφημιστικές εταιρίες. Καταυτό τον τρόπο ενώ το διαδίκτυο ξεκίνησε ως ένα αποκεντρωμένο ομότιμο δίκτυο, κατέληξε να στηρίζεται ως επί το πλείστον σε ελάχιστες εταιρίες στις οποίες οι χρήστες εναποθέτουν το μεγαλύτερο ποσοστό της διαδικτυακής τους δραστηριότητας. Γι' αυτό και θεωρούμε ότι η κεντρική δομή στην οποία κατέληξε το διαδίκτυο θα οδηγήσει στη επιτήρηση και την ανελευθερία των χρηστών του.

Τα Κεντρικοποιημένα κοινωνικά δίκτυα βασίζονται στη δομή ενός κεντρικού εξυπηρετητή που εξυπηρετεί τους πελάτες χρήστες. Το τι συμβαίνει με τα δεδομένα που κινούνται μέσα από τους κεντρικούς εξυπηρετητές επαφίονταν στη κρίση της εταιρίας στην οποία ανήκουν οι εξυπηρετητές. Λόγο της καπιταλιστικής οικονομίας στην οποία βασιζόμαστε η εταιρία θα προσπαθήσει να αποκομίσει το μεγαλύτερο δυνατό κέρδος από τα δεδομένα που δίνουν απλόχερα οι χρήστες.

4.4 Επίλογος

Τα αποκεντρωμένα συστήματα επιτρέπουν στο χρήστη να επιλεγεί τη φυσική τοποθεσία που θα αποθηκεύει τα δεδομένα του. Αντίστοιχα τα Κεντρικοποιημένα συστήματα υπηρεσιών αναγκάζουν τους χρήστες να εμπιστεύονται τα δεδομένα τους στη κρίση των εταιριών που τα διαχειρίζονται. Θεωρούμε αυτονόητο δικαίωμα των χρηστών του διαδικτύου να διασφαλίζεται η ιδιωτικότητα των επικοινωνιών τους και να μη χρησιμοποιούνται ως προϊόντα για πώληση.

5ο Κεφάλαιο

5.1 Γενική ιδέα

Αυτό που προτείνουμε είναι ένα δίκτυο φθηνών υπολογιστών που θα αποτελούνται από ένα λειτουργικό σύστημα και συγκεκριμένα προγράμματα που δημιουργούν μια πλατφόρμα ανάπτυξης εφαρμογών στις διαδεδομένες γλώσσες του διαδικτύου (html,css,javascript). Οι εφαρμογές αυτές θα αποτελούν ιστοσελίδες που θα εξυπηρετούνται από τους φθηνούς υπολογιστές οι οποίοι θα συνδέονται στον διαδίκτυο μέσω της οικιακής σύνδεσης του χρήστη και θα είναι προσβάσιμες από οπουδήποτε υπάρχει πρόσβαση στο διαδίκτυο. Οι υπολογιστές θα χρησιμοποιούν το DHT (Distributed Hash Table – Κατανεμημένος Πίνακας Κατακερματισμού) για την ανακάλυψη άλλων χρηστών όπου κάθε hash θα συνδέει κατανεμημένες εφαρμογές που θα υπάρχουν στους υπολογιστές που αναφέραμε και θα συγχρονίζει τα δεδομένα τους.

Οι χρήστες θα μπορούν να ανακοινώνουν πόρους (hash) στο κατανεμημένο πίνακα όπου θα διανέμονται στο δίκτυο ώστε να είναι αναζητήσαμε. Οι πόροι αυτοί αφού θα αναζητηθούν, επιστρέφουν σαν αποτέλεσμα τις διευθύνσεις IP που τις έχουν ανακοινώσει. Σε περίπτωση ενός χρήστη που ανακοινώνει τη παρουσία του στο κατανεμημένο δίκτυο, η αναζήτηση του πόρου βάσει του hash που απορρέει από το όνομα χρήστη μετά από κωδικοποίηση, βάσει του αλγορίθμου sha512, θα επιστρέφεται μια διεύθυνση IP, η οποία αντιστοιχεί στο συγκεκριμένο χρήστη.

Μετά από τη επιλογή και αποδοχή σύνδεσης με κάποιο χρήστη στο κατανεμημένο δίκτυο θα δημιουργείτε μια κρυπτογραφημένη απευθείας σύνδεση μεταξύ των χρηστών μέσω της οποίας θα ανταλλάσουν και δεδομένα.

Το περιβάλλον διεπαφής του χρήστη με τη πλατφόρμα θα γίνεται μέσω ενός περιηγητή όπου με το κατάλληλο όνομα χρήστη και κωδικό θα του δίνει δικαιώματα πρόσβασης στις αντίστοιχες βάσεις δεδομένων που αφορούν το συγκεκριμένο χρήστη. Η απλότητα χρήσης μέσω ενός κατανοητού περιβάλλοντος για τις ενέργειες που μπορεί να κάνει πρέπει να είναι βασικό χαρακτηριστικό της πλατφόρμας. Πολλά αντίστοιχα εγχειρήματα έχουν αποτύχει λόγο της δυσχρηστίας του περιβάλλοντος που αποτρέπει τους χρήστες από το να ασχοληθούν.

Παρακάτω εναποθέτουμε εικόνες, όπου φαίνεται η διαδικασία της δημιουργίας λογαριασμού, αναζήτησης και πρόσκλησης φίλων, αποδοχή αιτημάτων φιλίας και το περιβάλλον του κοινωνικού δικτύου. Απώτερος σκοπός είναι όλα αυτά τα περιβάλλοντα να ενοποιηθούν σε

ένα περιβάλλον παρόμοιο των δημοφιλών κοινωνικών δικτύων.

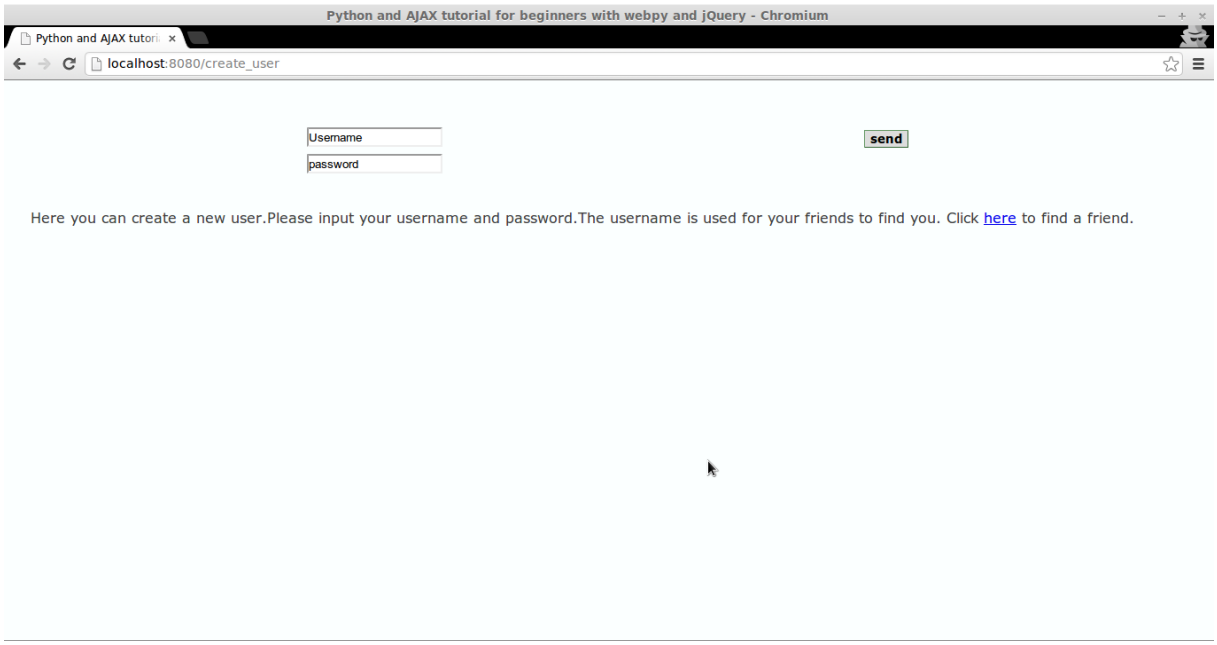


Figure 5.1 Δημιουργια νεου χρηστη (κομβος 1)

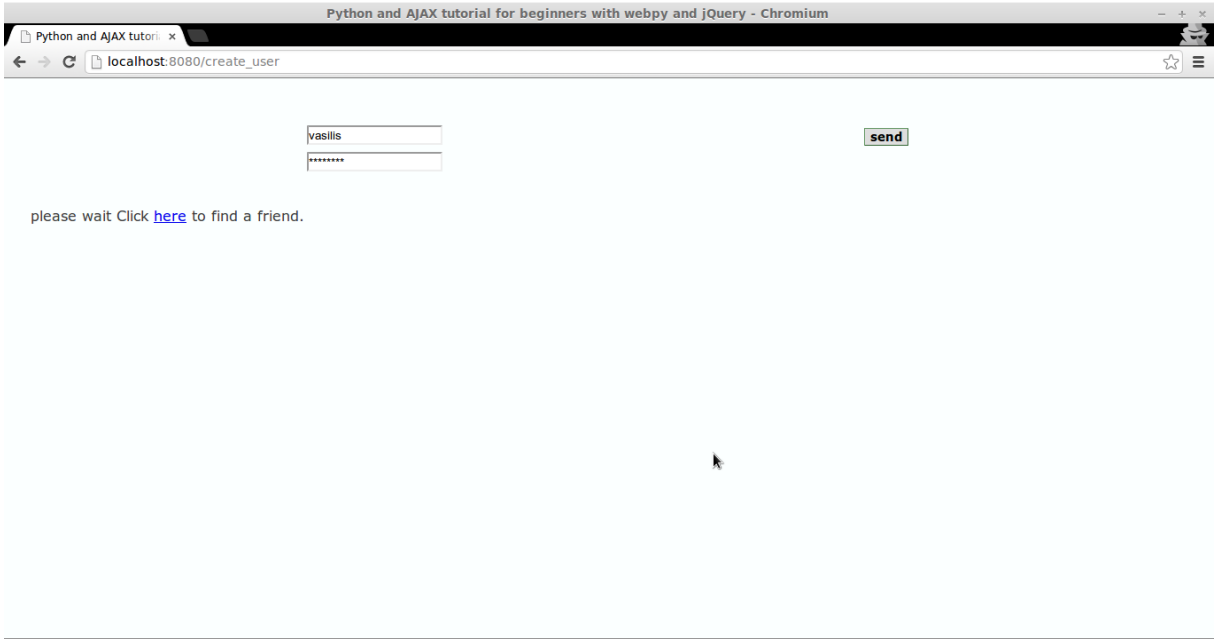


Figure 5.2 Αναζητηση για το αντο ονομα χρηστη ηδη χρησιμοποιηται (κομβος 1)

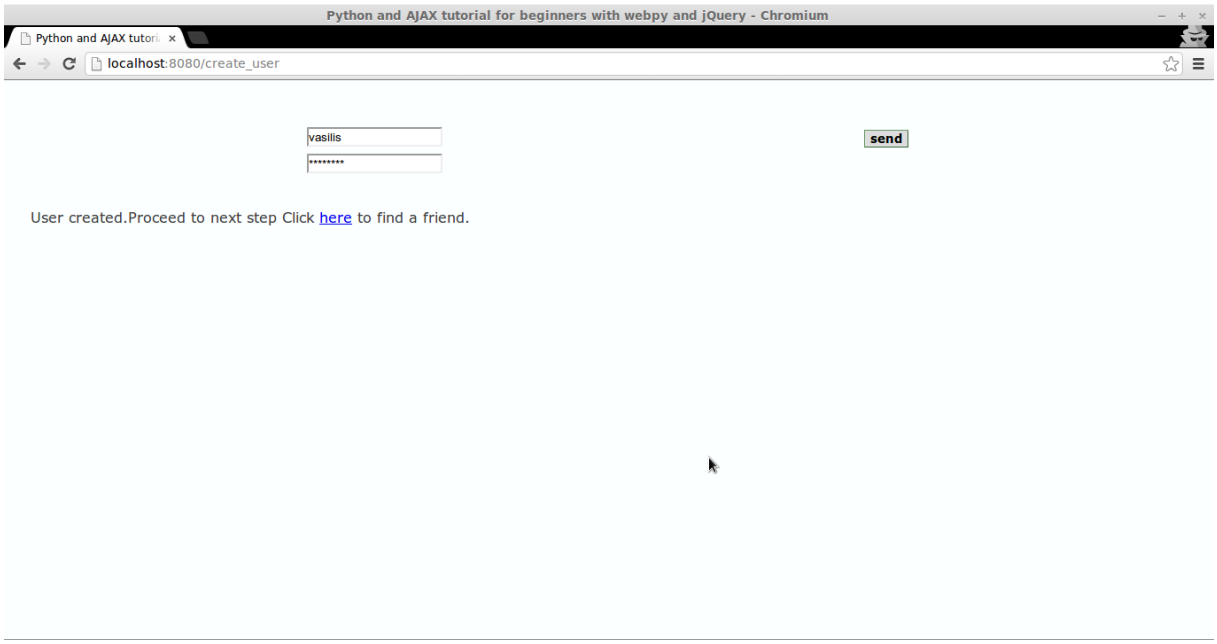


Figure 5.3 Μηνυμα οτι δημιουργηθηκε ο χρηστης (κομβος 1)

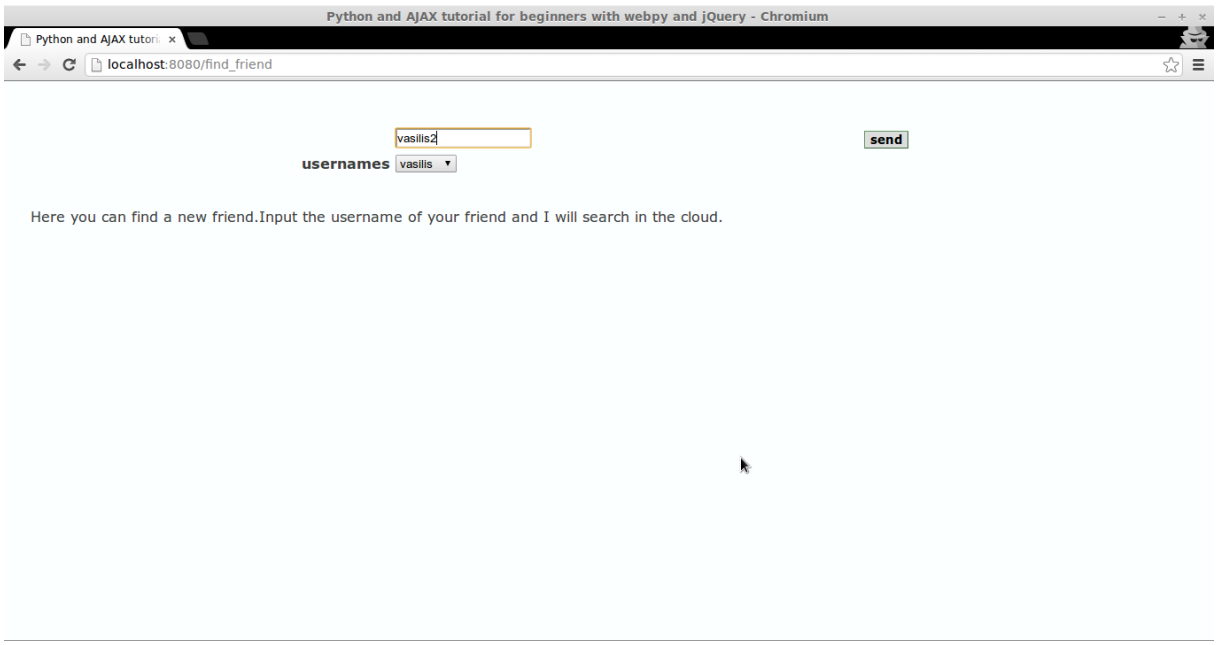


Figure 5.4 Αναζητηση φιλων (κομβος 1)

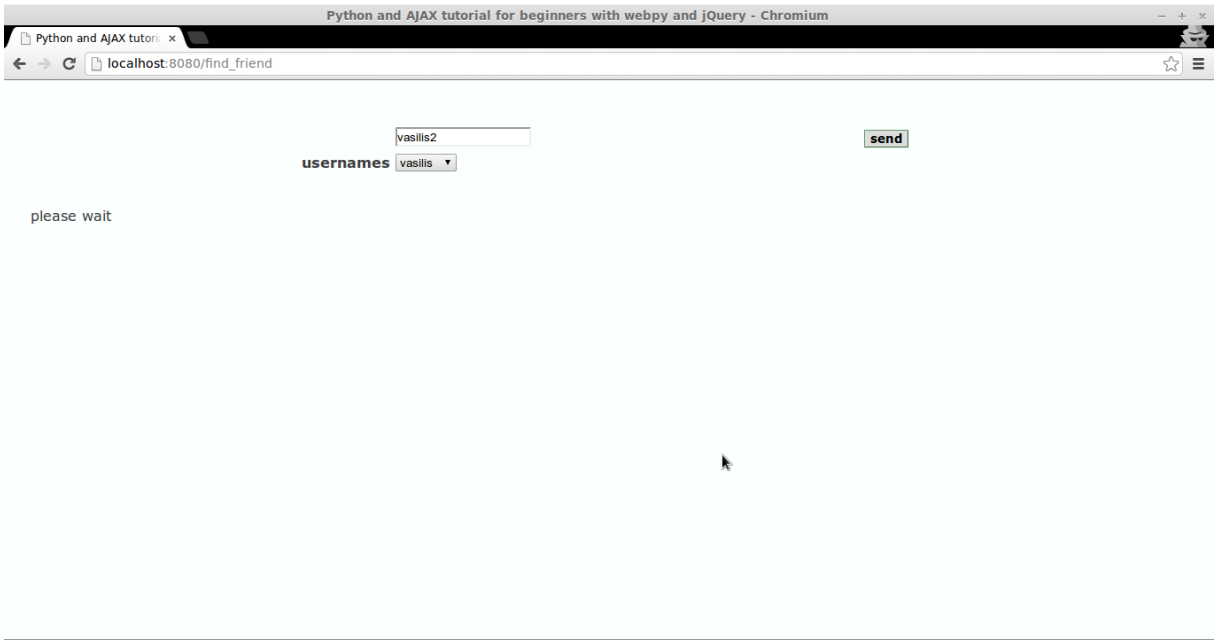


Figure 5.5 Αναμονη για ευρεση φιλου (κομβος 1)

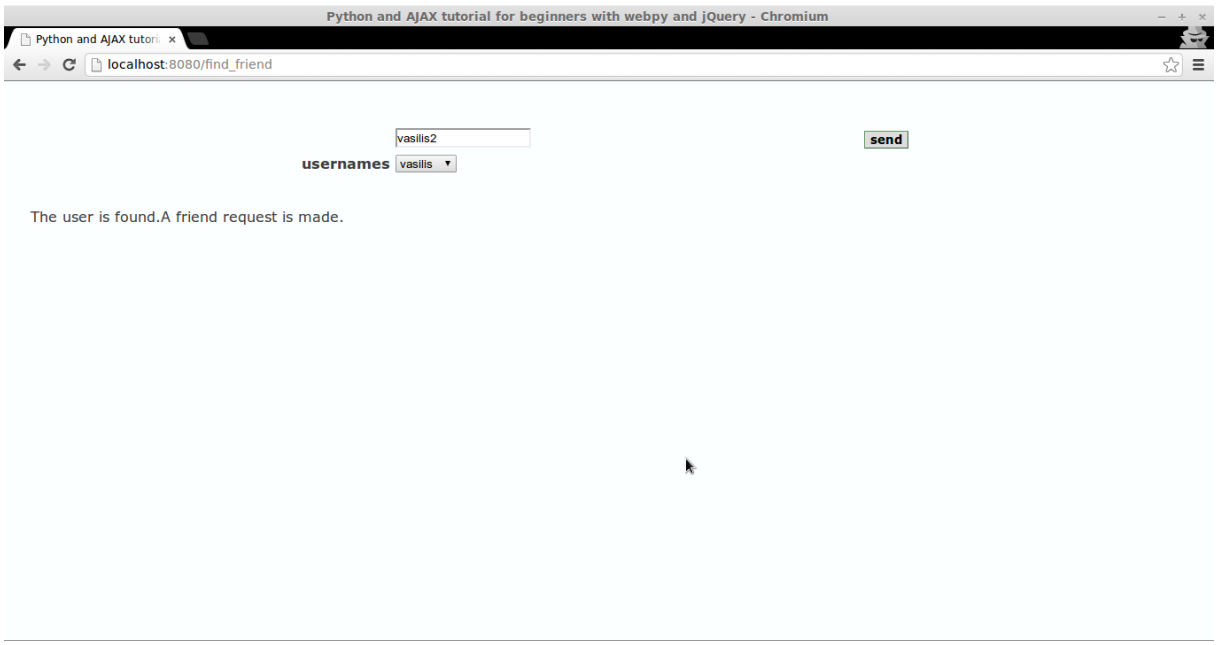


Figure 5.6 Μηνυμα αποστολης αιτηματος φιλιας (κομβος 1)

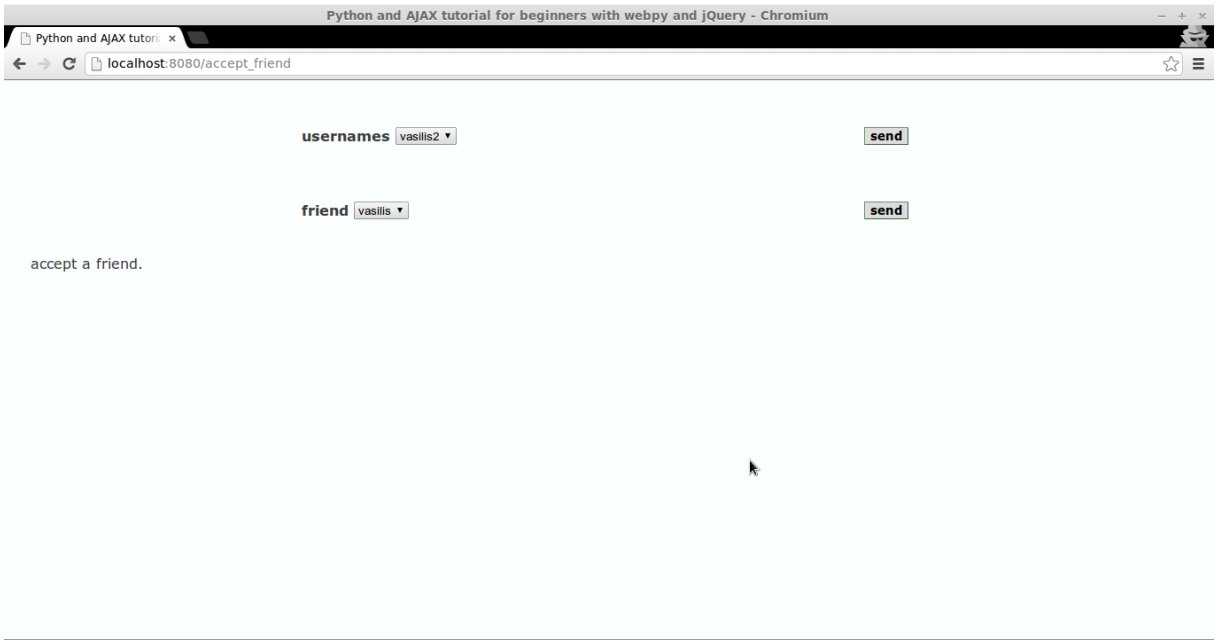


Figure 5.7 Επιλογη χρηστη στον οποιο εχουν σταλει αιτηματα φιλιας (κομβος 2)

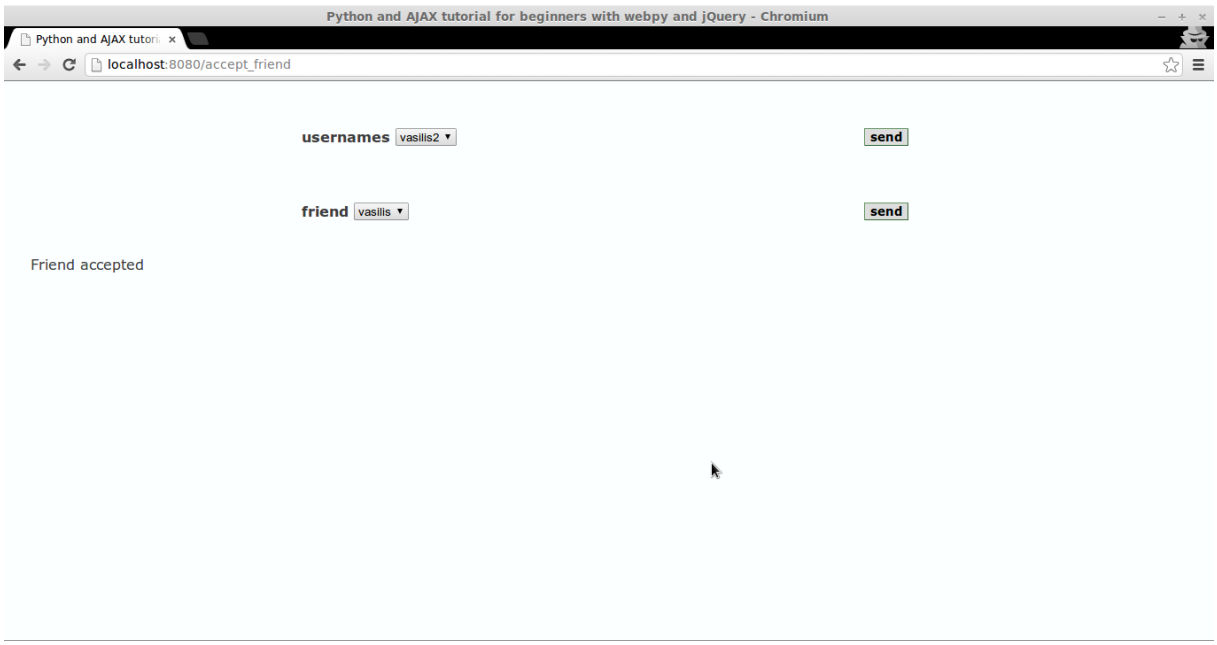


Figure 5.8 Αποδοχη ενος αιτηματος φιλιας (κομβος 2)

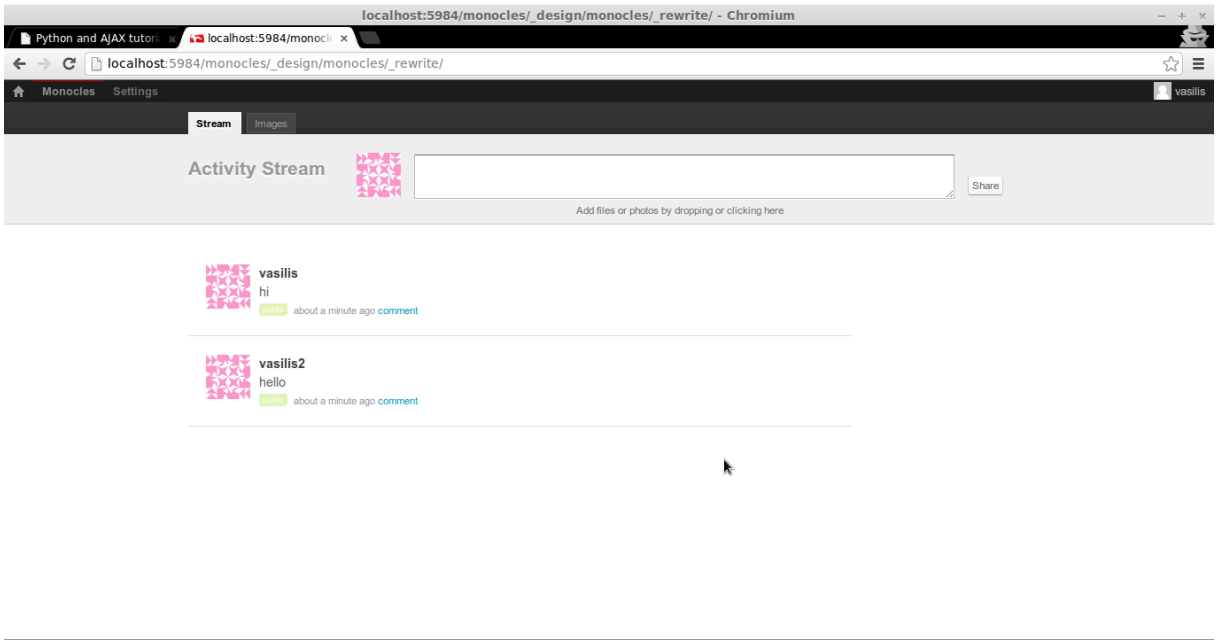


Figure 5.9 Περιβάλλον διεπαφής κοινωνικού δικτύου μεταξύ των χρηστών (κόμβος 1)

5.2 Απαιτήσεις για την υλοποίηση του συστήματος

Οι βασικότερες πτυχές που πρέπει να καλύπτει το σύστημα που αναπτύσσουμε είναι η ασφάλεια των επικοινωνιών και ευκολία χρήσης. Ζητούμενο είναι οι χρήστες να επικοινωνούν με κρυπτογράφηση χωρίς να χρειάζεται να κατέχουν ειδικές γνώσεις παραμετροποίησης σε προγράμματα κρυπτογράφησης. Επίσης, θεωρούμε ότι για να μπορέσει ένα τέτοιο σύστημα να φανεί ελκυστικό σε όσο το δυνατόν περισσότερους χρήστες χρειάζεται οι συνήθεις διαδικασίες που κάνουν οι χρήστες για την αρχική εισαγωγή και χρήση ενός κοινωνικού δικτύου να είναι το ίδιο εύκολα και απλά με τις διαδικασίες των κεντρικοποιημένων συστημάτων.

Η χρήση εφαρμογών ανοιχτού κώδικα θεωρείται επιβεβλημένη για να αυξηθεί η αξιοπιστία του συστήματος μιας και έχει αποδεδειχθεί επανειλημμένως τα τελευταία χρόνια ότι προγράμματα κλειστού κώδικα είναι πολύ πιθανό να περιέχουν τρύπες ασφάλειας που τοποθετούνται επιτηδευμένα από τις εταιρίες που τα αναπτύσσουνε.

Επιγραμματικά τα απαιτούμενα υποσυστήματα και οι γενικότερες δυνατότητες όπου ζητείται να έχει η πλατφόρμα δικτύωσης που θα αναπτύξουμε, στη συνέχεια αποτελείται από τα εξής:

- Μια εφαρμογή για βάσεις δεδομένων που επιτρέπει τον συγχρονισμό δεδομένων ανάμεσα σε ομότιμους κόμβους.
- Η δυνατότητα εξυπηρέτησης αρχείων κατανοητών από συγχρόνους φυλλομετρητές.
- Η δυνατότητα διαχείρισης των δικαιωμάτων πάνω στα δεδομένα για πολλαπλούς χρήστες.
- Ένα ομότιμα κατανεμημένο δίκτυο μέσω του οποίου οι χρήστες θα μπορούν να ανακοινώνουν τη παρουσία τους και θα μπορούν να ανακαλύπτουν νέους χρήστες.

- Μια εφαρμογή που θα μπορεί να αποκωδικοποιεί τα δεδομένα από τη βάση δεδομένων και να τα παρουσιάζει σε μια μορφή αντίστοιχη των σημερινών διαδικτυακών κοινωνικών δικτύων.
- Ένα πρόγραμμα που θα επιτρέπει την αποκεντρωμένη και κρυπτογραφημένη επικοινωνία μεταξύ των κόμβων.
- Ένα λειτουργικό σύστημα που θα στεγάζονται όλες οι παραπάνω εφαρμογές.
- Ένας υπολογιστής χαμηλού κόστους με κατάλληλη επεξεργαστική ισχύ για τη λειτουργία των παραπάνω.

5.3 Η βάση δεδομένων CouchDB

Το σύστημα διαχείρισης μη σχεσιακών βάσεων δεδομένων CouchDB (31)είναι ένας εξυπηρετητής βάσης δεδομένων προσανατολισμένος στα έγγραφα και είναι προσβάσιμο μέσω των REST APIs. Το couch είναι ένα ακρωνύμιο για το «Cluster Of Unreliable Commodity Hardware», που δίνει έμφαση στην κατανεμημένη φύση της βάσης δεδομένων. Το CouchDB έχει σχεδιαστεί για εφαρμογές που είναι προσανατολισμένες στα έγγραφα, όπως τα forums, τα wikis και τα emails. Το CouchDB project είναι μέρος του Apache Foundation και είναι πλήρως γραμμένο στη γλώσσα Erlang.

Η γλώσσα Erlang επιλέχθηκε λόγω του ότι είναι κατάλληλη για ταυτόχρονη επεξεργασία μέσω χαμηλού υπολογιστικού κόστους εφαρμογών και βασίζεται στις αρχές του δομημένου προγραμματισμού. Το CouchDB είναι ένα σύστημα βάσεων δεδομένων χωρίς κάποιο σχήμα. Το CouchDB δεν είναι μόνο μία σχεσιακή βάση δεδομένων, αλλά και ένας εξυπηρετητής για εφαρμογές γραμμένες σε JavaScript (Couchapps) κάτι που επιτρέπει στο πρόγραμμα που υλοποιούμε στη διπλωματική να εξυπηρετεί και κατανεμημένες εφαρμογές από τον προσωπικό εξυπηρετητή του κάθε χρήστη. Το πλεονέκτημα της χρησιμοποίησης του CouchDB ως εξυπηρετητή είναι ότι οι εφαρμογές στο CouchDB μπορούν να δημιουργηθούν απλά με την τοποθέτηση τους μέσα στη βάση δεδομένων και ότι οι εφαρμογές έχουν απευθείας πρόσβαση στη βάση δεδομένων χωρίς την καθυστέρηση που επιφέρει ένα πρωτόκολλο ερωτημάτων.

5.3.1 Το μοντέλο δεδομένων του CouchDB

Η πληροφορία στο CouchDB είναι οργανωμένη σε έγγραφα. Κάθε έγγραφο μπορεί να έχει έναν αριθμό ιδιοτήτων και κάθε ιδιότητα μπορεί να περιέχει λίστες από αντικείμενα. Τα έγγραφα αποθηκεύονται και ανακτώνται ως αντικείμενα JSON και γι' αυτό το λόγο η CouchDB μπορεί να υποστηρίξει τους τύπους δεδομένων:

- ***String*** (Αλφαριθμητικό)
- ***Number*** (Αριθμός)
- **Boolean** (Λογική έκφραση)
- ***Array*** (Πίνακας)

Κάθε έγγραφο στο CouchDB έχει ένα μοναδικό αναγνωριστή και επειδή το CouchDB χρησιμοποιεί αισιόδοξη αντιγραφή των δεδομένων στην πλευρά του εξυπηρετητή και στην πλευρά του πελάτη, κάθε έγγραφο έχει και έναν αναγνωριστή αναθεώρησης. Το αναγνωριστικό αναθεώρησης ανανεώνεται από το CouchDB κάθε φορά που το έγγραφο τροποποιείται. Οι λειτουργίες ενημέρωσης στο CouchDB γίνονται σε όλα τα έγγραφα. Αν ο πελάτης επιθυμεί να τροποποιήσει μία τιμή σε ένα έγγραφο, πρέπει το σύστημα να φορτώσει πρώτα το έγγραφο, να κάνει τις απαραίτητες αλλαγές σε αυτό και τελικά ο πελάτης να στείλει το συνολικό έγγραφο πίσω στη βάση δεδομένων. Το CouchDB χρησιμοποιεί το αναγνωριστικό αναθεώρησης μέσα στο έγγραφο για έλεγχο ταυτοχρονισμού, οπότε μπορεί να αναγνωρίσει αν κάποιος άλλος πελάτης έχει κάνει αλλαγές στο έγγραφο. Το μοντέλο ερωτημάτων στο CouchDB απαρτίζεται από δύο σενάρια: ένα είναι οι όψεις, που δημιουργούνται χρησιμοποιώντας τις λειτουργίες MapReduce και το άλλο σενάριο είναι το HTTP API ερωτημάτων, που επιτρέπει στους πελάτες να έχουν πρόσβαση και να κάνουν ερωτήματα στις όψεις. Η όψη στο CouchDB είναι βασικά μία συλλογή από ζευγάρια κλειδιών-τιμών, που είναι ταξινομημένα σύμφωνα με το κλειδί τους. Οι όψεις δημιουργούνται από καθορισμένες από το χρήστη μεθόδους MapReduce που καλούνται όταν το έγγραφο στη βάση δεδομένων ανανεώνεται ή δημιουργείται. Οι όψεις πρέπει να καθορίζονται πριν την εκτέλεση. Αυτό γίνεται με την εισαγωγή μίας νέας όψης, η οποία απαιτεί τη συνάρτηση MapReduce που θα κληθεί για κάθε έγγραφο της βάσης δεδομένων. Αυτός είναι και ο λόγος που το CouchDB δεν υποστηρίζει δυναμικά ερωτήματα.

5.3.2 Η αρχιτεκτονική του CouchDB

Το CouchDB έχει τρία κύρια συστατικά:

- ***Storage engine (Μηχανή αποθήκευσης):*** Η μηχανή αποθήκευσης είναι βασισμένη σε B-tree και είναι το κύριο συστατικό του συστήματος που διαχειρίζεται την αποθήκευση των εσωτερικών δεδομένων, των εγγράφων και των όψεων. Η πληροφορία στο CouchDB είναι προσβάσιμη μέσω κλειδιών ή διαστημάτων κλειδιών που δείχνουν απευθείας στις λειτουργίες του B-tree. Αυτή η απευθείας πρόσβαση έχει βελτιώσει σημαντικά την ταχύτητα.
- ***View engine (Μηχανή όψης):*** Η μηχανή όψης βασίζεται πάνω στο Mozilla SpiderMonkey και είναι γραμμένη με τη γλώσσα σεναρίων JavaScript. Επιτρέπει τη δημιουργία όψεων που γίνεται με MapReduce εργασίες. Οι ορισμοί των όψεων αποθηκεύονται σε έγγραφα σχεδίασης. Όταν ένας χρήστης διαβάζει δεδομένα σε μία όψη, το CouchDB διαβεβαιώνει ότι το αποτέλεσμα είναι ενημερωμένο. Οι όψεις μπορούν να χρησιμοποιηθούν για τη δημιουργία ευρετηρίων και την εξαγωγή δεδομένων από έγγραφα.
- ***Replicator (Αντιγραφέας):*** Ο αντιγραφέας είναι υπεύθυνος για την αντιγραφή των δεδομένων σε μία τοπική ή απομακρυσμένη βάση δεδομένων και για το συγχρονισμό των εγγράφων σχεδίασης. Αυτό το χαρακτηριστικό της βάσης δεδομένων είναι και βασικό στοιχειό για την υλοποίηση του προγράμματος που πραγματεύεται η διπλωματική εργασία.

5.3.3 Περιβάλλον διεπαφής

Το couchdb παρέχει βιβλιοθήκες για πολλές γλώσσες προγραμματισμού. Η γλωσσά προγραμματισμού που χρησιμοποιήσαμε για την υλοποίηση του προγράμματος είναι η Python. Για την Python υπάρχουν πέντε βιβλιοθήκες που επιτρέπουν τον έλεγχο των βάσεων δεδομένων. Οι δυο πιο διαδεδομένες είναι η couchdbkit και η couchdb-python. Εμείς επιλέξαμε τη couchdb-python αφού είναι και η πιο χρησιμοποιουμένη.

Πλατφόρμες στις οποίες λειτούργει το Couchdb:

- ***Windows***
- ***Linux***
- ***Mac OS X***
- ***Android***
- ***Φυλλομετρητές τελευταίας γενιάς μέσω του Pouchdb*** (MozillaFirefox,GoogleChrome,Safari,InternetExplorer)

5.3.4 Web εφαρμογές (couchapp)

Το couchdbπροσφέρει σαν δυνατότητα την εξυπηρέτηση αρχείων htmlκαι javascript, τα οποία αποτελούν από μόνα τους εφαρμογές, οι οποίες «τρέχουν» σε όλους τους συγχρόνους περιηγητές του διαδικτύου. Η διπλωματική βασίζεται σε μια από αυτές τις εφαρμογές (monocles), η οποία έχει αναπτυχτεί με τη λογική ενός αποκεντρωμένου κοινωνικού δικτύου. Η εφαρμογήmonocles θα αναλυθεί παρακάτω.

5.3.5 Αποκεντρωμένο κοινωνικό δίκτυοMonocles

Η εφαρμογή (couchapp) monocles είναι μια υλοποίηση του πρωτόκολλου που χρησιμοποιεί το diaspora βασισμένο σε html και javascript. Η αποθήκευση των δεδομένων γίνεται στη βάση δεδομένων couchdb και η εξυπηρέτηση των αρχείων γίνεται εξολοκλήρου από το couchdb. Χρησιμοποιεί τη λειτουργία replication του couchdb για τη δημιουργία “φίλων” στο κοινωνικό δίκτυο. Η εφαρμογή αυτή αναπτύχτηκε ως ένα ανοιχτού κώδικα εγχείρημα από το προγραμματιστή Max Ogden όπου αν και η ανάπτυξη του έχει σταματήσει εδώ και δυο χρόνια θεωρούμε ότι λόγο της μεγάλης διάδοσης των γλωσσών του διαδικτύου, στις οποίες βασίζεται η περαιτέρω ανάπτυξη του από τους χρήστες θα είναι κάτι εύκολο. Επιτρέπει όλες τις βασικές λειτουργίες ενός κοινωνικού δικτύου και χρησιμοποιούμε τον κώδικα που έχει αναπτυχτεί ως “proof of concept” για ένα αποκεντρωμένο κοινωνικό δίκτυο.

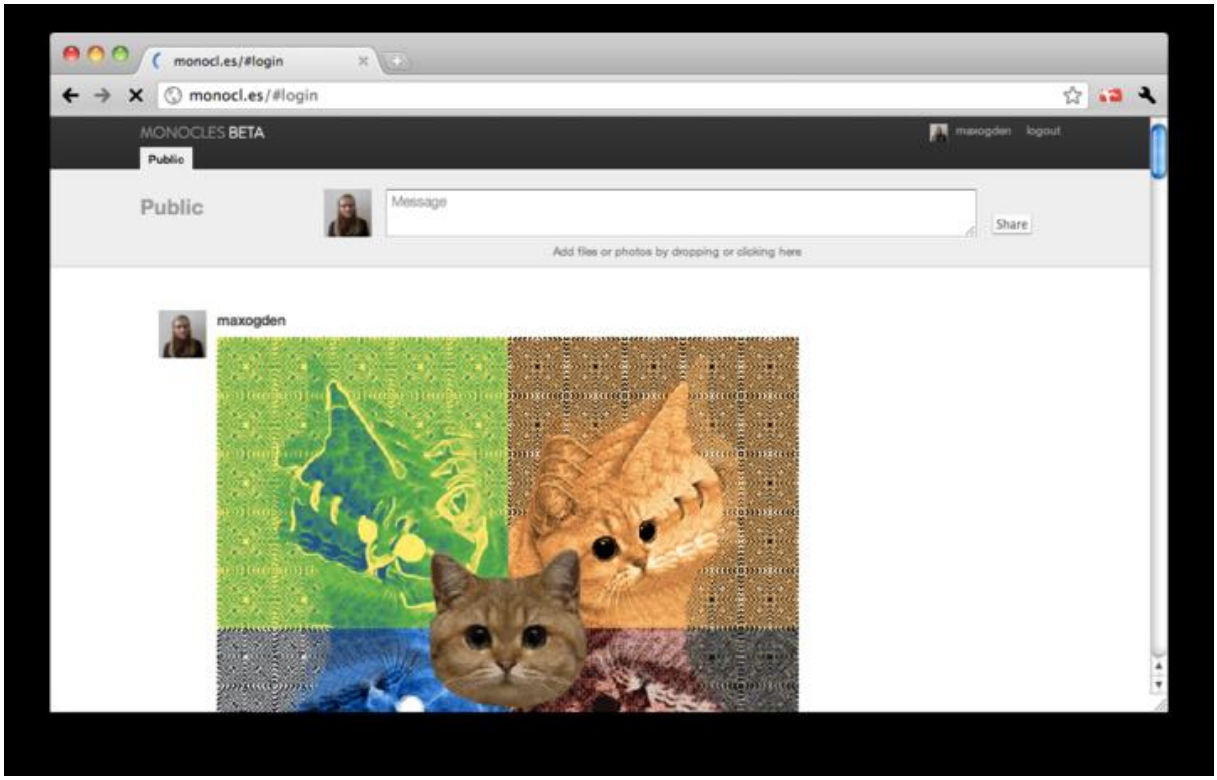


Figure 5.10 Το περιβάλλον διεπαφής του Monocles

5.4 Kadnode

Το Kadnode είναι ένα ανοιχτού κώδικα εγχείρημα από το προγραμματιστή moritz warning που βρίσκεται ακόμα σε πρώιμη φάση. Ο στόχος του εγχειρήματος είναι να δημιουργηθεί ένα αξιόπιστη εναλλακτική για μια αποκεντρωμένη υπηρεσία ονοματολογίας (DNS), η οποία θα βασίζεται εξολοκλήρου στο κατανεμημένο πίνακα hash (DHT) βάσει του οποίου λειτουργούν τα αποκεντρωτικά δίκτυα διαμοιρασμού bittorrent. Η λειτουργία του βασίζεται στη δυνατότητα αναγγελίας ενός πόρου κωδικοποιημένου σε μοναδικούς 40 χαρακτήρες, βάσει του αλγορίθμου md5 και στη δυνατότητα αναζήτησης αυτού του πόρου δίχως κάποια κεντρική υπηρεσία αναζήτησης (Εικ.5.2) για εύρεση της διεύθυνσης IP που το κατέχει.

Ουσιαστικά ο χρήστης επιλεγεί ένα όνομα χρήστη το οποίο ανακοινώνεται σαν πόρος στο DHT δίκτυο με κωδικοποίηση md5 και ύστερα κάποιος άλλος χρήστης μπορεί αναζητώντας το συγκεκριμένο όνομα χρήστη στο κατανεμημένο δίκτυο dht μπορεί να βρει την IP του χρήστη που την ανακοίνωσε.

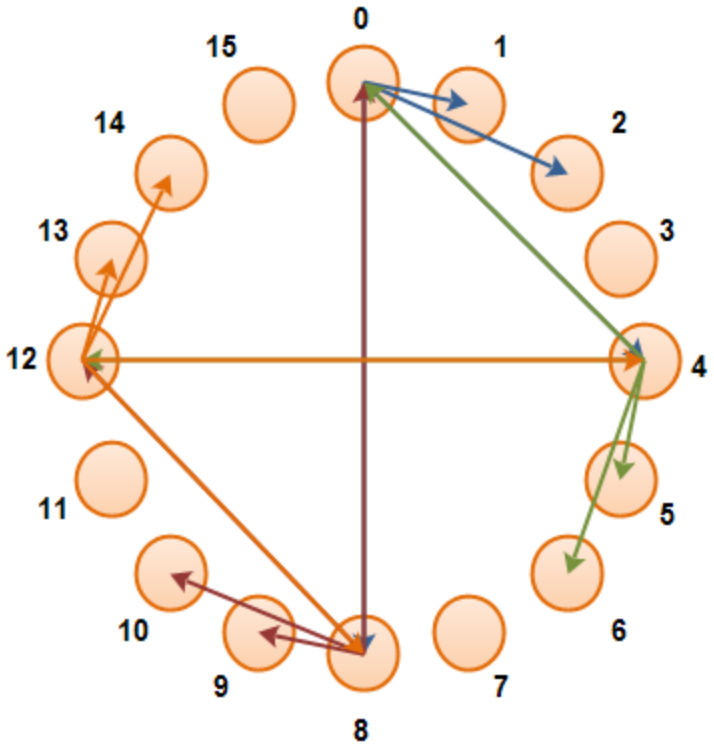


Figure 5.11 Αναζήτηση σε ένα δίκτυο Kademlia

Στη φάση ανάπτυξης που βρίσκεται το συγκεκριμένο εγχείρημα δεν έχει τη δυνατότητα να πιστοποιεί ότι ο χρήστης που ανακοίνωσε τον πόρο είναι και ο χρήστης που όντως αναζητείται. Γι' αυτό τον λόγο αντί της ανακοίνωσης ενός πόρου για ένα όνομα χρήστη χρησιμοποιείται ο αλγόριθμος sha512 για την ανακοίνωση 3 πόρων στο δίκτυο dht όπου η IP που θα βρεθεί και με τους τρεις πόρους θα είναι και με μεγαλύτερη πιθανότητα ο χρήστης, ο όποιος αναζητείται. Σε περίπτωση κακόβουλων ενεργειών, όπου κάποιος θα ανακοινώσει τους ίδιους πόρους για να υποδυθεί το πραγματικό χρήστη προτείνουμε τη χρήση δημοσίων και ιδιωτικών κρυπτογραφημένων κλειδιών, τα οποία θα ανταλλάσσονται μέσω ασφαλών μέσων ανάμεσα στους χρήστες που θα πιστοποιούν ότι κάποιος χρήστης στο δίκτυο είναι αυτός που αναζητείται μετά από κάποιο handshake μεταξύ των χρηστών. Η ασφαλής ανταλλαγή του δημόσιου κλειδιού μεταξύ των χρηστών μπορεί να επιτευχτεί και διαμοιράζοντας ένα torrent αρχείο που περιέχει το δημόσιο κλειδί κάποιου χρήστη το οποίο βάσει του hash που θα δημιουργηθεί θα διανέμεται μέσω κάποιας θεωρητικά έμπιστης επικοινωνίας μεταξύ των χρηστών. Αυτό πρακτικά είναι πολύ πιο εύχρηστο για τους χρήστες αφού το hash του δημοσίου κλειδιού είναι πολύ μικρότερο σε μήκος (40 χαρακτήρες) από το δημόσιο κλειδί (εκατοντάδες χαρακτήρες).

5.5 N2N ιδιωτικό εικονικό δίκτυο

Το N2N είναι ένας τύπος ιδιωτικού εικονικού δικτύου (vpn) που επιτρέπει τη δημιουργία αποκεντρωμένων ασφαλών δικτύων VPN μεταξύ χρηστών χωρίς τη χρήση κάποιου κεντρικού server. Αποτελείται από δυο συνιστώσες-προγράμματα:

- ***Supernode:*** Αποτελεί το πρόγραμμα μέσω του οποίου οι χρήστες ανακοινώνουν ότι θέλουν να συμμετέχουν σε ένα δίκτυο με κάποιο όνομα. Αποτελεί ένα σημείο

συνάντησης και συνεννόησης μεταξύ των επιμέρους χρηστών. Όλη η επικοινωνία που συμβαίνει μέσω του supernode είναι κρυπτογραφημένη και δεν μπορεί να υποκλαπεί. Η μόνη δυνατότητα για να συμμετέχει κάποιος σε ένα δίκτυο που λειτουργεί χρησιμοποιώντας το συγκεκριμένο supernode είναι να κατέχει τα κλειδιά κρυπτογράφησης της επικοινωνίας.

- ***Edge:*** Αποτελεί το πρόγραμμα το οποίο χρησιμοποιούν οι χρήστες για να συμμετέχουν σε ένα εικονικό δίκτυο το οποίο δημιουργεί μια εικονική διεπαφή επικοινωνίας αντίστοιχη μιας θύρας ethernet με δίκια του IP. Λαμβάνει ως παραμέτρους το όνομα της διεπαφής που θα χρησιμοποιηθεί, τη στατική διεύθυνση η οποία θα χρησιμοποιηθεί στο εικονικό δίκτυο, το όνομα του δικτύου, το κλειδί κρυπτογράφησης του δικτύου, και τη διεύθυνση IP του Supernode μέσω του οποίου θα συνδεθεί.

Ένας χρήστης θα τρέχει και τα δυο προγράμματα εφόσον το modem του υποστηρίζει upnp (plug 'nplay) άνοιγμα θυρών. Παρακάτω φαίνεται η δομή που διαμορφώνεται μεταξύ των supernodes και των edge πελατών (Εικ.5.3):

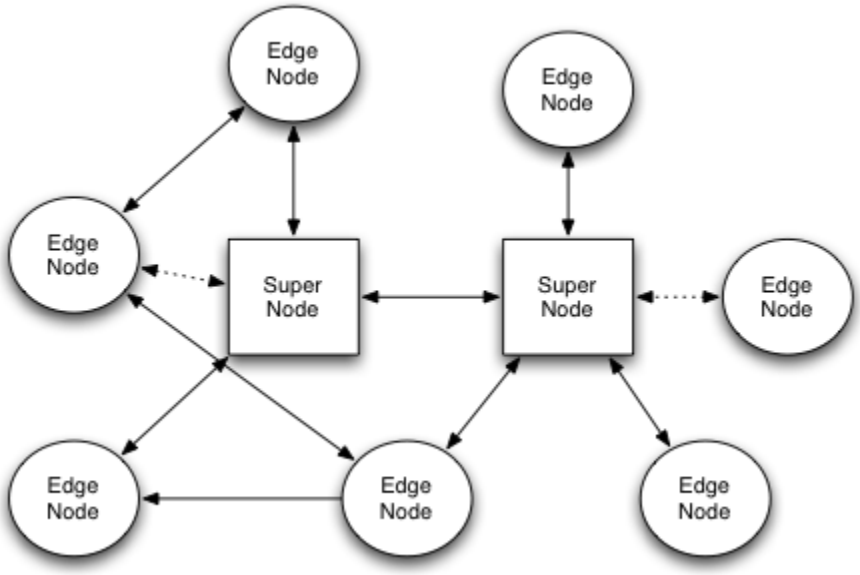


Figure 5.12 Γράφημα δομής του N2N

Το N2N δίνει τη δυνατότητα στους χρήστες να συνδέονται μεταξύ τους κρυπτογραφημένα διαπερνώντας πολλαπλά τείχη προστασίας που μπορεί να υπάρχουν στις συνδέσεις στο διαδίκτυο των χρηστών. Το μόνο που απαιτείται είναι ο χρήστης, ο οποίος θα λάβει το ρόλο του supernode να έχει δυνατότητα να προωθήσει μια θύρα στη σύνδεση του με το διαδίκτυο. Εμείς χρησιμοποιούμε την αρκετά διαδεδομένη λειτουργία των modem-router για upnp (plug and play) προώθηση θυρών έτσι ώστε να είναι όσο το δυνατόν πιο αυτοματοποιημένη η επικοινωνία μεταξύ των χρηστών.

Επιλέξαμε να χρησιμοποιούνται τουλάχιστον δυο supernodes μεταξύ των χρηστών έτσι ώστε να λυθεί το ζήτημα των δυναμικών IP που αφορά και τους περισσότερους χρήστες του

διαδικτύου. Θεωρούμε ότι η πιθανότητα να αλλάξει η διεύθυνση IP ταυτόχρονα και στους δυο χρήστες είναι πολύ μικρή, όποτε και η απρόσκοπτη επικοινωνία μεταξύ των χρηστών έχει πολύ μικρές πιθανότητες αποτυχίας. Σε περίπτωση που όντως αλλάξουν ταυτόχρονα οι διευθύνσεις IP και των δυο χρηστώνεπαναλαμβάνεται η αναζήτηση στο δίκτυο DHT έτσιώστε να βρεθούν οι νέες IP των χρηστών. Αυτόέχει ως αποτέλεσμα την απώλειαεπικοινωνίαςμεταξύ των χρηστών για περίπουμισήώρα που θεωρούμεότιείναιπολύμικρήαπώλεια σε σχέση με τη μικρήπιθανότηταυτής και των πλεονεκτημάτων που προσφέρει για τις επικοινωνίες των χρηστών το σύστημα που αναπτύσσουμε.

5.6 Raspberry pi

Το Raspberry pi είναιέναςυπολογιστήςμεγέθουςπιστωτικήςκάρτας (Εικ. 5.4) που αναπτύχτηκε στο ΗνωμένοΒασίλειοαπό το Raspberry pi Foundation με τη πρόθεση να προωθήσει τη διδασκαλία της επιστήμης των υπολογιστών στα σχολεία. Το Raspberry pi έχει Broadcom BCM2835 system on a chip (SoC) με εναν ARM1176JZF-S 700 MHz επεξεργαστή, και VideoCore IV GPU. Η μνήμη RAM στην αρχικήέκδοσηήταν στα 256 MB που αργότερα αναβαθμίστηκε στα 512 MB. Χρησιμοποιεί μια SD κάρτα για αποθήκευση του λειτουργικούσυστήματος και γενικότεροαποθηκευτικόχώρο.

Figure 5.13RaspberryPi

Το Raspberry pi τρέχειλειτουργικάσυστήματαβασισμένα στο πυρήνα linux και έχει ήδη αρκετές εκδόσεις του λειτουργικού συστήματος που μπορεί να υποστηρίξει. Μερικές από αυτές είναι :

- ***Arch-linux***
- ***Debian***
- ***Gentoo***
- ***Slackware***

Εμείς θα χρησιμοποιήσουμε την διανομή moebius(headless debian) γιατί αποτελεί μια σταθερή και εύκολη στη χρήση διανομή linux.

5.7 Επίλογος

Το σύστημα που προτείνω έχει πολλές χρήσεις. Μπορεί να δημιουργήσει ένα ανεξάρτητο κρυπτογραφημένο δίκτυο μεταξύ χρηστών που δε θέλουν να εμπιστευτούν τα δεδομένα τους σε κάποια εταιρία. Μπορεί να χρησιμοποιηθεί εσωτερικά σε μια επιχείρηση έτσι ώστε να συγχρονίζουν συνεχώς τα δεδομένα που χρειάζονται κατά τόπους οι αντιπροσωπείες. Μπορεί να χρησιμοποιηθεί μεταξύ επιχειρήσεων για δεδομένα που είναι κοινά. Μπορεί να χρησιμοποιηθεί για σημερινές ιστοσελίδες που θέλουν να έχουν παράλληλα μια peer-to-peer μορφή της ιστοσελίδας τους στηριγμένη από τους χρήστες της.

Οι εφαρμογές που μπορεί να εγκαταστήσει ο χρήστης κυμαίνονται από κοινωνικές εφαρμογές που ο χρήστης διαμοιράζεται δεδομένα με άλλους χρήστες μέχρι τελείως προσωπικές εφαρμογές που μόνο ο χρήστης έχει δικαίωμα πρόσβασης. Οι εφαρμογές που έχουν ήδη αναπτυχτεί είναι πάρα πολλές και μπορουν να βρεθουν στην ιστοσελιδα couchapp.org. Μερικά παραδείγματα αυτών είναι : εφαρμογή κοινωνικής δικτύωσης monocles (αντίστοιχη του Facebook), εφαρμογή για blog(blueink), εφαρμογή για παρουσιάσεις (boom amazing, couchdb projector), εφαρμογή αποθήκευσης επαφών (couchdb contact form), εφαρμογή συνεργατικής ζωγραφικής, εφαρμογή chat βάσει τοποθεσίας, εφαρμογή φόρουμ (modern forum), εφαρμογή αποθήκευσης δημοσιεύσεων στο Twitter, εφαρμογή για τη δημιουργία wiki(pages), εφαρμογή δημιουργίας στατιστικών πωλήσεων για επιχειρήσεις (sales stats), εφαρμογή chat(toast).

Οι εφαρμογές που αναφέρονται είναι ως επί το πλείστον γραμμένες σε γλωσσά που μπορούν να κατανοήσουν και να τρέξουν οι σημερινοί σύγχρονοι περιηγητές και τα δεδομένα που παρουσιάζουν παρέχονται από τη βάση δεδομένων couchdb. Η βάση δεδομένων couchdb παρέχει και την διαχείριση των χρηστών.

5.8 Μελλοντική ανάπτυξη

Οι δυνατότητες που μας δίνει ο ασφαλής συγχρονισμός των κόμβων των χρηστών και με τη χρήση των HTML5 εφαρμογών είναι ότι επειδή το couchdb είναι φτιαγμένο για το διαδίκτυο, μπορούμε να μεταφέρουμε το διαδίκτυο εκεί. Να χρησιμοποιούμε σαν πύλη του διαδικτύου το κόμβο μας και μέσα από αυτό να γνωρίζουμε ότι συμβαίνει στο ομότιμο δίκτυο και παράλληλα να βλέπουμε και το κεντρικοποιημένο κομμάτι ανώνυμα και χωρίς παρακολούθηση. Το cloud που οραματιστήκαν οι εταιρίες είναι φτιαγμένο για να τρέχει στους υπολογιστές της κάθε εταιρίας. Το cloud που μπορεί να δημιουργηθεί με χρήση του couchdb είναι απολύτως ιδιωτικό και ασφαλές.

Επίσης, οι εξελίξεις που συμβαίνουν στα λειτουργικά συστήματα των "έξυπνων" κινητών τηλεφώνων με τη δημιουργία του FirefoxOS και του Tizen φέρνουν τις HTML5 δυνατότητες στο προσκήνιο. Αυτό σημαίνει ότι το λειτουργικό στο κινητό θα μπορούσε να αποτελεί απλά το front-end του κόμβου του. Κατά μια έννοια το σύστημαμπορεί να αποτελέσει ο πράκτορας του χρήστη στο διαδίκτυο που του φέρνει τα δεδομένα που ζητάει.

Πώς μπορεί το ομότιμο να αντιγράφει το κεντρικοποιημένο;

Μέχρι στιγμής αυτό που έχει προσπαθήσει σε δίκτυα ανωνυμίας όπως το TOR είναι να δημιουργείται μια κρυπτογραφημένη γέφυρα μεταξύ τυχαίων χρηστών και να μεταφέρεται έτσι ένα αίτημα σε μια ιστοσελίδα με ασφάλεια. Αυτό του το πλεονέκτημα έχει το μειονέκτημα ότι η όλη διαδικασία είναι αργή.

Scrapers: Οι scrapers είναι προγράμματα τα όποια εξάγουν το ουσιώδες περιεχόμενο μιας ιστοσελίδας βάσει διαμόρφωσης που έχει κάνει ο χρήστης. Η παραμετροποίηση κάθε σελίδας είναι διαφορετική αλλά με τους visual scraper το να φτιάξει κάποιος ένα template για την αγαπημένη του σελίδα είναι εργασία 5 λεπτών. Και ύστερα μπορεί να τη μοιραστεί και με άλλους

Για παράδειγμα σε μια ειδησεογραφική ιστοσελίδα οι ειδήσεις αποτελούν το μικρότερο σε μέγεθος κομμάτι της σελίδας. Το μεγαλύτερο κομμάτι αποτελούν τα επισυναπτόμενα αρχεία javascript και κομμάτια html που επαναλαμβάνονται στη διάρκεια περιήγησης στη σελίδα. Αν αποθηκεύουμε σε μορφή Json το κείμενο των ειδήσεων και το συγχρονίζουμε μεταξύ των κόμβων με τη χρήση ενός couchapp, μπορούμε να παρουσιάζουμε τις ειδήσεις.

Σε περίπτωση που μια ιστοσελίδα θέλει να έχει και την peer-to-peer εκδοχή της θα μπορεί να αποθηκεύει τα δεδομένα της σε μορφή json στο couchdb και αυτά να μοιράζονται μέσα από το δίκτυο DHT. Χρησιμοποιώντας κρυπτογράφηση ιδιωτικού δημόσιου κλειδιού μπορούν να εξασφαλίζουν την εγκυρότητα των δεδομένων που κυκλοφορούν.

6ο Κεφάλαιο

6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλύσουμε τη διαδικασία που ακολουθήσαμε για να υλοποιήσουμε το σύστημα. Η υλοποίηση θα γίνει με σκοπό τη δημιουργία ενός κατανεμημένου κοινωνικού δικτυού βασισμένο στο couchapp monocles. Η υλοποίηση για αλλού είδους εφαρμογές, όπως blog , task manager κλπ είναι παρόμοια.

6.2 Hardware

Χρησιμοποιήσαμε τον μικροϋπολογιστή Raspberry pi λόγο της χαμηλής τιμής του που τον κάνει προσβάσιμο οικονομικά σε μεγαλύτερο κομμάτι του παγκοσμίου πληθυσμού. Επίσης, παρέχει και την απαραίτητη επεξεργαστική ισχύ για να μπορεί να "τρέξει" τα προγράμματα που θα αναφέρουμε.

Το παράδειγμα εφαρμογής βασίζεται σε δυο τέτοιους μικροϋπολογιστές που είναι συνδεδεμένοι σε δυο modem σε διαφορετικά σπίτια με καλώδιο ethernet. Αυτή η υλοποίηση αποδεικνύει τη συνεργασία των μικροϋπολογιστών και του προγράμματος μέσω του διαδικτύου. Είναι η μικρότερη δυνατή κλίμακα υλοποίησης που γίνεται λόγω του κόστους που θα επέφερε μια περαιτέρω κλιμάκωση.

Μια κάρτα SD αποτελεί τον αποθηκευτικό χώρο του λειτουργικού και των προγραμμάτων και γενικά των δεδομένων.

6.3 Software

6.3.1 Προετοιμασία

Η υλοποίηση του κομματιού που άφορα το software ξεκινάει με την εγκατάσταση του λειτουργικού προγράμματος Moebius που αποτελεί μια ελαφριά έκδοση του debian linux για το raspberrypi. Το Moebius είναι μια ελαφριά διανομή linux με τα ελάχιστα προγράμματα εγκατεστημένα. Επιλέχτηκε λόγω της ιδιαίτερης δυνατότητας της να παραμετροποιηθεί με τα απολύτως απαραίτητα πακέτα προγραμμάτων για το σύστημα που αναπτύξαμε. Η εγκατάσταση έγινε σε έναν προσωπικό υπολογιστή με λειτουργικό σύστημα Ubuntu. Ύστερα, η κάρτα SD εισήχθηκε στο μικροϋπολογιστή, ο οποίος συνδέθηκε με καλώδιο Ethernet στο modem της οικίας που γίνεται η υλοποίηση. Ο υπολογιστής παραμένει headless,δηλαδή χωρίς οθόνη, πληκτρολόγιο και ποντίκι. Ο έλεγχος στη συσκευή γίνεται μέσω SSH από το προσωπικό υπολογιστή. Με αυτό τον τρόπο θα καταφέρουμε μέσω τερματικού να εγκαταστήσουμε τα προγράμματα που θα αναφέρουμε παρακάτω,

Η γλώσσα που προτιμήσαμε να βασιστούμε για την ανάπτυξη του προγράμματοςείναι η γλώσσα Python. Είναι μια αρκετή δημοφιλής γλώσσα για την οποία παρέχονται περιβάλλοντα διεπαφής για όλα τα επιμέρους προγράμματα που θα χρησιμοποιήσουμε. Η εγκατάσταση της μέσω τερματικού γίνεται με την εντολή

Sudo apt-get install python

Ύστερα εγκαθιστούμε το couchdb που αποτελεί και το πακέτο διαχείρισης των βάσεων δεδομένων αλλά και των javascript εφαρμογών πάλι μέσω τερματικού με την εντολή

sudoapt-getinstallcouchdb

Για την επικοινωνία της γλώσσας python με το couchdb θα χρειαστεί να εγκαταστήσουμεέναεπιπλέονδιαχειριστήπακέτων. Οπότεγραφούμε στο τερματικό**sudoapt-getinstall python-pip**. Μετάεγκαθιστούμε το πακέτο**couchdb-python** με την εντολή**pip install couchdb**.

Μέχρι στιγμής έχουμε τα απαραίτητα προγράμματα για να ελέγξουμε τις βάσεις δεδομένων του couchdb μέσω της γλώσσας python. Για να μπορέσει το couchdb να επικοινωνεί με άλλους μικροϋπολογιστές που έχουν εγκατεστημένο το couchdb μέσω του κατανεμημένου πίνακα κατακερματισμού DHT θα πρέπει να εγκαταστήσουμε το Kadnode. Αυτό γίνεται με τις εντολές στο τερματικό:

- ***sudo apt-get install git-core***
- ***git clone https://github.com/mwarning/KadNode/***
- ***cd KadNode***
- ***make***
- ***sudo make install***

Για να γίνει εφικτή η κρυπτογραφημένη σύνδεση μεταξύ των χρηστών πρέπει να εγκαταστήσουμε το πρόγραμμα n2n που επιτρέπει τη δημιουργία peer-to-peer εικονικών δικτύων.

- ***sudo apt-get install n2n***

Τελευταίο πακέτο που χρειάζεται να εγκαταστήσουμε είναι το web.py web framework. Το πακέτο αυτό μας δίνει τη δυνατότητα να δημιουργήσουμε μια ιστοσελίδα που θα αποτελεί το περιβάλλον δημιουργίας, αναζήτησης και αποδοχήςχρηστών. Αυτό γίνεται με την εντολή

- ***sudo pip install web.py***

Η εγκατάσταση των javascript εφαρμογών θα γίνει πηγαίνοντας στη διεύθυνση**http://kan.so/docs/Setting_Up_A_Garden** και ακλουθώντας τις οδηγίες που αναγράφονται αποκτούμε το δικό μας market στο μικροϋπολογιστή μας. Ύστερα εγκαθιστούμε την εφαρμογή monocles που μας προσφέρει ένα κοινωνικό δίκτυο.

Έχοντας εγκαταστήσει όλα τα πακέτα που χρειάζονται για να είναι εφικτή η συνεργασία μεταξύ των επιμέρους προγραμμάτων τώρα μπορούμε να αναπτύξουμε τον κώδικα που θα πραγματοποιεί την επικοινωνίαμεταξύ του couchdb , του kadnode, του n2n και του web.py.

Ξεκινάμε με το κώδικα σε γλώσσα python ο οποίοςχωρίζεται σε τρεις τομείς:

- Δημιουργία χρήστη και ανακοίνωση στον πίνακα κατανεμημένου κατακερματισμούDHT
- Ανάγνωση και αποδοχή αιτημάτων φιλίας από άλλους χρήστες
- Αναζήτηση μέσω DHT και αποστολή αιτήματος φιλίας σε κάποιο χρήστη

Επίσης, έχουμε αναπτύξει ένα κομμάτι κώδικα που αναλαμβάνει να ανανεώνει τις εξωτερικές IP των χρηστών σε περίπτωση που είναι δυναμικές έτσι ώστε να μειωθεί η πιθανότητα απώλειας επικοινωνίας μεταξύ των χρηστών. Αυτό το κομμάτι κώδικα θα αναλυθεί σε επόμενο υποκεφάλαιο.

6.3.2 Κυρίως πρόγραμμα

import web	βιβλιοθήκη που χρειάζεται για την εξυπηρέτηση των αρχείων html και javascript
importsubprocess	βιβλιοθήκη για την έναρξη εξωτερικών διεργασιών
importcouchdb	βιβλιοθήκη για την επικοινωνία με τη βάση δεδομένων

couchdb

importtime βιβλιοθήκη απαραίτητη για να γίνονται παύσεις κάποιων δευτερόλεπτων στο κώδικα

fromuuidimportuuid4 βιβλιοθήκη που παράγει τυχαία νήματα 40 χαρακτήρων

fromhashlibimportsha512 βιβλιοθήκη για δημιουργία hash νημάτων από αλλά νήματα

fromcollectionsimportCounter βιβλιοθήκη που διαβάζει μια συλλογή νημάτων και τη χρησιμοποιούμε για να βρούμε το νήμα που εμφανίζεται περισσότερες φόρες

supernode_port="50020" αρχικοποιούμε τη θύρα στην οποία θα δέχεταισυνδέσεις το supernode

subprocess.call(["kadnode","--daemon"]) ξεκινάμε το kadnode

```
time.sleep(10)
subprocess.call(["kadnode-ctl","import","bttracker.debian.org"])
```

συνδεόμαστε σε ένα δημόσιο peer για να αρχικοποιήσουμε τη σύνδεση στο DHT

time.sleep(20)

```
urls = ('/create_user', 'create_user',
                '/find_friend/?','find_friend',
                '/accept_friend/?','accept_friend',)
```

ενημερώνουμε το webserver ποιες διευθύνσεις πρέπει να εξυπηρετεί

render = web.template.render('templates/') ενημερώνουμε το webserver σε ποιο φάκελο θα βρει τα αχρεία html

app = web.application(urls, globals())

#αρχικοποιούμε κάποιες μεταβλητές που θα χρησιμοποιηθούν στη συνεχεία

```
username=""
usernames=[]
```

#δημιουργούμε τις φόρμες εισαγωγής δεδομένων για τα δεδομένα που θα δέχονται

```
my_form = web.form.Form(
web.form.Textbox('',                                    class_='username',
id='username',value='Username '),
web.form.Textbox('',                                    class_='password',
id='password',value='password'),
            )
friend_form=web.form.Form(
                web.form.Textbox('',                        class_='friend',
id='friend',value='Friend..'),
                web.form.Dropdown('usernames', usernames)
                )
```

συνάρτηση που δέχεται ως input ένα νήμα-πόρο και μετά από αριθμό προσπαθειών επιστρέφει τις διευθύνσεις ip που βρέθηκε στη κατοχή τους το συγκεκριμένο νήμα

```
def find_friend_ip(domain):
        trynum=0
        supernodes=""
        while True:
                try:
                        if trynum==20:
                                return supernodes

        supernodes=subprocess.check_output(["kadnode-
ctl","lookup",domain])

                        return supernodes
                        break
                except subprocess.CalledProcessError, e:
                        time.sleep(5)
                        trynum=trynum+1
                        continue
```

```
class create_user:
```

κλάση για τη δημιουργίανέουχρήστη

```
        defGET(self):
```

συνάρτηση που επιστρέφει τη φόρμα νέου χρήστη στο φυλλομετρητή

```
                form = my_form()
                return render.create_user(form, "Here you can create
a new user.Please input your username and password.The username is
used for your friends to find you.")
```

```
        defPOST(self):
```

συνάρτηση που δέχεται το όνομα και το κώδικααπό το φυλλομετρητή του χρήστη και

προσπαθεί να τον ανακοινώσει στο DHT

```
form = my_form()
form.validates()

username = form.value['username']
password = form.value['password']
```

```
subprocess.Popen(["supernode","-l",supernode_port])
```

ξεκινάμε ένα supernode

```
subprocess.call(["upnpc","-r",supernode_port,"UDP"])
```

ανοίγουμε τη θύρα επικοινωνίας UDP του supernode στο modem-router

```
subprocess.call(["upnpc","-r",supernode_port,"TCP"])
```

ανοίγουμε τη θύρα επικοινωνίας TCP του supernode στο modem-router

```
hex_dig = sha512(username).hexdigest()
```

εξάγουμε ένα νήμα 120 χαρακτήρωναπό το όνομα χρήστη βάσει του αλγορίθμου sha512

αρχικοποίηση μεταβλητών απαραιτήτων για μετά

```
trynum=0
supernodeips=""
b=0
al=""
ips=[]
userhashes=[]
```

δημιουργούμε μια λούπα που σπάει το παραγμένο νήμα 120 χαρακτήρων σε 3 κομμάτια των 40 χαρακτήρων, τα οποία αποτελούν και πόρους που αναζητούμε στο DHT.Αν βρεθούν 2 από τους 3 πόρους στην ιδία διεύθυνση IP τότε ενημερώνουμε τον χρήστη ότι πρέπει να επιλέξει άλλο όνομα χρήστη γιατί αυτό που εισήγαγε ήδη χρησιμοποιείται

```
                for i in xrange(1,121):
                        b=b+1
                        al=al+hex_dig[i]
                        if b==40:
                                userhashes.append(al)

        supernodeips=supernodeips+find_friend_ip(al)
                                al=""
                                b=0
                for i in supernodeips.split("\n"):
                        if i!="":
                                ips.append(i.split(":")[0])
                valid=Counter(ips).most_common(1)
                if valid==[]:
                        print "cool"
                elif valid[0][1]>1:
                        return "Another user found with the same
username.Please use numbers within your username"
```

αν το όνομαχρήστη δε χρησιμοποιείται εξάγουμε μια τοπική IP από το όνομα χρήστη βάσει της οποίας θα συνδεθούμε στο τοπικό supernode

```
                lastipoct=0
                for letter in username:
                        lastipoct=lastipoct+ord(letter)
                lastipoct=lastipoct%253
                myip=str(lastipoct)+".0.0."+str(lastipoct)
```

αρχικοποιούμε τη σύνδεση με το couchdb

```
                couch                                                =
couchdb.Server("http://admin:admin@localhost:5984/")
```

δημιουργούμε ένα νέο χρήστη με το όνομα και κωδικό που επέλεξε στη βάση δεδομένων

```
                users=couch["_users"]
                users["org.couchdb.user:"+username]={"_id":
"org.couchdb.user:"+username,      "name":      username,      "roles":
["admin"],  "type": "user",  "password": password}
```

εισάγουμε το νέο χρήστη σε μια βάση δεδομένων για να μπορούμε να το αναζητήσουμε μετά. Αυτό γίνεται με τη προοπτική ότι θα μπορούν πολλοί χρήστες να δημιουργήσουν λογαριασμό στον ίδιο κόμβο.

```
profiles=couch["profiles"]
profiles[username]={"username":username}
```

ανακοινώνουμε τα 3 παραγμένα νήματααπό το όνομα χρήστη στο δίκτυο dht

```
dht_port=0
for userhash in userhashes:
    dht_port=dht_port-1
    subprocess.call(["kadnode-
ctl","announce",userhash,str(int(supernode_port)+dht_port),"100000
0"])
    time.sleep(2)
```

δημιουργούμε με το edge ένα νέο interface όπου μπορούν άλλοι χρήστες να συνδεθούν γνωρίζοντας μόνο το username

```
subprocess.call(["edge","-d","public","-a",myip,"-
c","public-"+username,"-k","public","-
l","localhost:"+supernode_port])
```

δημιουργούμε ένα νέο χρήστη που έχει δικαιώματα εγγραφής σε μια βάση,όπου θα αποθηκεύονται τα αιτήματα φιλίας

```
users["org.couchdb.user:public-"+username]={"_id":
"org.couchdb.user:public-"+username, "name": "public-
"+username, "roles": ["user"], "type": "user", "password":
"public"}
couch.create("public-"+username)
public=couch["public-"+username]

try :

    public["_security"]={"admins":{"names":["public-
"+username],"roles":[]},"members":{"names":["public-
"+username],"roles":[]}}
except:
    print "ok"
```

εφόσον ο λογαριασμός του χρήστη δημιουργήθηκε επιστρέφουμε στο φυλλομετρητή το παρακάτω μήνυμα

```
return "User created.Proceed to next step"
```

classfind_friend: κλάση για την αναζήτησηφίλων

defGET(self): όταν ζητηθεί από τον χρήστη να αναζητήσει ένα νέο φίλο,επιστρέφει τα προφίλ που έχουν δημιουργηθεί σε αυτό το κόμβο βάσει του όποιου θα γίνει ένα αίτημα φιλίας,

```
couch = couchdb.Server("http://admin:admin@localhost:5984/")
profiles=couch["profiles"]
usernames=[]
for i in profiles:

usernames.append((profiles[i]["username"],profiles[i]["username"]))
friend_form=web.form.Form(
web.form.Textbox('', class_='friend', id='friend',value='Friend..'),
web.form.Dropdown('usernames', usernames)
)
form = friend_form()
return render.find_friend(form, "Here you can find a new friend.Input the username of your friend and I will search in the cloud.")
```

defPOST(self):

όταν ο χρήστηςαποστείλει το προφίλ,βάσει του όποιου θα γίνει το αίτημαφιλίαςαυτή η συνάρτηση αναλαμβάνει την αναζήτηση και εφόσον βρεθεί την αποστολή αιτήματος φιλίας στονχρήστη που αναζητήθηκε

```
form = friend_form()
form.validates()

friend = form.value['friend']
username=form.value['username']
invalid=True
```

εφόσονβρεθεί το όνομα που αναζητήθηκε με πιθανότητα εγκυρότητας 66% από τη λούπα αυτή επιστρέφεται η διεύθυνσηIP

```
while invalid:
                        trynum=0
                        hex_dig = sha512(friend).hexdigest()
                        b=0
                        al=""
                        supernodeips=""
                        ips=[]
                        userhashes=[]
                        for i in xrange(1,121):
                                b=b+1
                                al=al+hex_dig[i]
                                if b==40:
                                        userhashes.append(al)

        supernodeips=supernodeips+find_friend_ip(al)
                                        al=""
                                        b=0
                        for i in supernodeips.split("\n"):
                                ips.append(i.split(":")[0])

                        print ips
                        friendip=Counter(ips).most_common(1)[0][0]
                        print "friends ip is "+friendip
                        if friendip!="":
                                for i in supernodeips.split("\n"):
                                        if friendip == i.split(":")[0]:
                                                friendsport =
str(int(i.split(":")[1])+(10-int(i.split(":")[1])%10))
                        if Counter(ips).most_common(1)[0][1]>1:
                                invalid=False
```

εδώεξάγουμε τις τοπικέςδιευθύνσειςIP που θα χρησιμοποιηθούν για το δημόσιο και ιδιωτικόειεικονικόδίκτυομεταξύ των χρηστών

```
lastipoct=0
for letter in friend:
        lastipoct=lastipoct+ord(letter)
melastipoct=0
for letter in username:
        melastipoct=melastipoct+ord(letter)
combinedoct=(melastipoct+lastipoct)%253
melastipoct=melastipoct%253
lastipoct=lastipoct%253

mixedip=str(combinedoct)+".0.0."+str(lastipoct+1)

myip=str(lastipoct)+".0.0."+str(lastipoct+1)

friendslocalip=str(lastipoct)+".0.0."+str(lastipoct)
```

```
friendship_password=uuid4().hex
```

δημιουργείταιέναςτυχαίοςκωδικός για την κρυπτογράφησημεταξύ των χρηστών.

Παρακάτω δημιουργείται ένας νέος λογαριασμός για τον φίλο που στον οποίο θα σταλθεί αίτημα φιλίας

```
couch = couchdb.Server("http://admin:admin@localhost:5984/")
users=couch["_users"]
users["org.couchdb.user:"+friend]={"_id": "org.couchdb.user:"+friend, "name": friend, "roles": ["admin"], "type": "user", "password": friendship_password}
```

δημιουργούμε μια βάσηδεδομένων για το νέοφίλο για μετέπειτα χρήση,όπως επίσης δίνουμε δικαιώματα εγγραφής στο νέοφίλο

```
couch.create(friend)
frienddb=couch[friend]


try :

    frienddb["_security"]={"admins":{"names":[friend],"roles":[]},"members":{"names":[friend],"roles":[]}}
except:
        print "ok"
```

συνδεόμαστε στο δημόσιοεικονικόδικτυού του φίλου που βρέθηκε

```
                subprocess.Popen(["edge","-d","edge2","-a",myip,"-
c","public-"+friend,"-k","public","-l",friendip+":"+friendsport])
                time.sleep(20)
                try:
```

βρίσκουμε τον αριθμό της διεργασίας edge που δημιουργήσαμε για μετέπειτακλείσιμο

```
pid=subprocess.check_output(["pgrep","-f","public-"+friend])
except:
                        print "ss"
                pid=pid.split("\n")[0]
```

συνδεόμαστε στη δημόσια βάση δεδομένων και εισάγουμε ένα νέο αίτημα φιλίας

```
                friendcouch=couchdb.Server("http://public-
"+friend+":public@"+friendslocalip+":5984")
                frienddb=friendcouch["public-"+friend]
                doc=uuid4().hex

        frienddb[doc]={"friend":username,"password":friendship_pa
ssword,"supernode_port":supernode_port}
```

καταργούμε τη διεργασίαedge που ξεκινήσαμεπροηγουμένως,γιατίπλέον δεν χρειάζεται

```
                subprocess.call(["kill",pid])
```

να δημιουργούμε τη κοινή βάση δεδομένωνβάσει της οποίας θα επικοινωνούμε,

```
                couch.create(username+"-"+friend)
                mergeddb=couch[username+"-"+friend]
```

βρίσκουμε την εξωτερική IP της σύνδεσης μας στο διαδίκτυο,

```
                find_ip_cmd="upnpc -s | grep ExternalIPAddress | sed
's/[^0-9\.]//g'"

        my_external_ip=subprocess.check_output(find_ip_cmd,shell=
True)
                my_external_ip=my_external_ip.split("\n")[0]

        mergeddb["ips"]={username:my_external_ip,friend:friendip}
```

δίνουμε δικαιώματα εγγραφής στο φίλο που δημιουργήσαμε,

```
                try :

        mergeddb["_security"]={"admins":{"names":[friend],"roles"
:[]},"members":{"names":[friend],"roles":[]}}
                except:
                        print "ok"
```

δημιουργούμε μια νέα καταχώρηση στη βάση δεδομένων των φίλων για να ενημερώνεται η IP εφόσον αλλάξει μέσω του script update_ip

```
                friendsdb=couch["friends"]
                doc=uuid4().hex

        friendsdb[doc]={"edgepid":"","friend":friend,"ip":friendip,"
localip":mixedip,"network":username+"-
"+friend,"password":friendship_password,"port":friendsport,"super
node_port":supernode_port,"username":username,"initiated_by":use
rname}
```

επιστρέφουμε στο φυλλομετρητή του χρήστη την IP του φίλου που δημιουργήσαμε,

```
                return supernodeips
```

```
class accept_friend:
```

η κλάση αυτή χρησιμοποιείται για την αποδοχή αιτημάτων φιλίας,

όταν ο χρήστης ζητήσει να δει τα αιτήματαφιλίας , επιλέγει το προφίλ που δέχτηκε αιτήματα φιλίας,

```
def GET(self):
        couch                                                   =
couchdb.Server("http://admin:admin@localhost:5984/")
                profiles=couch["profiles"]
                usernames=[]
                for i in profiles:

        usernames.append((profiles[i]["username"],profiles[i]["usern
ame"]))
                friend_form=web.form.Form(
web.form.Dropdown('usernames', usernames)
            )
                form = friend_form()
                return render.find_friend(form, "choose your user.")
```

εφόσον ο χρήστης έχει επιλέξει προφίλ, του επιστρέφονται τα αιτήματα φιλίας και μπορεί να τα αποδεχτεί,

```
        def POST(self):
                form = friend_form()
                form.validates()

                friend = form.value['friend']
                username=form.value['username']
                friends=[]
                if friend=="":
                        couch                                                   =
couchdb.Server("http://admin:admin@localhost:5984/")
                        friendrequests=couch["public-"+username]
                        for i in friendrequests:

        friends.append((friendrequests[i]["friend"],friendrequests[i]["
friend"]))
                        form=web.form.Form(
web.form.Dropdown('friend', friends)
            )

                        return render.find_friend(form, "accept a
friend.")
                else:
```

η περίπτωση που έχειαποδεχτείκάποιοαίτημαφιλίας

```
                        couch                                                   =
couchdb.Server("http://admin:admin@localhost:5984/")
                        friendrequests=couch["public-"+username]
                        for i in friendrequests:
                                if friend==friendrequests[i]["friend"]:

        friendship_password=friendrequests[i]["password"]
```

εξάγουμε τις τοπικέςIP που θα χρησιμοποιηθούν στο ιδιωτικό εικονικό δίκτυο μεταξύ των χρηστών

```
                    lastipoct=0
                    for letter in friend:
                            lastipoct=lastipoct+ord(letter)

                    melastipoct=0
                    for letter in username:
                            melastipoct=melastipoct+ord(letter)
                    combinedoct=(melastipoct+lastipoct)%253
                    melastipoct=melastipoct%253

                    lastipoct=lastipoct%253
                    myip=str(combinedoct)+".0.0."+str(lastipoct)


        friendslocalip=str(combinedoct)+".0.0."+str(melastipoct+1)
```

δημιουργούμε την κοινή βάση δεδομένων

```
                    couch.create(username+"-"+friend)
                    mergeddb=couch[username+"-"+friend]
```

αρχικοποιούμε τη βάση με ένα άδειο κοινωνικό δωμάτιο επικοινωνίας,

```
                    couch.replicate("monocles",username+"-
"+friend)
```

δημιουργούμε λογαριασμό για το νέο φίλο που αποδέχτηκε ο χρήστης,

```
                    users=couch["_users"]
                    users["org.couchdb.user:"+friend]={"_id":
"org.couchdb.user:"+friend,  "name": friend,  "roles": ["admin"],
"type": "user",  "password": friendship_password}
```

δίνουμεδικαιώματαεγγραφής στο νέοφίλο στη κοινήβάσηδεδομένων,

```
                    try :

        mergeddb["_security"]={"admins":{"names":[friend],"roles"
:[]},"members":{"names":[friend],"roles":[]}}
                    except:
                            print "ok"
```

δημιουργούμε μια νέακαταχώρηση στη βάσηδεδομένων των φίλων για να ενημερώνεται η IP εφόσον αλλάξει μέσω του script update_ip.

```
                    friendsdb=couch["friends"]
                    doc_id=uuid4().hex

        friendsdb[doc_id]={"edgepid":"","friend":friend,"ip":"","lo
calip":myip,"network":username+"-
"+friend,"password":friendship_password,"port":"","supernode_por
t":supernode_port,"username":username,"initiated_by":friend}
```

Ξεκινούμε το συγχρονισμό διπλής κατεύθυνσης των κοινών δεδομένων μεταξύ του φίλου και του χρήστη,

```
                    time.sleep(15)
                    replica=couch["_replicator"]
                    doc_id=uuid4().hex

        replica[doc_id]={"source":"http://admin:admin@localhost:59
84/"+username+"-
"+friend,"target":"http://"+username+":"+friendship_password+"@
"+friendslocalip+":5984/"+friend+"-
"+username,"continuous":1,"retries_per_request":"infinity"}
                    doc_id=uuid4().hex

        replica[doc_id]={"source":"http://"+username+":"+friendshi
p_password+"@"+friendslocalip+":5984/"+friend+"-
"+username,"target":"http://admin:admin@localhost:5984/"+userna
me+"-"+friend,"continuous":1,"retries_per_request":"infinity"}
```

ενημερώνουμε το φυλλομετρητή ότι το αίτημα φιλίας έγινε αποδεκτό,

```
                    return friend + " accepted"
```

ξεκινάμετην web εφαρμογή,

```
if __name__ == '__main__':
app.run()
```

6.3.3 Ανανέωση δυναμικών διευθύνσεων

import couchdb #βιβλιοθήκη για την επικοινωνία με τη βάσηδεδομένωνcouchdb

importsubprocess #βιβλιοθήκη για την έναρξηεξωτερικώνδιεργασιών

importtime #βιβλιοθήκηαπαραίτητη για να γίνονταιπαύσειςκάποιωνδευτερόλεπτων στο κώδικα

fromcollectionsimportCounter #βιβλιοθήκη που διαβάζει μια συλλογήνημάτων και τη χρησιμοποιούμε για να βρούμε το νήμα που εμφανίζεταιπερισσότερεςφόρες

fromhashlibimportsha512 #βιβλιοθήκη για δημιουργίαhashνημάτωναπόάλλανήματα

fromhashlibimportmd5 #βιβλιοθήκη για δημιουργίαhashνημάτωναπόάλλανήματα

#αρχικοποιούμε τη σύνδεση με το couchdb και διαβάζουμε τη βάσηδεδομένων των φίλων

```
server=couchdb.Server("http://admin:admin@localhost:5984")
friends=server["friends"]
```

συνάρτηση που δέχεται ως inputένανήμα-πόρο και μετάαπόαριθμόπροσπαθειώνεπιστρέφει τις διευθύνσειςip που βρέθηκε στη κατοχή τους το συγκεκριμένονήμα.

```
def find_friend(domain):
        trynum=0
        while True:
                try:
                        if trynum==40:
                                break

        supernodeips=subprocess.check_output(["kadnode-
ctl","lookup",domain])
                        return supernodeips
                        break
                except subprocess.CalledProcessError, e:
                        time.sleep(5)
                        trynum=trynum+1
                        continue
while True:
        time.sleep(10)
```

Βρίσκουμε την εξωτερικήIP της σύνδεσης μας στο διαδίκτυο

```
        find_ip_cmd="upnpc -s | grep ExternalIPAddress | sed 's/[^0-
9\.]//g'"
        my_external_ip=subprocess.check_output(find_ip_cmd,shell=
True)
        my_external_ip=my_external_ip.split("\n")[0]
```

Συγκρίνουμε αν η διεύθυνσηIP του χρήστηέχειαλλάξει. Αν άλλαξεαποθηκεύουμε την νέαδιεύθυνση στη βάση για να ενημερωθεί και ο φίλος.

```
        for c in friends:
                trynum=0
                supernodeips=""
                friend=friends[c]
                friendshipdb=server[friend["username"]+"-
"+friend["friend"]]
                friendshipdoc=friendshipdb["ips"]
                if                friendshipdoc[friend["username"]]!=
my_external_ip:
                        friendshipdoc[friend["username"]]=
my_external_ip
```

Αναζητάμε αν η διεύθυνση του φίλουάλλαξεμέσω του δικτύουDHT:

```
print "Searching for "+friend["friend"]


                hex_dig = sha512(friend["friend"]).hexdigest()
                invalid=True
                while invalid:
                        b=0
                        al=""
                        ips=[]
                        userhashes=[]
                        supernodeips=""
                        for i in xrange(1,121):
                                b=b+1
                                al=al+hex_dig[i]
                                if b==40:
                                        userhashes.append(al)

        supernodeips=supernodeips+find_friend(al)
                                        al=""
                                        b=0
                        for i in supernodeips.split("\n"):
                                ips.append(i.split(":")[0])

                        ip=Counter(ips).most_common(1)[0][0]
                        for i in supernodeips.split("\n"):
                                if ip == i.split(":")[0]:
                                        port                            =
str(int(i.split(":")[1])+(10-int(i.split(":")[1])%10))
                        if Counter(ips).most_common(1)[0][1]>1:
                                invalid= False

                print ip

                print port
```

Εφόσον η διεύθυνση του φίλουάλλαξε, το αποθηκεύουμε στη βάσηδεδομένων και επανεκκινούμε το edge με την ενημερωμένηδιεύθυνση:

```
                if ip != friend["ip"] or friend["port"]!=port:
                        print friend["friend"] +" changed ip.New ip is
"+ip
                        friend["ip"]=ip
                        friend["port"]=port



                        subprocess.call(["kill",friend["edgepid"]])

                        if
friend["initiated_by"]==friend["username"]:
                                subprocess.call(["edge","-
d",friend["friend"],"-a",friend["localip"],"-c",friend["network"],"-
k",friend["password"],"-l",ip+":"+friend["port"],"-
l",my_external_ip+":"+friend["supernode_port"]])
                        else:
                                subprocess.call(["edge","-
d",friend["friend"],"-a",friend["localip"],"-c",friend["network"],"-
k",friend["password"],"-
l",my_external_ip+":"+friend["supernode_port"],"-
l",ip+":"+friend["port"]])



                        try:

        pid=subprocess.check_output(["pgrep","-f",friend["ip"]])
                        except:
                                print "found pid"
                        friend["edgepid"]=pid.split("\n")[0]
                        friends[c]=friend
                        print "Saved changed ip and restarted edge"
```

7ο Συμπεράσματα

Το παραπάνω σύστημα δέχεται πολλές διορθώσεις και βελτιώσεις. Η υλοποίηση του έγινε για την επίδειξη των δυνατοτήτων που προσφέρει μια πλατφόρμα βάσεων δεδομένων όπως το couchdb σε συνδυασμό με το bittorrent πρωτόκολλο πάνω σε ένα πολύ φθηνό μικροϋπολογιστή προσβάσιμο στο μεγαλύτερο μέρος του πληθυσμού. Ένα πολύ βασικό χαρακτηριστικό του όλου εγχειρήματος είναι ότι βασίζεται σε δημοφιλή και με μεγάλες κοινότητες επιμέρους προγράμματα κάτι που σημαίνει πως το μεγαλύτερο μέρος της δουλειάς που χρειάζεται για την διατήρηση του συνολικού συστήματος ενημερωμένου είναι εξασφαλισμένο. Επίσης, οι υποδομές επικοινωνίας που χρειάζονται για να φέρουν τους μικροϋπολογιστές-κόμβους σε επικοινωνία μεταξύ τους είναι και αυτές εξασφαλισμένες αφού το bittorrent πρωτόκολλο,αποτελεί ιδιαίτερα μεγάλο ποσοστό του συνολικών δεδομένων που διακινούνται στο διαδίκτυο και οι DHT κομβοι αυξανονται συνεχώς.Μια απο τις βασικες επιτυχιες της υπάρχουσας υλοποίησης είναι ότι το σύστημα αποτελεί plug-and-play δηλαδή ο χρήστης να αγοράζει τη συσκευή και το μόνο που να χρειάζεται να κάνει να είναι η τροφοδότηση με ρεύμα και εισαγωγή ενός όνομα χρήστη και ενός κωδικού.

Ένα θέμα το οποίο προκύπτει είναι το πως θα πιστοποιείται η αποκλειστική ταυτότητα του κάθε χρήστη στη πρωτη γνωριμια μεσω του δικτυου DHT. Όταν λέμε ταυτότητα δεν εννοούμε απαραίτητα επώνυμη ταυτότητα άλλα και ψευδώνυμη. Η λύση που προτείνουμε είναι η χρήση PGP κρυπτογράφησης των μηνυμάτων γνωριμιας που θα διακινούνται,όπου η πηγή ενός μηνύματος θα διασφαλίζεται μέσω του δημόσιου κλειδιού που θα κατέχουν οι αναγνώστες και του ιδιωτικού κλειδιού που θα κατέχει ο συγγραφέας.

8ο Βιβλιογραφία

1. *Distributed Social Networks.* **Athanasios, Katsis.** 2013.

2. *Web 2.0 Summit 2011.*

3. **JM Maness - Webology, 2006 - doaj.org.**

4. **http://blog.compete.com/2011/04/26/compete-ranking-of-top-50-websites-for-march-2011-shows-traffic-springing-forward/.**

5. *Information Revelation and Internet Privacy Concerns.* **Alyson L. Young, A Quan-Haase.**

6. *The state of social media.* **Comscore, Mike Shaw.** 2012.

7. **Tavani 2010, 166 and Wacks 2010, 124, Zureik and Harling Stalker 2010, 15.**

8. **Ess 2009, 56, et al.**

9. **Jhally 2006, 60.**

10. **Andrejevic (2002, 2004).**

11. *On Diaspora's Social Network, You Own Your Data.* **Karen Weise, Bloomberg.**

12. **Maxwell Salzberg, Kickstarter.***Decentralize the web with Diaspora.*

13. *http://diasporial.com/tutorials/interface-and-aspects.*

14. **S Buchegger, D SchiÃ¶berg, LH Vu, A Datta.***PeerSoN: P2P social networking: early experiences and insights.*

15. **SEONG, Seok-Won.***PrPl: a decentralized social networking infrastructure. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond. ACM, 2010. p. 8.*

16. **Graffi, Kalman, et al.***"LifeSocial. KOM: A secure and P2P-based solution for online social networks." Consumer Communications and Networking Conference (CCNC), 2011 IEEE. IEEE, 2011.*

17. **Brogle, Marc, Dragan Milic, and Torsten Braun.***"QoS enabled multicast for structured P2P networks." Workshop on Peer-to-Peer Multicasting at the 4th IEEE Consumer Communications and Networking Conference. IEEE. 2007.*

18. **Rowstron, Antony, and Peter Druschel.***"Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems." Middleware 2001. Springer Berlin Heidelberg, 2001.*

19. **Kalofonos, Dimitris N., et al.***"Mynet: A platform for secure p2p personal and social networking services." Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on. IEEE, 2008.*

20. **Ruottu, Toni.***"Developing Peer-To-Peer Web Applications." (2011).*

21. **Strauch, Christof.***Ultra-Large Scale Sites.*

22. **Kriha., Walter.***"NoSQL databases.".*

23. **Brewer, Eric.***Pushing the CAP: strategies for consistency and availability.*

24. **Bartholomew, Daniel.***"SQL vs. NoSQL." Linux Journal 2010.195 (2010).*

25. **Franke, Craig, et al.***"Distributed semantic web data management in HBase and MySQL cluster." Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011.*

26. **Varia, Jinesh.***"Cloud architectures." (2008).*

27. **Chodorow, Kristina.***MongoDB: the definitive guide. O'Reilly, 2013.*

28. **George, Lars.***HBase: the definitive guide. O'Reilly Media, Inc., 2011.*

29. **Erman, Ilie, Popescu, 2010.**

30. **Steiner, Biersack, 2008.**

31. *http://wiki.apache.org/couchdb/Technical%20Overview.*

Παράρτημα

Δημιουργία χρηστη, Εύρεση φιλων, Αποδοχή αιτησεων φιλίας

importweb #βιβλιοθήκη που χρειάζεται για την εξυπηρέτηση των αρχείωνhtml και javascript

importsubprocess #βιβλιοθήκη για την έναρξηεξωτερικώνδιεργασιών

importcouchdb #βιβλιοθήκη για την επικοινωνία με τη βάσηδεδομένωνcouchdb

importtime #βιβλιοθήκηαπαραίτητη για να γίνονταιπαύσειςκάποιωνδευτερόλεπτων στο κώδικα

fromuuidimportuuid4 #βιβλιοθήκη που παράγειτυχαίανήματα 40 χαρακτήρων

fromhashlibimportsha512 #βιβλιοθήκη για δημιουργίαhashνημάτωναπόάλλανήματα

fromcollectionsimportCounter #βιβλιοθήκη που διαβάζει μια συλλογήνημάτων και τη χρησιμοποιούμε για να βρούμε το νήμα που εμφανίζεταιπερισσότερεςφόρες

```
def make_text(string):
return string
```

```
supernode_port="50020" #αρχικοποιούμε τη θύρα στην οποία θα δέχεται συνδέσεις το supernode
subprocess.call(["kadnode","--daemon"]) #ξεκινάμε το kadnode
time.sleep(10)
subprocess.call(["kadnode-ctl","import","bttracker.debian.org"]) #συνδεόμαστε σε ένα δημόσιο peer για να αρχικοποιήσουμε τη σύνδεση στο DHT
time.sleep(20)
```

```
urls = ('/create_user', 'create_user',
        '/find_friend/?','find_friend',
```

```
        '/accept_friend/?','accept_friend',)        #ενημερώνουμε το webserver ποιες διευθύνσειςπρέπει να εξυπηρετεί

render = web.template.render('templates/')        #ενημερώνουμε το webserver σε ποιο φάκελο θα βρει τα αχρείαhtml

app = web.application(urls, globals())

#αρχικοποιούμεκάποιεςμεταβλητές που θα χρησιμοποιηθούν στη συνεχεία

username=""

usernames=[]

#δημιουργούμε τις φόρμεςεισαγωγήςδεδομένων για τα δεδομένα που θα δέχονται

my_form = web.form.Form(

web.form.Textbox('', class_='username', id='username',value='Username '),

web.form.Textbox('', class_='password', id='password',value='password'),

    )

friend_form=web.form.Form(

        web.form.Textbox('', class_='friend', id='friend',value='Friend..'),

        web.form.Dropdown('usernames', usernames)

        )


# συνάρτηση που δέχεται ως inputένανήμα-πόρο και μετά από αριθμόπροσπαθειώνεπιστρέφει τις διευθύνσειςip που βρέθηκε στη κατοχή τους το συγκεκριμένονήμα

def find_friend_ip(domain):

    trynum=0

    supernodes=""

    while True:

        try:

            if trynum==20:

                return supernodes

            supernodes=subprocess.check_output(["kadnode-ctl","lookup",domain])
```

```
                return supernodes
                break
            except subprocess.CalledProcessError, e:
                time.sleep(5)
                trynum=trynum+1
                continue


classcreate_user:          #κλάσηγιατηδημιουργίανέουχρήστη
    DefGET(self): #συνάρτηση που επιστρέφει τη φόρμανέουχρήστη στο φυλλομετρητή
        form = my_form()
        return render.create_user(form, "Here you can create a new user.Please input your username and password.The username is used for your friends to find you.")

    defPOST(self): #συνάρτηση που δέχεται το όνομα και το κώδικααπό το φυλλομετρητή του χρήστη και προσπαθεί να τον ανακοινώσει στο DHT
        form = my_form()
        form.validates()

        username = form.value['username']
        password = form.value['password']


        subprocess.Popen(["supernode","-l",supernode_port])      #ξεκινάμε ένα supernode
        subprocess.call(["upnpc","-r",supernode_port,"UDP"])  #ανοίγουμε τη θυρα επικοινωνιας UDP του supernode στο modem-router
```

```
subprocess.call(["upnpc","-r",supernode_port,"TCP"])  #ανοιγουμε τη θυρα επικοινωνιας TCP του supernode στο modem-router


hex_dig = sha512(username).hexdigest()  #εξαγουμε ενα νημα 120 χαρακτηρων απο το ονομα χρηστη βασει του αλγοριθμου sha512

#αρχικοποιηση μεταβλητων απαραιτητων για μετα

trynum=0

supernodeips=""

b=0

al=""

ips=[]

userhashes=[]

#δημιουργουμε μια λουπα που σπαει το παραγμενο νημα 120 χαρακτηρων σε 3 κομματια των 40 χαρακτηρων

#αυτα αποτελουν και πορους που αναζητουμε στο DHT

#αν βρεθουν 2 απο τους 3 πορους στην ιδια διευθυνση IP τοτε ενημερωνουμε το χρηστη οτι πρεπει να επιλεξει αλλο

#ονομα χρηστη γιατι αυτο που εισηγαγε ηδη χρησιμοποιηται

for i in xrange(1,121):
        b=b+1
        al=al+hex_dig[i]
        if b==40:
                userhashes.append(al)
                supernodeips=supernodeips+find_friend_ip(al)
                al=""
                b=0
for i in supernodeips.split("\n"):
        if i!="":
                ips.append(i.split(":")[0])
valid=Counter(ips).most_common(1)
```

```
if valid==[]:
        print "cool"
elif valid[0][1]>1:
        return "Another user found with the same username.Please use numbers within your username"
#αν το ονομα χρηστη δε χρησιμοποιηται εξαγουμε μια τοπικη IP απο το ονομα χρηστη βασει της οποια θα συνδεθουμε στο τοπικο supernode
lastipoct=0
for letter in username:
        lastipoct=lastipoct+ord(letter)
lastipoct=lastipoct%253
myip=str(lastipoct)+".0.0."+str(lastipoct)

#αρχικοποιουμε τη συνδεση με το couchdb
couch = couchdb.Server("http://admin:admin@localhost:5984/")
#δημιουργουμε ενα νεο χρηστη με το ονομα και κωδικο που επελεξε στη βαση δεδομενων
users=couch["_users"]
users["org.couchdb.user:"+username]={"_id": "org.couchdb.user:"+username, "name": username, "roles": ["admin"], "type": "user", "password": password}
#εισαγουμε το νεο χρηστη σε μια βαση δεδομενων για να μπορουμε να το αναζητησουμε μετα
#αυτο γινεται με τη προοπτικη οτι θα μπορουν πολλοι χρηστες να δημιουργησουν λογαριασμο στον ιδιο κομβο
profiles=couch["profiles"]
profiles[username]={"username":username}
#ανακοινωνουμε τα 3 παραγμενα νηματα απο το ονομα χρηστη στο δικτυο dht
dht_port=0
for userhash in userhashes:
        dht_port=dht_port-1
```

```
                subprocess.call(["kadnode-
ctl","announce",userhash,str(int(supernode_port)+dht_port),"1000000"])

                time.sleep(2)

        #δημιουργουμε με το edge ενα νεο interface οπου μπορουν αλλοι χρηστες να
συνδεθουν γνωριζοντας μονο το username

        subprocess.call(["edge","-d","public","-a",myip,"-c","public-"+username,"-
k","public","-l","localhost:"+supernode_port])

        #δημιουργουμε ενα νεο χρηστη που εχει δικαιωματα εγγραφης σε μια βαση
οπου θα αποθηκευονται τα αιτηματα φιλιας

        users["org.couchdb.user:public-"+username]={"_id":
"org.couchdb.user:public-"+username, "name": "public-"+username, "roles": ["user"],
"type": "user", "password": "public"}

        couch.create("public-"+username)

        public=couch["public-"+username]



        try :

                public["_security"]={"admins":{"names":["public-
"+username],"roles":[]},"members":{"names":["public-"+username],"roles":[]}}

        except:

                print "ok"

        #εφοσον ο λογαριασμος του χρηστη δημιουργηθηκε επιστρεφουμε στο
φυλλομετρητη το παρακατω μηνυμα

        return "User created.Proceed to next step"


classfind_friend: #κλαση για την αναζητηση φιλων

    defGET(self): #οταν ζητηθει απο το χρηστη να αναζητησει ενα νεο φιλο
επιστρεφεται τα προφιλ που εχουν δημιουργηθει σε αυτο το κομβο βασει του οποιου θα γινει
ενα αιτημα φιλιας

        couch = couchdb.Server("http://admin:admin@localhost:5984/")

        profiles=couch["profiles"]

        usernames=[]

        for i in profiles:
```

```
                usernames.append((profiles[i]["username"],profiles[i]["username"]))
        friend_form=web.form.Form(
web.form.Textbox('', class_='friend', id='friend',value='Friend..'),
web.form.Dropdown('usernames', usernames)
    )
        form = friend_form()
        return render.find_friend(form, "Here you can find a new friend.Input the username of your friend and I will search in the cloud.")
```

defPOST(self): # όταν ο χρήστηςαποστείλει το προφίλβάσει του όποιου θα γίνει το αίτημαφιλίαςαυτή η συνάρτησηαναλαμβάνει την αναζήτηση και εφόσονβρεθεί την αποστολήαιτήματοςφιλίας στο χρηστή που αναζητήθηκε

```
        form = friend_form()
        form.validates()

        friend = form.value['friend']
        username=form.value['username']
        invalid=True
        whileinvalid: #εφόσονβρεθεί το όνομα που αναζητήθηκε με πιθανότηταεγκυρότητας 66% από τη λούπααυτήεπιστρέφεται η διεύθυνσηIP
                trynum=0
                hex_dig = sha512(friend).hexdigest()
                b=0
                al=""
                supernodeips=""
                ips=[]
                userhashes=[]
                for i in xrange(1,121):
                        b=b+1
                        al=al+hex_dig[i]
```

```
                if b==40:
                        userhashes.append(al)
                        supernodeips=supernodeips+find_friend_ip(al)
                        al=""
                        b=0
        for i in supernodeips.split("\n"):
                ips.append(i.split(":")[0])


        print ips
        friendip=Counter(ips).most_common(1)[0][0]
        print "friends ip is "+friendip
        if friendip!="":
                for i in supernodeips.split("\n"):
                        if friendip == i.split(":")[0]:
                                friendsport = str(int(i.split(":")[1])+(10-
int(i.split(":")[1])%10))
        if Counter(ips).most_common(1)[0][1]>1:
                invalid=False


        # εδώεξάγουμε τις τοπικέςδιευθύνσειςIP που θα χρησιμοποιηθούν για το
δημόσιο και ιδιωτικόεικονικόδίκτυομεταξύ των χρηστών
        lastipoct=0
        for letter in friend:
                lastipoct=lastipoct+ord(letter)
        melastipoct=0
        for letter in username:
                melastipoct=melastipoct+ord(letter)
        combinedoct=(melastipoct+lastipoct)%253
        melastipoct=melastipoct%253
```

```
lastipoct=lastipoct%253


mixedip=str(combinedoct)+".0.0."+str(lastipoct+1)

myip=str(lastipoct)+".0.0."+str(lastipoct+1)

friendslocalip=str(lastipoct)+".0.0."+str(lastipoct)


friendship_password=uuid4().hex #δημιουργειται ενας τυχαιος κωδικος  για την κρυπτογραφηση μεταξυ των χρηστων

#δημιουργειται ενας νεος λογοριασμος για τον φιλο που στον οποιο θα σταλθει αιτημα φιλιας

couch = couchdb.Server("http://admin:admin@localhost:5984/")

users=couch["_users"]

users["org.couchdb.user:"+friend]={"_id": "org.couchdb.user:"+friend, "name": friend,  "roles": ["admin"],  "type": "user",  "password": friendship_password}

#δημιουργουμε μια βαση δεδομενων για το νεο φιλο για μετεπειτα χρηση οπως επισης  δινουμε δικαιωματα εγγραφης στο νεο φιλο

couch.create(friend)

frienddb=couch[friend]


try :


    frienddb["_security"]={"admins":{"names":[friend],"roles":[]},"members":{"names":[friend],"roles":[]}}

except:

        print "ok"

#συνδεομαστε στο δημοσιο εικονικο δικτυου του  φιλου που βρεθηκε

subprocess.Popen(["edge","-d","edge2","-a",myip,"-c","public-"+friend,"-k","public","-l",friendip+":"+friendsport])

time.sleep(20)

try:

        pid=subprocess.check_output(["pgrep","-f","public-"+friend]) #βρισκουμε τον αριθμο της διεργασιας edge που δημιουργησαμε για μεταπειτα κλεισιμο
```

```
                except:
                        print "ss"
                pid=pid.split("\n")[0]


                # συνδεομαστε στη δημοσια βαση δεδομενων και εισαγουμε ενα νεο αιτημα φιλιας
                friendcouch=couchdb.Server("http://public-"+friend+":public@"+friendslocalip+":5984")
                frienddb=friendcouch["public-"+friend]
                doc=uuid4().hex

        frienddb[doc]={"friend":username,"password":friendship_password,"supernode_port":supernode_port}
                #καταργουμε τη διεργασια edge που ξεκινησαμε προηγουμενως γιατι πλεον δεν χρειαζεται
                subprocess.call(["kill",pid])


                #δημιουργουμε τη κοινη βαση δεδομενων βασει της οποιας θα επικοινωνουμε
                couch.create(username+"-"+friend)
                mergeddb=couch[username+"-"+friend]
                #βρισκουμε την εξωτερικη IP της συνδεσης μας στο διαδικτυο
                find_ip_cmd="upnpc -s | grep ExternalIPAddress | sed 's/[^0-9\.]//g'"
                my_external_ip=subprocess.check_output(find_ip_cmd,shell=True)
                my_external_ip=my_external_ip.split("\n")[0]
                mergeddb["ips"]={username:my_external_ip,friend:friendip}
                #δινουμε δικαιωματα εγγραφης στο φιλο που δημιουργησαμε
                try :

        mergeddb["_security"]={"admins":{"names":[friend],"roles":[]},"members":{"names":[friend],"roles":[]}}
                except:
```

print "ok"

#δημιουργουμε μια νεα καταχωρηση στη βαση δεδομενων των φιλων για να ενημερωνεται η IP εφοσον αλλαξει μεσω του scriptupdate_ip

friendsdb=couch["friends"]

doc=uuid4().hex

friendsdb[doc]={"edgepid":"","friend":friend,"ip":friendip,"localip":mixedip,"network":username+"-
"+friend,"password":friendship_password,"port":friendsport,"supernode_port":supernode_port,"username":username,"initiated_by":username}

#~ friendsdb[doc]={"edgepid":"","friend":friend,"ip":"","localip":mixedip,"network":username+"-
"+friend,"password":friendship_password,"port":"","supernode_port":supernode_port,"username":username}

#~ subprocess.call(["edge","-d","edge2","-a",mixedip,"-c",username+"-"+friend,"-k",friendship_password,"-l",friendip+":"+friendsport])

#~ couch.replicate(username+"-"+friend,"http://"+friendslocalip+":5984/"+friend+"-"+username,continuous=True)

#επιστρεφουμε στο φυλλομετρητη του χρηστη την IP του φιλου που δημιουργησαμε

returnsupernodeips

classaccept_friend: # η κλαση αυτη χρησιμοποιηται για την αποδοχη αιτηματων φιλιας

defGET(self): # οταν ο χρηστης ζητησει να δει τα αιτηματα φιλιας ,επιλεγει το προφιλ που δεχτηκε αιτηματα φιλιας

couch = couchdb.Server("http://admin:admin@localhost:5984/")

profiles=couch["profiles"]

usernames=[]

for i in profiles:

usernames.append((profiles[i]["username"],profiles[i]["username"]))

friend_form=web.form.Form(

web.form.Dropdown('usernames', usernames)

```
        )
            form = friend_form()
            return render.find_friend(form, "choose your user.")

    defPOST(self): #εφοσον ο χρηστης εχει επιλεξει προφιλ του επιστρεφονται τα αιτηματα φιλιας και μπορει να τα αποδεχτει
            form = friend_form()
            form.validates()

            friend = form.value['friend']
            username=form.value['username']
            friends=[]
            if friend=="":
                couch = couchdb.Server("http://admin:admin@localhost:5984/")
                friendrequests=couch["public-"+username]
                for i in friendrequests:

    friends.append((friendrequests[i]["friend"],friendrequests[i]["friend"]))
                form=web.form.Form(
web.form.Dropdown('friend', friends)
        )

                return render.find_friend(form, "accept a friend.")
            else:
                #η περιπτωση που εχει αποδεχτει καποιο αιτημα φιλιας
                couch = couchdb.Server("http://admin:admin@localhost:5984/")
                friendrequests=couch["public-"+username]
                for i in friendrequests:
                    if friend==friendrequests[i]["friend"]:
```

```
                    friendship_password=friendrequests[i]["password"]
                #εξαγουμε τις τοπικες IP που θα χρησιμοποιηθουν στο ιδιωτικο εικονικο δικτυο μεταξυ των χρηστων
                lastipoct=0
                for letter in friend:
                    lastipoct=lastipoct+ord(letter)

                melastipoct=0
                for letter in username:
                    melastipoct=melastipoct+ord(letter)
                combinedoct=(melastipoct+lastipoct)%253
                melastipoct=melastipoct%253
                lastipoct=lastipoct%253
                myip=str(combinedoct)+".0.0."+str(lastipoct)
                friendslocalip=str(combinedoct)+".0.0."+str(melastipoct+1)
                #δημιουργούμε τη κοινήβάσηδεδομένων
                couch.create(username+"-"+friend)
                mergeddb=couch[username+"-"+friend]
                #αρχικοποιούμε τη βάση με έναάδειοκοινωνικόδωμάτιοεπικοινωνίας
                couch.replicate("monocles",username+"-"+friend)
                #δημιουργούμελογαριασμό για το νέοφίλο που αποδέχτηκε ο χρήστης
                users=couch["_users"]
                users["org.couchdb.user:"+friend]={"_id": "org.couchdb.user:"+friend, "name": friend, "roles": ["admin"], "type": "user", "password": friendship_password}
                #δίνουμεδικαιώματαεγγραφής στο νέοφίλο στη κοινήβάσηδεδομένων
                try :
```

mergeddb["_security"]={"admins":{"names":[friend],"roles":[]},"members":{"names":[friend],"roles":[]}}

except:

print "ok"

#δημιουργούμε μια νέα καταχώρηση στη βάση δεδομένων των φιλών για να ενημερώνεται η IP εφόσον αλλάξει μέσω του scriptupdate_ip

friendsdb=couch["friends"]

doc_id=uuid4().hex

friendsdb[doc_id]={"edgepid":"","friend":friend,"ip":"","localip":myip,"network":username+"-"+friend,"password":friendship_password,"port":"","supernode_port":supernode_port,"username":username,"initiated_by":friend}

#~ subprocess.call(["edge","-d","edge3","-a",myip,"-c",friend+"-"+username,"-k",friendship_password,"-l","localhost:"+supernode_port])

#ξεκινούμε το συγχρονισμό διπλής κατεύθυνσης των κοινών δεδομένων μεταξύ του φίλου και του χρηστή

time.sleep(15)

replica=couch["_replicator"]

doc_id=uuid4().hex

replica[doc_id]={"source":"http://admin:admin@localhost:5984/"+username+"-"+friend,"target":"http://"+username+":"+friendship_password+"@"+friendslocalip+":5984/"+friend+"-"+username,"continuous":1,"retries_per_request":"infinity"}

doc_id=uuid4().hex

replica[doc_id]={"source":"http://"+username+":"+friendship_password+"@"+friendslocalip+":5984/"+friend+"-"+username,"target":"http://admin:admin@localhost:5984/"+username+"-"+friend,"continuous":1,"retries_per_request":"infinity"}

#~ couch.replicate(username+"-"+friend,"http://"+username+":"+friendship_password+"@"+friendslocalip+":5984/"+friend+"-"+username,continuous=True)

```
                                #~
couch.replicate("http://"+username+":"+friendship_password+"@"+friendslocalip+":5984/"+
friend+"-"+username,username+"-"+friend,continuous=True)


                                #ενημερώνουμε το φυλλομετράτεότι το αίτημαφιλίαςέγινεαποδεκτό

                                return friend + " accepted"




if __name__ == '__main__': #ξεκινάμετηνwebεφαρμογή

app.run()
```

Ανανέωση δυναμικών διευθυνσέων

import couchdb #βιβλιοθηκη για την επικοινωνια με τη βαση δεδομενων couchdb

import subprocess #βιβλιοθηκη για την εναρξη εξωτερικων διεργασιων

import time #βιβλιοθηκη απαραιτητη για να γινονται παυσεις καποιων δευτερολεπτων στο κωδικα

from collections import Counter #βιβλιοθηκη που διαβαζει μια συλλογη νηματων και τη χρησιμοποιουμε για να βρουμε το νημα που εμφανιζεται περισσοτερες φορες

from hashlib import sha512 #βιβλιοθηκη για δημιουργια hash νηματων απο αλλα νηματα

from hashlib import md5 #βιβλιοθηκη για δημιουργια hash νηματων απο αλλα νηματα

#αρχικοποιουμε τη συνδεση με το couchdb και διαβαζουμε τη βαση δεδομενων των φιλων

server=couchdb.Server("http://admin:admin@localhost:5984")

friends=server["friends"]

συναρτηση που δεχεται ως input ενα νημα-πορο και μετα απο αριθμο προσπαθειων επιστρεφει τις διευθυνσεις ip που βρεθηκε στη κατοχη τους το συγκεκριμενο νημα

```
def find_friend(domain):
        trynum=0
        while True:
                try:
                        if trynum==40:
                                break
                        supernodeips=subprocess.check_output(["kadnode-
ctl","lookup",domain])
                        return supernodeips
                        break
                except subprocess.CalledProcessError, e:
                        time.sleep(5)
                        trynum=trynum+1
```

```
                        continue



while True:
        time.sleep(10)
        #βρισκουμε την εξωτερικη IP της συνδεσης μας στο διαδικτυο

        find_ip_cmd="upnpc -s | grep ExternalIPAddress | sed 's/[^0-9\.]//g'"
        my_external_ip=subprocess.check_output(find_ip_cmd,shell=True)
        my_external_ip=my_external_ip.split("\n")[0]



        for c in friends:
                trynum=0
                supernodeips=""
                friend=friends[c]
                friendshipdb=server[friend["username"]+"-"+friend["friend"]]
                friendshipdoc=friendshipdb["ips"]
                if friendshipdoc[friend["username"]]!= my_external_ip:
                        friendshipdoc[friend["username"]]= my_external_ip
                print "Searching for "+friend["friend"]



                hex_dig = sha512(friend["friend"]).hexdigest()
                invalid=True
                while invalid:
                        b=0
```

```
                al=""
                ips=[]
                userhashes=[]
                supernodeips=""
                for i in xrange(1,121):
                        b=b+1
                        al=al+hex_dig[i]
                        if b==40:
                                userhashes.append(al)
                                supernodeips=supernodeips+find_friend(al)
                                al=""
                                b=0
                for i in supernodeips.split("\n"):
                        ips.append(i.split(":")[0])

                ip=Counter(ips).most_common(1)[0][0]
                for i in supernodeips.split("\n"):
                        if ip == i.split(":")[0]:
                                port = str(int(i.split(":")[1])+(10-
int(i.split(":")[1])%10))
                if Counter(ips).most_common(1)[0][1]>1:
                        invalid= False

        print ip
        #~ ip=ips[0].split(":")[0]
        print port

        if ip != friend["ip"] or friend["port"]!=port:
                print friend["friend"] +" changed ip.New ip is "+ip
```

```
                friend["ip"]=ip

                friend["port"]=port




                subprocess.call(["kill",friend["edgepid"]])

                #~ kill_child_processes(int(friend["edgepid"]))

                if friend["initiated_by"]==friend["username"]:

                        subprocess.call(["edge","-d",friend["friend"],"-
a",friend["localip"],"-c",friend["network"],"-k",friend["password"],"-
l",ip+":"+friend["port"],"-l",my_external_ip+":"+friend["supernode_port"]])

                else:

                        subprocess.call(["edge","-d",friend["friend"],"-
a",friend["localip"],"-c",friend["network"],"-k",friend["password"],"-
l",my_external_ip+":"+friend["supernode_port"],"-l",ip+":"+friend["port"]])



                #~ pro = subprocess.Popen(args=['gnome-terminal','--
command=edge -d edge2 -a '+friend["localip"]+'-c '+friend["network"]+" -k
"+friend["password"]+' -l '+ip+":"+friend["port"]])



                try:

                        pid=subprocess.check_output(["pgrep","-f",friend["ip"]])

                except:

                        print "found pid"

                friend["edgepid"]=pid.split("\n")[0]

                friends[c]=friend

                #~ couch.replicate(username+"-
"+friend,"http://"+username+":"+friendship_password+"@"+friendslocalip+":5984/"+friend+
"-"+username,continuous=True)

                #~
couch.replicate("http://"+username+":"+friendship_password+"@"+friendslocalip+":5984/"+
friend+"-"+username,username+"-"+friend,continuous=True)
```

```
print "Saved changed ip and restarted edge"
```