ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία εφαρμογής για αυτόματη ανταλλαγή πληροφοριών μεταξύ ασύρματων (802.11) συσκευών

dispel

Του φοιτητή

Αντωνιάδη Αντώνιου

Αρ. Μητρώου: 04/2562

Επιβλέπων καθηγητής

Ηλιούδης Χρήστος

Θεσσαλονίκη 2013

ΠΡΟΛΟΓΟΣ

Με την εργασία αυτή κατάφερα να αναπτύξω τις γνώσεις μου σε σημείο που ποτέ δε φανταζόμουν. Είμαι πολύ χαρούμενος που την ανέλαβα, παρά τη δυσκολία της και τη συνεχόμενη αφοσίωση που χρειαζόταν. Είναι με διαφορά το πιο σημαντικό έργο που έχω δημιουργήσει στον τομέα της Πληροφορικής, εντός και εκτός σχολής, καθώς επίσης είναι ένα project που θα συνεχίσω να αναπτύσσω στο μέλλον.

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία έχει στόχο τη μελέτη μίας εφαρμογής που δημιουργήθηκε από την ιδέα ενός συγκεκριμένου σχεδίου (concept), καθώς και την υλοποίησή της στο βαθμό που μπορούν να βγουν συμπεράσματα. Για τη δημιουργία της εφαρμογής έγινε μεγάλη μελέτη πάνω στο λειτουργικό σύστημα GNU/Linux, τη γλώσσα προγραμματισμού Python (έκδοση 2.7) και τις βιβλιοθήκες της, των ασύρματων δικτύων 802.11, ad-hoc, της client/server αρχιτεκτονικής, του δικτυακού προγραμματισμού χαμηλού επιπέδου (sockets), των εργαλείων και των βοηθητικών προγραμμάτων που χρησιμοποιούνται, καθώς και του πρωτοκόλλου TCP/IP σε βάθος.

Για την κατανόηση της λειτουργίας του προγράμματος, βασική προϋπόθεση είναι η κατανόηση των τεχνολογιών που χρησιμοποιούνται και για αυτό κάποια από τα κεφάλαια της εργασίας είναι αφιερωμένα στην ανάλυσή τους, και σε βάθος εισχώρηση σε συγκεκριμένα χαρακτηριστικά που χρειάστηκαν για τη δημιουργία της εφαρμογής.

Η εφαρμογή έχει την ονομασία "dispel" καθώς διασκορπίζει ασύρματα στις συσκευές στις οποίες τρέχει πληροφορίες και δεδομένα.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ πάρα πολύ τον κ. Στέφανο Χαρχαλάκη που μου έδωσε μια σημαντική ευκαιρία να δημιουργήσω μία πτυχιακή εργασία από την οποία κατόρθωσα να αποκομίσω μεγάλη εμπειρία στην έρευνα αλλά και τη δημιουργία λογισμικού.

Ευχαριστώ πολύ τον κ. Χρήστο Ηλιούδη για τη βοήθεια, τη συνεργασία και την υπομονή του.

Η εργασία αυτή είναι αφιερωμένη στους απανταχού υποστηρικτές του Free Software Foundation.

ΚΑΤΑΛΟΓΟΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Βασικός στόχος της εργασίας είναι η μελέτη μίας συγκεκριμένης αρχιτεκτονικής για την αυτόματη ανταλλαγή δεδομένων μέσω υπολογιστών καθώς και η εξόρυξη συμπερασμάτων. Ως proof of concept στο τέλος της εργασίας θα διαπιστωθεί εάν το μοντέλο που αναπτύχθηκε μπορεί να έχει πρακτική εφαρμογή και αν ναι μέχρι ποιο σημείο.

Η βασική ιδέα η οποία είναι στόχος να αποδειχθεί είναι η ασύρματη επικοινωνία μεταξύ συσκευών χωρίς την ύπαρξη κάποιου Access Point και με τη χρήση του SSID ως βασικό στοιχείο ορισμού της μοναδικότητας των συσκευών. Στο πρωτόκολλο 802.11 πάνω στο οποίο βασίζεται η εφαρμογή, γίνεται χρήση της client-server αρχιτεκτονικής.

Η εφαρμογή ονομάστηκε ‘dispel’ καθώς διασκορπά πληροφορίες και δεδομένα ασύρματα ενώ παράλληλα επικαλύπτει (έστω προσωρινά) τις ρυθμίσεις της ασύρματης κάρτας δικτύου για της ανάγκες της εφαρμογής, απαλλάσσοντας τη συσκευή από κάθε είδους εξάρτηση με κάποιο Access Point ή κάποιο ‘στατικό’ Ad-Hoc δίκτυο.

Γίνεται αναλυτική περιγραφή της ιδέας που θα υλοποιηθεί. Στα αρχικά κεφάλαια γίνεται αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν καθώς και το πως αυτές συνεισφέρουν στη λειτουργία της εφαρμογής. Η βασική μελέτη των τεχνολογιών αυτών ήταν το πιο κρίσιμο και χρονοβόρα κομμάτι της, αλλά και το πιο ενδιαφέρον και ουσιαστικό.

Το λειτουργικό σύστημα που χρησιμοποιείται είναι το GNU/Linux. Είναι σημαντικό να διευκρινιστεί αυτό καθώς η εφαρμογή είναι συμβατή μόνο με διανομές Linux. Η αξιοπιστία του λειτουργικού συστήματος καθώς και η φιλοσοφία του ελεύθερου λογισμικού είναι βασικές αρετές που αναγνωρίζονται από τους

απανταχού ειδικούς της πληροφορικής. Οι διανομές στις οποίες η εφαρμογή δοκιμάστηκε είναι οι εξής:

- Fedora 16 και Fedora 17
- Slackware 13.37 και Slackware 14
- Slacko (Puppy) 4 και Slacko (Puppy) 5

Η τελευταία διανομή είναι η πιο αξιόπιστη κατά τη γνώμη μου στην κατηγορία των φορητών διανομών. Η εφαρμογή δοκιμάστηκε εκεί για να περαιτέρω παρατήρηση της λειτουργίας της σε μη σταθερό περιβάλλον. Αξίζει να σημειωθεί πως η διανομές βασισμένες στο Puppy Linux φορτώνονται εξ' ολοκλήρου στη μνήμη κατά τη διάρκεια της εκκίνησης, κάτι που κάνει την ταχύτητα απόκρισης πάρα πολύ γρήγορη. Το αρνητικό είναι η έλλειψη σταθερότητας και η ανάγκη για αποθήκευση των δεδομένων περιοδικά στα διάφορα αποθηκευτικά μέσα.

Αξίζει να σημειωθεί ότι η εφαρμογή δοκιμάστηκε (τροποποιημένη) μη ασύρματα στο Ethernet στα πρώτα στάδια της δημιουργίας της και ανταποκρινόταν ταχύτατα ακόμη και σε υπολογιστή 300 MHZ με 64 MB RAM. Σκοπός βέβαια δεν είναι η ανάπτυξή της σε ενσύρματα δίκτυα αλλά στα πλαίσια της έρευνας πολλές διαφορετικές κατευθύνσεις ανάπτυξης ανακαλύφθηκαν.

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Python. Συγκεκριμένα, η εφαρμογή έχει δοκιμαστεί στις παρακάτω εκδόσεις:

- Python 2.6.5
- Python 2.6.6
- Python 2.7.3
- Python 2.7.4
- και πειραματικά στην Python 3.3.0 (δεν είναι 100% συμβατή)

Μπορεί να θεωρείται σχετικά αργή ως γλώσσα αλλά η δυνατότητά της να επικοινωνεί σε πολύ καλό βαθμό με το λειτουργικό Linux, η ευελιξία της στον κώδικα, καθώς και το ότι είναι ήδη εγκατεστημένη στις περισσότερες διανομές Linux είναι κάποια από τα κριτήρια που την κάνουν την καταλληλότερη. Αποτελεί

λογισμικό ανοιχτού κώδικα και η ίδια η γλώσσα με πολύ μεγάλο αριθμό προγραμματιστών που τη χρησιμοποιούν.

Τα πειράματα για την εφαρμογή εφαρμόστηκαν με τη χρήση διαφόρων ασύρματων καρτών, εκ των οποίων οι περισσότερες δεν είχαν υποστήριξη για το πρωτόκολλο 802.11n αλλά φυσικά υποστήριζαν τα 802.11b και 802.11g. Η συντριπτική πλειοψηφία των δοκιμών έγινε σε υπολογιστές τελικού χρήστη.

Πέρα από αυτά τα βασικά στοιχεία, η εφαρμογή χρησιμοποιεί πακέτα όπως το wireless-tools για την ανίχνευση δικτύων κλπ καθώς και επικοινωνεί άμεσα με το κέλυφος του λειτουργικού συστήματος αντλώντας πληροφορίες ή εκτελώντας εντολές. Ένα μεγάλο μέρος του προγράμματος χρησιμοποιεί φυσικά τις υπάρχουσες βιβλιοθήκες της Python 2 καθώς και εντολές Linux που μπορούν να βρεθούν σε κάθε διανομή.

Η σπουδαιότητα της εργασίας μπορεί κυρίως να συνοψιστεί στο ότι η έρευνα (θεωρητική και τεχνολογική) καθώς και πρακτική υλοποίηση του έργου είναι κάτι καινούριο που, τουλάχιστον από αναζητήσεις στο διαδίκτυο, κάποιος δε μπορεί να βρει κάτι απολύτως ίδιο ή τολμώντας να πω ακόμη και παρόμοιο.

Τις εφαρμογές που μπορεί να έχει ένα τέτοιο πρόγραμμα αρχικά εύκολα μπορεί να τις φανταστεί κανείς σε επίπεδο τελικού χρήστη, για παράδειγμα τη δημιουργία μίας εφαρμογής γραπτής/άμεσης επικοινωνίας (chat) μεταξύ υπολογιστών που υποστηρίζουν ασύρματη επικοινωνία σε ένα εύρος περιοχής. Από εκεί και πέρα άλλες πιθανές εφαρμογές είναι η χρήση του προγράμματος ως βοηθητικό προγράμματα ασύγχρονης επικοινωνίας για την ανταλλαγή στατιστικών δεδομένων, ρυθμίσεων κλπ μεταξύ υπολογιστών που είναι υπεύθυνοι για υπηρεσίες (servers).

Τα συμπεράσματα που βγαίνουν στο τέλος θα είναι η απόρροια όλης της μελέτης και έρευνας, της υλοποίησης, καθώς και της αξιολόγησης του concept. Η δυνατότητα εξέλιξης είναι πάντα παρούσα καθώς η εργασία αυτή δοκιμάζει κάτι καινούριο, αυθεντικό και μη βιβλιογραφημένο, με όλες τις δυσκολίες αλλά και την ικανοποίηση που την αποτελούν.

Γίνεται εκτενής ανάλυση του κώδικα, με λεπτομέρειες για τη χρήση κάθε συνάρτησης και παραμέτρου. Η εφαρμογή χωρίζεται στο μενού, μια ομάδα εντολών, τον daemon και μια σειρά από άλλα μέρη, που συνθέτουν το πρόγραμμα.

Η σειρά των κεφαλαίων που παρουσιάζονται είναι η σειρά με την οποία οι τεχνολογίες μελετήθηκαν και εφαρμόστηκαν. Τα βιβλία του τμήματος ήταν σε πολλές περιπτώσεις πολύ χρήσιμα για μελέτη (όπως στην περίπτωση των sockets) και αυτό με βοήθησε αρκετά. Βέβαια, η βιβλιογραφία που χρησιμοποιήθηκε είναι μεγάλη και ποικίλλει.

Η εργασία αυτή μου έδωσε τη δυνατότητα να αναπτύξω τις γνώσεις και τη δημιουργικότητά μου στο πεδίο της Πληροφορικής καθώς και να δω στην πράξη το πως οι ακαδημαϊκές γνώσεις ως υπόβαθρο μπορούν να εφαρμοστούν με τη δημιουργία μιας ερευνητικής εφαρμογής όπως η συγκεκριμένη.

ΚΕΦΑΛΑΙΟ 1

PROOF OF CONCEPT

ΕΙΣΑΓΩΓΗ

Σε αυτό κεφάλαιο αναλύεται τι είναι το Proof of Concept, ποιοι είναι οι στόχοι του, πώς γίνεται η αξιολόγηση των αποτελεσμάτων και αν και πότε θεωρείται ολοκληρωμένο ή επιτυχημένο.

Η εργασία αυτή στηρίζεται σε αυτό, καθώς το πλάνο και ο σχεδιασμός της είναι ένα έργο που θα υλοποιηθεί χωρίς κάποια θεωρητική βάση παρόμοιας εφαρμογής και επειδή το αποτέλεσμα της εργασίας και της εφαρμογής είναι η αξιολόγηση της αρχιτεκτονικής στην πράξη μέσω της εφαρμογής και όχι η λειτουργικότητα της εφαρμογής.

ΥΠΟΚΕΦΑΛΑΙΟ 1.1

ΤΙ ΕΙΝΑΙ PROOF OF CONCEPT

Proof of Concept (απόδειξη έννοιας ή POC) είναι μια δοκιμή μιας ιδέας που με την κατασκευή ενός πρωτοτύπου της εφαρμογής. Πρόκειται για μία καινοτόμα, μειωμένη έκδοση του συστήματος που προτίθενται να αναπτυχθεί. Για να δημιουργηθεί ένα πρωτότυπο, χρειάζεται εργαλεία, τις δεξιότητες, τις γνώσεις και τις προδιαγραφές του σχεδιασμού.

Είναι η επίτευξη μιας συγκεκριμένης μεθόδου ή ιδέας για να αποδείξει τη δυνατότητα πραγματοποίησής της, σκοπός της οποίας είναι η βεβαίωση ότι κάποια ιδέα ή θεωρία έχει τη δυνατότητα να χρησιμοποιηθεί. Μία απόδειξη της έννοιας είναι συνήθως μικρή και μπορεί να είναι ή να μην είναι πλήρης.

Στην ανάπτυξη λογισμικού, χρησιμοποιείται συχνά για να περιγράψει πολλές διαφορετικές διαδικασίες με διαφορετικούς στόχους και ρόλους συμμετέχοντα: Αυτό γίνεται από τους σχεδιαστές ή τους προγραμματιστές.

Μια απόδειξη της έννοιας μπορεί να αναφέρεται σε μια μερική λύση που περιλαμβάνει ένα σχετικά μικρό αριθμό των χρηστών που ενεργούν σε ρόλους ώστε να διαπιστωθεί αν το σύστημα ικανοποιεί κάποια πτυχή του σκοπού που έχει σχεδιαστεί για. Μόλις ένας ενδιαφερόμενος είναι ικανοποιημένος, ένα πρωτότυπο αναπτύσσεται το οποίο στη συνέχεια χρησιμοποιείται για να αναζητήσει χρηματοδότηση ή να αποδειχθεί.

Αντιθέτως, ο στόχος μιας απόδειξης της τεχνολογίας είναι να καθοριστεί η λύση σε κάποιο τεχνικό πρόβλημα, όπως το πώς δύο συστήματα μπορεί να ενσωματωθούν ή ότι η απόδοση μπορεί να επιτευχθεί με μια δεδομένη διαμόρφωση.

Ένα πιλοτικό έργο αναφέρεται σε μια αρχική εξάπλωση του συστήματος στην παραγωγή, με στόχο την περιορισμένη έκταση της προβλεπόμενης τελικής λύσης. Το πεδίο μπορεί να περιορίζεται από τον αριθμό των χρηστών που μπορούν να έχουν πρόσβαση στο σύστημα, οι διαδικασίες που επηρεάζονται, οι εταίροι των επιχειρήσεων που συμμετέχουν, ή άλλους περιορισμούς που προσιδιάζουν στον τομέα.

Σκοπός του πιλοτικού προγράμματος είναι να δοκιμάσει, συχνά σε ένα περιβάλλον παραγωγής, αν το σύστημα λειτουργεί όπως έχει σχεδιαστεί. Το POC μπορεί επίσης να αναφέρεται σε επιμέρους λύσεις που αφορούν ένα μικρό αριθμό χρηστών σε ρόλους που ενεργούν ώστε να διαπιστωθεί κατά πόσον ένα σύστημα ανταποκρίνεται σε ορισμένες απαιτήσεις. Ο γενικός στόχος του POC είναι να βρεθούν λύσεις σε τεχνικά προβλήματα όπως το πώς τα συστήματα μπορούν να ενσωματωθούν ή πως απόδοση μπορεί να επιτευχθεί μέσα από μια συγκεκριμένη διαμόρφωση.

Το POC περιλαμβάνει εκτενή έρευνα και αξιολόγηση και κατατίθεται ως ένα ενιαίο πακέτο προς τα ενδιαφερόμενα μέρη. Απαιτεί κάποιο προσυμφωνημένο σύνολο αποδείξεων ή δοκιμές που καθορίζουν τα κριτήρια επιτυχίας του. Οι απαιτήσεις της απόδειξης γενικά τεκμηριώνονται μετά από όταν το έργο σχεδιασμού έχει ολοκληρωθεί και πριν βρεθεί μία συγκεκριμένη λύση.

ΥΠΟΚΕΦΑΛΑΙΟ 1.2

ΤΟ DISPEL ΩΣ P.O.C

Σύμφωνα με τα παραπάνω μπορεί να διαπιστωθεί ότι το πρόγραμμα dispel (η εφαρμογή) που δημιουργήθηκε είναι ο τρόπος απόδειξης και η πρακτική εφαρμογή της αρχικής ιδέας, το πλάνου και της προτεινόμενης αρχιτεκτονικής. Εξ' αρχής η εργασία δεν είχε συγκεκριμένο τρόπο προσδιορισμού της ολοκλήρωσή της. Η υλοποίηση του concept θα μπορούσε να αποδείξει από τις πρώτες κιόλας μέρες της ανάθεσης της εργασίας μία λανθασμένη οπτική γωνία ή και ένα μη πραγματοποιήσιμο έργο καθώς η ιδέα θα μπορούσε να είναι εξαρχής λανθασμένη.

Είναι σημαντικό να σημειωθεί ότι οι έννοιες αυτές ήταν απολύτως απαραίτητες να εξηγηθούν καθώς μια λάθος εντύπωση για το στόχο της εργασίας θα σήμαινε λάθος προσέγγιση στην αξιολόγηση της εφαρμογής, των πειραμάτων και της προγραμματιστικής υλοποίησης.

Για αυτό λοιπόν το λόγο έγινε μεγάλη προσπάθεια ώστε να αναλυθεί κάθε λεπτομέρεια και κάθε κομμάτι ή μέρος του θεωρητικού σχεδιασμού να μπορεί να είναι ξεκάθαρο για ποιο λόγο ήταν σημαντικό να χρησιμοποιηθεί και ποια είναι η λειτουργία του.

Μία πιο ερευνητική προσέγγιση της εργασίας προσφέρει την ικανότητα καλύτερης κατανόησής της. Η εφαρμογή, και είναι σημαντικό να σημειωθεί αυτό, δεν είναι ένα πρόγραμμα αναγκαστικά ολοκληρωμένου που απευθύνεται σε τελικούς χρήστες και σκοπό έχει κάποια συγκεκριμένη λειτουργία για την ικανοποίηση των αυτών. Η εφαρμογή είναι ένα framework ανάπτυξης της ίδιας ή και άλλων εφαρμογών βασισμένες σε αυτήν με σκοπό την κατανόηση όλων των τεχνολογιών που χρησιμοποιούνται καθώς και τη μελλοντική εξέλιξή της σε μετατροπή του κώδικα ώστε να επιτευχθούν συγκεκριμένοι τρόποι δημιουργίας εφαρμογών για τελικούς χρήστες.

ΕΠΙΛΟΓΟΣ

Έχοντας ξεκαθαρίσει τον ορισμό του όρου Proof of Concept (POC) καθώς και το πως θα εφαρμοστεί στη συγκεκριμένη περίπτωση της εργασίας αυτής, το πρώτο μέρος για την κατανόηση του θεωρητικού σχεδιασμού έχει ολοκληρωθεί. Έχοντας υπ' όψιν όλα τα δεδομένα της σχεδίασης, η βάση για την προσέγγιση της έρευνας και της μελέτης έχει οριστεί.

Στα επόμενα κεφάλαια θα γίνεται αναφορά στις απαραίτητες τεχνολογίες που καταστούν τη βάση της εφαρμογής και που πρέπει να είναι πλήρως κατανοητές για την ερμηνεία της εφαρμογής. Γίνεται ανάλυση σε αυτές καθώς είναι το κλειδί στη σωστή λειτουργία της εφαρμογής και καθώς τα διάφορα εργαλεία τους και ευελιξία αυτών είναι ένα μεγάλο κομμάτι και προνόμιο αυτής.

Στο επόμενο κεφαλαίο αναλύεται το λειτουργικό σύστημα GNU/Linux.

ΚΕΦΑΛΑΙΟ 2

ΤΟ ΛΕΙΤΟΥΡΓΙΚΟ GNU/LINUX

ΕΙΣΑΓΩΓΗ

Ξεκινάμε με μία περίληψη του τρόπου με τον οποίο το Linux έγινε το λειτουργικό σύστημα που είναι σήμερα. Γίνεται αναφορά στην παλαιότερη και μελλοντική του ανάπτυξη καθώς και στα πλεονεκτήματα και μειονεκτήματα αυτού του συστήματος. Επίσης γίνεται λόγος το Λογισμικό Ανοιχτού Κώδικα (Open Source) και γίνεται προσπάθεια εξήγησης μερικών αντικειμένων για το GNU.

ΥΠΟΚΕΦΑΛΑΙΟ 2.1

ΙΣΤΟΡΙΑ

Για να κατανοήσουμε τη δημοτικότητα του Linux, χρειάζεται να ταξιδέψουμε πίσω στο χρόνο, περίπου 30 χρόνια...

Φανταστείτε τους υπολογιστές σαν μεγάλα σπίτια, ακόμα και στάδια. Ενώ τα μεγέθη των υπολογιστών αυτών αποτελούν ουσιαστικό πρόβλημα, υπήρχε κάτι που έκανε την κατάσταση ακόμα χειρότερη: κάθε υπολογιστής είχε διαφορετικό λειτουργικό σύστημα. Το λογισμικό έπρεπε πάντα να προσαρμόζεται για να εξυπηρετήσει ένα συγκεκριμένο σκοπό, και το λογισμικό ενός δοσμένου συστήματος δεν έτρεχε σε άλλο σύστημα. Η ικανότητα εργασίας με ένα σύστημα δεν σήμαινε αυτόματα την ικανότητα εργασίας και με κάποιο άλλο . Ήταν δύσκολο, τόσο για τους χρήστες όσο και για τους διαχειριστές συστήματος.

Οι υπολογιστές τότε ήταν υπερβολικά ακριβοί, και έπρεπε να γίνουν θυσίες ακόμα και μετά την αρχική αγορά μόνο και μόνο για να καταλάβουν οι χρήστες πως λειτουργούν. Το συνολικό κόστος ανά μονάδα υπολογιστικής ισχύος ήταν τεράστιο.

Τεχνολογικά ο κόσμος δεν ήταν τόσο ανεπτυγμένος, οπότε θα έπρεπε να ανεχθούν τους υπερμεγέθεις υπολογιστές για άλλη μια δεκαετία. Το 1969, μια ομάδα ερευνητών στα εργαστήρια Bell Labs ξεκίνησαν να εργάζονται για την ανεύρεση λύσης στο πρόβλημα του λογισμικού, έτσι ώστε να διευθετηθούν τα θέματα συμβατότητας. Αυτοί ανάπτυξαν ένα νέο λειτουργικό σύστημα που ήταν

1. Απλό και κομψό.
2. Γραμμένο στην γλώσσα προγραμματισμού C αντί για κώδικα assembly.
3. Ικανό να ανακυκλώνει κώδικα.

Οι ερευνητές των Bell Labs ονόμασαν το πρόγραμμα τους "UNIX."

Τα χαρακτηριστικά ανακυκλωμένου κώδικα ήταν πολύ σημαντικά. Μέχρι τότε, όλα τα εμπορικώς διαθέσιμα υπολογιστικά συστήματα ήταν γραμμένα σε κώδικα ειδικά ανεπτυγμένο για ένα σύστημα. Το UNIX από την άλλη χρειαζόταν μόνο ένα μικρό τμήμα από αυτόν τον εξειδικευμένο κώδικα, το οποίο είναι τώρα ο κοινώς επονομαζόμενος πυρήνας. Ο πυρήνας αυτός είναι το μοναδικό τμήμα κώδικα που χρειάζεται να προσαρμοστεί για κάθε συγκεκριμένο σύστημα και αποτελεί τη βάση του συστήματος UNIX.

Το λειτουργικό σύστημα και όλες οι άλλες λειτουργίες χτίστηκαν γύρω από τον πυρήνα και γράφηκαν σε γλώσσα ανώτερου επιπέδου, τη C. Η γλώσσα αυτή είχε αναπτυχθεί ειδικά για τη δημιουργία του συστήματος UNIX. Με τη χρήση αυτής της νέας τεχνικής, ήταν ευκολότερο να αναπτυχθεί ένα λειτουργικό σύστημα που μπορούσε να λειτουργήσει σε διάφορους τύπους υλικού.

Οι πωλητές λογισμικού ανταποκρίθηκαν γρήγορα, εφόσον μπορούσαν να πουλήσουν 10 φορές περισσότερο λογισμικό σχεδόν χωρίς προσπάθεια. Νέες περίεργες συνθήκες εμφανίστηκαν: φανταστείτε για παράδειγμα υπολογιστές από διαφορετικούς πωλητές να επικοινωνούν στο ίδιο δίκτυο, ή χρήστες που δουλεύουν σε διαφορετικά συστήματα, χωρίς την ανάγκη για επιπλέον εκπαίδευση, να χρησιμοποιούν έναν άλλο υπολογιστή. Το UNIX βοήθησε σε πολύ μεγάλο βαθμό τους χρήστες να συμβαδίσουν με διαφορετικά συστήματα.

Κατά τη διάρκεια των επόμενων δύο δεκαετιών η ανάπτυξη του UNIX συνεχίστηκε. Περισσότερα πράγματα ήταν δυνατό να γίνουν και περισσότεροι πωλητές υλικού και λογισμικού προσθέσανε υποστήριξη για το UNIX στα προϊόντα τους.

Το UNIX θεμελιώθηκε αρχικά μόνο σε πολύ μεγάλα περιβάλλοντα με υπολογιστές μεγάλης ισχύος (mainframes) και μίνι-υπολογιστές (σημειώστε ότι το PC είναι ένας “μικρό”-υπολογιστής). Έπρεπε να εργάζεσαι σε πανεπιστήμιο, για την κυβέρνηση ή για μεγάλους οικονομικούς οργανισμούς για να έρθεις σε επαφή με ένα σύστημα UNIX.

Όμως μικρότεροι υπολογιστές ανακαλύπτονταν, και μέχρι τα τέλη του '80, πολλοί άνθρωποι είχαν οικιακούς υπολογιστές. Μέχρι τότε, υπήρχαν διαθέσιμες αρκετές εκδόσεις του UNIX για αρχιτεκτονική PC, αλλά καμία από αυτές δεν ήταν πραγματικά δωρεάν και το πιο σημαντικό: ήταν όλες τρομερά αργές, οπότε οι περισσότεροι χρησιμοποιούσαν MS DOS ή Windows 3.1 στα οικιακά τους PC.

ΥΠΟΚΕΦΑΛΑΙΟ 2.2

Ο LINUS ΚΑΙ ΤΟ LINUX

Στο ξεκίνημα του '90 τα οικιακά PC ήταν επιτέλους αρκετά ικανά να εκτελέσουν ένα πλήρες UNIX. Ο Linus Torvalds, ένας νεαρός που σπούδαζε επιστήμη υπολογιστών στο πανεπιστήμιο του Ελσίνκι, σκέφτηκε ότι θα ήταν καλή ιδέα να υπάρχει μία σχεδόν δωρεάν διαθέσιμη ακαδημαϊκή έκδοση του UNIX, και αμέσως ξεκίνησε να γράφει κώδικα.

Ξεκίνησε να κάνει ερωτήσεις, ψάχνοντας απαντήσεις και λύσεις που θα τον βοηθούσαν να αποκτήσει UNIX στο PC του. Παρακάτω βρίσκεται μία από τις πρώτες του ταχυδρομικές επιστολές στο comp.os.minix, που χρονολογείται από το 1991:

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)

Newsgroups: comp.os.minix

Subject: Gcc-1.40 and a posix-question

Message-ID: <1991Jul3.100050.9886@klaava.Helsinki.FI>

Date: 3 Jul 91 10:00:50 GMT

Hello netlanders,

Due to a project I'm working on (in minix), I'm interested in the posix standard definition. Could somebody please point me to a (preferably) machine-readable format of the latest posix rules? Ftp-sites would be nice.

Από το ξεκίνημα, ο στόχος του Linus ήταν να υπάρχει ένα ελεύθερο σύστημα απολύτως σύμφωνο με το πρωτότυπο UNIX. Αυτός ήταν ο λόγος που ρώτησε για τις POSIX προδιαγραφές, με το POSIX να είναι ακόμα το πρότυπο του UNIX.

Εκείνες τις μέρες το plug-and-play δεν είχε εφευρεθεί ακόμη, αλλά ο κόσμος που ενδιαφερόταν να αποκτήσει ένα δικό του σύστημα UNIX ήταν τόσος πολύς, που αυτό ήταν μόνο ένα μικρό εμπόδιο. Νέοι οδηγοί δίσκων έγιναν διαθέσιμοι για κάθε είδους υλικό, σε μια διαρκώς αυξανόμενη ταχύτητα. Σχεδόν μόλις ένα νέο τμήμα υλικού γινόταν διαθέσιμο, κάποιος το αγόραζε και το υπέβαλλε σε δοκιμή για το Linux, όπως το σύστημα ονομάστηκε σταδιακά, διαθέτοντας περισσότερο ελεύθερο κώδικα για ένα ακόμα μεγαλύτερο εύρος υλικού. Αυτοί οι προγραμματιστές δεν σταμάτησαν στο δικό τους PC, κάθε τμήμα υλικού που μπορούσαν να βρουν ήταν χρήσιμο για το Linux.

Εκείνο τον καιρό, οι άνθρωποι αυτοί αποκαλούνταν "σπασίκλες" ή "φρικιά", αλλά δεν τους ένοιαζε, όσο η λίστα του υποστηριζόμενου υλικού μεγάλωνε όλο και περισσότερο. Χάρη σε αυτούς τους ανθρώπους, το Linux δεν είναι τώρα μόνο ιδανικό να τρέξει σε ένα νέο PC, αλλά είναι επίσης το επιλεγμένο σύστημα για το παλιό υλικό το οποίο θα ήταν άχρηστο αν το Linux δεν υπήρχε.

Δύο χρόνια μετά την επιστολή του Linus, υπήρχαν 12000 χρήστες Linux. Το σχέδιο, που έγινε δημοφιλές με όσους ασχολούνταν με αυτό από χόμπι, διευρυνόταν σταθερά, ενώ παρέμενε όλο το διάστημα στα όρια του προτύπου POSIX. Όλα τα χαρακτηριστικά του UNIX προστέθηκαν στα επόμενα δύο χρόνια, καταλήγοντας στο ώριμο λειτουργικό σύστημα που έχει γίνει σήμερα το Linux. Το Linux είναι ένας πλήρης κλώνος του UNIX, κατάλληλος για χρήση τόσο σε σταθμούς εργασίας όσο και σε μεσαίους και μεγάλους διακομιστές. Σήμερα, πολλοί από τους πιο σημαντικούς παίκτες στην αγορά υλικού και λογισμικού ο καθένας έχει την ομάδα του από προγραμματιστές Linux, στον τοπικό σας πωλητή μπορείτε ακόμα να αγοράσετε προ-εγκατεστημένα συστήματα Linux με επίσημη υποστήριξη – επίσης όμως υπάρχει ακόμα υλικό και λογισμικό που δεν υποστηρίζεται πλήρως.

ΥΠΟΚΕΦΑΛΑΙΟ 2.3

ΤΡΕΧΟΥΣΕΣ ΕΦΑΡΜΟΓΕΣ ΤΩΝ ΣΥΣΤΗΜΑΤΩΝ LINUX

Σήμερα το Linux έχει μπει στην αγορά υπολογιστών γραφείου. Οι προγραμματιστές Linux αρχικά επικεντρώθηκαν στη δικτύωση και τις υπηρεσίες, ενώ οι εφαρμογές γραφείου ήταν ο τελευταίος φραγμός που γκρεμίστηκε. Πολλές από τις εναλλακτικές λύσεις ξεκίνησαν τα τελευταία χρόνια για να κάνουν το Linux μία αποδεκτή επιλογή σε ένα σταθμό εργασίας, παρέχοντας μία απλή διεπαφή χρήστη και εφαρμογές γραφείου συμβατές με αυτές της MS όπως επεξεργαστές κειμένου, λογιστικά φύλλα, λογισμικό παρουσιάσεων κ.α..

Από την πλευρά των διακομιστών, το Linux είναι γνωστό σαν μία σταθερή και αξιόπιστη πλατφόρμα, παρέχοντας βάσεις δεδομένων και εμπορικές υπηρεσίες για εταιρίες όπως το αμερικανικό ταχυδρομείο. Ειδικά οι πάροχοι Διαδικτύου και οι διαδικτυακών υπηρεσιών εκδηλώνουν προτίμηση στο Linux ως τοίχο προστασίας (firewall), διακομιστή διαμεσολάβησης ή διακομιστή ιστού (proxy και web server), και θα βρείτε ένα κουτί Linux κοντά σε κάθε διαχειριστή συστήματος UNIX που εκτιμά έναν άνετο σταθμό διαχείρισης. Συστοιχίες (clusters) από μηχανές Linux χρησιμοποιούνται στη δημιουργία ταινιών όπως οι "Titanic", "Shrek" κ.α, σε ταχυδρομεία, αποτελούν τα νευραλγικά κέντρα που δρομολογούν την αλληλογραφία και σε μεγάλες μηχανές αναζήτησης. Αυτές είναι μόνο λίγες από τις χιλιάδες 'βαριές' εργασίες που το Linux πραγματοποιεί κάθε μέρα στον κόσμο.

Αξίζει να σημειωθεί ότι το σύγχρονο Linux τρέχει όχι μόνο σε σταθμούς εργασίας, μεσαίους και μεγάλους διακομιστές, αλλά ακόμη και σε συσκευές όπως PDA, κινητά τηλέφωνα, σε πληθώρα από ενσωματωμένες εφαρμογές ακόμα επίσης και σε πειραματικά ωρολόγια χειρός. Αυτό κάνει το Linux το μοναδικό λειτουργικό σύστημα στον κόσμο που καλύπτει τέτοια ευρεία γκάμα υλικού.

ΥΠΟΚΕΦΑΛΑΙΟ 2.4

ΑΝΟΙΚΤΟΣ ΚΩΔΙΚΑΣ

Η ιδέα πίσω από το Λογισμικό Ανοικτού Κώδικα είναι μάλλον απλή: όταν οι προγραμματιστές μπορούν να διαβάσουν, να διανείμουν και να αλλάξουν κώδικα, ο κώδικας θα ωριμάσει. Οι άνθρωποι μπορούν να τον υιοθετήσουν, να τον διαμορφώσουν, να τον αποσφαλματώσουν, και μπορούν αυτό να το κάνουν σε μία ταχύτητα που εκμηδενίζει την απόδοση αυτών που αναπτύσσουν λογισμικό στις συμβατικές εταιρίες. Το λογισμικό αυτό θα είναι πιο ευέλικτο και καλύτερης ποιότητας από το λογισμικό που αναπτύχθηκε χρησιμοποιώντας τα συμβατικά κανάλια, επειδή πολλοί άνθρωποι το έχουν δοκιμάσει σε πολλές διαφορετικές συνθήκες απ' ότι ο κλειστός δημιουργός λογισμικού θα μπορούσε ποτέ.

Η πρωτοβουλία για το Λογισμικό Ανοικτού Κώδικα ξεκίνησε για να ξεκαθαρίσει αυτό στον εμπορικό κόσμο, και πολύ αργά, οι εταιρίες άρχισαν να παίρνουν το μήνυμα. Ενώ πολλοί ακαδημαϊκοί και τεχνικοί είχαν ήδη πειστεί εδώ και 20 χρόνια ότι αυτός είναι ο τρόπος για να προχωρήσουμε, οι εταιρίες χρειάστηκαν εφαρμογές όπως το Διαδίκτυο για να αντιληφθούν ότι μπορούν να έχουν κέρδος από το Λογισμικό Ανοιχτού Κώδικα. Σήμερα το Linux έχει ξεπεράσει το στάδιο που ήταν αποκλειστικά ένα ακαδημαϊκό σύστημα, χρήσιμο μόνο σε μια χούφτα ανθρώπων με τεχνικό υπόβαθρο. Πλέον το Linux παρέχει περισσότερα από το λειτουργικό σύστημα: υπάρχει μια ολόκληρη υποδομή που υποστηρίζει την αλυσίδα της προσπάθειας δημιουργίας ενός λειτουργικού συστήματος, της κατασκευής και δοκιμής προγραμμάτων γι' αυτό, της προσφοράς όλων αυτών στους χρήστες, της παροχής συντήρησης, ενημερώσεων και υποστήριξης και προσαρμογών, κ.α.. Σήμερα, το Linux είναι έτοιμο να δεχθεί την πρόκληση ενός γρήγορα μεταβαλλόμενου κόσμου.

ΥΠΟΚΕΦΑΛΑΙΟ 2.5

LINUX ΚΑΙ GNU

Παρόλο που υπάρχει μεγάλος αριθμός υλοποιήσεων Linux, θα βρείτε πολλές ομοιότητες σε διαφορετικές διανομές, επειδή κάθε μηχάνημα Linux είναι ένα κτίσμα το οποίο μπορείτε να χρησιμοποιήσετε ακολουθώντας τις ανάγκες και την κρίση σας. Η εγκατάσταση του συστήματος είναι μόνο η αρχή μιας μακρόχρονης σχέσης. Ακριβώς όταν νομίσετε ότι έχετε ένα ωραίο λειτουργικό σύστημα, το Linux θα διεγείρει τη φαντασία και τη δημιουργικότητα σας, και όσο περισσότερο αντιλαμβάνεστε τη δύναμη που μπορεί να σας δώσει το σύστημα, τόσο περισσότερο θα προσπαθήσετε να επαναπροσδιορίσετε τα όρια του.

Το Linux μπορεί να φαίνεται διαφορετικό ανάλογα με τη διανομή, το υλικό και το προσωπικό σας γούστο, αλλά οι βασικές αρχές στις οποίες χτίζονται όλες οι γραφικές και οι άλλες διασυνδέσεις, παραμένουν οι ίδιες. Το σύστημα Linux βασίζεται στα GNU εργαλεία (το GNU δεν είναι UNIX), που παρέχουν ένα σύνολο από καθορισμένους τρόπους χειρισμού και χρήσης του συστήματος. Όλα τα εργαλεία GNU είναι Ανοιχτού Κώδικα, οπότε μπορούν να εγκατασταθούν σε κάθε σύστημα. Οι περισσότερες διανομές προσφέρουν προ-μεταγλωττισμένα πακέτα από τα πιο συχνά εργαλεία, όπως πακέτα RPM σε RedHat και πακέτα Debian (αποκαλούμενα επίσης deb ή dpkg) σε Debian, οπότε δεν χρειάζεται να είστε προγραμματιστές για να εγκαταστήσετε ένα πακέτο στο σύστημα σας. Παρόλα αυτά, αν κάνετε και σας αρέσει να κάνετε πράγματα μόνοι σας, θα απολαύσετε το Linux ακόμα περισσότερο, εφόσον οι περισσότερες διανομές περιέχουν ένα πλήρες σετ από εργαλεία ανάπτυξης, επιτρέποντας την εγκατάσταση νέου λογισμικού καθαρά από τον πηγαίο κώδικα. Το στήσιμο αυτό σας επιτρέπει επίσης να εγκαταστήσετε λογισμικό ακόμα και αν δεν υπάρχει σε προ-πακεταρισμένη μορφή κατάλληλη για το σύστημα σας.

Μία λίστα συνηθισμένου GNU λογισμικού:

- Bash: Ο φλοιός GNU
- GCC: Ο GNU Μεταγλωττιστής C
- GDB: Ο Αποσφαλματωτής GNU

- Coreutils: ένα σετ βασικών βοηθημάτων του UNIX στυλ, όπως ls, cat και chmod
- Findutils: για αναζήτηση και εύρεση αρχείων
- Fontutils: για μετατροπή γραμματοσειρών από μια μορφοποίηση σε άλλη ή για δημιουργία νέων γραμματοσειρών
- Το Gimp: GNU Image Manipulation Program (Πρόγραμμα Χειρισμού Εικόνων GNU)
- Gnome: Το περιβάλλον επιφάνειας εργασίας GNU
- Emacs: ένας πολύ δυνατός επεξεργαστής κειμένου
- Ghostscript και Ghostview: διερμηνευτής και γραφικό frontend για αρχεία PostScript.
- GNU SQL: συσχετιστικό σύστημα βάσεων δεδομένων
- Radius: ένας απομακρυσμένος server πιστοποίησης και υπολογισμών
- ...

Πολλές εμπορικές εφαρμογές είναι διαθέσιμες για Linux, και για περισσότερες πληροφορίες σχετικά με αυτά τα πακέτα αναφερόμαστε στην ειδική τους τεκμηρίωση. Θα συζητήσουμε μόνο για δωρεάν διαθέσιμο λογισμικό, το οποίο υπάρχει (στις περισσότερες περιπτώσεις) με αδειοδότηση GNU.

Για να εγκαταστήσετε παραλειπόμενα ή νέα πακέτα, θα χρειαστείτε κάποιο τύπο διαχείρισης λογισμικού. Οι πιο κοινές υλοποιήσεις περιλαμβάνουν τα RPM και dpkg. Το RPM είναι ο διαχειριστής πακέτων του RedHat (RedHat Package Manager), που χρησιμοποιείται σε πληθώρα συστημάτων Linux, ακόμα και αν το όνομα του δεν το υποδηλώνει αυτό. Το Dpkg είναι το σύστημα διαχείρισης πακέτων του Debian, το οποίο χρησιμοποιεί μια διασύνδεση που ονομάζεται apt-get, η οποία μπορεί επίσης να διαχειριστεί και τα πακέτα RPM.

Άλλοι πωλητές σχετικού λογισμικού μπορεί να έχουν τις δικές τους διαδικασίες εγκατάστασης, που μερικές φορές παρομοιάζονται με το InstallShield και άλλα, όπως είναι γνωστό για το MS Windows και άλλες πλατφόρμες (πχ ο Synaptic Package Manager του Ubuntu). Όσο προχωράτε με το Linux, πιθανόν θα έρθετε σε επαφή με ένα ή περισσότερα από αυτά τα προγράμματα.

Ο πυρήνας Linux δεν είναι τμήμα του σχεδίου GNU αλλά χρησιμοποιεί την ίδια αδειοδότηση με το GNU λογισμικό. Μία πληθώρα βοηθημάτων και εργαλείων ανάπτυξης (η σάρκα του συστήματος σας), τα οποία δεν είναι ειδικευμένα για το Linux, έχουν ληφθεί από το σχέδιο GNU. Επειδή κάθε χρησιμοποιήσιμο σύστημα πρέπει να περιέχει τόσο τον πυρήνα όσο και ένα ελάχιστο σετ από βοηθήματα, μερικοί υποστηρίζουν ότι ένα τέτοιο σύστημα πρέπει να αποκαλείται *GNU/Linux* σύστημα.

ΥΠΟΚΕΦΑΛΑΙΟ 2.6

ΔΙΚΤΥΩΣΗ

Ακολουθεί η ανάλυση μερικών δικτυακών εντολών Linux που χρησιμοποιήθηκαν στην εφαρμογή.

Η εντολή ip

Τα γραφικά εργαλεία και σενάρια των διαφόρων διανομών είναι front-ends της εντολής ip (ή των ifconfig και route σε παλιότερα συστήματα) για την εμφάνιση και διαμόρφωση της δικτυακής διαμόρφωσης του πυρήνα.

Η εντολή ip χρησιμοποιείται για να αποδίδει διευθύνσεις IP στις διεπαφές, να ρυθμίζει τις διαδρομές των υπό-δικτύων, να εμφανίζει πληροφορίες σχετικές με το TCP/IP κλπ..

Οι ακόλουθες εντολές εμφανίζουν διευθύνσεις IP και πληροφορίες δρομολόγησης:

```
benny@home benny> ip addr show
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    inet6 ::1/128 scope host
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:50:bf:7e:54:9a brd ff:ff:ff:ff:ff:ff
    inet 192.168.42.15/24 brd 192.168.42.255 scope global eth0
    inet6 fe80::250:bfff:fe7e:549a/10 scope link

benny@home benny> ip route show
192.168.42.0/24 dev eth0  scope link
```

127.0.0.0/8 dev lo scope link

default via 192.168.42.1 dev eth0

Στην εντολή ip addr show:

- Εμφανίζονται δύο διεπαφές δικτύου, ακόμη και σε συστήματα με μόνο μια κάρτα δικτύου: Το "lo" είναι ο τοπικός βρόχος, που χρησιμοποιείται για εσωτερική επικοινωνία. Το "eth0" είναι το συνηθισμένο όνομα της πραγματικής διεπαφής. Μην αλλάξετε ποτέ τη διαμόρφωση του τοπικού βρόχου, αλλιώς ο υπολογιστής σας θα έχει προβλήματα. Οι ασύρματες διεπαφές συνήθως ορίζονται ως "wlan0"; οι διεπαφές modem ως "ppp0", αλλά μπορεί να υπάρχουν και άλλα ονόματα.
- Οι διευθύνσεις IP σημειώνονται με "inet": ο τοπικός βρόχος έχει πάντα διεύθυνση 127.0.0.1, η πραγματική διεπαφή μπορεί να έχει οποιοδήποτε συνδυασμό.
- Η διεύθυνση υλικού (MAC address) της διεπαφής δικτύου, που μπορεί να απαιτείται ως τμήμα της διαδικασίας ταυτοποίησης για τη σύνδεση στο δίκτυο, σημειώνεται με "ether". Ο τοπικός βρόχος έχει 6 ζεύγη μηδενικά, ενώ η κάρτα δικτύου έχει 6 ζεύγη δεκαεξαδικών αριθμών, όπου τα 3 πρώτα ζεύγη καθορίζουν τον κατασκευαστή της κάρτας.

(Η εντολή ip route show δείχνει πληροφορίες δρομολόγησης, δηλαδή την προεπιλεγμένη διεύθυνση IP της πύλης με την οποία επικοινωνεί η διεπαφή δικτύου για την αποστολή/παραλαβή των πακέτων).

Η εντολή ifconfig

Αν και η εντολή ip είναι ο πιο σύγχρονος τρόπος διαμόρφωσης ενός συστήματος Linux, η εντολή ifconfig είναι ακόμη πολύ δημοφιλής. Χωρίς κάποια επιλογή εμφανίζει τις πληροφορίες της διεπαφής δικτύου:

```
els@asus:~$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:70:31:2C:1
          inet addr:60.138.67.31  Bcast:66.255.255.255  Mask:255.255.255.192
          inet6 addr: fe80::250:70ff:fe31:2c14/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31977764 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51896866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:802207 txqueuelen:1000
          RX bytes:2806974916 (2.6 GiB)  TX bytes:2874632613 (2.6 GiB)
          Interrupt:11 Base address:0xec00
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:765762 errors:0 dropped:0 overruns:0 frame:0
          TX packets:765762 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:624214573 (595.2 MiB)  TX bytes:624214573 (595.2 MiB)
```

Εδώ, επίσης, σημειώνουμε τις σημαντικότερες πλευρές διαμόρφωσης της διεπαφής:

- Η διεύθυνση IP σημειώνεται με "inet addr".
- Η διεύθυνση υλικού σημειώνεται με "HWaddr".

Τόσο η ifconfig όσο και η ip εμφανίζουν λεπτομερέστερες πληροφορίες διαμόρφωσης και αρκετά στατιστικά για κάθε διεπαφή, καθώς επίσης αν είναι σε λειτουργία ("UP" and "RUNNING").

(Σημειώνουμε ότι οι εντολές δείχνουν περισσότερες πληροφορίες για τη διεπαφή που είναι ενεργή).

ΕΠΙΛΟΓΟΣ

Μετά από την ανάλυση σε αυτό το κεφάλαιο είναι εμφανείς οι λόγοι για τους οποίους το λειτουργικό σύστημα που επιλέχθηκε για την εργασία είναι το Linux. Περισσότερο από το 95% των υπερυπολιστών στον κόσμο χρησιμοποιούν συστήματα UNIX-like. Οι ισχυρές και εντολές, η εύκολη επικοινωνία αυτών με τον πυρήνα και η άριστη συνεργασία του φλοιού με τη γλώσσα προγραμματισμού Python ενισχύουν την αποδοτικότητα στη συγγραφή κώδικα καθώς και την ευελιξία που έχει ο προγραμματιστής στην υλοποίηση των σχεδιασμών.

Η εφαρμογή dispel λειτουργεί μόνο σε συστήματα Linux. Έχει γίνει πειραματικά η τροποποίηση του κώδικα για την επικοινωνία της εφαρμογής με τον προεπιλεγμένο φλοιό του OpenBSD για δοκιμαστικούς μόνο σκοπούς. Η εφαρμογή λειτουργούσε κανονικά, κάτι που υπογραμμίζει την επεκτασιμότητα καθώς και τη σταθερότητα και μεταφερσιμότητα των συστημάτων που είναι βασισμένα στο UNIX.

Στο επόμενο κεφαλαίο αναλύεται η γλώσσα προγραμματισμού Python.

ΚΕΦΑΛΑΙΟ 3

Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

ΕΙΣΑΓΩΓΗ

Η εφαρμογή της πτυχιακής εργασίας αναπτύσσεται στη γλώσσα Python όπως προαναφέρθηκε στην εισαγωγή αυτού του βιβλίου. Η Python είναι μία από εκείνες τις σπάνιες γλώσσες που είναι ταυτόχρονα απλές και ισχυρές. Η Python είναι μια εύκολη στην εκμάθηση, ισχυρή γλώσσα προγραμματισμού. Έχει αποδοτικές υψηλού επιπέδου δομές δεδομένων και μια απλή αλλά παραγωγική προσέγγιση στον αντικειμενοστραφή προγραμματισμό.

Χαρακτηριστικά της Python:

- Απλή
- Εύκολη στην εκμάθηση
- Ελεύθερη και Ανοικτού κώδικα
- Γλώσσα υψηλού επιπέδου
- Φορητή
- Ερμηνευόμενη
- Επεκτάσιμη
- Εκτενείς βιβλιοθήκες

ΥΠΟΚΕΦΑΛΑΙΟ 3.1

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΛΩΣΣΑΣ Python

Εσοχή κώδικα

Τα κενά διαστήματα στον κώδικα, είναι σημαντικά για την Python, και ειδικά τα κενά στην αρχή της κάθε γραμμής. Αυτά ονομάζονται εσοχές κώδικα. Αυτά τα κενά που προηγούνται του κώδικα (είτε διαστήματα είτε στηλοθέτες), στην αρχή κάθε λογικής γραμμής, χρησιμοποιούνται για να δείξουν το επίπεδο εσοχής της λογικής γραμμής, το οποίο με τη σειρά του υποδεικνύει μια ομαδοποίηση δηλώσεων κώδικα. Αυτό σημαίνει ότι οι δηλώσεις στον κώδικα οι οποίες δουλεύουν μαζί, θα πρέπει να έχουν την ίδια εσοχή.

Η εντολή if

Η εντολή if χρησιμοποιείται για να ελέγξει μια συνθήκη, και αν η συνθήκη ικανοποιείται -ή σε όρους προγραμματισμού, αν η συνθήκη είναι αληθής- τότε εκτελείται ένα μπλοκ από εντολές, το οποίο ονομάζεται μπλοκ if, αλλιώς εκτελείται ένα άλλο μπλοκ εντολών, το οποίο ονομάζεται μπλοκ else. Ολόκληρο το τμήμα else είναι προαιρετικό.

Παράδειγμα χρήσης της εντολής if:

```
number = 23
guess = int(raw_input('Εισάγετε έναν ακέραιο: '))

if guess == number:
  print 'Συγχαρητήρια, το πετύχατε!'
  print "(αλλά δεν υπάρχει βραβείο)"
elif guess < number:
  print 'Χμ, δοκιμάστε έναν μεγαλύτερο ακέραιο.'
else:
```

```
print 'Χμ, δοκιμάστε έναν μικρότερο ακέραιο.'
```

Χρήση των συναρτήσεων

Οι συναρτήσεις ορίζονται με την λέξη-κλειδί def, ακολουθούμενη από ένα αναγνωριστικό όνομα για την συνάρτηση, και μετά ένα ζεύγος παρενθέσεων οι οποίες μπορεί να περιέχουν μερικά ονόματα μεταβλητών. Η γραμμή τελειώνει με μια άνω-κάτω τελεία. Στη συνέχεια ακολουθεί η πλειάδα εντολών που αποτελούν αυτή τη συνάρτηση. Το παράδειγμα που ακολουθεί δείχνει πόσο απλό είναι όλο αυτό στην πραγματικότητα:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Filename: function1.py

def sayHello():
  print 'Γεια σου, κόσμε!'

sayHello() # Κλήση
```

Η έξοδος του πιο πάνω:

```
$ python function1.py
Γεια σου, κόσμε!
```

Αρθρώματα

Ένα άρθρωμα είναι ουσιαστικά ένα αρχείο το οποίο περιλαμβάνει ότι συναρτήσεις και μεταβλητές θα οριστούν. Μετά μπορεί να χρησιμοποιηθεί το άρθρωμα αυτό σε άλλα προγράμματα, αρκεί το αρχείο αυτό να έχει την κατάληξη .py.

Ένα άρθρωμα μπορεί να εισαχθεί από άλλα προγράμματα για να χρησιμοποιηθεί και σ' αυτά η λειτουργικότητά του. Με τον ίδιο τρόπο χρησιμοποιείται και η πρότυπη βιβλιοθήκη της Python.

Χρήση του αρθρώματος sys:

Πρώτα εισάγεται το άρθρωμα sys με την εντολή import. Αυτό λέει στην Python ότι θέλετε να χρησιμοποιηθεί αυτό το άρθρωμα μέσα σε αυτό το πρόγραμμα. Το άρθρωμα sys περιέχει λειτουργικότητα η οποία έχει να κάνει με τον ερμηνευτή της Python και το περιβάλλον του.

Όταν η Python εκτελέσει την εντολή import sys, ψάχνει για το αρχείο με το όνομα sys.py μέσα στους φακέλους που υπάρχουν μέσα στην μεταβλητή sys.path. Αν το αρχείο βρεθεί, τότε όλες οι εντολές μέσα στο κύριο μπλοκ εκείνου του αρθρώματος εκτελούνται και το άρθρωμα είναι πλέον διαθέσιμο για χρήση. Αυτή η αρχικοποίηση γίνεται μόνο την πρώτη φορά την οποία εισάγετε ένα άρθρωμα. Επίσης το «sys» είναι συντομογραφία για το «system». Μέσα στο άρθρωμα sys υπάρχει η μεταβλητή argv. Η αναφορά σε αυτήν γίνεται βάζοντας μια τελεία ανάμεσα στο όνομα του αρθρώματος και το όνομα της μεταβλητής, δηλαδή ως sys.argv. Αυτός ο τρόπος αναφοράς έχει δύο κύρια πλεονεκτήματα:

1. επιβεβαιώνει ότι δεν θα συμπέσει το όνομα της μεταβλητής από το sys με το όνομα κάποιας μεταβλητής argv που θα ορισθεί στο πρόγραμμά, και
2. δείχνει ξεκάθαρα ότι η argv ανήκει στο άρθρωμα sys.

Η μεταβλητή sys.argv είναι μια λίστα από συμβολοσειρές. Συγκεκριμένα, η sys.argv περιέχει τη λίστα με τα ορίσματα από τη γραμμή εντολών, δηλαδή τα ορίσματα τα οποία περνάνε στο πρόγραμμά όταν κληθεί από τη γραμμή εντολών.

Λίστες

Μια λίστα είναι μια δομή δεδομένων, στην οποία τηρούνται μερικά αντικείμενα σε μια σειρά. Δηλαδή, μπορεί να αποθηκευτεί μια ακολουθία από αντικείμενα σε μια λίστα. Οι λίστες αντικειμένων θα πρέπει να περιέχονται μέσα σε αγκύλες (δηλαδή [και]) ώστε η Python να καταλαβαίνει σε τι αναφέρεστε. Αφού θα έχει δημιουργηθεί μια λίστα, μπορεί να προστεθούν ή να αφαιρεθούν αντικείμενα, ή να αναζητηθούν αντικείμενα μέσα στη λίστα. Αφού είναι δυνατή η πρόσθεση και η αφαίρεση αντικειμένων, οι λίστες είναι μεταβλητός τύπος δεδομένων, δηλαδή ένα τύπος ο οποίος μπορεί να αλλαχθεί.

Αρχεία

Μπορούν να ανοιχθούν και να τροποποιηθούν αρχεία για ανάγνωση και εγγραφή, δημιουργώντας ένα αντικείμενο της κλάσης file, και χρησιμοποιώντας τις μεθόδους: read, readline ή write, για ανάγνωση ή εγγραφή στο αρχείο. Η δυνατότητα ανάγνωσης και εγγραφής εξαρτάται από τη λειτουργία στην οποία θα ανοιχθεί το αρχείο. Στο τέλος της εργασίας με το αρχείο, μπορεί να κληθεί η μέθοδος close.

Χρήση του file

Στο παράδειγμα που ακολουθεί επιδεικνύεται η λειτουργία της κλάσης file:

```
#!/usr/bin/python
# Filename: using_file.py

poem = '''\
Programming is fun
When the work is done
if you wanna make your work also fun:
        use Python!
'''

f = file('poem.txt', 'w') # open for 'w'riting
f.write(poem) # write text to file
f.close() # close the file

f = file('poem.txt') # if no mode is specified, 'r'ead mode is assumed by default
while True:
        line = f.readline()
        if len(line) == 0: # Zero length indicates EOF
                break
        print line, # Notice comma to avoid automatic newline added by Python
f.close() # close the file
```

Έξοδος:

```
$ python using_file.py
Programming is fun
When the work is done
if you wanna make your work also fun:
        use Python!
```

Εξαιρέσεις

Οι εξαιρέσεις εμφανίζονται όταν συμβαίνει κάτι στο πρόγραμμα κατ' εξαίρεση, ή με άλλα λόγια, κάτι μη αναμενόμενο. Για παράδειγμα, αν γίνει προσπάθεια να διαβαστεί ένα αρχείο που δεν υπάρχει, ή αν σβηστεί κατά λάθος το αρχείο ενώ τρέχει το πρόγραμμα, τότε προκύπτει μια εξαίρεση. Η Python χειρίζεται αυτά τα γεγονότα εγείροντας ένα σφάλμα για κάθε περίπτωση.

Η σύνταξη Try...Except

Η σύνταξη try...except δοκιμάζει μια ενέργεια και προσπαθεί να χειριστεί την πιθανή αποτυχία της. Στο παράδειγμα που ακολουθεί, την ώρα που το πρόγραμμα προσπαθεί να διαβάσει είσοδο από τον χρήστη, αυτός πληκτρολογεί Ctrl-d.

```
>> s = raw_input('Enter something --> ')
Enter something --> Traceback (most recent call last):
  File "<stdin>", line 1, in ?
EOFError
```

Όλες οι εντολές που είναι πιθανό να εγείρουν κάποιο σφάλμα, μπαίνουν μέσα στο μπλοκ try, και όλα τα πιθανά σφάλματα και εξαιρέσεις στην except, η οποία μπορεί να χειριστεί είτε ένα συγκεκριμένο σφάλμα, είτε μια λίστα από σφάλματα/εξαιρέσεις. Αν δε δοθούν συγκεκριμένα ονόματα σφαλμάτων ή εξαιρέσεων, τότε χειρίζεται οτιδήποτε παρουσιαστεί. Για κάθε μπλοκ try, πρέπει να υπάρχει τουλάχιστον ένα μπλοκ except.

Αν εμφανιστεί κάποιο σφάλμα για το οποίο δεν υπάρχει πρόβλεψη και ειδική διαχείριση, τότε η Python το χειρίζεται ως οποιοδήποτε άλλο σφάλμα, και σταματά την εκτέλεση του προγράμματος, παρέχοντας και ένα μήνυμα λάθους.

ΕΠΙΛΟΓΟΣ

Από αυτό το κεφάλαιο συμπεραίνουμε ότι η Python είναι μία γλώσσα προγραμματισμού εύκολη στην κατανόηση και τη συγγραφή κώδικα. Η εξέλιξη της εφαρμογής πολλές φορές επιταχύνθηκε λόγω των χαρακτηριστικών που αναφέρθηκαν συνοπτικά στην εισαγωγή του κεφαλαίου αλλά και λόγω της εύκολης σύνταξης του κώδικα όπως φάνηκε στο υποκεφάλαιο 3.1.

Στο επόμενο κεφάλαιο γίνεται ανάλυση του δικτυακού μέρους της εργασίας.

ΚΕΦΑΛΑΙΟ 4

ΠΡΩΤΟΚΟΛΛΑ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΔΙΚΤΥΟΥ

ΕΙΣΑΓΩΓΗ

Απαραίτητη προϋπόθεση για τη δημιουργία της εφαρμογής είναι η πλήρης κατανόηση της σουίτας πρωτοκόλλων TCP/IP, του πρωτοκόλλου 802.11, της ασύρματης Ad-Hoc δικτύωσης καθώς και πολλών άλλων τεχνολογιών που αποτελούν τη βάση της δικτυακής αρχιτεκτονικής της εφαρμογής.

Σε αυτό το κεφάλαιο γίνεται επεξήγηση των προαναφερθέντων πρωτοκόλλων για να γίνει κατανοητό στα επόμενα κεφάλαια πως αυτό το σύμπλεγμα δικτύωσης λειτουργεί ώστε η εφαρμογή να μπορέσει να επιβεβαιώσει ή όχι αν το concept μπορεί να επιτευχθεί, και αν όχι, μέχρι ποιο βαθμό μπορεί να λειτουργήσει.

Επίσης γίνεται αναφορά στο πιθανώς πιο κρίσιμο κομμάτι της εργασίας, τον δικτυακό προγραμματισμό χαμηλού επιπέδου (low-level socket programming). Επεξηγούνται επίσης οι συναρτήσεις/μέθοδοι που χρησιμοποιούνται για τη δικτυακή επικοινωνία μεταξύ μηχανημάτων.

Το τελευταίο κομμάτι του κεφαλαίου αναλύει το εργαλεία που συνεισφέρουν στην αυτοματοποιημένη ασύρματη λειτουργία της εφαρμογής (wireless-tools).

ΥΠΟΚΕΦΑΛΑΙΟ 4.1

ΤΟ ΠΡΩΤΟΚΟΛΛΟ 802.11

Η βιομηχανία αποφάσισε ότι ένα πρότυπο για τα ασύρματα LAN μπορεί να είναι καλή ιδέα, έτσι η επιτροπή IEEE που τυποποίησε τα ενσύρματα LAN ανέλαβε να σχεδιάσει ένα πρότυπο για τα ασύρματα δίκτυα. Το πρότυπο στο οποίο κατέληξε ονομάστηκε 802.11 ή όπως είναι γνωστό με το εμπορικό όνομα: WiFi.

Το προτεινόμενο πρότυπο έπρεπε να λειτουργεί με δύο τρόπους:

Με παρουσία ενός σταθμού βάσης.

Με απουσία ενός σταθμού βάσης.

Στην πρώτη περίπτωση όλες οι επικοινωνίες πρέπει να περνούν από το σταθμό βάσης, ο οποίος ονομάζεται σημείο πρόσβασης στην ορολογία του 802.11 (Access Point). Στη δεύτερη περίπτωση οι υπολογιστές απλώς μεταδίδουν ο ένας στον άλλο - αυτός ο τρόπος λειτουργίας ονομάζεται δικτύωση ad hoc.

Η πρώτη απόφαση ήταν η ευκολότερη: πως θα ονομαζόταν το πρότυπο. Όλα τα άλλα πρότυπα για LAN είχαν αριθμούς όπως 802.1, 802.2, 802.3, μέχρι 802.10, έτσι το πρότυπο για τα ασύρματα LAN βαφτίστηκε 802.11. Οι υπόλοιπες αποφάσεις ήταν δυσκολότερες.

Πιο συγκεκριμένα, μερικές από τις πολλές προκλήσεις που έπρεπε να αντιμετωπιστούν ήταν: η ανεύρεση μιας κατάλληλης ζώνης συχνοτήτων, η οποία να είναι κατά προτίμηση διαθέσιμη σε όλο τον κόσμο, η αντιμετώπιση της πεπερασμένης εμβέλειας των ραδιοκυμάτων, η εξασφάλιση των ιδιωτικών δεδομένων των χρηστών, η λήψη πρόνοιας για την περιορισμένη ζωή της μπαταρίας, η ανησυχία για την ανθρώπινη ασφάλεια (προκαλούν καρκίνο τα ραδιοκύματα;), η κατανόηση του αντίκτυπου που θα είχε η μεταφερσιμότητα των υπολογιστών και τέλος η υλοποίηση ενός συστήματος με επαρκές εύρος ζώνης ώστε να είναι οικονομικά βιώσιμο.

Μετά από αρκετή εργασία η επιτροπή κατέληξε το 1997 σε ένα πρότυπο που αντιμετώπιζε αυτά και άλλα ζητήματα. Το ασύρματο LAN που περιγραφόταν λειτουργούσε είτε στο 1 Mbps είτε στα 2 Mbps. Σχεδόν αμέσως κάποιοι διαμαρτυρήθηκαν ότι ήταν πολύ αργό, έτσι ξεκίνησε η δουλειά για ταχύτερα πρότυπα. Αποτέλεσμα αυτής της δουλείας ήταν τα πρότυπα 802.11a, και το 802.11b.

Το πρότυπο 802.11 του 1999 καθορίζει πέντε επιτρεπόμενες τεχνικές μετάδοσης για το φυσικό επίπεδο. Η κάθε μία από τις πέντε κάνει δυνατή τη μετάδοση ενός πλαισίου MAC από τον ένα σταθμό στον άλλο. Οι τεχνικές διαφέρουν, όμως, στη χρησιμοποιούμενη τεχνολογία και στις ταχύτητες που επιτυγχάνουν. Αυτές οι τεχνικές επιγραμματικά είναι οι ακόλουθες:

Με υπέρυθρες

Με Εξάπλωση Φάσματος με Συνεχή Αλλαγή Συχνότητας (FHSS)

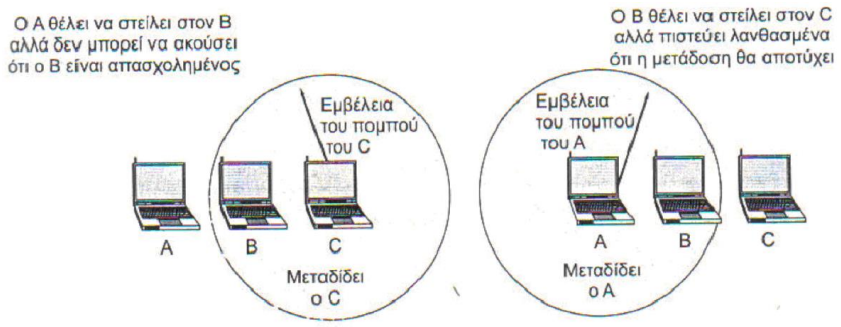
Με Εξάπλωση Φάσματος Άμεσης Ακολουθίας (DSSS)

Με Ορθογώνια Πολύπλεξη με Διαίρεση Συχνότητας (OFDM)

Με Εξάπλωση Φάσματος Άμεσης Ακολουθίας Υψηλού Ρυθμού Μετάδοσης (HR-DSSS)

Το πρωτόκολλο υποεπιπέδου MAC του 802.11 είναι αρκετά διαφορετικό από αυτό του Ethernet, λόγω της εγγενούς πολυπλοκότητας του ασύρματου περιβάλλοντος εν συγκρίσει με ένα ενσύρματο. Υπάρχουν κάποια προβλήματα τα οποία πρέπει να αντιμετωπιστούν.

Αρχικά, υπάρχει το πρόβλημα του κρυφού σταθμού. Αφού όλοι οι σταθμοί δεν είναι εντός της εμβέλειας όλων των άλλων, οι μεταδόσεις που πραγματοποιούνται σε ένα τμήμα μιας κυψέλης μπορεί να μη λαμβάνονται σε άλλα σημεία στην ίδια κυψέλη. Στο παράδειγμα-εικόνα που φαίνεται παρακάτω όταν ένας σταθμός C μεταδίδει στο B, αν ο A ανιχνεύσει ότι το κανάλι είναι διαθέσιμο, δεν θα ακούσει τίποτα και θα συμπεράνει εσφαλμένα ότι μπορεί να αρχίσει να μεταδίδει στο B.



(α) (β)

Σχήμα "Πρόβλημα κρυφού και εκτεθειμένου σταθμού"

(α) Το πρόβλημα του κρυφού σταθμού.

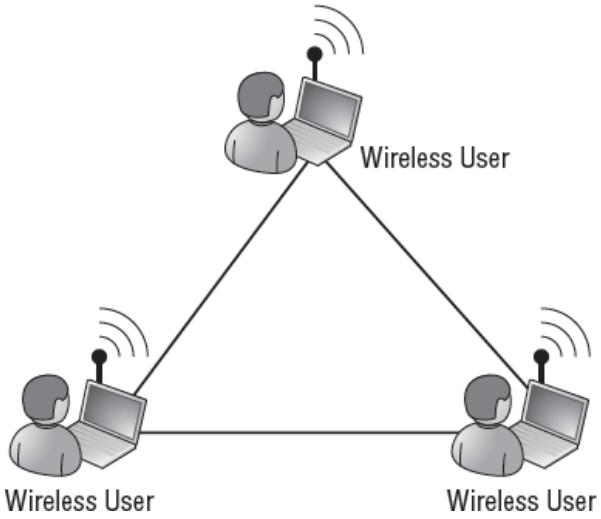
(β) Το πρόβλημα του εκτεθειμένου σταθμού.

ΥΠΟΚΕΦΑΛΑΙΟ 4.2

Ad-Hoc ΔΙΚΤΥΑ

Μία ακόμη χρήση των ασύρματων δικτύων είναι εκείνη της δικτύωσης ειδικού σκοπού ή αλλιώς της Ad-Hoc Networking. Στην περίπτωση αυτή δεν υπάρχει κάποιος κεντρικός υπολογιστής που να διαχειρίζεται το δίκτυο, αλλά απλά οι υπολογιστές είναι συνδεδεμένοι ο ένας με τον άλλο. Δίκτυα τέτοιου είδους συνήθως εγκαθίστανται προσωρινά και έχουν ως σκοπό την εξυπηρέτηση μιας άμεσης ανάγκης.

Για παράδειγμα, μπορούμε να σκεφτούμε την περίπτωση όπου μια ομάδα υπαλλήλων συνδέουν τους φορητούς υπολογιστές τους, ώστε να εξυπηρετηθούν οι ανάγκες μιας on-line σύσκεψης ή παρουσίασης, όπου σε έναν υπολογιστή θα γίνεται η παρουσίαση και οι υπόλοιποι θα μπορούν να την παρακολουθούν από τους προσωπικούς τους υπολογιστές.



Σχήμα "Ad-Hoc δίκτυο"

ΥΠΟΚΕΦΑΛΑΙΟ 4.3

Η ΣΟΥΙΤΑ ΠΡΩΤΟΚΟΛΛΩΝ TCP/IP

Στην καθημερινή μας ζωή, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς πρέπει να πραγματοποιηθεί κάποια διαδικασία. Στον κόσμο των δικτύων, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα οι υπολογιστές του δικτύου. Το πρωτόκολλο είναι αυτό που καθορίζει το πώς διακινούνται τα δεδομένα, το πώς γίνεται ο έλεγχος και ο χειρισμός των λαθών, κλπ. Το Internet δεν είναι ένα απλό δίκτυο, αλλά ένα διαδίκτυο. Χρειάζεται επομένως ένα σύνολο από συμβάσεις που να καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα υπολογιστές που μπορεί να είναι διαφορετικού τύπου και να ανήκουν σε διαφορετικά δίκτυα.

Ακριβώς αυτό το σύνολο συμβάσεων προσφέρει το TCP/IP. Όλοι οι υπολογιστές που είναι συνδεδεμένοι στα χιλιάδες μικρότερα δίκτυα του Internet τρέχουν το πρωτόκολλο TCP/IP κι έτσι μιλούν μια κοινή γλώσσα που τους επιτρέπει να συνεννοούνται παρά τις διαφορές τους.

Ας υποθέσουμε ότι θέλουμε να μεταφέρουμε δεδομένα από έναν υπολογιστή που είναι συνδεδεμένος στο Internet και βρίσκεται π.χ. στην Αμερική, στο MIT, σε έναν άλλον που είναι επίσης συνδεδεμένος στο Internet και βρίσκεται π.χ. στην Ελλάδα, στο ΑΤΕΙ Θεσσαλονίκης. Μεταξύ των δύο υπολογιστών παρεμβάλλεται το “σύννεφο” του Internet, δηλαδή ένα πλέγμα από συνδέσεις και ενδιάμεσους υπολογιστές.

Το Internet χρησιμοποιεί την ακόλουθη τεχνολογία μεταγωγής πακέτων για τη μεταφορά των δεδομένων: τα δεδομένα κόβονται σε κομμάτια που ονομάζονται πακέτα και σε κάθε πακέτο μπαίνει μια “επικεφαλίδα” με τις διευθύνσεις του υπολογιστή-αποστολέα και του υπολογιστή-παραλήπτη. Σημειώνουμε ότι σε κάθε υπολογιστή του Internet αντιστοιχίζεται μία διεύθυνση που ονομάζεται διεύθυνση IP.

Το πρωτόκολλο IP είναι υπεύθυνο για το πέρασμα του πακέτου από υπολογιστή σε υπολογιστή μέσα από το “σύννεφο” των συνδέσεων. Καθώς το IP δρομολογεί το κάθε πακέτο μέσα στο δίκτυο, προσπαθεί να το παραδώσει, αλλά δεν μπορεί να εγγυηθεί ούτε ότι το πακέτο θα φτάσει στον προορισμό του ούτε ότι τα διάφορα πακέτα που αποτελούν τα αρχικά δεδομένα θα φτάσουν με τη σειρά με την οποία στάλθηκαν, ούτε ότι το περιεχόμενο των πακέτων θα φτάσει αναλλοίωτο.

Το TCP προσφέρει ένα αξιόπιστο πρωτόκολλο πάνω από το IP. Εγγυάται ότι τα πακέτα θα παραδοθούν στον προορισμό τους, ότι θα φτάσουν με τη σειρά με την οποία στάλθηκαν και ότι τα περιεχόμενα των πακέτων θα φτάσουν αναλλοίωτα (δηλαδή ακριβώς όπως στάλθηκαν). Το TCP δουλεύει ως εξής: το κάθε πακέτο δεδομένων αριθμείται. Ο υπολογιστής-παραλήπτης και ο υπολογιστής-αποστολέας, αλλά όχι οι ενδιάμεσοι υπολογιστές, παρακολουθούν τους αριθμούς των πακέτων και ανταλλάσσουν μεταξύ τους πληροφορίες. Ο παραλήπτης λαμβάνει το πρώτο πακέτο, το δεύτερο, κλπ. Σε περίπτωση που παρουσιαστεί κάποιο πρόβλημα στο δίκτυο είτε χαθεί κάποιο πακέτο κατά τη διάρκεια της μετάδοσης, το ξαναζητάει και ο αποστολέας είναι υπεύθυνος για την αναμετάδοση του. Ο παραλήπτης ελέγχει επίσης αν το περιεχόμενο των πακέτων φτάνει σωστά.

Η μέθοδος αυτή εξασφαλίζει αξιοπιστία και ταχύτητα διότι οι ενδιάμεσοι υπολογιστές δεν εκτελούν ελέγχους.

Τώρα λοιπόν που γνωρίσαμε το TCP/IP μπορούμε να δώσουμε έναν πιο “επίσημο” ορισμό του Internet: ένα δίκτυο αποτελούμενο από δίκτυα υπολογιστών που επικοινωνούν χρησιμοποιώντας το πρωτόκολλο TCP/IP. Η διαδρομή που ακολουθεί ένα πακέτο μέσα από το “σύννεφο” των συνδέσεων δεν είναι προκαθορισμένη.

Το πρωτόκολλο IP είναι υπεύθυνο για το πέρασμα ενός πακέτου δεδομένων από υπολογιστή σε υπολογιστή. Όλα τα δίκτυα που συνδέονται στο Internet “καταλαβαίνουν” τη γλώσσα IP κι έτσι μπορούν να συνεννοούνται και να ανταλλάσσουν δεδομένα με ομοιόμορφο τρόπο.

Τα δίκτυα του Internet συνδέονται μεταξύ τους με ειδικούς υπολογιστές που ονομάζονται δρομολογητές (routers) ή πύλες (gateways). Ένας router είναι λοιπόν ένας υπολογιστής που συνδέει δύο ή περισσότερα δίκτυα (που μπορεί να είναι διαφορετικού τύπου) και έτσι ανήκει σε δύο ή περισσότερα δίκτυα ταυτόχρονα.

Η δουλειά των routers είναι να δρομολογούν τα πακέτα των δεδομένων μέσα από τα διάφορα δίκτυα που αποτελούν το Internet μέχρις ότου τα επιδώσουν στον προορισμό τους. Ας δούμε πώς γίνεται αυτό:

Ας θεωρήσουμε πάλι ότι ένας υπολογιστής που βρίσκεται κάπου στο Internet θέλει να στείλει δεδομένα σε κάποιον άλλον υπολογιστή. Τα δεδομένα κόβονται σε πακέτα και το IP που εκτελείται στον υπολογιστή-αποστολέα ετοιμάζεται να στείλει το κάθε πακέτο. Εισάγει λοιπόν στην επικεφαλίδα του πακέτου τις IP διευθύνσεις του αποστολέα και του παραλήπτη και κατόπιν, βάσει των διευθύνσεων αυτών, ελέγχει αν ο παραλήπτης βρίσκεται στο ίδιο δίκτυο με τον αποστολέα.

Εάν ναι, το πακέτο στέλνεται κατευθείαν στον παραλήπτη χωρίς να χρειαστεί να διαβεί τα όρια του δικτύου. Εάν όχι, προωθείται στον router που είναι συνδεμένος με το δίκτυο. Ο router με τη σειρά του ελέγχει αν ο παραλήπτης βρίσκεται σε κάποιο από τα υπόλοιπα δίκτυα με τα οποία είναι συνδεδεμένος. Εάν ναι, το πακέτο στέλνεται κατευθείαν στον παραλήπτη στο δίκτυο αυτό. Εάν όχι, το πακέτο προωθείται στον επόμενο router, κ.ο.κ. μέχρις ότου το πακέτο προωθηθεί τελικά στον router που είναι συνδεδεμένος στο ίδιο δίκτυο με τον παραλήπτη. Το πακέτο μπορεί έτσι να περάσει από πολλούς routers μέχρις ότου φτάσει στον προορισμό του.

Οι routers διατηρούν πίνακες που προσδιορίζουν την κατεύθυνση που πρέπει να πάρει ένα πακέτο προκειμένου να φτάσει στον προορισμό του. Βάσει αυτών των πινάκων αποφασίζουν ποιος θα είναι ο επόμενος router στον οποίο θα πρέπει να προωθήσουν το πακέτο. Κάθε φορά, το πακέτο μετακινείται όλο και πιο κοντά προς τον προορισμό του έως ότου τελικά τον φτάσει.

Ένα μεγάλο πλεονέκτημα αυτής της μεθόδου είναι ότι η διαδρομή που ακολουθεί ένα πακέτο δεν είναι προκαθορισμένη, αλλά επιλέγεται δυναμικά. Έτσι, οι routers μπορούν να επιλέγουν εναλλακτικούς δρόμους για ένα πακέτο σε περίπτωση που μια συγκεκριμένη σύνδεση του δικτύου παρουσιάζει πρόβλημα και βρίσκεται προσωρινά σε αχρηστία.

ΥΠΟΚΕΦΑΛΑΙΟ 4.4

Network sockets

(ΔΙΚΤΥΑΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΧΑΜΗΛΟΥ ΕΠΙΠΕΔΟΥ)

Τα sockets είναι οι διεπαφές που επιτρέπουν τη σύνδεση σε κάθε άλλο μηχάνημα δικτύου. Για κάθε υπηρεσία ή πρόγραμμα υπάρχει ένας μοναδιαίος αριθμός (socket) για τη σωστή διαμοίραση των εισερχόμενων και εξερχόμενων δεδομένων από και προς έναν υπολογιστή. Για παράδειγμα, ένα πρόγραμμα Web server δέχεται από προεπιλογή δεδομένα και περιμένει συνδέσεις στην πόρτα 80. Ένα socket είναι ο συνδυασμός της IP διεύθυνσης και του αριθμού της πόρτας ούτως ώστε τα δύο μέρη με τις κατάλληλες συναρτήσεις να καταφέρουν να ανοίξουν μία επικοινωνία και να ανταλλάξουν δεδομένα.

Από την πλευρά του προγραμματιστή υπάρχουν δύο βασικές διαφορές μεταξύ της χρήσης μιας Unix socket και μίας TCP/IP. Πρώτον, η διεύθυνση της υποδοχής είναι ένα μονοπάτι στο σύστημα αρχείων, αντί για μια πλειάδα που περιέχει το όνομα διακομιστή και την πόρτα. Δεύτερον, ο κόμβος που δημιουργήθηκε στο σύστημα αρχείων για να εκπροσωπεί την υποδοχή παραμένει όταν η υποδοχή είναι κλειστή, και πρέπει να αφαιρεθεί κάθε φορά που ξεκινά ο διακομιστής.

Οι διευθύνσεις Socket διακρίνονται ως εξής: Ένα String χρησιμοποιείται για την οικογένεια AF_UNIX. Ένα ζευγάρι (host, port) χρησιμοποιείται για την οικογένεια διεύθυνση AF_INET, όπου host είναι ένα string που αντιπροσωπεύει είτε ένα domain name όπως «daring.cwi.nl» ή μια διεύθυνση IPv4, όπως 100.50.200.5, και το port είναι ένας ακέραιος που συμβολίζει τον αριθμό θύρας.

Για διευθύνσεις IPv4, οι δύο ειδικές μορφές είναι αποδεκτές αντί μιας διεύθυνσης υποδοχής: η κενή συμβολοσειρά αντιπροσωπεύει INADDR_ANY, και το «<broadcast>» αντιπροσωπεύει INADDR_BROADCAST.

- socket.AF_UNIX
- socket.AF_INET
- (Socket.AF_INET6)

Οι σταθερές αυτές αντιπροσωπεύουν τις οικογένειες διευθύνσεων και πρωτοκόλλων που χρησιμοποιούνται συνήθως για το πρώτο δεδομένο/arguement για μια συνάρτηση υποδοχής.

- socket.SOCK_STREAM
- socket.SOCK_DGRAM
- (Socket.SOCK_RAW)
- (Socket.SOCK_RDM)
- (Socket.SOCK_SEQPACKET)

Οι σταθερές αυτές αντιπροσωπεύουν τους τύπους υποδοχής, που χρησιμοποιούνται συνήθως για το δεύτερο argument για τη συνάρτηση υποδοχής.

Αναλυτικά τα sockets θα περιγραφούν στο μέρος της εργασίας όπου ερμηνεύεται ο κώδικας του Python script. Όμως είναι σημαντικό να αναφερθούν από τώρα ακόμη οι παρακάτω συναρτήσεις που συνήθως υπάρχουν σε βιβλιοθήκες που αφορούν low-level προγραμματισμό:

socket.accept()

Αποδοχή μιας σύνδεσης. Το socket πρέπει να είναι συνδεδεμένο σε μια διεύθυνση για ακρόαση για τις συνδέσεις. Η τιμή επιστροφής είναι ένα ζευγάρι (σύνδεση, διεύθυνση) όπου σύνδεση είναι ένα νέο αντικείμενο που μπορεί να χρησιμοποιήσει η υποδοχή για την αποστολή και λήψη δεδομένων σχετικά με τη σύνδεση, και η διεύθυνση είναι η διεύθυνση που δεσμεύεται με το socket στο άλλο άκρο της σύνδεσης.

socket.bind(address)

Ομαδοποίηση (δέσιμο) του socket. Η υποδοχή δεν πρέπει να είναι ήδη δεσμευμένη. (Η μορφή της διεύθυνσης εξαρτάται από την οικογένεια διευθύνσεων)

socket.close()

Κλείσιμο της υποδοχής. Όλες οι μελλοντικές εργασίες σχετικά με το αντικείμενο αυτό θα αποτύχουν. Το άλλο άκρο δε θα λαμβάνει πλέον τα δεδομένα. Τα sockets πολλές φορές κλείνουν αυτόματα από τον garbage-collector.

socket.connect(address)

Σύνδεση σε απομακρυσμένη υποδοχή στη συγκεκριμένη διεύθυνση. (Η μορφή της διεύθυνσης εξαρτάται από την οικογένεια διευθύνσεων).

socket.connect_ex(διεύθυνση)

Όπως socket.connect (address), αλλά επιστρέφει ένα δείκτη σφάλματος αντί της εξαίρεσης για τα σφάλματα που επιστρέφονται. Ο δείκτης σφάλματος είναι μηδέν εάν η πράξη πέτυχε, διαφορετικά είναι η τιμή της μεταβλητής errno. Αυτό είναι χρήσιμο για να στηριχθούν ασύγχρονες συνδέσεις

socket.fileno()

Επιστροφή περιγραφέα αρχείου του socket (ένας μικρού ακέραιου αριθμού). Αυτό είναι χρήσιμο με τη select.select().

socket.getpeername()

Επιστρέφει την απομακρυσμένη διεύθυνση στην οποία το socket είναι συνδεδεμένο. Αυτό είναι χρήσιμο για να μαθευτεί τον αριθμό της θύρας του απομακρυσμένου IPv4/v6 socket, για παράδειγμα. (Η μορφή της επιστρεφόμενης διεύθυνσης εξαρτάται από την οικογενειακή διεύθυνση). Σε μερικά συστήματα η λειτουργία αυτή δεν υποστηρίζεται.

socket.getsockname()

Επιστρέφει δική του διεύθυνση του socket. Αυτό είναι χρήσιμο για να μαθευτεί ο αριθμός θύρας της υποδοχής IPv4/v6 (Η μορφή της διεύθυνσης εξαρτάται από την οικογένεια διευθύνσεων).

socket.listen(backlog)
Ακούει για συνδέσεις στην υποδοχή. Το όρισμα backlog καθορίζει τον μέγιστο αριθμό των συνδέσεων όπου πρέπει να είναι τουλάχιστον 0. Η μέγιστη τιμή εξαρτάται από το σύστημα (συνήθως 5), η ελάχιστη τιμή είναι αναγκαστικά μηδέν.

socket.recv(bufsize[,flags])
Λήψη δεδομένων από το socket. Η τιμή επιστροφής είναι μια συμβολοσειρά που αντιπροσωπεύει τα δεδομένα που λαμβάνονται. Το μέγιστο ποσό των δεδομένων που πρέπει να λαμβάνονται ταυτόχρονα καθορίζεται από το bufsize. Το προεπιλογές στο μηδέν.

socket.recvfrom(bufsize[,flags])
Λήψη δεδομένων από το socket. Η τιμή επιστροφής είναι ένα ζευγάρι (string, διεύθυνση), όπου string είναι μία συμβολοσειρά που αντιπροσωπεύει τα δεδομένα που λαμβάνονται και η διεύθυνση είναι η διεύθυνση της υποδοχής για την αποστολή των δεδομένων. Το προεπιλεγμένο είναι μηδέν. (Η μορφή της διεύθυνσης εξαρτάται από την οικογένεια διευθύνσεων).

socket.send(string[,flags])
Αποστολή δεδομένων στην υποδοχή. Το socket πρέπει να συνδεθεί με ένα απομακρυσμένο socket. Το προαιρετικό όρισμα flags έχει την ίδια έννοια όπως για τη recv() παραπάνω. Επιστρέφει τον αριθμό των bytes για αποστολή. Οι εφαρμογές είναι υπεύθυνες για τον έλεγχο ότι όλα τα δεδομένα έχουν σταλεί. Εάν μόνον ορισμένα από τα δεδομένα διαβιβάσθηκαν, η εφαρμογή θα πρέπει να επιχειρήσει την παράδοση των υπόλοιπων δεδομένων.

socket.sendall(string[,flags])
Αποστολή δεδομένων στην υποδοχή. Το socket πρέπει να συνδεθεί με μία απομακρυσμένη. Το προαιρετικό όρισμα σημαίες έχει την ίδια έννοια όπως για τη recv() παραπάνω. Σε αντίθεση με τη send(), αυτή η μέθοδος συνεχίζει να στέλνει τα δεδομένα από την ουρά σε σειρά έως ότου είτε όλα τα δεδομένα έχουν σταλεί είτε υπάρξει ένα σφάλμα. Τίποτα δεν επιστρέφεται σε επιτυχή αποστολή. Σε περίπτωση λάθους, μια εξαίρεση εμφανίζεται, και δεν υπάρχει κανένας τρόπος για να προσδιοριστεί πόσα δεδομένα, αν υπάρχουν, εστάλησαν επιτυχώς.

socket.sendto(string[,flags],address)
Αποστολή δεδομένων στην υποδοχή (socket). Η υποδοχή δεν πρέπει να συνδεθεί με μία απομακρυσμένη υποδοχή, δεδομένου ότι η υποδοχή προορισμού καθορίζεται από τη διεύθυνση. Το προαιρετικό όρισμα flags έχει την ίδια έννοια όπως για τη recv() παραπάνω. Επιστρέφει τον αριθμό των bytes που έχουν αποσταλεί. (Η μορφή της διεύθυνσης εξαρτάται από την οικογένεια διευθύνσεων).

socket.family
Η address-family της υποδοχής.

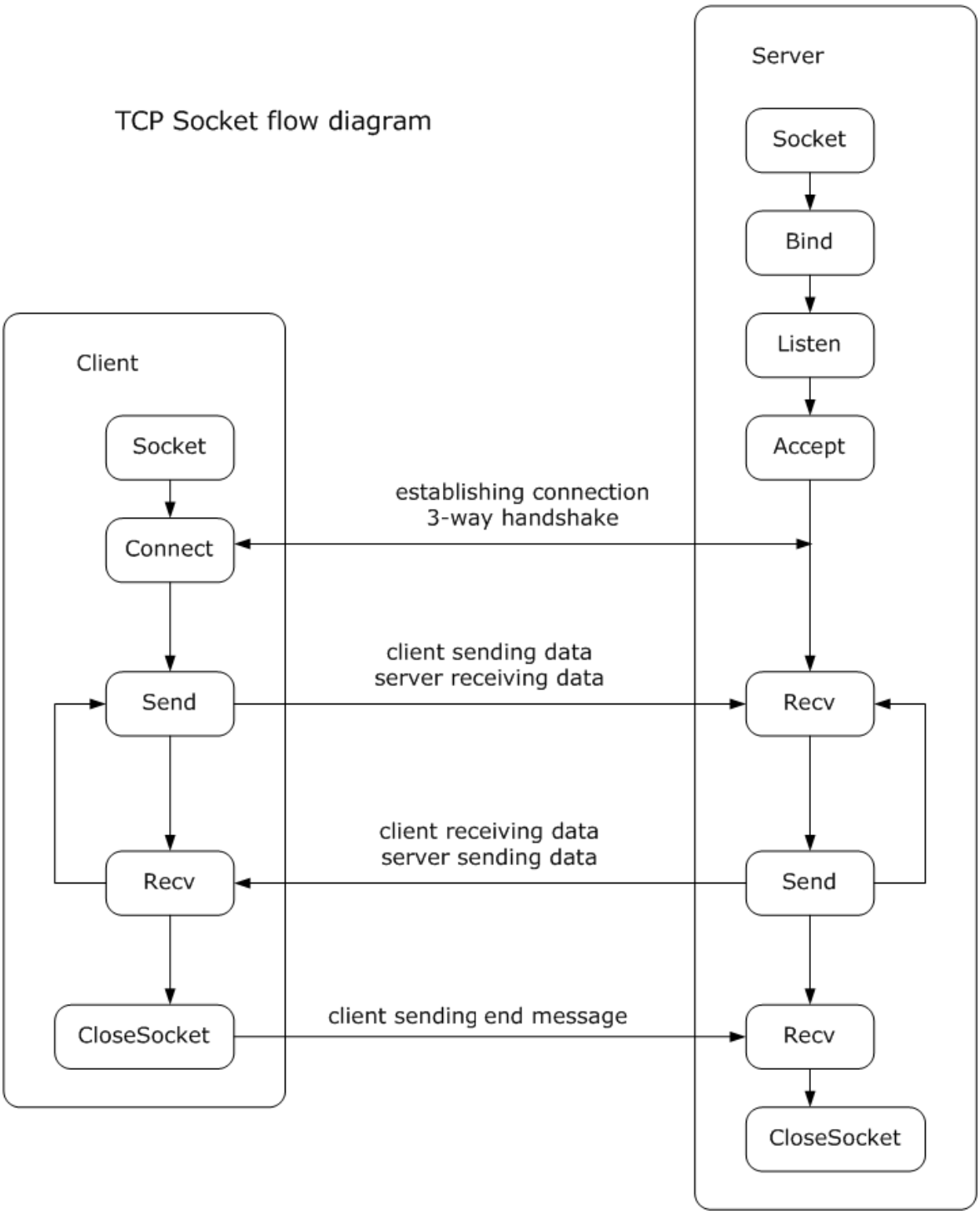
socket.type
Ο τύπος της υποδοχής.

socket.proto
Το πρωτόκολλο της υποδοχής.

socket.shutdown(how)
Τερματισμός της λειτουργίας του ένας ή και των δύο άκρων της σύνδεσης. Αν το how όρισμα είναι SHUT_RD, περαιτέρω λήψεις δεν επιτρέπονται. Αν το how όρισμα είναι SHUT_WR, περαιτέρω αποστολές δεν επιτρέπονται. Αν το όρισμα how είναι SHUT_RDWR, δεν επιτρέπονται ούτε αποστολές ούτε λήψεις. Ανάλογα με την πλατφόρμα (το λειτουργικό σύστημα), κλείνοντας το ένα άκρο της σύνδεσης μπορεί να προκαλέσει το κλείσιμο και της άλλης μεριάς.

Αυτές είναι γενικά οι συναρτήσεις που χρειάζονται για την υλοποίηση μίας εφαρμογής όπως αυτή της εργασίας. Ανάλογα με τη γλώσσα προγραμματισμού, το λειτουργικό σύστημα αλλά και το σχεδιασμό και την υλοποίηση που θα πραγματοποιηθεί, η επικοινωνία μέσω των δικτυακών υποδοχών (sockets) ποικίλλει.

Στην επόμενη σελίδα το σχεδιάγραμμα δείχνει ένα παράδειγμα client/server αρχιτεκτονικής όπου μερικές από τις παραπάνω συναρτήσεις (οι πιο ζωτικές για τη λειτουργία μιας εφαρμογής) παίρνουν μέρος.



Διάγραμμα "Client/Server επικοινωνία με sockets και οι συναρτήσεις που χρησιμοποιούνται"

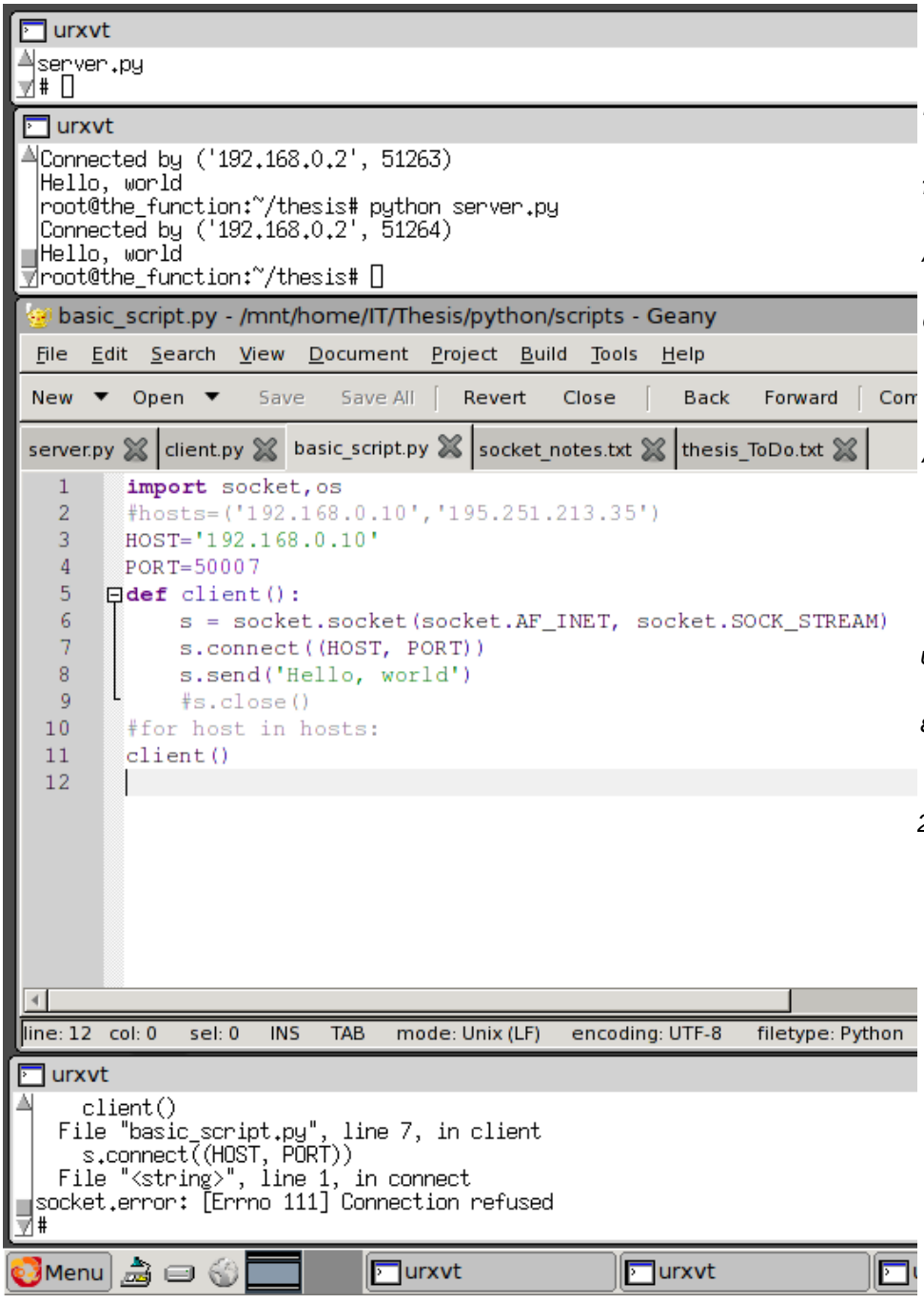
Τέλος, ακολουθεί ένα πολύ απλό παράδειγμα κώδικα σε γλώσσα Python για την επικοινωνία ενός πελάτη με έναν διακομιστή μέσω των δικτυακών υποδοχών (network sockets). Ο server απαντάει στον client τα ίδια δεδομένα που λήφθηκαν και η σύνδεση τερματίζεται.

```
# Το πρόγραμμα του server

import socket

HOST = ''               # Συμβολικό όνομα (όλα τα interfaces)
PORT = 50007            # Μη χρησιμοποιούμενη θύρα

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()

print 'Σύνδεση από', addr

while 1:
   #Σημαντικό: Ο server δε στέλνει και λαμβάνει send()/recv() στο socket το οποίο ακούει αλλά σε ένα νέο που επιστρέφει η accept().
```

```
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
conn.close()
```

```
# Το πρόγραμμα του πελάτη

import socket

HOST = '192.168.1.12'     # Ο απομακρυσμένος υπολογιστής
PORT = 50007              # Η θύρα που χρησιμοποιείται από τον server

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

s.send('Hello, world')
data = s.recv(1024)
s.close()

#Εκτύπωση των δεδομένων χρησιμοποιώντας τη συνάρτηση repr()
```

```
print 'Received', repr(data)
```



Εικόνα "Δοκιμαστικό περιβάλλον λειτουργίας στατικών sockets στο φορητό λειτουργικό σύστημα Puppy Linux σε έναν υπολογιστή με επεξεργαστή Pentium 2 στα 300 MHZ!"

ΥΠΟΚΕΦΑΛΑΙΟ 4.5

ΤΑ ΕΡΓΑΛΕΙΑ wireless-tools

Το Wireless Tools είναι ένα πακέτο του Linux με εντολές (text-based εργαλεία), που προορίζονται για την υποστήριξη και τη διευκόλυνση της διαμόρφωσης των ασύρματων συσκευών που χρησιμοποιούν το Linux Wireless Extention. Είναι ένα ουσιαστικό μέρος της εργασίας καθώς προσφέρει ευελιξία αλλά και καθοριστική λειτουργία στην εφαρμογή.

Οι εντολές αυτής της σουίτας εργαλείων είναι οι εξής:

ifrename

Η ifrename επιτρέπει τη μετονομασία των ασύρματων διεπαφών δικτύου (interfaces) με βάση διάφορα κριτήρια για ορισμό νέου ονόματος σε κάθε interface.

Από προεπιλογή, τα ονόματα των interfaces είναι δυναμικά, και κάθε προσαρμογέας δικτύου αντιστοιχίζεται στο πρώτο διαθέσιμο όνομα (eth0, eth1 ...), ενώ οι διασυνδέσεις του δικτύου που δημιουργήθηκαν μπορεί να διαφέρουν. Η ifrename επιτρέπει στο χρήστη να αποφασίσει τι όνομα μια διεπαφή δικτύου θα έχει. Μπορεί να χρησιμοποιήσει μια ποικιλία από επιλογές για να ταιριάζει με τα ονόματα με τις διασυνδέσεις του δικτύου για το σύστημα. Ο πιο κοινός επιλογέας είναι η διεύθυνση MAC.

Η ifrename πρέπει να εκτελεστεί πριν ενεργοποιηθούν οι ασύρματοι προσαρμογείς, όπου είναι και ο λόγος που αυτό είναι κυρίως χρήσιμο σε διάφορα αρχικά σενάρια του λειτουργικού (init, hotplug), αλλά σπάνια χρησιμοποιούνται άμεσα από το χρήστη. Από προεπιλογή μετονομάζει όλες τις διασυνδέσεις του συστήματος με τη χρήση αντιστοίχησης που ορίζεται στο /etc/iftab.

Iwconfig

Η iwconfig χρησιμοποιείται για την εμφάνιση και την αλλαγή των παραμέτρων της διασύνδεσης δικτύου που είναι ειδικά για την ασύρματη λειτουργία (π.χ. το όνομα του interface, συχνότητα, SSID). Μπορεί επίσης να χρησιμοποιηθεί για την εμφάνιση των ασύρματων στατιστικών στοιχείων (που προέρχονται από το /proc/net/wireless).

Στα συστήματα διανομής δωρεάν λογισμικού Berkeley UNIX, ο ρόλος του iwconfig γίνεται με μια διευρυμένη εντολή ifconfig.

iwevent

Η iwevent εμφανίζει τα ασύρματα γεγονότα που δημιουργούνται από τους οδηγούς και τη θέσπιση αλλαγών που λαμβάνονται μέσω της υποδοχής RTNetlink. Κάθε γραμμή εμφανίζει το συγκεκριμένο ασύρματο γεγονός που περιγράφει τι συνέβη στην καθορισμένη διασύνδεση. Δεν παίρνει κανένα όρισμα.

iwgetid

Η iwgetid αναφέρει το ESSID ή το σημείο πρόσβασης του ασύρματου δικτύου που χρησιμοποιείται. Εξ ορισμού θα εκτυπώσει το ESSID της συσκευής. Η πληροφορία που γνωστοποιείται είναι η ίδια με εκείνη που δείχνεται από την iwconfig, αλλά η iwgetid είναι ευκολότερο να ενσωματωθεί σε διάφορα σενάρια (scripts).

iwlist

Η iwlist χρησιμοποιείται για τη σάρωση των διαθέσιμων ασύρματων δικτύων και εμφανίζει πρόσθετες πληροφορίες σχετικά με αυτές που δεν εμφανίζονται με το iwconfig. Χρησιμοποιείται για την εκτύπωση μιας κατηγορίας πληροφοριών εμφανίζοντας σε λεπτομερή μορφή όλες τις πληροφορίες σχετικά με αυτή την κατηγορία, συμπεριλαμβανομένων των πληροφοριών που ήδη φαίνονται από την iwconfig.

Εικόνα "Η επιλογές της iwlist"

Η εντολή κατά κύριο λόγο χρησιμοποιείται για να παράγει μια λίστα με κοντινά σημεία ασύρματης πρόσβασης, διευθύνσεις MAC και τα SSIDs τους.

iwpriv

Η iwpriv χρησιμοποιείται για το χειρισμό των παραμέτρων και των ρυθμίσεων συγκεκριμένα για κάθε οδηγό (σε αντίθεση με το iwconfig η οποία χρησιμοποιείται γενικά).

Χωρίς κανένα όρισμα, η iwpriv απαριθμεί τις διαθέσιμες private εντολές σε κάθε περιβάλλον εργασίας, καθώς και τις παραμέτρους που χρειάζονται. Χρησιμοποιώντας αυτές τις πληροφορίες, ο χρήστης μπορεί να εφαρμόσει αυτές τις εντολές σε συγκεκριμένες ασύρματες διασυνδέσεις δικτύου.

iwspy

Η iwspy χρησιμοποιείται για την παρακολούθηση ενός σετ κόμβων και για να καταγράφει την ποιότητα της καθεμιάς σύνδεσης με αυτούς.

Οι πληροφορίες που συγκεντρώνονται είναι οι ίδιες με αυτές που υπάρχουν στο /proc/net/wireless: η ποιότητα της σύνδεσης, η ισχύς του σήματος και το επίπεδο θορύβου. Αυτές οι πληροφορίες ενημερώνονται κάθε φορά που ένα νέο πακέτο παραλαμβάνεται, έτσι ώστε κάθε διεύθυνση της λίστας προσθέτει κάποια επιβάρυνση στο πρόγραμμα οδήγησης. Η λειτουργία αυτή είναι χρήσιμη κυρίως σε ad hoc και master λειτουργία.

ΕΠΙΛΟΓΟΣ

Αυτά που αναφέρθηκαν στο κεφάλαιο αυτό είναι ένα μέρος της μελέτης και έρευνας που έγινε σε όλα αυτά τα θέματα. Αναφέρθηκαν οι βασικές λειτουργίες, οι χρήσεις, τα εργαλεία, τα πρωτόκολλα και παραδείγματα που βοηθάνε στην κατανόηση των επόμενων κεφαλαίων καθώς και στην ανάλυση της πρακτικής υλοποίησής τους.

Στη συνέχεια της εργασίας όπου τα κεφάλαια παρουσιάζονται με τη σειρά όπου και οι αντίστοιχες τεχνολογίες και υλοποιήσεις μελετήθηκαν και αντίστοιχα πραγματοποιήθηκαν θα δοθεί η ευκαιρία στην επανάληψη των πιο βασικών μελών την θεμάτων που αναφέρθηκαν.

Στο επόμενο κεφάλαιο η εργασία περνάει σε διαφορετικό πεδίο καθώς αναλύει το σχεδιασμό του έργου.

ΚΕΦΑΛΑΙΟ 5

Ο ΘΕΩΡΗΤΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ

ΕΙΣΑΓΩΓΗ

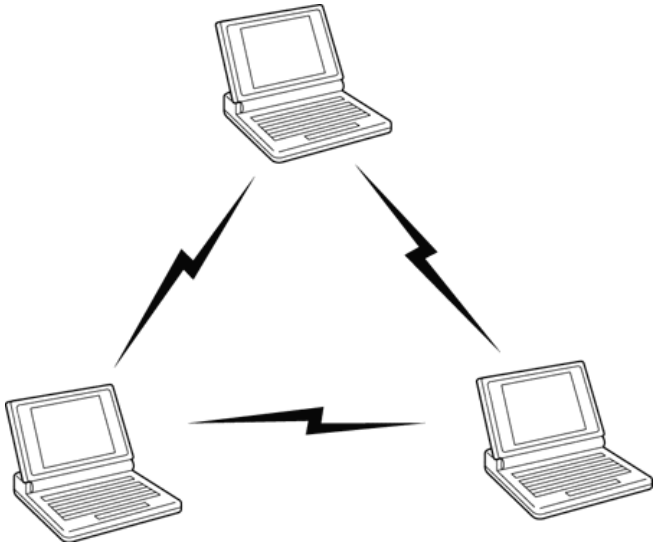
Σε αυτό το κεφάλαιο αναπτύσσεται το concept της εργασίας. Γίνεται ανάλυση της εξέλιξης της διαδικασίας σχεδίασης του έργου και παρουσιάζονται μερικά πρωτότυπα όπου συντέλεσαν τη βάση για την μετέπειτα εξέλιξη της εργασίας.

Είναι σημαντικό να σημειωθεί ότι η γενική λειτουργία της εφαρμογής ήταν ορισμένη σε μεγάλο βαθμό κατά την ανάληψη της πτυχιακής εργασίας. Η λεπτομερής ανάπτυξη και η κατεύθυνση υλοποίησης είναι κάτι που θα μπορούσε να έχει υλοποιηθεί με διάφορους τρόπους. Οι λόγοι που η εφαρμογή αναπτύχθηκε με τον συγκεκριμένο τρόπο ήταν η καλύτερη πιθανή λύση δεδομένου του συνδυασμού της αρχικής ιδέας και μιας πρακτικής υλοποίησης.

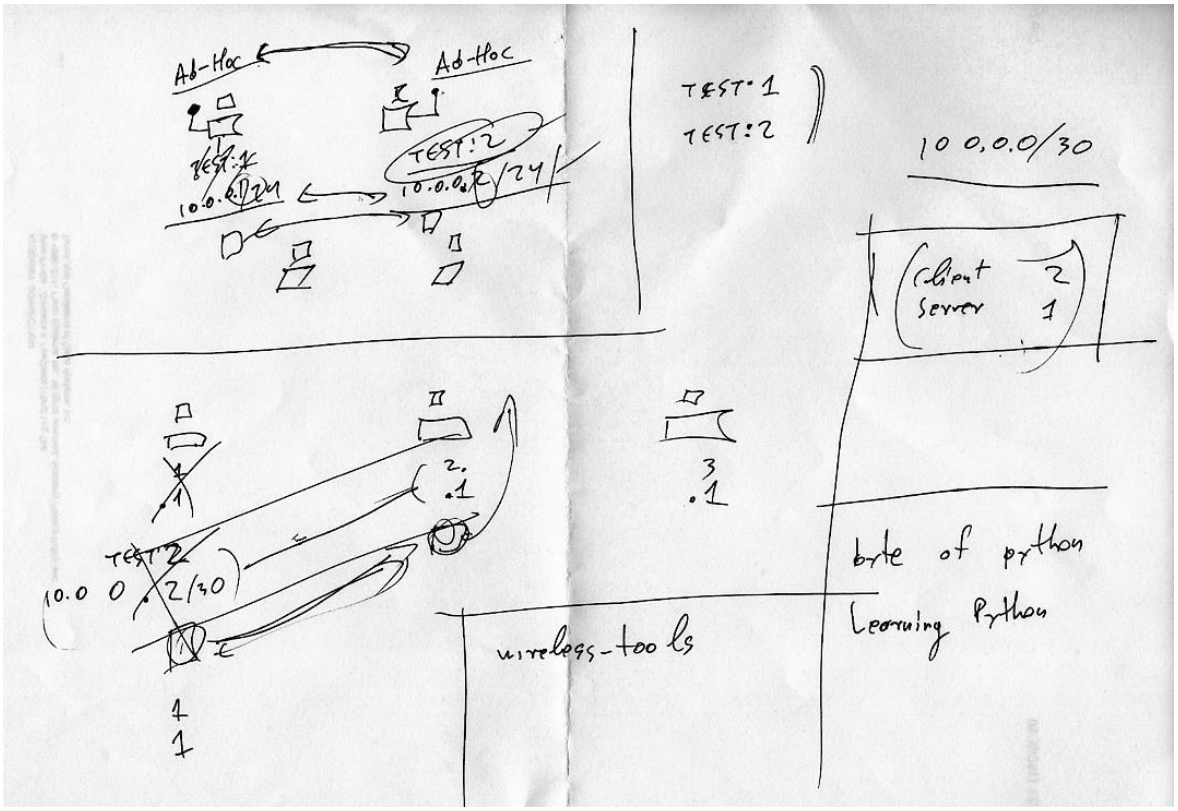
ΥΠΟΚΕΦΑΛΑΙΟ 5.1

Η ΑΡΧΙΚΗ ΙΔΕΑ

Στόχος είναι να διαπιστωθεί κατά πόσο η θεωρητική προσέγγιση, που όπως προαναφέρθηκε, ορίστηκε με την ανάθεση της εργασίας, μπορεί να αποδειχθεί στην πράξη μέσω της εφαρμογής. Αυτό είναι κάτι που εξαρτάται από πολλούς παράγοντες, οι περισσότεροι εκ των οποίων δημιουργούν περιορισμούς αλλά και ευκολίες ανάπτυξης κατά την υλοποίηση.



Για να κατανοήσουμε τον θεωρητικό σχεδιασμό είναι απαραίτητο να υπενθυμίσουμε κάποια πράγματα για την εφαρμογή. Η εφαρμογή με το όνομα “dispel” είναι ουσιαστικά ένα script, ή καλύτερα δύο scripts, γραμμένα στη γλώσσα προγραμματισμού Python για λειτουργία σε GNU/Linux λειτουργικά συστήματα. Απαραίτητη προϋπόθεση είναι η παρουσία μίας τουλάχιστον ασύρματης κάρτας δικτύου σε κάθε υπολογιστή που παίρνει μέρος στην επικοινωνία. Πρέπει επίσης η ομάδα εργαλείων wireless-tools να είναι εγκατεστημένη και η έκδοση της Python να είναι η 2.6.x ή η 2.7.x.



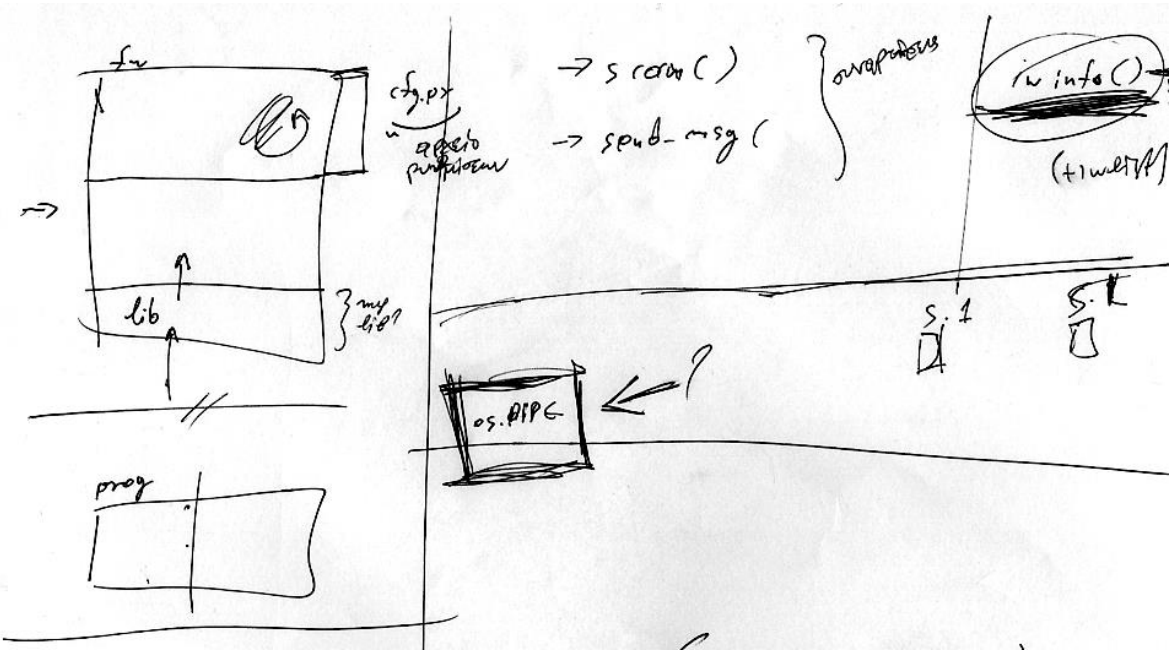
Εικόνα "Σημειώσεις από την αρχική σχεδίαση της λειτουργίας της εφαρμογής dispel"

Παρακάτω θα αναφερθούν μερικά από τα βασικά σκέλη της λειτουργίας του προγράμματος, έτσι όπως ορίστηκαν στα πρώτα στάδια σχεδιασμού και ανάπτυξης.

- Για την επικοινωνία των επιμέρους υπολογιστών θα χρησιμοποιούνται private IP διευθύνσεις. Οι προεπιλεγμένες τιμές είναι:
  - SERVER_IP="10.8.0.1"
  - CLIENT_IP="10.8.0.2"
  - SUBNET_MASK="255.255.255.252"

Η διεύθυνση 10.8.0.1 θα υποδηλώνει ότι ο υπολογιστής είναι σε λειτουργία server, ενώ η διεύθυνση 10.8.0.2 ότι είναι σε λειτουργία client (πελάτη).

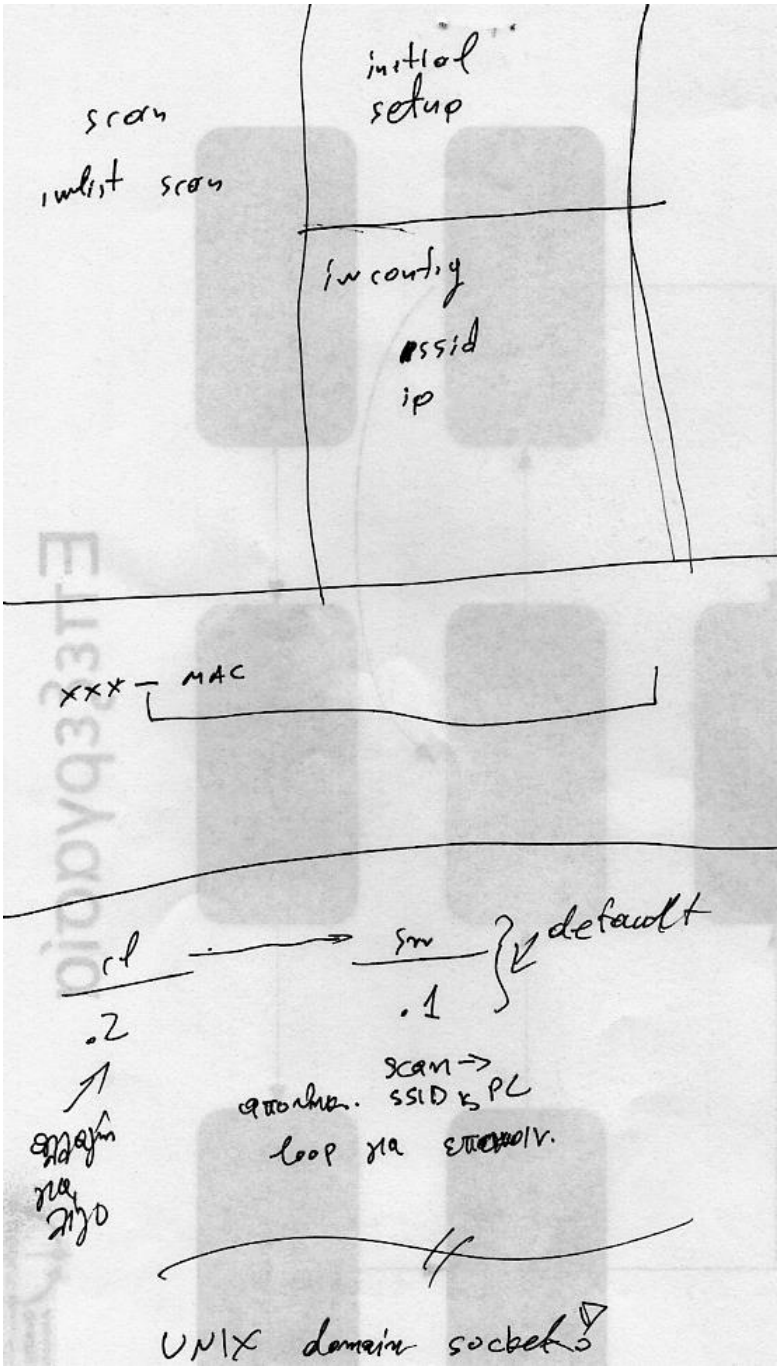
- Το SSID του κάθε μηχανήματος θα παίζει πολύ σημαντικό ρόλο. Το πρώτο μέλος του θα είναι κοινό για να μπορέσουν οι υπόλοιποι υπολογιστές να αντιληφθούν ότι ο υπολογιστής τρέχει την εφαρμογή dispel και το δεύτερο μέλος θα είναι μοναδικό ούτως ώστε το κάθε μηχάνημα να ξεχωρίζει μοναδιαία μέσα στο πλήθος των ασύρματων συσκευών που παίρνουν μέρος στην επικοινωνία. Η μορφή που μπορεί να έχει ένα SSID είναι "dispel%SA25-FA014BB1", όπου "dispel%SA25-" είναι το κοινό μέρος.



Εικόνα "Αρχικό draft σχεδίασης της επικοινωνίας των scripts με τον φλοιό του Λ.Σ. Και το πακέτο wireless-tools"

- Η εντολή iwlist scan[ning] θα εντοπίζει αυτόματα τα ασύρματα δίκτυα της περιοχής που ο υπολογιστής μπορεί να ανιχνεύσει. Αυτό θα γίνεται με την επικοινωνία της Python με τον φλοιό του λειτουργικού συστήματος και τη χρήση της αντίστοιχης εντολής.

- Μετά από τη σάρωση των ασύρματων δικτύων τα SSIDs που εντοπίστηκαν θα αποθηκεύονται για μελλοντική (σε milliseconds) επικοινωνία. Μετά, ο κάθε υπολογιστής θα επικοινωνεί με τους υπόλοιπους μέσω ενός loop για τον καθέναν ξεχωριστά.
- Οι πληροφορίες που θα αντλούνται από το shell θα φιλτράρονται (βιβλιοθήκη “re” της Python) για να κρατούνται μόνο οι πληροφορίες που μας ενδιαφέρουν.
- Οι υπολογιστές θα γίνονται clients (IP διεύθυνση που τελειώνει σε .2) για ένα πολύ σύντομο χρονικό διάστημα όπου θα "συνδέονται στον" άλλον (server) υπολογιστή.
- Αν υπάρχουν πολλοί υπολογιστές κοντά, η διαδικασία επικοινωνίας θα γίνεται από κάθε έναν υπολογιστή ξεχωριστά προς όλους τους υπολογιστές που θα βρεθούν.
- Οι εντολές iwconfig (ifconfig για wireless) θα χρησιμοποιούνται για τον ορισμό των IP διευθύνσεων κατά το initial setup αλλά και τις αλλαγές την ώρα που το πρόγραμμα τρέχει.
- Οι ρυθμίσεις θα μπορούν να αποθηκεύονται σε ένα αρχείο conf.py στον ίδιο κατάλογο. Ρυθμίσεις όπως το interval της επικοινωνίας (π.χ. σε δευτερόλεπτα), το ποιο interface θα είναι σε λειτουργία, ποιο ασύρματο κανάλι θα χρησιμοποιηθεί, τις διευθύνσεις του επιπέδου Internet, τη θύρα (port) επικοινωνίας, το τι δεδομένα θα ανταλλάσσονται μεταξύ των υπολογιστών κ.λ.π.
- Η εφαρμογή θα μπορεί να χειρισθεί άμεσα από τον χρήστη μέσω του text-based μενού που θα δημιουργηθεί.
- Θα υπάρχουν συναρτήσεις για την εμφάνιση πληροφοριών και βοήθειας (εντολές μενού κ.λ.π.).
- Το τρέξιμο του script θα μπορεί να παίρνει κάποια βασικά ορίσματα.
- Θα γίνεται έλεγχος για τον αν ο χρήστης είναι ο 'root'. Αυτό είναι αναγκαστικό για την αλλαγή ρυθμίσεων από το λειτουργικό σύστημα.



Εικόνα "Αρχική σχεδίαση της επικοινωνίας του loop με το αρχείο ρυθμίσεων και η χρήση των εντολών iwconfig και iwlist"

- Κατά την εκκίνηση του προγράμματος θα ελέγχεται αν η Python και τα εργαλεία wireless-tools είναι εγκατεστημένα.
- Το μοναδικό μέρος του SSID θα μπορεί να ορίζεται δυναμικά παίρνοντας κομμάτι από τη διεύθυνση MAC του interface. Αυτό βοηθάει στην αυτόματη λειτουργία της εφαρμογής καθώς και την ελάττωση των ορισμάτων που πρέπει να θέσει ο χρήστης. Να σημειωθεί πως είναι μόνο μία δυνατότητα, το μοναδικό μέρος του SSID μπορεί να οριστεί στο αρχείο conf.py.
- Οι εντολές Linux θα ενθυλακώνονται και παραμετροποιούνται μέσα στις μεθόδους που θα οριστούν από την Python. Δύο από αυτές τις μεθόδους που θα οριστούν στον κώδικα θα είναι υπεύθυνες για τη σωστή εκκίνηση και τον καθαρό τερματισμό του λογισμικού.
- Η ανταλλαγή των δεδομένων θα γίνεται μέσα στο loop όπου οι εκάστοτε υπολογιστές θα ανοίγουν συνδέσεις μεταξύ τους.

ΕΠΙΛΟΓΟΣ

Η σχεδίαση της εφαρμογής δεν είναι απλά ένας τρόπος υλοποίησης του κώδικα. Το concept που θα αναπτυχθεί είναι ένας νέος, δοκιμαστικός, ερευνητικός τρόπος επικοινωνίας ασύρματων συσκευών.

Για την επικοινωνία αυτή οι υπολογιστές επικοινωνούν μεταξύ τους με το άνοιγμα (και αντίστοιχα το άμεσο κλείσιμο όταν μεταφερθούν τα δεδομένα) πολλαπλών συνδέσεων Ad-Hoc. Φυσικά κάθε φορά ένας υπολογιστής μπορεί να επικοινωνεί ταυτόχρονα μόνο με έναν άλλο, αλλά λόγω της γρήγορης εναλλαγής θα γίνει προσπάθεια προσημείωσης πολλαπλού δικτύου όπου όλα τα μηχανήματα επικοινωνούν μεταξύ τους μέσω του ίδιου μέσου.

Καθώς το θεωρητικό μέρος έχει ορισθεί, στο επόμενο κεφάλαιο αναλύεται ο τρόπος με τον οποίο αναπτύχθηκε το λογισμικό.

ΚΕΦΑΛΑΙΟ 6

Η ΠΡΑΚΤΙΚΗ ΥΛΟΠΟΙΗΣΗ

ΕΙΣΑΓΩΓΗ

Έπειτα από την ανάλυση του σχεδιασμού του θεωρητικού μοντέλου της εφαρμογής σειρά έχει η ανάλυση του κώδικα και των πρακτικών λειτουργιών της εφαρμογής. Γίνεται παρουσίαση του κώδικα και επεξήγηση των βασικών μερών του. Στο επόμενο υποκεφάλαιο υπάρχουν εικόνες που αντιστοιχούν σε διάφορα στάδια υλοποίησης της εφαρμογής. Η βελτίωση του κώδικα ήταν διαρκής καθώς μία σειρά από επιμέρους προβλήματα έπρεπε να αντιμετωπιστεί.

Η δυσκολία της ανάπτυξης έγκειται στο ότι πολλές διαφορετικές αρχιτεκτονικές και τεχνολογίες πρέπει να συνδυαστούν για να δημιουργήσουν μία δυναμική υλοποίηση του έργου.

ΥΠΟΚΕΦΑΛΑΙΟ 6.1

ΤΟ ΑΡΧΕΙΟ ΡΥΘΜΙΣΕΩΝ conf.py

Το conf.py είναι το αρχείο/script κειμένου όπου ορίζονται οι ρυθμίσεις για τη λειτουργία της εφαρμογής. Στα πρώτα στάδια ανάλυσης ήταν ενσωματωμένο μέσα στο dispel.py script όπου είναι υπεύθυνο για την επικοινωνία (το βασικό μέρος του κώδικα) αλλά γρήγορα έγινε αντιληπτό πως ένα ξεχωριστό αρχείο ρυθμίσεων εγγυάται εύκολη παραμετροποίηση και πιο ευανάγνωστο κώδικα.

Επίσης το αρχείο ρυθμίσεων μπορεί πολύ εύκολα να αντιγραφεί από τους χρήστες στο κάθε μηχάνημα ώστε οι υπολογιστές να επικοινωνούν με τις ίδιες ρυθμίσεις χωρίς την ανάγκη επίδρασης του χρήστη στον ορισμών των μεταβλητών των ρυθμίσεων του προγράμματος.

dispel05f.py conf.py

```
#Example:
#CRYPTOGRAPHY="open"


#INTERFACE valid options: "<interface-string>"
#Example:
#INTERFACE="wlan0"
INTERFACE='wlan0'


#PORT valid options: <integer>
#Example:
#PORT=50008
PORT=50048


#SERVER_IP, CLIENT_IP and SUBNET_MASK valid options: <string-ip>
#Example:
#CLIENT_IP="192.168.0.2"
SERVER_IP="10.8.0.1"
CLIENT_IP="10.8.0.2"
SUBNET_MASK="255.255.255.252"


#SENDING_DATA valid options: "TEST", "<path/file>"
#Example:
#SENDING_DATA="/proc/net/wireless"
SENDING_DATA="/proc/net/socstat/"
```

Εικόνα "Από την προετοιμασία του αρχείου ρυθμίσεων (αρχική έκδοση)"

Παρακάτω φαίνεται ο κώδικας της τελευταίας έκδοσης του conf.py. Η επιλογές θα μπορούσαν να ήταν περισσότερες ή και μόνο δύο ή τρεις. Επιλέχθηκε να μπορούν να οριστούν οι ρυθμίσεις που είναι σημαντικές τόσο για τη λειτουργία της εφαρμογής όσο και για την καλύτερη ερμηνεία των συμπερασμάτων.

Ο κώδικας του αρχείου ρυθμίσεων conf.py:

```
#!/usr/bin/env python


#*****************************************************************

# dispel configuration file                                     *

# Please see the valid options of each variable/setting below. *

#*****************************************************************


#Set a unique dispel ID:

#DISPEL_ID valid options: "MAC", "<STRING-NICKNAME>"


#When DISPEL_ID="MAC" the dispel id is automatically set from retrieved information of
the MAC address.

#Example: DISPEL_ID="Maria Callas"

DISPEL_ID="MAC"
```

```
#The wireless interface to use:

#INTERFACE valid options: "<STRING-INTERFACE>"

INTERFACE="wlan0"




#The channel used for the wireless connection:

#CHANNEL valid options: <integer[0-13]>

#Integer for the number of the channel.

CHANNEL=6


#Server and client IP settings:

#SERVER_IP, CLIENT_IP and SUBNET_MASK valid options: "<STRING-ADDRESS>"

SERVER_IP="10.8.0.1"

CLIENT_IP="10.8.0.2"

SUBNET_MASK="255.255.255.252"


#The port that dispel will use:
```

#PORT valid options: <integer>

PORT=50048

#Maximum queued connections:

#Only change the setting if you know what you're doing.

#BACKLOG valid options: <integer[0-5]>

BACKLOG=5

#Maximum data received at once (a small power of 2 is preferred):

#BUF_SIZE valid options: <integer>

BUF_SIZE=2048

#How often (in seconds) the daemon will send data:

#INTERVAL valid options: <integer>

INTERVAL=3

#What information the program will send:

#When not setting "TEST", set which command's results to send.

#SENDING_DATA valid options: "TEST", "<COMMAND>"

```
#Example: SENDING_DATA="TEST"

#Example: SENDING_DATA="iw dev wlan0 link"

#Example: SENDING_DATA="iw dev wlan1 station dump"

SENDING_DATA="hostname ; cat /proc/net/dev"



#Under construction for future use:

#CRYPTOGRAPHY valid options: "off", "open", etc...

#Example:

#CRYPTOGRAPHY="open"



#Set the colors:

#Only change the color settings if you know what you're doing.

#COLOR valid options: '\033[<integer>m'

#See examples below:

OTHER='\033[96m'

HEADER = '\033[95m'

OKBLUE = '\033[94m'

OKGREEN = '\033[92m'

WARNING = '\033[93m'
```

```
FAIL = '\033[91m'

ENDC = '\033[0m'
```

Όπως φαίνεται από τον κώδικα και τα συνοδευόμενα σχόλια οι ρυθμίσεις που μπορούν να οριστούν είναι:

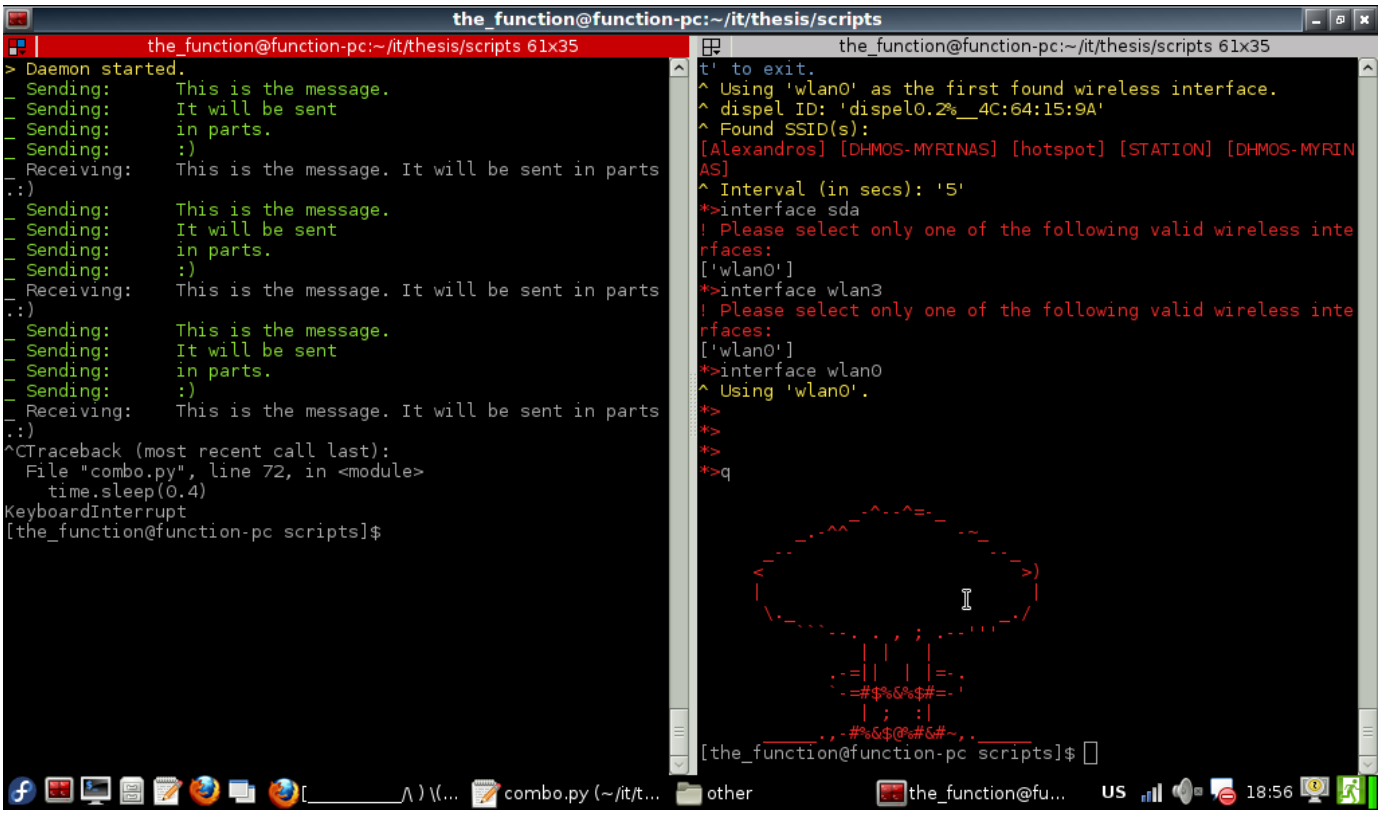
- Το αν το δεύτερο μέρος του SSID θα ορίζεται δυναμικά από τη διεύθυνση MAC ή αν θα είναι η συμβολοσειρά που ορίζει ο χρήστης.
- Το interface που θα χρησιμοποιείται για την ασύρματη επικοινωνία μεταξύ των υπολογιστών.
- Το ασύρματο κανάλι για την ανταλλαγή πληροφοριών.
- Οι IP διευθύνσεις και το δίκτυο που θα ορίζονται στους υπολογιστές στις καταστάσεις:
  - Server (διακομιστή)
  - Client (πελάτη)

- Το backlog (Το όρισμα backlog καθορίζει τον μέγιστο αριθμό των συνδέσεων όπου πρέπει να είναι τουλάχιστον 0).
- Το μέγεθος του buffer για τα δεδομένα.
- Το interval, δηλαδή κάθε πόσα δευτερόλεπτα ο κάθε υπολογιστής θα στέλνει δεδομένα ως client.
- Τον τύπο των δεδομένων που θα στέλνονται (μεταβλητή SENDING_DATA). Επιλογές είναι ένα δοκιμαστικό κείμενο ή το output ενός αρχείου/εντολής με (στατιστικά) δεδομένα για τον υπολογιστή
- Τέλος, ορίζονται οι μεταβλητές χρωμάτων για την ευκολότερη ανάγνωση των μηνυμάτων κατά τη διάρκεια λειτουργίας της εφαρμογής.

ΥΠΟΚΕΦΑΛΑΙΟ 6.2

dispel.py – ΤΟ ΒΑΣΙΚΟ SCRIPT ΤΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ

Έχοντας αναλύσει τον σχεδιασμό, τα εργαλεία και τις τεχνολογίες που χρησιμοποιούνται και έχοντας περιγράψει τη βασική λειτουργία της εφαρμογής καθώς και το αρχείο των (απαραίτητων) ρυθμίσεων θα παρουσιαστεί ο κώδικας του dispel, δηλαδή του βασικού script της εφαρμογής όπου είναι υπεύθυνο για την επικοινωνία μεταξύ των συσκευών.



Εικόνα "Η πρώτη δοκιμαστική (Alpha) έκδοση του dispel στο λειτουργικό σύστημα Fedora 16"

Ακολουθεί ο κώδικας της εφαρμογής όπως υλοποιήθηκε στην έκδοση **“06g”** καθώς και η επεξήγησή του.

```
#!/usr/bin/env python
```

Εδώ δηλώνουμε ότι για να τρέξει το script χρησιμοποιούμε τη γλώσσα python (environment: python)

```
#*************************************************************

# dispel core script                                  *

#*************************************************************


import subprocess,re,sys,os,conf,select,socket,time

from subprocess import Popen, PIPE, STDOUT
```

Γίνεται εισαγωγή των παραπάνω βιβλιοθηκών όπου θα χρησιμοποιηθούν από την εφαρμογή σε πολλές περιπτώσεις.

- *re: για το φιλτράρισμα κειμένου*
- *subprocess: για την επικοινωνία της Python με τον φλοιό από το λειτουργικό σύστημα*
- *os: για τη γρήγορη εκτέλεση εντολών στο Linux shell*
- *select και socket: είναι οι βασικές βιβλιοθήκες για τη δημιουργία, τη λειτουργία, τη διαχείριση και την επικοινωνία των sockets*
- *time: χρήση της για την περιοδική αποστολή δεδομένων (σε δευτερόλεπτα)*
- *conf: εισαγωγή των μεταβλητών από το αρχείο ρυθμίσεων*

```
dispel_version="0.6g"
```

Η έκδοση του script

ΠΡΟΣΟΧΗ: Διαφορετικές εκδόσεις της εφαρμογής είναι εσκεμμένα μη συμβατές μεταξύ τους

```
dispel_version_id="dispel"+dispel_version+"%__"
```

Το κοινό μέρος του SSID που δηλώνει την έκδοση της εφαρμογής

```
def info():
        print conf.WARNING+"@ dispel info page:"+conf.ENDC
        print conf.OKBLUE+'@ Version: dispel'+dispel_version+conf.ENDC
        print conf.OKBLUE+"@ This is free software."+conf.ENDC
        print conf.OKBLUE+'@ Tested on: Fedora & Puppy linux, Python 2.6 &
2.7'+conf.ENDC
        print conf.OKBLUE+'@ Created by: Antonios Antoniadis (Thesis for the ATEI of
Thessaloniki)'+conf.ENDC
        print conf.OKBLUE+"@ For bugs and problems found, please
contact."+conf.ENDC
        print conf.OKBLUE+'@ antonisonline (at) yahoo (dot) gr'+conf.ENDC
```

```
        print conf.OKBLUE+'@ antant (at) it (dot) theithe (dot) gr'+conf.ENDC
```

Η συγκεκριμένη μέθοδος εκτυπώνει πληροφορίες για την εφαρμογή.

```
def help():
        print conf.WARNING+"? dispel help page:"+conf.ENDC

        print conf.OKBLUE+"? Run python 'dispel<version> [OPTIONS]'"+conf.ENDC
        print conf.OKBLUE+"?        -h, --help to print the help page."+conf.ENDC
        print conf.OKBLUE+"?        -i, --info to print dispel's information."+conf.ENDC
        print conf.OKBLUE+"?        -d, --daemon to start dispel only as daemon (no
command line interface/menu)."+conf.ENDC
        print conf.OKBLUE+"? Type 'help' or 'h' to view this help info."+conf.ENDC
        print conf.OKBLUE+"? Type 'info' or 'i' to view dispel's information."+conf.ENDC
        print conf.OKBLUE+"? Type 'scan' or 's' to scan for valid dispel."+conf.ENDC
        print conf.OKBLUE+"? Type 'dispel' or 'd' to start the dispel daemon."+conf.ENDC
        print conf.OKBLUE+"? Type anything and press ENTER while the daemon is
running to stop it."+conf.ENDC
        print conf.OKBLUE+"? Type 'shell <command [arguments]>' to run shell command.
[e.g.: 'shell ls']"+conf.ENDC
        print conf.OKBLUE+"? Type 'interface <interface typenumber>' to set which wlan
interface dispel will use. [e.g: 'interface wlan0']"+conf.ENDC
```

print conf.OKBLUE+"? Type 'interval <integer_number>' to set daemon's interval. [e.g.: 'interval 6'] (This setting will effect the program only until it terminates. For permanent change set the interval value in the 'conf.py' configuration file.)"+conf.ENDC

print conf.OKBLUE+"? Set the values of the variables in the 'conf.py' configuration file to change the program's preferences. Please see the comments in the 'conf.py' file for additional help."+conf.ENDC

print conf.OKBLUE+"? Type 'quit' or 'q' or 'exit' to exit this very program."+conf.ENDC

Η παραπάνω μέθοδος δίνει όλες τις πληροφορίες για τη διαχείριση της εφαρμογής μέσω του μενού κειμένου που χρησιμοποιεί ο χρήστης καθώς και άλλες χρήσιμες πληροφορίες.

unusable_variable=os.system("clear")

Η συνάρτηση αυτή "καθαρίζει" την οθόνη από το output προηγούμενων εντολών στο τερματικό (χρήση της βιβλιοθήκης os για την εκτέλεση της εντολής)

hi_flag=False

Μία σημαία για την ένδειξη του αν ζητήθηκαν (ως ορίσματα) οι εκτυπώσεις πληροφοριών ή/και βοήθειας

daemon_flag=False

Μία σημαία για την ένδειξη του αν ο χρήστης θα έχει δυνατότητα πρόσβασης στο μενού ή η εφαρμογή θα τρέχει αμέσως με τις ρυθμίσεις όπως ορίστηκαν στο αρχείο ρυθμίσεων.

```
for arg in sys.argv:
        if arg=='--help' or arg=='-h':
                hi_flag=True
                help()
        elif arg=='--info' or arg=='-i':
                hi_flag=True
                info()
        elif arg=='--daemon' or arg=='-d':
                daemon_flag=True
if hi_flag==True and daemon_flag==False:
        exit()
```

Έλεγχος αν ζητήθηκε απλά η εκτύπωση πληροφοριών (και εμφάνιση αυτών αν ισχύει) καθώς και αν η εφαρμογή θα ξεκινήσει να τρέχει αυτόματα

```
welcome=os.popen('echo Welcome,').readline().split("\n")
user=os.popen('whoami').readline().split("\n")
```

Μήνυμα καλωσορίσματος και αποθήκευση του χρήστη που τρέχει την εφαρμογή (εντολή Linux: whoami)

```
print conf.FAIL+"""**************************************

*    """+conf.ENDC+"""dispel """+dispel_version+"""  (alpha version)"""+conf.FAIL+"""   *

**************************************"""+conf.ENDC


print conf.OKGREEN+"^",welcome[0],user[0]+'.'+conf.ENDC
```

Μήνυμα πληροφοριών για την εφαρμογή (έκδοση κλπ) και εκτύπωση του χρήστη που χρησιμοποιεί το πρόγραμμα

```
if user[0]!='root':
        print conf.FAIL+"# Please run the program as the 'root' user."+conf.ENDC
        exit()


print conf.OKBLUE+"? Use the 'help' command for useful information and type 'quit' to
exit."+conf.ENDC
```

Εδώ γίνεται έλεγχος αν ο χρήστης είναι ο χρήστης 'root' καθώς μόνο έτσι η εφαρμογή θα μπορεί να έχει δικαίωμα πρόσβασης σε όλα τις απαραίτητες εντολές και καθώς και τα αρχεία του λειτουργικού συστήματος.

```
print conf.WARNING+"^ Checking 'python' and 'iwconfig'."+conf.ENDC

pytn_check_var=os.system("which python")


if pytn_check_var!=0:
```

```
        print conf.FAIL+"! 'python' didn't found. Now exiting dispel."+conf.ENDC

        exit()


iwcnfg_check_var=os.system("which iwconfig")

if iwcnfg_check_var!=0:

        print conf.FAIL+"! 'iwconfig' didn't found. Now exiting dispel."+conf.ENDC

        exit()
```

Έλεγχος για το αν είναι εγκατεστημένη η Python και τα εργαλεία wireless-tools καθώς και τα δύο αυτά στοιχεία είναι απολύτως απαραίτητα για τη λειτουργία του προγράμματος

```
def set_dispel_id():
```

Η συνάρτηση που ορίζει το μοναδιαίο μέρος (dispel ID) του SSID

```
        cmd='ifconfig '+conf.INTERFACE

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)


        ifconfig_output=p.stdout.read()
```

Το παραπάνω κομμάτι κώδικα περιλαμβάνει συχνά χρησιμοποιούμενες εντολές/μεθόδους.

“p” είναι το αποτέλεσμα που επιστρέφεται από την Popen, όπου διαβάζεται έπειτα από την stdout.read()

Η Popen είναι η μέθοδος που επικοινωνεί με το λειτουργικό σύστημα. Τα ορίσματά της είναι η εντολή που θα εκτελεστεί, ο φλοιός, το που θα διασωληνωθούν τα stdin και stdout κλπ.

Η συγκεκριμένη μέθοδος είναι ευρέως χρησιμοποιούμενη από τους απανταχού προγραμματιστές της Python και τείνει να εξαλείψει τη χρήση της παλαιότερης os.system()

```
if conf.DISPEL_ID=="MAC":
```

Έλεγχος εάν στο αρχείο ρυθμίσεων έχει ορισθεί το dispel ID να ορίζεται αυτόματα από μέρος της διεύθυνσης MAC

```
try:
```

Χρήση εξαιρέσεων καθώς υπάρχει περίπτωση να μη βρεθεί κάποιο interface

```
mac=re.findall('[H][W][a][d][d][r][ ](.*)',ifconfig_output)
```

Φιλτράρισμα με τη βιβλιοθήκη re της Python

```
mac=[x.strip(' ') for x in mac]
```

Απαλοιφή των κενών

```
mac_part=re.findall('.*[:](.*[:].*[:].*[:].*)',mac[0])[0]
```

Εύρεση του κατάλληλου μέρους της διεύθυνσης

```
dispel_id="dispel"+dispel_version+"%__"+mac_part
```

Ορισμός του dispel ID με τη χρήση της έκδοσης και του μέρους της MAC στο SSID του μηχανήματος

```
return dispel_id
```

Επιστροφή της τιμής

```
except Exception:
print conf.FAIL+"^ Cannot retrieve the MAC address for the '"+conf.INTERFACE+"' interface."+conf.ENDC

else:
try:
dispel_id="dispel"+dispel_version+"%__"+conf.DISPEL_ID
```

Εάν ο χρήστης έχει ορίσει δικό του dispel ID θα χρησιμοποιηθεί αυτό και δε αντληθεί τίποτα δυναμικά

```
except NameError:
print conf.FAIL+"^ Cannot set the dispel_id settled in the 'conf.py' file. Now setting the dispel_id with information from the MAC address."+conf.ENDC
```

```
    mac=re.findall('[H][W][a][d][d][r][ ](.*)',ifconfig_output)


    mac=[x.strip(' ') for x in mac]

    mac_part=re.findall('.*[:](.*[:].*[:].*[:].*)',mac[0])[0]

    dispel_id="dispel"+dispel_version+"%__"+mac_part

    return dispel_id
```

Το παραπάνω κομμάτι κώδικα είναι παρόμοιο με αυτό της προηγούμενης σελίδας.

Αν ο χρήστης δεν έχει ορίσει καθόλου το dispel ID ή ούτε τον τρόπο δημιουργίας του, τότε η εφαρμογή αυτόματα, για αποφυγή κατάρρευσης, χρησιμοποιεί την ίδια τεχνική όπως πριν για να αντλήσει το μοναδιαίο μέρος από τον υπολογιστή.

```
def bring_interface_up():

        cmd='ifconfig '+conf.INTERFACE+' up'

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Συνάρτηση για ενεργοποίηση του ασύρματου interface

```
def bring_lo_down():

        cmd='ifconfig lo down'

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Συνάρτηση για την απενεργοποίηση του ασύρματου interface

Αυτό είναι σημαντικό γιατί ο υπολογιστής μπερδεύεται όταν το interface είναι σε χρήση και πολλές φορές υπάρχει ο κίνδυνος να στείλει δεδομένα στον εαυτό του

```
def bring_lo_up():

        cmd='ifconfig lo up'

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Συνάρτηση για την ενεργοποίηση του loopback interface

```
def get_valid_dispel_ssids():
```

Η συνάρτηση βρίσκει SSIDs που είναι ορισμένα από το dispel σε άλλους υπολογιστές

```
        cmd='iwlist '+conf.INTERFACE+' scan'

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Χρήση της iwlist για scanning χρησιμοποιώντας το interface που έχει οριστεί στο αρχείο ρυθμίσεων

```
iwlist_scanning_output=p.stdout.read()
```

Και αποθήκευση των αποτελεσμάτων

```
try:
        essids_list=re.findall('[E][S][S][I][D][:]["](.*)["]',iwlist_scanning_output)
```

Φιλτράρισμα για να κρατήσουμε μόνο τα SSIDs και όχι τις υπόλοιπες πληροφορίες

```
        valid_dispel_ssids=[]
        for dispel_ssid in essids_list:

                if dispel_ssid.startswith(dispel_version_id)==True:
                        if valid_dispel_ssids.count(dispel_ssid) < 1:
                                valid_dispel_ssids.append(dispel_ssid)
        if dispel_id in valid_dispel_ssids:
                valid_dispel_ssids.remove(dispel_id)
```

Το παραπάνω τμήμα κώδικα ελέγχει εάν από τα SSIDs που βρέθηκαν στους γειτονικούς υπολογιστές υπάρχουν κάποια ορισμένα από το dispel, δηλαδή ποιοι υπολογιστές τρέχουν την εφαρμογή

```
except RuntimeError:
        valid_dispel_ssids=[]
```

```
        return valid_dispel_ssids

def set_server_ip():

        cmd='ifconfig '+conf.INTERFACE+' '+conf.SERVER_IP+' netmask
'+conf.SUBNET_MASK

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Συνάρτηση για τον ορισμό της IP διεύθυνσης της κατάστασης διακομιστή

```
def set_client_ip():

        cmd='ifconfig '+conf.INTERFACE+' '+conf.CLIENT_IP+' netmask
'+conf.SUBNET_MASK

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Συνάρτηση για τον ορισμό της IP διεύθυνσης της κατάστασης πελάτη

```
def set_ssid(ssid):
```

Η συνάρτηση αυτή ορίζει το SSID

```
        cmd='ifconfig '+conf.INTERFACE+' down'

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Απενεργοποίηση του interface πριν τον ορισμό των αλλαγών (είναι απαραίτητο)

```
        cmd='iwconfig '+conf.INTERFACE+' mode Ad-Hoc channel '+str(conf.CHANNEL)+'
essid '+ssid
```

Ορισμός του δικτύου ως Ad-Hoc με τη βοήθεια της εντολής iwconfig του πακέτου wireless-tools

```
        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)

        cmd='ifconfig '+conf.INTERFACE+' up'

        p = Popen(cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT,
close_fds=True)
```

Στη συνέχεια ενεργοποιούμε το interface

```
        return


def startup_dispel():
```

Η συνάρτηση που τρέχει πριν την ενεργοποίηση του loop. Καλεί άλλες συναρτήσεις για την αρχικοποίηση των ρυθμίσεων της εφαρμογής πριν αρχίσει η λειτουργία της.

```
        bring_interface_up()

        dispel_id=set_dispel_id()
```

Αποθηκεύουμε το SSID αφού έχουμε ενεργοποιήσει το interface

```
print conf.WARNING+"^ Please wait a few seconds while setting the dispel ID
('"+dispel_id+"')."+conf.ENDC
```

Αυτό είναι ένα μήνυμα που δηλώνει ότι χρειάζεται κάποιος χρόνος για τον ορισμό του SSID και για να καταλάβει ο χρήστης ότι το πρόγραμμα συνεχίζει να τρέχει

```
set_ssid(dispel_id)
```

Ορισμός του SSID

```
set_server_ip()
```

Από προεπιλογή ο κάθε υπολογιστής είναι σε κατάσταση server όπου αναμένει συνδέσεις και έπειτα τα δεδομένα

```
time.sleep(8)
```

Μία περίοδος λίγων δευτερολέπτων για να σιγουρέψουμε ότι ο ορισμός του SSID έχει πραγματοποιηθεί.

```
bring_lo_down()
```

Απενεργοποίηση του loopback interface

```
return dispel_id
```

Επιστροφή της τιμής του dispel ID

```
def exit_dispel():
```

Με αυτή τη συνάρτηση το πρόγραμμα κλείνει επαναφέροντας τις ρυθμίσεις σε μία κατάσταση όπου το πρόγραμμα βρισκόταν κατά την εκκίνηση

```
        print conf.WARNING+"^ Restoring dispel settings."+conf.ENDC

        bring_interface_up()

        set_ssid(dispel_id)

        set_server_ip()

        bring_lo_up()

        print conf.FAIL+"! Exiting dispel."+conf.ENDC

        exit()
```

Οι παραπάνω συναρτήσεις έχουν ήδη εξηγηθεί. Με την exit() τερματίζεται το πρόγραμμα (συνάρτηση της Python)

```
def dispel():
```

Αυτή η μέθοδος είναι το βασικό μέρος του προγράμματος

```
print(conf.WARNING+"> Daemon started."+conf.ENDC)
```

Μήνυμα για την έναρξη του loop

```
set_server_ip()
```

Ενεργοποίηση της κατάστασης διακομιστή

```
server_sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
client_sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Εδώ δημιουργούνται δύο sockets. Ένα για την κατάσταση client και ένα για την κατάσταση server. Τα ορίσματα δηλώνουν ότι τα sockets θα είναι τύπου Internet TCP/IP.

```
input=[server_sock,sys.stdin]
```

Λίστα όπου δηλώνει ότι ο υπολογιστής σε κατάσταση server μπορεί να δέχεται επικοινωνία με το socket του server ή είσοδο από το πληκτρολόγιο για το μενού που χρησιμοποιεί ο χρήστης.

```
output=[client_sock]
```

Δήλωση πως ο υπολογιστής σε κατάσταση client δέχεται επικοινωνία με το socket του client

```
running = 1
```

```
while running == 1:
```

Το loop που θα τρέχει συνεχόμενα

```
valid_dispel_ssids=get_valid_dispel_ssids()
```

Εύρεση: ποια SSIDs είναι συμβατά με το dispel, δηλαδή η υπολογιστές τρέχουν την εφαρμογή.

```
if valid_dispel_ssids:

print conf.WARNING+"^ Valid dispel"+dispel_version+" SSID(s)
found:"+conf.ENDC

for valid_ssid in valid_dispel_ssids:

print conf.FAIL+"["+valid_ssid+"]"+conf.ENDC,

print ""
```

Εκτύπωση των συμβατών SSIDs που βρέθηκαν

```
for dispel_machine in valid_dispel_ssids:

set_ssid(dispel_machine)
```

Για κάθε υπολογιστή που βρέθηκε που τρέχει την εφαρμογή θα γίνουν τα παρακάτω:

```
inputready,outputready,exceptready =
select.select(input,output,[])
```

Η συνάρτηση select είναι ένας τρόπος διαχείρισης δεδομένων στα sockets. Επιστρέφει τιμές σε μεταβλητές για τα δεδομένα που είναι έτοιμα να αποσταλούν ή να παραληφθούν και ελέγχει στο input αν έχει δοθεί εντολή από το πληκτρολόγιο

```
for i in inputready:
if i == server_sock:
```

Αν πρόκειται για server socket τότε

```
server_sock=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
```

Δημιούργησε μια νέα σύνδεση για την επικοινωνία

```
SERVER_BIND=(conf.SERVER_IP,conf.PORT)
server_sock.bind(SERVER_BIND)
```

Σύνδεση του socket με τη διεύθυνση IP και τη θύρα του άλλου μηχανήματος

```
server_sock.listen(conf.BACKLOG)
```

Συνάρτηση που αναμένει συνδέσεις

```
client, address = server_sock.accept()
```

Συνάρτηση που δέχεται συνδέσεις από απομακρυσμένους υπολογιστές

```
input.append(client)
```

Προσθήκη του client στη λίστα με τους υπόλοιπους

```
elif i == sys.stdin:
```

Σε αυτό το σημείο, γίνεται ο εξής έλεγχος: Αν ο χρήστης έχει πληκτρολογήσει κάτι δίνεται εντολή για το σταμάτημα της λειτουργίας του προγράμματος

```
junk = sys.stdin.readline()
running = 0
server_sock.close()
```

Και κλείνει το socket

```
print(conf.WARNING+"< Daemon
stopped.\n"+conf.ENDC),
```

```
else:
```

```
                    data = i.recv(conf.BUF_SIZE)
```

Αν στο input αντί για νέα σύνδεση ή δεδομένα από το πληκτρολόγιο, υπάρχουν δεδομένα που πρόκειται να παραληφθούν, γίνεται η αποθήκευσή τους στη μεταβλητή data

```
                    if data:

                        print(conf.OKBLUE+"<-Receiving data
from "+dispel_machine+":"+str(i.getpeername())+":\n"+conf.ENDC+data),
```

Αν υπάρχουν δεδομένα εκτυπώνονται και εμφανίζονται πληροφορίες για τον υπολογιστή που τα έστειλε

```
                    else:

                    i.close()

                    input.remove(i)
```

Αν δεν υπάρχει τίποτα προς παραλαβή, το socket κλείνει

```
            for o in outputready:
```

Εδώ ο υπολογιστής περνάει σε κατάσταση client και τώρα θα επιχειρήσει την αποστολή των δεδομένων

```
            set_client_ip()
```

Ορισμός IP διεύθυνσης client

```
client_sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Δημιουργία νέου socket για επικοινωνία όπως προηγουμένως

```
SERVER_ADDRESS=(conf.SERVER_IP,
conf.PORT)
client_sock.connect((SERVER_ADDRESS))
```

Επιχείρηση σύνδεσης στον απομακρυσμένο server

```
if conf.SENDING_DATA=="TEST":
```

Αν στο αρχείο ρυθμίσεων έχει οριστεί SENDING_DATA=="TEST" τότε θα γίνει αποστολή δοκιμαστικού μηνύματος μόνο

```
        cmd='echo Sample Message.'
    else:
        cmd=conf.SENDING_DATA
    p = Popen(cmd, shell=True, stdin=PIPE,
stdout=PIPE, stderr=STDOUT, close_fds=True)
    output = p.stdout.read()
```

Αλλιώς θα αποσταλούν τα δεδομένα από την εντολή ή το αρχείο που έχει ορισθεί στο αρχείο ρυθμίσεων

```
print(conf.OKGREEN+"->Sending data to
"+dispel_machine+":"+str(i.getsockname())+".")
```

```
o.send(poutput)
```

Αποστολή των δεδομένων και εκτύπωση μηνύματος πληροφοριών

```
set_server_ip()
```

```
set_ssid(dispel_id)
```

Επαναφορά σε κατάσταση client

```
else:
    print conf.FAIL+"! No valid dispel"+dispel_version+" ssids found.\n"+conf.ENDC,
    bring_interface_up()
    set_ssid(dispel_id)
```

Αν δε βρέθηκαν κατάλληλα SSIDs δε πραγματοποιείται αποστολή

```
time.sleep(conf.INTERVAL)
```

Αυτό είναι το χρονικό διάστημα όπου ο υπολογιστής θα περιμένει μέχρι να ξαναστείλει τα δεδομένα. Αυτό είναι σημαντικό καθώς χωρίς αυτό οι υπολογιστές θα αντάλλαζαν δεδομένα αστραπιαία χωρίς ο χρήστης να μπορούσε να τα διαβάσει

```
dispel_id=startup_dispel()
```

Καλείται η μέθοδος αρχικοποίησης των ρυθμίσεων

```
if daemon_flag==True:
        print conf.OKBLUE+"? Type anything and press ENTER while the daemon is
running to stop it."+conf.ENDC
        dispel()
        exit_dispel()
```

Αν έχει δοθεί το όρισμα -d τότε δε θα εμφανιστεί το μενού, το loop θα ξεκινήσει άμεσα και η εφαρμογή θα τερματιστεί όταν ο χρήστης πληκτρολογήσει κάτι

```
loop_flag=1
shell_string="
while 1:

        sys.stdout.write(conf.FAIL+"*>"+conf.ENDC)
```

Αυτό είναι το loop για το μενού. Ο χρήστης θα εισάγει εντολές και αυτές αφού ελεγχθούν θα εκτελεστούν παρακάτω

```
        user_selection=raw_input()
```

Συνάρτηση της Python για την αποθήκευση των πληκτρολογημένων δεδομένων από τον χρήστη

```
if user_selection=='quit' or user_selection=='q' or user_selection=='exit':


        exit_dispel()
```

Έλεγχος εάν ο χρήστης δήλωσε ότι θέλει να τερματίσει την εφαρμογή

```
elif user_selection=='help' or user_selection=='h':


        help()
```

Έλεγχος εάν ο χρήστης δήλωσε ότι θέλει να εκτυπώσει τη σελίδα της βοήθειας και αν ναι, εκτύπωσή της

```
elif user_selection=='info' or user_selection=='i':


        info()
```

Έλεγχος εάν ο χρήστης δήλωσε ότι θέλει να εκτυπώσει τη σελίδα πληροφοριών και αν ναι, εκτύπωσή της

```
elif user_selection=='scan' or user_selection=='s':
```

Αν ο χρήστης έδωσε την εντολή για σκανάρισμα των γειτονικών SSIDs γίνονται τα παρακάτω

```
                valid_ssids_scanned=get_valid_dispel_ssids()
                if valid_ssids_scanned:
                        print conf.WARNING+"^ Valid dispel"+dispel_version+" SSID(s)
found:"+conf.ENDC
                        for valid_ssid in valid_ssids_scanned:
                                print conf.FAIL+"["+valid_ssid+"]"+conf.ENDC
                else:
                        print conf.FAIL+"! No valid dispel"+dispel_version+" ssids
found."+conf.ENDC
```

Αναζητούνται και εκτυπώνονται με τις αντίστοιχες μεθόδους που αναλύθηκαν προηγουμένως τα SSIDs, εκτός αν δε βρεθεί κανένα κατάλληλο SSID

```
        elif user_selection=='dispel' or user_selection=='d':
                dispel()
```

Αν ο χρήστης πληκτρολογήσει 'dispel' ή 'd' η εφαρμογή ξεκινάει να λειτουργεί!

```
        else:
```

Μία ακόμη δυνατότητα από το μενού είναι ο χρήστης να μπορεί να τρέχει εντολές του φλοιού μέσα από την εφαρμογή.

```
                user_selection_splitted=user_selection.split()
```

```
if len(user_selection_splitted)==0:
    continue
```

Αν πατηθεί απλά το Enter δε γίνεται τίποτα

```
if user_selection_splitted[0]=='shell':
```

Αν ο χρήστης πληκτρολογήσει το παραπάνω θα εμφανιστούν πληροφορίες για το τρέχον shell του Linux

```
    if len(user_selection_splitted)==1:
        os.system('echo "$SHELL"')
        print conf.FAIL+"! Please enter a command to be executed by the shell. Syntax: 'shell <command [arguments]>'."+conf.ENDC
    user_selection_splitted.remove('shell')
    for item in user_selection_splitted:
        shell_string=shell_string+item+' '
    if len(user_selection_splitted)!=0:
        print conf.WARNING+"$ Executing: '"+conf.HEADER+shell_string+conf.WARNING+"'"+conf.ENDC
        os.system(shell_string)
        shell_string=''
```

Η λειτουργία του παραπάνω κομματιού κώδικα:

Αν ο χρήστης πληκτρολογήσει shell και έπειτα μια εντολή Linux, αφού γίνει έλεγχος λάθους, η εντολή θα εκτελεστεί κανονικά σαν ο χρήστης να ήταν μπροστά σε ένα τερματικό.

Αυτό είναι πολύ σημαντικό καθώς έτσι ο χρήστης μπορεί να έχει πρόσβαση στο λειτουργικό σύστημα άμεσα μέσα από το μενού της εφαρμογής

```
elif user_selection_splitted[0]=='interval':
    if len(user_selection_splitted)==1:
        print conf.WARNING+"^ Interval (in secs): '"+str(conf.interval)+"'"+conf.ENDC
        print conf.FAIL+"! Please enter an integer number for the interval."+conf.ENDC
    elif len(user_selection_splitted)==2:
        user_selection_splitted.remove('interval')
```

Αν ο χρήστης έχει πληκτρολογήσει την εντολή interval τότε γίνονται τα παρακάτω:

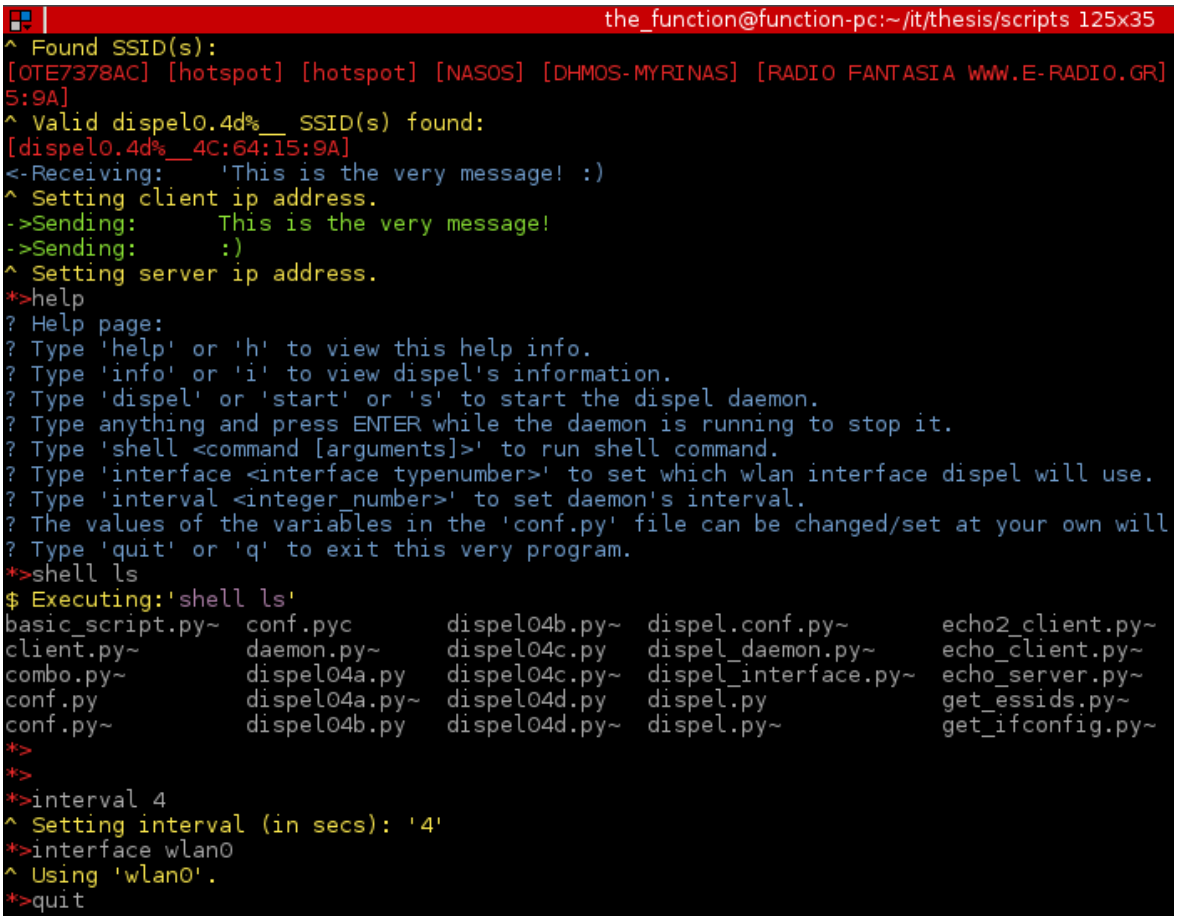
```
        try:
            conf.interval=int(user_selection_splitted[0])
            print conf.WARNING+"^ Setting interval (in secs): '"+str(conf.interval)+"'"+conf.ENDC
        except ValueError:
            print conf.FAIL+"! Please enter an integer number for the interval."+conf.ENDC
```

```
else:
        print conf.FAIL+"! Please enter an integer number for the
interval."+conf.ENDC
```

Μετά από έλεγχο για την ύπαρξη λάθους, ορίζεται ως interval ο ακέραιος αριθμός που έχει πληκτρολογήσει ο χρήστης

```
else:
    print conf.FAIL+"! Please enter a valid command."+conf.ENDC
```

Τέλος, αν στο μενού ο χρήστης έχει πληκτρολογήσει κάτι που δεν αντιστοιχεί σε κάποια εντολή του προγράμματος εκτυπώνεται ένα μήνυμα λάθους και ο χρήστης συνεχίζει να βρίσκεται στο μενού.



Εικόνα "Εκτύπωση πληροφοριών, ορισμός καθυστέρησης στην αποστολή δεδομένων και του interface που θα χρησιμοποιηθεί (παλαιότερη έκδοση), εκτέλεση εντολής Linux μέσα από την εφαρμογή και έξοδος"

ΥΠΟΚΕΦΑΛΑΙΟ 6.3

GPLv3

Η εφαρμογή της πτυχιακής εργασίας διανέμεται κάτω από τη GPLv3.

Η Γενική Άδεια Δημόσιας Χρήσης GNU (GPL) είναι μία άδεια χρήσης ελεύθερου λογισμικού. Η πιο πρόσφατη έκδοση της άδειας, η έκδοση 3, κυκλοφόρησε στις 29 Ιουνίου 2007.

Η άδεια αυτή δίνει στους κατόχους ενός προγράμματος τα ακόλουθα τέσσερα δικαιώματα:

- να τρέξουν ένα πρόγραμμα για οποιοδήποτε λόγο
- να μελετήσουν τη λειτουργία ενός προγράμματος και να το τροποποιήσουν
- να διανείμουν αντίγραφα του προγράμματος ώστε να βοηθήσουν άλλους χρήστες
- να βελτιώσουν το πρόγραμμα ώστε να ωφεληθεί ολόκληρη η κοινότητα χρηστών

Προϋποθέσεις για τα παραπάνω είναι ο ανοιχτός κώδικας, δηλαδή ο κώδικας του προγράμματος να είναι γνωστός και προσβάσιμος στον χρήστη.

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο έγινε ανάλυση και επεξήγηση του κώδικα της εφαρμογής με τη βοήθεια της οποίας αναδείχθηκαν συμπεράσματα για την αξιολόγηση της αρχικής ιδέας και την υλοποίησή της.

Ο κώδικας αποτελεί ένα σύνολο εντολών και μεθόδων του GNU/Linux, της γλώσσας προγραμματισμού Python καθώς και άλλων εργαλείων. Ο κώδικας που δημιουργήθηκε βάσει αυτών συντελεί το πρόγραμμα που μέσω της Python συντονίζει όλες τις λειτουργίες.

Όπως φάνηκε, οι δικτυακές υποδοχές (sockets) παίζουν πολύ σημαντικό ρόλο στη λειτουργία του προγράμματος. Επίσης είναι φανερό πως το SSID του κάθε μηχανήματος είναι τρόπος μονοσήμαντου ορισμού του, έτσι όπως προτάθηκε κατά την ανάληψη της εργασίας.

Στο επόμενο κεφάλαιο αναφέρονται τα συμπεράσματα που εξήχθησαν έπειτα από δοκιμές σε διάφορες χρονικές περιόδους καθώς και εκδόσεις του λογισμικού.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Μετά από δοκιμές που έγιναν με περισσότερους από δύο υπολογιστές που υποστηρίζουν ασύρματη δικτύωση βγήκαν τα συμπεράσματα για την εφαρμογή καθώς και παρατηρήθηκαν στοιχεία για τη λειτουργία της.

Παρατηρήσεις:

Α) Η εφαρμογή έχει ολοκληρωθεί και λειτουργεί χωρίς λάθη ως προς την ορθότητα του κώδικά της στον εκάστοτε υπολογιστή που τρέχει. Αυτό βέβαια δε σημαίνει πως δε μπορεί να εξελιχθεί περαιτέρω, αντιθέτως, λόγω του πρωτότυπης ιδέας, η εφαρμογή αποτελεί ένα έργο συνεχούς βελτιστοποίησης.

Β) Η εφαρμογή τρέχει σε ικανοποιητικές ταχύτητες, δηλαδή η καθυστέρηση και η ταχύτητα απόκρισης του προγράμματος είναι πολύ μικρή και ανεπαίσθητη από τον χρήστη. Αξίζει να σημειωθεί πως η γλώσσα Python θεωρείται γενικά μία αργή γλώσσα προγραμματισμού σε σχέση με μία όπου ο κώδικάς της μεταγλωττίζεται.

Γ) Ο σχεδιασμός του έργου ορίζει πως το SSID παίζει σημαντικό ρόλο, προσδιορίζοντας μονοσήμαντα την κάθε συσκευή. Αυτό βέβαια, αποτελεί μεν ένα πλεονέκτημα στην ευκολία κατανόησης της εφαρμογής και υλοποίησης του κώδικά της, αλλά είναι δε ένα από τα πρώτα μειονεκτήματα που παρατηρήθηκαν. Αυτό συμβαίνει επειδή η αλλαγή του SSID με χρήση εντολών του λειτουργικού συστήματος διαχειριζόμενες μέσω της Python καθυστερεί. Για τη σωστή αλλαγή του SSID το ασύρματο interface πρέπει να απενεργοποιηθεί και να ενεργοποιηθεί ξανά, κάτι που για τα δεδομένα του χρόνου εκτέλεσης του υπόλοιπου μέρους του κώδικα, προκαλεί καθυστέρηση. Η καθυστέρηση αυτή δεν προκαλεί πρόβλημα στην εκτέλεση, αλλά ο κώδικας έπρεπε να προσαρμοστεί ώστε ο χρήστης να μη περιμένει χωρίς την εκτύπωση κάποιου μηνύματος ή εμφάνισης δεδομένων για αυτό το χρονικό διάστημα.

Δ) Στα στάδια υλοποίησης το πρόγραμμα έτρεχε και οι πληροφορίες που έπρεπε να ανταλλαχθούν φαινόταν ότι ανταλλάσσονταν σωστά (περίπου στην έκδοση 04.x). Με μια πιο προσεκτική ανάλυση όμως αποδείχθηκε πως ο κάθε υπολογιστής δεν επικοινωνούσε με τους άλλους, απλά μπερδευόταν και έστελνε τα

δεδομένα στον εαυτό του. Αυτό είναι η πρώτη απόδειξη ότι η προσέγγιση του σχεδίου όπου ο υπολογιστής είναι ανά πάσα στιγμή μόνο πελάτης ή μόνο διακομιστής είναι ένα πιθανό αδιέξοδο. Ο κώδικας έπειτα αναπτύχθηκε ώστε στις πληροφορίες που ανταλλάσσονται να είναι ξεκάθαρο ποιος είναι ο αποστολέας και ποιος ο δέκτης. Στις τελευταίες εκδόσεις του προγράμματος (06.g) ο κώδικας έπρεπε να αλλαχθεί για τη διόρθωση απαραίτητων λειτουργιών, κάτι που αποτέλεσε η εφαρμογή να μη μπορεί να ανταλλάσσει δεδομένα. Αυτό συμβαίνει επειδή τα sockets που πρέπει να δημιουργούνται πρέπει να είναι μοναδιαία και να κλείνουν μετά από κάθε μεταφορά δεδομένων, κάτι που δε μπορεί να υλοποιηθεί αν ο η κάθε οντότητα (υπολογιστής) δε μπορεί να είναι ταυτόχρονα πελάτης και διακομιστής – τουλάχιστον δεν είναι εφικτό σύμφωνα με το πως αρχικά σχεδιάστηκε η εφαρμογή, με το ποια ήταν η βασική, αρχική ιδέα.

Συμπεράσματα:

1) Η χρήση του SSID ως τρόπος μοναδιαίας ταυτοποίησης υπολογιστών μπορεί να υλοποιηθεί, αλλά είναι αργός και δημιουργεί πρόβλημα στην αναγνώριση του κάθε υπολογιστή ως παραλήπτη ή δέκτη, καθώς τη στιγμή της ανταλλαγής πληροφοριών μεταξύ δύο υπολογιστών το SSID τους είναι το ίδιο.

2) Για την υλοποίηση του προγράμματος τα sockets πρέπει να αρχικοποιούνται και να τερματίζονται κάθε φορά μετά από μία ανταλλαγή πληροφοριών και όχι να χρησιμοποιούνται ξανά σε ένα loop. Αυτό είναι κάτι πολύ βασικό καθώς το πως η συναρτήσεις των sockets είναι υλοποιημένες (στις βιβλιοθήκες της Python αλλά το ίδιο θα συνέβαινε π.χ. και αν χρησιμοποιούνταν η C++) δυσκολεύει μία δυναμική υλοποίηση του έργου της πτυχιακής εργασίας καθώς η δυναμικότητά της έπρεπε να σχεδιαστεί και να υλοποιηθεί, όχι απλά να χρησιμοποιηθεί ως έτοιμο πρόγραμμα ή εργαλείο.

3) Μία υλοποίηση με ένα εργαλείο/βιβλιοθήκη όπως το “Twisted Python” θα έλυνε το πρόβλημα αυτό αλλά θα ήταν μία έτοιμη λύση όπου η σε βάθος δημιουργία του προγράμματος δε θα ήταν εφικτή, όπως εφικτή δε θα ήταν και η πραγματοποίηση της έρευνας όπου και έγινε καθ’ όλη τη διάρκεια της εργασίας.

ΜΕΛΛΟΝΤΙΚΕΣ ΚΑΤΕΥΘΥΝΣΕΙΣ ΕΞΕΛΙΞΗΣ

Μία διαφορετική προσέγγιση στο σχεδιασμό θα μπορούσε να φέρει απτά αποτελέσματα. Για παράδειγμα η χρήση έτοιμων εργαλείων, όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, θα μπορούσε - πιθανώς - να δημιουργήσει μία εφαρμογή όπου θα λειτουργούσε αποτελεσματικά. Μία τέτοια προσέγγιση βέβαια θα απαγόρευε την πραγματοποίηση έρευνας και υλοποίησης χαμηλού επιπέδου. Παρ' όλα αυτά, μια τέτοια υλοποίηση μπορεί να φέρει νέου είδους συμπεράσματα.

Η επιπλέον ανάπτυξη της εφαρμογής με τη χρήση των παρακάτω στοιχείων θα επιτρέψει τη μελλοντική βελτίωση και ακεραιότητα στη λειτουργία της:

- Χρήση sockets με κρυπτογράφηση (υπάρχει αντίστοιχη βιβλιοθήκη στην Python αλλά μπορεί να υλοποιηθεί και ξεχωριστά πιο αναλυτικά αλλά πιθανώς με επίπτωση στο χρόνο εκτέλεσης)
- Η εφαρμογή να τρέχει ως Linux daemon με χειρισμό μέσω π.χ. του init.d
- Η εφαρμογή να μπορεί να λειτουργήσει στατικά (π.χ. μόνο μία φορά) ούτως ώστε να μπορεί να είναι εργαλείο άντλησης πληροφοριών από απομακρυσμένα μηχανήματα – φυσικά με την κατάλληλη τροποποίηση και προσθήκη κώδικα.
- Η άντληση πληροφοριών για την διεύθυνση MAC πρέπει να διαφοροποιηθεί ώστε να είναι συμβατή σε περισσότερες διανομές του GNU/Linux (π.χ. ανά διανομή διαφέρει η ονοματοδοσία για τα δικτυακά interfaces)
- Αν η εφαρμογή τρέχει σε ικανοποιητικές ταχύτητες, θα αποτελέσει ένα framework για ανάπτυξη νέων εφαρμογών βασισμένη σε αυτή (π.χ. Chat) όπως ακριβώς θεωρήθηκε στην περιγραφή της εργασίας κατά την ανάθεσή της.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Α. Αλεξόπουλος και Γ. Λαγογιάννης (2010) Τηλεπικοινωνίες και Δίκτυα Υπολογιστών 7η έκδοση
- Douglas E. Comer (2003) Internetworking with TCP/IP (Διαδίκτυα με TCP/IP) 4η έκδοση
- Swaroop C H (2003) Byte of Python Version 1.20
- John Goerzen (2010) Apress Foundations of Python 3 Network Programming 2nd Edition
- David Ascher, Alex Martelli & Anna Ravenscroft (2005) O'Reilly Python Cookbook 2nd Edition
- Noah Gift & Jeremy M. Jones (2008) O'Reilly Python for Unix and Linux System Administration
- Machtelt Garrels (2002) Introduction to Linux (Πρώτη δημοσίευση Ελληνικής μετάφρασης Οκτώβριος 2007)