ΤΕΙ Κρήτης

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Με θέμα :

Πρόγραμμα Java/Android πραγματοποιημένο με socket programming με δυνατότητες αποστολής/λήψης αρχείων και φακέλων και δυνατότητα λήψης Snapshot αυτόματα για έλεγχο χώρου.

Μπαρνιαδάκη Παυλίνα-Αικατερίνη , Α.Μ. 4239

Επιβλέπων καθηγητής : Αντώνιος Κωνσταντάρας

Περιεχόμενα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω καταρχάς όλους όσους συνέβαλαν στην πραγματοποίηση της πτυχιακής εργασίας και στην δοκιμή του προγράμματος. Ευχαριστώ πάνω από όλα θερμά τον καθηγητή κ. Αντώνιο Κωνσταντάρα για την βοήθεια και την εμπιστοσύνη που μου έδειξε, για τις υποδείξεις που μου έκανε και την πολύτιμη καθοδήγηση του. Ευχαριστώ την γραμματεία της σχολής, κ. Ζυμβραγού Ελένη και κ. Γκατζούνη Καλλιόπη που ήταν πάντα διαθέσιμες για την ενημέρωση οποιουδήποτε θέματος και την (οικογενειακή) εμψύχωση που μου παρείχαν. Ευχαριστώ τους συμφοιτητές μου, Χριστοφυλλάκη Κωνσταντίνο και Νίκη Κωνσταντίνα γιατί χάρης την δική τους στήριξη αποφάσισα να παρακολουθήσω την σχολή. Ευχαριστώ επίσης την Μπελεχάκη Χρύσα για την βοήθεια της και όλους τους συμφοιτητές που συμπορευτήκαμε, τόσο από το τμήμα Ηλ. Μηχανικών όσο και από το τμήμα Φυσικών Πόρων. Ένα μεγάλο ευχαριστώ οφείλω στον αδερφό μου, για όσα έκανε για μένα, και στην οικογένεια μου που επέμενε και στήριξε τις σπουδές μου με όλους τους δυνατούς τρόπους. Η πτυχιακή μου είναι αφιερωμένη στην μητέρα μου.

ΕΙΣΑΓΩΓΗ

Συντομη Περιγραφή Προγράμματος

1 Περιγραφή συστήματος

Η εφαρμογή για υπολογιστές αλλά και για κινητές συσκευές απευθύνεται στους φοιτητές του **Τμήματος Ηλεκτρονικής** του **ΤΕΙ Κρήτης** και στο ευρύτερο δίκτυο και έχει ως κύριο στόχο την ανταλλαγή αρχείων και μηνυμάτων.

Το πρόγραμμα παρέχει τη δυνατότητα ανταλλαγής στιγμιότυπων κάμερας webcam (εάν υπάρχει).

2 Πλαίσιο χρήσης

Το σύστημα αναμένεται να χρησιμοποιείται για μικρά χρονικά διαστήματα (έως και ελάχιστα δευτερόλεπτα σε ορισμένες περιπτώσεις) ανάλογα την εκάστοτε ανάγκη που
εξυπηρετεί.

Δεν αναμένεται ο Client (Πελάτης) να χρησιμοποιεί το σύστημα τακτικά και για μεγάλα διαστήματα.

Επιπλέον, οι Client (Πελάτες) αναμένεται να χρησιμοποιούν το σύστημα σε θορυβώδη περιβάλλοντα (π.χ., αίθουσες διαλέξεων, κυλικεία, κλπ.) και πολλές φορές εν κινήσει.

Η εφαρμογή πρέπει να σχεδιαστεί έτσι ώστε να εξυπηρετεί τις ανάγκες του χρήστη άμεσα και εύκολα.

Περιβάλλον χρήσης:

Δεν προκύπτει κάποια ιδιαίτερη απαίτηση για το σύστημα από το περιβάλλον χρήσης. Οι Client (Πελάτες) θα έχουν πρόσβαση στο σύστημα μέσω **desktop PC** (είτε από το σπίτι, είτε από κάποιο εργαστήριο υπολογιστών του Ιδρύματος) ή μέσω **laptop** (από σπίτι ή σε κάποιο χώρο με πρόσβαση στο διαδίκτυο) ή μέσω μίας **κινητής συσκευής** που υποστηρίζει **Android**.

Για τη χρήση του desktop PC/Laptop απαιτείται να είναι εγκατεστημένη η τελευταία έκδοση της πλατφόρμας **Java** (Version 8 Update 45+)

Λειτουργικό σύστημα:

Το σύστημα θα «τρέχει» σωστά και αποδοτικά σε Android, Windows, MAC OS X και Linux με την προϋπόθεση ότι έχει εγκατασταθεί η τελευταία έκδοση της Java. Δεν απαιτείται επιπλέον ειδικό λογισμικό ή υλικό από τη μεριά του χρήστη για να λειτουργήσει σωστά το πρόγραμμα.

3 Λειτουργίες του Συστήματος και Λειτουργικές Απαιτήσεις

Για την υλοποίηση θα χρησιμοποιήσουμε μια τροποποιημένη έκδοση του πρωτοκόλλου FTP (RFC 959) το οποίο χρησιμοποιείται για την μεταφορά αρχείων από ένα FTP Server (εξυπηρέτης) σε έναν FTP Client (πελάτης). Το πρωτόκολλο FTP χρησιμοποιεί TCP συνδέσεις.

- Μια σύνδεση για την ανταλλαγή μηνυμάτων ελέγχου (control messages),
- Μια σύνδεση για την μεταφορά δεδομένων (data connection).

Συγκεκριμένα:

3.1 Εκκίνηση

- Η εκκίνηση θα γίνεται αρχικά από τον Server (εξυπηρέτη).

  Έπειτα, δίνεται η δυνατότητα επιλογής για τον Client (πελάτη) συσκευής (κινητό με Android ή υπολογιστής).

- Ο Server (εξυπηρέτης) στη συνέχεια θα περιμένει την απόπειρα σύνδεση από τον Client, δηλαδή θα «ακούει» (listen) στη θύρα που θα δώσει ο χρήστης του Server.

3.2 Δημιουργία σύνδεσης

- Στην περίπτωση του Server (εξυπηρέτη), η δημιουργίας μίας σύνδεσης θα γίνεται χειροκίνητα με εισαγωγή έγκυρης θύρας (1025-65534).

- Μία και μόνο δημιουργία σύνδεσης (σύνδεση αποδοχής) επιτρέπεται κάθε φορά.

- Στην περίπτωση του Client (πελάτη), η δημιουργίας μίας σύνδεσης θα γίνεται μόνο με χειροκίνητη εισαγωγή hostname ή IPv4, καθώς και χειροκίνητη εισαγωγή αριθμού θύρας.

  Προσοχή: Στην περίπτωση κινητής συσκευής (Android) είναι η έγκυρη μόνον η εισαγωγή Ipv4.

- Μόλις η ζεύξη επιτευχθεί, τότε θα εμφανίζεται στα μηνύματα ελέγχου το αντίστοιχο μήνυμα ότι επιτεύχτει η σύνδεση καθώς και από που προέρχεται η σύνδεση (επίδειξη διεύθυνσης IP).

- Θα υπάρχει κάποιο heartbeat ώστε να γίνεται timeout όταν η σύνδεση παραμένει ανενεργή για πολύ ώρα (βλ. 3.7)

3.3 Ανταλλαγή μηνυμάτων

- Το πρόγραμμα θα επιτρέπει τη γραπτή αμφίδρομη ηλεκτρονική επικοινωνία μεταξύ Server και Client.

- Η επίδειξη της ταυτότητας (Server ή Client) θα γίνεται πάντα πριν την εκτύπωση οιασδήποτε μηνύματος.

3.4 Λίστα αρχείων

- Το πρόγραμμα θα επιτρέπει την αποστολή/παραλαβή της λίστας των αρχείων που υπάρχουν στο directory του.

- Η μεταφορά της πληροφορίας γίνεται πάνω από την σύνδεση ελέγχου (control channel), και θα εκτυπώνει την πληροφορία στα μηνύματα ελέγχου.

- Δεν θα επιτρέπεται αποστολή αρχείων πέρα από το directory (parent/children directories) για αποφυγή πιθανής υποκλοπής.

- Η αποστολή φακέλων θα επιτυγχάνεται μέσω συμπίεσης των περιεχομένων του φακέλου σε συμπιεσμένη μορφή ZIP και μετέπειτα αποστολής του.

3.5 Μεταφορά αρχείων

- Το πρόγραμμα δημιουργεί μια νέα σύνδεση TCP στη θύρα port+1 και περιμένει για την μεταφορά του αρχείου από τον Server (εξυπηρέτης).

- Μια σύνδεση μεταφοράς δεδομένων (data connection) χρησιμοποιείται αποκλειστικά για τη μεταφορά ενός αρχείου.

- Στη συνέχεια, η σύνδεση αυτή κλείνει, όντας η μοναδική (ή , εκάστοτε μοναδική κατά περίπτωση) σύνδεση η οποία κλείνει κατά τη διάρκεια της

επικοινωνίας Server (εξυπηρέτη) και Client (πελάτη).

3.6 Αρχείο καταγραφής (logging)

- Θα υπάρχει δυνατότητα καταγραφής όλων των συμβάντων του προγράμματος (και των λαθών).

3.7 Αποχώρηση / Πρόωρος τερματισμός

- Σε περίπτωση αποχώρησης, θα τερματίζονται όλες οι συνδέσεις.
- Σε περίπτωση πρόωρης αποχώρησης, τότε το πρόγραμμα θα τερματίζει της συνδέσεις βάσει του timeout.

3.8 Λήψη φωτογραφίας JPEG από κάμερα (snapshot)

- Δυνατότητα λήψης ενός στιγμιότυπου από webcam υπολογιστή.
- Αποστολή ποσοστιαίας διαφοράς μεταξύ των δύο πιο προσφάτων απεσταλμένων στιγμιότυπων και ειδοποίηση του Client (πελάτη) για τυχόν ανορθόδοξες παρεκκλίσεις.
- Οι επιτρεπτές (ή μή) παρεκκλίσεις ορίζονται αποκλειστικά από τον Client (πελάτη).

4 Σχεδίαση

Η σχεδίαση του προγράμματος θα γίνει βάσει της θεωρίας της Αλληλεπίδρασης Ανθρώπου-Υπολογιστή (Human-Computer Interface), του Σχεδιασμού Διεπαφής Χρήστη (User Interface – UI – Design) αλλά και του Σχεδιασμού Εμπειρίας Χρήστη (User Experience –UX- Design).

Διαθέσιμη δωρεάν βιβλιογραφία:

- MSDN User Interface Principles

  https://msdn.microsoft.com/en-us/library/windows/desktop/ff728831(v=vs.85).aspx
- Official Android Design Principles

  http://developer.android.com/design/get-started/principles.html

5 Υλοποίηση

Για desktop/laptop PC θα χρησιμοποιηθεί το JDK (Java 8) και συγκεκριμένα το Eclipse IDE με Windowbuilder.

Για κινητές συσκευές και tablet που υποστηρίζουν Android θα χρησιμοποιηθεί το Android SDK και συγκεκριμένα το Android Studio.

Σε κάθε περίπτωση θα χρησιμοποιηθεί έντονος προγραμματισμός με sockets (TCP Socket programming).

Η τελική υλοποίηση θα είναι cross-platform αρκεί να είναι εγκατεστημένη η Java.

Το πρόγραμμα (λόγω της φύσης του) θα εξυπηρετεί μόνο GUI έκδοση και δε θα παρέχεται command-line interface ή command-line arguments.

ΘΕΩΡΙΤΙΚΟ ΥΠΟΒΑΘΡΟ ΠΡΟΓΡΑΜΜΑΤΟΣ

Κεφάλαιο Πρώτο

Εισαγωγή στο Διαδίκτυο

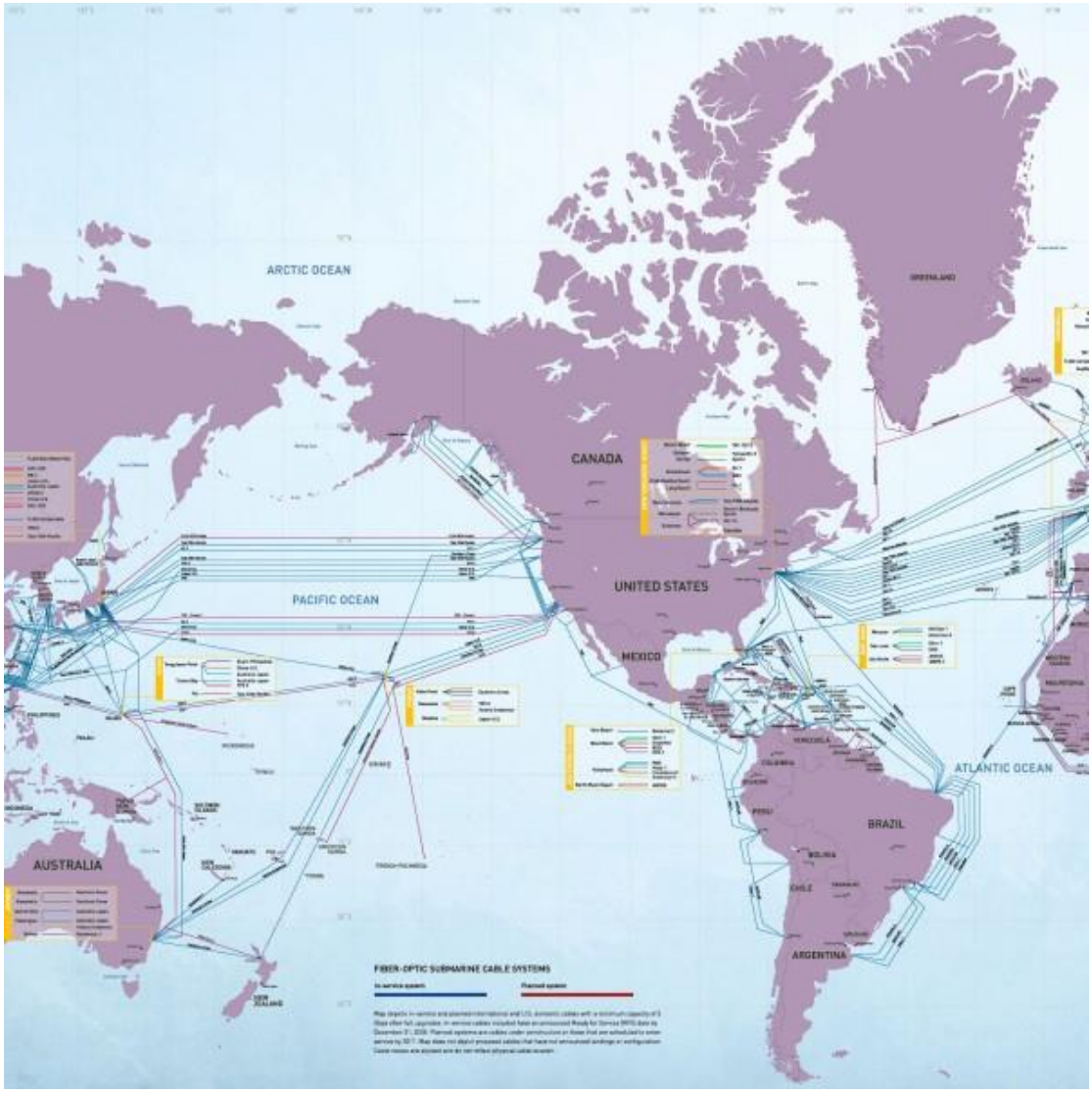
(εικ 1)

Τι είναι το διαδίκτυο?

1.1 Τι είναι το Διαδίκτυο

Το διαδίκτυο πήρε το όνομα του από τις λέξεις **Δια**σύνδεση **Δικτύων,** η αντίστοιχα στην αγγλική ορολογία Internet, **Intern**ational **Net**work.
Το δημόσιο Διαδίκτυο είναι ένα παγκόσμιο σύστημα διασυνδεδεμένων δικτύων υπολογιστών, ένα σύστημα δηλαδή που συνδέει εκατομμύρια υπολογιστικές συσκευές σε όλο τον κόσμο. Οι περισσότερες συσκευές που συνδέονται στο διαδίκτυο είναι παραδοσιακά επιτραπέζια PC, laptops, και την τελευταία εικοσαετία φορητές συσκευές όπως tablet, smartphones. Δεν είναι όμως μόνο αυτά. Τα τελευταία χρόνια μέσω διαδικτύου συνδέονται ολοένα και περισσότερα μη παραδοσιακά τερματικά συστήματα διαδικτύου, από PDA's (Personal digital assistants), τηλεοράσεις, αυτοκίνητα, ψυγεία, συστήματα συναγερμού, μέχρι και τοστιέρες η μηχανές καφέ. Όλες αυτές οι συσκευές ονομάζονται υπολογιστές υπηρεσίας (hosts) ή τερματικά συστήματα (end systems), και ανταλλάσουν μηνύματα (πακέτα) με την χρήση διαφόρων πρωτοκόλλων (τυποποιημένοι κανόνες επικοινωνίας) τα οποία υλοποιούνται σε επίπεδο υλικό και λογισμικό.

Τα τερματικά αυτά συστήματα συνδέονται μεταξύ τους με ζεύξεις επικοινωνίας. Υπάρχουν πολλοί τύποι ζεύξεων επικοινωνίας όπως το ομοαξονικό καλώδιο, το χάλκινο καλώδιο, οι οπτικές ίνες και το ραδιοφάσμα.



(εικ 2) : Υποθαλάσσιο δίκτυο οπτικών ινών

Ανάλογα με τον τύπο της ζεύξης, τα δεδομένα μεταδίδονται με διαφορετικούς ρυθμούς. Ο ρυθμός μετάδοσης ονομάζεται εύρος ζώνης (bandwith), και ορίζεται ως η διαφορά της μεγαλύτερης συχνότητας μείον την μικρότερη συχνότητα του φάσματος συχνοτήτων του ($BW = f_{max}-f_{min}$), αλλα όσον αφορά τις ζεύξεις τυπικά μετράται σε bit/sec.

Τα τερματικά συστήματα, δεν συνδέονται απευθείας μεταξύ τους άμεσα (χρησιμοποιώντας δηλαδή μια ζεύξη επικοινωνίας) αλλά έμμεσα μέσω ηλεκτρονικών συσκευών οι οποίες αναλαμβάνουν την λήψη και αποστολή πακέτων, τους δρομολογητές (routers).

Οι δρομολογητές ανήκουν στο επίπεδο 3 (layer 3) του μοντέλου OSI (Open Systems Interconnection), το επίπεδο δικτύου (Network Layer).

Ένας δρομολογητής λοιπόν, λαμβάνει ένα κομμάτι πληροφορίας που φτάνει μια από τις εισερχόμενες ζεύξεις επικοινωνίας του και το προωθεί σε μια από τις

εξερχόμενες ζεύξεις επικοινωνίας. Το κομμάτι αυτό της πληροφορίας, ονομάζεται πακέτο. Η διαδρομή που παίρνει το πακέτο ονομάζεται μονοπάτι. Το σημαντικό σε αυτό το σημείο, είναι ότι αντί να υπάρχει μια αποκλειστική διαδρομή ανάμεσα στα τερματικά συστήματα που συνδέονται μεταξύ τους, το Διαδίκτυο χρησιμοποιεί μια τεχνική γνωστή και ως μεταγωγή πακέτου, που επιτρέπει σε περισσότερα από ένα τερματικά συστήματα να μοιράζονται είτε ολόκληρη την διαδρομή είτε τμήματά της ταυτόχρονα.

Τα τερματικά συστήματα, οι δρομολογητές και διάφορα άλλα κομμάτια του διαδικτύου εκτελούν πρωτόκολλα. Τα κύρια πρωτόκολλα του διαδικτύου ονομάζονται TCP/IP .

Το Πρωτόκολλο IP είναι υπεύθυνο για τη δρομολόγηση των πακέτων δεδομένων ανάμεσα στα διάφορα δίκτυα, ανεξάρτητα από την υποδομή τους, και αποτελεί το κύριο πρωτόκολλο πάνω στο οποίο είναι βασισμένο το Διαδίκτυο, καθορίζοντας την μορφή των πακέτων που στέλλονται και λαμβάνονται ανάμεσα σε δρομολογητές και τερματικά συστήματα. Το πρωτόκολλο IP εισήχθη από τους Vint Cerf και Bob Kahn το 1974 και η πρώτη μεγάλης κλίμακας έκδοση του Πρωτοκόλλου IP, ήταν η έκδοση 4 (**IPv4**) η οποία επικρατεί μέχρι και σήμερα σε όλο το Διαδίκτυο

Το πρωτόκολλο TCP (Transmittion Control protocol, πρωτόκολλο ελέγχου μετάδοσης) , σαν στόχο έχει να επιβεβαιώνεται η αξιόπιστη αποστολή και λήψη δεδομένων, όπως επίσης να μεταφέρονται τα δεδομένα χωρίς λάθη μεταξύ του στρώματος δικτύου (network layer) και του στρώματος εφαρμογής (application layer) και, φτάνοντας στο πρόγραμμα του στρώματος εφαρμογής, να έχουν σωστή σειρά.

Ιστορικά στοιχεία του Διαδικτύου.

1.2 Ιστορικά Στοιχεία διαδικτύου

Οι πρώτες απόπειρες για την δημιουργία ενός διαδικτύου ξεκίνησαν στις ΗΠΑ κατά την διάρκεια του ψυχρού Πολέμου, στα τέλη της δεκαετίας του ’60. Την εποχή εκείνη, ο Ψυχρός Πόλεμος ήταν μεγάλη απειλή για τις ΗΠΑ. Η Ρωσία είχε ήδη στείλει στο διάστημα τον δορυφόρο Σπούτνικ 1 κάνοντας τους Αμερικανούς να φοβούνται όλο και περισσότερο για την ασφάλεια της χώρας τους. Υπήρχε ένα μεγάλο πρόβλημα, σχετικά με τον τρόπο της επιτυχούς επικοινωνίας μεταξύ των αμερικάνικών αρχών, μετά από έναν ενδεχόμενο πυρηνικό πόλεμο. Η Αμερική χρειαζόταν ένα δίκτυο διοίκησης κι ελέγχου που θα συνέδεε πόλεις, πολιτείες και στρατιωτικές βάσεις. Το πρόβλημα όμως ήταν ότι η τεχνική υποδομή ενός τέτοιου δικτύου θα ήταν πάντα τρωτή σε επίθεση, με τα κέντρα οργάνωσης του δικτύου να είναι ευάλωτα και ν’ αποτελούν προφανείς στρατιωτικούς στόχους. Θέλοντας λοιπόν να προστατευτούν από μια πιθανή πυρηνική επίθεση των Ρώσων δημιούργησαν την υπηρεσία προηγμένων αμυντικών ερευνών *ARPA* (Advanced Research Project Agency) γνωστή ως DARPA (*Defense Advanced Research Projects Agency*) στις μέρες μας. Ο οργανισμός αυτός, προσανατολισμένος σε ερευνητικά προγράμματα υψηλής τεχνολογίας, ξεκίνησε μια ερευνητική δραστηριότητα σχετικά με τα δίκτυα μεταγωγής δεδομένων, τα λεγόμενα Packet Switched Networks. Η τεχνική στα δίκτυα αυτής της υλοποίησης (η οποία σήμερα χρησιμοποιείται ευρύτατα) βασίζεται στον τεμαχισμό σε πακέτα των δεδομένων που πρόκειται να μεταφερθούν. Τα πακέτα αυτά δρομολογούνται από κόμβο σε κόμβο και συναρμολογούνται ξανά όταν φτάσουν στον προορισμό τους.

Το αρχικό θεωρητικό υπόβαθρο δόθηκε από τον Τζ. Λικλάιντερ (J.C.R. Licklider) που ανέφερε σε συγγράμματά του το "γαλαξιακό δίκτυο". Η θεωρία αυτή υποστήριζε την ύπαρξη ενός δικτύου υπολογιστών που θα ήταν συνδεδεμένοι μεταξύ τους και θα μπορούσαν να ανταλλάσσουν γρήγορα πληροφορίες και προγράμματα. Το επόμενο θέμα που προέκυπτε ήταν ότι το δίκτυο αυτό θα έπρεπε να ήταν αποκεντρωμένο έτσι ώστε ακόμα κι αν κάποιος κόμβος του δεχόταν επίθεση να υπήρχε δίοδος επικοινωνίας για τους υπόλοιπους υπολογιστές. Τη λύση σε αυτό έδωσε ο Πολ Μπάραν (Paul Baran) με τον σχεδιασμό ενός κατανεμημένου δικτύου επικοινωνίας που χρησιμοποιούσε την ψηφιακή τεχνολογία. Πολύ σημαντικό ρόλο έπαιξε και η θεωρία ανταλλαγής πακέτων του Λέοναρντ Κλάινροκ (Leonard Kleinrock), που υποστήριζε ότι πακέτα πληροφοριών που θα περιείχαν την προέλευση και τον προορισμό τους μπορούσαν να σταλούν από έναν υπολογιστή σε έναν άλλο.

(εικ 3) : Το «πρώτο διαδίκτυο» ARPANET.

Στηριζόμενο λοιπόν σε αυτές τις τρεις θεωρίες δημιουργήθηκε το πρώτο είδος διαδικτύου γνωστό ως ARPANET. Εγκαταστάθηκε και λειτούργησε για πρώτη φορά το 1969 με 4 κόμβους μέσω των οποίων συνδέονται 4 μίνι υπολογιστές (mini computers 12k): του πανεπιστημίου της Καλιφόρνια στην Σάντα Μπάρμπαρα του πανεπιστημίου της Καλιφόρνια στο Λος Άντζελες, το SRI στο Στάνφορντ και το πανεπιστήμιο της Γιούτα. Η ταχύτητα του δικτύου έφθανε τα 50 kbps και έτσι επιτεύχθηκε η πρώτη dial up σύνδεση μέσω γραμμών τηλεφώνου. Μέχρι το 1972 οι συνδεδεμένοι στο ARPANET υπολογιστές έχουν φτάσει τους 23, οπότε και εφαρμόζεται για πρώτη φορά το σύστημα διαχείρισης ηλεκτρονικού ταχυδρομείου (e-mail).

Παράλληλα δημιουργήθηκαν και άλλα δίκτυα, τα οποία χρησιμοποιούσαν διαφορετικά πρωτόκολλα (όπως το x.25 και το UUCP) τα οποία συνδέονταν με το ARPANET. Το πρωτόκολλο που χρησιμοποιούσε το ARPANET ήταν το NCP (Network Control Protocol), το οποίο, όμως, είχε το μειονέκτημα ότι λειτουργούσε μόνο με συγκεκριμένους τύπους υπολογιστών. Έτσι, δημιουργήθηκε η ανάγκη στις αρχές του 1970 για ένα πρωτόκολλο που θα ένωνε όλα τα δίκτυα που είχαν δημιουργηθεί μέχρι τότε. Το 1974 λοιπόν, δημοσιεύεται η μελέτη των Βιντ Σερφ (Vint Cerf) και Μπομπ Κάαν (Bob Kahn) από την οποία προέκυψε το πρωτόκολλο TCP (Transmission Control Protocol) που αργότερα το 1978 έγινε TCP/IP, προσετέθη δηλαδή το Internet Protocol (IP), ώσπου το 1983 έγινε το μοναδικό πρωτόκολλο που ακολουθούσε το ARPANET.

Το 1984 υλοποιείται το πρώτο DNS (Domain Name System) σύστημα στο οποίο καταγράφονται 1000 κεντρικοί κόμβοι και οι υπολογιστές του διαδικτύου πλέον αναγνωρίζονται από διευθύνσεις κωδικοποιημένων αριθμών. Ένα ακόμα σημαντικό βήμα στην ανάπτυξη του Διαδικτύου έκανε το Εθνικό Ίδρυμα Επιστημών (National

Science Foundation, NSF) των ΗΠΑ, το οποίο δημιούργησε την πρώτη διαδικτυακή πανεπιστημιακή ραχοκοκκαλιά (backbone), το NSFNet, το 1986. Ακολούθησε η ενσωμάτωση άλλων σημαντικών δικτύων, όπως το Usenet, το Fidonet και το Bitnet.

Ο όρος Διαδίκτυο/Ιντερνετ ξεκίνησε να χρησιμοποιείται ευρέως την εποχή που συνδέθηκε το ARPANET με το NSFNet και Internet σήμαινε οποιοδήποτε δίκτυο χρησιμοποιούσε TCP/IP. Η μεγάλη άνθιση του Διαδικτύου όμως, ξεκίνησε με την εφαρμογή της υπηρεσίας του Παγκόσμιου Ιστού από τον Τιμ Μπέρνερς-Λι στο ερευνητικό ίδρυμα CERN το 1989, ο οποίος είναι στην ουσία, η "πλατφόρμα", η οποία κάνει εύκολη την πρόσβαση στο Ίντερνετ, ακόμα και στη μορφή που είναι γνωστό σήμερα.

Μετά την άρση του περιορισμού στην εμπορική χρήση του δικτύου, το 1991, η ανάπτυξη του Internet πραγματοποιήθηκε με εκθετικούς ρυθμούς. Ως τότε, το Internet ήταν περιορισμένο σε ερευνητική, εκπαιδευτική και κυβερνητική χρήση. Οι εμπορικές χρήσεις του Διαδικτύου απαγορεύονταν, εκτός αν εξυπηρετούσαν άμεσα τους στόχους της έρευνας και της εκπαίδευσης.

Το 1993 το CERN παρουσίασε το πλοηγητή σελίδων. Το 1994 ο Mark Andreessen σχεδίασε ένα γραφικό εργαλείο για το World Wide Web, το Mosaic για τα X Windows. Αποτέλεσε το πιο πετυχημένο πρόγραμμα πλοήγησης του World Wide Web, κατοπινή εξέλιξη του οποίου αποτέλεσε ο Netscape. Και οι δύο έδιναν τη δυνατότητα ανταλλαγής σελίδων με γραφικές αναπαραστάσεις, κάτι που οδήγησε σε μια πραγματική έκρηξη στην ανάπτυξη και χρήση του διαδικτύου.

Το 1994, στα 25α γενέθλια του ARPAnet, περισσότεροι από 3 εκατομμύρια διακομιστές ήταν συνδεδεμένοι. Δύο χρόνια αργότερα, το 1996, ο αριθμός των διακομιστών πλησίασε τα 10.000.000.

Έτος	Γεγονός
1957	Δημιουργία του ARPA
1969	Δημιουργία του ARPAnet
1982	Υιοθέτηση του TCP/IP
1983	Ενσωμάτωση του TCP/IP στο UNIX
1986	Δημιουργία του NSFnet
1990	Δημιουργία του HTTP στο CERN
1992	Κυκλοφορία του MOSAIC
1993	Κυκλοφορία του NETSCAPE NAVIGATOR
1995	Κυκλοφορία του INTERNET EXPLORER

Έτος	Αριθμός υπολογιστών
1977	111
1981	213
1983	562
1984	1.000
1986	5.000
1987	10.000
1989	100.000
1992	1.000.000
2001	150.000.000-175.000.000
2002	>200.000.000
2010	80% του πλανήτη θα είναι στο διαδίκτυο

(εικ 4) : Χρονοδιάγραμμα Ιντερνετ

Πρωτόκολλα Μεταφοράς Διαδικτύου.

1.3 Πρωτόκολλα Μεταφοράς Διαδικτύου

Στο διαδίκτυο και τα δίκτυα TCP/IP κάνουν διαθέσιμα δύο πρωτόκολλα μεταφοράς σε εφαρμογές, το UDP (User Datagram) και το TCP (Transmission Control Protocol).

Όταν κάποιος προγραμματιστής ενδιαφέρεται να δημιουργήσει μια νέα εφαρμογή, η πρωταρχική απόφαση που πρέπει να πάρει είναι ποιο από τα δύο παραπάνω πρωτόκολλα θα χρησιμοποιήσει, καθώς καθενα από αυτά προσφέρει ένα διαφορετικό μοντέλο υπηρεσιών στις εφαρμογές που θέλει να υλοποιήσει.

1.3,1 Πρωτόκολλο TCP.

Το TCP *(Transmission Control Protocol - Πρωτόκολλο Ελέγχου Μεταφοράς)* είναι ένα από τα κυριότερα πρωτόκολλα που χρησιμοποιούνται στο διαδίκτυο. Βρίσκεται πάνω από το IP protocol *(πρωτόκολλο IP)*. Οι κύριοι στόχοι του πρωτοκόλλου TCP, όπως αναφέρθηκαν παραπάνω, είναι να επιβεβαιώνεται η αξιόπιστη αποστολή και λήψη δεδομένων, επίσης να μεταφέρονται τα δεδομένα χωρίς λάθη μεταξύ του στρώματος δικτύου (network layer) και του στρώματος εφαρμογής (application layer) και, φτάνοντας στο πρόγραμμα του στρώματος εφαρμογής, να έχουν σωστή σειρά. Οι περισσότερες υπηρεσίες διαδικτύου, χρησιμοποιούν το πρωτόκολλο TCP, όπως :

1) Το πρωτόκολλο SMTP (Simple Mail Transfer Protocol) (port 25)
2) TelNet (Telecommunication Network) (Port 23)
3) FTP (File transfer Protocol, *Πρωτόκολλο Μεταφοράς Αρχείων*)
4) HTTP (HyperText Transfer Protocol, Πρωτόκολλο Μεταφοράς Υπερκειμένου) (port 80)

Όταν ένας προγραμματιστής επιλέγει το πρωτόκολλο TCP σαν πρωτόκολλο μεταφοράς, η εφαρμογή που δημιουργεί δεχεται αυτές τις υπηρεσίες:

- Συνδεσιστρεφής υπηρεσία : Ονομάζεται αλλιώς και διαδικασία χειραψίας, και προειδοποιεί τον πελάτη και τον εξυπηρέτη, να προετοιμαστούν για μια ανταλλαγή πακέτων. Αφού επιτευχθεί η χειραψία, υπάρχει μια σύνδεση TCP αναμεσα στις sockets των δύο διεργασιών. Η σύνδεση μεταξύ πελάτη και εξυπηρετή, είναι αμφίδρομοι και μπορούν και οι δύο ταυτόχρονα να στελνουν μηνύματα ή πακέτα μεταξύ τους. Όταν η εφαρμογή τελειώσει την μεταφορά μηνυμάτων ή δεδομένων, πρέπει να τερματίσει και την σύνδεση.
- Αξιόπιστη υπηρεσία μεταφοράς : Το πρωτόκολλο TCP είναι αξιόπιστο οσον αφορά την μη απώλεια δεδομένων : Όταν η μια πλευρά της εφαρμογής

στελνει ένα ρευμα bytes σε μια socket, μπορει να βασίζεται στο TCP να παραδώσει το ίδιο ρευμα bytes στην socket λήψης, χωρίς απώλεια byte ή επαναλαμβανόμενα bytes.

Το TCP περιλαμβάνει και έναν μηχανισμό ελέγχου συμφόρτησης, μια υπηρεσία για την συνολική ευεξία του Διαδικτύου αντί του άμεσου όφελους των επικοινωνιακών διεργασιών. Το αρνητικό είναι ότι στο TCP εχει βλαπτική επίδραση σε εφαρμογές πραγματικού χρόνου όπως η μετάδοση ήχου και βίντεο, γιαυτό και οι προγραμματιστές πραγματικού χρόνου προτιμούν την υλοποίηση των προγραμμάτων τους με UDP, καθώς και το UDP πέρα από τα παραπάνω είναι ανεκτό σε απώλειες πραγματικού χρόνου.

Μια σύνδεση TCP παρέχει μια αμφίδρομη μεταφορά δεδομένων. Αν υπάρχει μια σύνδεση TCP αναμεσα σε μια διαδικασία A σε έναν υπολογιστή και σε μια διαδικασία B σε έναν άλλον υπολογιστή, τοτε τα δεδομένα ρέουν ταυτόχρονα από το A στο B αλλα και από το B στο A.

Μια σύνδεση TCP επίσης είναι πάντα από σημείο προς σημείο, δηλαδή αναμεσα σε έναν αποστολέα και έναν δέκτη. Η λεγόμενη «πολυεκπομπή» , η μεταφορά δηλαδή δεδομένων από ένα αποστολέα σε πολλούς δέκτες δεν είναι δυνατή με το TCP.

Καθορισμός σύνδεσης TCP

Υποθέτουμε ότι μια διεργασία που πραγματοποιείται σε έναν υπολογιστή υπηρεσίας, θέλει να εκκινήσει σύνδεση με μια άλλη διεργασία σε έναν υπολογιστή υπηρεσίας. Οι διεργασίες αυτές ονομάζονται διεργασία πελάτη (client) και διεργασία εξυπηρέτη (host) . Η διεργασία εφαρμογής client ειδοποιεί πρώτα το επίπεδο μεταφοράς πελάτη ότι θέλει να καθορίσει μια σύνδεση με μια διεργασία στον εξυπηρέτη. Με την Java αυτό υλοποιείται εκδίδοντας την εντολή :

Socket clientSocket = new Socket (“hostname” , “portNumber”) ;

Όπου Hostname είναι το όνομα του εξυπηρέτη και portNumber ταυτοποιεί την διεργασία στον εξυπηρέτη. Το επίπεδο μετά στο client (πελάτη) προχωρεί κατόπιν και καθορίζει ότι έγινε σύνδεση TCP με το TCP στον host. Η διαδικασία καθορισμού σύνδεσης αποτελεί την τριμερή χειραψία, καθώς στέλνονται τρία τμήματα για την πραγματοποίηση της.

Αφού γίνει η «χειραψία», οι δυο διεργασίες εφαρμογής μπορούν να στείλουν δεδομένα μεταξύ τους. Η διεργασία client περνά ένα ρευμα δεδομένων μέσω της socket. Η μέγιστη ποσότητα δεδομένων που μπορούν να συλλεχθούν και να

τοποθετηθούν μεσα σε ένα τμήμα περιορίζεται από το μέγιστο στο μέγεθος τμήματος (MSS, maximum segment size). Το MSS καθορίζεται από την υλοποίηση TCP (που καθορίζεται από το λειτουργικό σύστημα) και συχνά μπορεί να παραμετροποιηθεί (πχ τιμών : 1.500 bytes, 536 bytes, 523 bytes).

Δομή Τμήματος TCP

Τα πακέτα του πρωτοκόλλου TCP καλούνται segments (τμήματα).

Ένα από τα κυριότερα μέρη ενός segment είναι η TCP επικεφαλίδα (TCP header), η οποία παρέχει συγκεκριμένες πληροφορίες για το πρωτόκολλο TCP. Το ελάχιστο μέγεθος της επικεφαλίδας είναι 5 words και το μέγιστο 15 words (απουσία ή παρουσία όλων των options αντίστοιχα).

(εικόνα στην επομενη σελίδα)

Source Port

Αυτό το πεδίο προσδιορίζει την port (θύρα) του αποστολέα

Destination Port

Αυτό το πεδίο προσδιορίζει την port (θύρα) του παραλήπτη

Sequence Number

Ο sequence number (αριθμός ακολουθίας) έχει διπλό ρόλο:

- Εαν υπάρχει η SYN flag (SYN σημαία) τότε είναι ο αρχικός αριθμός ακολουθίας (ISN - initial sequence number) και η πρώτη octet δεδομένων του πακέτου είναι ο ISN+1.
- Αλλιώς, εαν δεν υπάρχει η SYN flag, τότε η πρώτη octet δεδομένων είναι ο αριθμός ακολουθίας.

TCP επικεφαλίδα				
+	Bits 0 - 3	4 - 9	10 - 15	16 - 31
0	Source Port Θύρα Προέλευσης			Destination Port Θύρα Προορισμο
32	Sequence Number Αριθμός ακολουθίας			
64	Acknowledgment Number Αριθμός επιβεβαίωσης			
96	Data Offset	Reserved	Flags Σημίς	Window Παράθυρο
128	Checksum Άθροισμα ελέγχου			Urgent Pointer Επείγοντα δεδομένα
160	Options Επιλογές (προαιετικές)			
160/92+	Data Δεδομένα			

Acknowledgment number

Όταν υπάρχει η ACK flag η τιμή αυτού του πεδίου δείχνει τον επόμενο sequence number (αριθμό ακολουθίας) που αναμένει ο αποστολέας.

Data offset

Είναι ο αριθμός από words μεγέθους 32 bit στην επικεφαλίδα TCP (TCP header). Καθορίζει το μέγεθος της επικεφαλίδας (πολλαπλάσιο του 32) και επομένως δείχνει και την αρχή των δεδομένων[2].

Reserved

Πεδίο 6 bit "κρατημένων" (αγγλ. reserved) για μελλοντική χρήση. Η τιμή των bit πρέπει να είναι 0.

Flags *(επίσης γνωστό ως bits ελέγχου - Control bits)*

Περιέχει 6 bit - σημαίες:

Σημαία	Σημασία	Προέλευση ονομασίς
URG	Το πεδίο urgentpointer είναι σημαντικό	**URG**ent
ACK	Το πεδί επιβεβαίωσης είναι σημαντικό	**ACK**nowledgment
PSH	Λειτουργία ώθησης	**P**u**SH**
RST	Επαναρύθμιση σύνδεσης	**R**e**S**e**T**
SYN	Σγρονιμός αριθμών ακολουθίας	**SYN**chronize
FIN	Ο αποστολέας δεν στέλνει άλα δεδομένα	**FIN** (=τέλος)

Window

Ο αριθμός από octets δεδομένων (bytes) που επιθυμεί να δεχτεί ο αποστολέας του πακέτου, αρχίζοντας από εκείνη που δείχνει το πεδίο επιβεβαίωσης (acknowledgment field).

Checksum

Το πεδίο Checksum μεγέθους 16 bit χρησιμοποιείται για έλεγχο λαθών στην επικεφαλίδα και στα δεδομένα.

Options

Μεταβλητή, η οποία καθορίζει ειδικές επιλεγόμενες ρυθμίσεις και μπορεί να καταλάβει χώρο στο τέλος της επικεφαλίδας TCP (TCP header). Το μήκος τους είναι πολλαπλάσιο των 8 bit και σε το περιεχόμενο της επικεφαλίδας μετά την τελευταία επιλογή πρέπει να γεμίζει (πχ. με μηδενικά - 0). Με αυτόν τον τρόπο το data offset θα δείχνει σωστά την αρχή των δεδομένων.

Urgent pointer

Εάν είναι ενεργοποιημένο το URG bit ελέγχου, τότε αυτό το πεδίο δείχνει τον αριθμό ακολουθίας (sequence number)της octet που βρίσκεται αμέσως μετά το

τελευταίο byte από τα επείγοντα δεδομένα. Έτσι παρουσιάζει τη θέση του τελευταίου byte με επείγοντα δεδομένα.

1.3,2 Πρωτόκολλο UDP.

Το πρωτόκολλο User Datagram Protocol (UDP), είναι ένα απλό, ελαφρύ πρωτόκολλο μεταφοράς, από τα βασικά πρωτόκολλα του διαδικτύου. Μία εναλλακτική ονομασία του πρωτοκόλλου είναι Universal Datagram Protocol. Διάφορα προγράμματα χρησιμοποιούν το πρωτόκολλο UDP για την αποστολή σύντομων μηνυμάτων (γνωστών και ως datagrams).

Χρησιμοποιείται σε :

1) Domain Name System (DNS)
2) IPTV
3) Voice over IP (VoiP)
4) Trivial File Transfer Protocol (TFTP)
5) Παιχνίδια online.

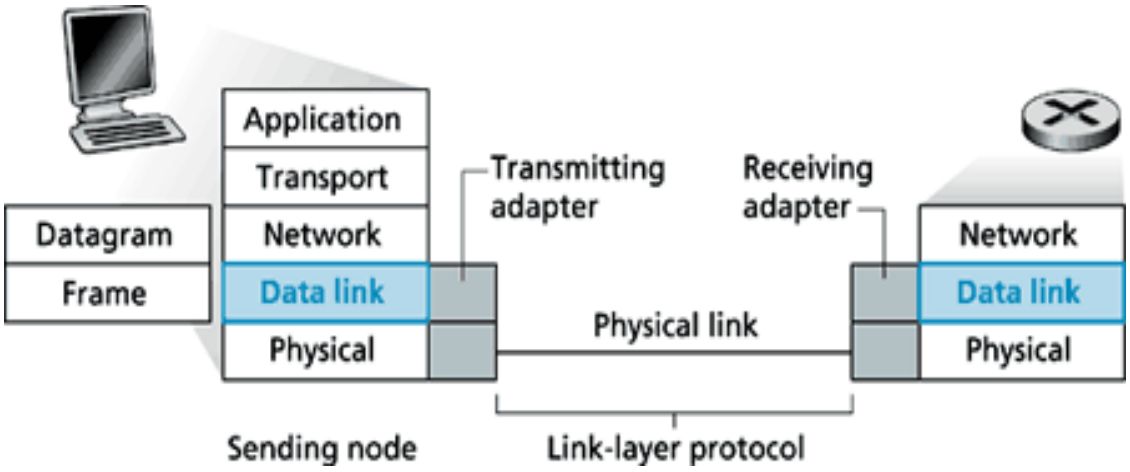
Το UDP παρέχει μια αναξιόπιστη υπηρεσία μεταφοράς δεδομένων και είναι ασυνδεσιστρεφές.

Γίνεται αναφορά στο πρωτόκολλο UDP για να εξηγηθεί γιατί στο πρόγραμμα που υλοποίησα χρησιμοποίησα πρωτόκολλο FTP. Το FTP υπερτερεί του UDP όσον αφορά την **αξιοπιστία**, καθώς γίνεται έλεγχος ώστε να διασφαλίζεται ότι τα πακέτα θα φτάσουν αναλλοίωτα στον χρήστη, και σε περίπτωση μη επιβεβαίωσης λήψης του πακέτου το πακέτο προαποστέλλεται, ενώ στο UDP δεν είναι σίγουρος ο χρήστης εάν το πακέτο θα φτάσει σωστά στον προορισμό του ή εάν θα χαθεί μέσα στο δίκτυο καθώς δεν έχει προβλεφθεί η δυνατότητα επιβεβαίωσης λήψης πακέτου από τον παραλήπτη, ούτε η επαναμετάδοση ενός χαμένου πακέτου.

Το FTP υπερτερεί του UDP και στην σειρά πακέτων. Εάν δύο πακέτα αποσταλούν σε μία σύνδεση το ένα μετά το άλλο, τότε το πρωτόκολλο TCP εγγυάται ότι θα φτάσουν στον παραλήπτη με την ίδια σειρά με την οποία στάλθηκαν. Στην περίπτωση που λείπει ένα πακέτο και έρθουν μελλοντικά πακέτα, τότε αυτά κατακρατούνται στην προσωρινή μνήμη (buffer) μέχρις ότου φτάσει το πακέτο που λείπει. Τότε αναδιατάσσονται και εμφανίζονται με την σωστή σειρά στον παραλήπτη. Τα πακέτα UDP, σε αντίθεση με το TCP, δεν αριθμούνται και κατά συνέπεια δεν υπάρχει κάποια συγκεκριμένη σειρά με την οποία θα πρέπει να φτάσουν στον παραλήπτη.

Επίπεδο ζεύξης Δεδομένων (Data Link Layer)

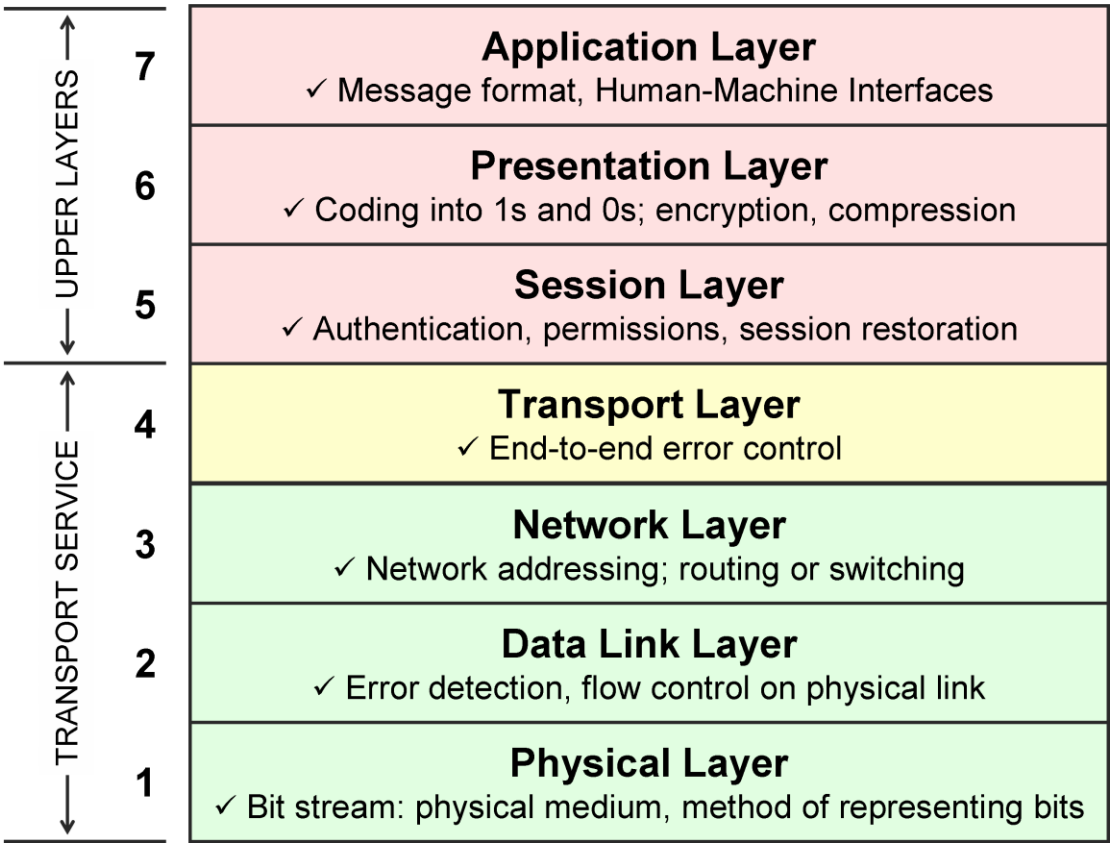
1.4 Επίπεδο ζεύξης Δεδομένων



(εικ 5) : Επίπεδο ζεύξης Δεδομένων

Πώς τα πακέτα στέλνοται επάνω σε ζευξεις μέσα στην διαδρομή επικοινωνίας από άκρο ως άκρο?

Συζητώντας για το επίπεδο ζεύξης, αναγνωρίζουμε δύο είδη διαφορετικών τύπων καναλιών. Ο πρώτος τύπος, αποτελείται από κανάλια εκπομπής, τα οποία είναι κοινά σε δίκτυα τοπικής περιοχής (LAN) , ασύρματα LAN, δορυφορικά δίκτυα και δίκτυα πρόσβασης υβριδικού ινοομοαξονικού καλωδίου (hybrid fiber-cable, HFC).Για κανάλι εκπομπής, πολλοί υπολογιστές υπηρεσίας συνδέονται στο ίδιο κανάλι επικοινωνίας και χρειάζεται ένα πρωτόκολλο πρόσβασης για να συντονίζει τις μεταδώσεις του και να αποφεύγει τις «συγκρούσεις». Ο δεύτερος τύπος καναλιού επιπέδου ζεύξης είναι η ζεύξη επικοινωνίας σημείου προς σημείο, όπως συμβαίνει μεταξύ δυο δρομολογητών ή μεταξύ ενός οικιακού μόντεμ μέσω τηλεφώνου και ενός δρομολογητή ISP. Ο συντονισμός της πρόσβασης σε μια ζεύξη είναι τετριμμένος, αλλα υπάρχουν σημαντικά θέματα που αφορούν τα πλαίσια, την αξιόπιστη μεταφορά δεδομένων, την ανίχνευση σφαλμάτων και τον έλεγχο ροής.



(εικ 6) : Μοντέλο δικτύωσης OSI

Το Επίπεδο Ζεύξης Δεδομένων, αποτελεί το πρώτο εκ των τεσσάρων επιπέδων TCP/IP (αντίστοιχα είναι το δεύτερο επίπεδο του μοντέλου δικτύωσης OSI). Το επίπεδο ζεύξης δεδομένων ενός δικτύου καθορίζεται από πρωτόκολλα τα οποία ρυθμίζουν τη μετάδοση δεδομένων σε ένα τηλεπικοινωνιακό κανάλι αποτελούμενο από ένα μοναδικό φυσικό μέσο (π.χ. σε ενσύρματο τοπικό δίκτυο, όπου το κοινό φυσικό μέσο είναι ένα καλώδιο, σε ασύρματο τοπικό δίκτυο, όπου το κοινό φυσικό μέσο είναι ο ελεύθερος χώρος, ή σε σύνδεση από σημείο-σε-σημείο, όπου το φυσικό μέσο δεν είναι κοινό καθώς μπορεί να προσπελαστεί μόνο από τους κόμβους στα δύο άκρα επικοινωνίας).

Ένα πρωτόκολλο επιπέδου ζεύξης χρησιμοποιείται για να μεταφέρει ένα δεδομενόγραμμα επάνω σε μια ζεύξη, ορίζοντας την μορφή των πακέτων (PDU) που ανταλλάζονται μέσα στους κόμβους και στα άκρα της ζεύξης καθώς και τις ενέργειες οι οποίες πραγματοποιούνται από αυτούς τους κόμβους όταν ανταλλάζουν πακέτα.

Οι μονάδες δεδομένων που ανταλλάσονται από ένα Data Link Layer ονομάζονται frames.

Οι υπηρεσίες που παρέχουν τα επίπεδα ζεύξης δεδομένων είναι οι εξής :

1) Πλαισίωση. Τα πρωτόκολλα επιπέδου ζευξης ενθυλακώνουν κάθε δεδομενόγραμμα επιπέδου ζεύξης μέσα σε ένα πλαίσιο επιπέδου ζεύξης, πριν το μεταδώσουν στην ζεύξη. Οι κεφαλίδες πλαισίου, συχνά περιλαμβάνουν πεδία για την καλούμενη φυσική διεύθυνση ενός κόμβου, η οποία είναι πλήρως διακριτή από την διεύθυνση επιπέδου δικτύου ενός κόμβου (πχ. IP)
2) Πρόσβαση ζεύξης. Ένα πρωτόκολλο πρόσβασης μέσου (MAC,media access control) καθορίζει τους κανόνες με βάση τους οποίους ένα πλαίσιο μεταδίδεται επάνω στην ζεύξη. Αυτό το πρωτόκολλο εξυπηρετεί στο να συντονίζει τις μεταδόσεις πλαισίου των πολλών κόμβων.
3) Αξιόπιστη Παράδοση. Όταν παρέχεται αυτή η υπηρεσία, εγγυάται ότι θα μεταφέρει κάθε δεδομενόγραμμα επιπέδου δικτύου επάνω στην ζεύξη χωρίς σφάλματα. Μια τέτοια υπηρεσία χρησιμοποιείται συχνά για ζεύξεις που υπάρχει πιθανότητα να έχουν μεγάλο αριθμό σφαλμάτων, όπως παραδειγματος χάρη μια ασύρματη ζευξη, με στόχο να διορθόσει το σφάλμα τοπικά χωρίς να ζητήσει αναμετάδοση από άκρο σε άκρο των δεδομένων από ένα πρωτόκολλο επιπέδου μεταφοράς η εφαρμογής. Μερικές φορές βέβαια αυτός ο επιπρόσθετος φόρτος θεωρείται άχρηστος, για ζευξεις με χαμηλό αριθμό σφαλμάτων όπως π.χ. για ζεύξεις οπτικών ινών, ομοαξονικών καλωδίων και χάλκινων συνεστραμμένων ζευγών. Για αυτό το λόγο, πολλά πρωτόκολλα επιπέδου ενσύρματης ζευξης δεν παρέχουν μια υπηρεσία αξιόπιστης παράδοσης.
4) Έλεγχος ροής. Οι κόμβοι, σε κάθε πλευρά της ζεύξης, εχουν συγκεκριμένη χωρητικότητα καταχώρησης πλαισίων. Ο Έλεγχος ροής, αποτρέπει το πρόβλημα υπερχείλισης, ετσι ώστε να μην δεχθεί ο κόμβος λήψης πλαίσια με ρυθμό ταχύτερο από αυτόν που μπορεί να επεξεργαστεί.
5) Ανίχνευση σφαλμάτων. Ο δέκτης ενός κόμβου, μπορει να κάνει λάθος, ότι ένα bit στο πλαίσιο είναι μηδέν, ενω μεταδώθηκε σαν ένα και το αντίστροφο. Τέτοιου τύπου σφάλματα, συμβαίνουν από εξασθένηση σήματος και ηλεκτρομαγνητικό θόρυβο. Δεν υπάρχει όμως λόγος να προωθηθεί κάποιο δεδομενόγραμα που περιέχει ένα σφάλμα, και για αυτό, πολλά πρωτόκολλα ζεύξης παρέχουν ένα μηχανισμό για ανίχνευση ύπαρξης ενός η περισσοτέρων σφαλμάτων. Αυτό γίνεται κάνοντας τον κόμβο μετάδοσης να θέτει bits ανίχνευσης σφάλματος μέσα στο πλαίσιο και κάνοντας τον κόμβο δέκτη να εκτελεί έναν έλεγχο σφάλματος. Είναι η πιο εξεζητιμένη υπηρεσία και υλοποιείται μέσω υλικού.
6) Διόρθωση σφάλματος. Είναι παρόμοια με την ανίχνευση σφάλματος, εκτός του ότι ένας δεκτης δεν μπορεί μονο να ανιχνεύσει σφάλματα που έχουν

εισαχθεί μέσα στο πλαίσιο, αλλα επίσης καθορίζει και το σημείο του πλαισίου που συνέβησαν τα σφάλματα και έτσι να τα διορθώσει.

7) Ημιαμφίδρομη και αμφίδρομη. Με την αμφίδρομη μετάδοση, οι κόμβοι και στα δύο άκρα μιας ζεύξης μπορούν να μεταδώσουν πακέτα ταυτόχρονα. Με ημιαμφίδρομη μετάδοση, ενας κόμβος δεν μπορεί να μεταδίδει και να λαμβάνει ταυτόχρονα.

Για μία δεδομένη ζευξη επικοινωνίας, το πρωτόκολλο επιπέδου ζεύξης, υλοποιείται μέσα σε έναν προσαρμογέα. Ένας προσαρμογέας, είναι μια πλακέτα (ή κάρτα PCMCIA) που συνήθως περιέχει τσιπ RAM, DSP, μια διασύνδεση διαυλου υπολογιστή υπηρεσίας και μια διασύνδεση ζεύξης.

Οι προσαρμογείς είναι γνωστοί σαν κάρτες διασύνδεσης δικτύου (Network Interface Cards).

(εικ 7-8) : Κάρτα διασύνδεσης δικτύου, πλακέτα PCMCIA

Ο Προσαρμογέας, ενθυλακώνει το δεδομενόγραμμα μέσα σε ένα πλαίσιο, και μεταδίδει το πλαίσιο μέσα στην ζευξη επικοινωνίας. Από την άλλη πλευρά, ο προσαρμογέας λήψης, δέχεται όλο το πλαίσιο, εξάγει το δεδομενόγραμμα επιπέδου δικτύου και το περνά στο επίπεδο δικτύου. Αν το πρωτόκολλο επιπέδου ζευξης, παρέχει ανίχνευση σφαλμάτων, τότε ο προσαρμογέας απόστολής θέτει bits ανίχνευσης σγαλμάτων και ο προσαρμογέας λήψης κάνει έλεγο του σφάλματος. Αν το πρωτόκολλο επιπέδου ζεύξης παρέχει αξιόπιστη παράδοση, τότε οι μηχανισμοί για αξιόπιστη μετάδοση υλοποιούνται πλήρως μεσα στους προσαρμογείς. Αν το πρωτόκολλο επιπέδου ζεύξης παρέχει τυχαία πρόσβαση, τότε το πρωτόκολλο τυχαίας πρόσβασης υλοποείται εξ ολοκλήρου στους προσαρμογείς.

Ένας προσαρμογέας, είναι μια ημιαυτόνομη μονάδα. Για παράδειγμα, αν δεχτεί ένα πλαίσιο, καθορίζει αν το πλαίσιο έχει σφάλμα και μπορεί να απορρίψει το πλαίσιο, χωρίς να ειδοποιήσει τον «πατρικό» του κόμβο. Επειδή όμως δεν είναι αυτόνομος, τυπικά περιέχεται μεσα στο ίδιο φυσικό κουτί με τον υπόλοιπο κόμβο, μοιράζεται ισχύ και διαύλους με τον υπόλοιπο κόμβο και τελικά βρίσκεται υπο τον έλεγχο του κόμβου.

Για δημοφιλείς τεχνολογίες επιπέδου ζεύξης, όπως το Ethernet, η διασύνδεση ζεύξης υλοποιείται από ολοκληρωμένα κυκλώματα, τα οποια μπορούν να αγοραστούν εύκολα στην αγορά, και κοστίζουν λιγότερο από 30 ευρώ για ρυθμούς μετάδοσης από 10Mbps μέχρι 100Mbps.

Δικτύωση και εφαρμογές πολυμέσων

1.5 Δικτύωση και εφαρμογές πολυμέσων

(εικ 9-10)

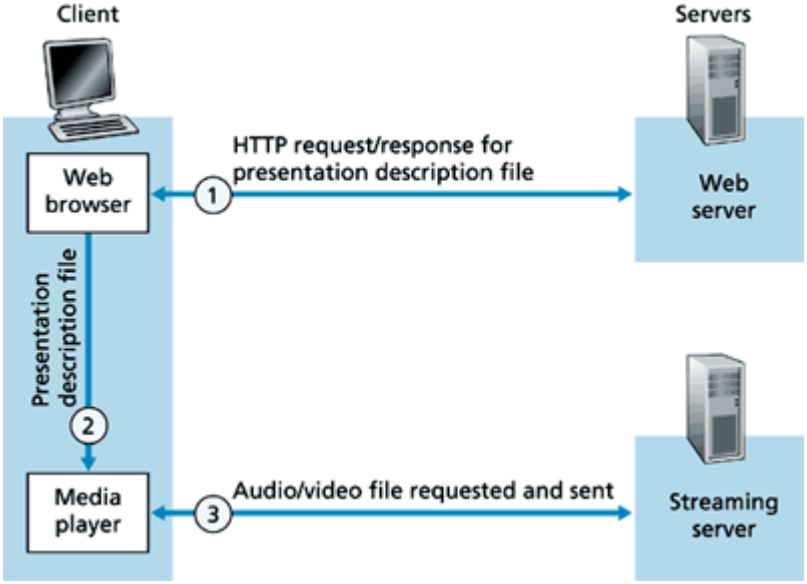
Τα τελευταία χρόνια, η ανάπτυξη και η χρήση εφαρμογών μετάδοσης ήχου, εικόνας και βίντεο στο Διαδίκτυο έχει παρουσιάσει εκρηκτική αύξηση. Συνεχώς παρουσιάζονται νέες εφαρμογές διαδικτύου, όπως εφαρμογές συνεχούς μετάδοσης μέσων, μετάδοσης συνεχούς ροής δεδομένων βίντεο (streaming) , τηλεφωνία μέσω IP, ραδιόφωνο μέσω διαδικτύου, διαδραστικά παιχνίδια (interactive gaming) , εικονικοί κόσμοι (virtual worlds) και χιλιάδες άλλα παραδείγματα. Οι απαιτήσεις αυτών των εφαρμογών, είναι πολύ υψηλές, και πολλές από αυτές τις εφαρμογές είναι ευαίσθητες στην καθυστέρηση που εισάγεται κατά την μετάδοση των δεδομένων καθώς και στις διακυμάνσεις της καθυστέρησης, αλλα μπορούν να ανεχθούν περιστασιακή απώλεια δεδομένων.

Δύο λοιπόν από αυτούς τους άξονες – θέματα χρονισμού και «ανοχή» στην απώλεια δεδομένων, είναι εξαιρετικά σημαντικοί για τις δικτυακές εφαρμογές πολυμέσων. Στις περισσότερες εφαρμογές πολυμέσων, τα πακέτα που καθυστερούν να φτάσουν από τον αποστολέα στον παραλήπτη περισσότερο από μερικές εκατοντάδες χιλιοστά του δεπτερολέπτου, είναι ουσιαστικά άχρηστα. Από την άλλη, η ανοχή στις απώλειες, μπορεί παραδείγματος χάριν να προκαλέσει μικρά προβλήματα στην αναπαραγωγή ήχου/βίντεο, αλλα συνήθως οι απώλειες αυτές αποκρύπτονται, πλήρως η μερικώς.

Μετάδοση αποθηκευμένου Ήχου και Βίντεο με συνεχή Ροή.

Σε αυτή την κατηγορία, οι clients ζητούν κατ'απαίτηση (on demand) αρχεία συμπιεσμένου ήχου και βίντεο τα οποια είναι αποθηκευμένα σε εξυπηρετητές (πχ αρχεία βίντεο όπως ταινίες, διαλέξεις, τηλεοπτικές εκπομπές). Αυτή η κατηγορία εχει τρία χαρακτηριστικά :

1) Αποθηκευμένα μέσα (stored media). Το περιεχόμενο των πολυμέσων είναι προ-καταγεγραμμένο και αποθηκευμένο σε έναν εξυπηρέτη. Ο χρήστης μπορει να διακόπτει προσωρινά την αναπαραγωγή, να μεταβαίνει στην αρχή η στο τέλος ή να ταξινομεί το περιεχόμενο των πολυμέσων. Το διάστημα που μεσολαβεί, πρέπει να διακυμαίνεται από ένα έως δέκα δεπτερόλεπτα για να είναι αποδεκτός χρόνος απόκρισης.
2) Μετάδοση με συνεχή ροή (streaming). Ο πελάτης ξεκινά την αναπαραγωγή ήχου-βίντεο λίγα δεπτερόλεπτα αφού αρχίσει να λαμβάνει το αρχείο από τον εξυπηρέτη. Ουσιαστικά, αναπαράγει ήχο/βίντεο από κάποιο σημείο του αρχείου ενώ ταυτόχρονα λαμβάνει τα επόμενα τμήματα του αρχείου αυτού. Η τεχνική αυτή αποφευγει την μεταφορά ολόκληρου του αρχείου πρίν ξεκινήσει η αναπαραγωγή. Παραδείγματα προγραμμάτων μετάδοσης με συνεχή ροή : Windows media (Microsoft), QuickTime player (Apple)



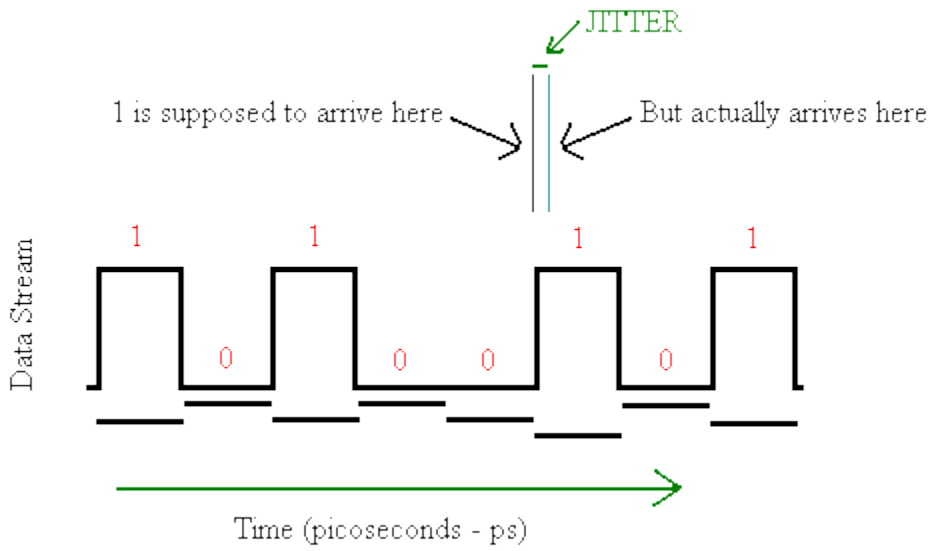
(εικ 11) : Παράδειγμα μεταφοράς με συνεχή ρόή

3) Συνεχής αναπαραγωγή (continuous playout). Με την έναρξη της αναπαραγωγής του περιεχομένου, προχωρά σύμφωνα με τις χρονικές ρυθμίσεις της αρχικής εγγραφής. Τα δεδομένα πρέπει να λαμβάνονται έγκαιρα από τον εξυπηρέτη για να είναι δυνατή η συνεχής αναπαραγωγή του περιεχομένου του πελάτη.

Προβλήματα πολυμέσων στο Σημερινό Δίαδίκτυο

Το πρωτόκολλο IP όπως προαναφέραμε, παρέχει υπηρεσίες βέλτιστης προσπάθειας για όλα τα δεδομενογράμματα που διακινεί. Ουσιαστικά, κανει ό,τι καλύτερο μπορεί για να μεταφέρει κάθε δεδομένο όσο το δυνατόν γρηγορότερα, αλλα δεν υπόσχεται τίποτα σχετικά με την καθυστέρηση ενός μεμονωμένου πακέτου από άκρο σε άκρο. Επίσης, δεν δίνεται καποια υπόσχεση σχετικά με την διακύμανση της καθυστέρησης κατά την μετάδοση πακέτων. Επειδή τα πρωτόκολλα TCP και UDP λειτουργούν πάνω στο IP, προκύπτει ότι κανένα από τα δύο αυτά πρωτόκολλα δεν δίνει εγγυήσεις σχετικά με την καθυστέρηση στις εφαρμογές που τα χρησιμοποιούν. Επειδή δεν γίνεται καμία προσπάθεια για την έγκαιρη παράδοση των πακέτων, η ανάπτυξη πολυμεσικών διαδικτυακών εφαρμογών είναι πρόκληση. Μέχρι σήμερα, η μετάδοση πολυμέσων για το διαδίκτυο ειχε σημαντική αλλα περιορισμένη επιτυχία. Για να το εξηγήσουμε καλύτερα, η μετάδοση ενός βίντεο αποθηκευμένου, με συνεχή ροή, εχει καθυστέρηση λόγω της αλληλεπίδρασης με το χρήστη πέντε έως δέκα δεπτερόλεπτα, αλλα σε περιόδους αιχμής, με αυξημένο κυκλοφοριακό φόρτο, μπορεί να μην υπάρχει ίδια απόδοση.

Οι διαδραστικές εφαρμογές θέτουν αυστηρούς περιορισμούς όσον αφορά την καθυστέρηση της μετάδοσης και τις διακυμάνσεις (packet jitter)

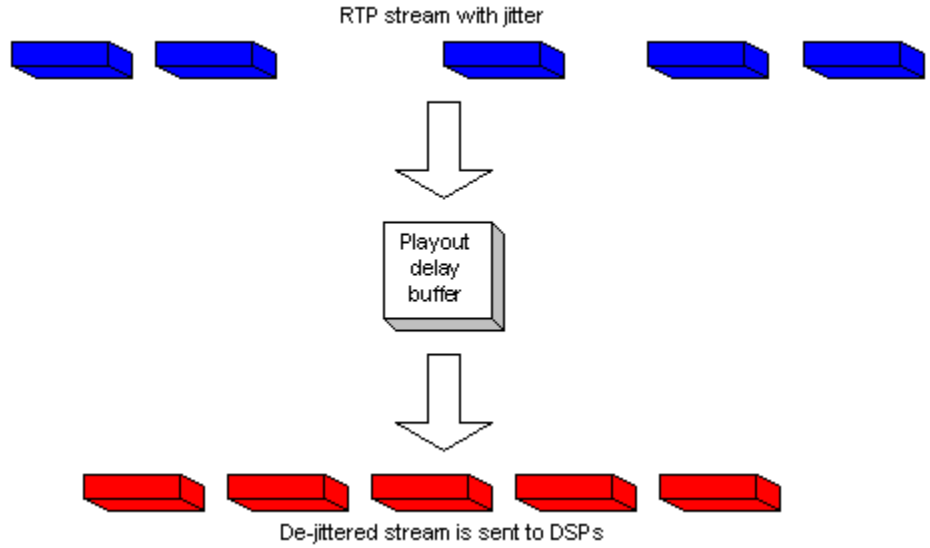


(εικ 12) : Παράδειγμα διακύμανσης καθυστέρησης

Ο όρος packet jitter εκφράζει την διακύμανση της καθυστέρησης κατά την μετάδοση πακέτων από το ίδιο ρεύμα δεδομένων. Οι εφαρμογές μετάδοσης ήχου μπορούν να λειτουργήσουν καλά σε περιοχές που έχουν μεγάλο ευρος ζώνης και κατά συνέπια οι καθυστερήσεις και η διακύμανση τους είναι αμελητέα.

Η ποιότητα όμως μπορει να μειωθεί σε απαράδεκτο βαθμό όταν η ροή δεδομένων ήχου/βίντεο χρειαστέι να περάσει από ένα κόμβο με χαμηλή ταχύτητα μετάδοσης.

Προς το παρόν, ειμαστε υποχρεωμένοι να ζούμε με τις υπηρεσίες βέλτιστης προσπάθειας. Αλλα με δεδομένο αυτό τον περιορισμό, μπορούμε να κάνουμε συγκεκριμένες επιλογές για την σχεδίαση και να χρησιμοποιήσουμε μερικές τεχνικές για να βελτιόσουμε την ποιότητα (όπως την αντιλαμβάνεται ο χρήστης) μιας πολυμεσικής διαδικτυακής εφαρμογής. Παράδειγμα πάνω σε αυτό, μπορούμε να καθυστερήσουμε την αναπαραγωγή κατά 100 χιλιοστά του δεπτερολέπτου ή περισσότερο για να κάνουμε λιγότερο εμφανή την προκαλούμενη από το διαδίκτυο διακύμανση τον καθυστερίσεων (jitter)



(εικ 13) : Παράδειγμα βελτίωσης διακυμάνσεων καθυστέρησης

Συμπίεση ήχου και βίντεο

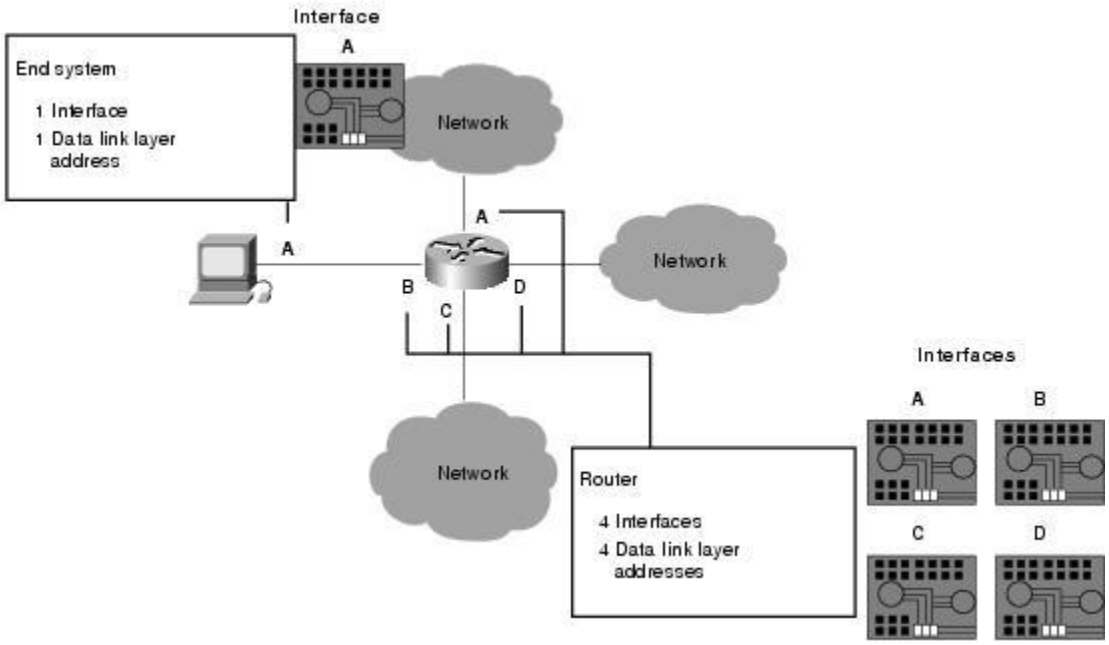
Για να μπορέσει να μεταδοθεί μέσω ενός δικτύου υπολογιστών, τοσο το βίντεο όσο και ο ήχος, πρέπει να μετατραπεί σε ψηφιακή μορφή και κατόπιν να συμπιεστεί. Τα δίκτυα υπολογιστών διακινούν bits, οποτε όλη η προς μετάδωση πληροφορία να παρουσιάζεται σαν μια σειρά από bit.

Η συμπίεση (compression) είναι πολύ σημαντική, επειδη σε μη συμπιεσμένη μορφή τόσο το βίντεο όσο και ο ήχος καταναλώνουν τεράστιες ποσότητες αποθηκευμένου χώρου και εύρους ζώνης. Η εξάλειψη της πλεονασματικής πληροφορίας από το ψηφιοποιημένο σήμα ήχου ή βίντεο μπορει να μειώσει και μάλιστα σε μεγάλο βαθμό, το ποσό των δεδομένων που χρειάζεται να αποθηκευτούν και να μεταδοθούν.

Επίπεδο δικτύου και Δρομολόγηση

1.6 Επίπεδο δικτύου και Δρομολόγηση

Το επίπεδο μεταφοράς, παρέχει υπηρεσίες επικοινωνίας ανάμεσα σε δύο διεργασίες που εκτελούνται σε δύο διαφορετικούς υπολογιστές υπηρεσίας. Για να παρέχει τις υπηρεσίες επικοινωνίας διεργασίας προς διεργασία, το επίπεδο μεταφοράς βασίζεται στην υπηρεσία επικοινωνίας υπολογιστή υπηρεσίας με υπολογιστή υπηρεσίας, που παρέχεται από το επίπεδο μεταφοράς.



(εικ 14) : Παράδειγμα επιπέδου μεταφοράς και δρομολόγησης

Ο ρόλος του επιπέδου δικτύου είναι απατηλά απλός – η μεταφορά πακέτων από τον έναν υπολογιστή *υπ*ηρεσίας αποστολής σε έναν υπολογιστή υπηρεσίας λήψης. Για αυτό πρέπει να καθοριστούν τρεις σημαντικές λειτουργίες επιπέδου δικτύου:

1) Καθορισμός διαδρομής. Το επίπεδο δικτύου πρέπει να καθορίσει την οδό ή την διαδρομή που λαμβάνεται από πακέτα, καθώς αυτά ρέουν από έναν αποστολέα σε ένα δέκτη. Οι αλγόριθμοι που υπολογίζουν αυτές τις

διαδρομές αναφέρονται σαν αλγόριθμοι δρομολόγησης. Αυτοι χωρίζονται σε δύο κατηγορίες : Τους αλγορίθμους διανύσματος απόστασης και αλγορίθμους κατάστασης συνδέσμων, οι οποίες περιέχουν σχεδόν το κάθε αλγόριθμο δρομολόγησης που χρησιμοποιείται σήμερα στο Ίντερνετ.

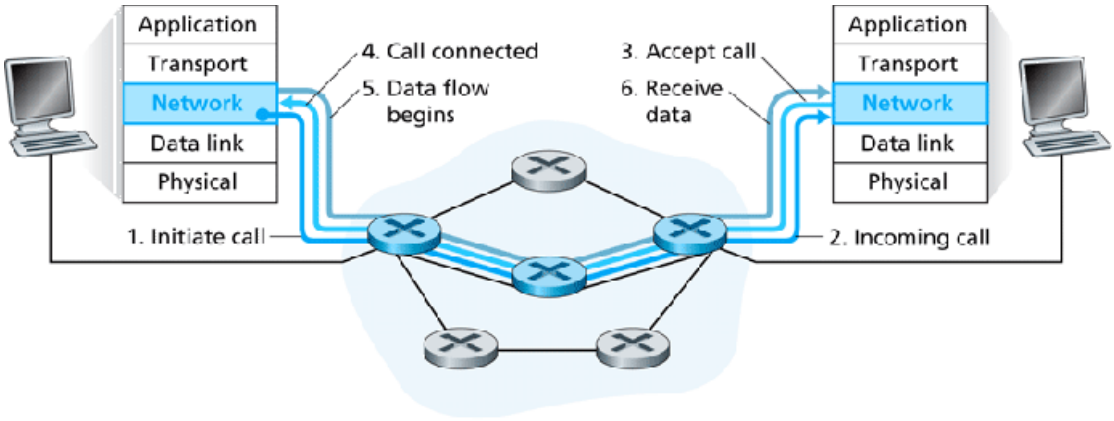
i) Οι αλγόριθμοι διανύσματος απόστασης (Distance vector algorithms) χρησιμοποιούν τον αλγόριθμο Bellman-Ford. Αυτή η διαδικασία αναθέτει έναν αριθμό, το *κόστος*, σε κάθε ένα από τις συνδέσεις μεταξύ των κόμβων σε ένα δίκτυο. Οι κόμβοι θα στέλνουν πληροφορίες από το σημείο Α στο σημείο Β μέσω της διαδρομής που έχει το μικρότερο *συνολικό κόστος* (δηλ. το αποτέλεσμα που βγαίνει από την άθροιση του κόστους μεταξύ των κόμβων που χρησιμοποιήθηκαν). Ένα πρωτόκολλο που χρησιμοποιεί αλγόριθμο διανυσμάτων απόστασης είναι το RIP, το αρχικό εσωτερικό πρωτόκολλο πύλης δικτύου του Internet. Αργότερα, αντικαταστάθηκε από το OSPF, το οποίο αποτελεί υλοποίηση ενός αλγορίθμου κατάστασης συνδέσεων.

ii) Αλγόριθμοι κατάστασης συνδέσεων (Link-state algorithms). Όταν εφαρμόζονται αλγόριθμοι κατάστασης συνδέσμων, ο κάθε κόμβος χρησιμοποιεί σαν αρχικά δεδομένα ένα χάρτη του δικτύου με την μορφή γράφου. Για να παραχθεί αυτός, κάθε κόμβος πλημμυρίζει ολόκληρο το δίκτυο με πληροφορίες σχετικά με το με ποιούς άλλους κόμβους μπορεί να συνδεθεί, εν συνεχεία κάθε κόμβος συγκεντρώνει όλες αυτές τις πληροφορίες και σχηματίζει έναν χάρτη. Χρησιμοποιώντας αυτό το χάρτη, κάθε δρομολογητής αποφασίζει ανεξάρτητα την καλύτερη διαδρομή από τον εαυτό του προς κάθε άλλο κόμβο. Ο αλγόριθμος που χρησιμοποιείται για να επιλεγεί η βέλτιστη διαδρομή, ο αλγόριθμος του Dijkstra, το κάνει αυτό δημιουργώντας μια δομή δεδομένων, ένα δέντρο, με τον τρέχοντα κόμβο σαν ρίζα του δέντρου, που περιέχει όλους τους υπόλοιπους κόμβους του δικτύου. Ξεκινάει με ένα δέντρο που περιέχει μόνο τον εαυτό του. Μετά, έναν ένα τη φορά, από το σύνολο των κόμβων που δεν έχουν προστεθεί στο δέντρο, προσθέτει τον κόμβο που έχει το μικρότερο κόστος για να φτάσει έναν γειτονικό κόμβο ο οποίος ήδη υπάρχει στο δέντρο. Αυτό συνεχίζεται μέχρις ότου όλοι οι κόμβοι να υπάρχουν στο δέντρο. Αυτό το δέντρο εξυπηρετεί στην κατασκευή του πίνακα δρομολόγησης του κάθε κόμβου, δείχνοντας το καλύτερο επόμενο βήμα (hop), για να φτάσει από τον εαυτό του σε οποιονδήποτε άλλο κόμβο στο δίκτυο.

2) Προώθηση. Όταν ένα πακέτο φτάνει στην είσοδο ενός δρομολογητή, ο δρομολογητής πρέπει να τον μεταφέρει στην κατάλληλη εξερχόμενη ζεύξη.

3) Καθορισμός κλήσης. Για τα TCP, απαιτείται μια τριμερής χειραψία πριν αρχίσει η πραγματική ροή των δεδομένων από τον αποστολέα στον δέκτη. Αυτό επιτρέπει στον αποστολέα και στον δέκτη να καθορίσουν τις απαιτούμενες πληροφορίες κατάστασης (πχ. Έναν αριθμό ακολουθίας και αρχικό μέγεθος παραθύρου ελέγχου ροής). Ορισμένες αρχιτεκτονικές επιπέδου δικτύου, παραδείγματος χάρη τα ATM, απαιτούν δρομολογητές επάνω στην επιλεγμένη διαδρομή από την προέλευση στον προορισμό να κάνουν χειραψία μεταξύ τους για να καθορίσουν την κατάσταση πριν αρχίσουν να ρέουν πακέτα δεδομένων του επιπέδου δικτύου. Σε επίπεδο δικτύου, αυτή η διαδικασία ονομάζεται καθορισμός κλήσης.

1.6,1 Μοντέλο υπηρεσίας δικτύου

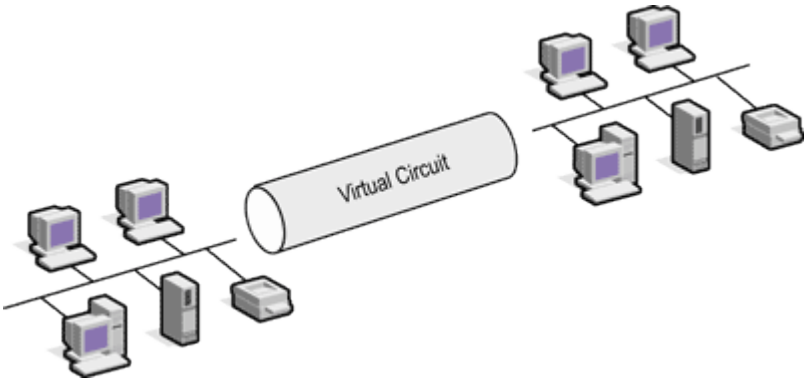


(εικ 15) : Παράδειγμα υπηρεσίας δικτύου

Το μοντέλο υπηρεσίας δικτύου ορίζει τα χαρακτηριστικά της μεταφοράς δεδομένων από άκρο σε άκρο ανάμεσα σε ένα «άκρο» του δικτύου και σε ένα άλλο, δηλαδή ανάμεσα στα τερματικά συστήματα αποστολής και λήψης. Η σημαντικότερη αφαίρεση που παρέχεται από το επίπεδο του δικτύου προς τα ανώτερα επίπεδα είναι το αν το επίπεδο δικτύου χρησιμοποιεί εικονικά δίκτυα (Virtual Circuits, VC).

Αρχικά πραγματοποιείται ο καθορισμός VC, που ο αποστολέας έρχεται σε επαφή με το επίπεδο δικτύου, καθορίζει την διεύθυνση του δέκτη και περιμένει από το δίκτυο να καθορίσει έναν VC. Το επίπεδο δικτύου καθορίζει την διαδρομή ανάμεσα σε αποστολέα και δέκτη, δηλαδή την σειρά ζεύξεων και μεταγωγέων πακέτων που θα

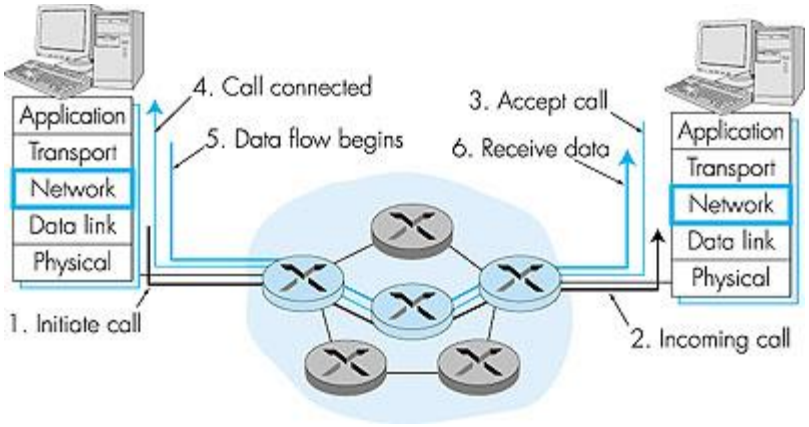
περάσουν όλα τα πακέτα του VC. Όταν καθορίζεται το VC, δεσμεύεται ένας πόρος, όπως π.χ. εύρος ζώνης. Αφού καθοριστεί το VC, αρχίζει η ροή των δεδομένων επάνω σε αυτό. Κατόπιν, ο αποστολέας ή ο δέκτης, πληροφορεί το επίπεδο δικτύου για την επιθυμία του να τερματίσει το VC.



(εικ 16) : Παράδειγμα εικονικού δικτύου

Υπάρχει διάκριση ανάμεσα στον καθορισμό του VC και στον καθορισμό της σύνδεσης στο επίπεδο μεταφοράς. Ο καθορισμός της σύνδεσης μεταφοράς περιλαμβάνει αποκλειστικά τα δύο τερματικά συστήματα, ενώ το σε ένα επίπεδο δικτύου εικονικού κυκλώματος, οι μεταγωγείς πακέτων επάνω στην διαδρομή ανάμεσα στα δύο τερματικά συστήματα εμπλέκονται στον καθορισμό του εικονικού κυκλώματος και κάθε μεταγωγέας πακέτων γνωρίζει πλήρως για όλα τα VC που περνούν από μέσα του.

Τα μηνύματα που στέλνουν τα τερματικά συστήματα μέσα στο δίκτυο για να δηλώσουν την εκκίνηση ή τον τερματισμό ενός VC, και τα μηνύματα που περνούν ανάμεσα στους μεταγωγής του VC είναι γνωστά ως μηνύματα σηματοδοσίας και τα πρωτόκολλα που χρησιμοποιούνται για την ανταλλαγή αυτών των μηνυμάτων συχνά αναφέρονται ως πρωτόκολλα σηματοδοσίας.



(εικ 17) : Επίπεδο δικτύου δεδομενογράμματος

Με ένα επίπεδο δικτύου δεδομενογράμματος, κάθε φορά που το τερματικό σύστημα θέλει να στείλει ένα πακέτο, σφραγίζει το πακέτο με την διεύθυνσή του τερματικού συστήματος προορισμού και μετά στέλνει το πακέτο μέσα στο δίκτυο. Αυτό πραγματοποιείται χωρίς τον καθορισμό VC, και οι μεταγωγείς πακέτων σε ένα τέτοιο δίκτυο δεν διατηρούν καμία πληροφορία κατάστασης για το VC. Αντί αυτού, οι μεταγωγείς πακέτων προωθούν ένα πακέτο προς το προορισμό εξετάζοντας την διεύθυνση προορισμού του πακέτου, ταξινομώντας ένα πίνακα προώθησης με την διεύθυνση προορισμού. Οι πίνακες προώθησης έχουν δυνατότητα τροποποίησης, και για αυτό το λόγο μπορεί τα πακέτα που στέλνονται να έχουν διαφορετικές διαδρομές μέσα στο δίκτυο και έτσι να φτάνουν εκτός σειράς. Το διαδίκτυο χρησιμοποιεί ένα επίπεδο δικτύου δεδομενογράμματος, που καλείται επίσης υπηρεσία βέλτιστης προσπάθειας. Με την υπηρεσία αυτή, ο χρονισμός ανάμεσα σε πακέτα δεν είναι εγγυημένο ότι θα διατηρηθεί, δεν είναι εγγυημένο ότι τα πακέτα θα φτάνουν με την σειρά με την οποία στάλθηκαν, ούτε υπάρχει εγγύηση παράδοσης των μεταδιδόμενων πακέτων.

Μια εναλλακτική ορολογία για την υπηρεσία VC την υπηρεσία δεδομενογράμματος είναι υπηρεσία συνδεσιστρεφούς επιπέδου και υπηρεσία ασυνδεσιστρεφούς επιπέδου δικτύου αντίστοιχα.

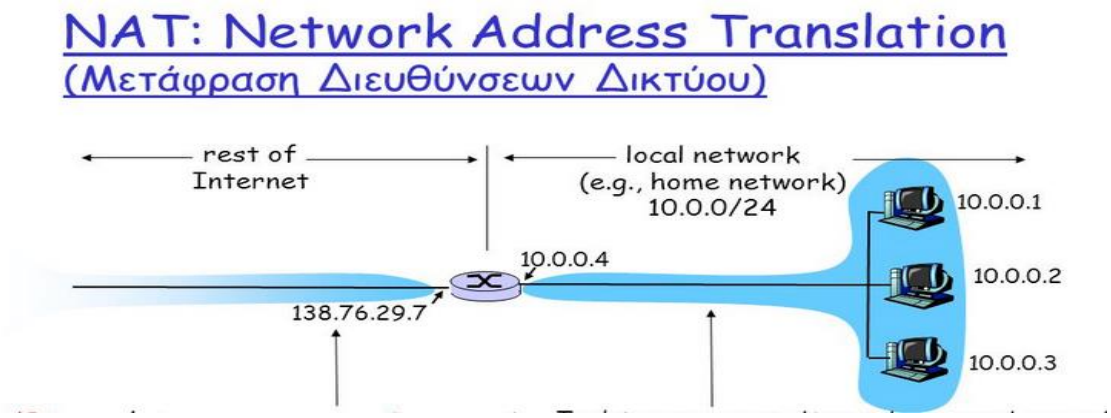
1.6,2 Μεταφραστές Διευθύνσεων Δικτύου (NAT)

Κάθε συσκευή με δυνατότητες IP, χρειάζεται μια διεύθυνση IP. Με την εξάπλωση των καλούμενων δικτύων SOHO (small office, home office) αυτό σημαίνει ότι όταν

ένα SOHO θέλει να εγκαταστήσει ένα LAN για διασύνδεση πολλαπλών υπολογιστών, πρέπει να του δοθεί μια περιοχή διευθύνσεων από τον ISP για να καλύπτονται όλοι οι υπολογιστές του SOHO. Αν το δίκτυο μεγάλωνε, θα έπρεπε να δεσμευθεί ένα μεγαλύτερο μπλοκ διευθύνσεων.

Για μια απλούστερη προσέγγιση στην κατανομή διευθύνσεων, χρησιμοποιείται το Network Address Translation, NAT, η μετάφραση διευθύνσεων Δικτύου. Ο Μεταφραστής Διευθύνσεων Δικτύου σχεδιάστηκε για απλοποίηση και διατήρηση των IP διευθύνσεων αφού αυτό που κάνει είναι να επιτρέπει σε ιδιωτικά δίκτυα που χρησιμοποιούν μη εγγεγραμμένες IP διευθύνσεις να έχουν σύνδεση με το Internet. Το σύστημα NAT λειτουργεί σε κάποιον δρομολογητή, ο οποίος συνδέει συνήθως δύο δίκτυα και μεταφράζει τις ιδιωτικές (μη μοναδικές στον παγκόσμιο ιστό) διευθύνσεις του εσωτερικού δικτύου σε νόμιμες διευθύνσεις προτού τα πακέτα προωθηθούν σε άλλο δίκτυο. Σαν μέρος αυτής της λειτουργίας το NAT μπορεί να ρυθμιστεί να κάνει γνωστή μόνο μία διεύθυνση στον έξω κόσμο για ολόκληρο το δίκτυο που συνδέει με αυτόν. Αυτό το χαρακτηριστικό παρέχει επιπλέον ασφάλεια αφού κρύβει ολόκληρο το εσωτερικό δίκτυο από το κόσμο πίσω από μία διεύθυνση.

Επιπλέον, μία επιχείρηση μπορεί να θέλει να έχει σύνδεση με το Internet χρησιμοποιώντας όμως παραπάνω από έναν παροχέα υπηρεσιών internet (ISP) για διάφορους λόγους. Το να διατηρεί κανείς σύνδεση στο internet μέσω παραπάνω του ενός ISP μπορεί να θεωρηθεί σαν ένας τρόπος αύξησης της αξιοπιστίας της σύνδεσης στο internet. Τέτοιου είδους sites με πολλαπλές συνδέσεις ονομάζονται "multi-homed". Όταν η σύνδεση από τον ένα παροχέα πέφτει τότε η εταιρία περνάει σε κάποιον άλλο διατηρώντας τη σύνδεσή της έτσι πάντα. Ακόμα ένα πλεονέκτημα αυτού του σχήματος είναι το ότι η επιχείρηση μπορεί να διανείμει το φορτίο της σε διαφορετικές συνδέσεις. Για επιχειρήσεις μάλιστα που εκτείνονται σε μεγάλη γεωγραφική περιοχή ένα τέτοιο σχήμα θα σήμαινε και καλύτερη διαδικασία δρομολόγησης.



(εικ 18) : Παράδειγμα δρομολογητή NAT σε οικιακό δίκτυο

Η παραπάνω εικόνα δείχνει την λειτουργία ενός δρομολογητή με δυνατότητες NAT. Εχει μια διασύνδεση που είναι τμήμα του οικιακού δικτύου δεξιά της εικόνας. Η διευθυνσιοδότηση μεσα στο οικιακό δίκτυο είναι ακριβώς όπως βλέπουμε παραπάνω : και οι τέσσερις διασυνδέσεις μέσα στο οικιακό δίκτυο έχουν την ίδια διεύθυνση δικτύου 10.0.0/24. Ο δρομολογητής με δυνατότητες NAT δεν εκτελεί ένα πρωτόκολλο δρομολόγησης δια-ΑΣ με τον συνδεδεμένο δρομολογητή ISP. Ο δρομολογητής NAT συμπεριφέρεται στον εξωτερικό κόσμο σαν ΜΙΑ μόνο συσκευή με ΜΙΑ μόνο διεύθυνση IP. Όλη η κίνηση που φεύγει από τον οικιακό δρομολογητή για το ευρύτερο δίκτυο εχει μια διεύθυνση προέλευσης IP 138.76.29.7. Στην ουσία, ο δρομολογητής με δυνατότητες NAT κρύβει τις λεπτομέρειες του οικιακού δικτύου από τον έξω κόσμο.

Τώρα, αν όλα τα δεδομενογράμματα που φτάνουν στον δρομολογητή NAT από το WAN (δίκτυο ευρείας περιοχής) έχουν την ίδια διεύθυνση προορισμού IP, πως ξέρει ο δρομολογητής τον εσωτερικό υπολογιστή υπηρεσίας στον οποίο πρέπει να προωθήσει το δεδομένο δεδομενόγραμμα? Το τέχνασμα εδώ είναι να χρησιμοποιηθεί ένας πίνακας μετάφρασης NAT στον δρομολογητή NAT και να περιλάβουμε αριθμούς θυρών και διευθύνσεις IP στις καταχωρήσεις του πίνακα.

Επίπεδο μεταφοράς

1.7 Επίπεδο μεταφοράς

Το επίπεδο μεταφοράς, το οποίο βρίσκεται ανάμεσα στο επίπεδο εφαρμογής και το επίπεδο του δικτύου είναι το κεντρικό κομμάτι της αρχιτεκτονικής οργάνωσης σε διαδοχικά επίπεδα του δικτύου.

Ένα πρωτόκολλο επιπέδου μεταφοράς παρέχει την δυνατότητα λογικής επικοινωνίας ανάμεσα σε διεργασίες εφαρμογών που εκτελούνται σε διαφορετικούς υπολογιστές υπηρεσίας. Με τον όρο λογική επικοινωνία εννοούμε ότι οι υπολογιστές υπηρεσίας λειτουργούν σαν να είναι απευθείας συνδεδεμένοι μεταξύ τους, ακόμα και αν βρίσκονται από το ένα στο άλλο άκρο του πλανήτη. Η εφαρμογή χρησιμοποιεί την λογική επικοινωνία χωρίς να την απασχολεί πχ. Το φυσικό μέσο (οπτική ίνα, ασύρματη επικοινωνία, χαλκός) που χρησιμοποιείται για την μεταφορά των μηνυμάτων.

Τα πρωτόκολλα επιπέδου μεταφοράς υλοποιούνται στα τερματικά συστήματα, αλλά όχι σε δρομολογητές του δικτύου.

Ενώ ένα πρωτόκολλο μεταφοράς παρέχει λογική επικοινωνία ανάμεσα σε διεργασίες που εκτελούνται σε διαφορετικούς υπολογιστές υπηρεσίας, ένα πρωτόκολλο επιπέδου δικτύου παρέχει λογική επικοινωνία ανάμεσα σε υπολογιστές υπηρεσίας.

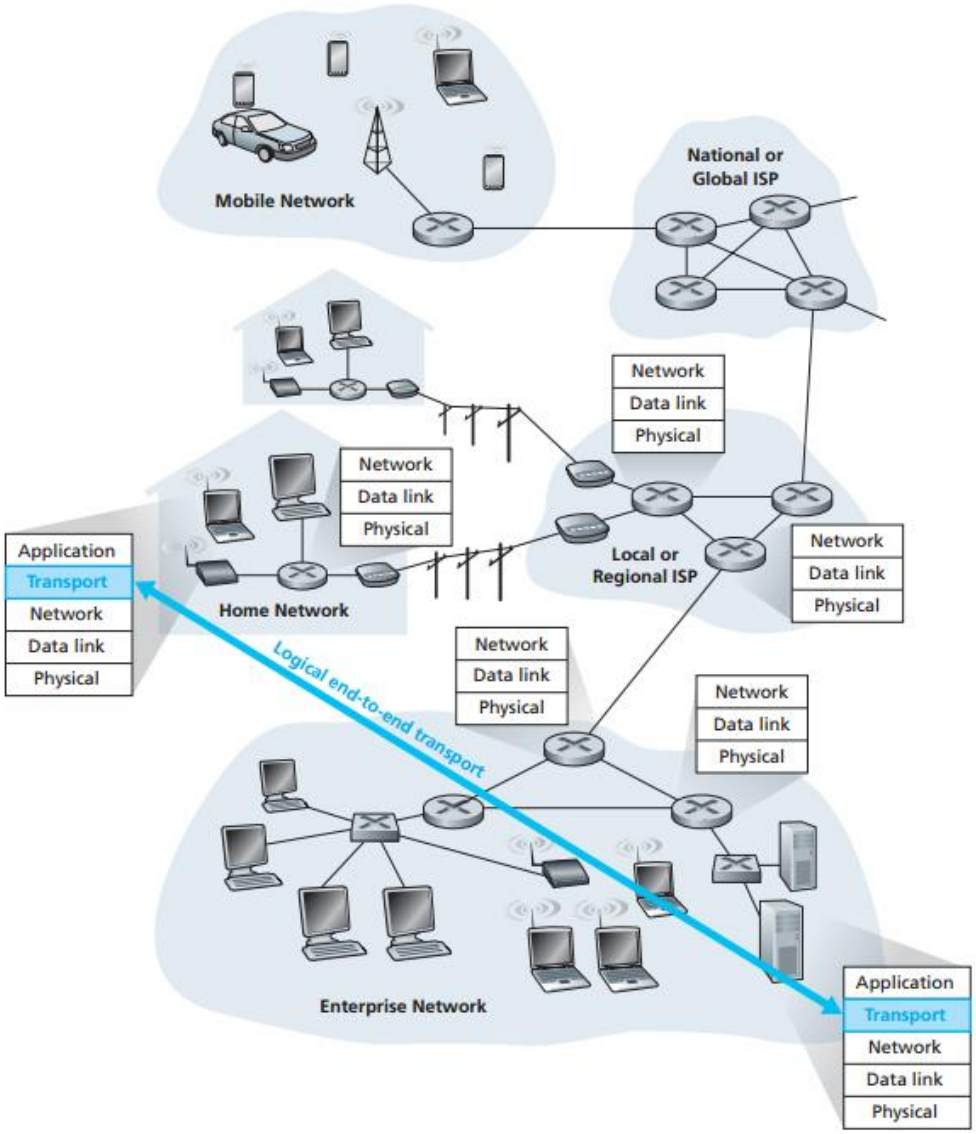
Το διαδίκτυο, και γενικά ένα δίκτυο TCP/IP, κάνει διαθέσιμα δυο διαφορετικά πρωτόκολλα επιπέδου μεταφοράς στο επίπεδο εφαρμογής, το UDP και το TCP. Στην προκειμένη θα αναφερθούμε μόνο στο TCP καθώς είναι το πρωτόκολλο που χρησιμοποιήθηκε για την κατασκευή της εφαρμογής.

Το πρωτόκολλο επιπέδου δικτύου (IP) παρέχει λογική επικοινωνία ανάμεσα σε υπολογιστές υπηρεσίας. Το μοντέλο υπηρεσίας IP είναι μια υπηρεσία παράδοσης βέλτιστης προσπάθειας. Αυτό σημαίνει ότι το IP κάνει την “καλύτερη προσπάθεια» για να παραδώσει τμήματα στις συσκευές που επικοινωνούν, αλλά δεν δίνει καμία εγγύηση για την ακεραιότητα των δεδομένων, κάτι που την καθιστά αναξιόπιστη υπηρεσία.

Με την χρήση του πρωτοκόλλου TCP, το οποίο όπως είδαμε συνοπτικά στην αρχή, παρέχει αξιόπιστη μεταφορά δεδομένων. Χρησιμοποιώντας έλεγχο ροής, αριθμούς ακολουθίας, γνωστοποιήσεις και χρονομετρητές, το TCP διαβεβαιώνει ότι παραδίδονται τα δεδομένα σωστά και με σωστή σειρά. Το TCP επίσης, παρέχει έλεγχο συμφόρησης, μια υπηρεσία που απαγορεύει σε οποιαδήποτε σύνδεση TCP να πλημμυρίσει τις συνδέσεις και τους μεταγωγείς ανάμεσα σε επικοινωνούντες

υπολογιστές υπηρεσίας με υπερβολική ποσότητα κίνησης. Καταρχάς, το TCP επιτρέπει σε συνδέσεις TCP που διασχίζουν μια ζεύξη δικτύου με συμφόρηση, να μοιράζονται εξ ίσου το εύρος ζώνης της ζεύξης.

Ένα παράδειγμα λογικής μεταφοράς είναι το παρακάτω :



(εικ 19) : Δίκτυο Λογικής μεταφοράς

Επίπεδο Εφαρμογής

1.8 Επίπεδο Εφαρμογής

(εικ 20) : Εικονίδια εφαρμογών

Οι εφαρμογές δικτύου είναι ο λόγος ύπαρξης ενός δικτύου υπολογιστών. Στην σημερινή εποχή κατακλυζόμαστε από όλο ένα και περισσότερο νέες εφαρμογές με σκοπό την ευχρηστία, την απλότητα, την διευκόλυνση της καθημερινότητας και ένα σορό σκοπούς, κάτι που οδήγησε στην δημιουργία έξυπνων και εντυπωσιακών εφαρμογών. Πως όμως όλα αυτά υλοποιούνται στην πράξη ?

1.8,1 Αρχές Πρωτοκόλλων Επιπέδου Εφαρμογής

Οι εφαρμογές του δικτύου παρουσιάζουν πολλές διαφορές μεταξύ τους και έχουν πολλά αλληλεπιδρώντα συστατικά, αλλά το λογισμικό παραμένει πάντα ο πυρήνας τους. Με βάση την ορολογία των λειτουργικών συστημάτων, δεν επικοινωνούν πραγματικά κομμάτια λογισμικού, αλλά διεργασίες (progresses). Όταν οι επικοινωνούσες διεργασίες εκτελούνται στο ίδιο τερματικό σύστημα, επικοινωνούν

μεταξύ τους χρησιμοποιώντας διαδιεργασιακή επικοινωνία. Διεργασίες που έχουν διαφορετικά τερματικά συστήματα, επικοινωνούν με την ανταλλαγή μηνυμάτων πάνω στο δίκτυο υπολογιστών.

Είναι σημαντικό να ξεχωρίσουμε τις εφαρμογές δικτύου και τα πρωτόκολλα επιπέδου εφαρμογής. Ένα πρωτόκολλο επιπέδου εφαρμογής αποτελεί ένα κομμάτι μιας εφαρμογής, και ορίζει πως οι διεργασίες μιας εφαρμογής, που εκτελούνται σε διαφορετικά τερματικά συστήματα, παίρνουν μηνύματα η μια με στην άλλη. Πιο συγκεκριμένα, το πρωτόκολλο επιπέδου εφαρμογής ορίζει:

- Τους τύπους των μηνυμάτων που ανταλλάζονται (πχ αιτήσεων, αποκρίσεων)
- Την σύνταξη των διαφόρων τύπων μηνυμάτων
- Την σημασία των πληροφοριών μέσα στα πεδία
- Κανόνες για το πότε και το πώς μια διεργασία στέλνει μηνύματα και αποκρίνεται σε μηνύματα

Ορισμένα πρωτόκολλα επιπέδου εφαρμογής καθορίζονται σε RFC, δηλαδή στον δημόσιο τομέα. Άλλα πρωτόκολλα επιπέδου εφαρμογής, δεν διατίθενται στον δημόσιο τομέα (πχ. Πολλά προϊόντα τηλεφωνίας μέσω διαδικτύου)

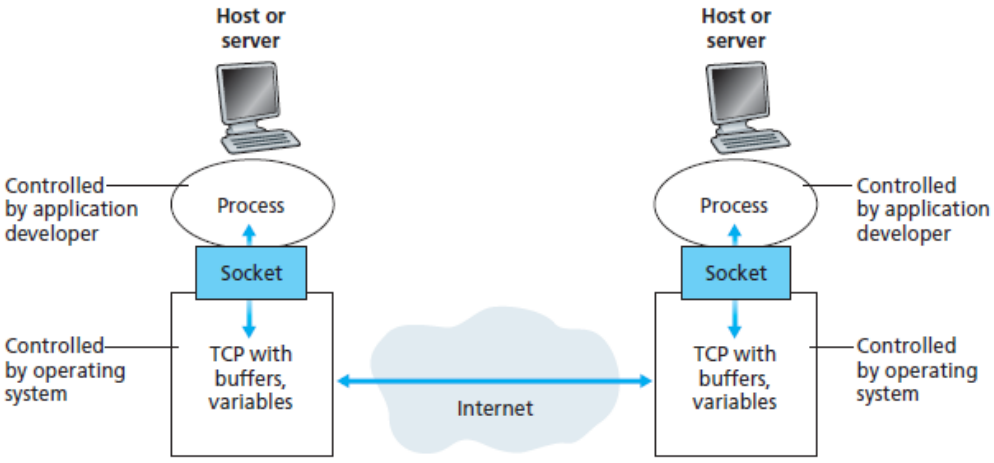
1.8,2 Πλευρές Πελάτη και Εξυπηρέτη

Μια εφαρμογή δικτύου, χωρίζεται σε δύο τμήματα. Ένα τμήμα αποτελεί τον πελάτη, και έναν τον εξυπηρέτη. Και οι δύο πλευρές όμως έχουν τερματικά συστήματα που επικοινωνούν μεταξύ τους. Ένα παράδειγμα πάνω σε αυτό, είναι ότι στο e-mail, ο εξυπηρέτης αποστολής υλοποιεί από την πλευρά πελάτη του SMTP και ο εξυπηρέτης λήψης από την πλευρά εξυπηρέτη του SMTP. Σαν ένα άλλο παράδειγμα, αν θεωρήσουμε το FTP, που χρησιμοποιείται για μεταφορά αρχείων μεταξύ δυο υπολογιστών υπηρεσίας, όταν υπάρχει μια σύνδεση FTP ανάμεσα σε δύο υπολογιστές υπηρεσίας, τότε και οι δύο υπολογιστές μπορούν να μεταφέρουν ένα αρχείο στον άλλο υπολογιστή κατά την διάρκεια της σύνδεσης. Η διαφορά εδώ, είναι ότι ο υπολογιστής υπηρεσίας που εκκινεί την σύνοδο ονομάζεται πελάτης.

1.8,3 Διεργασίες και διευθυνσιοδότηση

Δύο διεργασίες λοιπόν, επικοινωνούν μεταξύ τους επάνω σε ένα δίκτυο λαμβάνοντας και στέλνοντας μηνύματα. Μια διεργασία, λαμβάνει και δέχεται μηνύματα από το δίκτυο, μέσω της **socket (πόρτας)** της. Όταν το μήνυμα φτάσει στον υπολογιστή υπηρεσίας προορισμού, περνά μέσα από την πόρτα της

διεργασίας λήψης και η διεργασία λήψης δρα στο μήνυμα. Στην παρακάτω εικόνα, φαίνεται η επικοινωνία socket ανάμεσα σε δύο διεργασίες, που επικοινωνούν μέσω διαδικτύου. Το πρωτόκολλο που έχει χρησιμοποιηθεί είναι το TCP, και όπως φαίνεται σε αυτή την εικόνα, μια socket είναι η διασύνδεση ανάμεσα στο επίπεδο εφαρμογής και στο επίπεδο μεταφοράς μέσα σε έναν υπολογιστή υπηρεσίας. Αναφέρεται επίσης και ως API (application programmer's interface) ανάμεσα στην εφαρμογή και το διαδίκτυο. Ο προγραμματιστής εφαρμογής έχει τον έλεγχο στο επίπεδο εφαρμογής της socket, αλλά έχει μικρό έλεγχο στο επίπεδο μεταφοράς socket. Ο μόνος έλεγχος που έχει ένας προγραμματιστής εφαρμογής στην πλευρά του επιπέδου μεταφοράς είναι η επιλογή πρωτοκόλλου μεταφοράς και ίσως κάποια μερική παραμετροποίηση του επιπέδου μεταφοράς. Όταν ο προγραμματιστής της εφαρμογής επιλέξει ένα πρωτόκολλο μεταφοράς, η εφαρμογή δημιουργείται πάνω σε αυτό.



(εικ 21) : Διεργασίες μιας εφαρμογής, socket και πρωτοκόλλου μεταφοράς

Για να στείλει μια διεργασία μήνυμα από έναν υπολογιστή υπηρεσίας σε μια διεργασία σε άλλον υπολογιστή υπηρεσίας, η διεργασία αποστολής πρέπει να δηλώσει την διεργασία λήψης. Σε αυτό έρχεται η διεύθυνση IP. Μια διεύθυνση IP είναι μια ποσότητα 32-bit που προσδιορίζει μοναδικά τον υπολογιστή υπηρεσίας, και εφόσον ο υπολογιστής αυτός είναι συνδεδεμένος στο δημόσιο διαδίκτυο, είναι παγκόσμια και μοναδική.

Εκτός του ότι λοιπόν μια εφαρμογή αποστολής πρέπει να γνωρίζει την διεύθυνση του υπολογιστή υπηρεσίας, πρέπει να παρέχει πληροφορίες που θα επιτρέπουν στον υπολογιστή υπηρεσίας λήψης να κατευθύνει το μήνυμα στην κατάλληλη διεργασία μέσα σε αυτόν τον υπολογιστή υπηρεσίας. Ένας **αριθμός θύρας** είναι

απαραίτητος για αυτό. Όταν λοιπόν κάποιος δημιουργεί μια εφαρμογή δικτύου, στην εφαρμογή πρέπει να οριστεί ένας νέος αριθμός θύρας.

Μεταξύ του χρήστη, και της εφαρμογής δικτύου, υπάρχει η ιδέα του πράκτορα χρήστη (user agent). Στο παραπάνω παράδειγμα εφαρμογής ηλεκτρονικού ταχυδρομείου, ο πράκτορας χρήστης είναι ένας «αναγνώστης ταχυδρομείου» που επιτρέπει σε ένα χρήστη να συνθέσει και να διαβάσει μηνύματα.

1.8,4 Απαραίτητες Υπηρεσίες Εφαρμογής

Όταν αναπτύσσουμε μια εφαρμογή, πρέπει να επιλέξουμε ένα επίπεδο μεταφοράς. Πρέπει να γίνει πολύ προσεκτική επιλογή στο επίπεδο μεταφοράς, ώστε το πρωτόκολλο να ταιριάζει περισσότερο με τις ανάγκες της εφαρμογής. Μπορεί να γίνει μια γενική κατάταξη των απαιτήσεων μιας εφαρμογής σε τρείς διαστάσεις :

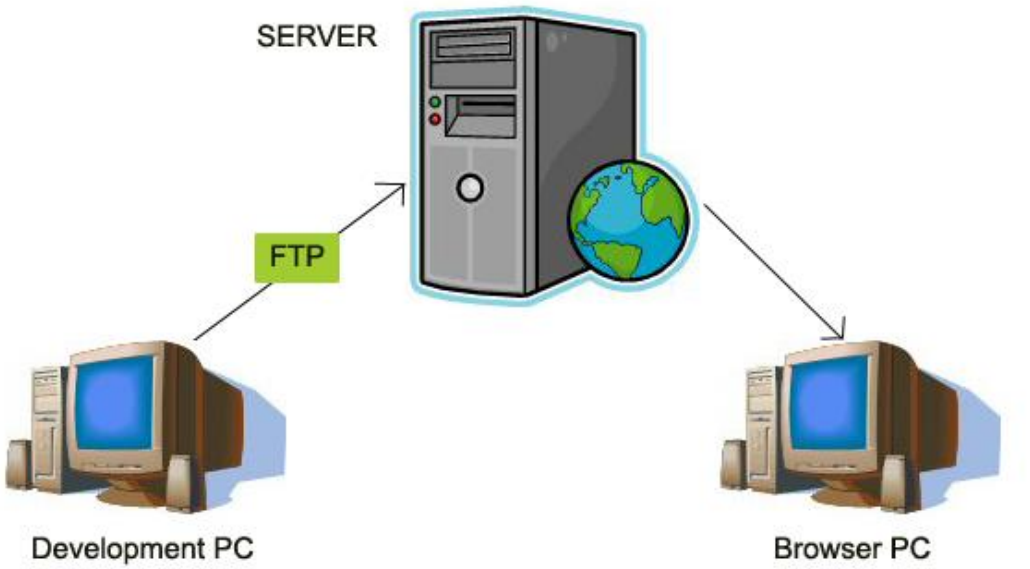
Απώλεια δεδομένων, εύρος ζώνης και χρονισμός.

- Αξιόπιστη μεταφορά δεδομένων : Υπάρχουν εφαρμογές με ανοχές η όχι στις απώλειες (Πχ. Δεν μπορούμε να έχουμε απώλεια δεδομένων αρχείων μιας χρηματοοικονομικής εφαρμογής, γιατί μπορεί να έχει καταστροφικές συνέπειες!) Εφαρμογές με ανοχές στις απώλειες είναι όπως είδαμε παραπάνω εφαρμογές πολυμέσων, που έχουμε απώλεια σε κάποια ποσότητα και παραμορφώνει την ποιότητα. Τα αποτελέσματα μιας απώλειας στην ποιότητα εφαρμογής, και η ποσότητα της ανεκτής απώλειας εξαρτάται κατά πολύ από την εφαρμογή και το σχήμα κωδικοποίησης της.
- Εύρος ζώνης : Υπάρχουν εφαρμογές που πρέπει να μεταφέρουν δεδομένα με έναν ορισμένο τρόπο για να είναι αποτελεσματικές, όπως συγκεκριμένο ρυθμό λήψης. Αν το εύρος ζώνης δεν είναι διαθέσιμο, τότε η εφαρμογή πρέπει να κωδικοποιεί με έναν διαφορετικό ρυθμό και να δέχεται αρκετό εύρος ζώνης ώστε να υποστηρίζει αυτό τον ρυθμό κωδικοποίησης. Υπάρχουν εφαρμογές ευαίσθητες στο εύρος ζώνης και ελαστικές εφαρμογές που χρησιμοποιούν όσο ευρος ζώνης είναι διαθέσιμο.
- Χρονισμός. Διαδραστικές εφαρμογές, εφαρμογές πραγματικού χρόνου, παιχνίδια πολλών παικτών απαιτούν στενούς χρονικούς περιορισμούς παράδοσης δεδομένων για να είναι αποτελεσματικές.

Υπηρεσίες που παρέχονται από το TCP πρωτόκολλο.

Το TCP πρωτόκολλο και η λειτουργίες που παρέχει αναφέρθηκε στην σελίδα

1.8,5 Μεταφορά αρχείων Μέσω FTP.



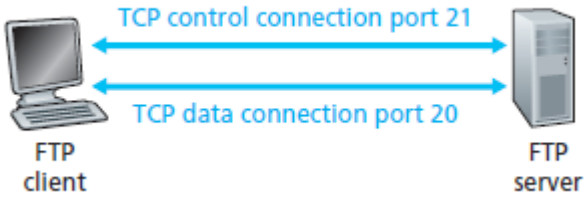
(εικ 22) : FTP server

Σε μία σύνοδο FTP, ο χρήστης σε έναν υπολογιστή υπηρεσίας που θέλει να μεταφέρει αρχεία προς η από ένα απομακρυσμένο υπολογιστή, πρέπει να ακολουθήσει μια διαδικασία.

Καταρχάς, πρέπει να δώσει ένα όνομα χρήστη και έναν κωδικό πρόσβασης. Αφου δώσει αυτές τις πληροφορίες, ο εξυπηρέτης εξουσιοδοτεί τον χρήστη, ο χρήστης αντιγράφει ένα η περισσότερα αρχεία που είναι αποθηκευμένα στο τοπικό σύστημα αρχείων ή το αντίστροφο. Το FTP χρησιμοποιεί δύο παράλληλες συνδέσεις TCP για την μεταφορά ενός αρχείου, μια σύνδεση ελέγχου και μια σύνδεση δεδομένων. Η σύνδεση ελέγχου χρησιμοποιείται για αποστολή πληροφοριών ελέγχου όπως όνομα χρήστη, κωδικό πρόσβασης, εντολές για τοποθέτηση και λήψη αρχείων. Το FTP λέγεται ότι στέλνει τις πληροφορίες ελέγχου εξωζωνικά (έναντι του HTTP που στέλνει πληροφορίες ελέγχου ενδοζωνικά) .

Όταν ένας χρήστης εκκινεί μια σύνοδο FTP, η πλευρά του χρήστη (πελάτη FTP) εκκινεί πρώτα μια σύνδεση ελέγχου TCP με την πλευρά εξυπηρέτη στην θύρα του εξυπηρέτη με αριθμό 21. Η πλευρά πελάτη FTP στέλνει όνομα χρήστη και κωδικό στην σύνδεση ελέγχου της, όπως και εντολές για αλλαγή του απομακρυσμένου καταλόγου. Όταν η πλευρά εξυπηρέτη δεχθεί επάνω στην σύνδεση ελέγχου μια εντολή από μια μεταφορά αρχείου, η πλευρά εξυπηρέτη εκκινεί μια σύνδεση δεδομένων TCP προς την πλευρά πελάτη. Το FTP

στέλνει ακριβώς ένα αρχείο επάνω στην σύνδεση δεδομένων TCP και μετά κλείνει την σύνδεση δεδομένων. Αν τώρα, στην διάρκεια μιας συνόδου, ο χρήστης θέλει να μεταφέρει ένα αρχείο διαφορετικό, το FTP ανοίγει μια άλλη σύνδεση δεδομένων. Αυτό μας εξηγεί ότι ενώ η σύνδεση ελέγχου παραμένει ανοιχτή κατ' απαίτηση χρήστη, οι συνδέσεις δεδομένων δεν είναι παραμένουσες.



(εικ 23) : Tcp σύνδεση ελέγχου

Εντολές και αποκρίσεις FTP.

Οι εντολές, από τον πελάτη στον εξυπηρέτη, και οι αποκρίσεις από τον εξυπηρέτη στον πελάτη στέλνονται μέσα της σύνδεσης ελέγχου με μορφή ASCII ή επτά bit. Με αυτήν την λειτουργία, οι εντολές FTP μπορούν να διαβαστούν από ανθρώπους. Ορισμένες από τις πιο συνηθισμένες εντολές είναι :

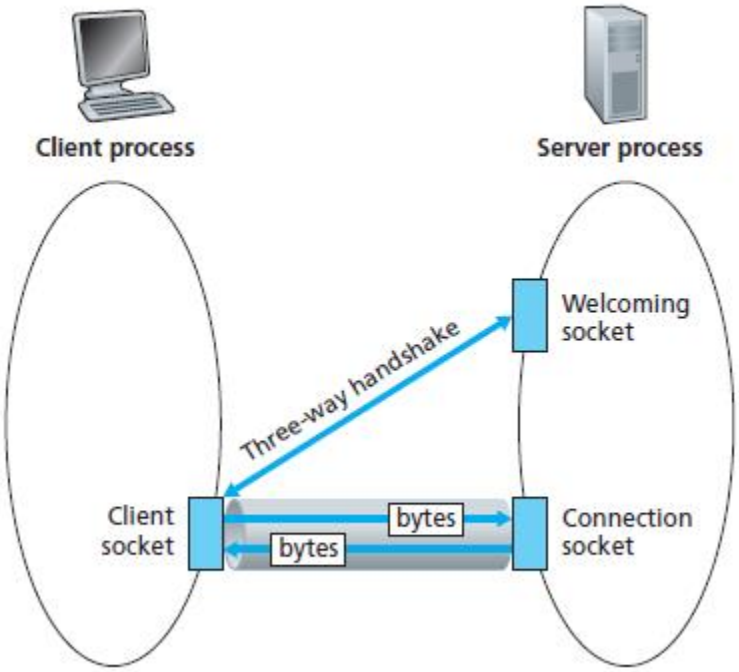
- USER username: Στέλνει ταυτότητα χρήστη στον εξυπηρέτη.
- PASS password: Στέλνει κωδικό πρόσβασης χρήστη
- LIST: Ζητά από τον εξυπηρέτη να στείλει πίσω μια λίστα με όλα τα αρχεία στον τρέχοντα κατάλογο. Η λίστα στέλνεται σε μια νέα μη παραμένουσα σύνδεση δεδομένων.
- RETR filename: Επαναφέρει ένα αρχείο από τον τρέχοντα κατάλογο του απομακρισμένου υπολογιστή υπηρεσίας. Εκκινεί τον απομακρυσμένο υπολογιστή υπηρεσίας, ώστε να εκκινήσει μια σύνδεση δεδομένων και να στείλει το αιτούμενο αρχείο επάνω στην σύνδεση δεδομένων.
- STOR filename: Πραγματοποιεί αποθήκευση ενός αρχείου στον τρέχοντα κατάλογο του απομακρυσμένου υπολογιστή.

Υπάρχει αντιστοιχία ένα προς ένα ανάμεσα στην εντολή που εκδίδει ο χρήστης και στην εντολή FTP που στέλνεται επάνω στην σύνδεση ελέγχου.

1.8,6 Προγραμματισμός Socket με TCP

Υπάρχουν δύο είδη εφαρμογών πελάτη/εξυπηρέτη. Ένα είδος εφαρμογής, είναι η υλοποίηση ενώς προτοκόλλου που ορίζεται σε ένα RFC. Το RFC, ή αλλιως Request For Comments, είναι ναι μια σειρά κειμένων (εγγράφων) και σημειώσεων για την τεχνική και την οργάνωση που διέπουν το Για μια τέτοια υλοποίηση, απαιτείται τα προγράμματα πελάτη/εξυπηρέτη να συμμορφώνονται με τους κανόνες που υπαγορεύονται από το RFC. Όταν ένα πρόγραμμα πελάτη υλοποιεί ένα πρωτόκολλο που έχει ορισθεί με RFC, πρέπει να χρησιμοποιεί αριθμό θύρας που σχετίζεται με το πρωτόκολλο.

Όπως αναφέραμε παραπάνω, οι διεργασίες που εκτελούνται σε διάφορους υπολογιστές επικοινωνούν μεταξύ τους στελνοντας μηνύματα σε sockets.



(εικ 24) : Socket πελάτη, Socket υποδοχής, Socket σύνδεσης και τριμερής χειραψία

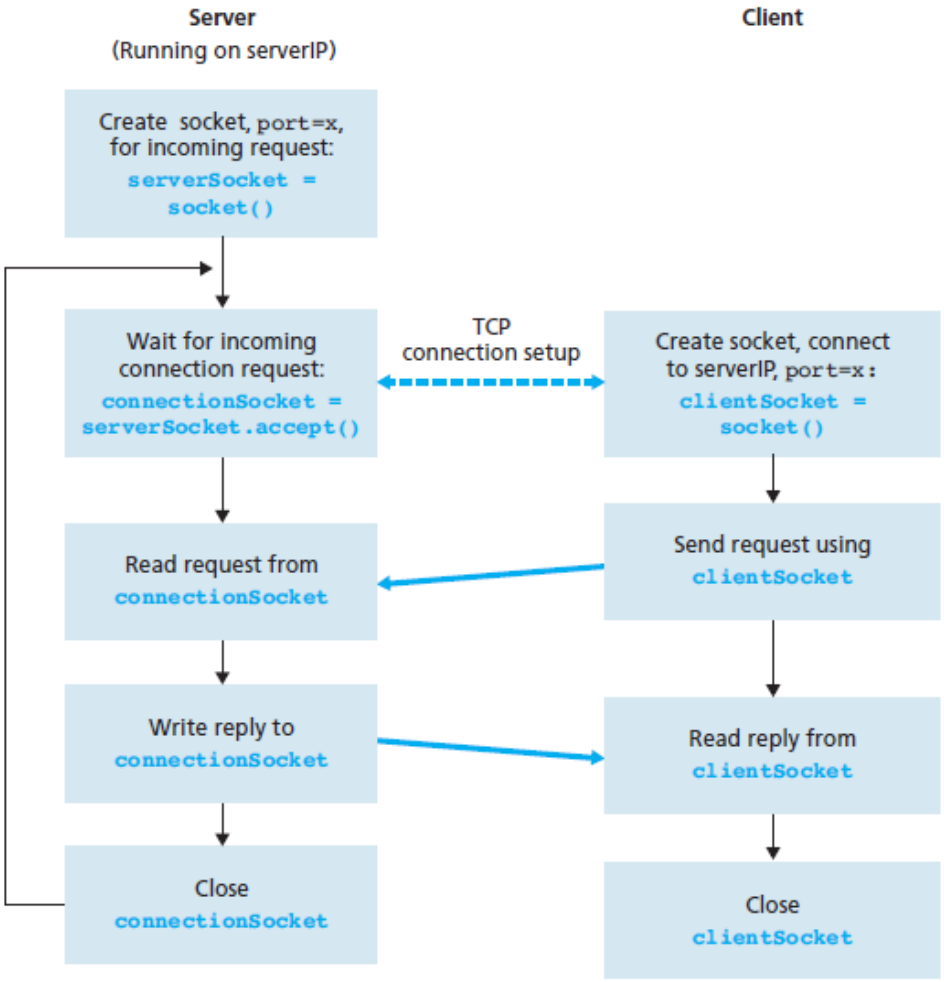
Ο πελάτης, κάνει έναρξη της επαφής με τον εξυπηρέτη. Για να μπορέσει ο εξυπηρέτης να αρχίσει την μεταφορά δεδομένων με τον πελάτη, πρέπει να είναι έτοιμος. Αυτό προϋποθέτει να εκτελείται σαν μια διεργασία, πριν ο πελάτης προσπαθήσει να ξεκινήσει την σύνδεση. Με εκτελούμενη την διεργασία εξυπηρέτη, η διεργασία πελάτη μπορεί να εκκινήσει μια σύνδεση TCP προς τον εξυπηρέτη. Το πρόγραμμα πελάτη για αυτό, δημιουργεί μια socket. Όταν ο πελάτης δημιουργήσει την socket του, καθορίζει την διεύθυνση IP του εξυπηρέτη και τον αριθμό θύρας

που θα γίνει η «χειραψία» που καθορίζει την σύνδεση TCP με τον εξυπηρέτη. Η τριμερής χειραψία είναι τελείως διαφανής στα προγράμματα πελάτη και εξυπηρέτη. Όταν ο εξυπηρέτης ξεκινά να συνδεθεί μέσω της χειραψίας με τον πελάτη, δημιουργεί μια νέα socket, που είναι αποκλειστική για τον πελάτη. Στο τέλος της φάσης της χειραψίας, υπάρχει μια σύνδεση TCP ανάμεσα στο socket του πελάτη και στην νέα socket του εξυπηρέτη. Η διεργασία πελάτη πλέον μπορεί να στείλει αυθαίρετα bytes μέσα σε αυτήν την socket. Το TCP, παρέχει σαν εγγύηση ότι τα bytes θα φτάσουν με την σειρά που στέλνονται. Η διεργασία πελάτη μπορεί να λαμβάνει και εκείνη bytes. Επειδή οι sockets παίζουν τον κεντρικό ρόλο σε τέτοιου τύπου εφαρμογές, ο προγραμματισμός τους ονομάζεται προγραμματισμός socket.

Κάτι άλλο που πρέπει να αναφέρουμε είναι ι ιδέα του ρεύματος (stream). Ένα ρεύμα, είναι μια αλληλουχία χαρακτήρων που ρέουν από μέσα προς τα έξω σε μια διεργασία. Χωρίζεται σε ρεύμα εισόδου και ρεύμα εξόδου για αυτή την διεργασία. Αν το ρεύμα είναι ρεύμα εισόδου, τότε προσαρτάται σε κάποια πηγή εισόδου, όπως μια πρότυπη είσοδος (πληκτρολόγιο) είτε μια socket από το διαδίκτυο. Αν το ρεύμα είναι ρεύμα εξόδου, τότε προσαρτάται σε κάποια πηγή εξόδου όπως πρότυπη έξοδο (οθόνη) είτε μια socket από το διαδίκτυο.

1.8,7 Παράδειγμα Εφαρμογής Server/Client σε Java για εξήγηση εννοιών

- Ενας πελάτης διαβάζει μια γραμμή από την πρότυπη είσοδο του (πληκτρολόγιο) και στέλνει την γραμμή έξω από την socket προς τον εξυπηρέτη
- Ο εξυπηρέτης διαβάζει μια γραμμή από την socket σύνδεσης
- Ο εξυπηρέτης μετατρέπει την γραμμή σε κεφαλαία
- Ο εξυπηρέτης στέλνει την τροποποιημένη γραμμή έξω από την socket σύνδεσης προς τον πελάτη
- Ο πελάτης διαβάζει την τροποποιημένη γραμμή από την socket του και την εκτυπώνει στην πρότυπη έξοδο του (οθόνη)



(εικ 25) : Παράδειγμα εφαρμογής πελάτη/εξυπηρέτη με TCP Socket Programming

TCPClient.java

```
1  import java.io.*;
2  import java.net.*;
3
4  class TCPClient
5  {
6   public static void main(String argv[]) throws Exception
7   {
8    String sentence;
9    String modifiedSentence;
10   BufferedReader inFromUser = new BufferedReader( new InputStreamReader(System.in));
11   Socket clientSocket = new Socket("localhost", 6789);
12   DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
13   BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
14   sentence = inFromUser.readLine();
15   outToServer.writeBytes(sentence + '\n');
16   modifiedSentence = inFromServer.readLine();
17   System.out.println("FROM SERVER: " + modifiedSentence);
18   clientSocket.close();
19  }
20 }
```

Το πρόγραμμα TCPClient δημιουργεί τρια ρεύματα και μια socket, όπως φαίνεται και στην παραπάνω εικόνα.

Η socket καλεί τον client socket. Το ρεύμα inFromUser είναι ένα ρεύμα εισόδου στο πρόγραμμα, που προσαρτάται στην πρότυπη είσοδο. Όταν ο χρήσης πληκτρολογεί στο πληκτρολόγιο, οι χαρακτήρες αυτοί ρέουν στο ρεύμα inFromUser. Το ρεύμα inFromServer είναι ένα άλλο ρευμα εισόδου στο πρόγραμμα, και προσαρτάται στην socket. Τέλος, το ρεύμα outToServer είναι ρεύμα εξόδου.

```
Import java.io.*;
Import java.net.*;
```

- *Τα παραπάνω είναι πακέτα της Java. Το πακέτο java.io περιέχει κλάσεις για ρευματα εισόδου/εξόδου, (BufferedReader / DataOutputStream) και το πακέτο java.net παρέχει κλάσεις για υποστήρηξη δικτύου (Κλάσεις socket, server-socket, όπως το αντικείμενο clientSocket).*

```
class TCPClient {

public static void main(String argv[]) throws Exception

}
```

- *Η πρώτη γραμμή είναι η αρχή ενός μπλόκ ορισμού κλάσης. Μια κλάση περιέχει μεταβλητές και μεθόδους. Η κλάση TCPClient εχει την μέθοδο main. Όταν ο διερμηνευτής Java εκτελεί μια εφαρμογή, εκκινή καλώντας την μέθοδο main της κλάσης. Η μέθοδος Main κατόπιν καλεί όλες τις άλλες μεθόδους.*

```
String sentence;
String modifiedsentence;
```

- *Οι παραπάνω γραμμές δηλώνουν αντικείμενα τύπου String. Το αντικείμενο Sentence είναι το string που πληκτρολογείται από τον χρήστη και στέλνεται σ τον εξυπηρέτη. Το αντικείμενο modifiedSentence είναι το string που λαμβάνεται από τον εξυπηρέτη και στέλνεται σε μια πρότυπη έξοδο του χρήστη.*

```
BufferedReader inFromUser =
new BufferedReader( new InputStreamReader(System.in));
```

- *Η παραπάνω γραμμή χρησιμοποιεί το αντικείμενο ρεύματος inFromUser τύπου Buffered Reader. Το ρεύμα εισόδου αρχικοποιείται με System.in, το οποίο προσαρτά το ρεύμα στην πρότυπη είσοδο. Η γραμμή επιτρέπει στον πελάτη να διαβάσει κείμενο από το πληκτρολόγιο του.*

```
Socket clientSocket = new Socket("localhost", 6789);
```

- *Η παραπάνω γραμμή δημιουργεί το αντικείμενο clientsocket τύπου socket. Επίσης εκκινεί την σύνδεση TCP ανάμεσα στον πελάτη και στον εξυπηρέτη. Το στρίνγκ «hostname» πρέπει να αντικαθιστάται από το όνομα υπολογιστή υπηρεσίας του εξυπηρέτη. Πριν εκκινήσει πραγματικά η σύνδεση TCP ο πελάτης κάνει μια αναζήτηση DNS με όνομα υπολογιστής υπηρεσίες για να πάρει την διεύθυνση IP του υπολογιστή υπηρεσίας. Ο αριθμός 6789 είναι ο αριθμός θύρας.*

```
DataOutputStream outToServer =
   new DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer =
  new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

- *Οι παραπάνω γραμμές δημιουργούν αντικείμενα ρεύματος που προσαρτώνται στην socket. Το ρεύμα outToServer παρέχει την έξοδο διεργασίας προς την socket. Το ρεύμα inFromServer παρέχει την είσοδο διεργασίας από την socket*

```
sentence = inFromUser.readLine();
```

- *Η παραπάνω γραμμή τοποθετεί μια γραμμή που πληκτρολογείται από τον χρήστη μέσα από το string sentence. Το string sentence συνεχίζει να συλλέγει χαρακτήρες μέχρι ο χρήστης να τελειώσει τη γραμμή, πληκτρολογώντας ένα χαρακτήρα επαναφοράς. Η γραμμή περνά από την πρότυπη είσοδο, μέσω του ρευματος inFromUser στο string sentence.*

```
outToServer.writeBytes(sentence + '\n');
```

- *Η παραπάνω γραμμή στέλνει το string sentence μαζί με ένα χαρακτήρα τροφοδότησης στο ρεύμα outToServer.*

```
modifiedSentence = inFromServer.readLine();
```

- *Όταν φτάνουν οι χαρακτήρες από τον εξυπηρέτη, αυτοί ρέουν μέσω του ρευματος Infromserver και τοποθετούνται μεσα στο string modifiedSentence. Συνεχίζουν να συλλέγονται χαρακτήρες μέσα στο modifiedSentence μέχρι η γραμμή να τελειώσει με ένα χαρακτήρα επαναφοράς.*

```
System.out.println("FROM SERVER: " + modifiedSentence);
```

- *Εκτυπώνεται στην οθόνη το string modifiedSentence που επιστρέφεται από τον εξυπηρέτη.*

```
clientSocket.close();
```

- *Αυτή η γραμμή κλείνει την socket και τερματίζει την σύνδεση TCP. Το TCP στέλνει ένα μήνυμα TCP στο TCP εξυπηρέτη.*

TCPServer.java

```
import java.io.*;
import java.net.*;

class TCPServer
{
   public static void main(String argv[]) throws Exception
      {
         String clientSentence;
         String capitalizedSentence;
         ServerSocket welcomeSocket = new ServerSocket(6789);

         while(true)
         {
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient =
               new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();
            System.out.println("Received: " + clientSentence);
            capitalizedSentence = clientSentence.toUpperCase() + '\n';
            outToClient.writeBytes(capitalizedSentence);
         }
      }
}
```

Το TCPserver έχει πολλές ομοιότητες με το TCPclient. Η πρώτη γραμμή στο TCPServer είναι σαφώς διαφορετική

ServerSocket welcomeSocket = new ServerSocket(6789);

- *Είναι ένα είδος πόρτας, που κάνει ακρόαση για την σύνδεση κάποιου πελάτη. Συγκεκριμένα, ελέγχει την θύρα 6789.*

Socket connectionSocket = welcomeSocket.accept();

- *Αυτή η γραμμή δημιουργεί μια νέα Socket που καλείται connection socket, όταν ένας πελάτης συνδέεται στο welcomeSocket. Επειδή το socket έχει τον αριθμό θύρας 6789, το TCP καθορίζει μια απευθείας εικονική σύνδεση ανάμεσα στο clientsocket στον πελάτη και το connectionSocket του εξυπηρέτη. Ο πελάτης και ο εξυπηρέτης μπορούν να στέλνουν ελεύθερα bytes επάνω σε αυτή την σύνδεση. Το πρόγραμμα κατόπιν δημιουργεί αντικείμενα ρεύματος, ανάλογα με τα αντικείμενα ρεύματος που δημιουργούνται στο clientSocket.*

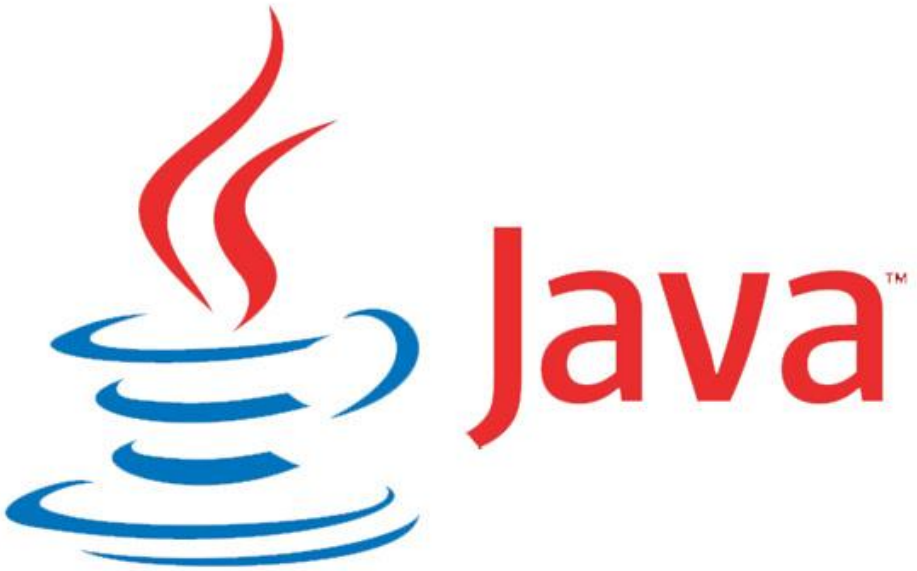
capitalizedSentence = clientSentence.toUpperCase() + '\n';

- Η εντολή-πυρήνας τη εφαρμογής. Δέχεται την γραμμή που στέλνεται από τον πελάτη, την μετατρέπει σε κεφαλαία γράμματα, και προσθέτει ένα χαρακτήρα τροφοδότησης.

ΚΕΦΑΛΑΙΟ 2ο

Γλώσσα προγραμματισμού

“ Java”



(εικ 26) Το λογότυπο της Java.

Κεφάλαιο 2ο : Γλώσσα Προγραμματισμού “Java”

Οι εφαρμογές υπολογιστών αναπτύσσονται ραγδαία τα τελευταία χρόνια, ανεξάρτητα του αντικειμένου τους και την ομάδα χρηστών που της χρησιμοποιεί και σίγουρα τρέχουν σε πολλά τερματικά συστήματα τοπικά η σε κάποιο ευρύτερο δίκτυο διασύνδεσης. Το αναπτυσσόμενο λογισμικό απαιτείται να λειτουργεί σωστά σε πολλά διαφορετικά περιβάλλοντα και αρχιτεκτονικές υπολογιστών, όπως και επίσης να παρέχει την δυνατότητα να συνεργάζεται με άλλες εφαρμογές. Το πρόβλημα που δημιουργείται είναι ότι οι απαιτήσεις για ταχύτητα και μεταφορά είναι συνήθως αντικρουόμενες, ενώ στο θέμα ασφάλειας δεν δίδεται η δέουσα προσοχή. Υπάρχουν λοιπόν, γλώσσες προγραμματισμού οι οποίες ευνοούν την

μεταφορά δεδομένων αλλά υστερούν στην ταχύτητα, λογω του ότι τα προγράμματα ερμηνεύονται αντί να μεταγλωττίζονται, και υπάρχουν γλώσσες προγραμματισμού οι οποίες είναι γρήγορες, αλλά η ταχύτητα τους οφείλεται στο ότι είναι σχεδιασμένες για συγκεκριμένες υπολογιστικές αρχιτεκτονικές.

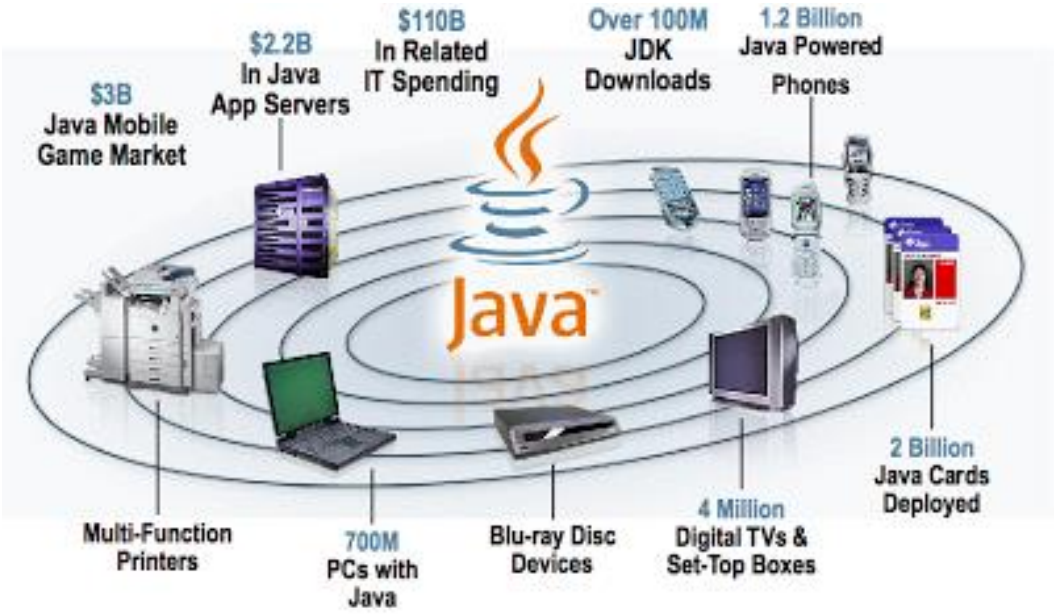
Η ανάπτυξη λογισμικού έχει προσανατολιστεί προς τον αντικειμενοστρεφή προγραμματισμό (object-oriented programming) όπου το λογισμικό δομείται σε αυτόνομες μονάδες οι οποίες έχουν σαφή λειτουργικότητα και διαπροσωπεία.

Η Java λοιπόν, είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε για να δώσει λύση στα προαναφερθέντα προβλήματα. Η επιτυχία της οφείλεται στο ότι μπορεί να χρησιμοποιηθεί για προγραμματισμών ασφαλών, υψηλής απόδοσης εφαρμογών σε ιντερνέτ, οι οποίες μπορούν να τρέξουν αυτούσιες σε διαφορετικά προγραμματιστικά περιβάλλοντα και αρχιτεκτονικές, και ότι παρέχει την δυνατότητα μεταφοράς δυναμικού περιεχομένου σε εφαρμογές πολυμέσων.

2.1,1 Ιστορία της Java

Η γλώσσα της Java αρχικά αναπτύχθηκε από την Sun Microsystems υπό την αιγίδα των James Gosling και Bill Joy, σαν μέρος ενός ερευνητικού έργου ανάπτυξης λογισμικού για ηλεκτρονικές συσκευές καταναλωτικού επιπέδου (π.χ video,τηλεόραση). Ο τρόπος αυτός χρησιμοποίησης της, την μετέτρεψε σε μία ιδανική γλώσσα για την διανομή εκτελέσιμων προγραμμάτων μέσω του WWW (Παγκόσμιου Ιστού) καθώς επίσης σε μία γενικού σκοπού γλώσσα προγραμματισμού για την ανάπτυξη προγραμμάτων τα οποία θα είναι εύκολα στην χρήση και θα μπορούν να μεταφέρονται σε διαφορετικά λειτουργικά συστήματα.

Ο "πατέρας" της Java, James Gosling, που εργαζόταν εκείνη την εποχή για την Sun, έκανε ήδη πειραματισμούς πάνω στη C++ και είχε παρουσιάσει κατά καιρούς κάποιες πειραματικές γλώσσες (C++ ++) ως πρότυπα για το νέο εργαλείο που αναζητούσαν στην Sun. Τελικά μετά από λίγο καιρό κατέληξαν με μια πρόταση για το επιτελείο της εταιρίας, η οποία ήταν η γλωσσα Oak. Το όνομά της το πήρε από το ομώνυμο δένδρο (βελανιδιά) το οποίο ο Gosling είχε έξω από το γραφείο του και έβλεπε κάθε μέρα.



(εικ 27)

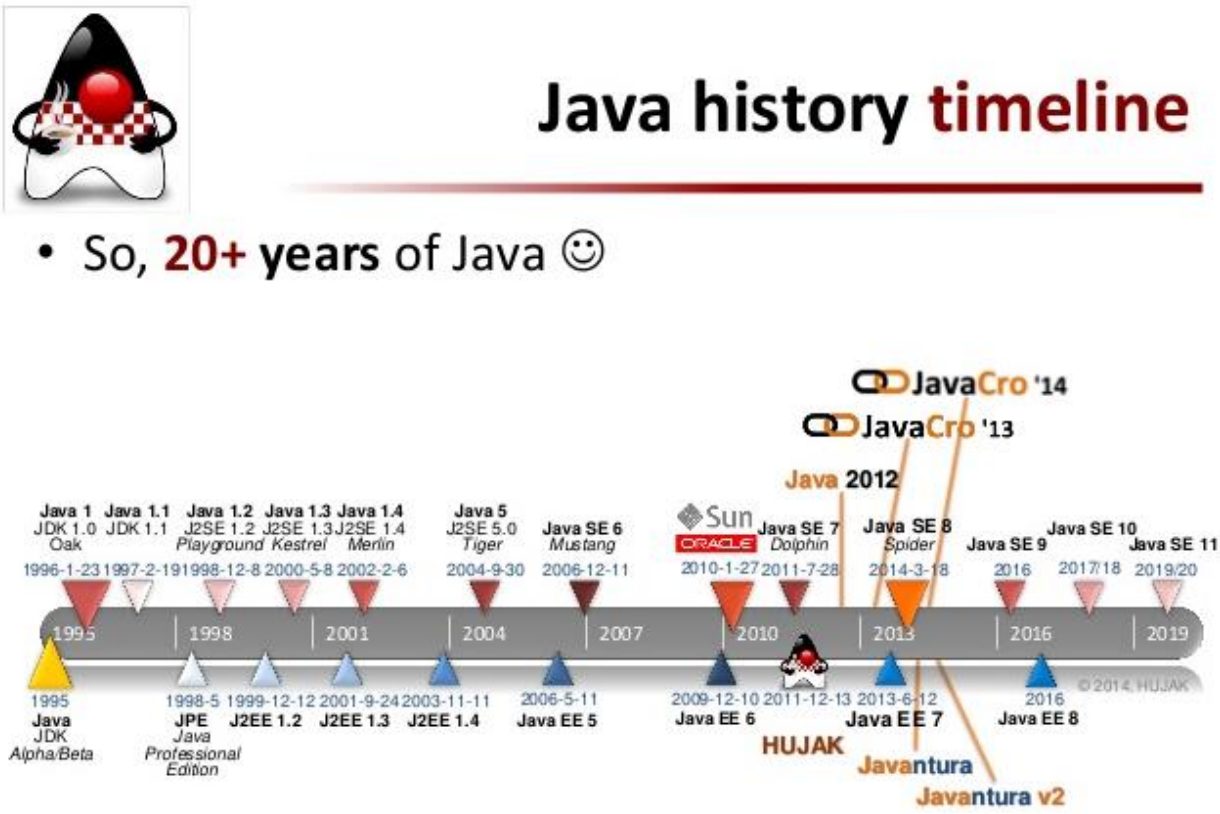
Από την Oak στη Java

Η Oak ήταν μία γλώσσα που διατηρούσε μεγάλη συγγένεια με την C++. Παρόλα αυτά είχε πολύ πιο έντονο αντικειμενοστρεφή (object oriented) χαρακτήρα σε σχέση με την C++ και χαρακτηριζόταν για την απλότητα της. Σύντομα οι υπεύθυνοι ανάπτυξης της νέας γλώσσας ανακάλυψαν ότι το όνομα Oak ήταν ήδη κατοχυρωμένο οπότε κατά την διάρκεια μιας εκ των πολλών συναντήσεων σε κάποιο τοπικό καφέ αποφάσισαν να μετονομάσουν το νέο τους δημιούργημα σε Java που εκτός των άλλων ήταν το όνομα της αγαπημενης ποικιλιας καφέ για τους δημιουργούς της. Η επίσημη εμφάνιση της Java αλλα και του HotJava (πλοηγός με υποστήριξη Java) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η Sun την ανακοίνωσε στο συνέδριο Sun World 1995. Ο πρώτος μεταγλωττιστής (compiler) της ήταν γραμμένος στη γλώσσα C από τον James Gosling. Το 1994, ο A.Van Hoff ξαναγράφει τον μεταγλωττιστή της γλώσσας σε Java, ενώ το Δεκέμβριο του 1995 πρώτες οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια να χρησιμοποιήσουν τη Java για την δημιουργία λογισμικού. Από εκεί και πέρα η Java ακολουθεί μία ανοδική πορεία και είναι πλέον μία από τις πιο δημοφιλείς γλώσσες στον χώρο της πληροφορικής. Στις 13 Νοεμβρίου του 2006

η Java έγινε πλέον μια γλώσσα ανοιχτού κώδικα (GPL) όσον αφορά το μεταγλωττιστή (javac) και το πακέτο ανάπτυξης (JDK, Java Development Kit).

Η εξαγορά από την Oracle και το μέλλον της Java

Στις 27 Απριλίου 2010 η εταιρία λογισμικού Oracle Corporation ανακοίνωσε ότι μετά από πολύμηνες συζητήσεις ήρθε σε συμφωνία για την εξαγορά της Sun Microsystems και των τεχνολογιών (πνευματικά δικαιώματα/ πατέντες) που η δεύτερη είχε στην κατοχή της ή δημιουργήσει. Η συγκεκριμένη συμφωνία θεωρείται σημαντική για το μέλλον της Java και του γενικότερου οικοσυστήματος τεχνολογιών γύρω από αυτή μιας και ο έμμεσος έλεγχος της τεχνολογίας και η εξέλιξη της περνάει σε άλλα χέρια.



(εικ 28) 20+ Χρόνια Java

2.1,2 Χαρακτηριστικά της Java

Τα πιο βασικά χαρακτηριστικά της Java είναι τα ακόλουθα:

• **Είναι σχετικά απλή**:

Το συγκεκριμένο χαρακτηριστικό αναφέρεται σε πολλές πηγές σχετικές με τη γλώσσα Java, αλλά είναι καθαρά υποκειμενικό. Η Java είναι όσο απλή μπορεί να είναι Εισαγωγή στη Γλώσσα Προγραμματισμού Java 11 μία γλώσσα προγραμματισμού. Συγκρινόμενη με τη C++, είναι απλούστερη μιας και έχει εξαλειφθεί η χρήση δεικτών (pointers) ενώ η διαχείριση της μνήμης γίνεται από την ίδια τη γλώσσα.

• **Είναι μεταγλωττιζόμενη (compiled) και ερμηνευόμενη (interpreted**):

Σε αντίθεση με τις C/C++ που είναι compiled γλώσσες, η διαδικασία παραγωγής εκτελέσιμου κώδικα στη Java κάνει χρήση και των δύο αυτών τεχνικών. Η διαδικασία αυτή περιγράφεται αναλυτικά σε επόμενη υποενότητα.

• **Είναι αμιγώς αντικειμενοστρεφής (pure OOD)**:

Η Java υποστηρίζει αποκλειστικά το μοντέλο αντικειμενοστρεφούς προγραμματισμού (object-oriented paradigm) και δεν υπάρχει δυνατότητα χρήσης της σύμφωνα με κάποιο άλλο μοντέλο (π.χ. διαδικαστικό).

• **Είναι φορητή σε επίπεδο μεταγλωττισμένου κώδικα**:

Ένα από τα πιο ισχυρά χαρακτηριστικά της Java. Αυτό πρακτικά σημαίνει πως ένας προγραμματιστής μπορεί να γράψει και μεταγλωττίσει ένα πρόγραμμα π.χ. σε Windows και στη συνέχεια να πάρει το αρχείο που παράχθηκε από τη μεταγλώττιση και να το τρέξει σε ένα μηχάνημα Unix χωρίς καμία αλλαγή. Απαραίτητη προϋπόθεση είναι στο μηχάνημα αυτό (Unix) να είναι εγκατεστημένο το αντίστοιχο JRE.

• **Κάνει αυστηρό έλεγχο τύπων (strongly typed)**:

Η Java απαιτεί από τον προγραμματιστή να κάνει σωστή χρήση των τύπων (περισσότερα για τους τύπους στην ενότητα 2) και δεν επιτρέπει αυθαίρετες μετατροπές όπως οι C/C++.

• **Είναι γλώσσα υψηλού επιπέδου**:

Η εκμάθηση της Java και η σύνταξη κώδικα είναι σχετικά απλή μιας και η γλώσσα κάνει χρήση λέξεων που βρίσκονται πιο κοντά στη φυσική γλώσσα (Αγγλικά) παρά στη γλώσσα μηχανής.

• **Παρέχει υψηλό επίπεδο ασφάλειας:**

Η εκτέλεση προγραμμάτων ελέγχεται από μηχανισμούς ασφαλείας που αποτρέπουν την κακόβουλου κώδικα.

• **Υποστηρίζει πολυμέσα:**

Η Java είναι από τις ελάχιστες γλώσσες της κατηγορίας που παρέχουν έμφυτη υποστήριξη για την ανάπτυξη πολυμεσικών (multimedia) εφαρμογών.

• **Είναι κατάλληλη για προγραμματισμό δικτυακών εφαρμογών:**

Όπως αναφέρθηκε σε προηγούμενη υποενότητα, η Java διευκολύνει την υλοποίηση τόσο δικτυακών (network) όσο και διαδικτυακών (web) εφαρμογών.

• **Υποστηρίζει πολυνηματική επεξεργασία (multi-threaded processing):**

Και στην περίπτωση αυτή, η Java είναι μία από τις ελάχιστες γλώσσες της κατηγορίας της που παρέχει έμφυτη υποστήριξη για την ανάπτυξη multi-threaded εφαρμογών.

• **Κάνει αυτόματη διαχείριση μνήμης**:

Στη Java, η διαχείριση της μνήμης ελέγχεται αποκλειστικά από αυτήν μέσω ενός υποπρογράμματος που ονομάζεται garbage collector (αποκομιστής απορριμάτων) και ο προγραμματιστής δεν εμπλέκεται ποτέ στη διαδικασία αυτή.

• **Είναι δυναμική:**

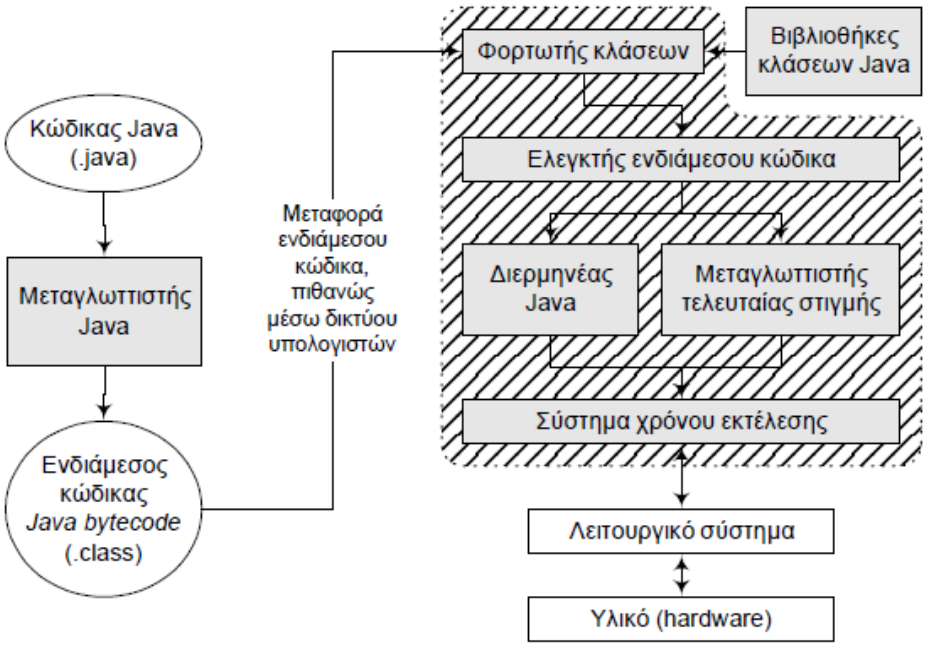
Προσαρμόζεται εύκολα σε διαφορετικά περιβάλλοντα και απαιτήσεις και είναι ιδανική για τη διασύνδεση και επικοινωνία ετερογενών συστημάτων. Η Java ενημερώνεται συνεχώς ενσωματώνοντας και υποστηρίζοντας τις τελευταίες τεχνολογικές εξελίξεις.

• **Κάνει αποκλειστική χρήση της δυναμικής διασύνδεσης (dynamic binding):**

Στη Java η διασύνδεση των δεδομένων και των μεθόδων που αυτά υποστηρίζουν γίνεται κατά την εκτέλεση του προγράμματος (run-time).

Γενικά, η φιλοσοφία της γλώσσας όσον αφορά το προγραμματιστικό μοντέλο που υιοθετεί είναι πως ο προγραμματιστής θα πρέπει να προστατεύεται από τη γλώσσα και τα προγράμματα να είναι ασφαλή και αξιόπιστα.

2.1,3 Εκτέλεση προγραμμάτων



(εικ 29) Εκτέλεση προγραμμάτων Java

Παραπάνω απεικονίζεται η διαδικασία μεταγλώττισης και εκτέλεσης προγραμμάτων Java.

Αρχικά, γίνεται η μεταγλώττιση του κώδικα Java που έχει γράψει ο προγραμματιστής. Για τον σκοπό αυτό εκτελείται ενας Java Compilers που μεταφράζει τον κώδικα Java σε έναν ενδιάμεσο κώδικα (bytecode). Ο ενδιάμεσος κώδικας αποθηκεύεται σε ένα η περισσότερα αρχεία, που είτε χρησιμοποιούνται τοπικά ειτε μεταφέρονται σε άλλους υπολογιστές μέσω του δικτυου.

Μετα τα «ινία» παίρνει η εικονική μηχανή της Java (Java Virtual Machine) , στην οποία φορτώνονται οι κλάσεις και αναλαμβάνει το φόρτωμα του ενδιάμεσο κώδικα που υλοποιεί καποια ζητούμενη κλάση. Αν η κλάση δεν υπάρχει σε μια βιβλιοθήκη κλάσεων της Java, ο φορτωτής κλάσεων αναλαμβάνει την μεταφορά της μέσω διαδικτύου. Κατόπιν γίνεται ο έλεγχος ορθότητας του κώδικα από τον bytecode verifier, που λειτουργεί και ως προστασία της ασφάλειας του υπολογιστικού συστήματος.

Κατόπιν, ακολουθεί η εκτέλεση του ενδιάμεσου κώδικα που πραγματοποιείται είτε από τον διερμηνέα (interpreter) είτε από τον μεταγλωττιστή τελευταίας στιγμής (just-in-time compiler) . Μετά τα τμήματα αυτά επικοινωνούν με τον σύστημα χρόνου εκτέλεσης το οποίο αλληλεπιδρά με το λειτουργικό σύστημα του υπολογιστή και με αυτό τον τρόπο αποκτά πρόσβαση στο υλικό (hardware) και επικοινωνεί τελικά με τον χρήστη.

ΚΕΦΑΛΑΙΟ 3ο

Γλώσσα προγραμματισμού

“ Android”

(εικ 30) λογότυπο Android

3ο Κεφάλαιο – Γλώσσα προγραμματισμού Android

Το Android είναι λειτουργικό σύστημα που αρχικά δημιουργήθηκε το 2007 για συσκευές κινητής τηλεφωνίας και κατέκλισε την αγορά και για συσκευές πέρα από κινητά τηλέφωνα.

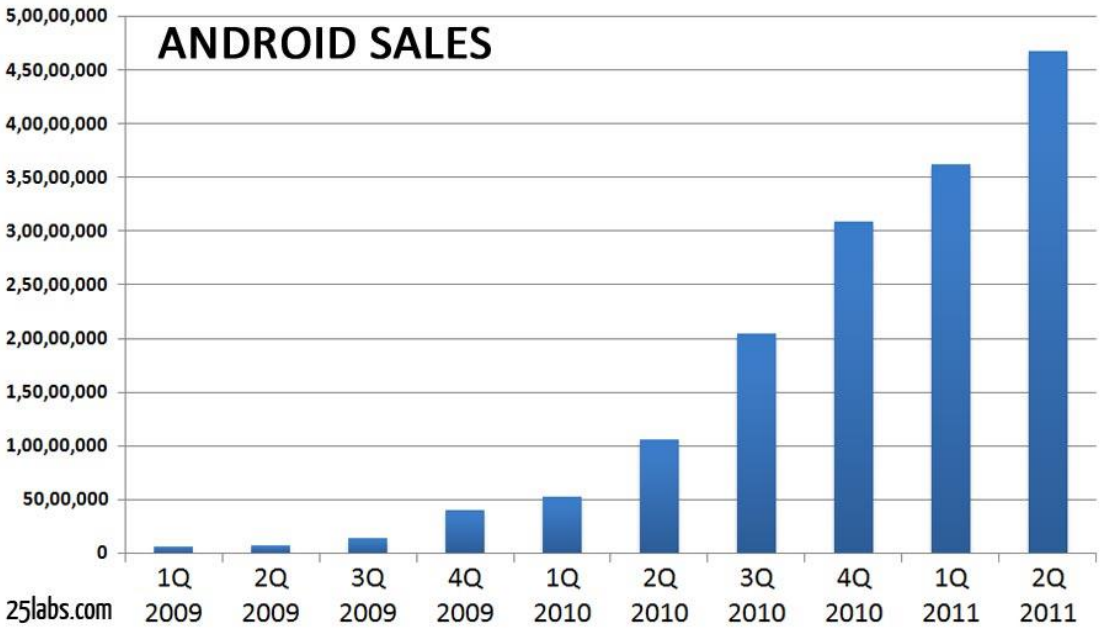
Το Android τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Handset Alliance |Open Handset Alliance .

(εικ 31)

Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλεφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φυτογραφικές μηχανές, συνηθισμένους Η/Υ (π.χ. το HP Slate 21) και σε άλλες ηλεκτρονικές συσκευές.

(εικ 32-33-34) Συσκευές συμβατές με λογισμικό Android

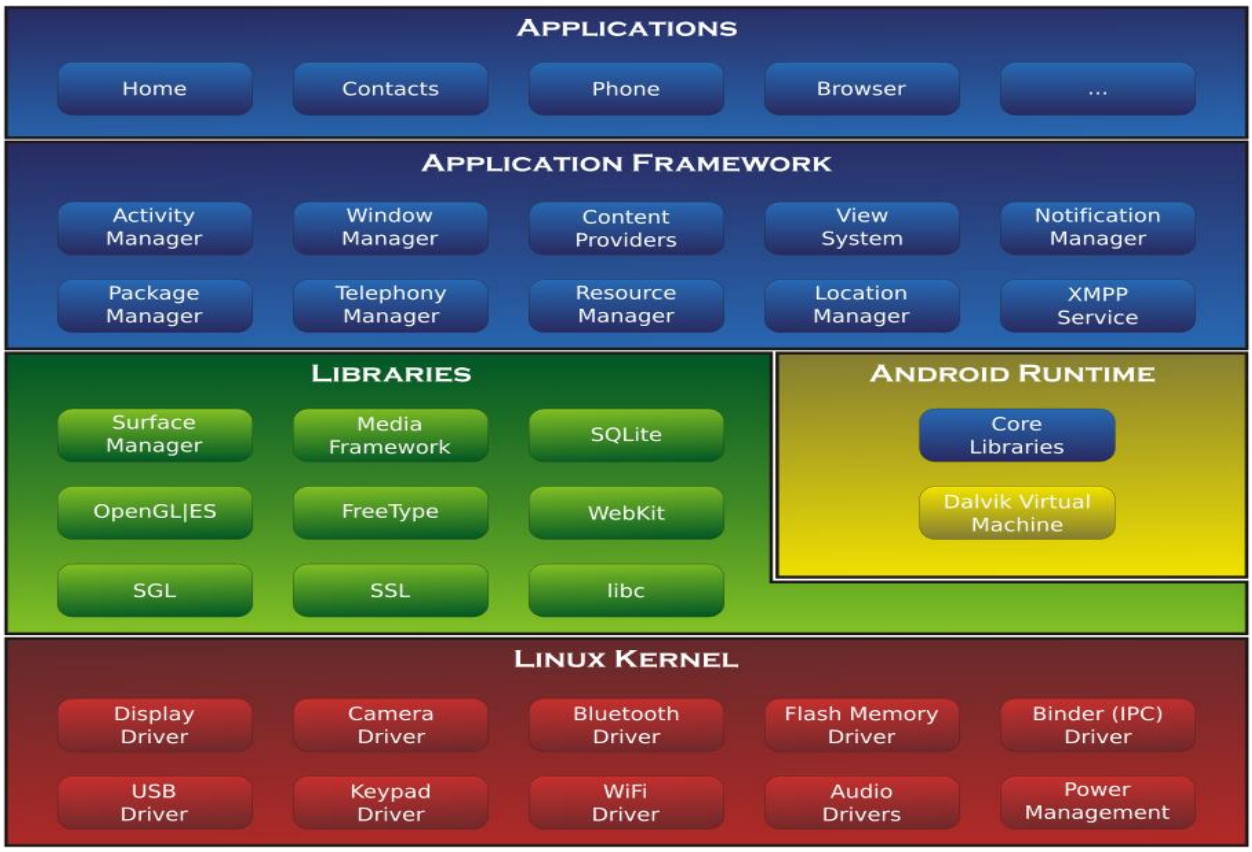
Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί.



(εικ 35) Πωλήσεις Android 2009-2011

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android είναι ένα ρομπότ σε χρώμα πράσινου μήλου και σχεδιάστηκε από τη γραφίστρια Irina Blok.

Αρχιτεκτονική Android



(εικ 36) Αρχιτεκτονική Android

Linux Kernel : Αυτός είναι ο Kernel στον οποίο βασίζεται το Android και βρίσκεται στο χαμηλότερο επίπεδο. Παρέχει τους Drivers τους οποιούς χρειάζεται για να τρέξει το σύστημα, όπως της οθόνης, της κάμερας κ.α.

Libraries : Οι βιβλιοθήκες(Libraries) όλο τον κώδικα που περιέχει το Android OS. Παραδείγματος χάριν, η SQLite βιβλιοθήκη παρέχει υποστήριξη έτσι ώστε μια εφαρμογή να χρησιμοποιήσει την αποθήκευση δεδομένων, η Webkit βιβλιοθήκη παρέχει λειτουργίες για το διαδικτυακό σερφάρισμα.

Android Runtime : Στο ίδιο επίπεδο με τις βιβλιοθήκες, το Android Runtime παρέχει ένα σύνολο βασικών βιβλιοθηκών που επιτρέπουν στους προγραμματιστές να γράψουν εφαρμογές χρησιμοποιώντας JAVA. Επίσης περιλαμβάνει την Dalvik virtual machine, που επιτρέπει κάθε εφαρμογή να τρέξει την δικιά της εργασία, μαζί με την δικιά της ξεχωριστή Dalvik virtual machine. Η Dalvik είναι μια εξειδικευμένη virtual machine, ειδικά διαμορφωμένη για κινητές συσκευές που έχουν περιορισμένη μνήμη και ισχύ.
Application Framework : Εκθέτει διάφορες δυνατότητες του Android στους προγραμματιστές των εφαρμογών ώστε να τις χρησιμοποιήσουνε στις εφαρμογές τους.

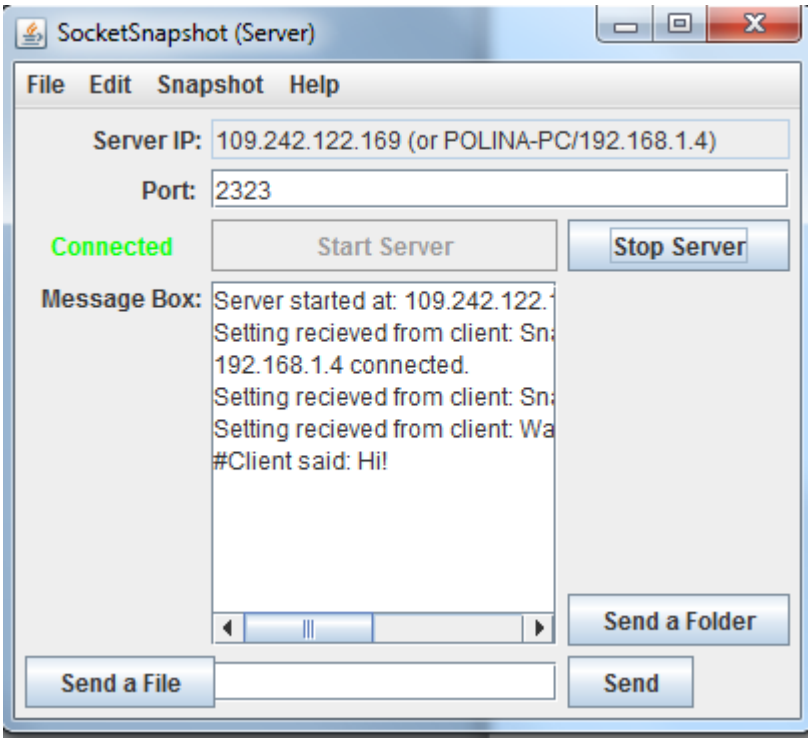
Applications : Το πιο υψηλό επίπεδο, εδώ βρισκουμε εφαρμογές που έρχονται μαζί με την Android συσκευή σου(όπως τηλέφωνο, επαφές, μουσική κ.α.), όπως επίσης εφαρμογές που κάνεις εγκατάσταση.Οποιαδήποτε εφαρμογή είναι σε αυτό το επίπεδο.

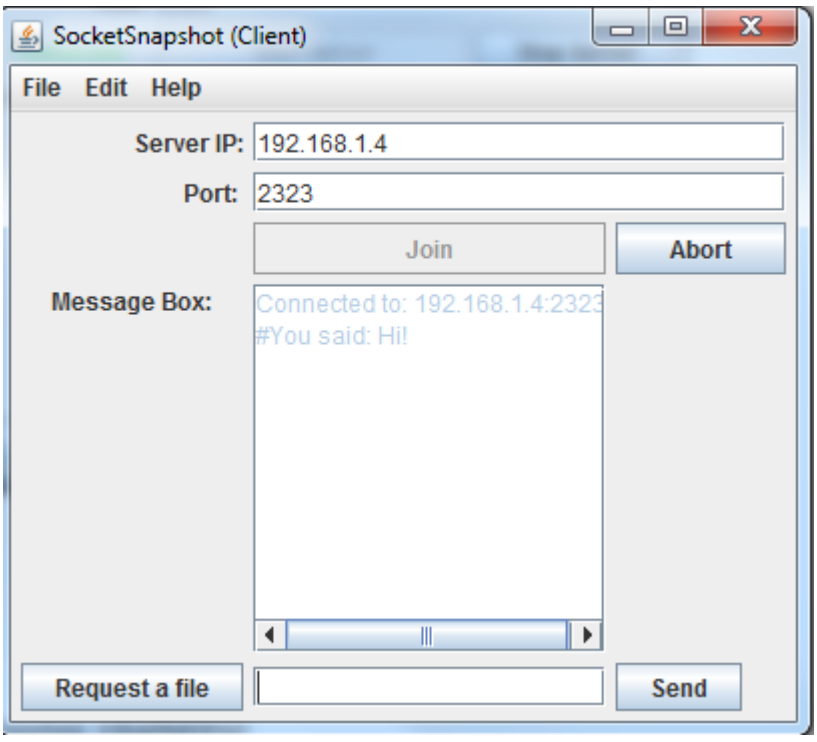
Εκδόσεις Android

(εικ 37 εκδόσεις Android)

- Cupcake (1.5)
- Donut (1.6)
- Eclair (2.0–2.1) (έφερε αλλαγές στο περιβάλλον και έφερε υποστήριξη HTML5 στον περιηγητή(browser)
- Froyo (2.2–2.2.3) (έδωσε μεγαλύτερη ταχύτητα και έφερε υποστήριξη flash μαζί με δυνατότητα wifi hotspot !)
- Gingerbread (2.3–2.3.7) (βελτίωσε το περιβάλλον όπως και δυνατότητες για ποιο "βαριές" εφαρμογές, επιπλέον πρόσθεσε υποστήριξη NFC(Near Field Communication)
- Honeycomb (3.0–3.2.6) (έφερε αλλαγές κυρίως στο γραφικό περιβάλλον και πρόσθεσε υποστήριξη πολλαπλών πυρήνων μαζί με βελτιωμένα γραφικά.)
- Ice Cream Sandwich (4.0–4.0.4) (έχει σκοπό να "ενώσει" τις εκδόσεις για ταμπλέτες και κινητά και να προσθεσει υποστήριξη για την Google TV.)
- Jelly Bean (4.1–4.3.1)
- KitKat (4.4–4.4.4, 4.4W–4.4W.2)
- Lollipop (5.0–5.1.1) (Έχει κάνει βελτιώσεις στην κατανάλωση ενέργειας και στη συνδεσιμότητα.)

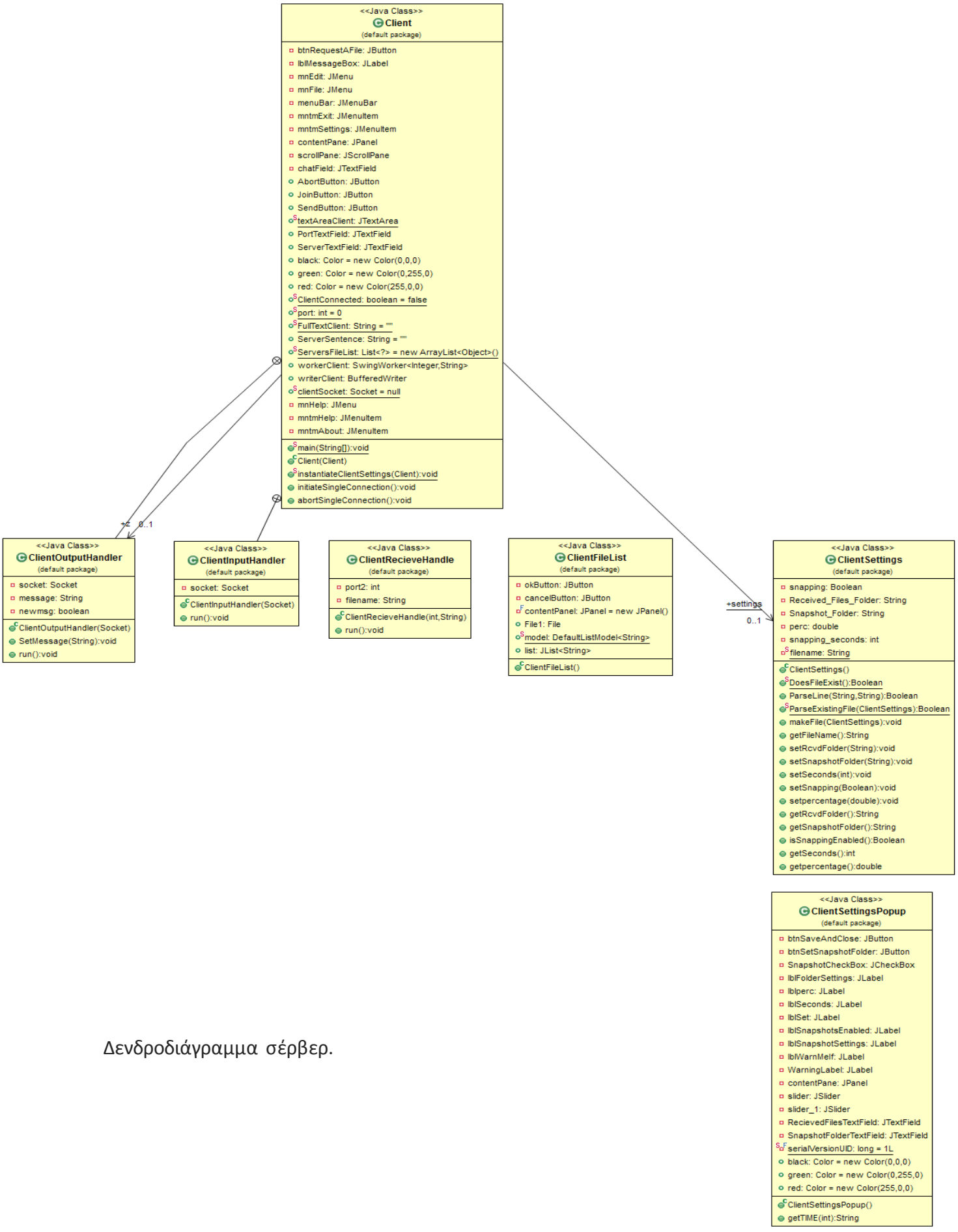
ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ



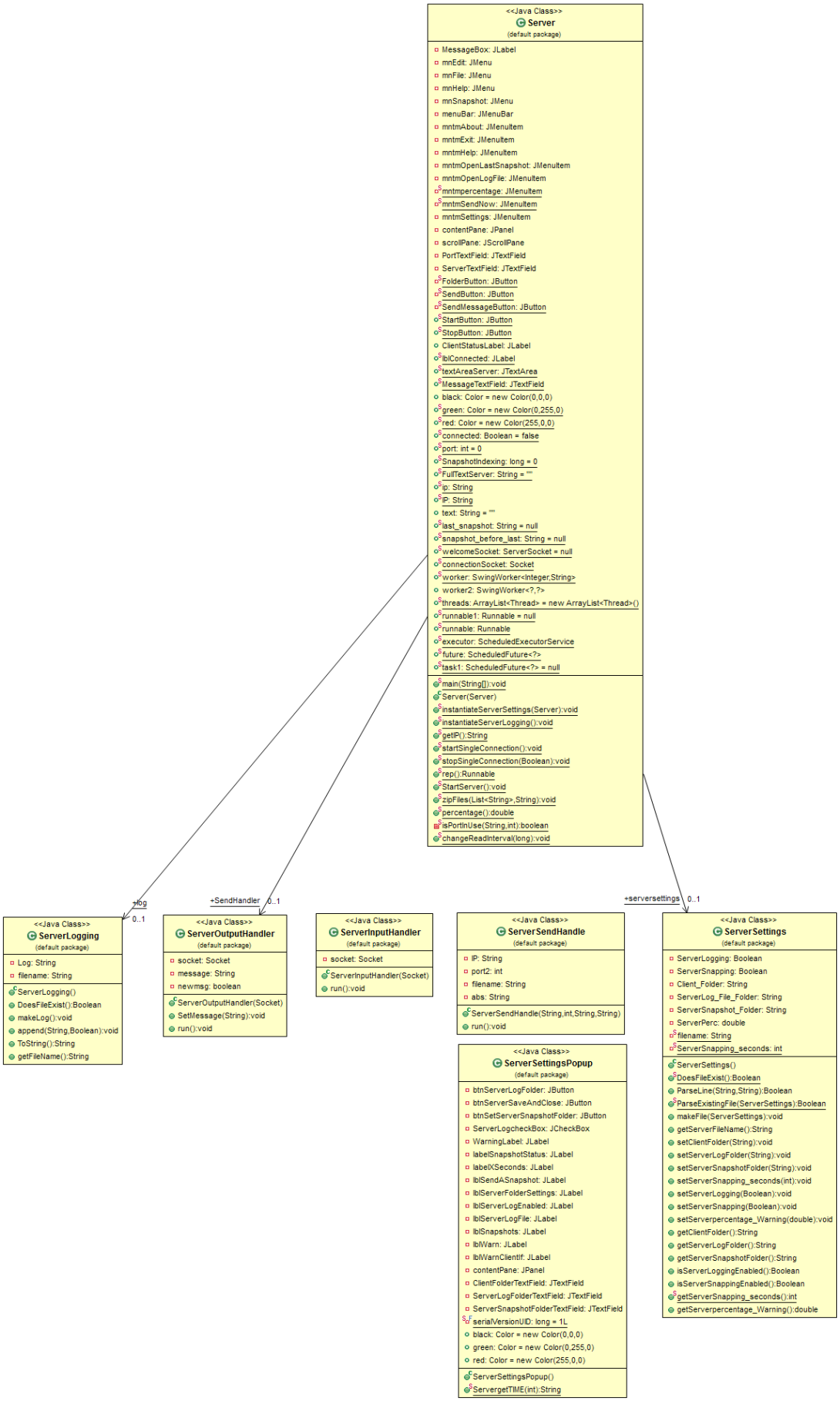


Screen Shot της Java Εφαρμογής Server-Client

ΔΕΝΔΡΟΔΙΑΓΡΑΜΜΑ ΕΦΑΡΜΟΓΗΣ JAVA



Δενδροδιάγραμμα σέρβερ.



Δενδροδιάγραμμα Client.

Κώδικας Εφαρμογής Java

Κώδικας Server

Server.java

```
import java.awt.Color;
import java.awt.Desktop;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.URLConnection;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CancellationException;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;
import java.util.concurrent.TimeUnit;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.SwingWorker;
import javax.swing.border.EmptyBorder;
```

```
import com.github.sarxos.webcam.Webcam;
import com.github.sarxos.webcam.WebcamException;

/**
* SocketSnapshot's Server's main class.
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

@SuppressWarnings("serial")
public class Server extends JFrame {
/*******************************************
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+     *
*     |P|r|i|v|a|t|e| |V|a|r|i|a|b|l|e|s|     *
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+     *
*******************************************/

/* Java SWING (GUI) */

private JLabel MessageBox;
private JMenu mnEdit;
private JMenu mnFile;
private JMenu mnHelp;
private JMenu mnSnapshot;
private JMenuBar menuBar;
private JMenuItem mntmAbout;
private JMenuItem mntmExit;
private JMenuItem mntmHelp;
private JMenuItem mntmOpenLastSnapshot;
private JMenuItem mntmOpenLogFile;
private static JMenuItem mntmpercentage;
private static JMenuItem mntmSendNow;
private JMenuItem mntmSettings;
private JPanel contentPane;
private JScrollPane scrollPane;
private JTextField PortTextField;
private JTextField ServerTextField;
private static JButton FolderButton;
private static JButton SendButton;
private static JButton SendMessageButton;

/*******************************************
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+     *
*     |P|u|b|l|i|c| |V|a|r|i|a|b|l|e|s|     *
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+     *
*******************************************/

/* Java SWING (GUI) */

public static JButton StartButton;
public static JButton StopButton;
public           JLabel ClientStatusLabel;
public static JLabel lblConnected;
public static JTextArea textAreaServer;
public static JTextField MessageTextField;

/* Colors */

public Color black        = new Color(0,0,0);
public static Color green = new Color(0,255,0);
public static Color red   = new Color(255,0,0);

/* Booleans */

public static Boolean connected = false;

/* Numbers */
public static int port = 0;
public static long SnapshotIndexing = 0;

/* Strings */

public static String FullTextServer = "";
public static String ip;
public static String IP;
```

```
public           String text = "";
public static String last_snapshot = null;
public static String snapshot_before_last = null;

/* Logging */

public static ServerLogging log;

/* Server-only stuff */

public static ServerOutputHandler SendHandler;
public static ServerSettings serversettings;
public static ServerSocket welcomeSocket = null;
public static Socket connectionSocket;

/* Threads and tasks */

public static SwingWorker<Integer, String> worker;
public        SwingWorker<?, ?> worker2;
public static ArrayList<Thread> threads = new ArrayList<Thread>();
public static Runnable runnable1 = null;
public static Runnable runnable;
public static ScheduledExecutorService executor;
public static ScheduledFuture<?> future;
public static ScheduledFuture<?> task1 = null;

/**
* In Java, the main method is where control enters a program or piece of code;
* this is where a program starts its execution.
*
* Specifically, it instantiates a new Server instance,
* thus a new frame (the Server frame).
*
* @param  args Contains the supplied command-line arguments as an array of String
objects.
*/

public static void main(String[] args) {
EventQueue.invokeLater(new Runnable() {
public void run() {
try {
Server frame = null;
frame = new Server(frame);
frame.setVisible(true);
} catch (Exception e) {
e.printStackTrace();
}
}
});
}

/**
* The Server's constructor;
*
* It does everything that's required for the program to run properly (settings,logfile
et.c.),
* while using other classes as well to perform the required operations.
*
* @param frame The Server's main frame.
*/

public Server(Server frame) {

/* Instantiate Server Settings */

instantiateServerSettings(frame);

/* Instantiate Server Log */

instantiateServerLogging();

/*********************************************************************************************
*                                                                                            *
* Side Note:                                                                                                *
*                                                                                            *
* No further comments will be given about:                                                   *
*                                                                                            *
```

```
* 1. Logs:
*
* try {
*
*         // If Server logging is enabled
*
*         if (serversettings.isServerLoggingEnabled() == true){
*
*                 // Log it
*
*                 log.append(e2.toString(),false);
*  }
*
* // Print out the Stack Trace in case of exception
*
* } catch (IOException e1) {
*         e1.printStackTrace();
* }
*
* 2. Single DataOutputStream (sending a line (string ending with CRLF)):
*
*
* // Create a new data output stream from the connection socket
*
* DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
*
* // Write bytes to client (whereas "Lorem ipsum" is the string and CRLF is the "\r\n"
*
* outToClient.writeBytes("Lorem ipsum\r\n");
*
* // This forces any buffered output bytes to be written out to the stream.
*
*
* outToClient.flush();
*
*
*
*****************************************************************************************/

/*****************************************************************************************
*                                                                                       *
*    ,ad8888ba,                                  88          88                  88   *
*   d8"'    `"8b                                 88          ""                  88   *
*  d8'                                           88                              88   *
*  88            8b,dPPYba,  ,adPPYYba,  8b,dPPYba,   88,dPPYba,   88   ,adPPYba,  ,adPPYYba,  88   *
*  88      88888 88P'   "Y8  ""     `Y8  88P'    "8a  88P'    "8a  88  a8"     ""  ""     `Y8  88   *
*  Y8,        88  88          ,adPPPPP88  88       d8  88       88  88  8b          ,adPPPPP88  88   *
*   Y8a.    .a88  88          88,    ,88  88b,   ,a8"  88       88  88  "8a,   ,aa  88,    ,88  88   *
*    `"Y88888P"   88          `"8bbdP"Y8  88`YbbdP"'   88       88  88   `"Ybbd8"'  `"8bbdP"Y8  88   *
*                                         88                                                    *
*                                         88                                                    *
*****************************************************************************************/

*************************************************
*                                               *
*  88        88                                 *
*  88        88                                 *
*  88        88                                 *
*  88        88  ,adPPYba,   ,adPPYba,  8b,dPPYba,  *
*  88        88  I8[    ""  a8P_____88  88P'   "Y8  *
*  88        88   `"Y8ba,   8PP"""""""  88          *
*  Y8a.    .a8P  aa    ]8I  "8b,   ,aa  88          *
*    `"Y8888Y"'   `"YbbdP"'   `"Ybbd8"'  88          *
*                                               *
*************************************************

*****************************************************************************************
*                                                                                       *
*  88                                    ad88                                       *
*  88               ,d                  d8"                                        *
*  88               88                  88                                         *
*  88  8b,dPPYba,  MM88MMM  ,adPPYba,  8b,dPPYba,  MM88MMM  ,adPPYYba,   ,adPPYba,   ,adPPYba,  *
*  88  88P'   `"8a   88    a8P_____88  88P'   "Y8    88     ""     `Y8  a8"     ""  a8P_____88  *
*  88  88       88   88    8PP"""""""  88            88     ,adPPPPP88  8b          8PP"""""""  *
*  88  88       88   88,   "8b,   ,aa  88            88     88,    ,88  "8a,   ,aa  "8b,   ,aa  *
*  88  88       88   "Y888  `"Ybbd8"'  88            88     `"8bbdP"Y8   `"Ybbd8"'   `"Ybbd8"'  *
*                                                                                       *
*****************************************************************************************/


/* Set frame's (window's) title */

this.setTitle("SocketSnapshot (Server)");

/* Set Default Close Operation ([X] button) to terminate the program */
```

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

/* Set Window's default boundaries (can be resized, and also the window is responsive)
*/

setBounds(100, 100, 408, 365);

/*
* Create a new JPanel and set the correct spacing and alignment.
*
* The JPanel class provides general-purpose containers for lightweight components.
*
*
*/

contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);

/* Use GridBagLayout for better alignment. */

GridBagLayout gbl_contentPane = new GridBagLayout();
gbl_contentPane.columnWidths = new int[]{93, 201, 201, 93, 0};
gbl_contentPane.rowHeights = new int[]{20, 20, 23, 216, 0, 0};
gbl_contentPane.columnWeights = new double[]{0.0, 1.0, 1.0, 0.0, Double.MIN_VALUE};
gbl_contentPane.rowWeights = new double[]{0.0, 0.0, 0.0, 1.0, 0.0, Double.MIN_VALUE};
contentPane.setLayout(gbl_contentPane);

/***
*
*    88b           d88
*    888b         d888
*    88`8b       d8'88
*    88 `8b     d8' 88   ,adPPYba,  8b,dPPYba,   88       88  ,adPPYba,
*    88  `8b   d8'  88  a8P     88  88P'   `"8a  88       88  I8[    ""
*    88   `8b d8'   88  8PP"""""""  88       88  88       88   `"Y8ba,
*    88    `888'    88  "8b,   ,aa  88       88  "8a,   ,a88  aa    ]8I
*    88     `8'     88   `"Ybbd8"'  88       88   `"YbbdP'Y8  `"YbbdP"'
*
*
*/

/* Add a Menu Bar */

menuBar = new JMenuBar();
setJMenuBar(menuBar);

/***
* Add Menu "File" to the Menu Bar.
*
*     +-+-+-+-+
*     |F|i|l|e|
*     +-+-+-+-+
*
*/

mnFile = new JMenu("File");
menuBar.add(mnFile);

/***
*  Add Menu Item "Open last snapshot" to "File" Menu.
*
*     +-+-+-+-+ +-+-+-+-+ +-+-+-+-+-+-+-+-+
*     |O|p|e|n| |l|a|s|t| |s|n|a|p|s|h|o|t|
*     +-+-+-+-+ +-+-+-+-+ +-+-+-+-+-+-+-+-+
*/

mntmOpenLastSnapshot = new JMenuItem("Open last snapshot");

/* Add an Action Listener to the "Open last Snapshot" Menu Item. */

mntmOpenLastSnapshot.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent arg0){
if ((serversettings.isServerSnappingEnabled() == true) && (last_snapshot != null)){
String absolute = serversettings.getServerSnapshotFolder().replace("\\",
File.separator);;
```

```
File f = new File(absolute+last_snapshot+".jpg");
try {
Desktop.getDesktop().open(f);
} catch (FileNotFoundException e) {
JOptionPane.showMessageDialog(null,
"File not found",
"Can't open file",
JOptionPane.ERROR_MESSAGE);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
} catch (IOException e2){
JOptionPane.showMessageDialog(null,
"Cannot open " + last_snapshot + ".jpg" + "\nAccess denied.\n.",
"Can't open file",
JOptionPane.ERROR_MESSAGE);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e2.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
}else{
JOptionPane.showMessageDialog(null,
"You have not taken any snapshots yet!\n.",
"Error",
JOptionPane.ERROR_MESSAGE);
}
}
});
mnFile.add(mntmOpenLastSnapshot);

/***
*     Add "Open log file.." Menu Item to "File" Menu
*     +-+-+-+-+ +-+-+-+ +-+-+-+-+
*     |O|p|e|n| |l|o|g| |f|i|l|e|
*     +-+-+-+-+ +-+-+-+ +-+-+-+-+
*/

mntmOpenLogFile = new JMenuItem("Open log file");

/* Add an Action Listener to the "Open log file" Menu Item. */

mntmOpenLogFile.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
if (serversettings.isServerLoggingEnabled() == true){
if (log.DoesFileExist()){
try {
File file = new File("./" + log.getFileName());
Desktop.getDesktop().open(file);
} catch (IOException e1) {
JOptionPane.showMessageDialog(null,
"Cannot open " + last_snapshot + ".jpg" + "\nAccess denied.\n.",
"Can't open file",
JOptionPane.ERROR_MESSAGE);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e1.toString(),false);
}
} catch (IOException e2) {
e1.printStackTrace();
}
}
}else{
JOptionPane.showMessageDialog(null,
"File not found",
"Can't open file",
JOptionPane.ERROR_MESSAGE);
}
}else{
JOptionPane.showMessageDialog(null,
```

```
"Logging is disabled",
"Please enable logging",
JOptionPane.ERROR_MESSAGE);
}
}
});
mnFile.add(mntmOpenLogFile);

/***
*     Add "Exit" Menu Item to "File" Menu
*     +-+-+-+-+
*     |E|x|i|t|
*     +-+-+-+-+
*/

mntmExit = new JMenuItem("Exit");

/* Add an Action Listener to the "Exit" Menu Item. */

mntmExit.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
System.exit(0);
}
});
mnFile.add(mntmExit);

/***
* Add "Edit" Menu to the Menu Bar.
*
*     +-+-+-+-+
*     |E|d|i|t|
*     +-+-+-+-+
*
*/

mnEdit = new JMenu("Edit");
menuBar.add(mnEdit);

/***
* Add "Settings" Menu Item to "Edit" Menu.
*     +-+-+-+-+-+-+-+-+-+
*     |S|e|t|t|i|n|g|s|
*     +-+-+-+-+-+-+-+-+-+
*
*/

mntmSettings = new JMenuItem("Settings");

/* Add an Action Listener to the "Settings" Menu Item. */

mntmSettings.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
EventQueue.invokeLater(new Runnable() {
public void run() {
try {
ServerSettingsPopup frame = new ServerSettingsPopup();
frame.setVisible(true);
} catch (Exception e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
}
});
}
});
mntmSettings.setHorizontalAlignment(SwingConstants.TRAILING);
mnEdit.add(mntmSettings);

/***
* Add "Snapshot" Menu to Menu Bar
*     +-+-+-+-+-+-+-+-+
*     |S|n|a|p|s|h|o|t|
```

```
*     +-+-+-+-+-+-+-+-+
*/

mnSnapshot = new JMenu("Snapshot");
menuBar.add(mnSnapshot);

/***
* Add "Send a Snapshot Now" Menu Item to "Snapshot" Menu
*     +-+-+-+-+ +-+ +-+-+-+-+-+-+-+-+ +-+-+-+
*     |S|e|n|d| |a| |S|n|a|p|s|h|o|t| |N|o|w|
*     +-+-+-+-+ +-+ +-+-+-+-+-+-+-+-+ +-+-+-+
*/

mntmSendNow = new JMenuItem("Send a Snapshot Now");
mntmSendNow.setEnabled(false);

/* Add an Action Listener to the "Send a Snapshot Now" Menu Item */

mntmSendNow.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

/* Initialize webcam and image variables. */

Webcam webcam = null;
BufferedImage image = null;
try{

/* Get default webcam. */

webcam = Webcam.getDefault();

/* If locked unlock. */

if (webcam.getLock().isLocked()) {
webcam.getLock().unlock();
}

/* if still locked cancel the task. */

if (webcam.getLock().isLocked() == true){
JOptionPane.showMessageDialog(null,
"Can't open take a screenshot.\n.",
"Can't unlock webcam",
JOptionPane.ERROR_MESSAGE);
return;
}

/* Set an acceptable dimension (640x480). */

webcam.setViewSize(new Dimension(640, 480));

/* If it isn't open then open it. */

if (webcam.isOpen() == false){
webcam.open();
}

/* "Sleep" the thread -- workaround for "fast" (modern) processors. */

Thread.sleep(100);

/* Get image from webcam. */

image = webcam.getImage();
Thread.sleep(100);

/* Close the webcam. */

webcam.close();
Thread.sleep(100);

/* Any exception leads to thread closure. */

} catch (WebcamException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
```

```
}
} catch (IOException e1) {
e1.printStackTrace();
}
JOptionPane.showMessageDialog(null,
"Can't open take a screenshot.\n.",
"Fail",
JOptionPane.ERROR_MESSAGE);
}
if (image == null){
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append("Failed to take a snapshot",true);
}
} catch (IOException e1) {
e1.printStackTrace();
}
JOptionPane.showMessageDialog(null,
"Can't open take a screenshot.\n.",
"Fail",
JOptionPane.ERROR_MESSAGE);
}

/* Name the webcam snapshot in a SnapshotXXXXX (numeric prefix) format. */

String formatted = String.format("%05d", SnapshotIndexing);

/* Set filename and absolute path. */

String absolute = serversettings.getServerSnapshotFolder().replace("\\",
File.separator);;
String filename = "Snapshot" + formatted + ".jpg";
try {

/* Increment the snapshot index (number of snapshots). */

SnapshotIndexing++;

/* Write the image to an image file. */

ImageIO.write(image, "JPG", new File(absolute+filename));

/* If snapshot before last (previous snapshot) is null. */

if (snapshot_before_last == null){

/* Consider both snapshots (last and previous) to be the same. */

snapshot_before_last = filename;
last_snapshot = filename;
}else{

/* Else if they aren't the same. */

Thread.sleep(100);
Boolean diff = ((snapshot_before_last.equals(last_snapshot)) == true);
String temp = last_snapshot;
String temp2 = snapshot_before_last;
last_snapshot = filename;
snapshot_before_last = temp;

/* Delete the snapshot before before last snapshot. */

if (diff == false){
File f = new File(absolute+temp2);
if (f.exists()){
f.delete();
}
}
}

/* In case of exception decrement the snapshot index. */

} catch (IOException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
```

```
}
} catch (IOException e1) {
e1.printStackTrace();
}
SnapshotIndexing--;
}

/* Send the file. */

try {
DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
outToClient.writeBytes("#" + filename +"\r\n");
outToClient.flush();
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
Thread thread = new Thread(new ServerSendHandle(IP,port+1,filename,absolute));
threads.add(thread);
thread.start();
}
});
mnSnapshot.add(mntmSendNow);

/***
* Add "View current percentage difference" Menu Item to "Snapshot" Menu
*     +-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+
*     |V|i|e|w| |c|u|r|r|e|n|t| |p|e|r|c|e|n|t|a|g|e| |d|i|f|f|e|r|e|n|c|e|
*          +-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+
*/

mntmpercentage = new JMenuItem("View current percentage difference");

/* Add an Action Listener to the "View current percentage difference" Menu Item */

mntmpercentage.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

/* Initialize webcam and image variables. */

Webcam webcam = null;
BufferedImage image = null;
try{

/* Get default webcam. */

webcam = Webcam.getDefault();

/* If locked unlock. */

if (webcam.getLock().isLocked()) {
webcam.getLock().unlock();
}

/* if still locked cancel the task. */

if (webcam.getLock().isLocked() == true){
JOptionPane.showMessageDialog(null,
"Can't open take a screenshot.\n.",
"Can't unlock webcam",
JOptionPane.ERROR_MESSAGE);
return;
}

/* Set an acceptable dimension (640x480). */

webcam.setViewSize(new Dimension(640, 480));

/* If it isn't open then open it. */

if (webcam.isOpen() == false){
```

```
webcam.open();
}

/* "Sleep" the thread -- workaround for "fast" (modern) processors. */

Thread.sleep(100);

/* Get image from webcam. */

image = webcam.getImage();
Thread.sleep(100);

/* Close the webcam. */

webcam.close();
Thread.sleep(100);

/* Any exception leads to thread closure. */

} catch (WebcamException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
JOptionPane.showMessageDialog(null,
"Can't open take a screenshot.\n.",
"Fail",
JOptionPane.ERROR_MESSAGE);
}
if (image == null){
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append("Failed to take a snapshot",true);
}
} catch (IOException e1) {
e1.printStackTrace();
}
JOptionPane.showMessageDialog(null,
"Can't open take a screenshot.\n.",
"Fail",
JOptionPane.ERROR_MESSAGE);
}

/* Name the webcam snapshot in a SnapshotXXXXX (numeric prefix) format. */

String formatted = String.format("%05d", SnapshotIndexing);

/* Set filename and absolute path. */

String absolute = serversettings.getServerSnapshotFolder().replace("\\",
File.separator);;
String filename = "Snapshot" + formatted + ".jpg";
try {

/* Increment the snapshot index (number of snapshots). */

SnapshotIndexing++;

/* Write the image to an image file. */

ImageIO.write(image, "JPG", new File(absolute+filename));

/* If snapshot before last (previous snapshot) is null. */

if (snapshot_before_last == null){

/* Consider both snapshots (last and previous) to be the same. */

snapshot_before_last = filename;
last snapshot = filename;
}else{

/* Else if they aren't the same. */
```

```
Thread.sleep(100);
Boolean diff = ((snapshot_before_last.equals(last_snapshot)) == true);
String temp = last_snapshot;
String temp2 = snapshot_before_last;
last_snapshot = filename;
snapshot_before_last = temp;

/* Delete the snapshot before before last snapshot. */

if (diff == false){
File f = new File(absolute+temp2);
if (f.exists()){
f.delete();
}
}
}

/* In case of exception decrement the snapshot index. */

} catch (IOException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
SnapshotIndexing--;
}

if ((snapshot_before_last != null) && (snapshot_before_last != last_snapshot)){
double p = percentage();
if (p > serversettings.getServerpercentage_Warning()){
try {
Thread.sleep(100);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(p+"% was noticed.",true);
}
} catch (IOException e1) {
e1.printStackTrace();
}
FullTextServer += p+"% was noticed.\r\n";
textAreaServer.setText(FullTextServer);
DataOutputStream outToClient = new
DataOutputStream(Server.connectionSocket.getOutputStream());
outToClient.writeBytes(p+"% was noticed.\r\n");
outToClient.flush();
} catch (IOException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
}
}
}
});
mnSnapshot.add(mntmpercentage);
mntmpercentage.setEnabled(false);

/***
* Add "Help" Menu to the Menu Bar.
*
*     +-+-+-+-+
*     |H|e|l|p|
*     +-+-+-+-+
*
*/

mnHelp = new JMenu("Help");
menuBar.add(mnHelp);

/***
```

```
* Add "Help" Menu Item to "Help" Menu.
*     +-+-+-+-+
*     |H|e|l|p|
*     +-+-+-+-+
*
*/

mntmHelp = new JMenuItem("Help");
mnHelp.add(mntmHelp);
mntmHelp.setEnabled(false);

/***
* Add "About" Menu Item to "Help" Menu.
*     +-+-+-+-+-+
*     |A|b|o|u|t|
*     +-+-+-+-+-+
*/

mntmAbout = new JMenuItem("About..");
mntmAbout.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
JOptionPane.showMessageDialog(null,
"This program was created as a part of the project of Thesis of Paulina Barniadaki,\n"
+ "student of Electronic Engineering Department, School of Applied Sciences.",
"About this program..",
JOptionPane.INFORMATION_MESSAGE);
}
});
mnHelp.add(mntmAbout);

/***
*
*    88                          88                          88
*    88                          88                          88
*    88                          88                          88
*    88           ,adPPYYba,   88,dPPYba,     ,adPPYba,   88  ,adPPYba,
*    88           ""      `Y8  88P'    "8a  a8P_____88   88  I8[    ""
*    88           ,adPPPPP88  88       d8  8PP"""""""   88   `"Y8ba,
*    88           88,     ,88  88b,   ,a8"  "8b,   ,aa   88  aa    ]8I
*    88888888888  `"8bbdP"Y8  8Y"Ybbd8"'    `"Ybbd8"'   88  `"YbbdP"'
*
*
*/

/***
* Create a JLabel "Server IP".
*
*     +-+-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+
*     |S|e|r|v|e|r| |I|P| |L|a|b|e|l|
*     +-+-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+
*
* With the JLabel class, you can display unselectable text and images.
*/

JLabel ServerIPLabel = new JLabel("Server IP:");
GridBagConstraints gbc_ServerIPLabel = new GridBagConstraints();
gbc_ServerIPLabel.anchor = GridBagConstraints.EAST;
gbc_ServerIPLabel.insets = new Insets(0, 0, 5, 5);
gbc_ServerIPLabel.gridx = 0;
gbc_ServerIPLabel.gridy = 0;
contentPane.add(ServerIPLabel, gbc_ServerIPLabel);

/***
* Create a JLabel "Port".
*     +-+-+-+-+ +-+-+-+-+-+
*     |P|o|r|t| |L|a|b|e|l|
*     +-+-+-+-+ +-+-+-+-+-+
*
* With the JLabel class, you can display unselectable text and images.
*/

JLabel PortLabel = new JLabel("Port: ");
GridBagConstraints gbc PortLabel = new GridBagConstraints();
gbc_PortLabel.anchor = GridBagConstraints.EAST;
gbc_PortLabel.insets = new Insets(0, 0, 5, 5);
gbc_PortLabel.gridx = 0;
gbc_PortLabel.gridy = 1;
```

```
contentPane.add(PortLabel, gbc_PortLabel);

/***
* Create a JLabel "Connected"/"Not connected" (dynamic). By default it's "Not
connected".
*      +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |C|o|n|n|e|c|t|e|d| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*
* With the JLabel class, you can display unselectable text and images.
*/

lblConnected = new JLabel("");
lblConnected.setForeground(red);
lblConnected.setText("Not connected");
GridBagConstraints gbc_lblConnected = new GridBagConstraints();
gbc_lblConnected.insets = new Insets(0, 0, 5, 5);
gbc_lblConnected.gridx = 0;
gbc_lblConnected.gridy = 2;
contentPane.add(lblConnected, gbc_lblConnected);

/***
*      +-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+-+
*      |M|e|s|s|a|g|e| |B|o|x| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+-+
*/

MessageBox = new JLabel("Message Box:");
GridBagConstraints gbc_MessageBox = new GridBagConstraints();
gbc_MessageBox.anchor = GridBagConstraints.NORTHEAST;
gbc_MessageBox.insets = new Insets(0, 0, 5, 5);
gbc_MessageBox.gridx = 0;
gbc_MessageBox.gridy = 3;
contentPane.add(MessageBox, gbc_MessageBox);
/***
*
*   888888888888                          88888888888  88            88          88
*        88                         ,d    88           ""            88          88
*        88                         88    88                         88          88
*        88   ,adPPYba,  8b,     ,d8  MM88MMM  88aaaaa      88   ,adPPYba,  88   ,adPPYb,88  ,adPPYba,  ,adPPYba,
*        88  a8P     88   `Y8, ,8P'     88     88"""""      88  a8P     88  88  a8"    `Y88  I8[    ""  I8[    ""
*        88  8PP"""""""     )888(       88     88           88  8PP"""""""  88  8b       88   `"Y8ba,    `"Y8ba,
*        88  "8b,   ,aa   ,d8" "8b,     88,    88           88  "8b,   ,aa  88  "8a,   ,d88  aa    ]8I  aa    ]8I
*        88   `"Ybbd8"'  8P'     `Y8    "Y888  88           88   `"Ybbd8"'  88   `"8bbdP"Y8  `"YbbdP"'  `"YbbdP"'
*
*
*/

/***
* Create a JTextField for Server's IP.
*      +-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+-+-+-+-+
*      |S|e|r|v|e|r| |I|P| |T|e|x|t|F|i|e|l|d|
*      +-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

ServerTextField = new JTextField();
ServerTextField.setEditable(false);
GridBagConstraints gbc_ServerTextField = new GridBagConstraints();
gbc_ServerTextField.anchor = GridBagConstraints.NORTH;
gbc_ServerTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_ServerTextField.insets = new Insets(0, 0, 5, 0);
gbc_ServerTextField.gridwidth = 3;
gbc_ServerTextField.gridx = 1;
gbc_ServerTextField.gridy = 0;
contentPane.add(ServerTextField, gbc_ServerTextField);
ServerTextField.setColumns(10);
ServerTextField.setText(getIP());

/***
* Create a JTextField for Server's Port.
*
*      +-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*      |P|o|r|t| |T|e|x|t|F|i|e|l|d|
*      +-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/
```

```
PortTextField = new JTextField();
PortTextField.setColumns(10);
GridBagConstraints gbc_PortTextField = new GridBagConstraints();
gbc_PortTextField.anchor = GridBagConstraints.NORTH;
gbc_PortTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_PortTextField.insets = new Insets(0, 0, 5, 0);
gbc_PortTextField.gridwidth = 3;
gbc_PortTextField.gridx = 1;
gbc_PortTextField.gridy = 1;
contentPane.add(PortTextField, gbc_PortTextField);

/***
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+
*     |M|e|s|s|a|g|e| |T|e|x|t|F|i|e|l|d|
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+
*/

MessageTextField = new JTextField();
MessageTextField.addActionListener(new ActionListener() {

/* Add an Action Listener to Message Text Field (enter keypress listener). */

public void actionPerformed(ActionEvent arg0) {
SendHandler.SetMessage(MessageTextField.getText());
}
});
MessageTextField.setEnabled(false);
GridBagConstraints gbc_textField = new GridBagConstraints();
gbc_textField.gridwidth = 2;
gbc_textField.insets = new Insets(0, 0, 0, 5);
gbc_textField.fill = GridBagConstraints.HORIZONTAL;
gbc_textField.gridx = 1;
gbc_textField.gridy = 4;
contentPane.add(MessageTextField, gbc_textField);
MessageTextField.setColumns(10);

/***
*
*    88888888ba
*    88      "8b               ,d      ,d
*    88      ,8P               88      88
*    88aaaaaa8P'  88      88  MM88MMM  MM88MMM  ,adPPYba,   8b,dPPYba,   ,adPPYba,
*    88""""""8b,  88      88    88       88    a8"     "8a  88P'   `"8a  I8[    ""
*    88      `8b  88      88    88       88    8b       d8  88       88   `"Y8ba,
*    88      a8P  "8a,   ,a88    88,      88,   "8a,   ,a8"  88       88  aa    ]8I
*    88888888P"    `"YbbdP'Y8    "Y888    "Y888  `"YbbdP"'   88       88  `"YbbdP"'
*
*
*/

/***
*
*     ____ ____ ____ ____ ____ _________ ____ ____ ____ ____ ____ ____
*     ||S |||t |||a |||r |||t |||       |||S |||e |||r |||v |||e |||r ||
*     ||__|||__|||__|||__|||__|||_______|||__|||__|||__|||__|||__|||__||
*     |/__\|/__\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

StartButton = new JButton("Start Server");

/* Add a tooltip to "Start Server" button. */

StartButton.setToolTipText("<html>Start a Server on this machine"
+ " with given port. The Server will be listening to this"
+ " port by the time this button is pressed. <br>You can "
+ "cancel anytime afterwards by press the \"Stop Server\" "
+ "button.</br><br></br> <br>An attempt to retrieve the external "
+ "IP(v4) of this <br>machine will be made. </br>\r\n<br></br>"
+ "\r\n<br></br>\r\nThe external IP (public Internet IP) along with"
+ " the machine's local IP (and hostname <br>most likely) will"
+ " be shown at Server IP TextField.</br>\r\n<br></br></html>");
GridBagConstraints gbc_StartButton = new GridBagConstraints();
gbc_StartButton.gridwidth = 2;
gbc_StartButton.anchor = GridBagConstraints.NORTH;
gbc_StartButton.fill = GridBagConstraints.HORIZONTAL;
gbc_StartButton.insets = new Insets(0, 0, 5, 5);
gbc_StartButton.gridx = 1;
gbc_StartButton.gridy = 2;
contentPane.add(StartButton, gbc_StartButton);
```

```
/* Add an action listener to the "Start Server" button. */

StartButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0){

/* Check if input is correct. */

if (PortTextField.getText() == null || PortTextField.getText().equals("") ||
PortTextField.getText().isEmpty()){
FullTextServer += "Please enter a valid port.\r\n";
textAreaServer.setText(FullTextServer);
return;
}
try {
port = Integer.parseInt(PortTextField.getText());
if (!((port > 1025) && (port < 65536))){
port = 0;
}
} catch (NumberFormatException e) {
port = 0;
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
if (port == 0){
FullTextServer += "Please set a valid port.\r\n";
textAreaServer.setText(FullTextServer);
return;
}

/* Check if port is in use. */

try {
if (isPortInUse(ip,port) == true){
FullTextServer += "Sorry, the port is in use.\r\n";
textAreaServer.setText(FullTextServer);
return;

/* If not then start the Server. */

}else{
startSingleConnection();

}
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
}
});

/***
*      ___ ___ ___ ___ ________ ___ ___ ___ ___ ___ ___
*     ||S |||t |||o |||p |||      |||S |||e |||r |||v |||e |||r ||
*     ||  |||  |||  |||  |||      |||  |||  |||  |||  |||  |||  ||
*     |/__\|/__\|/__\|/__\|/______\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

StopButton = new JButton("Stop Server");

/* Add a tooltip to "Stop Server" button. */

StopButton.setToolTipText("<html>Stop listening to the given port."
+ "\r\n<br></br>\r\n<br></br>\r\nWarning: This will stop "
+ "ALL current actions! Not recommended to use if other actions "
+ "<br>are in progress.<br><br></br> Use with caution.</br>\r\n"
+ "<br></br>");
StopButton.setEnabled(false);
```

```
GridBagConstraints gbc_StopButton = new GridBagConstraints();
gbc_StopButton.anchor = GridBagConstraints.NORTH;
gbc_StopButton.fill = GridBagConstraints.HORIZONTAL;
gbc_StopButton.insets = new Insets(0, 0, 5, 0);
gbc_StopButton.gridx = 3;
gbc_StopButton.gridy = 2;
contentPane.add(StopButton, gbc_StopButton);

/* Add an action listener to the "Stop Server" button. */

StopButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
try {
stopSingleConnection(false);
} catch (IOException e3) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e3.toString(),false);
}
} catch (IOException e) {
e.printStackTrace();
}
}
mntmpercentage.setEnabled(false);
mntmSendNow.setEnabled(false);
StartButton.setEnabled(true);
StopButton.setEnabled(false);
lblConnected.setForeground(red);
lblConnected.setText("Not connected");
connected = false;
try {
if (connectionSocket != null){
connectionSocket.close();
connectionSocket = null;
}
} catch (IOException e1) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
try {
if (welcomeSocket != null){
welcomeSocket.close();
welcomeSocket = null;
}
} catch (IOException e1) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
};
}
FullTextServer += "Server stopped.\r\n";
textAreaServer.setText(FullTextServer);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append("Server stopped.",true);
}
} catch (IOException e) {
e.printStackTrace();
}
if (!((IP == null || IP.equals("") || IP.isEmpty()))){
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(IP + " kicked.",true);
IP = null;
}
} catch (IOException e) {
e.printStackTrace();
}
FullTextServer = FullTextServer + IP + " kicked.\r\n";
```

```
textAreaServer.setText(FullTextServer);
}
}
});

/***
*
*    ____ ____ ____ ____ ________ ____ ________ ____ ____ ____ ____ ____ ____
*    ||S |||e |||n |||d |||      |||a |||      |||F |||o |||l |||d |||e |||r ||
*    ||__|||__|||__|||__|||______|||__|||______|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/______\|/__\|/______\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

FolderButton = new JButton("Send a Folder");

/* Add a tooltip to "Send a Folder" button. */

FolderButton.setToolTipText("<html> Send a folder to the client"
+ " from this machine. \r\n<br></br>\r\n<br> A file chooser"
+ " will pop up. </br> \r\n<br></br>\r\n<br> Note: You"
+ " <u>won't</u> be sending mutiple files consecutivly"
+ " -</br?\r\n<br> </br>\r\n<br> Instead, the folder "
+ "contents (not recursivly, only non-folders) will be"
+ " zipped,</br>\r\n<br>and will be send to the client"
+ " as a single folder in one single connection. </br>");

/* Add an Action Listener to "Send a Folder" button. */

FolderButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
final JFileChooser chooser = new JFileChooser() {
public void approveSelection() {
if (getSelectedFile().isFile()) {
return;
} else
super.approveSelection();
}
};
chooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
int returnVal = chooser.showOpenDialog(getParent());
if(returnVal == JFileChooser.APPROVE_OPTION) {
File f = chooser.getSelectedFile();
String zipFileName = chooser.getSelectedFile().getName();
File[] listOfFiles = f.listFiles();
List<String> files = new ArrayList<String>();
for (File file : listOfFiles) {
if (file.isFile()) {
files.add(file.getAbsolutePath());
}
}
zipFiles(files,zipFileName);
String filename = zipFileName + ".zip";
String absolute = "./" + filename;
DataOutputStream outToClient = null;
try {
outToClient = new DataOutputStream(connectionSocket.getOutputStream());
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
try {
outToClient.writeBytes("#" + filename +"\r\n");
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
try {
outToClient.flush();
```

```
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
Thread thread = new Thread(new ServerSendHandle(IP,port+1,filename,absolute));
threads.add(thread);
thread.start();
}
}
});
FolderButton.setEnabled(false);
GridBagConstraints gbc_FolderButton = new GridBagConstraints();
gbc_FolderButton.insets = new Insets(0, 0, 5, 0);
gbc_FolderButton.anchor = GridBagConstraints.SOUTHWEST;
gbc_FolderButton.gridx = 3;
gbc_FolderButton.gridy = 3;
contentPane.add(FolderButton, gbc_FolderButton);

/***
*     ____ ____ ____ ____ ________ ____ ________ ____ ____ ____ ____
*    ||S |||e |||n |||d |||      |||a |||      |||F |||i |||l |||e ||
*    ||  |||  |||  |||  |||      |||  |||      |||  |||  |||  |||  ||
*    ||__\||__\||__\||__\||______\||__\||______\||__\||__\||__\||__\|
*/

SendButton = new JButton("Send a File");

/* Add a tooltip to "Send a File" button. */

SendButton.setToolTipText("<html> Send a file to the client from this machine."
+ " \r\n<br></br>\r\n<br> A file chooser will pop up. </br> ");
SendButton.setEnabled(false);

/* Add an Action Listener to "Send a File" button. */

SendButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
JFileChooser chooser = new JFileChooser();
int returnVal = chooser.showOpenDialog(getParent());
if(returnVal == JFileChooser.APPROVE_OPTION) {
String filename = chooser.getSelectedFile().getName();
String absolute = chooser.getSelectedFile().getAbsolutePath();
try {
DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
outToClient.writeBytes("#" + filename +"\r\n");
outToClient.flush();
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
Thread thread = new Thread(new ServerSendHandle(IP,port+1,filename,absolute));
threads.add(thread);
thread.start();

}
}
});
GridBagConstraints gbc_SendButton = new GridBagConstraints();
gbc_SendButton.anchor = GridBagConstraints.NORTHWEST;
gbc_SendButton.insets = new Insets(0, 0, 0, 5);
gbc_SendButton.gridwidth = 2;
gbc_SendButton.gridx = 0;
gbc_SendButton.gridy = 4;
contentPane.add(SendButton, gbc_SendButton);

/***
```

```
*
*     ____ ____ ____ ____ ________ ____ ____ ____ ____ ____ ____ ____
*    ||S |||e |||n |||d |||      |||M |||e |||s |||s |||a |||g |||e ||
*    ||__|||__|||__|||__|||______|||__|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/______\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

SendMessageButton = new JButton("Send");

/* Add a tooltip to "Send" button. */

SendMessageButton.setToolTipText("Send a message (assuming non-null string in Message
TextField).");

/* Add an Action Listener to "Send" button. */

SendMessageButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
SendHandler.SetMessage(MessageTextField.getText());
}
});
SendMessageButton.setEnabled(false);
GridBagConstraints gbc_btnSend = new GridBagConstraints();
gbc_btnSend.anchor = GridBagConstraints.WEST;
gbc_btnSend.gridx = 3;
gbc_btnSend.gridy = 4;
contentPane.add(SendMessageButton, gbc_btnSend);

/***
*
*   888888888888                                  db
*        88                          ,d           d88b
*        88                          88          d8''8b
*        88  ,adPPYba,  8b,     ,d8  MM88MMM       d8'  `8b     8b,dPPYba,   ,adPPYba,  ,adPPYYba,
*        88  a8P_____88   `Y8, ,8P'    88         d8YaaaaY8b    88P'   "Y8  a8P_____88  ""     `Y8
*        88  8PP"""""""     )888(      88        d8""""""""8b   88          8PP"""""""  ,adPPPPP88
*        88  "8b,   ,aa   ,d8" "8b,    88,      d8'        `8b  88          "8b,   ,aa  88,    ,88
*        88   `"Ybbd8"'  8P'     `Y8   "Y888   d8'          `8b 88           `"Ybbd8"'  `"8bbdP"Y8
*
*
*/

/***
*      +-+-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+
*      |M|e|s|s|a|g|e| |B|o|x| |A|r|e|a|
*      +-+-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+
*/

/* Make Message Box scrollable. */

scrollPane = new JScrollPane();
GridBagConstraints gbc_scrollPane = new GridBagConstraints();
gbc_scrollPane.gridwidth = 2;
gbc_scrollPane.fill = GridBagConstraints.BOTH;
gbc_scrollPane.insets = new Insets(0, 0, 5, 5);
gbc_scrollPane.gridx = 1;
gbc_scrollPane.gridy = 3;
contentPane.add(scrollPane, gbc_scrollPane);

textAreaServer = new JTextArea();

/* Make Message Box uneditable. */

textAreaServer.setEditable(false);

scrollPane.setViewportView(textAreaServer);
}

/**
*
* Instantiate a new ServerSettings class (please see the ServerSettings class)
* By doing this the program's settings will be running as intended.
*
* @param frame The Server's main class (frame).
*/

public static void instantiateServerSettings(Server frame){
serversettings = new ServerSettings();
try {

/* Check if a file with settings already exists*/
```

```
if (ServerSettings.DoesFileExist()) {

/* Check if the settings are in correct format */

if (ServerSettings.ParseExistingFile(serversettings) == false){

/* If they are incorrect, delete the file and make a file with new (default) settings
*/

File file = new File("./" + serversettings.getServerFileName());
file.delete();
serversettings.makeFile(serversettings);
}

/* If no such file exists, make a new file with new (default) settings */

}else{
serversettings.makeFile(serversettings);
}

/* Throw exception in case of error */

} catch (IOException e2) {

/* Throw error message dialog in case of error           *
* and terminate the program (settings file is mandatory */

JOptionPane.showMessageDialog(null,
"Cannot write ServerSettings.cfg\nAccess denied.\n.",
"Write Error",
JOptionPane.ERROR_MESSAGE);
frame.setVisible(false);
frame.dispose();
e2.printStackTrace();
}
}

/**
*
* Check if logging is enabled in settings
* and instantiate a new log file otherwise do nothing
*
*/

public static void instantiateServerLogging(){

if (serversettings.isServerLoggingEnabled() == true){
log = new ServerLogging();
try {
log.makeLog();
} catch (IOException e) {
/* Throw error message dialog in case of error */
JOptionPane.showMessageDialog(null,
"Cannot write ServerLog.log\nAccess denied.\n.",
"Write Error",
JOptionPane.ERROR_MESSAGE);
e.printStackTrace();
}
}
}

/**
*
* Attempt to get the external IP of the device
* otherwise return the local IP.
*
*  @return The Server's IP
*/

public static String getIP(){
InetAddress Local_IP = null;
try {

/* icanhazip.com --> returns your IP address */

URL whatismyip = new URL("http://icanhazip.com/");
URLConnection connection = whatismyip.openConnection();
```

```
connection.addRequestProperty("Protocol", "Http/1.1");
connection.addRequestProperty("Connection", "keep-alive");
connection.addRequestProperty("Keep-Alive", "1000");
connection.addRequestProperty("User-Agent", "Web-Agent");

BufferedReader in =
new BufferedReader(new InputStreamReader(connection.getInputStream()));

/* You get the IP as a String */

ip = in.readLine();
/* If unable to retrieve external (public) IP then */
} catch (UnknownHostException | MalformedURLException e1) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e1.toString(),false);
}
} catch (IOException e2) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e2.toString(),false);
}
} catch (IOException e5) {
e5.printStackTrace();
}
}

/* Use hostname*/

try {
Local_IP = InetAddress.getLocalHost();
} catch (UnknownHostException e4) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e4.toString(),false);
}
} catch (IOException e3) {
e3.printStackTrace();
}
}

/* Get local IP as a String */

ip = Local_IP.getHostAddress();

/* If unable to retrieve external (public) IP then */

} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e2) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e2.toString(),false);
}
} catch (IOException e5) {
e5.printStackTrace();
}
}

/* Use hostname */

try {
Local_IP = InetAddress.getLocalHost();
} catch (UnknownHostException e4) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e4.toString(),false);
}
} catch (IOException e3) {
e3.printStackTrace();
}
}
ip = Local_IP.getHostAddress();
}
```

```
/* Also get the local IPv4 */

String last = "";
try {
String ip2 = InetAddress.getLocalHost().toString();
Local_IP = InetAddress.getLocalHost();
if (ip == Local_IP.getHostAddress()){
last = ip2;
}else{
last = ip + " (or " + ip2 + ")";
}
} catch (UnknownHostException e2) {
e2.printStackTrace();
}

/* Return either the external ipv4 + hostname + local ipv4   *
* or hostname + local ipv4
        */

return last;
}

/**
*
* Start the connection on a separate thread (Swing Thread)
* @throws IOException In case of error.
*/

public static void startSingleConnection() throws IOException {

/* Enable the appropriate buttons */

StopButton.setEnabled(true);
StartButton.setEnabled(false);

/* Write the appropriate message to the message box */

FullTextServer = FullTextServer + "Server started at: " + ip + ":" + port + "\r\n";
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append("Server started at: " + ip + ":" + port,true);
}
} catch (IOException e) {
e.printStackTrace();
}
textAreaServer.setText(FullTextServer);

/* Meanwhile start the Server (through StartServer method) in the background */

worker = new SwingWorker<Integer, String>() {
protected Integer doInBackground() throws Exception {
welcomeSocket = new ServerSocket(port);
StartServer();
return 1;
}
protected void done(){
try {
} catch (CancellationException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
return;
}
}

/*
* (non-Javadoc)
* Here we receive the values that we publish().
* They may come grouped in chunks.
* @see javax.swing.SwingWorker#process(java.util.List)
*/
```

```
protected void process(List<String> chunks) {
for (final String string : chunks) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(string,true);
}
} catch (IOException e) {
e.printStackTrace();
}
FullTextServer = FullTextServer + string + "\r\n";
textAreaServer.setText(FullTextServer);
}
}
};
/* Start the thread */
worker.execute();
}

/**
*
* Stops the server and all current (either running or not) threads.
*
*  @param restart Specify whether we just stop the server or stop and restart it
*  @throws IOException In case of error.
*/

public static void stopSingleConnection(Boolean restart) throws IOException{

/* Stop every single thread from the thread arraylist */

for (int i = 0; i < threads.size(); i++) {
((Thread) threads.get(i)).interrupt();
}

/* Clear (empty) the thread arraylist */

threads.clear();

/* Stop the webcam thread (executor/task) as well (only if it's scheduled at the
moment) */

if (task1 != null){
task1.cancel(true);
task1 = null;
}

/* Enable/disable the appropriate buttons and fields */

SendMessageButton.setEnabled(false);
FolderButton.setEnabled(false);
SendButton.setEnabled(false);
MessageTextField.setEnabled(false);
worker.cancel(true);

/* Set the label to "Not connected" */

lblConnected.setForeground(red);
lblConnected.setText("Not connected");

/* Make the boolean connected false */

connected = false;

/* Restart the server if restart boolean variable is true */

if (restart == true){
startSingleConnection();
}
}

/**
*
* The snapshot process (runnable)
*
*  @return restart the process/(to be) scheduled task
*/
```

```
public static Runnable rep(){

/* If the Runnable is NOT initialized. */

if (runnable1 == null){
return new Runnable(){
public void run(){

/* If snapshots in settings are enabled. */

if (serversettings.isServerSnappingEnabled() == true){

/* Initialize webcam and image variables. */

Webcam webcam = null;
BufferedImage image = null;
try{

/* Get default webcam. */

webcam = Webcam.getDefault();

/* If locked unlock. */

if (webcam.getLock().isLocked()) {
webcam.getLock().unlock();
}

/* if still locked cancel the task. */

if (webcam.getLock().isLocked() == true){
task1.cancel(true);
}

/* Set an acceptable dimension (640x480). */

webcam.setViewSize(new Dimension(640, 480));

/* If it isn't open then open it. */

if (webcam.isOpen() == false){
webcam.open();
}

/* "Sleep" the thread -- workaround for "fast" (modern) processors. */

Thread.sleep(100);

/* Get image from webcam. */

image = webcam.getImage();
Thread.sleep(100);

/* Close the webcam. */

webcam.close();
Thread.sleep(100);

/* Any exception leads to thread closure. */

} catch (WebcamException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
task1.cancel(true);
}
if (image == null){
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append("Failed to take a snapshot",true);
}
} catch (IOException e1) {
e1.printStackTrace();
```

```
}
task1.cancel(true);
}

/* Name the webcam snapshot in a SnapshotXXXXX (numeric prefix) format. */

String formatted = String.format("%05d", SnapshotIndexing);
try {

/* Set filename and absolute path. */

String absolute = serversettings.getServerSnapshotFolder().replace("\\",
File.separator);;
String filename = "Snapshot" + formatted + ".jpg";

/* Increment the snapshot index (number of snapshots). */

SnapshotIndexing++;

/* Write the image to an image file. */

ImageIO.write(image, "JPG", new File(absolute+filename));

/* If snapshot before last (previous snapshot) is null. */

if (snapshot_before_last == null){

/* Consider both snapshots (last and previous) to be the same. */

snapshot_before_last = filename;
last_snapshot = filename;
}else{

/* Else if they aren't the same. */

Thread.sleep(100);
Boolean diff = ((snapshot_before_last.equals(last_snapshot)) == true);
String temp = last_snapshot;
String temp2 = snapshot_before_last;
last_snapshot = filename;
snapshot_before_last = temp;

/* Delete the snapshot before before last snapshot. */

if (diff == false){
File f = new File(absolute+temp2);
if (f.exists()){
f.delete();
}
}
}

/* In case of exception decrement the snapshot index. */

} catch (IOException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
SnapshotIndexing--;
}

/* If there are two snapshots, print the percentage to the client (if it exceeds the
given %). */

if ((snapshot_before_last != null) && (snapshot_before_last != last_snapshot)){
double p = percentage();
if (p > serversettings.getServerpercentage_Warning()){
try {
Thread.sleep(100);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(p+"% was noticed.",true);
}
```

```
} catch (IOException e1) {
e1.printStackTrace();
}
FullTextServer += p+"% was noticed.\r\n";
textAreaServer.setText(FullTextServer);
DataOutputStream outToClient = new
DataOutputStream(Server.connectionSocket.getOutputStream());
outToClient.writeBytes("&" + p+"\r\n");
outToClient.flush();
} catch (IOException | InterruptedException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
}
}
}
};

/* Else just return the current runnable. */

}else{
return runnable1;
}
}

/**
*
* Starts the Server.
* The socket listens to the given port.
* And all the needed procedures are awaiting to be executed upon client's successful
connection.
*
*/

@SuppressWarnings("static-access")
public static void StartServer(){
try {

/* Listen to port given previously. */

connectionSocket = welcomeSocket.accept();

/* Every single line (in this method) is executed only AFTER
* a client has successfully connected to this listening socket.
*/

/* Set the boolean to true.. this boolean variable helps to indicate
* whether the client has joined (is currently connected) or not.
*/

connected = true;

/* Helps to keep the connection alive. */

connectionSocket.setKeepAlive(true);

/* Start the webcam snapshot taking task. */

executor = Executors.newScheduledThreadPool(5);
task1 = executor.scheduleAtFixedRate(rep(),0,
serversettings.getServerSnapping_seconds(), TimeUnit.SECONDS);

/* Time is initially 0. */

int time = 0;
if (serversettings.isServerSnappingEnabled() == true){
time = serversettings.getServerSnapping_seconds();
}

/* Set a timeout so if no data is sent or received in
* webcam snapshot seconds 1000 seconds then shut down the connection to the client
```

```
* and restart the server.
*/

connectionSocket.setSoTimeout(time+1000000);
welcomeSocket.setSoTimeout(time+1000000);

/* In case of exception (time-out et.c.) */

} catch (SocketTimeoutException s) {

/* Set the label to "Not connected". */

lblConnected.setForeground(red);
lblConnected.setText("Not connected");

/* Set the boolean to false (explained above). */

Server.connected = false;
Server.FullTextServer +=  IP + " disconnected.\r\n";
Server.textAreaServer.setText( Server.FullTextServer);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(IP + " disconnected.",true);
}
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
try {
if ( Server.connectionSocket != null){
Server.connectionSocket.close();
Server.connectionSocket = null;
Server.welcomeSocket.close();
Server.welcomeSocket = null;
Server.stopSingleConnection(true);
}
} catch (IOException e1) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}

}
/* Same procedure. */
} catch (SocketException e) {
lblConnected.setForeground(red);
lblConnected.setText("Not connected");
Server.connected = false;
Server.FullTextServer +=  IP + " disconnected.\r\n";
Server.textAreaServer.setText( Server.FullTextServer);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append( IP + " disconnected.",true);
}
} catch (IOException e9) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e9.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
try {
if ( Server.connectionSocket != null){

Server.connectionSocket.close();
Server.connectionSocket = null;
Server.welcomeSocket.close();
```

```
Server.welcomeSocket = null;
Server.stopSingleConnection(true);
}
} catch (IOException e1) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}

}
/* Same procedure. */
} catch (IOException e) {
lblConnected.setForeground(red);
lblConnected.setText("Not connected");
Server.connected = false;
Server.FullTextServer +=  IP + " disconnected.\r\n";
Server.textAreaServer.setText( Server.FullTextServer);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append( IP + " disconnected.",true);
}
} catch (IOException e8) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e8.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
try {
if ( Server.connectionSocket != null){

Server.connectionSocket.close();
Server.connectionSocket = null;
Server.welcomeSocket.close();
Server.welcomeSocket = null;
Server.stopSingleConnection(true);
}
} catch (IOException e1) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}

}
}

/* If this line is reached, then the connection was successful. */

/* Set label text to connected. */

lblConnected.setForeground(green);
lblConnected.setText("Connected");

/**
* (non-Javadoc)
* Create a new thread which consists a new Server Input Handler (explained in the
* perspective class)
* @see ftp.ServerInputHandler
*/

Thread thread = new Thread(new ServerInputHandler(connectionSocket));

/* Add it to the Thread Arraylist.*/

threads.add(thread);

/* Start the thread */

thread.start();
```

```
/* Do the same for the output handler. */

SendHandler = new ServerOutputHandler(connectionSocket);
Thread thread2 = new Thread(SendHandler);
threads.add(thread2);
thread2.start();

/* Get the IP of the client and print it. */

String IP_old = connectionSocket.getInetAddress().toString();
IP = IP_old.substring(1);
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(IP + " connected",true);
}
} catch (IOException e) {
e.printStackTrace();
}
FullTextServer = FullTextServer + IP + " connected.\r\n";
textAreaServer.setText(FullTextServer);

/* Enable/disable the appropriate buttons and fields. */

SendButton.setEnabled(true);
FolderButton.setEnabled(true);
SendMessageButton.setEnabled(true);
SendButton.setEnabled(true);
MessageTextField.setEnabled(true);
mntmpercentage.setEnabled(true);
mntmSendNow.setEnabled(true);
}

/**
* Creates a new zip archive from a list of files.
*
* @param files The list of files to be zipped
* @param filename The file of the (zip) archive
*/

public static void zipFiles(List<String> files, String filename){

/* Initialize variables. */

FileOutputStream fos = null;
ZipOutputStream zipOut = null;
FileInputStream fis = null;
try {

/* Create a new instance of FileOutputStream. */

fos = new FileOutputStream("./" + filename + ".zip");

/* Create a new instance of ZipOutputStream using  the FileOutputStream. */

zipOut = new ZipOutputStream(new BufferedOutputStream(fos));

/* For every single file, add it to the archive */

for(String filePath:files){
File input = new File(filePath);
fis = new FileInputStream(input);
ZipEntry ze = new ZipEntry(input.getName());
zipOut.putNextEntry(ze);
byte[] tmp = new byte[4*1024];
int size = 0;
while((size = fis.read(tmp)) != -1){
zipOut.write(tmp, 0, size);
}
zipOut.flush();
fis.close();
}

/* Close the ZipOutputStream */

zipOut.close();
} catch (FileNotFoundException e) {
```

```
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
} finally{
/* Close the FileOutputStream regardless of exceptions and errors. */
try{
if(fos != null) fos.close();
} catch(Exception ex){
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(ex.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
}
}

/**
* Calculate the percentage difference.
*
* @return The percentage difference.
*/

public static double percentage(){

/* Initialize images. */

BufferedImage img1 = null;
BufferedImage img2 = null;
try {

/* Read both files (images - last two taken snapshots). */

File img_a = new File(last_snapshot);
File img_b = new File(snapshot_before_last);
if ((last_snapshot == null) || (snapshot_before_last == null)){
return (double)-1;
}
img1 = ImageIO.read(img_a);
img2 = ImageIO.read(img_b);
} catch (IOException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}

/* Calculate the difference */

int width1 = img1.getWidth(null);
int width2 = img2.getWidth(null);
int height1 = img1.getHeight(null);
int height2 = img2.getHeight(null);

/* Dimensions must be the same (just in case). */

if ((width1 != width2) || (height1 != height2)) {
return (double)-1;
}
long diff = 0;
```

```
/* Calculate color offset difference for every single pixel. */

for (int y = 0; y < height1; y++) {
for (int x = 0; x < width1; x++) {
int rgb1 = img1.getRGB(x, y);
int rgb2 = img2.getRGB(x, y);
int r1 = (rgb1 >> 16) & 0xff;
int g1 = (rgb1 >>  8) & 0xff;
int b1 = (rgb1      ) & 0xff;
int r2 = (rgb2 >> 16) & 0xff;
int g2 = (rgb2 >>  8) & 0xff;
int b2 = (rgb2      ) & 0xff;
diff += Math.abs(r1 - r2);
diff += Math.abs(g1 - g2);
diff += Math.abs(b1 - b2);
}
}
double n = width1 * height1 * 3;
double p = diff / n / 255.0;

/* Return the percetange % . */

return p * 100.0;
}

/**
* Returns whether the port is in use or not.
*
* @param host The "owner" (= machine running) of the port to be checked.
* @param port The port to be checked.
* @return If port is in use.
* @throws IOException
*/

private static boolean isPortInUse(String host, int port) throws IOException {
/* Assume no connection is possible. */
boolean result = false;
try {
(new Socket(host, port)).close();
result = true;
}catch(SocketException e) {
try {
if (serversettings.isServerLoggingEnabled() == true){
log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}

return result;
}

/**
* Changes the time of snapshot to be sent/taken/checked.
*
* @param time The new scheduling time (in seconds).
*/

public static void changeReadInterval(long time){
if(time > 0){
if (task1 != null){
task1.cancel(true);
}
task1 = executor.scheduleAtFixedRate(rep(), 0, time, TimeUnit.SECONDS);
}
}
}

/**
* SocketSnapshot's Server's Input Handler (sub-)class
*
* Note: Because this class works side-by-side with Server class,
* it's not set "public" (thus no separate class file is required).
*
* @author  Paulina Barniadaki
```

```
* @version 2.0
* @since   2015-06-01
*/

class ServerInputHandler implements Runnable {
private Socket socket;

/**
* Constructor of ServerInputHandler.
* @param socket Initialize the socket with the given socket.
*/

public ServerInputHandler(Socket socket) {
this.socket = socket;
}

/**
* The running (thread) method.
*/

@SuppressWarnings("static-access")
public void run() {
BufferedReader reader = null;
Boolean error = false;
ArrayList <String> errors = new ArrayList<String>();
try {

/* The Input Stream (single line input with CRLF feed) from the given socket. */

reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
} catch (IOException e2) {
error = true;
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e2.toString(),false);
}
} catch (IOException e) {
e.printStackTrace();
}
}

/* All client messages are being "caught" in this loop.
* It can only be terminated from elsewhere.          */

while ((Server.connected == true) && (error == false)) {

/* Initially the client message is null. */

String text = "";

/* If connection is still established. */

if (Server.connected == true){
try {

/* Attempt to read a line with CRLF (from the client */

text = reader.readLine();

/* If it's not a command (see commands) then print it */

if ((text.length() > 0) && ((text.charAt(0) != '#') && (text.charAt(0) != '!') &&
(text.charAt(0) != '~'))){
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("#Client said: " + text,true);
}
} catch (IOException e) {
e.printStackTrace();
}
Server.FullTextServer = Server.FullTextServer + "#Client said: " + text + "\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
continue;
}
/* Catch I/O errors */
} catch (IOException e1) {
error = true;
```

```
errors.add(e1.toString());
/* Stop the loop if error (exception) is caught. */
}finally{
if (error == true){
break;
}
}
}

/* If the text is null don't do anything. */

if ((text == null || text.equals("") || text.isEmpty())){
continue;
}else{

/* File list command was given from client */

if (text.charAt(0) == '#'){
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("Sending the requested filelist...",true);
}
} catch (IOException e) {
e.printStackTrace();
}
Server.FullTextServer = Server.FullTextServer + "Sending the requested
filelist...\r\n";
Server.textAreaServer.setText(Server.FullTextServer);

/* Create a new list of files (initially empty) */

List<String> results = new ArrayList<String>();

/* Get the list with files (in File format, not string which is required). */

File[] files = new File("./").listFiles();

//If this pathname does not denote a directory, then listFiles() returns null.

@SuppressWarnings("unused")

/* Parse all the files in the "files" file array. */

int i = 0;
for (File file : files) {

/* If it's a file (and not a folder) */

if (file.isFile()) {
i++;
results.add(file.getName());
/* Else if it's a folder then add the prefix "<Folder>" to the file along with the
filename */
}else{
results.add("<Folder> "+ file.getName());
}
}
DataOutputStream outToClient = null;
try {
outToClient = new DataOutputStream(Server.connectionSocket.getOutputStream());
} catch (IOException e1) {
error = true;
errors.add(e1.toString());
}

/* Send the whole (String) File list to the client (one at a time) */

for (int i1 = 0; i1 < results.size(); i1++){
try {
Thread.sleep(100);
outToClient.writeBytes("|" + results.get(i1) +"\r\n");
outToClient.flush();
} catch (IOException | InterruptedException e) {
error = true;
errors.add(e.toString());;
}
```

```
}
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("Filelist sent succesfully!",true);
}
} catch (IOException e) {
e.printStackTrace();
}
Server.FullTextServer = Server.FullTextServer + "Filelist sent succesfully!\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
continue;

/* File get command was requested from the client. */

}else if (text.charAt(0) == '!'){

/* If a "<" is met at the first character of client's message
* then most likely it's a folder that it was requested to be sent. */

if (text.charAt(1) == '<'){

/* Remove the prefix "<Folder>" */

String fff = "<Folder>";
if (text.indexOf(fff) == -1){
return;
}
int pos = text.indexOf('>');

/* And save the rest of the String to a new String. */

String substring = text.substring(pos+2);

/* Zip the folder contents (as explained in the perspective method). */

File f = new File(substring);
String zipFileName = f.getName();
File[] listOfFiles = f.listFiles();
List<String> files = new ArrayList<String>();
for (File file : listOfFiles) {
if (file.isFile()) {
files.add(file.getAbsolutePath());
}
}
Server.zipFiles(files,zipFileName);

/* Send the zipped archive. */

String filename = zipFileName + ".zip";
String absolute = "./" + filename;
DataOutputStream outToClient = null;
try {
outToClient = new DataOutputStream(Server.connectionSocket.getOutputStream());
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}

/* Notify the client that a file is a to be sent. */

try {
outToClient.writeBytes("#" + filename +"\r\n");
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
try {
```

```
outToClient.flush();
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}

/* Initialize a new Server File Send Handle, add it to the perspective
* array-list and start the thread (thus sending the file to the client).
*/

Thread thread = new Thread(new
ServerSendHandle(Server.IP,Server.port+1,filename,absolute));
Server.threads.add(thread);
thread.start();
return;
}

/* Otherwise do the same for a non-folder (skipping zipping) */

String filename = text.substring(1,text.length());
File f = new File(filename);
String absolute = f.getAbsolutePath();
try {
DataOutputStream outToClient = new
DataOutputStream(Server.connectionSocket.getOutputStream());
outToClient.writeBytes("#" + filename +"\r\n");
outToClient.flush();
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("Attempting to send " + filename,true);
}
} catch (IOException e) {
e.printStackTrace();
}
Server.FullTextServer =  Server.FullTextServer + "Attempting to send " + filename +
"\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
Thread thread = new Thread(new
ServerSendHandle(Server.IP,Server.port+1,filename,absolute));
Server.threads.add(thread);
thread.start();
continue;

/* If setting configuration command received from client. */

}else if (text.charAt(0) == '~'){

/* If the text size is bigger than 1. */

if (text.length() > 1){

/* If it's "~se" then it means it's Snapshots Enabled */

if (text.equalsIgnoreCase("~se")){
Server.serversettings.setServerSnapping(true);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append( "Setting recieved from client: Snapshots enabled",true);
}
} catch (IOException e) {
e.printStackTrace();
}
```

```
Server.FullTextServer = Server.FullTextServer + "Setting recieved from client:
Snapshots enabled\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
try {
Server.serversettings.makeFile(Server.serversettings);
} catch (IOException e) {
error = true;
errors.add(e.toString());;
}
continue;

/* If it's "~sd" then it means it's Snapshots Disabled */

}else if (text.equalsIgnoreCase("~sd")){
Server.serversettings.setServerSnapping(false);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append( "Setting recieved from client: Snapshots disabled",true);
}
} catch (IOException e) {
e.printStackTrace();
}
Server.FullTextServer = Server.FullTextServer + "Setting recieved from client:
Snapshots disabled\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
try {
Server.serversettings.makeFile(Server.serversettings);
} catch (IOException e) {
e.printStackTrace();
}
continue;

/* If it's "~w" then it means we have to change the default warning percentage
* to the client's requested percetange. */

}else if (text.substring(0,2).equalsIgnoreCase("~w")){

/* A lot of type-casting is required. */

double number = (double)0;
try {
number = Double.parseDouble(text.substring(2));
if (!((number >= (double)0) && (number <= (double)100))){
continue;
}
}catch(NumberFormatException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
continue;
}
if (Server.serversettings.isServerSnappingEnabled() == true){
Server.serversettings.setServerpercentage_Warning(number);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append( "Setting recieved from client: Warn me if " +
Server.serversettings.getServerpercentage_Warning() + "% difference is noticed",true);
}
} catch (IOException e) {
e.printStackTrace();
}
Server.FullTextServer = Server.FullTextServer + "Setting recieved from client: Warn me
if " + Server.serversettings.getServerpercentage_Warning() + "% difference is
noticed\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
try {
Server.serversettings.makeFile(Server.serversettings);
} catch (IOException e) {
error = true;
errors.add(e.toString());;
}
continue;
}
```

```
}

/* If none of the above commands are met,
* then it means that we have to change Snapshot Seconds (if we get a number
~[1,2000)). */

/* Check if there's a number after the tilde ("~"). */

int number;
try {
number = Integer.parseInt(text.substring(1));
if (!((number >= 1) && (number < 2000))){
continue;
}
}catch(NumberFormatException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}
continue;
}

/* If it is then change the setting. */

if (Server.serversettings.isServerSnappingEnabled() == true){
Server.serversettings.setServerSnapping_seconds(number);
Server.changeReadInterval(number);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("Setting recieved from client: Snapshot every " +
ServerSettingsPopup.ServergetTIME(Server.serversettings.getServerSnapping_seconds()),t
rue);
}
} catch (IOException e) {
error = true;
errors.add(e.toString());;
}
Server.FullTextServer = Server.FullTextServer + "Setting recieved from client:
Snapshot every " +
ServerSettingsPopup.ServergetTIME(Server.serversettings.getServerSnapping_seconds()) +
"\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
try {
Server.serversettings.makeFile(Server.serversettings);
} catch (IOException e) {
error = true;
errors.add(e.toString());;
}
continue;
}
}
}
}
}

/* Every single line (in this (sub-)class only) after this means that the main loop is
stopped. */

/* A little house-keeping */

if (reader != null){
try {
reader.close();
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
reader = null;
}
```

```
/* If client disconnected prematurely then notify everyone and restart the server. */

for (int i = 0; i < errors.size(); i++){
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(errors.get(i),false);
}
} catch (IOException e) {
e.printStackTrace();
}
}
Server.connected = false;
Server.FullTextServer +=  Server.IP + " disconnected.\r\n";
Server.textAreaServer.setText( Server.FullTextServer);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(Server.IP + " disconnected.",true);
}
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
try {
if ( Server.connectionSocket != null){
Server.connectionSocket.close();
Server.connectionSocket = null;
Server.welcomeSocket.close();
Server.welcomeSocket = null;
Server.stopSingleConnection(true);
}
} catch (IOException e1) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}

}
}
}

/**
* SocketSnapshot's Server's File Sending Handler (sub-)class.
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

class ServerSendHandle implements Runnable {
private String IP;
private int port2;
private String filename;
private String abs;

/**
* Constructor of Server's File Sending Handler.
* @param IP The IP of the client.
* @param port2 The port of the Client (port+1)
* @param filename The filename of the file to be sent.
* @param abs The absolute path of the file.
*/

public ServerSendHandle(String IP,int port2,String filename,String abs) {
this.IP = IP;
this.port2 = port2;
this.filename = filename;
this.abs = abs;
}
```

```
/**
*  The runnable.
*/

public void run() {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("Attempting to send " + filename,true);
}
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
Server.FullTextServer =  Server.FullTextServer + "Attempting to send " + filename +
"...\r\n";
Server.textAreaServer.setText(Server.FullTextServer);

Socket SendSocket = null;
OutputStream os = null;
FileInputStream fis = null;
BufferedInputStream bis = null;
try {

/* The Server becomes Client and the Client becomes Server. */

SendSocket = new Socket(IP, port2);
os = SendSocket.getOutputStream();

/* Send the file through the output stream. */

File myFile = new File(abs);
byte[] mybytearray = new byte[(int) myFile.length() + 1];
fis = new FileInputStream(myFile);
bis = new BufferedInputStream(fis);
bis.read(mybytearray, 0, mybytearray.length);
os.write(mybytearray, 0, mybytearray.length);
os.flush();
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(filename + " sent successfully!",true);
}
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}

/* Notify that it's complete. */

Server.FullTextServer =  Server.FullTextServer + filename + " sent successfully!\r\n";
Server.textAreaServer.setText(Server.FullTextServer);

/* Delete duplicate file. */

File file = new File("./" + filename);
if (file.exists()){
file.delete();
}
bis.close();
SendSocket.close();
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
```

```
}
}
}
}

/**
* SocketSnapshot's Server's Output Handler (sub-)class.
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

class ServerOutputHandler implements Runnable {
private Socket socket;
private String message;
private boolean newmsg;

/**
* Constructor of Server Output Handler.
* @param socket The socket to be set.
*/

public ServerOutputHandler(Socket socket) {
this.socket = socket;
this.message = "";
this.newmsg = false;
}

/**
* If a new message was given (probably by an action listener above in the main
* class) then set it.
* @param message
*/

public void SetMessage(String message) {
this.message = message;
this.newmsg = true;
}

/**
* The runnable.
*/

public void run() {
Boolean error = false;
BufferedWriter writer = null;
try {
writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

/* Loop while client is connected. */

while(Server.connected){

/* Workaround for fast (modern) processors. */

Thread.sleep(1);

/* If empty skip. */

if ((message == null || message.equals("") || message.isEmpty())){
continue;
}else{

/* Workaround for fast (modern) processors. */

Thread.sleep(1);


/* If a new message has been set (externally). */

if (newmsg == true){
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append("#You said: " + message,true);
}
} catch (IOException e) {
```

```
e.printStackTrace();
}
Server.FullTextServer =  Server.FullTextServer + "#You said: " + message + "\r\n";
Server.textAreaServer.setText(Server.FullTextServer);
writer.write(message);
writer.newLine();
writer.flush();
newmsg = false;
Server.MessageTextField.setText("");
}
}
}

/* Catch any kind of error, log it, print it. */

} catch (Exception exp) {
error = true;
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(exp.toString(),false);
}
} catch (IOException e1) {
e1.printStackTrace();
}

} finally {
try {
if ((error == true) ||
(InetAddress.getByName(Server.connectionSocket.getInetAddress().getHostName().toString
()).isReachable(10000)==false)){
Server.lblConnected.setForeground(Server.red);
Server.lblConnected.setText("Not connected");
Server.connected = false;
Server.FullTextServer +=  Server.IP + " disconnected.\r\n";
Server.textAreaServer.setText( Server.FullTextServer);
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(Server.IP + " disconnected.",true);
}
} catch (IOException e) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
try {
if ( Server.connectionSocket != null){
Server.connectionSocket.close();
Server.connectionSocket = null;
Server.welcomeSocket.close();
Server.welcomeSocket = null;

Server.stopSingleConnection(true);

}
} catch (IOException e1) {
try {
if (Server.serversettings.isServerLoggingEnabled() == true){
Server.log.append(e1.toString(),false);
}
} catch (IOException e2) {
e2.printStackTrace();
}
}
}
} catch (IOException e) {
e.printStackTrace();
}finally{
}
}
}
}
```

ServerLogging.java

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Date;

/**
* SocketSnapshot's Server's Logging class.
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

public class ServerLogging {
private String Log;
private String filename;

/**
*  The constructor.
*/

public ServerLogging(){
filename = "ServerLog.log";
this.Log = "";
}

/**
* Self-explanatory.
* @return Returns true if  file exists, false otherwise.
*/

public Boolean DoesFileExist(){
File f = new File("./" + this.filename);
if (f.exists() && !f.isDirectory()) {
return true;
}
return false;
}

/**
* Creates a new log if the log doesn't already exist.
* @throws IOException In case of error.
*/

public void makeLog() throws IOException{
if (DoesFileExist()){
File file = new File("./" + this.filename);
file.delete();
}
Files.write(Paths.get("./" + this.filename), this.Log.getBytes());
}

/**
* Appends the log
* @param str The String to append.
* @param who Boolean which allows the method
*                        to decided whether it's a stack
*                        trace or a message box event (action) to be logged.
* @throws IOException In case of error.
*/

public void append(String str,Boolean who) throws IOException{
if (!DoesFileExist()){
return;
}
String str2 = "";
```

```
Date date = new Date();
if (who == true){
str2 = "#### Message Box Action ####";
}else{
str2 = "#### Stack ####";
}
this.Log = this.Log + str2 + " " + date.toString() + "\r\n" + str + "\r\n";
makeLog();
}

/**
* @return The Log itself.
*/

public String ToString(){
return Log;
}

/**
* @return The Filename itself.
*/

public String getFileName(){
return filename;
}
}
```

ServerSettings.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;

/**
* SocketSnapshot's Server's Settings class.
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

public class ServerSettings {
private Boolean ServerLogging;
private Boolean ServerSnapping;
private String Client_Folder;
private String ServerLog_File_Folder;
private String ServerSnapshot_Folder;
private double ServerPerc;
private static String filename;
private static int ServerSnapping_seconds;

/**
* The constructor.
*/

public ServerSettings(){
Path relativePath = Paths.get("./");
Path fullPath = relativePath.toAbsolutePath();
String str = fullPath.toString();
filename = "ServerSettings.cfg";
this.Client_Folder = str.substring(0,str.length()-2);
this.ServerLog_File_Folder = str.substring(0,str.length()-1);
this.ServerSnapshot_Folder = str.substring(0,str.length()-1);
ServerLogging = true;
ServerSnapping = true;
ServerSnapping_seconds = 60;
ServerPerc = (double)10;
}

/**
* Self-explanatory.
* @return Returns true if  file exists, false otherwise.
*/

public static Boolean DoesFileExist(){
File f = new File("./" + filename);
if (f.exists() && !f.isDirectory()) {
return true;
}
return false;
}

/**
*
* Parse a single String line and check whether it is a valid settings configuration
command
* or not.
*
* @param left The substring left of "=".
* @param right The substring right of "=".
* @return True if correct line, false otherwise.
```

```
*/

public Boolean ParseLine(String left,String right){
if ((left.equals("[Client_Folder]")
|| (left.equals("[ServerLog_File_Folder]")
|| (left.equals("[ServerSnapshot_Folder]")))))){
if (right == null || right.equals("") || right.isEmpty()){
if (left.equals("[ServerLog_File_Folder]")){
setServerLogFolder("");
return true;
}else if (left.equals("[ServerSnapshot_Folder]")){
setServerSnapshotFolder("");
return true;
}
return false;
}else{
File f = new File(right);
if(f.exists() && f.isDirectory()){
if (left.equals("[Client_Folder]")){
setClientFolder(right);
}else if (left.equals("[ServerLog_File_Folder]")){
setServerLogFolder(right);
}else{
setServerSnapshotFolder(right);
}
return true;
}
return false;
}
}else if ((left.equals("[ServerLogging]") || (left.equals("[ServerSnapping]")))){
if (right.equals("true") || (right.equals("false"))){
if (left.equals("[ServerLogging]")){
if (right.equals("true")){
setServerLogging(true);
}else{
setServerLogging(false);
}
}else{
if (right.equals("true")){
setServerSnapping(true);
}else{
setServerSnapping(false);
}
}
return true;
}
}else if (left.equals("[ServerSnapping_seconds]")){
int number = 0;
try {
number = Integer.parseInt(right);
if (!((number > 1) && (number < 65536))){
return false;
}
} catch (NumberFormatException e) {
return false;
}
setServerSnapping_seconds(number);
return true;
}else if (left.equals("[Serverpercentage]")){
double number = 0;
try {
number = Double.parseDouble(right);
if (!((number >= (double)0) && (number <= (double)100))){
return false;
}
} catch (NumberFormatException e) {
return false;
}
setServerpercentage_Warning(number);
return true;
}

return false;

}

/**
```

```
*
* Parses a whole Server Settings file.
*
* @param settings The ServerSettings to be parsed.
* @return True if the settings were in correct format, false otherwise.
*/

public static Boolean ParseExistingFile(ServerSettings settings){
if (!DoesFileExist()) {

return false;
}
List<String> strings = new ArrayList<String>();
BufferedReader br = null;
try {
String sCurrentLine;
br = new BufferedReader(new FileReader("./" + filename));
while ((sCurrentLine = br.readLine()) != null) {
strings.add(sCurrentLine);
}
} catch (IOException e) {
e.printStackTrace();
} finally {
try {
if (br != null)br.close();
} catch (IOException ex) {
ex.printStackTrace();
}
}
if (strings.size() != 7){
return false;
}
for (int i = 0; i < strings.size(); i++){
String current = strings.get(i);
int charCount = current.replaceAll("[^=]", "").length();
if (charCount != 1){

return false;
}
String[] parts =  strings.get(i).split("=");
if (parts.length != 1 && parts.length != 2){
return false;
}
if (parts.length == 2){
if (settings.ParseLine(parts[0],parts[1]) == false){
return false;
}
}else{
if (settings.ParseLine(parts[0],"") == false){
return false;
}
}
}

/* Check if everything is filled up. */

if ((settings.isServerLoggingEnabled() == true) && ((settings.getServerLogFolder() !=
null) || (!(settings.getServerLogFolder().equals(""))) ||
!(settings.getServerLogFolder().isEmpty()))){
return false;
}else if ((settings.isServerSnappingEnabled() == true) &&
((settings.getServerSnapshotFolder() != null) ||
(!(settings.getServerSnapshotFolder().equals(""))) ||
!(settings.getServerSnapshotFolder().isEmpty()))){
return false;
}
return true;
}

/**
* Creates a new ServerSettings file (using the current ServerSettings in this class).
* @param settings The ServerSettings to be created.
* @throws IOException In case of error.
*/

@SuppressWarnings({ "static-access" })
public void makeFile(ServerSettings settings) throws IOException{
```

```
if (DoesFileExist()) {
File file = new File("./" + settings.getServerFileName());
file.delete();
makeFile(settings);
}else{
String content = "";
content += "[Client_Folder]=" + settings.getClientFolder() + "\r\n";
content += "[ServerLog_File_Folder]=" + settings.getServerLogFolder() + "\r\n";
content += "[ServerSnapshot_Folder]=" + settings.getServerSnapshotFolder() + "\r\n";
content += "[ServerLogging]=" + settings.isServerLoggingEnabled() + "\r\n";
content += "[ServerSnapping]=" + settings.isServerSnappingEnabled() + "\r\n";
content += "[ServerSnapping_seconds]=" + settings.getServerSnapping_seconds() +
"\r\n";
content += "[Serverpercentage]=" + settings.getServerpercentage_Warning();
Files.write(Paths.get("./" + settings.getServerFileName()), content.getBytes());
}
}

/**
*
* @return Set the Server Settings filename.
*/

public String getServerFileName(){
return filename;
}

/**
*
* @param FolderPath Set the Client Folder path.
*/

public void setClientFolder(String FolderPath){
this.Client_Folder = FolderPath;
}

/**
*
* @param FolderPath Set the Server Log Folder path.
*/

public void setServerLogFolder(String FolderPath){
this.ServerLog_File_Folder = FolderPath;
}

/**
*
* @param FolderPath Set the Server Snapshot Folder Folder path.
*/

public void setServerSnapshotFolder(String FolderPath){
this.ServerSnapshot_Folder = FolderPath;
}

/**
*
* @param ServerSnapping_seconds Set the "take a snapshot every x" seconds to be set.
*/

@SuppressWarnings({ "static-access" })
public void setServerSnapping_seconds(int ServerSnapping_seconds){
this.ServerSnapping_seconds = ServerSnapping_seconds;
}

/**
*
* @param value Set whether Server Logging is enabled.
*/

public void setServerLogging(Boolean value){
this.ServerLogging = value;
}

/**
*
* @param value Set whether Server Snapshots are enabled.
*/
```

```
public void setServerSnapping(Boolean value){
this.ServerSnapping = value;
}

/**
*
* @param value Set Snapshot Warning threshold.
*/

public void setServerpercentage_Warning(double value){
this.ServerPerc = value;
}

/**
*
* @return The Client Folder path.
*/

public String getClientFolder(){
return Client_Folder;
}

/**
*
* @return The Server Log Folder path.
*/

public String getServerLogFolder(){
return ServerLog_File_Folder;
}

/**
*
* @return  The Server Snapshot Folder path.
*/

public String getServerSnapshotFolder(){
return ServerSnapshot_Folder;
}

/**
*
* @return True if Server Logging is enabled, false otherwise.
*/

public Boolean isServerLoggingEnabled(){
return ServerLogging;
}

/**
*
* @return True if Server Snapshots are enabled, false otherwise.
*/

public Boolean isServerSnappingEnabled(){
return ServerSnapping;
}

/**
*
* @return The "Take a snapshot every x" seconds.
*/

public static int getServerSnapping_seconds(){
return ServerSnapping_seconds;
}

/**
*
* @return The Snapshot Sending (Percentage thresohold).
*/

public double getServerpercentage_Warning(){
return ServerPerc;
}
}
```

ServerSettingsPopUp.java

```
import java.awt.Color;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.util.concurrent.TimeUnit;

import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRootPane;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

/**
* SocketSnapshot's Server Settings Popup
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

public class ServerSettingsPopup extends JFrame {

/* Private Variables - JAVA SWING (GUI). */

private JButton btnServerLogFolder;
private JButton btnServerSaveAndClose;
private JButton btnSetServerSnapshotFolder;
private JCheckBox ServerLogcheckBox;
private JLabel WarningLabel;
private JLabel labelSnapshotStatus;
private JLabel labelXSeconds;
private JLabel lblSendASnapshot;
private JLabel lblServerFolderSettings;
private JLabel lblServerLogEnabled;
private JLabel lblServerLogFile;
private JLabel lblSnapshots;
private JLabel lblWarn;
private JLabel lblWarnClientIf;
private JPanel contentPane;
private JTextField ClientFolderTextField;
private JTextField ServerLogFolderTextField;
private JTextField ServerSnapshotFolderTextField;

/* Serials.
*
* SerialVersionUID is a unique identifier for each class,
*  by which Java Virtual Machine compares the versions of class.
*  If we don't provide this field explicitly , Java Virtual Machine
*  generates it automatically for every class . The Algorithm for
*  generating SerialVersionUID is highly dependent on the Java Virtual Machine
version,
*  and class structure.
*/

private static final long serialVersionUID = 1L;
```

```
/* Colors. */

public Color black = new Color(0,0,0);
public Color green = new Color(0,255,0);
public Color red = new Color(255,0,0);

/**
* Creates the Server Settings Popup.
*/

@SuppressWarnings("static-access")
public ServerSettingsPopup(){
this.setTitle("SocketSnapshot (Server) Settings");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setUndecorated(true);
getRootPane().setWindowDecorationStyle(JRootPane.INFORMATION_DIALOG);
setBounds(100, 100, 503, 278);
contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);
GridBagLayout gbl_contentPane = new GridBagLayout();
gbl_contentPane.columnWidths = new int[]{14, 141, 0, 0, 0};
gbl_contentPane.rowHeights = new int[]{0, -11, 8, 27, 0, 0, 0, 0, 0, 0, 0, 20, 0};
gbl_contentPane.columnWeights = new double[]{0.0, 0.0, 1.0, 1.0, Double.MIN_VALUE};
gbl_contentPane.rowWeights = new double[]{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, Double.MIN_VALUE};
contentPane.setLayout(gbl_contentPane);

/***
*
*    88                          88                          88
*    88                          88                          88
*    88                          88                          88
*    88           ,adPPYYba,     88,dPPYba,     ,adPPYba,    88   ,adPPYba,
*    88           ""     `Y8     88P'    "8a   a8P     88    88   I8[    ""
*    88           ,adPPPPP88     88       d8   8PP"""""""    88    `"Y8ba,
*    88           88,    ,88     88b,   ,a8"   "8b,   ,aa    88   aa    ]8I
*    88888888888  `"8bbdP"Y8     8Y"Ybbd8"'     `"Ybbd8"'    88   `"YbbdP"'
*
*
*/

/***
*
* Create a JLabel "Warning".
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |W|a|r|n|i|n|g| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

WarningLabel = new JLabel("");
WarningLabel.setForeground(Color.RED);
GridBagConstraints gbc_WarningLabel = new GridBagConstraints();
gbc_WarningLabel.anchor = GridBagConstraints.WEST;
gbc_WarningLabel.gridwidth = 2;
gbc_WarningLabel.insets = new Insets(0, 0, 5, 5);
gbc_WarningLabel.gridx = 1;
gbc_WarningLabel.gridy = 2;
contentPane.add(WarningLabel, gbc_WarningLabel);

/***
*
* Create a JLabel "Folder Settings".
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |F|o|l|d|e|r| |S|e|t|t|i|n|g|s| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblServerFolderSettings = new JLabel("Folder Settings");
GridBagConstraints gbc_lblServerFolderSettings = new GridBagConstraints();
gbc_lblServerFolderSettings.gridwidth = 3;
gbc_lblServerFolderSettings.insets = new Insets(0, 0, 5, 0);
gbc_lblServerFolderSettings.gridx = 1;
gbc_lblServerFolderSettings.gridy = 3;
```

```
contentPane.add(lblServerFolderSettings, gbc_lblServerFolderSettings);

/***
*
* Create a JLabel "Log file Settings".
*     +-+-+-+ +-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*     |L|o|g| |F|i|l|e| |S|e|t|t|i|n|g|s| |L|a|b|e|l|
*     +-+-+-+ +-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblServerLogFile = new JLabel("Log File Settings");
GridBagConstraints gbc_lblServerLogFile = new GridBagConstraints();
gbc_lblServerLogFile.gridwidth = 3;
gbc_lblServerLogFile.insets = new Insets(0, 0, 5, 0);
gbc_lblServerLogFile.gridx = 1;
gbc_lblServerLogFile.gridy = 7;
contentPane.add(lblServerLogFile, gbc_lblServerLogFile);

/***
*
* Create a JLabel "Log Enabled".
*     +-+-+-+ +-+-+-+-+-+-+-+
*     |L|o|g| |E|n|a|b|l|e|d|
*     +-+-+-+ +-+-+-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblServerLogEnabled = new JLabel("Log enabled?");
GridBagConstraints gbc_lblServerLogEnabled = new GridBagConstraints();
gbc_lblServerLogEnabled.insets = new Insets(0, 0, 5, 5);
gbc_lblServerLogEnabled.gridx = 1;
gbc_lblServerLogEnabled.gridy = 8;
contentPane.add(lblServerLogEnabled, gbc_lblServerLogEnabled);

/***
*
* Create a JLabel "Snapshots Enabled".
*     +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
*     |S|n|a|p|s|h|o|t|s| |E|n|a|b|l|e|d|
*     +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-
* With the JLabel class, you can display unselectable text and images.
*/

lblSnapshots = new JLabel("Snapshots?");
GridBagConstraints gbc_lblSnapshots = new GridBagConstraints();
gbc_lblSnapshots.insets = new Insets(0, 0, 5, 5);
gbc_lblSnapshots.gridx = 1;
gbc_lblSnapshots.gridy = 9;
contentPane.add(lblSnapshots, gbc_lblSnapshots);

/***
*
* Create a JLabel "Snapshot Status".
*     +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+
*     |S|n|a|p|s|h|o|t| |S|t|a|t|u|s|
*     +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

labelSnapshotStatus = new JLabel("");
if (Server.serversettings.isServerSnappingEnabled() == true){
labelSnapshotStatus.setText("Enabled!");
}else{
labelSnapshotStatus.setText("Disabled!");
}
GridBagConstraints gbc labelSnapshotStatus = new GridBagConstraints();
gbc_labelSnapshotStatus.gridwidth = 2;
gbc_labelSnapshotStatus.insets = new Insets(0, 0, 5, 0);
gbc labelSnapshotStatus.gridx = 2;
gbc_labelSnapshotStatus.gridy = 9;
contentPane.add(labelSnapshotStatus, gbc_labelSnapshotStatus);

/***
*
* Create a JLabel "Send a snapshot every".
*     +-+-+-+-+ +-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
```

```
*      |S|e|n|d| |a| |s|n|a|p|s|h|o|t| |e|v|e|r|y|
*      +-+-+-+-+ +-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblSendASnapshot = new JLabel("Send a snapshot every");
GridBagConstraints gbc_lblSendASnapshot = new GridBagConstraints();
gbc_lblSendASnapshot.insets = new Insets(0, 0, 5, 5);
gbc_lblSendASnapshot.gridx = 1;
gbc_lblSendASnapshot.gridy = 10;
contentPane.add(lblSendASnapshot, gbc_lblSendASnapshot);

/***
*
* Create a JLabel "Seconds".
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |S|e|c|o|n|d|s| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

labelXSeconds = new JLabel("");
String temp = ServergetTIME(Server.serversettings.getServerSnapping_seconds());
labelXSeconds.setText(temp);
GridBagConstraints gbc_labelXSeconds = new GridBagConstraints();
gbc_labelXSeconds.gridwidth = 2;
gbc labelXSeconds.insets = new Insets(0, 0, 5, 0);
gbc_labelXSeconds.gridx = 2;
gbc_labelXSeconds.gridy = 10;
contentPane.add(labelXSeconds, gbc_labelXSeconds);

/***
*
* Create a JLabel "Warn Client if % exceeds".
*      +-+-+-+-+ +-+-+-+-+-+-+ +-+-+-+-+-+
*      |W|a|r|n| |C|l|i|e|n|t| |L|a|b|e|l|
*      +-+-+-+-+ +-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblWarnClientIf = new JLabel("Warn client if % exceeds");
GridBagConstraints gbc_lblWarnClientIf = new GridBagConstraints();
gbc_lblWarnClientIf.insets = new Insets(0, 0, 5, 5);
gbc_lblWarnClientIf.gridx = 1;
gbc_lblWarnClientIf.gridy = 11;
contentPane.add(lblWarnClientIf, gbc_lblWarnClientIf);

/***
*
* Create a JLabel "Warn".
*      +-+-+-+-+ +-+-+-+-+
*      |W|a|r|n| |L|a|b|e|l|
*      +-+-+-+-+ +-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblWarn = new JLabel("");
lblWarn.setText(Server.serversettings.getServerpercentage_Warning() + "%");
GridBagConstraints gbc_lblWarn = new GridBagConstraints();
gbc_lblWarn.gridwidth = 2;
gbc_lblWarn.insets = new Insets(0, 0, 5, 5);
gbc_lblWarn.gridx = 2;
gbc_lblWarn.gridy = 11;
contentPane.add(lblWarn, gbc_lblWarn);

/***
*
*   88888888888                          888888888  88            88          88
*        88                             ,d     88  ""            88          88
*        88                             88     88                88          88
*        88   ,adPPYba,  8b,     ,d8  MM88MMM  88aaaaa      88   ,adPPYba,  88   ,adPPYb,88  ,adPPYba,  ,adPPYba,
*        88  a8P_____88   `Y8, ,8P'     88     88"""""      88  a8P_____88  88  a8"    `Y88  I8[    ""  I8[    ""
*        88  8PP"""""""    )888(       88     88           88  8PP"""""""  88  8b       88   `"Y8ba,    `"Y8ba,
*        88  "8b,   ,aa   ,d8" "8b,     88,    88           88  "8b,   ,aa  88  "8a,   ,d88  aa    ]8I  aa    ]8I
*        88   `"Ybbd8"'  8P'     `Y8    "Y888  88           88   `"Ybbd8"'  88   `"8bbdP"Y8   `"YbbdP"'   `"YbbdP"'
*
*
*/

/***
* Create a JTextField for Client's Folder.
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+
*      |C|l|i|e|n|t| |F|o|l|d|e|r| |T|e|x|t|F|i|e|l|d|
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+
```

```
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

ClientFolderTextField = new JTextField();
ClientFolderTextField.setText(Server.serversettings.getClientFolder());
ClientFolderTextField.setEditable(false);

GridBagConstraints gbc_ClientFolderTextField = new GridBagConstraints();
gbc_ClientFolderTextField.gridwidth = 2;
gbc_ClientFolderTextField.insets = new Insets(0, 0, 5, 0);
gbc_ClientFolderTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_ClientFolderTextField.gridx = 2;
gbc_ClientFolderTextField.gridy = 4;
contentPane.add(ClientFolderTextField, gbc_ClientFolderTextField);
ClientFolderTextField.setColumns(10);

/***
* Create a JTextField for Server Log Folder.
*     +-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+-+-+-+-+-+
*     |S|e|r|v|e|r| |L|o|g| |T|e|x|t|F|i|e|l|d|
*     +-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

ServerLogFolderTextField = new JTextField();
ServerLogFolderTextField.setText(Server.serversettings.getServerLogFolder());
ServerLogFolderTextField.setEditable(false);
ServerLogFolderTextField.setColumns(10);
GridBagConstraints gbc_ServerLogFolderTextField = new GridBagConstraints();
gbc_ServerLogFolderTextField.gridwidth = 2;
gbc_ServerLogFolderTextField.insets = new Insets(0, 0, 5, 0);
gbc_ServerLogFolderTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_ServerLogFolderTextField.gridx = 2;
gbc_ServerLogFolderTextField.gridy = 5;
contentPane.add(ServerLogFolderTextField, gbc_ServerLogFolderTextField);

/***
* Create a JTextField for Server Snapshot Folder.
*     +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*     |S|e|r|v|e|r| |S|n|a|p|s|h|o|t| |T|e|x|t|F|i|e|l|d|
*     +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

ServerSnapshotFolderTextField = new JTextField();
ServerSnapshotFolderTextField.setText(Server.serversettings.getServerSnapshotFolder())
;
ServerSnapshotFolderTextField.setEditable(false);
ServerSnapshotFolderTextField.setColumns(10);
GridBagConstraints gbc_ServerSnapshotFolderTextField = new GridBagConstraints();
gbc_ServerSnapshotFolderTextField.gridwidth = 2;
gbc_ServerSnapshotFolderTextField.insets = new Insets(0, 0, 5, 0);
gbc_ServerSnapshotFolderTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_ServerSnapshotFolderTextField.gridx = 2;
gbc_ServerSnapshotFolderTextField.gridy = 6;
contentPane.add(ServerSnapshotFolderTextField, gbc_ServerSnapshotFolderTextField);


/***
*
*    88888888ba
*    88      "8b                ,d       ,d
*    88      ,8P                88       88
*    88aaaaaa8P'  88       88  MM88MMM  MM88MMM  ,adPPYba,   8b,dPPYba,   ,adPPYba,
*    88""""""8b,  88       88    88       88    a8"     "8a  88P'   `"8a  I8[    ""
*    88      `8b  88       88    88       88    8b       d8  88       88   `"Y8ba,
*    88      a8P  "8a,   ,a88    88,      88,   "8a,   ,a8"  88       88  aa    ]8I
*    88888888P"    `"YbbdP'Y8    "Y888    "Y888  `"YbbdP"'   88       88  `"YbbdP"'
*
*
*/
```

```
/***
*      ____ ____ ____ ________ ____ ____ ____ ____ ____ ____ ________ ____ ____ ____
____ ____ ____
*     ||S |||e |||t |||       |||C |||l |||i |||e |||n |||t |||       |||F |||o |||l
|||d |||e |||r ||
*
||__|||__|||__|||_______|||__|||__|||__|||__|||__|||__|||_______|||__|||__|||__|||__||
|__|||__||
*
|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|
/__\|/__\|
*/

JButton btnSetClientFolder = new JButton("Set Client Folder...");
btnSetClientFolder.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
final JFileChooser chooser = new JFileChooser() {
private static final long serialVersionUID = 1L;

public void approveSelection() {
if (getSelectedFile().isFile()) {
return;
} else
super.approveSelection();
}
};
chooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
int returnVal = chooser.showOpenDialog(getParent());
if (chooser.getSelectedFile() == null){
return;
}
String absolute = chooser.getSelectedFile().getAbsolutePath();
if(returnVal == JFileChooser.APPROVE_OPTION) {
WarningLabel.setText("");
ClientFolderTextField.setForeground(black);
ClientFolderTextField.setText(absolute);
}
}
});
btnSetClientFolder.setToolTipText("<html>Click to set/change the folder where the
recieved files and folders (in .ZIP format)\r\n<br>\r\nshould be
saved.</br>\r\n<br></br>\r\n<br></br>\r\nThe root directory of the folder that this
application was launcher is the default.</br>\r\n</html>");
GridBagConstraints gbc_btnSetClientFolder = new GridBagConstraints();
gbc_btnSetClientFolder.fill = GridBagConstraints.HORIZONTAL;
gbc_btnSetClientFolder.insets = new Insets(0, 0, 5, 5);
gbc_btnSetClientFolder.gridx = 1;
gbc_btnSetClientFolder.gridy = 4;
contentPane.add(btnSetClientFolder, gbc_btnSetClientFolder);

/***
*      ____ ____ ____ ________ ____ ____ ____ ________ ____ ____ ____ ____ ____ ____
*     ||S |||e |||t |||       |||L |||o |||g |||       |||F |||o |||l |||d |||e |||r ||
*     ||__|||__|||__|||_______|||__|||__|||__|||_______|||__|||__|||__|||__|||__|||__||
*     |/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

btnServerLogFolder = new JButton("Set log folder...");
btnServerLogFolder.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
final JFileChooser chooser = new JFileChooser() {
/**
*
*/
private static final long serialVersionUID = 1L;

public void approveSelection() {
if (getSelectedFile().isFile()) {
return;
} else
super.approveSelection();
}
};
chooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
int returnVal = chooser.showOpenDialog(getParent());
if (chooser.getSelectedFile() == null){
```

```
return;
}
String absolute = chooser.getSelectedFile().getAbsolutePath();
if(returnVal == JFileChooser.APPROVE_OPTION) {
WarningLabel.setText("");
ServerLogFolderTextField.setForeground(black);
ServerLogFolderTextField.setText(absolute);
}
}
});
btnServerLogFolder.setToolTipText("<html>Click to set/change the folder where the log
file should be saved.\r\n<br></br>\r\n<br>A logfile is a file that records either
events that occur in this program as well as messages</br>\r\n <br> between different
users.</br>\r\n<br></br>\r\n<br> Logging is the act of keeping a log. In the simplest
case, messages are written to a single logfile.</br></html>");
GridBagConstraints gbc_btnServerLogFolder = new GridBagConstraints();
gbc_btnServerLogFolder.fill = GridBagConstraints.HORIZONTAL;
gbc_btnServerLogFolder.insets = new Insets(0, 0, 5, 5);
gbc_btnServerLogFolder.gridx = 1;
gbc_btnServerLogFolder.gridy = 5;
contentPane.add(btnServerLogFolder, gbc_btnServerLogFolder);

/***
*                                _ ____ ____ ____ ____ ____ ____ ____ ____ ____ ________
____ ____ ____ ____ ____ ____
*    ||S |||e |||t |||       |||S |||n |||a |||p |||s |||h |||o |||t |||s |||
|||F |||o |||l |||d |||e |||r ||
*
||__|||__|||__|||_______|||__|||__|||__|||__|||__|||__|||__|||__|||__|||_______|||__||
|__|||__|||__|||__|||__||
*
|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|/_______\|/__\|
/ _\|/__\|/__\|/__\|/__\|
*/

btnSetServerSnapshotFolder = new JButton("Set Snapshot folder...");
btnSetServerSnapshotFolder.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
final JFileChooser chooser = new JFileChooser() {
/**
*
*/
private static final long serialVersionUID = 1L;

public void approveSelection() {
if (getSelectedFile().isFile()) {
return;
} else
super.approveSelection();
}
};
chooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
int returnVal = chooser.showOpenDialog(getParent());
if (chooser.getSelectedFile() == null){
return;
}
String absolute = chooser.getSelectedFile().getAbsolutePath();
if(returnVal == JFileChooser.APPROVE_OPTION) {
WarningLabel.setText("");
ServerSnapshotFolderTextField.setForeground(black);
ServerSnapshotFolderTextField.setText(absolute);
}
}
});
GridBagConstraints gbc_btnSetServerSnapshotFolder = new GridBagConstraints();
gbc_btnSetServerSnapshotFolder.fill = GridBagConstraints.HORIZONTAL;
gbc_btnSetServerSnapshotFolder.insets = new Insets(0, 0, 5, 5);
gbc_btnSetServerSnapshotFolder.gridx = 1;
gbc_btnSetServerSnapshotFolder.gridy = 6;
contentPane.add(btnSetServerSnapshotFolder, gbc_btnSetServerSnapshotFolder);

/***
*
*    ||S |||a |||v |||e |||       |||a |||n |||d |||       |||C |||l |||o |||s |||e ||
*    ||__|||__|||__|||__|||_______|||__|||__|||__|||_______|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|
*/
```

```
btnServerSaveAndClose = new JButton("Save and Close");
btnServerSaveAndClose.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
String rcvdf = ClientFolderTextField.getText();
String logf  = ServerLogFolderTextField.getText();
String snapsf = ServerSnapshotFolderTextField.getText();
if (rcvdf.equals("") || rcvdf == null || rcvdf.isEmpty()){
WarningLabel.setText("Please check the text marked in red.");
ClientFolderTextField.setForeground(red);
ClientFolderTextField.setText("Please select a folder.");
}
if (ServerLogcheckBox.isSelected() == true){
if (logf.equals("") || logf == null || logf.isEmpty()){
WarningLabel.setText("Please check the text marked in red.");
ServerLogFolderTextField.setForeground(red);
ServerLogFolderTextField.setText("Please select a folder.");
}
}
if ((ServerSnapshotFolderTextField.getText().equals("Please select a folder."))
|| (ServerLogFolderTextField.getText().equals("Please select a folder."))
|| ( ClientFolderTextField.getText().equals("Please select a folder."))){
return;
}
Server.serversettings.setClientFolder(rcvdf);
Server.serversettings.setServerLogging(ServerLogcheckBox.isSelected());
if (Server.serversettings.isServerLoggingEnabled()){
Server.serversettings.setServerLogFolder(logf);
}else{
Server.serversettings.setServerLogFolder("");
File f = new File("./" + "ServerLog.log");
if (f.exists() && !f.isDirectory()) {
f.delete();
}
}
if (Server.serversettings.isServerSnappingEnabled()){
Server.serversettings.setServerSnapshotFolder(snapsf);
}else{
Server.serversettings.setServerSnapshotFolder("");
}
try {
Server.serversettings.makeFile(Server.serversettings);
JOptionPane.showMessageDialog(null,"Settings saved successfully!");
setVisible(false);
dispose();
} catch (IOException e1) {
e1.printStackTrace();
}

}
});

GridBagConstraints gbc_btnServerSaveAndClose = new GridBagConstraints();
gbc_btnServerSaveAndClose.fill = GridBagConstraints.HORIZONTAL;
gbc_btnServerSaveAndClose.gridwidth = 3;
gbc_btnServerSaveAndClose.gridx = 1;
gbc_btnServerSaveAndClose.gridy = 12;
contentPane.add(btnServerSaveAndClose, gbc_btnServerSaveAndClose);

/***
*
*     ,ad888ba,   88                              88        88888888ba
*    d8"'    `"8b 88                              88        88      "8b
*   d8'          88                              88        88      ,8P
*   88           88,dPPYba,    ,adPPYba,   ,adPPYba,  88   ,d8   88aaaaaa8P'   ,adPPYba,  8b,     ,d8  ,adPPYba,  ,adPPYba,
*   88           88P'    "8a  a8P_____88  a8"     ""  88 ,a8"    88""""""8b,  a8"     "8a  `Y8, ,8P'  a8P_____88  I8[    ""
*   Y8,          88       88  8PP"""""""  8b          8888[      88      `8b  8b       d8    )888(    8PP"""""""   `"Y8ba,
*    Y8a.    .a8P 88       88  "8b,   ,aa  "8a,   ,aa  88`"Yba,   88      a8P  "8a,   ,a8" ,d8" "8b,  "8b,   ,aa  aa    ]8I
*     `"Y8888Y"'  88       88   `"Ybbd8"'   `"Ybbd8"'  88   `Y8a  88888888P"    `"YbbdP"'  8P'     `Y8  `"Ybbd8"'  `"YbbdP"'
*
*
*/

/***
*      +-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+-+-+-+-+
*      |S|e|r|v|e|r| |L|o|g| |C|h|e|c|k|b|o|x|
*      +-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+-+-+-+-+
*/

ServerLogcheckBox = new JCheckBox("");
ServerLogcheckBox.setSelected(Server.serversettings.isServerLoggingEnabled());
ServerLogcheckBox.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
```

```
if (ServerLogcheckBox.isSelected() == true){
btnServerLogFolder.setEnabled(true);
ServerLogFolderTextField.setText(Server.serversettings.getServerLogFolder());
}else{
btnServerLogFolder.setEnabled(false);
ServerLogFolderTextField.setText("");
}
}
});
GridBagConstraints gbc_ServerLogcheckBox = new GridBagConstraints();
gbc_ServerLogcheckBox.gridwidth = 2;
gbc_ServerLogcheckBox.insets = new Insets(0, 0, 5, 0);
gbc_ServerLogcheckBox.gridx = 2;
gbc_ServerLogcheckBox.gridy = 8;
contentPane.add(ServerLogcheckBox, gbc_ServerLogcheckBox);

if (ServerLogcheckBox.isSelected() == true){
btnServerLogFolder.setEnabled(true);
ServerLogFolderTextField.setText(Server.serversettings.getServerLogFolder());
}else{
btnServerLogFolder.setEnabled(false);
ServerLogFolderTextField.setText("");
}
}

/**
* Convert seconds (integer format) to time (String format)
* @param time_wanted The time in seconds (integer).
* @return String format of the seconds.
*/

public static String ServergetTIME(int time_wanted) {
String output text = "";
long hours = TimeUnit.SECONDS.toHours(time_wanted);
long minute = TimeUnit.SECONDS.toMinutes(time_wanted) -
(TimeUnit.SECONDS.toHours(time wanted)* 60);
long second = TimeUnit.SECONDS.toSeconds(time_wanted) -
(TimeUnit.SECONDS.toMinutes(time_wanted) *60);
if (hours != 0){
output_text += hours + " hour";
if (hours != 1){
output_text += "s, ";
}else{
if (second != 0 && minute != 0){
output_text += ", ";
}
}
}
if (minute != 0){
output_text += minute + " minute";
if (minute != 1){
output_text += "s, ";
}else{
if (second != 0){
output_text += ", ";
}
}
}
if (second != 0){
output_text += second + " second";
if (second != 1){
output_text += "s";
}
}
return output_text;
}

}
```

Κώδικας Client

Client.java

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.CancellationException;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingWorker;
import javax.swing.border.EmptyBorder;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;

/**
* SocketSnapshot's Client's main class
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

@SuppressWarnings("serial")
public class Client extends JFrame{

/*******************************************
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+     *
*     |P|r|i|v|a|t|e| |V|a|r|i|a|b|l|e|s|     *
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+     *
*******************************************/

/* Java SWING (GUI) */

private JButton btnRequestAFile;
private JLabel lblMessageBox;
private JMenu mnEdit;
private JMenu mnFile;
private JMenuBar menuBar;
```

```
private JMenuItem mntmExit;
private JMenuItem mntmSettings;
private JPanel contentPane;
private JScrollPane scrollPane;
private JTextField chatField;

/*********************************************
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+       *
*     |P|u|b|l|i|c| |V|a|r|i|a|b|l|e|s|       *
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+       *
*********************************************/

/* Java SWING (GUI) */

public JButton AbortButton;
public JButton JoinButton;
public JButton SendButton;
public static JTextArea textAreaClient;
public JTextField PortTextField;
public JTextField ServerTextField;

/* Colors */

public Color black = new Color(0,0,0);
public Color green = new Color(0,255,0);
public Color red = new Color(255,0,0);

/* Booleans */

public static boolean ClientConnected = false;

/* Numbers */

public static int port = 0;

/* Strings */

public static String FullTextClient = "";
public String ServerSentence = "";

/* Client-only stuff */

public static List<?> ServersFileList = new ArrayList<Object>();
public ClientOutputHandler z;
public static ClientSettings settings;
public SwingWorker<Integer, String> workerClient;
public BufferedWriter writerClient;
public static Socket clientSocket = null;
private JMenu mnHelp;
private JMenuItem mntmHelp;
private JMenuItem mntmAbout;

/**
* In Java, the main method is where control enters a program or piece of code;
* this is where a program starts its execution.
*
* Specifically, it instantiates a new Client instance,
* thus a new frame (the Client frame).
*
* @param args Contains the supplied command-line arguments as an array of String
objects.
*/

public static void main(String[] args) {
EventQueue.invokeLater(new Runnable() {
public void run() {
try {
Client frame = null;
frame = new Client(frame);
frame.setVisible(true);
} catch (Exception e) {
e.printStackTrace();
}
}
});
}
```

```
/**
* The Client's constructor;
*
* It does everything that's required for the program to run properly (settings,
et.c.),
* while using other classes as well to perform the required operations.
*
* @param frame The Client's main frame.
*/

public Client(Client frame) {

/* Instantiate Server Settings */

instantiateClientSettings(frame);

/*********************************************************************************************
*                                                                                            *
*    ,ad8888ba,                                  88          88                          88  *
*   d8"'    `"8b                                 88          ""                          88  *
*  d8'                                           88                                      88  *
*  88            8b,dPPYba,  ,adPPYYba,  8b,dPPYba,   88,dPPYba,   88   ,adPPYba,  ,adPPYYba,  88  *
*  88      88888  88P'   "Y8  ""     `Y8  88P'    "8a  88P'    "8a  88  a8"     ""  ""     `Y8  88  *
*  Y8,        88  88          ,adPPPPP88  88       d8  88       88  88  8b          ,adPPPPP88  88  *
*   Y8a.    .a88  88          88,    ,88  88b,   ,a8"  88       88  88  "8a,   ,aa  88,    ,88  88  *
*    `"Y88888P"   88          `"8bbdP"Y8  88`YbbdP"'   88       88  88   `"Ybbd8"'  `"8bbdP"Y8  88  *
*                                         88                                                 *
*                                         88                                                 *
*********************************************************************************************

************************************************
*                                              *
*  88       88                                 *
*  88       88                                 *
*  88       88                                 *
*  88       88  ,adPPYba,   ,adPPYba,  8b,dPPYba,  *
*  88       88  I8[    ""  a8P_____88  88P'   "Y8  *
*  88       88   `"Y8ba,   8PP"""""""  88          *
*  Y8a.    .a8P  aa    ]8I  "8b,   ,aa  88          *
*   `"Y8888Y"'   `"YbbdP"'   `"Ybbd8"'  88          *
*                                              *
************************************************

*****************************************************************************************
*                                                                                       *
*  88                                    ad88                                       *
*  88              ,d                   d8"                                         *
*  88              88                   88                                          *
*  88  8b,dPPYba,  MM88MMM  ,adPPYba,  8b,dPPYba,  MM88MMM  ,adPPYYba,   ,adPPYba,   ,adPPYba,  *
*  88  88P'   `"8a   88    a8P_____88  88P'   "Y8    88     ""     `Y8  a8"     ""  a8P_____88  *
*  88  88       88   88    8PP"""""""  88           88     ,adPPPPP88  8b          8PP"""""""  *
*  88  88       88   88,   "8b,   ,aa  88           88     88,    ,88  "8a,   ,aa  "8b,   ,aa  *
*  88  88       88   "Y888  `"Ybbd8"'  88           88     `"8bbdP"Y8   `"Ybbd8"'   `"Ybbd8"'  *
*                                                                                       *
*****************************************************************************************/

/* Set frame's (window's) title. */

this.setTitle("SocketSnapshot (Client)");

/* Set Default Close Operation ([X] button) to terminate the program. */

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

/* Set Window's default boundaries (can be resized, and also the window is
responsive). */

setBounds(100, 100, 408, 365);

/*
* Create a new JPanel and set the correct spacing and alignment.
*
* The JPanel class provides general-purpose containers for lightweight components.
*
*/

contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);

/* Use GridBagLayout for better alignment. */

GridBagLayout gbl_contentPane = new GridBagLayout();
gbl_contentPane.columnWidths = new int[]{71, 201, 105, 0};
gbl_contentPane.rowHeights = new int[]{24, 0, 0, 196, 31, 0};
gbl_contentPane.columnWeights = new double[]{0.0, 1.0, 1.0, Double.MIN_VALUE};
gbl_contentPane.rowWeights = new double[]{0.0, 0.0, 0.0, 1.0, 1.0, Double.MIN_VALUE};
contentPane.setLayout(gbl_contentPane);

/***
*
*    88b           d88
```

```
*    888b         d888
*    88`8b       d8'88
*    88 `8b     d8' 88   ,adPPYba,  8b,dPPYba,   88       88  ,adPPYba,
*    88  `8b   d8'  88  a8P____88  88P'   `"8a  88       88  I8[    ""
*    88   `8b d8'   88  8PP"""""""  88       88  88       88   `"Y8ba,
*    88    `888'    88  "8b,   ,aa  88       88  "8a,   ,a88  aa    ]8I
*    88     `8'     88   `"Ybbd8"'  88       88   `"YbbdP'Y8  `"YbbdP"'
*
*
*/

/* Add a Menu Bar */

menuBar = new JMenuBar();
setJMenuBar(menuBar);

/***
* Add Menu "File" to the Menu Bar.
*
*     +-+-+-+-+
*     |F|i|l|e|
*     +-+-+-+-+
*
*/

mnFile = new JMenu("File");
menuBar.add(mnFile);

/***
* Add Menu "Exit" to the Menu Bar.
*
*     +-+-+-+-+
*     |E|x|i|t|
*     +-+-+-+-+
*
*/

mntmExit = new JMenuItem("Exit");

/* Add an Action Listener to the "Exit" Menu Item */

mntmExit.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
System.exit(0);
}
});
mnFile.add(mntmExit);

/***
* Add "Edit" Menu to the Menu Bar.
*
*     +-+-+-+-+
*     |E|d|i|t|
*     +-+-+-+-+
*
*/

mnEdit = new JMenu("Edit");
menuBar.add(mnEdit);

/***
* Add "Settings" Menu Item to "Edit" Menu.
*     +-+-+-+-+-+-+-+-+
*     |S|e|t|t|i|n|g|s|
*     +-+-+-+-+-+-+-+-+
*
*/

mntmSettings = new JMenuItem("Settings");
mntmSettings.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
EventQueue.invokeLater(new Runnable() {
public void run() {
try {
ClientSettingsPopup frame = new ClientSettingsPopup();
frame.setVisible(true);
} catch (Exception e) {
```

```
e.printStackTrace();
}
}
});
}
});
mnEdit.add(mntmSettings);

/***
* Add "Help" Menu to the Menu Bar.
*
*     +-+-+-+-+
*     |H|e|l|p|
*     +-+-+-+-+
*
*/

mnHelp = new JMenu("Help");
menuBar.add(mnHelp);

/***
* Add "Help" Menu Item to "Help" Menu.
*     +-+-+-+-+
*     |H|e|l|p|
*     +-+-+-+-+
*
*/

mntmHelp = new JMenuItem("Help");
mnHelp.add(mntmHelp);
mntmHelp.setEnabled(false);

/***
* Add "About" Menu Item to "Help" Menu
*     +-+-+-+-+-+-+
*     |A|b|o|u|t|
*     +-+-+-+-+-+-+
*/

mntmAbout = new JMenuItem("About..");
mntmAbout.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
JOptionPane.showMessageDialog(null,
"This program was created as a part of the project"
+ " of Thesis of Paulina Barniadaki,\n"
+ "student of Electronic Engineering Department,"
+ " School of Applied Sciences.",
"About this program..",
JOptionPane.INFORMATION_MESSAGE);
}
});
mnHelp.add(mntmAbout);

/***
*
*    88                          88                          88
*    88                          88                          88
*    88                          88                          88
*    88           ,adPPYYba,   88,dPPYba,     ,adPPYba,   88   ,adPPYba,
*    88           ""     `Y8   88P'    "8a   a8P_____88   88   I8[    ""
*    88           ,adPPPPP88   88       d8   8PP"""""""   88    `"Y8ba,
*    88           88,    ,88   88b,   ,a8"   "8b,    ,aa   88   aa    ]8I
*    88888888888  `"8bbdP"Y8   8Y"Ybbd8"'     `"Ybbd8"'   88   `"YbbdP"'
*
*
*/

/***
* Create a JLabel "Server IP".
*
*     +-+-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+
*     |S|e|r|v|e|r| |I|P| |L|a|b|e|l|
*     +-+-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+
*
* With the JLabel class, you can display unselectable text and images.
*/
```

```
JLabel ServerIPLabel = new JLabel("Server IP:");
GridBagConstraints gbc_ServerIPLabel = new GridBagConstraints();
gbc_ServerIPLabel.anchor = GridBagConstraints.EAST;
gbc_ServerIPLabel.insets = new Insets(0, 0, 5, 5);
gbc_ServerIPLabel.gridx = 0;
gbc_ServerIPLabel.gridy = 0;
contentPane.add(ServerIPLabel, gbc_ServerIPLabel);

/***
* Create a JLabel "Port".
*      +-+-+-+-+ +-+-+-+-+-+
*      |P|o|r|t| |L|a|b|e|l|
*      +-+-+-+-+ +-+-+-+-+-+
*
* With the JLabel class, you can display unselectable text and images.
*/

JLabel PortLabel = new JLabel("Port: ");
GridBagConstraints gbc_PortLabel = new GridBagConstraints();
gbc_PortLabel.anchor = GridBagConstraints.EAST;
gbc_PortLabel.insets = new Insets(0, 0, 5, 5);
gbc_PortLabel.gridx = 0;
gbc_PortLabel.gridy = 1;
contentPane.add(PortLabel, gbc_PortLabel);

/***
* Create a JLabel "Message Box".
*      +-+-+-+-+-+-+-+-+ +-+-+-+
*      |M|e|s|s|a|g|e| |B|o|x|
*      +-+-+-+-+-+-+-+-+ +-+-+-+
*
* With the JLabel class, you can display unselectable text and images.
*/

lblMessageBox = new JLabel("Message Box:");
GridBagConstraints gbc lblMessageBox = new GridBagConstraints();
gbc_lblMessageBox.anchor = GridBagConstraints.NORTH;
gbc_lblMessageBox.insets = new Insets(0, 0, 5, 5);
gbc_lblMessageBox.gridx = 0;
gbc_lblMessageBox.gridy = 3;
contentPane.add(lblMessageBox, gbc_lblMessageBox);


/***
*
*   888888888888                          88888888888  88            88          88
*        88                         ,d    88           ""            88          88
*        88                         88    88                         88          88
*        88   ,adPPYba,  8b,     ,d8  MM88MMM  88aaaaa      88   ,adPPYba,  88   ,adPPYb,88  ,adPPYba,  ,adPPYba,
*        88  a8P     88   `Y8, ,8P'     88     88"""""      88  a8P     88  88  a8"     `Y88  I8[    ""  I8[    ""
*        88  8PP"""""""    )888(       88     88           88  8PP"""""""  88  8b       88   `"Y8ba,    `"Y8ba,
*        88  "8b,   ,aa   ,d8" "8b,     88,    88           88  "8b,   ,aa  88  "8a,   ,d88  aa    ]8I  aa    ]8I
*        88   `"Ybbd8"'  8P'     `Y8    "Y888  88           88   `"Ybbd8"'  88   `"8bbdP"Y8   `"YbbdP"'   `"YbbdP"'
*
*
*/

/***
* Create a JTextField for Server's IP.
*      +-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+-+-+-+-+
*      |S|e|r|v|e|r| |I|P| |T|e|x|t|F|i|e|l|d|
*      +-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

ServerTextField = new JTextField();
GridBagConstraints gbc ServerTextField = new GridBagConstraints();
gbc_ServerTextField.anchor = GridBagConstraints.SOUTH;
gbc_ServerTextField.fill = GridBagConstraints.HORIZONTAL;
gbc ServerTextField.insets = new Insets(0, 0, 5, 0);
gbc_ServerTextField.gridwidth = 2;
gbc_ServerTextField.gridx = 1;
gbc ServerTextField.gridy = 0;
contentPane.add(ServerTextField, gbc_ServerTextField);
ServerTextField.setColumns(10);

/***
* Create a JTextField for Server's Port.
*
```

```
*     +-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*     |P|o|r|t| |T|e|x|t|F|i|e|l|d|
*     +-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

PortTextField = new JTextField();
PortTextField.setColumns(10);
GridBagConstraints gbc_PortTextField = new GridBagConstraints();
gbc_PortTextField.anchor = GridBagConstraints.NORTH;
gbc_PortTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_PortTextField.insets = new Insets(0, 0, 5, 0);
gbc_PortTextField.gridwidth = 2;
gbc_PortTextField.gridx = 1;
gbc_PortTextField.gridy = 1;
contentPane.add(PortTextField, gbc_PortTextField);

/***
* Create a JTextField for Chat Message.
*
*     +-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*     |C|h|a|t| |T|e|x|t|F|i|e|l|d|
*     +-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

chatField = new JTextField();
chatField.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
z.SetMessage(chatField.getText());
}
});
chatField.setEnabled(false);
chatField.setColumns(10);
GridBagConstraints gbc_chatField = new GridBagConstraints();
gbc_chatField.fill = GridBagConstraints.HORIZONTAL;
gbc_chatField.insets = new Insets(0, 0, 0, 5);
gbc_chatField.gridx = 1;
gbc_chatField.gridy = 4;
contentPane.add(chatField, gbc_chatField);

/***
*
*    88888888ba
*    88      "8b               ,d      ,d
*    88      ,8P               88      88
*    88aaaaaa8P'  88      88  MM88MMM  MM88MMM  ,adPPYba,   8b,dPPYba,   ,adPPYba,
*    88""""""8b,  88      88    88       88    a8"     "8a  88P'   `"8a  I8[    ""
*    88      `8b  88      88    88       88    8b       d8  88       88   `"Y8ba,
*    88      a8P  "8a,   ,a88    88,      88,   "8a,   ,a8"  88       88  aa    ]8I
*    88888888P"    `"YbbdP'Y8    "Y888    "Y888  `"YbbdP"'   88       88  `"YbbdP"'
*
*
*/

/***
*
*     ____ ____ ____ ____ ________ ____ ____ ____ ____ ____ ____
*    ||J |||o |||i |||n |||      |||B |||u |||t |||t |||o |||n ||
*    ||__|||__|||__|||__|||______|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/______\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

JoinButton = new JButton("Join");

/* Add a tooltip to "Join" button. */

JoinButton.setToolTipText("Join an Server which is listening"
+ " to the port that you gave in the Port TextField.");
GridBagConstraints gbc JoinButton = new GridBagConstraints();
gbc_JoinButton.anchor = GridBagConstraints.NORTH;
gbc_JoinButton.fill = GridBagConstraints.HORIZONTAL;
gbc JoinButton.insets = new Insets(0, 0, 5, 5);
gbc_JoinButton.gridx = 1;
gbc_JoinButton.gridy = 2;
contentPane.add(JoinButton, gbc_JoinButton);
```

```
/* Add an action listener to the "Join" button. */

JoinButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

/* Check if input is correct. */

if (ServerTextField.getText() == null || ServerTextField.getText().equals("") ||
ServerTextField.getText().isEmpty()){
FullTextClient += "Please enter a hostname\r\n.";
textAreaClient.setText(FullTextClient);
return;
}
if (PortTextField.getText() == null || PortTextField.getText().equals("") ||
PortTextField.getText().isEmpty()){
FullTextClient += "Please enter a port\r\n.";
textAreaClient.setText(FullTextClient);
return;
}

/* Check if port is in use. */

try {
port = Integer.parseInt(PortTextField.getText());
if (!((port > 1025) && (port < 65536))){
port = 0;
}
} catch (NumberFormatException e) {
port = 0;
}
if (port == 0){
FullTextClient += "Please set a valid port.\r\n.";
textAreaClient.setText(FullTextClient);
return;
}
try {

/* Attempt to join the Server. */

initiateSingleConnection();
} catch (IOException e) {
e.printStackTrace();
}
}
});

/***
*     ____ ____ ____ ____ ____ _________ ____ ____ ____ ____ ____ ____
*    ||A |||b |||o |||r |||t |||       |||B |||u |||t |||t |||o |||n ||
*    ||__|||__|||__|||__|||__|||_______|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

AbortButton = new JButton("Abort");

/* Add a tooltip to "Abort" button. */

AbortButton.setToolTipText("Abort the currently connected Server.");

/* Add an action listener to the "Abort" button. */

AbortButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

/* Enable disable the appropriate buttons. */

AbortButton.setEnabled(false);
JoinButton.setEnabled(true);
SendButton.setEnabled(false);
btnRequestAFile.setEnabled(false);
chatField.setEnabled(false);
FullTextClient = FullTextClient + "Aborted from: " + ServerTextField.getText() + ":" +
port + "\r\n";
textAreaClient.setText(FullTextClient);
abortSingleConnection();

/* Shut down the connection. */
```

```
try {
if (clientSocket != null){
clientSocket.close();
clientSocket = null;
}
} catch (IOException e1) {
e1.printStackTrace();
}
return;
}
});

AbortButton.setEnabled(false);
GridBagConstraints gbc_AbortButton = new GridBagConstraints();
gbc_AbortButton.insets = new Insets(0, 0, 5, 0);
gbc_AbortButton.anchor = GridBagConstraints.NORTH;
gbc_AbortButton.fill = GridBagConstraints.HORIZONTAL;
gbc_AbortButton.gridx = 2;
gbc_AbortButton.gridy = 2;
contentPane.add(AbortButton, gbc_AbortButton);

/***
*
*    ||S |||e |||n |||d |||       |||M |||e |||s |||s |||a |||g |||e ||
*    ||__|||__|||__|||__|||_______|||__|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

SendButton = new JButton("Send");

/* Add a tooltip to "Send" button. */

SendButton.setToolTipText("Send a message (assuming non-null string in Message
TextField).");
SendButton.setEnabled(false);

/* Add an Action Listener to "Send" button */

SendButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
z.SetMessage(chatField.getText());
}
});

GridBagConstraints gbc_SendButton = new GridBagConstraints();
gbc_SendButton.anchor = GridBagConstraints.WEST;
gbc_SendButton.gridx = 2;
gbc_SendButton.gridy = 4;
contentPane.add(SendButton, gbc_SendButton);

/***
*     ____ ____ ____ ____ ____ ____ ____ _________ ____ _________ ____ ____ ____ ____
*    ||R |||e |||q |||u |||e |||s |||t |||       |||a |||       |||F |||i |||l |||e ||
*    ||__|||__|||__|||__|||__|||__|||__|||_______|||__|||_______|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/__\|/__\|/__\|/_______\|/__\|/_______\|/__\|/__\|/__\|/__\|
*/

btnRequestAFile = new JButton("Request a file");
btnRequestAFile.setToolTipText("<html> This will pop up a new dialog"
+ " which contains the follow interactions: \r\n<br></br>\r\n<br>"
+ "- Request a file list </br>\r\n<br>- Get file </br>\r\n<br>"
+ "- Cancel  </br>");
btnRequestAFile.setEnabled(false);
btnRequestAFile.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

/* Create a new dialog, containing the ClientFileList class (frame). */

try {
ClientFileList dialog = new ClientFileList();
dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
dialog.setVisible(true);

} catch (Exception e) {
e.printStackTrace();
}
```

```
}
});

GridBagConstraints gbc_btnRequestAFile = new GridBagConstraints();
gbc_btnRequestAFile.insets = new Insets(0, 0, 0, 5);
gbc_btnRequestAFile.gridx = 0;
gbc_btnRequestAFile.gridy = 4;
contentPane.add(btnRequestAFile, gbc_btnRequestAFile);
/***
*
*   888888888888                                  db
*        88                              ,d      d88b
*        88                              88     d8'`8b
*        88   ,adPPYba,  8b,     ,d8  MM88MMM   d8'  `8b     8b,dPPYba,   ,adPPYba,  ,adPPYYba,
*        88  a8P____88   `Y8, ,8P'     88      d8YaaaaY8b    88P'   "Y8  a8P____88  ""     `Y8
*        88  8PP"""""""     )888(       88     d8""""""""8b   88          8PP"""""""  ,adPPPPP88
*        88  "8b,   ,aa   ,d8" "8b,     88,   d8'        `8b  88          "8b,   ,aa  88,    ,88
*        88   `"Ybbd8"'  8P'     `Y8    "Y888 d8'          `8b 88           `"Ybbd8"'  `"8bbdP"Y8
*
*
*/

/***
*      +-+-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+
*      |M|e|s|s|a|g|e| |B|o|x| |A|r|e|a|
*      +-+-+-+-+-+-+-+-+ +-+-+-+ +-+-+-+-+
*/

/* Make Message Box scrollable. */

scrollPane = new JScrollPane();
GridBagConstraints gbc_scrollPane = new GridBagConstraints();
gbc_scrollPane.fill = GridBagConstraints.BOTH;
gbc_scrollPane.insets = new Insets(0, 0, 5, 5);
gbc_scrollPane.gridx = 1;
gbc_scrollPane.gridy = 3;
contentPane.add(scrollPane, gbc_scrollPane);

textAreaClient = new JTextArea();

/* Make Message Box uneditable. */

textAreaClient.setEditable(false);
textAreaClient.setEnabled(false);
scrollPane.setViewportView(textAreaClient);
}

/**
*
* Instantiate a new Settings class (please see the Settings class)
* By doing this the program's settings will be running as intended.
*
* @param frame The Client's main class (frame).
*/

public static void instantiateClientSettings(Client frame){
settings = new ClientSettings();
try {
if (ClientSettings.DoesFileExist()) {
if (ClientSettings.ParseExistingFile(settings) == false){
File file = new File("./" + settings.getFileName());
file.delete();
settings.makeFile(settings);
}
}else{
settings.makeFile(settings);
}
} catch (IOException e2) {
JOptionPane.showMessageDialog(null,
"Cannot write Settings.cfg\nAccess denied.\n.",
"Write Error",
JOptionPane.ERROR MESSAGE);
frame.setVisible(false);
frame.dispose();
e2.printStackTrace();
}
}

/**
* Attempt to connect to the Server listening to the given port.
* And all the needed procedures are awaiting to be executed upon client's successful
connection.
```

```
* @throws IOException In case of error.
*/

public void initiateSingleConnection() throws IOException {

/* Enable/Disable the appropriate buttons and fields. */

AbortButton.setEnabled(true);
JoinButton.setEnabled(false);
SendButton.setEnabled(true);
btnRequestAFile.setEnabled(true);
chatField.setEnabled(true);

/* Create a new swing worker - required for multithreading. */

workerClient = new SwingWorker<Integer, String>() {
protected Integer doInBackground() throws Exception {
try {
Boolean only_repeat_once = false;

/* Connect to the listening socket. */

clientSocket = new Socket(ServerTextField.getText(), port);

/* Publish it (See the process() method below). */

publish("Connected to: " + ServerTextField.getText() + ":" + port);

/* Start Client handlers and output handler threads. */

new Thread(new ClientInputHandler(clientSocket)).start();
z = new ClientOutputHandler(clientSocket);
Thread c = new Thread(z);
c.start();

/* Send the client's settings only once (see the boolean variable above) */

if (only_repeat_once == false){
DataOutputStream outToServer = null;
try {
outToServer = new DataOutputStream(clientSocket.getOutputStream());
} catch (IOException e3) {
e3.printStackTrace();
}
try {
if (settings.isSnappingEnabled() == true){
outToServer.writeBytes("~se\r\n");
}else{
outToServer.writeBytes("~sd\r\n");
}
outToServer.flush();
} catch (IOException e4) {
e4.printStackTrace();
}
try {
if (settings.isSnappingEnabled() == true){
outToServer.writeBytes("~" + settings.getSeconds() + "\r\n");
outToServer.flush();
}
} catch (IOException e6) {
e6.printStackTrace();
}
try {
if (settings.isSnappingEnabled() == true){
outToServer.writeBytes("~w" + settings.getpercentage() + "\r\n");
outToServer.flush();
}
} catch (IOException e8) {
e8.printStackTrace();
}
only_repeat_once = true;
}

/* Terminate the connection in-case of error. */

} catch (UnknownHostException ex) {
publish("Unable to connect to: " + ServerTextField.getText() + ":" + port);
```

```
AbortButton.setEnabled(false);
JoinButton.setEnabled(true);
SendButton.setEnabled(false);
btnRequestAFile.setEnabled(false);
chatField.setEnabled(false);
try {
if (clientSocket != null){
clientSocket.close();
clientSocket = null;
}
} catch (IOException e1) {
e1.printStackTrace();
}

/* The same. */

} catch (IOException ex) {
publish("Unable to connect to: " + ServerTextField.getText() + ":" + port);
AbortButton.setEnabled(false);
JoinButton.setEnabled(true);
SendButton.setEnabled(false);
btnRequestAFile.setEnabled(false);
chatField.setEnabled(false);
try {
if (clientSocket != null){
clientSocket.close();
clientSocket = null;
}
} catch (IOException e1) {
e1.printStackTrace();
}
}
return 1;
}
protected void done(){
try {
} catch (CancellationException e) {
return;
}
}
protected void process(List<String> chunks) {

// Here we receive the values that we publish().

for (final String string : chunks) {
FullTextClient = FullTextClient + string + "\r\n";
textAreaClient.setText(FullTextClient);
}
}
};
workerClient.execute();
}

/**
* Terminates the connection.
*/

public void abortSingleConnection(){
ClientConnected = false;
workerClient.cancel(true);
}

/**
* SocketSnapshot's Client's Input Handler (sub-)class
*
* Note: Because this class works side-by-side with Client class,
* it's not set "public" (thus no separate class file is required).
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

public static class ClientInputHandler implements Runnable {
private Socket socket;

public ClientInputHandler(Socket socket) {
```

```
this.socket = socket;
}

public void run() {
ClientConnected = true;
BufferedReader reader = null;
try {
reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
while (ClientConnected) {
String text = "";
if (ClientConnected == true){
text = reader.readLine();
}

/* If given (Server output) string is NOT empty. */

if ((text != null) && !(text.equals("")) && !(text.isEmpty())){

/* If the prefix is "#" then it's the file send command. */

if ((text.charAt(0) == '#') && (text.length() > 1)){
String filename = text.substring(1,text.length());
Client.FullTextClient =  Client.FullTextClient + "Attempting to recieve " + filename +
"\r\n";
Client.textAreaClient.setText(Client.FullTextClient);
new Thread(new ClientRecieveHandle(port+1,filename)).start();
continue;

/* If the prefix is "|" then it's the command whereas
* the file list (one string line means one file/folder) is sent. */

}else if ((text.charAt(0) == '|') && (text.length() == 1)){
ClientFileList.model.addElement(text.substring(1,text.length()));
continue;

/* Else if the prefix is "&" it means that it's the warn command. */

}else if ((text.charAt(0) == '&') && (text.length() > 1)){
Date date = new Date();
Client.FullTextClient =  Client.FullTextClient + "A dangerous " + text.substring(1) +
"% was noticed at " + date.toString() + "\r\n";
Client.textAreaClient.setText(Client.FullTextClient);
continue;
}
}
if ((text == null || text.equals("") || text.isEmpty())){
continue;
}

/* Else just print the input to the message box. */

Client.FullTextClient =  Client.FullTextClient + "#Server said: " + text + "\r\n";
Client.textAreaClient.setText( Client.FullTextClient);
}
reader.close();
}catch (Exception e) {
}
}
}

/**
* SocketSnapshot's Client's Output Handler (sub-)class
*
* Note: Because this class works side-by-side with Client class,
* it's not set "public" (thus no separate class file is required).
*
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
* @see     ftp.ServerOutputHandler
*/

class ClientOutputHandler implements Runnable {
private Socket socket;
private String message;
private boolean newmsg;
```

```
public ClientOutputHandler(Socket socket) {
this.socket = socket;
this.message = "";
this.newmsg = false;
}

public void SetMessage(String message) {
this.message = message;
this.newmsg = true;
}

public void run() {
BufferedWriter writerClient = null;
try {
writerClient = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
while(ClientConnected){
Thread.sleep(1);
if ((message == null || message.equals("") || message.isEmpty())){
continue;
}else{
Thread.sleep(1);
if (newmsg == true){
Client.FullTextClient =  Client.FullTextClient + "#You said: " + message + "\r\n";
Client.textAreaClient.setText(Client.FullTextClient);
writerClient.write(message);
writerClient.newLine();
writerClient.flush();
newmsg = false;
chatField.setText("");
}
}
}
} catch (Exception exp) {
} finally {
}
}
}
}

/**
* SocketSnapshot's Client's File Recieve Handler (sub-)class
*
* Note: Because this class works side-by-side with Client class,
* it's not set "public" (thus no separate class file is required).
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
* @see     ftp.ServerSendHandle
*/

class ClientRecieveHandle implements Runnable {

private int port2;
private String filename;

public ClientRecieveHandle(int port2,String filename) {
this.port2 = port2;
this.filename = filename;
}

public void run() {
ServerSocket RecieveSocket = null;
try {
RecieveSocket = new ServerSocket(port2);
} catch (IOException e) {
e.printStackTrace();
}
Socket connectionSocket = null;
try {
connectionSocket = RecieveSocket.accept();
} catch (IOException e) {
e.printStackTrace();
}
InputStream is = null;
try {
```

```
is = connectionSocket.getInputStream();
} catch (IOException e) {
e.printStackTrace();
}
int filesize = 6022386;
int bytesRead = 0;
int current = 0;
byte[] mybytearray = new byte[filesize];
FileOutputStream fos = null;
try {
File file = new File(Client.settings.getRcvdFolder() + "\\" + filename);
fos = new FileOutputStream(file);
} catch (FileNotFoundException e) {
e.printStackTrace();
}
BufferedOutputStream bos = new BufferedOutputStream(fos);
try {
bytesRead = is.read(mybytearray, 0, mybytearray.length);
} catch (IOException e) {
e.printStackTrace();
}
current = bytesRead;
do{
try {
bytesRead = is.read(mybytearray, current,(mybytearray.length - current));
} catch (IOException e) {
e.printStackTrace();
}
if (bytesRead >= 0){
current += bytesRead;
}
}while(bytesRead > -1);
try {
bos.write(mybytearray, 0, current);
} catch (IOException e) {
e.printStackTrace();
}
Client.FullTextClient =  Client.FullTextClient + "Successfully recieved " + filename +
"\r\n";
Client.textAreaClient.setText(Client.FullTextClient);
try {
bos.flush();
} catch (IOException e) {
e.printStackTrace();
}
try {
bos.close();
} catch (IOException e) {
e.printStackTrace();
}
try {
connectionSocket.close();
} catch (IOException e) {
e.printStackTrace();
}
try {
RecieveSocket.close();
} catch (IOException e) {
e.printStackTrace();
}
}

}
```

ClientFileList.java

```
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.DataOutputStream;
import java.io.File;
import java.io.IOException;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.border.EmptyBorder;

/**
* SocketSnapshot's Client's File List Dialog
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

@SuppressWarnings("serial")
public class ClientFileList extends JDialog {

/* Private variables. */

private JButton okButton;
private JButton cancelButton;
private final JPanel contentPanel = new JPanel();

/* Public variables. */

public File File1;
public static DefaultListModel<String> model;
public JList<String> list;

/**
* Creates the dialog.
*/

public ClientFileList() {
setBounds(100, 100, 222, 384);
GridBagLayout gridBagLayout = new GridBagLayout();
gridBagLayout.columnWidths = new int[]{206, 0};
gridBagLayout.rowHeights = new int[]{47, 266, 23, 0};
gridBagLayout.columnWeights = new double[]{0.0, Double.MIN_VALUE};
gridBagLayout.rowWeights = new double[]{0.0, 0.0, 0.0, Double.MIN_VALUE};
getContentPane().setLayout(gridBagLayout);
{

/***
*     ____ ____ ____ ____ ____ ____ ____ ________ ____ ____ ____ ____ ____ ____ ____ ____
*    ||R |||e |||q |||u |||e |||s |||t |||        |||F |||i |||l |||e |||l |||i |||s |||t ||
*    ||__|||__|||__|||__|||__|||__|||__|||________|||__|||__|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/__\|/__\|/__\|/________\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

JButton btnRequestFilelist = new JButton("Request Filelist");

/* Add a tooltip. */
```

```
btnRequestFilelist.setToolTipText("Request a filelist from the Server and "
+ "print it to the one (and only) messagebox. (Tip: It might need "
+ "2 keypresses in order to successfully recieve the filelist)");


GridBagConstraints gbc_btnRequestFilelist = new GridBagConstraints();
gbc_btnRequestFilelist.fill = GridBagConstraints.BOTH;
gbc_btnRequestFilelist.insets = new Insets(0, 0, 5, 0);
gbc_btnRequestFilelist.gridx = 0;
gbc_btnRequestFilelist.gridy = 0;
getContentPane().add(btnRequestFilelist, gbc_btnRequestFilelist);
btnRequestFilelist.addActionListener(new ActionListener() {

/* Action Listener */

public void actionPerformed(ActionEvent arg0) {

/* Request a filelist from the Server and print it to the one (and only) messagebox
* (Tip: It might need 2 keypresses in order to successfully recieve the filelist).  */

DataOutputStream outToServer = null;
try {
outToServer = new DataOutputStream(Client.clientSocket.getOutputStream());
} catch (IOException e3) {
e3.printStackTrace();
}
try {
outToServer.writeBytes("#\r\n");
} catch (IOException e2) {
e2.printStackTrace();
}
try {
outToServer.flush();
} catch (IOException e1) {
e1.printStackTrace();
}
okButton.setEnabled(true);
model.clear();
}
});
}
{
model = new DefaultListModel<String>();
}
contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
GridBagConstraints gbc_contentPanel = new GridBagConstraints();
gbc_contentPanel.fill = GridBagConstraints.HORIZONTAL;
gbc_contentPanel.insets = new Insets(0, 0, 5, 0);
gbc_contentPanel.gridx = 0;
gbc_contentPanel.gridy = 1;
getContentPane().add(contentPanel, gbc_contentPanel);
GridBagLayout gbl_contentPanel = new GridBagLayout();
gbl_contentPanel.columnWidths = new int[]{0, 0};
gbl_contentPanel.rowHeights = new int[]{257, 0};
gbl_contentPanel.columnWeights = new double[]{1.0, Double.MIN_VALUE};
gbl_contentPanel.rowWeights = new double[]{0.0, Double.MIN_VALUE};
contentPanel.setLayout(gbl_contentPanel);
{
JScrollPane scrollPane = new JScrollPane();
GridBagConstraints gbc_scrollPane = new GridBagConstraints();
gbc_scrollPane.fill = GridBagConstraints.BOTH;
gbc_scrollPane.gridx = 0;
gbc_scrollPane.gridy = 0;
contentPanel.add(scrollPane, gbc_scrollPane);
{
list = new JList<String>( model );
scrollPane.setViewportView(list);
list.addMouseListener(new MouseAdapter() {
@SuppressWarnings("unused")
public void mouseClicked(MouseEvent evt) {
JList<?> list2 = (JList<?>)evt.getSource();
if (evt.getClickCount() == 2) {
String filename = list.getSelectedValue().toString();
try {
DataOutputStream outToServer = new
DataOutputStream(Client.clientSocket.getOutputStream());
outToServer.writeBytes("!" + filename +"\r\n");
```

```
outToServer.flush();
} catch (IOException e) {
e.printStackTrace();
}
JOptionPane.showMessageDialog(null,"Started recieving " + filename + ". Please
consider checking the messagebox for more information about the progress.");
}
}

});
}
}
{
JPanel buttonPane = new JPanel();
GridBagConstraints gbc_buttonPane = new GridBagConstraints();
gbc_buttonPane.anchor = GridBagConstraints.NORTH;
gbc_buttonPane.fill = GridBagConstraints.HORIZONTAL;
gbc_buttonPane.gridx = 0;
gbc_buttonPane.gridy = 2;
getContentPane().add(buttonPane, gbc_buttonPane);
GridBagLayout gbl_buttonPane = new GridBagLayout();
gbl_buttonPane.columnWidths = new int[]{101, 100, 0};
gbl_buttonPane.rowHeights = new int[]{23, 0};
gbl_buttonPane.columnWeights = new double[]{0.0, 0.0, Double.MIN_VALUE};
gbl_buttonPane.rowWeights = new double[]{0.0, Double.MIN_VALUE};
buttonPane.setLayout(gbl_buttonPane);
{

/***
*      ___ ___ ___
*     ||G |||e |||t ||
*     ||__|||__|||__||
*     |/__\|/__\|/__\|
*/

okButton = new JButton("Get");

/* Add a tooltip. */

okButton.setToolTipText("Get (attempt to receive) the SELECTED file from the list in
the TextArea. ");

/* Add an Action Listener */

okButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

/* Get (attempt to receive) the SELECTED file from the list in the TextArea. */

String filename = list.getSelectedValue().toString();
try {
DataOutputStream outToServer = new
DataOutputStream(Client.clientSocket.getOutputStream());
outToServer.writeBytes("!" + filename +"\r\n");
outToServer.flush();
} catch (IOException e) {
e.printStackTrace();
}
JOptionPane.showMessageDialog(null,"Started recieving " + filename + ". Please
consider checking the messagebox for more information about the progress.");
}
});
okButton.setEnabled(false);
GridBagConstraints gbc_okButton = new GridBagConstraints();
gbc_okButton.fill = GridBagConstraints.HORIZONTAL;
gbc_okButton.anchor = GridBagConstraints.NORTH;
gbc_okButton.insets = new Insets(0, 0, 0, 5);
gbc_okButton.gridx = 0;
gbc_okButton.gridy = 0;
buttonPane.add(okButton, gbc_okButton);
getRootPane().setDefaultButton(okButton);
}
{
```

```
/***
*      ____ ____ ____ ____ ____ ____
*     ||C |||a |||n |||c |||e |||l ||
*     ||__|||__|||__|||__|||__|||__||
*     |/__\|/__\|/__\|/__\|/__\|/__\|
*/

cancelButton = new JButton("Cancel");

/* Add a tooltip. */

cancelButton.setToolTipText("Suspends this dialog.");

/* Add an Action Listener. */

cancelButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
setVisible(false);
dispose();
}
});
cancelButton.setActionCommand("Cancel");
GridBagConstraints gbc_cancelButton = new GridBagConstraints();
gbc_cancelButton.fill = GridBagConstraints.HORIZONTAL;
gbc_cancelButton.anchor = GridBagConstraints.NORTH;
gbc_cancelButton.gridx = 1;
gbc_cancelButton.gridy = 0;
buttonPane.add(cancelButton, gbc_cancelButton);
}
}
}

}
```

ClientSettings.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;

/**
* SocketSnapshot's Client's Settings class.
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

public class ClientSettings {
private Boolean snapping;
private String Received_Files_Folder;
private String Snapshot_Folder;
private double perc;
private int snapping_seconds;
private static String filename;

/**
* The constructor.
*/

public ClientSettings(){
Path relativePath = Paths.get("./");
Path fullPath = relativePath.toAbsolutePath();
String str = fullPath.toString();
filename = "Settings.cfg";
this.Received_Files_Folder = str.substring(0,str.length()-2);
this.Snapshot_Folder = str.substring(0,str.length()-1);
snapping = true;
snapping_seconds = 60;
perc = (double)10;
}

/**
* Self-explanatory.
* @return Returns true if  file exists, false otherwise.
*/

public static Boolean DoesFileExist(){
File f = new File("./" + filename);
if (f.exists() && !f.isDirectory()) {
return true;
}
return false;
}

/**
*
* Parse a single String line and check whether it is a valid settings configuration
command
* or not.
*
* @param left The substring left of "=".
* @param right The substring right of "=".
* @return True if correct line, false otherwise.
*/

public Boolean ParseLine(String left,String right){
if ((left.equals("[Received_Files_Folder]")
|| (left.equals("[Snapshot_Folder]")))){
if (right == null || right.equals("") || right.isEmpty()){
if (left.equals("[Snapshot_Folder]")){
setSnapshotFolder("");
return true;
```

```
}
return false;
}else{
File f = new File(right);
if(f.exists() && f.isDirectory()){
if (left.equals("[Received_Files_Folder]")){
setRcvdFolder(right);
}else{
setSnapshotFolder(right);
}
return true;
}
return false;
}
}else if (((left.equals("[snapping]")))){
if (right.equals("true") || (right.equals("false"))){
if (right.equals("true")){
setSnapping(true);
}else{
setSnapping(false);
}
return true;
}
}else if (left.equals("[snapping_seconds]")){
int number = 0;
try {
number = Integer.parseInt(right);
if (!((number > 1) && (number < 2000))){
return false;
}
} catch (NumberFormatException e) {
return false;
}
setSeconds(number);
return true;
}else if (left.equals("[percentage]")){
double number = 0;
try {
number = Double.parseDouble(right);
if (!((number >= (double)0) && (number <= (double)100))){
return false;
}
} catch (NumberFormatException e) {
return false;
}
setpercentage(number);
return true;
}
return false;
}

/**
*
* Parses a whole Client Settings file.
*
* @param settings The ClientSettings to be parsed.
* @return True if the settings were in correct format, false otherwise.
*/

public static Boolean ParseExistingFile(ClientSettings settings){
if (!DoesFileExist()) {
return false;
}
List<String> strings = new ArrayList<String>();
BufferedReader br = null;
try {
String sCurrentLine;
br = new BufferedReader(new FileReader("./" + filename));
while ((sCurrentLine = br.readLine()) != null) {
strings.add(sCurrentLine);
}
} catch (IOException e) {
e.printStackTrace();
} finally {
try {
if (br != null)br.close();
} catch (IOException ex) {
```

```
ex.printStackTrace();
}
}
if (strings.size() != 7){
return false;
}
for (int i = 0; i < strings.size(); i++){
String current = strings.get(i);
int charCount = current.replaceAll("[^=]", "").length();
if (charCount != 1){

return false;
}
String[] parts =  strings.get(i).split("=");
if (parts.length != 1 && parts.length != 2){
return false;
}
if (parts.length == 2){
if (settings.ParseLine(parts[0],parts[1]) == false){
return false;
}
}else{
if (settings.ParseLine(parts[0],"") == false){
return false;
}
}
}

/* check if everything is filled up */

if ((settings.isSnappingEnabled() == true) && ((settings.getSnapshotFolder() != null)
|| (!(settings.getSnapshotFolder().equals(""))) ||
!(settings.getSnapshotFolder().isEmpty()))){
return false;
}
return true;
}

/**
* Creates a new ClientSettings file (using the current ClientSettings in this class).
* @param settings The ClientSettings to be created.
* @throws IOException In case of exception.
*/

public void makeFile(ClientSettings settings) throws IOException{
if (DoesFileExist()) {
File file = new File("./" + settings.getFileName());
file.delete();
makeFile(settings);
}else{
String content = "";
content += "[Received_Files_Folder]=" + settings.getRcvdFolder() + "\r\n";
content += "[Snapshot_Folder]=" + settings.getSnapshotFolder() + "\r\n";
content += "[snapping]=" + settings.isSnappingEnabled() + "\r\n";
content += "[snapping_seconds]=" + settings.getSeconds() + "\r\n";
content += "[percentage]=" + settings.getpercentage();
Files.write(Paths.get("./" + settings.getFileName()), content.getBytes());
}
}

/**
* @return Set the Client Settings filename.
*/

public String getFileName(){
return filename;
}

/**
*
* @param FolderPath Set the Client's Received Files Folder Path.
*/

public void setRcvdFolder(String FolderPath){
this.Received_Files_Folder = FolderPath;
}
```

```
/**
*
* @param FolderPath Set the Client's Snapshot Folder Path.
*/

public void setSnapshotFolder(String FolderPath){
this.Snapshot_Folder = FolderPath;
}

/**
*
* @param seconds Set the "take a snapshot every x" seconds to be set.
*/

public void setSeconds(int seconds){
this.snapping_seconds = seconds;
}

/**
*
* @param value Set whether Client Snapshots are enabled.
*/

public void setSnapping(Boolean value){
this.snapping = value;
}

/**
*
* @param value Set Snapshot Warning threshold.
*/

public void setpercentage(double value){
this.perc = value;
}

/**
*
* @return The Client's Received Files Folder Path.
*/

public String getRcvdFolder(){
return Received_Files_Folder;
}

/**
*
* @return The Client's Snapshot Folder Path.
*/

public String getSnapshotFolder(){
return Snapshot_Folder;
}

/**
*
* @return True if Client Snapshots are enabled, false otherwise.
*/

public Boolean isSnappingEnabled(){
return snapping;
}

/**
*
* @return The "Take a snapshot every x" seconds.
*/

public int getSeconds(){
return snapping_seconds;
}

/**
*
* @return The Snapshot Sending (Percentage thresohold).
*/
```

```
public double getpercentage(){
return perc;
}
}
```

ClientSettingsPopUp.java

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.GridBagLayout;
import javax.swing.JButton;
import java.awt.GridBagConstraints;
import javax.swing.JTextField;
import java.awt.Insets;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JRootPane;
import javax.swing.JSlider;
import javax.swing.JLabel;
import javax.swing.JCheckBox;
import java.util.concurrent.TimeUnit;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Color;
import java.io.DataOutputStream;
import java.io.IOException;

/**
* SocketSnapshot's Client Settings Popup
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/
public class ClientSettingsPopup extends JFrame {

/* Private Variables - JAVA SWING (GUI). */

private JButton btnSaveAndClose;
private JButton btnSetSnapshotFolder;
private JCheckBox SnapshotCheckBox;
private JLabel lblFolderSettings;
private JLabel lblperc;
private JLabel lblSeconds;
private JLabel lblSet;
private JLabel lblSnapshotsEnabled;
private JLabel lblSnapshotSettings;
private JLabel lblWarnMeIf;
private JLabel WarningLabel;
private JPanel contentPane;
private JSlider slider;
private JSlider slider_1;
private JTextField RecievedFilesTextField;
private JTextField SnapshotFolderTextField;

/* Serials.
*
* SerialVersionUID is a unique identifier for each class,
*  by which Java Virtual Machine compares the versions of class.
*  If we don't provide this field explicitly , Java Virtual Machine
*  generates it automatically for every class . The Algorithm for
*  generating SerialVersionUID is highly dependent on the Java Virtual Machine
version,
*  and class structure.
*/
```

```
private static final long serialVersionUID = 1L;

/* Colors. */

public Color black = new Color(0,0,0);
public Color green = new Color(0,255,0);
public Color red = new Color(255,0,0);

/**
* Creates the Client Settings Popup.
*/
public ClientSettingsPopup() {
this.setTitle("SocketSnapshot (Client) Settings");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setUndecorated(true);
getRootPane().setWindowDecorationStyle(JRootPane.INFORMATION_DIALOG);
setBounds(100, 100, 503, 346);
contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);
GridBagLayout gbl contentPane = new GridBagLayout();
gbl_contentPane.columnWidths = new int[]{14, 141, 0, 0, 0, 0};
gbl_contentPane.rowHeights = new int[]{0, -11, 8, 0, 27, 0, 0, 0, 37, 0, 0, 0, 0, 0,
20, 0};
gbl_contentPane.columnWeights = new double[]{0.0, 0.0, 1.0, 1.0, 0.0,
Double.MIN_VALUE};
gbl contentPane.rowWeights = new double[]{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, Double.MIN_VALUE};
contentPane.setLayout(gbl_contentPane);


/***
*
*     ad88888ba   88  88           88
*    d8"     "8b  88  ""           88
*    Y8,          88               88
*    `Y8aaaaa,    88  88   ,adPPYb,88   ,adPPYba,  8b,dPPYba,  ,adPPYba,
*      `"""""8b,  88  88  a8"    `Y88  a8P_____88  88P'   "Y8  I8[    ""
*            `8b  88  88  8b       88  8PP"""""""  88           `"Y8ba,
*    Y8a     a8P  88  88  "8a,   ,d88  "8b,   ,aa  88          aa    ]8I
*     "Y88888P"   88  88   `"8bbdP"Y8   `"Ybbd8"'  88          `"YbbdP"'
*
*
*/

/***
*      +-+-+-+-+-+-+-+
*      |S|l|i|d|e|r|
*      +-+-+-+-+-+-+-+
*/

slider = new JSlider();
slider.setMinimum(1);
//check all files for 2000
slider.setMaximum(2000);
//if settings don't exist
slider.setValue(Client.settings.getSeconds());
// else get settings from file
slider.addChangeListener(new ChangeListener() {
public void stateChanged(ChangeEvent e) {
JSlider source = (JSlider)e.getSource();
if (!source.getValueIsAdjusting()) {
int time_wanted = (int)source.getValue();
String temp2 = getTIME(time_wanted);
lblSeconds.setText(temp2);
}
}
});
GridBagConstraints gbc slider = new GridBagConstraints();
gbc_slider.gridwidth = 2;
gbc_slider.insets = new Insets(0, 0, 5, 5);
gbc slider.fill = GridBagConstraints.BOTH;
gbc_slider.gridx = 2;
gbc_slider.gridy = 9;
contentPane.add(slider, gbc_slider);
```

```
/***
*      +-+-+-+-+-+-+-+ +-+
*      |S|l|i|d|e|r| |1|
*      +-+-+-+-+-+-+-+ +-+
*/

slider_1 = new JSlider();
slider_1.setValue((int) Client.settings.getpercentage());
slider_1.addChangeListener(new ChangeListener() {
public void stateChanged(ChangeEvent e) {
JSlider source = (JSlider)e.getSource();
if (!source.getValueIsAdjusting()) {
String temp3 = (double)source.getValue() + "%";
lblperc.setText(temp3);
}
}
});
GridBagConstraints gbc_slider_1 = new GridBagConstraints();
gbc_slider_1.fill = GridBagConstraints.HORIZONTAL;
gbc_slider_1.gridwidth = 2;
gbc_slider_1.insets = new Insets(0, 0, 5, 5);
gbc_slider_1.gridx = 2;
gbc_slider_1.gridy = 11;
contentPane.add(slider_1, gbc_slider_1);

/***
*
*    88                    88                    88
*    88                    88                    88
*    88                    88                    88
*    88          ,adPPYYba,  88,dPPYba,    ,adPPYba,  88  ,adPPYba,
*    88          ""     `Y8  88P'    "8a  a8P_____88  88  I8[    ""
*    88          ,adPPPPP88  88       d8  8PP"""""""  88   `"Y8ba,
*    88          88,    ,88  88b,   ,a8"  "8b,   ,aa  88  aa    ]8I
*    88888888888  `"8bbdP"Y8  8Y"Ybbd8"'    `"Ybbd8"'  88  `"YbbdP"'
*
*
*/

/***
*
* Create a JLabel "Warning".
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |W|a|r|n|i|n|g| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

WarningLabel = new JLabel("");
WarningLabel.setForeground(Color.RED);
GridBagConstraints gbc_WarningLabel = new GridBagConstraints();
gbc_WarningLabel.anchor = GridBagConstraints.WEST;
gbc_WarningLabel.gridwidth = 2;
gbc_WarningLabel.insets = new Insets(0, 0, 5, 5);
gbc_WarningLabel.gridx = 1;
gbc_WarningLabel.gridy = 2;
contentPane.add(WarningLabel, gbc_WarningLabel);

/***
*
* Create a JLabel "Folder Settings".
*      +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |F|o|l|d|e|r| |S|e|t|t|i|n|g|s| |L|a|b|e|l|
*      +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblFolderSettings = new JLabel("Folder Settings");
GridBagConstraints gbc_lblFolderSettings = new GridBagConstraints();
gbc_lblFolderSettings.gridwidth = 3;
gbc_lblFolderSettings.insets = new Insets(0, 0, 5, 5);
gbc_lblFolderSettings.gridx = 1;
gbc_lblFolderSettings.gridy = 4;
contentPane.add(lblFolderSettings, gbc_lblFolderSettings);

/***
* Create a JLabel "Snapshot Settings".
*      +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |S|n|a|p|s|h|o|t| |S|e|t|t|i|n|g|s| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+
```

```
* With the JLabel class, you can display unselectable text and images.
*/

lblSnapshotSettings = new JLabel("Snapshot Settings");
GridBagConstraints gbc_lblSnapshotSettings = new GridBagConstraints();
gbc_lblSnapshotSettings.gridwidth = 3;
gbc_lblSnapshotSettings.insets = new Insets(0, 0, 5, 5);
gbc_lblSnapshotSettings.gridx = 1;
gbc_lblSnapshotSettings.gridy = 7;
contentPane.add(lblSnapshotSettings, gbc_lblSnapshotSettings);

/***
*
* Create a JLabel "Snapshots Enabled".
*     +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
*     |S|n|a|p|s|h|o|t|s| |E|n|a|b|l|e|d|
*     +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblSnapshotsEnabled = new JLabel("Snapshots enabled?");
GridBagConstraints gbc_lblSnapshotsEnabled = new GridBagConstraints();
gbc_lblSnapshotsEnabled.insets = new Insets(0, 0, 5, 5);
gbc lblSnapshotsEnabled.gridx = 1;
gbc_lblSnapshotsEnabled.gridy = 8;
contentPane.add(lblSnapshotsEnabled, gbc_lblSnapshotsEnabled);

/***
*     +-+-+-+-+ +-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+ +-+-+-+-+-+
*     |T|a|k|e| |a| |S|n|a|p|s|h|o|t| |E|v|e|r|y| |L|a|b|e|l|
*     +-+-+-+-+ +-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+ +-+-+-+-+-+
*/

lblSet = new JLabel("Take a snapshot every");
GridBagConstraints gbc_lblSet = new GridBagConstraints();
gbc lblSet.insets = new Insets(0, 0, 5, 5);
gbc_lblSet.gridx = 1;
gbc_lblSet.gridy = 9;
contentPane.add(lblSet, gbc_lblSet);

/***
*
* Create a JLabel "Seconds".
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+
*     |S|e|c|o|n|d|s| |L|a|b|e|l|
*     +-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblSeconds = new JLabel("");
GridBagConstraints gbc_lblSeconds = new GridBagConstraints();
gbc_lblSeconds.gridwidth = 2;
gbc_lblSeconds.insets = new Insets(0, 0, 5, 5);
gbc_lblSeconds.gridx = 2;
gbc_lblSeconds.gridy = 10;
contentPane.add(lblSeconds, gbc_lblSeconds);
String temp = getTIME(slider.getValue());
lblSeconds.setText(temp);

/***
*
* Create a JLabel "Warn me if".
*     +-+-+-+-+ +-+-+ +-+-+ +-+-+-+-+-+
*     |W|a|r|n| |m|e| |i|f| |L|a|b|e|l|
*     +-+-+-+-+ +-+-+ +-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblWarnMeIf = new JLabel("Warn me if % difference exceeds");
GridBagConstraints gbc lblWarnMeIf = new GridBagConstraints();
gbc_lblWarnMeIf.insets = new Insets(0, 0, 5, 5);
gbc_lblWarnMeIf.gridx = 1;
gbc lblWarnMeIf.gridy = 11;
contentPane.add(lblWarnMeIf, gbc_lblWarnMeIf);

/***
*
```

```
* Create a JLabel "Percentage".
*      +-+-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+
*      |P|e|r|c|e|n|t|a|g|e| |L|a|b|e|l|
*      +-+-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+
* With the JLabel class, you can display unselectable text and images.
*/

lblperc = new JLabel("");
lblperc.setText((double)slider_1.getValue()+"%");
GridBagConstraints gbc_lblperc = new GridBagConstraints();
gbc_lblperc.gridwidth = 2;
gbc_lblperc.insets = new Insets(0, 0, 5, 5);
gbc_lblperc.gridx = 2;
gbc_lblperc.gridy = 12;
contentPane.add(lblperc, gbc_lblperc);

/***
*
*    88888888ba
*    88      "8b                ,d       ,d
*    88      ,8P                88       88
*    88aaaaaa8P'  88       88  MM88MMM  MM88MMM  ,adPPYba,   8b,dPPYba,   ,adPPYba,
*    88""""""8b,  88       88    88       88    a8"     "8a  88P'    `"8a  I8[    ""
*    88      `8b  88       88    88       88    8b       d8  88       88   `"Y8ba,
*    88      a8P  "8a,   ,a88    88,      88,   "8a,   ,a8"  88       88  aa    ]8I
*    88888888P"    `"YbbdP'Y8    "Y888     "Y888  `"YbbdP"'   88       88  `"YbbdP"'
*
*
*/

/***
*   ||S ||||e ||||t ||||       ||||R ||||e ||||c ||||e ||||i ||||v ||||e ||||d ||||       ||||F ||||i ||||l ||||e ||||s ||||       ||||F ||||o ||||l ||||d ||||e ||||r ||||
*   ||__||||__||||__||||_____||||__||||__||||__||||__||||__||||__||||__||||__||||_____||||__||||__||||__||||__||||_____||||__||||__||||__||||__||||__||||__||||
*   |/__\||/__\||/__\||_____\||/__\||/__\||/__\||/__\||/__\||/__\||/__\||/__\||/_____\||/__\||/__\||/__\||/__\||/_____\||/__\||/__\||/__\||/__\||/__\||/__\||/__\|
*/

JButton btnSetRecievedFiles = new JButton("Set Recieved Files folder...");
btnSetRecievedFiles.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
final JFileChooser chooser = new JFileChooser() {
private static final long serialVersionUID = 1L;

public void approveSelection() {
if (getSelectedFile().isFile()) {
return;
} else
super.approveSelection();
}
};
chooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
int returnVal = chooser.showOpenDialog(getParent());
if (chooser.getSelectedFile() == null){
return;
}
String absolute = chooser.getSelectedFile().getAbsolutePath();
if(returnVal == JFileChooser.APPROVE_OPTION) {
WarningLabel.setText("");
RecievedFilesTextField.setForeground(black);
RecievedFilesTextField.setText(absolute);
}
}
});
btnSetRecievedFiles.setToolTipText("<html>Click to set/change the folder where the
recieved files and folders (in .ZIP format)\r\n<br>\r\nshould be
saved.</br>\r\n<br></br>\r\n<br></br>\r\nThe root directory of the folder that this
application was launcher is the default.</br>\r\n</html>");
GridBagConstraints gbc_btnSetRecievedFiles = new GridBagConstraints();
gbc btnSetRecievedFiles.fill = GridBagConstraints.HORIZONTAL;
gbc_btnSetRecievedFiles.insets = new Insets(0, 0, 5, 5);
gbc_btnSetRecievedFiles.gridx = 1;
gbc btnSetRecievedFiles.gridy = 5;
contentPane.add(btnSetRecievedFiles, gbc_btnSetRecievedFiles);

/***
*
*    ||S |||e |||t |||       |||S |||n |||a |||p |||s |||h |||o |||t |||       |||F |||o |||l |||d |||e |||r ||
*    ||__|||__|||__|||_____|||__|||__|||__|||__|||__|||__|||__|||__|||_____|||__|||__|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|_____\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|/__\|_____\|/__\|/__\|/__\|/__\|/__\|/__\|
*/

btnSetSnapshotFolder = new JButton("Set Snapshot folder...");
btnSetSnapshotFolder.addActionListener(new ActionListener() {
```

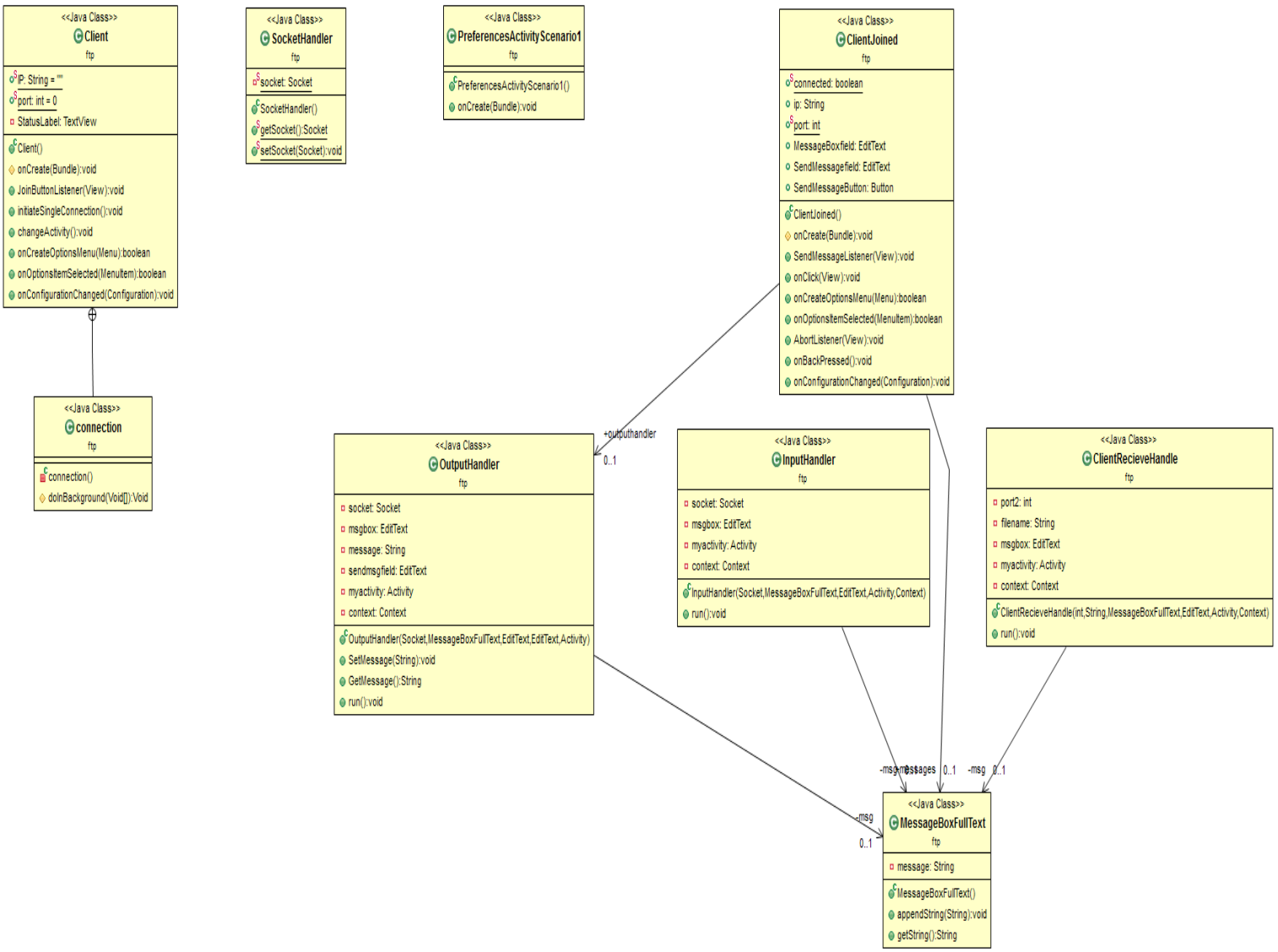
```
public void actionPerformed(ActionEvent e) {
final JFileChooser chooser = new JFileChooser() {
private static final long serialVersionUID = 1L;

public void approveSelection() {
if (getSelectedFile().isFile()) {
return;
} else
super.approveSelection();
}
};
chooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
int returnVal = chooser.showOpenDialog(getParent());
if (chooser.getSelectedFile() == null){
return;
}
String absolute = chooser.getSelectedFile().getAbsolutePath();
if(returnVal == JFileChooser.APPROVE_OPTION) {
WarningLabel.setText("");
SnapshotFolderTextField.setForeground(black);
SnapshotFolderTextField.setText(absolute);
}
}
});
GridBagConstraints gbc_btnSetSnapshotFolder = new GridBagConstraints();
gbc_btnSetSnapshotFolder.fill = GridBagConstraints.HORIZONTAL;
gbc_btnSetSnapshotFolder.insets = new Insets(0, 0, 5, 5);
gbc_btnSetSnapshotFolder.gridx = 1;
gbc_btnSetSnapshotFolder.gridy = 6;
contentPane.add(btnSetSnapshotFolder, gbc_btnSetSnapshotFolder);

/***
*
*    ||S |||a |||v |||e |||       |||a |||n |||d |||       |||C |||l |||o |||s |||e ||
*    ||__|||__|||__|||__|||_______|||__|||__|||__|||_______|||__|||__|||__|||__|||__||
*    |/__\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/_______\|/__\|/__\|/__\|/__\|/__\|
*/

btnSaveAndClose = new JButton("Save and Close");
btnSaveAndClose.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
String rcvdf = RecievedFilesTextField.getText();
String snapsf = SnapshotFolderTextField.getText();
if (rcvdf.equals("") || rcvdf == null || rcvdf.isEmpty()){
WarningLabel.setText("Please check the text marked in red.");
RecievedFilesTextField.setForeground(red);
RecievedFilesTextField.setText("Please select a folder.");
}
if (SnapshotCheckBox.isSelected() == true){
if (snapsf.equals("") || snapsf == null || snapsf.isEmpty()){
WarningLabel.setText("Please check the text marked in red.");
SnapshotFolderTextField.setForeground(red);
SnapshotFolderTextField.setText("Please select a folder.");
}
}
if ((!((WarningLabel.getText().equals("")) || (WarningLabel.getText() == null) ||
(WarningLabel.getText().isEmpty())))
|| (SnapshotFolderTextField.getText().equals("Please select a folder."))
|| ( RecievedFilesTextField.getText().equals("Please select a folder."))){
return;
}
Client.settings.setRcvdFolder(rcvdf);
Client.settings.setSnapping(SnapshotCheckBox.isSelected());
if (Client.settings.isSnappingEnabled()){
Client.settings.setSnapshotFolder(snapsf);
Client.settings.setSeconds(slider.getValue());
Client.settings.setpercentage((double)slider_1.getValue());
}else{
Client.settings.setSnapshotFolder("");
Client.settings.setSeconds(0);
Client.settings.setpercentage((double)0);
}
try {
Client.settings.makeFile(Client.settings);
JOptionPane.showMessageDialog(null,"Settings saved successfully!");
if (Client.ClientConnected == true){
DataOutputStream outToServer = null;
```

```
try {
outToServer = new DataOutputStream(Client.clientSocket.getOutputStream());
} catch (IOException e3) {
e3.printStackTrace();
}
try {
if (Client.settings.isSnappingEnabled() == true){
outToServer.writeBytes("~se\r\n");
}else{
outToServer.writeBytes("~sd\r\n");
}
} catch (IOException e4) {
e4.printStackTrace();
}
try {
outToServer.flush();
} catch (IOException e5) {
e5.printStackTrace();
}
try {
if (Client.settings.isSnappingEnabled() == true){
outToServer.writeBytes("~" + Client.settings.getSeconds() + "\r\n");
}
} catch (IOException e6) {
e6.printStackTrace();
}
try {
if (Client.settings.isSnappingEnabled() == true){

outToServer.flush();
}
} catch (IOException e7) {
e7.printStackTrace();
}


try {
if (Client.settings.isSnappingEnabled() == true){
outToServer.writeBytes("~w" + Client.settings.getpercentage() + "\r\n");
}
} catch (IOException e6) {
JOptionPane.showMessageDialog(null,"FAIL from client: sending perc: " +
Client.settings.getpercentage());
e6.printStackTrace();
}
try {
if (Client.settings.isSnappingEnabled() == true){
outToServer.flush();
}
} catch (IOException e7) {
e7.printStackTrace();
}
}
setVisible(false);
dispose();
} catch (IOException e1) {
e1.printStackTrace();
}

}
});
GridBagConstraints gbc_btnSaveAndClose = new GridBagConstraints();
gbc_btnSaveAndClose.insets = new Insets(0, 0, 5, 5);
gbc_btnSaveAndClose.fill = GridBagConstraints.HORIZONTAL;
gbc_btnSaveAndClose.gridwidth = 3;
gbc_btnSaveAndClose.gridx = 1;
gbc_btnSaveAndClose.gridy = 13;
contentPane.add(btnSaveAndClose, gbc_btnSaveAndClose);

/**
*
*    888888888888                          88888888888  88            88          88
*         88                          ,d     88           ""            88          88
*         88                          88     88                         88          88
*         88   ,adPPYba,  8b,     ,d8  MM88MMM  88aaaaa      88   ,adPPYba,  88   ,adPPYb,88  ,adPPYba,  ,adPPYba,
*         88  a8P_____88   `Y8, ,8P'     88     88"""""      88  a8P_____88  88  a8"     `Y88  I8[    ""  I8[    ""
*         88  8PP"""""""     )888(       88     88           88  8PP"""""""  88  8b       88   `"Y8ba,    `"Y8ba,
*         88  "8b,   ,aa   ,d8" "8b,     88,    88           88  "8b,   ,aa  88  "8a,   ,d88  aa    ]8I  aa    ]8I
*         88   `"Ybbd8"'  8P'     `Y8    "Y888  88           88   `"Ybbd8"'  88   `"8bbdP"Y8   `"YbbdP"'  `"YbbdP"'
*
*
```

```
*/

/***
*      +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+ +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*      |R|e|c|e|i|v|e|d| |F|i|l|e|s| |F|o|l|d|e|r| |T|e|x|t|F|i|e|l|d|
*      +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+ +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*/

RecievedFilesTextField = new JTextField();

RecievedFilesTextField.setText(Client.settings.getRcvdFolder());
RecievedFilesTextField.setEditable(false);

GridBagConstraints gbc_RecievedFilesTextField = new GridBagConstraints();
gbc_RecievedFilesTextField.gridwidth = 2;
gbc_RecievedFilesTextField.insets = new Insets(0, 0, 5, 5);
gbc_RecievedFilesTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_RecievedFilesTextField.gridx = 2;
gbc_RecievedFilesTextField.gridy = 5;
contentPane.add(RecievedFilesTextField, gbc_RecievedFilesTextField);
RecievedFilesTextField.setColumns(10);

/***
* Create a JTextField for Server Snapshot Folder.
*      +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
*      |S|n|a|p|s|h|o|t| |F|o|l|d|e|r| |T|e|x|t|F|i|e|l|d|
*      +-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
* A JTextField is a basic text control that enables the user to type a small amount of
text.
*/

SnapshotFolderTextField = new JTextField();
SnapshotFolderTextField.setText(Client.settings.getSnapshotFolder());
SnapshotFolderTextField.setEditable(false);
SnapshotFolderTextField.setColumns(10);
GridBagConstraints gbc SnapshotFolderTextField = new GridBagConstraints();
gbc_SnapshotFolderTextField.gridwidth = 2;
gbc_SnapshotFolderTextField.insets = new Insets(0, 0, 5, 5);
gbc_SnapshotFolderTextField.fill = GridBagConstraints.HORIZONTAL;
gbc_SnapshotFolderTextField.gridx = 2;
gbc_SnapshotFolderTextField.gridy = 6;
contentPane.add(SnapshotFolderTextField, gbc_SnapshotFolderTextField);

/***
*
*     ,ad8888ba,   88                              88        88888888ba
*    d8"'    `"8b  88                              88        88      "8b
*   d8'            88                              88        88      ,8P
*   88             88,dPPYba,    ,adPPYba,   ,adPPYba,  88   ,d8   88aaaaaa8P'   ,adPPYba,  8b,     ,d8  ,adPPYba,  ,adPPYba,
*   88             88P'    "8a  a8P     88  a8"     ""  88 ,a8"    88""""""8b,  a8"     "8a  `Y8, ,8P'  a8P     88  I8[    ""
*   Y8,            88      88  8PP"""""""  8b          8888[      88      `8b  8b       d8    )888(    8PP"""""""   `"Y8ba,
*    `Y8a.   .a8P  88      88  "8b,   ,aa  "8a,   ,aa  88`"Yba,   88      a8P  "8a,   ,a8"  ,d8" "8b,  "8b,   ,aa  aa    ]8I
*     `"Y8888Y"'   88      88   `"Ybbd8"'   `"Ybbd8"'  88   `Y8a  88888888P"    `"YbbdP"'  8P'     `Y8  `"Ybbd8"'  `"YbbdP"'
*
*
*/

/***
*      +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
*      |S|n|a|p|s|h|o|t|s| |C|h|e|c|k|b|o|x|
*      +-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
*/

SnapshotCheckBox = new JCheckBox("");
SnapshotCheckBox.setSelected(Client.settings.isSnappingEnabled());
SnapshotCheckBox.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
if (SnapshotCheckBox.isSelected() == true){
btnSetSnapshotFolder.setEnabled(true);
slider.setEnabled(true);
slider.setValue(60);
String temp = getTIME(slider.getValue());
lblSeconds.setText(temp);
SnapshotFolderTextField.setText(Client.settings.getSnapshotFolder());
slider 1.setEnabled(true);
lblperc.setText((double)slider_1.getValue()+"%");
}else{
btnSetSnapshotFolder.setEnabled(false);
SnapshotFolderTextField.setText("");
lblSeconds.setText("");
slider.setEnabled(false);
```

```
slider_1.setEnabled(false);
lblperc.setText("");
}
}
});
GridBagConstraints gbc_SnapshotCheckBox = new GridBagConstraints();
gbc_SnapshotCheckBox.gridwidth = 2;
gbc_SnapshotCheckBox.insets = new Insets(0, 0, 5, 5);
gbc_SnapshotCheckBox.gridx = 2;
gbc_SnapshotCheckBox.gridy = 8;
contentPane.add(SnapshotCheckBox, gbc_SnapshotCheckBox);

if (SnapshotCheckBox.isSelected() == true){
btnSetSnapshotFolder.setEnabled(true);
slider.setEnabled(true);
slider.setValue(60);
String temp2 = getTIME(slider.getValue());
lblSeconds.setText(temp2);
SnapshotFolderTextField.setText(Client.settings.getSnapshotFolder());
slider_1.setEnabled(true);
lblperc.setText((double)slider_1.getValue()+"%");
}else{
btnSetSnapshotFolder.setEnabled(false);
SnapshotFolderTextField.setText("");
lblSeconds.setText("");
slider.setEnabled(false);
slider_1.setEnabled(false);
lblperc.setText("");
}

}

/**
* Convert seconds (integer format) to time (String format)
* @param time_wanted The time in seconds (integer).
* @return String format of the seconds.
*/

public String getTIME(int time_wanted) {
String output_text = "";
long hours = TimeUnit.SECONDS.toHours(time_wanted);
long minute = TimeUnit.SECONDS.toMinutes(time_wanted) -
(TimeUnit.SECONDS.toHours(time_wanted)* 60);
long second = TimeUnit.SECONDS.toSeconds(time_wanted) -
(TimeUnit.SECONDS.toMinutes(time_wanted) *60);
if (hours != 0){
output_text += hours + " hour";
if (hours != 1){
output_text += "s, ";
}else{
if (second != 0 && minute != 0){
output_text += ", ";
}
}
}
if (minute != 0){
output_text += minute + " minute";
if (minute != 1){
output_text += "s, ";
}else{
if (second != 0){
output_text += ", ";
}
}
}
if (second != 0){
output text += second + " second";
if (second != 1){
output_text += "s";
}
}
return output_text;
}
}
```

ΔΕΝΔΡΟΔΙΑΓΡΑΜΜΑ ΕΦΑΡΜΟΓΗΣ ANDROID



Κώδικας Εφαρμογής Android

Client.java

```
package ftp.SocketSnapshot_Client_Android;

import android.content.Intent;
import android.content.res.Configuration;
import android.graphics.Color;
import android.os.AsyncTask;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import java.io.IOException;
import java.net.*;

/**
* SocketSnapshot's Client's main class
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/


public class Client extends ActionBarActivity {
public static String IP = "";
public static int port = 0;
private TextView StatusLabel;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.setContentView(R.layout.activity_client);
}

/**
* The Action Listener for the Join Button.
* @param view The view to be passed.
*/

public void JoinButtonListener(View view) {
/* Get textfields and labels by ids */
EditText text1 = (EditText) findViewById(R.id.ServerTextField);
String ServerTextFieldValue = text1.getText().toString();
EditText text2 = (EditText) findViewById(R.id.ServerPortField);
String ServerPortFieldValue = text2.getText().toString();
StatusLabel = (TextView) findViewById(R.id.statusLabel);
/* If empty return */
if (ServerTextFieldValue == null || ServerTextFieldValue.equals("") ||
ServerTextFieldValue.isEmpty()) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please enter an IPv4 address.");
return;
}
if (ServerPortFieldValue == null || ServerPortFieldValue.equals("") ||
ServerPortFieldValue.isEmpty()) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please enter a port.");
return;
}
int PORT = 0;
/* Check if given port is a valid port */
try {
PORT = Integer.parseInt(ServerPortFieldValue);
```

```
if (!((PORT > 1025) && (port < 65536))) {
PORT = 0;
}
} catch (NumberFormatException e) {
PORT = 0;
}
if (PORT == 0) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please set a valid port.");
return;
}
/* Check if IPv4 is valid */
try {
if (ServerTextFieldValue == null || ServerTextFieldValue.isEmpty()) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please set a valid IPv4 address.");
return;
}

String[] parts = ServerTextFieldValue.split("\\.");
if (parts.length != 4) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please set a valid IPv4 address.");
return;
}

for (String s : parts) {
int i = Integer.parseInt(s);
if ((i < 0) || (i > 255)) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please set a valid IPv4 address.");
return;
}
}
if (ServerTextFieldValue.endsWith(".")) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please set a valid IPv4 address.");
return;
}
} catch (NumberFormatException nfe) {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Please set a valid IPv4.");
return;
}
IP = ServerTextFieldValue;
port = PORT;
initiateSingleConnection();
}

/**
* Start a new connection task.
*/

public void initiateSingleConnection() {
connection connectiontask = new connection();
connectiontask.execute();
}

/**
* Change activity!
*/

public void changeActivity(){
StatusLabel.setTextColor(Color.GREEN);
StatusLabel.setText("Connected to " + IP + ":" + port + ".");
Intent intent = new Intent(Client.this, ClientJoined.class);
intent.putExtra("myip", IP);
intent.putExtra("myport", Integer.toString(port));
startActivity(intent);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_client, menu);
return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId();

//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
Intent i = new Intent(this, PreferencesActivityScenario1.class);
startActivity(i);
return true;
}
return false;
}

/**
* Override onConfigurationChanged so that configuration won't be changed (to landscape
et.c.).
* @param newConfig The Configuration.
*/

public void onConfigurationChanged(Configuration newConfig){
// ignore orientation change
if (newConfig.orientation != Configuration.ORIENTATION_LANDSCAPE){
super.onConfigurationChanged(newConfig);
}
}

/**
* Nested class.
* This class extends AsyncTask and attempts to connect to a socket with
* the given IP and Port.
*
* This multithreading task is required otherwise it's impossible to connect.
*/

private class connection extends AsyncTask<Void, Void, Void> {
protected Void doInBackground(Void... params) {
Socket clientSocket = null;
try {
clientSocket = new Socket(IP, port);
} catch (UnknownHostException ex) {
runOnUiThread(new Runnable() {
@Override
public void run() {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Unable to connect to " + IP + ":" + port + ".");
}
});
try {
if (clientSocket != null) {
clientSocket.close();
clientSocket = null;
}
} catch (IOException e1) {}
ex.printStackTrace();
return null;
} catch (ConnectException ex2){
runOnUiThread(new Runnable() {
@Override
public void run() {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Unable to connect to " + IP + ":" + port + ".");
}
});
try {
if (clientSocket != null) {
clientSocket.close();
clientSocket = null;
}
} catch (IOException e1) {}
ex2.printStackTrace();
return null;
} catch (IOException ex) {
runOnUiThread(new Runnable() {
```

```
@Override
public void run() {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Unable to connect to " + IP + ":" + port + ".");
}
});
try {
if (clientSocket != null) {
clientSocket.close();
clientSocket = null;
}
} catch (IOException e1) {}
ex.printStackTrace();
return null;
}
if (clientSocket != null) {
SocketHandler.setSocket(clientSocket);
runOnUiThread(new Runnable() {
@Override
public void run() {
changeActivity();
}
});
}else{
runOnUiThread(new Runnable() {
@Override
public void run() {
StatusLabel.setTextColor(Color.RED);
StatusLabel.setText("Unable to connect to " + IP + ":" + port + ".");
}
});
}
return null;
}
}
}
```

ClientJoined

```
package ftp.SocketSnapshot_Client_Android;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.TaskStackBuilder;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

/**
* SocketSnapshot's Client Joined class.
*
* Notice: runOnUiThread is required for every single
* change on the UI. Otherwise changes won't appear to the user.
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

public class ClientJoined extends ActionBarActivity implements View.OnClickListener {
public static boolean connected;
static {
connected = false;
}
public String ip;
public static int port;
public MessageBoxFullText messages;
public EditText MessageBoxfield;
public EditText SendMessagefield;
public Button SendMessageButton;
public OutputHandler outputhandler;

/**
* The constructor (initializes only a new message box {multi-line textfield}.
*/

public ClientJoined() {
messages = new MessageBoxFullText();
}

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.setContentView(R.layout.activity_client_joined);
MessageBoxfield = (EditText) findViewById(R.id.MessageBoxField);
SendMessagefield = (EditText) findViewById(R.id.SendMessageField);
SendMessageButton = (Button) findViewById(R.id.send_message_button);
SendMessagefield.setOnClickListener((View.OnClickListener) this);
```

```
/* Get data from previous Activity (Client.class) */

Intent intent = getIntent();
ip = intent.getStringExtra("myip");
String myport = intent.getStringExtra("myport");
port = Integer.parseInt(myport);
messages.appendString("Connected to: " + ip + ":" + port + "\r\n");
runOnUiThread(new Runnable() {
@Override
public void run() {
MessageBoxfield.setText(messages.getString());
}
});

/* Start new thread for handling input. */

new Thread(new InputHandler(SocketHandler.getSocket(), messages, MessageBoxfield,
ClientJoined.this, getApplicationContext())).start();

/* Send preferences (only once). */

try {
SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(ClientJoined.this);
Boolean isSnappingEnabled = sp.getBoolean("snapshotsenabled", true);
int SnappingSeconds = 0;
double SnappingPercetange = (double) 0;
if (isSnappingEnabled == true) {
String SnappingSecondsString = sp.getString("snapshotseconds", "60");
String SnappingPercetangeString = sp.getString("snapshotpercetange", "10");
SnappingSeconds = Integer.parseInt(SnappingSecondsString);
SnappingPercetange = (double) Integer.parseInt(SnappingPercetangeString);
}

DataOutputStream outToServer = new
DataOutputStream(SocketHandler.getSocket().getOutputStream());
if (isSnappingEnabled == true) {
outToServer.writeBytes("~se\r\n");
} else {
outToServer.writeBytes("~sd\r\n");
}
outToServer.flush();
if (isSnappingEnabled == true) {
outToServer.writeBytes("~" + ((SnappingSeconds > 2000) ? 2000 : SnappingSeconds) +
"\r\n");
outToServer.flush();
}
if (isSnappingEnabled == true) {
outToServer.writeBytes("~w" + ((SnappingPercetange > 100) ? 100 : SnappingPercetange)
+ "\r\n");
outToServer.flush();
}
} catch (IOException e3) {
e3.printStackTrace();
}
}

/**
* Send a message.
* @param view The view.
*/

public void SendMessageListener(View view) {
this.outputhandler = new OutputHandler(SocketHandler.getSocket(), messages,
MessageBoxfield, SendMessagefield, ClientJoined.this);
outputhandler.SetMessage(SendMessagefield.getText().toString());
Thread c = new Thread(outputhandler);
c.start();
try {
Thread.sleep(100);
c.interrupt();
} catch (InterruptedException e) {
e.printStackTrace();
}

}
```

```
/**
* If the input (message textfield) is clicked (touched) then erase
* the pre-defined text.
* @param view The view.
*/

public void onClick(View view) {
if (view.getId() == R.id.SendMessageField) {
runOnUiThread(new Runnable() {
@Override
public void run() {
SendMessagefield.setText("");
}
});
}
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_client_joined, menu);
return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId();

//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
return true;
}

return super.onOptionsItemSelected(item);
}

/**
* Action Listener to the Abort Button.
* @param view The View.
*/

public void AbortListener(View view) {
new AlertDialog.Builder(this)
.setTitle("Are you sure?")
.setMessage("Abort connection?")
.setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
try {
if (SocketHandler.getSocket() != null) {
SocketHandler.getSocket().close();
SocketHandler.setSocket(null);
}
} catch (IOException e1) {

} finally {
SocketHandler.setSocket(null);
}
Intent intent = new Intent(ClientJoined.this, Client.class);
startActivity(intent);
finish();
}
})
.setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
return;
}
})
.setIcon(android.R.drawable.ic_dialog_alert)
.show();
}

/**
* Action Listener to the Back Button.
```

```
* @param view The View.
*/

public void onBackPressed() {
new AlertDialog.Builder(this)
.setTitle("Are you sure?")
.setMessage("Abort connection?")
.setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
try {
if (SocketHandler.getSocket() != null) {
SocketHandler.getSocket().close();
SocketHandler.setSocket(null);
}
} catch (IOException e1) {

} finally {
SocketHandler.setSocket(null);
}
Intent intent = new Intent(ClientJoined.this, Client.class);
startActivity(intent);
finish();
}
})
.setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
return;
}
})
.setIcon(android.R.drawable.ic_dialog_alert)
.show();
}

/**
* Override onConfigurationChanged so that configuration won't be changed (to landscape
et.c.).
* @param newConfig The Configuration.
*/

public void onConfigurationChanged(Configuration newConfig){
// ignore orientation change
if (newConfig.orientation != Configuration.ORIENTATION_LANDSCAPE){
super.onConfigurationChanged(newConfig);
}
}
}

/**
* Nested class.
* This class is required in order to update MessageBox (aka multi-line textfield).
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

class MessageBoxFullText {
private String message;

public MessageBoxFullText() {
this.message = "";
}

public void appendString(String str) {
this.message += str;
}

public String getString() {
return message;
}
}

/**
* The Input Handler.
*
* Please see the Java (Desktop) application for more details.
*
* The implementation is almost identical.
```

```
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

class InputHandler implements Runnable {
private Socket socket;
private MessageBoxFullText msg;
private EditText msgbox;
private Activity myactivity;
private Context context;

public InputHandler(Socket socket,MessageBoxFullText msg,EditText msgbox,Activity
myactivity,Context context) {
this.socket = socket;
this.msg = msg;
this.msgbox = msgbox;
this.myactivity = myactivity;
this.context = context;
}

public void run() {
ClientJoined.connected = true;
BufferedReader reader = null;
try {
reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
while (ClientJoined.connected) {
String text = "";
if (ClientJoined.connected == true) {
text = reader.readLine();
}
if ((text != null) && !(text.equals("")) && !(text.isEmpty())) {
if (text.charAt(0) == '#') {
String filename = text.substring(1, text.length());
msg.appendString("Attempting to recieve " + filename + "\r\n");
myactivity.runOnUiThread(new Runnable() {
@Override
public void run() {
msgbox.setText(msg.getString());
}
});

new Thread(new ClientRecieveHandle(ClientJoined.port + 1,
filename,msg,msgbox,myactivity,context)).start();
continue;
} else if (text.charAt(0) == '|') {
continue;
} else if (text.charAt(0) == '&') {
Date date = new Date();
msg.appendString("A dangerous " + text.substring(1) + "% was noticed at " +
date.toString() + "\r\n");
NotificationCompat.Builder mBuilder =
new NotificationCompat.Builder(context)
.setSmallIcon(R.drawable.ic_stat_notification_sms_failed)
.setContentTitle("Warning - SocketSnapshot")
.setContentText("A dangerous " + text.substring(1) + "% was noticed at " +
date.toString());
Intent resultIntent = new Intent(context, ClientJoined.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(context);
stackBuilder.addParentStack(ClientJoined.class);
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(
0,
PendingIntent.FLAG_UPDATE_CURRENT
);
mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
(NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(123, mBuilder.build());
myactivity.runOnUiThread(new Runnable() {
@Override
public void run() {
msgbox.setText(msg.getString());
}
});
```

```
continue;
}
}
if ((text == null || text.equals("") || text.isEmpty())) {
continue;
}
msg.appendString("#Server said: " + text + "\r\n");
myactivity.runOnUiThread(new Runnable() {
@Override
public void run() {
msgbox.setText(msg.getString());
}
});
}
reader.close();
} catch (Exception e) {
return;
}
}
}

/**
* The Output Hanlder.
*
* Please see the Java (Desktop) application for more details.
*
* The implementation is almost identical.
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/


class OutputHandler implements Runnable {
private Socket socket;
private MessageBoxFullText msg;
private EditText msgbox;
private String message;
private EditText sendmsgfield;
private Activity myactivity;
private Context context;

public OutputHandler(Socket socket,MessageBoxFullText msg,EditText msgbox,EditText
sendmsgfield,Activity myactivity) {
this.socket = socket;
this.message = "";
this.msg = msg;
this.msgbox = msgbox;
this.sendmsgfield = sendmsgfield;
this.myactivity = myactivity;
this.context = context;
}

public void SetMessage(String message) {
this.message = message;
}

public String GetMessage() {
return message;
}

public void run() {
DataOutputStream outToServer = null;
try {
msg.appendString("#You said: " + GetMessage() + "\r\n");
myactivity.runOnUiThread(new Runnable() {
@Override
public void run() {
msgbox.setText(msg.getString());
}
});
outToServer = new DataOutputStream(SocketHandler.getSocket().getOutputStream());
outToServer.writeBytes(GetMessage() + "\r\n");
outToServer.flush();
myactivity.runOnUiThread(new Runnable() {
@Override
```

```
public void run() {
sendmsgfield.setText("");
}
});
} catch (IOException e) {
e.printStackTrace();
}finally{
outToServer = null;
return;
}
}
}

/**
* The Client Receive Hanlder.
*
* Please see the Java (Desktop) application for more details.
*
* The implementation is almost identical.
*
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/

class ClientRecieveHandle implements Runnable {
private int port2;
private String filename;
private MessageBoxFullText msg;
private EditText msgbox;
private Activity myactivity;
private Context context;

public ClientRecieveHandle(int port2,String filename,MessageBoxFullText msg,EditText
msgbox,Activity myactivity,Context context) throws IOException {
this.port2 = port2;
this.filename = filename;
this.msg = msg;
this.msgbox = msgbox;
this.myactivity = myactivity;
this.context = context;
}

public void run() {
ServerSocket RecieveSocket = null;
try {
RecieveSocket = new ServerSocket(port2);
Socket connectionSocket = RecieveSocket.accept();
InputStream is = connectionSocket.getInputStream();
int filesize = 6022386;
int bytesRead = 0;
int current = 0;
byte[] mybytearray = new byte[filesize];
String baseFolder;
// check if external storage is available
if (Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
baseFolder =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS).toStrin
g();
// revert to using internal storage (not sure if there's an equivalent to the above)
} else {
baseFolder = context.getFilesDir().getAbsolutePath();
}
File file = new File(baseFolder + File.separator + filename);
FileOutputStream fos = null;
fos = new FileOutputStream(file);
BufferedOutputStream bos = new BufferedOutputStream(fos);
bytesRead = is.read(mybytearray, 0, mybytearray.length);
current = bytesRead;
do {
bytesRead = is.read(mybytearray, current, (mybytearray.length - current));
if (bytesRead >= 0) {
current += bytesRead;
}
} while (bytesRead > -1);
bos.write(mybytearray, 0, current);
msg.appendString("Successfully recieved " + filename + "\r\n");
```

```
myactivity.runOnUiThread(new Runnable() {
@Override
public void run() {
msgbox.setText(msg.getString());
}
});
bos.flush();
bos.close();
connectionSocket.close();
RecieveSocket.close();
return;
} catch (IOException e) {
e.printStackTrace();
}
}
}
```

PreferencesActivityScenario1.java

```
package ftp.SocketSnapshot_Client_Android;

import android.preference.PreferenceActivity;
import android.os.Bundle;


public class PreferencesActivityScenario1 extends PreferenceActivity {

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
addPreferencesFromResource(R.xml.preferences_scenario1);
}
}
```

SocketHandler.java

```
package ftp.SocketSnapshot_Client_Android;

import java.net.Socket;

/**
* @author  Paulina Barniadaki
* @version 2.0
* @since   2015-06-01
*/
public class SocketHandler {
private static Socket socket;

public static synchronized Socket getSocket(){
return socket;
}

public static synchronized void setSocket(Socket socket){
SocketHandler.socket = socket;
}
}
```

Γραφικά Android

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="ftp.SocketSnapshot_Client_Android" >

<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"
android:maxSdkVersion="18">
</uses-permission>

<uses-permission
android:name="android.permission.INTERNET">
</uses-permission>


<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name="ftp.SocketSnapshot_Client_Android.Client"
android:label="@string/app_name"
android:screenOrientation="portrait"
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenS
ize|smallestScreenSize"
android:windowSoftInputMode="adjustPan"
android:alwaysRetainTaskState="true">
<intent-filter android:label="@string/app_name">
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
android:name="ftp.SocketSnapshot_Client_Android.ClientJoined"
android:label="@string/title_activity_joined_client"
android:screenOrientation="portrait"
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenS
ize|smallestScreenSize"
android:windowSoftInputMode="adjustPan"
android:alwaysRetainTaskState="true">
</activity>
<activity
android:name="ftp.SocketSnapshot_Client_Android.PreferencesActivityScenario1"></activi
ty>
</application>

</manifest>
```

Activity_Client.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".Client"
android:id="@+id/Client"><![CDATA[


android:orientation="horizontal"
android:layout_width="match_parent"
```

```
android:layout_height="match_parent">


]]>

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:text="                    Server IPv4"
android:id="@+id/ServerIPLabel"
android:layout_row="4"
android:layout_column="0"
android:layout_gravity="center"
android:layout_rowSpan="3"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true" />

<EditText
android:layout_width="337dp"
android:layout_height="64dp"
android:id="@+id/ServerTextField"
android:layout_row="16"
android:layout_column="0"
android:layout_gravity="fill_horizontal|top"
android:layout_below="@+id/ServerIPLabel"
android:layout_alignRight="@+id/ServerIPLabel"
android:layout_alignEnd="@+id/ServerIPLabel"
android:editable="true"
android:enabled="true"
android:inputType="text" />

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:text="                    Server Port"
android:id="@+id/ServerPortLabel"
android:layout_gravity="center"
android:layout_below="@+id/ServerTextField"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_alignRight="@+id/ServerTextField"
android:layout_alignEnd="@+id/ServerTextField" />

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Join"
android:id="@+id/button"
android:onClick="JoinButtonListener"
android:layout_centerVertical="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true" />

<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/ServerPortField"
android:layout_below="@+id/ServerPortLabel"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_alignRight="@+id/ServerPortLabel"
android:layout_alignEnd="@+id/ServerPortLabel" />

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:id="@+id/statusLabel"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true" />

</RelativeLayout>
```

activity_client_joined.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context="ftp.SocketSnapshot_Client_Android.ClientJoined"
android:id="@+id/ClientJoined">

<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:inputType="textMultiLine"
android:ems="10"
android:id="@+id/MessageBoxField"
android:layout_below="@+id/textView3"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="false"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:layout_above="@+id/SendMessageField"
android:enabled="true"
android:clickable="false"
android:cursorVisible="false"
android:focusable="false"
android:focusableInTouchMode="false"/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:text="                    Message Box"
android:id="@+id/textView3"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_alignParentTop="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true" />

<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/SendMessageField"
android:layout_alignParentBottom="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="66dp"
android:layout_alignRight="@+id/MessageBoxField"
android:layout_alignEnd="@+id/MessageBoxField"
android:editable="true"
android:enabled="true"
android:text="Enter a message.." />

<Button
style="?android:attr/buttonStyleSmall"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Send Message"
android:onClick="SendMessageListener"
android:id="@+id/send_message_button"
android:layout_alignParentBottom="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:enabled="true" />

<Button
style="?android:attr/buttonStyleSmall"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Abort"
android:id="@+id/abortbutton"
```

```
android:onClick="AbortListener"
android:layout_alignParentBottom="true"
android:layout_alignRight="@+id/SendMessageField"
android:layout_alignEnd="@+id/SendMessageField"
android:enabled="true" />
</RelativeLayout>
```

activity_preference.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context="ftp.SocketSnapshot_Client_Android.PreferenceActivity">

<TextView android:text="@string/hello_world" android:layout_width="wrap_content"
android:layout_height="wrap_content" />

</RelativeLayout>
```

Αξιολόγηση - Περιορισμοί

Εφαρμογή Server υπολογιστή

- Δεν είναι δυνατή η αποστολή λίστας αρχείων χωρίς την προηγούμενη αίτηση από τον Client.
- Δεν είναι δυνατή η αίτηση λίστας αρχείων από των Client.
- Δεν είναι δυνατή η λήψη οιασδήποτε αρχείου από τον Client.
- Δεν είναι δυνατή η απομακρυσμένη αλλαγή των ρυθμίσεων του Client από τον Server (remote settings).
- Δεν είναι δυνατή η αλλαγή ρυθμίσεων σχετικών με τη λήψη φωτογραφιών από Webcam (παρά μόνο χειροκίνητα με επεξεργασία του .cfg αρχείου).

  Να σημειωθεί ότι οιασδήποτε αλλαγή αυτών των ρυθμίσεων, θα απανωγραφεί από τις απομακρυσμένες ρυθμίσεις του Client, κατά τη σύνδεση του (ανεξάρτητα εάν συμπίπτουν).
- Δεν είναι δυνατή η αποστολή της ίδιας της φωτογραφίας (webcam). Εναλλακτικά, αποστέλλεται ειδοποίηση στον Client σε περίπτωση παρατήρησης της ορισμένης παρέκκλισης.
- Ενδέχεται σε ορισμένες περιπτώσεις να μην είναι άμεσα «αισθητή» η (πρόωρη και μη) αποχώρηση του Client.

  Αυτό οφείλεται στη φύση του προγραμματισμού με sockets.

Εφαρμογή Client υπολογιστή

- Δεν είναι δυνατή η λήψη τα λίστας αρχείων του Server πέρα της λίστας αρχείων του περιεχομένου του τρέχοντος φακέλου που τρέχει ο Server.
- Δεν είναι δυνατή η αποστολή αρχείων από τον Client στο Server.
- Δεν είναι δυνατή η λήψη φωτογραφίας (webcam) από τον Client (και κατά συνέπεια η αποστολή στον Server).
- Ενδέχεται σε ορισμένες περιπτώσεις να μην είναι άμεσα «αισθητός» ο αυθαίρετος τερματισμός του Server.

  Αυτό οφείλεται στη φύση του προγραμματισμού με sockets.

Εφαρμογή Client κινητού ή tablet (Android)

- Όλα του «Εφαρμογή Client υπολογιστή»

- Δεν είναι εφικτή η λειτουργία στο παρασκήνιο (background).

  Δηλαδή δεν μπορεί ο χρήστης να χειρίζεται άλλη εφαρμογή (ή άλλες εφαρμογές) παράλληλα (στο προσκήνιο και μη).

  Κάτι τέτοιο απαιτεί από την εφαρμογή για να επιτευχθεί να λειτουργεί ως υπηρεσία (Service) δεν είναι εφικτό διότι δεν το επιτρέπει μία Activity – χρειάζεται υπηρεσία (Service).

  Κατά συνέπεια δεν παρέχεται καμμία εγγύηση για την ακεραιότητα της λειτουργίας της εφαρμογής εάν ο χρήστης προβεί σε μία τέτοια ενέργεια (εναλλαγή Activity).

- Δεν είναι δυνατή η εισαγωγή hostname αντί για IPv4.

- Δεν είναι δυνατή η αλλαγή προορισμού των αρχείων.
  Ως προεπιλογή ορίζονται ο φάκελος “Downloads”.

- Δεν υπάρχει καταγραφή συμβάντων (αρχείο καταγραφής).

Δεν είναι δυνατή η αίτηση λίστας αρχείων (και κατά συνέπεια και παραλαβή κάποιου συγκεκριμένου αρχείου)
Αυτό βέβαια δεν επηρεάζει τη μονομερή αποστολή αρχείων που επιτυγχάνεται από τον Server.

ΠΑΡΑΡΤΗΜΑ 1

Εγκατάσταση APK : Για να μπορέσετε να το εγκαταστήσετε στο τηλέφωνό σας, θα πρέπει να βεβαιωθείτε ότι οι εφαρμογές τρίτων επιτρέπεται στη συσκευή σας. Πηγαίνετε στο Μενού> Ρυθμίσεις> Ασφάλεια> και τσεκάρετε το "Unknown Πηγές" για να επιτρέψει το τηλέφωνό σας για να εγκαταστήσετε εφαρμογές από άλλες πηγές εκτός του Google Play Store.

Αν θέλετε, μπορείτε επίσης να κατεβάσετε μια εφαρμογή όπως το ES File Explorer,ώστε να μπορείτε εύκολα να βρείτε τα αρχεία για το Android συσκευή σας.Συνδέστε τη συσκευή σας σε αυτό.Σε αυτό το σημείο, μπορεί να σας ζητηθεί εάν θέλετε απλά να φορτίσετε το τηλέφωνό σας ή να το συνδέσετε σε μία «συσκευή πολυμέσων."

Επιλέξτε "συσκευή πολυμέσων". Στη συνέχεια, απλά βρείτε το φάκελο του τηλεφώνου σας στον υπολογιστή σας (αυτό θα είναι το "Ο Υπολογιστής μου" ή "Υπολογιστής" για υπολογιστές με Windows) και αντιγράψτε το αρχείο APK σε ένα φάκελο της επιλογήςσας στο Android smartphone σας.

Τώρα θα είστε σε θέση να ψάξετε για τη θέση του αρχείου στο "αρχεία μου" φάκελο στη συσκευή σας.Βρείτε το αρχείο APK, χτυπήστε το, στη συνέχεια, πατήστε "εγκατάσταση".

Εγκατάσταση APK αρχεία από τον browser σας

Μπορείτε επίσης να εγκαταστήσετε το APK αρχεία από το πρόγραμμα περιήγησης στο Android smartphone ή tablet σας.Απλά ανοίξτε τον browser σας, βρείτε το APK αρχείο που θέλετε να κατεβάσετε και πατήστε το - θα πρέπει στη συνέχεια να είναι σε θέση να δείτε το κατέβασμα στην επάνω γραμμή της συσκευής σας. Μόλις κατεβάσει, ανοίξτε "downloads" σας, πατήστε στο .apk αρχείο, και πατήστε "ναι" όταν σας ζητηθεί.Η εφαρμογή θα ξεκινήσει η εγκατάσταση στη συσκευή σας.

Εντολές Προγράμματος

Αξίζει να σημειωθεί ότι κάποια μηνύματα που αποστέλλονται ανάμεσα στις δύο εφαρμογές (Server και Client) δεν είναι στην πραγματικότητα μηνύματα, αλλά εντολές.

Φυσικά, καλό θα ήταν να ΜΗΝ χρησιμοποιηθούν οι εντολές (τα μηνύματα) αυτά γιατί τότε δεν υπάρχει εγγύηση για την αλάθητη λειτουργία του προγράμματος.

Αυτές οι εντολές είναι:

Για τον Server:

Εντολές με πρόθεμα «~»:

~se = Ενεργοποίηση Snapshots

Απομακρυσμένη ρύθμιση του Server από τον Client.

~sd = Απενεργοποίηση Snapshots

Απομακρυσμένη ρύθμιση του Server από τον Client.

~w<double> = Ορισμός της επί της 100% διαφοράς μεταξύ των δύο τελευταίων snapshot για ειδοποίηση.

Απομακρυσμένη ρύθμιση του Server από τον Client.

~<integer> = Ορισμός των δευτερολέπτων που απαιτούνται για αναμονή μέχρι την αίτηση νέας λήψης snapshot.

Απομακρυσμένη ρύθμιση του Server από τον Client.

Εντολές με πρόθεμα «**!**»:

!<filename> = Αίτημα για λήψη αρχείου.

Αίτημα από τον Client στον Server.

Εντολή «**#**»:

= Αίτημα λίστας του Client για τον Server .

Για τον Client:

Εντολές με πρόθεμα «**#**»:

#(filename) = Λήψη αρχείου στον Client από τον Server.

Εντολές με πρόθεμα «**|**»:

|(filename) = Λήψη και αρχειοθέτηση ενός στοιχείου μίας λίστας αρχείων του Server.

Εντολές με πρόθεμα «**&**»:

&(double) = Λήψη της επί της 100% διαφοράς μεταξύ των δύο τελευταίων snapshot για ειδοποίηση.

Βιβλιογραφία

Κεφάλαιο πρώτο :

(1,1)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach, 6th edition, James F. Kurose, Keith W. Ross

http://el.wikipedia.org/wiki/%CE%95%CF%80%CE%AF%CF%80%CE%B5%CE%B4%CE%BF_%CE%B6%CE%B5%CF%8D%CE%BE%CE%B7%CF%82_%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD

http://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

http://webtrends.about.com/od/Mobile-Web-Beginner/tp/Wi-fi-Appliances.htm

http://el.wikipedia.org/wiki/%CE%95%CF%8D%CF%81%CE%BF%CF%82_%CE%B6%CF%8E%CE%BD%CE%B7%CF%82

http://slideplayer.gr/slide/2701358/

http://el.wikipedia.org/wiki/Transmission_Control_Protocol

http://el.wikipedia.org/wiki/%CE%94%CF%81%CE%BF%CE%BC%CE%BF%CE%BB%CE%BF%CE%B3%CE%B7%CF%84%CE%AE%CF%82

εικόνα 1η http://now24.gr/wp-content/uploads/2015/05/Internet.jpg

εικόνα 2η *http://www.sheetsdb.net/internet-submarine-cable-map/*

(1,2)

http://users.sch.gr/tsibinos/internet_history/history_1.html

http://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

εικόνα 3η http://thisdayintechhistory.com/wp-content/uploads/2013/11/ARPANET-map-SRI.jpg

εικόνα 4η http://users.sch.gr/tsibinos/internet_history/history_1.html

(1,3)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach, 6th edition, James F. Kurose, Keith W. Ross

http://el.wikipedia.org/wiki/Transmission_Control_Protocol

http://el.wikipedia.org/wiki/SMTP

http://el.wikipedia.org/wiki/Telnet

http://el.wikipedia.org/wiki/File_Transfer_Protocol

http://el.wikipedia.org/wiki/%CE%A0%CF%81%CF%89%CF%84%CF%8C%CE%BA%CE%BF%CE%BB%CE%BB%CE%BF_%CE%9C%CE%B5%CF%84%CE%B1%CF%86%CE%BF%CF%81%CE%AC%CF%82_%CE%A5%CF%80%CE%B5%CF%81%CE%BA%CE%B5%CE%B9%CE%BC%CE%AD%CE%BD%CE%BF%CF%85

(1,4)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach , 6th edition, James F. Kurose, Keith W. Ross

http://el.wikipedia.org/wiki/%CE%95%CF%80%CE%AF%CF%80%CE%B5%CE%B4%CE%BF_%CE%B6%CE%B5%CF%8D%CE%BE%CE%B7%CF%82_%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD

εικονα 5η http://www.studycampus.com/PgD/cnm/lesson5_files/link_layer.gif

εικονα 6η http://www.cisco1900router.com/wp-content/uploads/2013/03/1-tutorial-osi-7-layer-model.gif

eikona 7η http://www.slideshare.net/ashonphull/nic-card-presentation-by-asp

eikona 8η http://www.wpclipart.com/computer/hardware/networking/pcmcia_wireless.png.html

εικονα 9η http://s.hswstatic.com/gif/streaming-video-audio-7.gif

Εικονα 10η http://www.business-superstar.com/uploads/blog/internet_radio-100043602-large.jpg

εικόνα 11η http://www.studycampus.com/PgD/cnm/lesson6_files/streaming.gif

εικονα 12η http://www.hometheaterhifi.com/volume_1_1/images/jittertiming.gif

εικόνα 13η http://www.cisco.com/c/dam/en/us/support/docs/voice/voice-quality/18902-Fg2.gif

(1,5)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach , 6th edition, James F. Kurose, Keith W. Ross

(1,6)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach , 6th edition, James F. Kurose, Keith W. Ross

http://www.telecom.tuc.gr/courses/net2/res/ch4.pdf

1,6 δευτερη παράγραφος : http://www.islab.demokritos.gr/gr/html/ptixiakes/kostas-aris_ptyxiakh/Phtml/nat.htm

εικονα 14η http://docwiki.cisco.com/w/images/4/4f/Technology_Handbook-01-2-13.jpg

εικονα 15η http://phoenix.goucher.edu/~kelliher/s2011/cs325/apr06img4.png

εικονα 16η http://www.webclasses.net/courses/protocols/7.0/demobuild/units/media/figures/unit01/VirtualCirucit.gif

εικονα 17η http://netlab.ulusofona.pt/rc/book/4-network/4_01/04-02.jpg

(1,7)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach , 6th edition, James F. Kurose, Keith W. Ross

(1.8)

Δικτύωση Υπολογιστών, Δεύτερη έκδοση, James F. Kurose, Keith W. Ross, εκδόσεις Μ. Γκιούρδας

Computer Networking – A top-down approach , 6th edition, James F. Kurose, Keith W. Ross

εικόνα 20 *https://t2.ftcdn.net/jpg/00/45/16/47/400_F_45164792_4TONLaM9xt2sAeG1s66O39P2MHBaoR5i.jpg*

εικόνα 22 *http://websitelanguage.com/new/wp-content/uploads/2013/02/ftp.jpg*

εικόνες 21,23,24,25 από το βιβλίο Computer Networking – A top-down approach , 6th edition, James F. Kurose, Keith W. Ross

Κεφάλαιο δεύτερο :

http://users.softlab.ntua.gr/~bxb/courses/unipi2001_te/00-CourseNotes/031-OO&Java/TE031-2.pdf

http://aetos.it.teithe.gr/~sfetsos/java_book_EMP.pdf

http://www.it.uom.gr/project/java2001/k11.html

https://el.wikipedia.org/wiki/Java

εικ 26 http://blog.newrelic.com/wp-content/uploads/java-logo.jpg

εικόνα 27 http://1.bp.blogspot.com/_RsWE5WD6TtQ/SD5CmbyLbBI/AAAAAAAAADI/XRzCR2Arpic/s400/fragmentation-fig1.png

εικόνα 28 http://image.slidesharecdn.com/javanturav2-theroadtojava-hujakoraclecroatia-brankomihaljevialeksanderradovandukovukmanovi-141130060735-conversion-gate01/95/javantura-v2-the-road-to-java-hujak-oracle-croatia-branko-mihaljevi-aleksander-radovan-duko-vukmanovi-3-638.jpg?cb=1417328508

εικόνα 29 http://aetos.it.teithe.gr/~sfetsos/java_book_EMP.pdf

Κεφάλαιο Τρίτο :

http://www.myphone.gr/forum/showthread.php?t=306146#post3645016

http://www.androidcentral.com/devices

εικόνα 30 https://en.wikipedia.org/wiki/File:Android_robot.svg

εικόνα 31 http://crbtech.in/Android/images/android-mobile-cell-phone.jpg

εικόνα 32http://www.mobot.net/writer/wordpress/wp-content/uploads/2015/04/Android-Tablet-Shipments-Surging-Globally.jpg

εικόνα 33 https://www.tripletremelo.com/wp-content/uploads/2014/03/im-android-watch.jpg

εικόνα 34 http://www.theandroidsoul.com/wp-content/uploads/2015/04/android-auto.jpg

εικόνα 35 http://25labs.com/wp-content/uploads/2011/09/android-mobile-sales.jpg

εικόνα 36 http://www.codeproject.com/Articles/803452/Article-Very-Basic-Introduction-to-Android

εικόνα 37 http://www.knowahead.in/wp-content/uploads/2012/08/Android-Versions.jpg

Βιβλιογραφία Παρατήματος : http://www.yac.mx/el/mobile-security/android/what-is-an-apk-file-and-how-to-install-them.html