ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

" Σχεδιασμός και υλοποίηση εφαρμογής για την παροχή διαδραστικών υπηρεσιών πολυμέσων σε περιβάλλον επίγειας ψηφιακής τηλεόρασης"

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: **ΠΑΠΑΔΟΓΙΑΝΝΗΣ ΔΗΜΗΤΡΗΣ**
ΗΜΕΡΟΜΗΝΙΑ: **15/05/2007**

ΕΙΣΗΓΗΤΗΣ: **Δρ. ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ**

Στην οικογένεια μου
με αγάπη

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μου εργασίας, η οποία υλοποιήθηκε στο εργαστήριο Έρευνας και Ανάπτυξης Τηλεπικοινωνιακών Συστημάτων (ΠΑΣΙΦΑΗ) του Α.Τ.Ε.Ι Κρήτης, θα ήθελα να ευχαριστήσω όλους τους ανθρώπους οι οποίοι ο καθένας με τον τρόπο του βοήθησαν στην ολοκλήρωση αυτής της πτυχιακής εργασίας.

Κατά κύριο λόγο, οφείλω να ευχαριστήσω τον επιβλέποντά μου από το Α.Τ.Ε.Ι Κρήτης Δρ. Πάλλη Ευάγγελο ο οποίος μου έδωσε την ευκαιρία να συνεργαστώ μαζί του. Ευγνωμοσύνη οφείλω και στους εργαστηριακούς συνεργάτες του εργαστηρίου Έρευνας και Ανάπτυξης Τηλεπικοινωνιακών Συστημάτων (ΠΑΣΙΦΑΗ), κ. Μαστοράκη Γεώργιο, κ Μαρκάκη Ευάγγελο και κ. Σιδέρη Ανάργυρο, οι οποίοι εξασφάλισαν την παροχή πλούσιας υλικοτεχνικής υποδομής, αλλά καθώς και για τις χρήσιμες συμβουλές τους όλο αυτό το διάστημα.. Ιδιαίτερες ευχαριστίες οφείλονται επίσης και στον συνάδελφο Βορνιωτάκη Νικόλαο για την πολύτιμη βοήθεια του αλλά και για την ανοχή του.

Τέλος, ευχαριστώ μέσα από την καρδία μου όλους και όλες που ήταν δίπλα μου, για την απεριόριστη ουσιαστική ψυχολογική υποστήριξη τους σε όλη την διάρκεια της πτυχιακής εργασίας και όχι μόνο….

Ηράκλειο, Μάιος 2007
Παπαδογιάννης Δημήτρης.

Περιεχόμενα:

Περιεχόμενα Πινάκων:

Περιεχόμενα Σχημάτων:

Ακρωνύμια

A

ATSC-T Advanced Television Systems Committee Terrestrial

ATSC-C Advanced Television Systems Committee Cable

API Application Program(ming) Interface

AC-3 Dolby Digital

B

BPSK Binary phase-shift keying

BER Bit Error Rate

C

COFDM Coded Orthogonal Frequency Division Multiplexing

CMN Cell Main Node

D

DTV Digital Television

DVB-C Digital Video Broadcasting-Cable

DVB-H Digital Video Broadcasting-Handover

DVB-S Digital Video Broadcasting-Satellite

DVB-T Digital Video Broadcasting-Terrestrial

DVD Digital Video Decoder

DQPSK Differential Phase-Shift Keying

DSM-CC Digital storage media command and control

DVB-J Digital Video Broadcasting Java

E

ETSI European Telecommunications Standard Institute

G

GUI Graphical User Interface

I

IEEE Institute of Electrical & Electronics Engineers

IFFT Inverse Fast Fourier Transform

IP Internet Protocol

ISDB-T International Standard Digital Broadcasting-Terrestrial

ISDB-S International Standard Digital Broadcasting- Satellite

ISDB-C International Standard Digital Broadcasting- Cable

J

JVM Java Virtual Machine

M

MHP Multimedia Home Platform

MPE Multi Protocol Encapsulation

MPEG Motion Pictures Experts Group

N

NIC Network Interface Controller

O

OCAP OpenCable Application Platform

OPEN TV Core middleware

P

PES Packetised Elementary Stream

PID Programm Identifier

Q

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

R

RDVB-T Regenerative Digital Video Broadcasting-Terrestrial

RS Reed Solomon

T

TCP Transmission Control Protocol

TS Transport Stream

TV Television

U

UDP User Datagram Protocol

UHF Ultra high frequency

V

VLC VideoLAN Client

W

WLAN Wireless Lan

X

XML Extensible Markup Language

1 ΕΙΣΑΓΩΓΗ

Μέχρι πρόσφατα, οι τηλεπικοινωνίες και η ευρύ-εκπομπή θεωρούνταν ως δυο ξεχωριστοί τεχνολογικοί τομείς που ακολουθούσαν παράλληλες και ασύνδετες πορείες για την παροχή διαφορετικών υπηρεσιών στους χρήστες/ πελάτες τους, ενώ οι παροχείς υπηρεσιών ευρύ-εκπομπής αντιμετώπιζαν την από ένα σημείο σε πολλά σημεία (point-to-multipoint) μετάδοση μονόδρομων υπηρεσιών (π.χ. ραδιόφωνο, TV, κ.λπ.), οι χειριστές τηλεπικοινωνιών εστίασαν στην παροχή αμφίδρομων από ένα σημείο σε ένα σημείο υπηρεσιών (point-to point) (π.χ. τηλέφωνο, πολυμέσα κατ' απαίτηση κ.λπ.).

Οι πρόσφατες πρόοδοι ωστόσο στις τηλεπικοινωνίες (ασύρματες και σταθερές), η εμφάνιση των ψηφιακών τεχνολογιών εκπομπής και η παγκόσμια κυριαρχία των δικτύων IP, προετοίμασαν το έδαφος προς τη σύγκλισή τους σε τεχνολογικό επίπεδο, το οποίο επιτρέπει την πραγματοποίηση μιας ενοποιημένης υποδομής δίνοντας την δυνατότητα πρόσβασης και για να μεταδοθεί και για να διανεμηθεί υλικό.

Ωστόσο, η ευρεία ποικιλία υπηρεσιών, η ετερογένεια των εφαρμογών, οι ποικίλες διατάξεις και η ανάγκη για αλληλεπίδραση στις πρόσφατες εφαρμογές, απαιτούν την σύμπραξη (καλύτερα από τη σύγκλιση) μεταξύ των τομέων εκπομπής και τηλεπικοινωνιών όχι μόνο σε τεχνολογικό αλλά και σε επίπεδο υπηρεσιών. Προς αυτά και εκτός από τα τεχνολογικά ζητήματα που λύνονται, μια κοινή, ανοικτή και εξελικτική πλατφόρμα Middle-ware πρέπει να αναπτυχθεί, προσφέροντας τη διαφανή (στο χρήστη) πρόσβαση σε οποιοδήποτε είδος περιεχομένου πέρα από τη στοίβα πρωτοκόλλου IP.

Από αυτή την άποψη, ο βασικός στόχος αυτής της πτυχιακής είναι η μελέτη, σχεδίαση, ολοκλήρωση και εφαρμογή μιας πλατφόρμας Mddle-ware, η οποία επιτρέπει την πρόσβαση σε ψηφιακά αμφίδρομα τηλεοπτικά προγράμματα, υπηρεσίες IP και πολυμέσα κατ' απαίτηση (multimedia-on-demand), συμβάλλοντας, επομένως, προς τη σύμπραξη της εκπομπής, των τηλεπικοινωνιών και των τομέων διαδικτύου και σε τεχνολογικό και σε επίπεδο υπηρεσιών.

Για να επιτύχει αυτό, αυτό το έργο υιοθετεί μια ήδη εφαρμοσμένη υποδομή συστημάτων DVB-T προτεινόμενη από το έργο IST ATHENA [3] που χρησιμοποιεί ένα αναγεννημένο ρεύμα DVB-T προκειμένου να διασυνδεθούν οι κόμβοι διανομής, επιτρέποντας την πρόσβαση σε υπηρεσίες IP, και σε ψηφιακά τηλεοπτικά προγράμματα και μπορεί να χρησιμοποιηθεί και από παροχείς υπηρεσιών και από χρήστες επιτρέποντας την μεταξύ τους σύγκλιση. Επιπλέον αυτό το πρόγραμμα εισάγει μια πλατφόρμα Mddle-ware που ενσωματωμένη με την υποδομή ATHENA θα συνδέσει διαφανώς τους συμμετέχοντες (συμπεριλαμβανομένων των τελικών χρηστών) στο ρεύμα μεταφοράς (TS).

2 Ψηφιακή τηλεόραση

2.1 Πρότυπα Ψηφιακής Τηλεόρασης

Σε αντίθεση με τα αναλογικά τηλεοπτικά συστήματα, όπου το αναλογικό σήμα χρησιμοποιείται για εκπομπή, στα ψηφιακά τηλεοπτικά (DTV) συστήματα τα τηλεοπτικά προγράμματα εκπέμπονται μέσω ψηφιακά διαμορφωμένων σημάτων με κάθε σύστημα να υλοποιείται χρησιμοποιώντας ένα συγκεκριμένο πρότυπο ψηφιακής τηλεόρασης. Σήμερα συναντάμε διάφορα πρότυπα ψηφιακής τηλεόρασης που καθένα έχει τα δικά του χαρακτηριστικά και δυνατότητες. Ο πίνακας 1 συνοψίζει τις προδιαγραφές των προτύπων ψηφιακής τηλεόρασης.

Πρότυπο	*Τύπος συστήματος*	*Τηλεοπτική και ακουστική συμπίεση*	*Διαμόρφωση*	*Εύρος ζώνης καναλιού*	*Αναλογία bit (Mbit/s)*
DVB	DVB-T	MPEG-2	QPSK QAM (C)OFDM	8MHz	Μέχρι 31,688 Mbit/s
	DVB-S		QPSK		Μέχρι 54 Mbit/s
	DVB-C		QAM		Μέχρι 38,5 Mbit/s
ATCS	ATSC-T	MPEG-2 με AC-3 για ακουστική κωδικοποίηση	8VSB	6 MHz	19.39 Mbit/s
	ATSC-C		16 VSB ή 256QAM		38.78 Mbit/s
ISDB	ISDB-T	MPEG-2	16QAM 64QAM QPSK και DQPSK with OFDM	5,6MHz	19 Mbit/s (64QAM)
	ISDB-S		TC8PSK QPSK BPSK	34,5MHz	51 Mbit/s (TC8PSK)
	ISDB-C		64QAM	6MHz	31,544 Mbit/s

Πίνακας 1: Οι παράμετροι των προτύπων ψηφιακής τηλεόρασης.

Υπάρχουν δύο ακόμα συστήματα που ακολουθούν το πρότυπο ψηφιακής τηλεόρασης που δεν αναφέρεται στον πίνακα 1 και αυτά είναι:

DVB-H: Digital Video Broadcasting-Handheld. Φορητή Ψηφιακή Εκπομπή Βίντεο είναι μια προδιαγραφή γνωστή επίσημα ως *πρότυπο ETSI EN 302.304* [14] και είναι ένα από τα πρότυπα DVB.Το DVB-H υιοθετεί το DVB-T φυσικό στρώμα με μερικές αλλαγές και στόχους στην παράδοση των υπηρεσιών DVB στις φορητές συσκευές.

DVB-S2: Digital Video Broadcasting-Satellite Δορυφορική Ψηφιακή Εκπομπή Βίντεο 2 [15] είναι μια εκσυγχρονισμένη προδιαγραφή για τις ευρυζωνικές δορυφορικές εφαρμογές (μια αναπροσαρμογή του DVB-S) παρέχοντας 30% μεγαλύτερη αποδοτικότητα από το DVB-S ενώ διατηρεί αντίθετη συμβατότητα με τα συστήματα DVB-S.

Μια ερώτηση που θα μπορούσε να κάνει κάποιος είναι τι μπορεί να προσφέρει η ψηφιακή τηλεόραση που δεν μπορεί η αναλογική τηλεόραση; Η απάντηση σε αυτή την ερώτηση μπορεί να δοθεί με την επεξήγηση των πλεονεκτημάτων [Rysdale96] της ψηφιακής τηλεόρασης, μερικά από αυτά είναι τα ακόλουθα.:

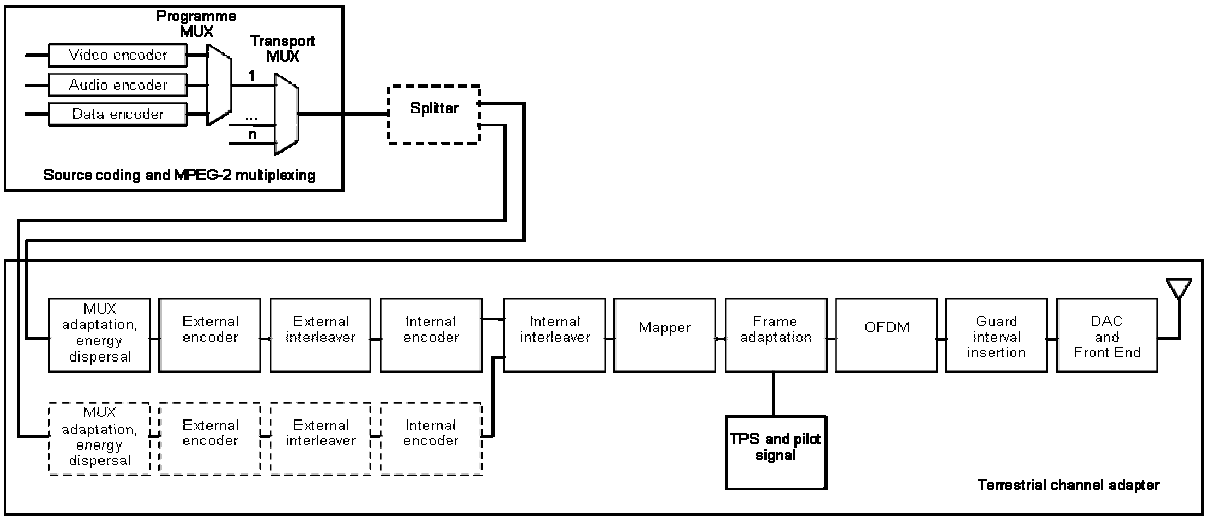
1. Καλύτερη ποιότητα εικόνας με μεγαλύτερη ανοχή στις ατέλειες της καλωδιωμένης ή ασύρματης σύνδεσης.
2. Λιγότερα BER σε σύγκριση με την αναλογική μετάδοση. Παραδείγματος χάριν η μεταδιδόμενη ισχύς μπορεί να μειωθεί κατά 30dB χωρίς διαστρέβλωση εικόνας.
3. Καλύτερη εκμετάλλευση φάσματος λόγω της συμπίεσης του σήματος ζωνών βάσης. Παραδείγματος χάριν ένα UHF κανάλι μπορεί να φιλοξενήσει τέσσερα υψηλής ποιότητας (ποιότητα DVD) προγράμματα ψηφιακής τηλεόρασης αντί ενός αναλογικού τηλεοπτικού προγράμματος.
4. Ευέλικτες τεχνικές πολύπλεξης που επιτρέπουν τη συνύπαρξη διάφορων προγραμμάτων / υπηρεσιών με ορισμένη ποιότητα κάθε μιας.
5. Μεταβλητή μετάδοση bitrate ανάλογα με τις απαιτήσεις του προγράμματος.
6. Η ενσωμάτωση των αμφίδρομων υπηρεσιών πολυμέσων από μια πλατφόρμα DTV με την προϋπόθεση ότι υπάρχει ένα κανάλι επιστροφής.
7. Βελτίωση του προγράμματος με την υιοθέτηση τυποποιημένων αρχιτεκτονικών (ex. MHP, OpenTV).

2.2 Ψηφιακή επίγεια τηλεοπτική εκπομπή

Το DVB-T πρότυπο (ETSI 300 744: Digital Video Broadcasting (DVB): Η δομή πλαισίου, η κωδικοποίηση καναλιών και διαμόρφωση για ψηφιακή επίγεια τηλεόραση) [5] καθορίζουν τις προδιαγραφές για την επίγεια εκπομπή ψηφιακών τηλεοπτικών προγραμμάτων. Το DVB-T αναπτύχθηκε από το forum DVB [4] και εγκρίθηκε από το ETSI [16] ως ευρωπαϊκό πρότυπο το 1997. Ενώ το πρότυπο DVB-T καθορίζει ένα σύστημα μετάδοσης με ευρεία κάλυψη δεν καθορίζει τις τεχνολογικές προδιαγραφές αντίστροφων πορειών αλλά μόνο περιγράφει τη λειτουργία του διαμορφωτή που έχει το σήμα ζωνών βάσης και τον αλγόριθμο συμπίεσης εικόνας που περιγράφεται από το MPEG-2 [17] πρότυπο . Ένας διαμορφωτής DVB-T λαμβάνει το πολυπλεγμένο MPEG-2 ρεύμα μεταφοράς (TS) που περιέχει τα πολυπλεγμένα στοιχεία, τις τηλεοπτικές και ακουστικές υπηρεσίες ως σήμα ζωνών βάσης και παράγει το σήμα μετάδοσης RF. Το σήμα RF θα έχει εύρος ζώνης 8MHz και θα συγκεντρωθεί σε ένα UHF κανάλι (κανάλια 21-69 της UHF ζώνης συχνοτήτων). Η λειτουργία ενός διαμορφωτή DVB-T για τη μετατροπή ενός σήματος ζωνών βάσης σε ένα TS απεικονίζεται στο σχήμα 2.1.Οι λειτουργίες που εφαρμόζονται στο TS είναι οι ακόλουθες:

- Προσαρμογή πολυπλεγμένης μεταφοράς και τυχαιοποίησης για την διανομή ενέργειας .
- Εξωτερική κωδικοποίηση (προστασία λάθους με τον κώδικα Reed -Solomon).
- Εξωτερική συνέλιξη διαστρωμάτωσης .
- Εσωτερική κωδικοποίηση με τον διάτρητο κώδικα συνέλιξης.
- Εσωτερική διαστρωμάτωση (σε χρόνο και συχνότητα) .
- Χαρτογράφηση και διαμόρφωση φέροντος .
- OFDM πολύπλεξη με τον αντίστροφο μετασχηματισμό Φουριέ (IFFT) και διαμόρφωση του IF φέροντος.
- Ανοδική μετατροπή στη συχνότητα RF.

Πρέπει να αναφερθεί ότι η τελευταία λειτουργία (ανοδική μετατροπή) δεν υποστηρίζεται από διάφορους διαμορφωτές και μια πρόσθετη ενότητα είναι απαραίτητη για την λειτουργία ανοδικής μετατροπής.



Σχήμα 2.1: Διάγραμμα λειτουργειών ενός διαμορφωτή DVB-T.

Η παράμετρος που μπορεί να είναι μεγάλου ενδιαφέροντος για μια πλατφόρμα DVB-T είναι το διαθέσιμο εύρος ζώνης bitrate που μπορεί να μεταφερθεί από ένα ψηφιακό σήμα (MPEG-2 TS ρυθμός μετάδοσης). Αυτή η τιμή εξαρτάται από το διάστημα φρουράς, τον τύπο διαμόρφωσης και την αναλογία κώδικα.

Αυτές οι παράμετροι (που εφαρμόζονται για έναν διαμορφωτή) θα διορίσουν το διαθέσιμο bit rate για ένα σύστημα DVB.

Διαμόρφωση	*Αναλογία κώδικα*	*Διάστημα φρουράς*			
		1/4	**1/8**	**1/16**	**1/32**
QPSK	**1/2**	4.976	5.529	5.855	6.032
	2/3	6.635	7.373	7.806	8.043
	3/4	7.465	8.294	8.782	9.048
	5/6	8.294	9.216	9.758	10.053
	7/8	8.709	9.676	10.246	10.556
16QAM	**1/2**	9.953	11.059	11.709	12.064
	2/3	13.271	14.745	15.612	16.086
	3/4	14.929	16.588	17.564	18.096
	5/6	16.588	18.431	19.516	20.107
	7/8	17.418	19.353	20.491	21.112
64QAM	**1/2**	14.929	16.588	17.564	18.096
	2/3	19.906	22.118	23.419	24.128
	3/4	22.394	24.882	26.346	27.144
	5/6	24.882	27.647	29.273	30.160
	7/8	26.126	29.029	30.737	31.668

Πίνακας 2: Το διαθέσιμο bit rate προς το σχέδιο διαμόρφωσης.

Πρέπει να αναφερθεί ότι καθώς η αξία του bit rate αυξάνεται το μεταδιδόμενο σήμα γίνεται περισσότερο ευάλωτο στην εξασθένιση και σε επιδράσεις πολλαπλών διαδρομών. Γι' αυτό το λόγο πρέπει να υπάρξει μια χρυσή τομή μεταξύ της χωρητικότητας του καναλιού και της ευπάθειας του σήματος.

3 Αμφίδρομο DVB

Αν και το πρότυπο DVB-T ορίζει μονόδρομο τρόπο μετάδοσης, η ενθυλάκωση των πακέτων IP στο ρεύμα μεταφοράς είναι δυνατή. Αυτό το χαρακτηριστικό μπορεί να επιτρέψει την παροχή αμφίδρομων εφαρμογών υπό τον όρο ότι υπάρχει επιστροφής κανάλι. Η εισαγωγή ενός καναλιού επιστροφής μπορεί να προσφέρει την επικοινωνία μεταξύ του φορέα παροχής υπηρεσιών και του χρήστη. Με αυτόν τον τρόπο μπορεί να διαμορφωθεί ένα ασυμμετρικό υβριδικό δίκτυο συμβατό με το πρότυπο ETS 300 802 [10] .

3.1 IP μέσω DVB

Όπως αναφέρθηκε ένα χαρακτηριστικό του πρότυπου DVB-T είναι η ενθυλάκωση των πακέτων IP στα πακέτα μεταφοράς. Τα πακέτα έχουν μήκος 188 bytes χρησιμοποιώντας τέσσερα από αυτά ως header(επικεφαλίδα). Για την ενσωμάτωση των πακέτων IP στο TS και για να διακριθούν από τα ψηφιακά πακέτα τηλεοπτικών προγραμμάτων χρειαζόμαστε μια διαδικασία που θα εκτελέσει τη λειτουργία της χαρτογράφησης, την προσαρμογή και την κατάτμηση, αυτές οι λειτουργίες ορίζονται από το πρότυπο ETSI EN 301 192 (Digital Video Broadcasting (DVB): DVB προδιαγραφή για εκπομπή δεδομένων) [7]. Αυτό το πρότυπο καθορίζει τέσσερις τεχνικές ενθυλάκωσης.

- Τεχνική σωλήνωσης στοιχείων (Data piping technique). Αυτή η τεχνική επιτρέπει στα πακέτα IP δεδομένων να ενσωματωθούν ως ωφέλιμο φορτίο στα πακέτα δεδομένων MPEG-2 TS.

- Ροή δεδομένων (Data Streaming) όπου το ρεύμα δεδομένων διαμορφώνεται με ένα συμβατό MPEG-2 στοιχειώδες ρεύμα το οποίο από κάθε στροφή οργανώνεται σε πακέτα ακολουθώντας την δομή Πακεταρισμένο Στοιχειώδες Ρεύμα (PES). Τελικά τα PES πακέτα τεμαχίζονται και διανέμονται στο MPEG-2 ωφέλιμο φορτίο πακέτων μεταφοράς.

- Ενθυλάκωση πολυπρωτοκόλλων (Multiprotocol Encapsulation). Αυτή η τεχνική επιτρέπει τη μεταφορά διάφορων πρωτοκόλλων (π.χ. TCP/IP) μέσω του καναλιού DVB έχοντας ενσωματώσει τα πακέτα δεδομένων σε τμήματα δεδομένων όπως καθορίζονται στο πρότυπο MPEG-2 DSM-CC. Αυτά τα τμήματα δεδομένων είναι συμβατά στο private_section που καθορίζεται στο at ISO/IEC 13818-1 και μπορούν να ενσωματωθούν απευθείας στο TS. Για τη διάκριση αυτών των πακέτων χρησιμοποιούμε έναν αριθμό ταυτότητας πακέτου (PID) , το πεδίο MAC και την IP διεύθυνση.

- Ιπποδρόμιο δεδομένων (Data Carousel). Αυτή η τεχνική είναι κατάλληλη για μη-αμφίδρομη μετάδοση δεδομένων ενώ τα μεταδιδόμενα δεδομένα είναι συνήθως στόχος σε μεγάλη ομάδα χρηστών. Τα δεδομένα ιπποδρομίου μεταδίδονται κατά μια περιοδική αναλογία.

3.2 Middle-ware

Το Middle-ware είναι ένας σχετικά καινούριος όρος στην πληροφορική (εξελίχθηκε κατά τη διάρκεια του 1990's [2]), συνήθως αναφέρεται ως λογισμικό ικανό να διασυνδέσει τμήματα και εφαρμογές λογισμικού σε έναν ή μεταξύ πολλών υπολογιστών μέσω ενός δικτύου [1][2]. Επιπλέον το Middle-ware έχει την ίδια έννοια όταν χρησιμοποιείται στην ψηφιακή τηλεόραση, είναι μια στοίβα λογισμικού που βρίσκεται μεταξύ της εφαρμογής και των οδηγών των συσκευών. Στην αγορά ψηφιακής τηλεόρασης συναντάμε διάφορες τεχνολογίες Middle-ware όπως MediaHighway και OpenTV που είναι ιδιόκτητες τεχνολογίες Middle-ware και MHP, JavaTV και OCAP που είναι ανοικτές λύσεις Middle-ware. Ένα σημαντικό ζήτημα που πρέπει να αντιμετωπιστεί σχετικά με τις τεχνολογίες Middle-ware για την ψηφιακή τηλεόραση είναι η διαλειτουργικότητα μεταξύ αυτών των τεχνολογιών. Αυτή η εργασία προτείνει μια ανοικτή πλατφόρμα Middle-ware που επιτρέπει την παροχή διαδραστικής εφαρμογής που επεκτείνεται σε ένα περιβάλλον DVB-T.

3.3 Multimedia Home Platform (MHP) και JavaTV

Το MHP [8] και JavaTV [9] είναι δύο παραδείγματα ανοικτών τεχνολογιών Middle-ware. Το JavaTV αναπτύχθηκε από τα SUN Microsystems INC [18] ενώ το MHP αναπτύχθηκε από το forum DVB.

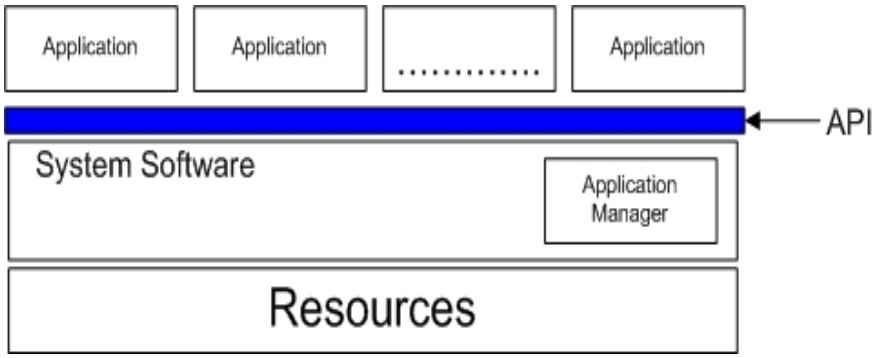
3.3.1 MHP

DVB-MHP (Digital Video Broadcasting: Multimedia Home Platform) [8] σχεδιάστηκε από το forum DVB και την EBU (Ευρωπαϊκή Ένωση εκπομπής) και καθιερώθηκε από το ETSI. Το DVB-MHP πρότυπο προσθέτει μια τεχνική λύση για την υλοποποίηση αμφίδρομων και μη αμφίδρομων ψηφιακών εφαρμογών. Βασικά το MHP μπορεί να περιγραφεί ως μοντέλο εφαρμογής που μπορεί να περιγράψει πώς συμπεριφέρονται οι εφαρμογές MHP, πώς μπορεί ένας δέκτης να ενημερωθεί για τη διαθεσιμότητα μιας εφαρμογής και πώς ορισμένα αρχεία που απαιτούνται για την εφαρμογή φορτώνονται στο δέκτη. Το MHP μπορεί επίσης να περιγραφεί ως περιβάλλον εκτέλεσης εφαρμογής στο οποίο μπορούν να επεκταθούν ψηφιακές αμφίδρομες εφαρμογές, αυτό το περιβάλλον εφαρμογής χρησιμοποιεί μια Java Virtual Machine ενώ η πρόσβαση στους πόρους παρέχεται από ένα γενικό API (διεπαφή προγραμματισμού εφαρμογής) βασισμένο σε ένα υποσύνολο της personal Java 1.2.

Ένα άλλο ζήτημα που καθορίζεται στο πρότυπο DVB-MHP είναι ότι η παραλαβή και η παρουσίαση των εφαρμογών MHP εκτελούνται σε ένα ουδέτερο πλαίσιο πωλητών, συντακτών και παροχέων υπηρεσιών που σημαίνει ότι οι εφαρμογές από διαφορετικούς φορείς παροχής υπηρεσιών είναι διαλειτουργικές με διαφορετικές υλοποιήσεις MHP σε μια οριζόντια αγορά.

Γενικά όλα τα μοντέλα MHP καθορίζονται από τρία στρώματα: Το στρώμα πόρων, το στρώμα λογισμικό συστήματος και το στρώμα εφαρμογής. Το στρώμα πόρων καθορίζει τις οντότητες υλικού ή λογισμικού που εκτελούν διάφορες εργασίες (πρώην επεξεργασία MPEG), το στρώμα λογισμικό συστήματος χρησιμοποιεί τους διαθέσιμους πόρους και απομονώνει την εφαρμογή από το υλικό.

Για τον έλεγχο των πόρων τους και την πρόσβαση στο περιεχόμενο MHP οι MHP εφαρμογές χρησιμοποιούν το MHP API, αυτό το API παρέχει ένα στρώμα αφαίρεσης μεταξύ των εφαρμογών και των τελικών υλοποιήσεων υλικού ή λογισμικού MHP. Οι εφαρμογές MHP μπορούν να αναπτυχθούν χρησιμοποιώντας διάφορα πακέτα λογισμικού που είναι βασισμένα στη γλώσσα προγραμματισμού Java (DVB-J) ή χρησιμοποιώντας τη γλώσσα DVB-HTML που είναι ένα σύνολο ενοτήτων XHTML 1.1.



Σχήμα 3.1: Βασική αρχιτεκτονική MHP

Για την παροχή επικοινωνίας με διαφορετικούς τύπους δικτύων το MHP καθορίζει τα πρωτόκολλα δικτύων και μεταφορών που μπορούν να χρησιμοποιηθούν και είναι κατάλληλα για εκπομπή και αμφίδρομες εφαρμογές. Αυτά τα πρωτόκολλα είναι τα DSM-CC User-to-User, Data and Object Carousel πρωτόκολλα ενώ υπάρχει υποστήριξη για τη χρησιμοποίηση την IP με τη χρησιμοποίηση της τεχνικής Multiprotocol Encapsulation.

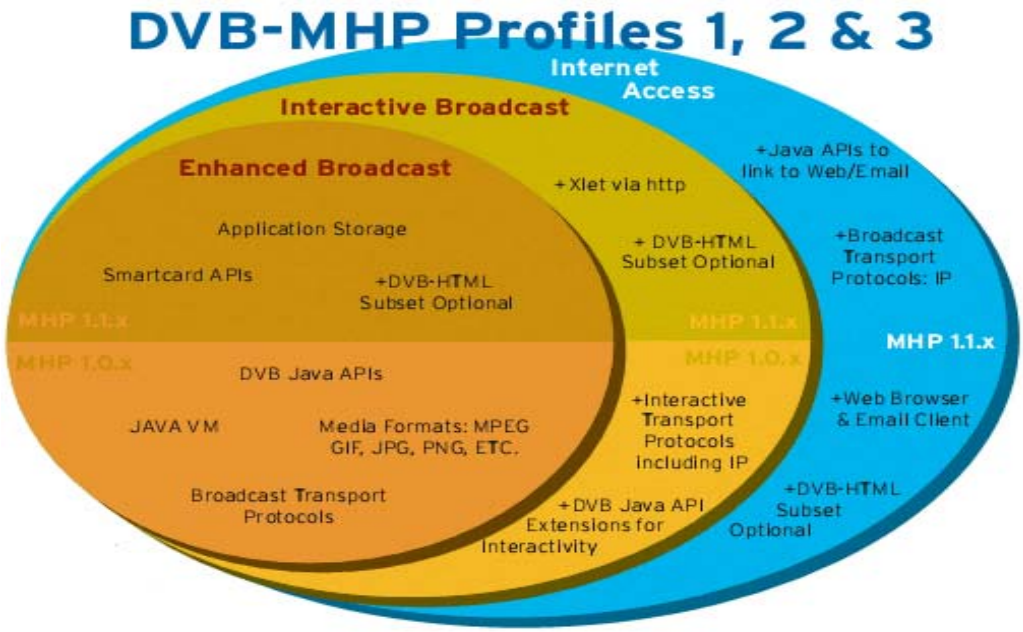
3.3.2 JavaTV

Η τεχνολογία JavaTV για την ψηφιακή τηλεόραση είναι βασισμένη στην πλατφόρμα Java™. Τα κύρια συστατικά της JavaTV είναι η Java Virtual Machine εικονική μηχανή της Java (JVM) και ένα σύνολο βιβλιοθηκών της Java. Η τεχνολογία JavaTV καθορίζει ένα API ως επέκταση στην πλατφόρμα της Java που παρέχει ένα σύνολο βιβλιοθηκών ορισμένες για τηλεόραση. Για την εκτέλεση εφαρμογών χρησιμοποιείται ένα JVM διανεμημένο στον δέκτη ψηφιακής τηλεόρασης. Αν και οι βιβλιοθήκες ουσιώδης κλάσης JVM και JavaTV καθορίζουν ένα λειτουργικό περιβάλλον εκτέλεσης, προαιρετικές βιβλιοθήκες μπορούν να περιληφθούν για συγκεκριμένες απαιτήσεις σχετικά με μια εφαρμογή.

3.3.3 Περιοχές Εφαρμογής/Υπηρεσιών MHP & JavaTV

Οι περιοχές εφαρμογής που καθορίζονται από το πρότυπο MHP είναι οι ακόλουθες:

- Ενισχυμένη εκπομπή: Η ενισχυμένη εκπομπή περιλαμβάνει την τηλεοπτική και ακουστική εκπομπή μαζί με εφαρμογές που φορτώνονται στο τερματικό του χρήστη και παρέχουν τοπική αλληλεπίδραση. Αυτή η περιοχή εφαρμογών δεν απαιτεί οποιοδήποτε είδος καναλιού αλληλεπίδρασης.
- Αμφίδρομη εκπομπή: Η αμφίδρομη εκπομπή περιλαμβάνει αμφίδρομες υπηρεσίες που μπορούν να συνδεθούν ή να είναι ανεξάρτητες με άλλες υπηρεσίες εκπομπής.
- Πρόσβαση Διαδικτύου: Η περιοχή υπηρεσιών πρόσβασης Διαδικτύου εστιάζει στην παροχή πρόσβασης Διαδικτύου με μια πλατφόρμα DVB.



Σχήμα 3.2:Περιοχές εφαρμογής MHP [6]

4 Αρχιτεκτονική συστήματος DVB-T

Οι πρόσφατες ερευνητικές προσπάθειες δείχνουν ότι οι αναπαραγωγικές διαμορφώσεις και αρχιτεκτονικές DVB-T (RDVB-T) μπορούν να χρησιμοποιηθούν ως βάση για την πραγματοποίηση υποδομών που μπορεί να χρησιμοποιηθούν κοινά και ουδέτερα από παροχείς υπηρεσιών, χειριστές τηλεπικοινωνιών και παροχείς περιεχομένου [3]. Από αυτή την άποψη αυτή η πτυχιακή υιοθετεί μια ήδη προτεινόμενη και εφαρμοσμένη υποδομή συστήματος DVB-T που εκτός από την παροχή ψηφιακού τηλεοπτικού προγράμματος μπορεί να χρησιμοποιηθεί για την παροχή υπηρεσιών IP.

4.1 Γενική αρχιτεκτονική συστήματος DVB-T

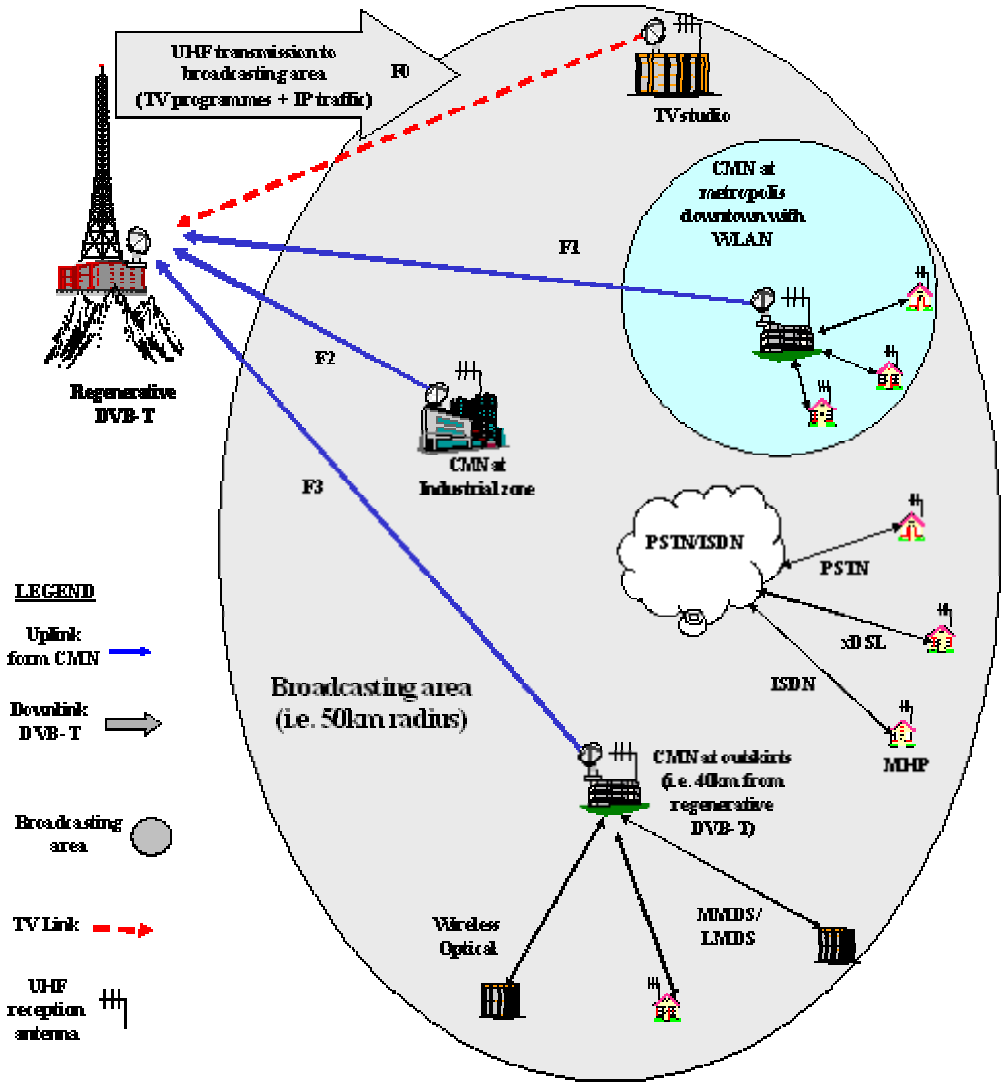
Όπως αναφέρθηκε μια πλατφόρμα DVB-T μπορεί εκτός από τη μετάδοση MPEG-2 τηλεοπτικών προγραμμάτων να χρησιμοποιηθεί για την παροχή αμφίδρομων υπηρεσιών IP προϋποθέτοντας την ύπαρξη ενός καναλιού επιστροφής που επιτρέπει στους χρήστες να αποκτήσουν πρόσβαση σε αυτές τις υπηρεσίες ενώ η παροχή πραγματοποιείται μέσω downlink DVB-T. Μια τέτοια υποδομή έχει προταθεί και έχει επεκταθεί από το IST έργο ATHENA [3]. Αυτό το πρόγραμμα υιοθετεί την υποδομή συστήματος ATHENA DVB-T.

Το IST έργο ATHENA έχει προτείνει και επεκτείνει ένα πλήρες σύστημα DVB-T ικανό να παρέχει αμφίδρομες και μονόδρομες υπηρεσίες. Η συνολική αρχιτεκτονική περιλαμβάνει δύο υποσυστήματα (σχήμα 4.1):

1. Διάφοροι ενδιάμεσοι κόμβοι διανομής (CMN) και
2. Ένα κεντρικό σημείο εκπομπής.

Κάθε CMN φιλοξενεί διάφορους χρήστες ενώ κάθε χρήστης μπορεί να αποκτήσει πρόσβαση στις υπηρεσίες IP μέσω του αντιστοιχισμένου CMN χρησιμοποιώντας ευρυζωνικές point-to-multipoint συνδέσεις (δηλ. WLAN). Όλη η IP κίνηση που προέρχεται από τους χρήστες συγκεντρώνεται και διαβιβάζεται στο κεντρικό σημείο εκπομπής από το CMN μέσω αφιερωμένων συνδέσμων point-to-point. Στο σημείο εκπομπής όλη η IP κίνηση που προέρχεται από όλα τα CMN φιλτράρεται, αναπαράγεται και πολλαπλασιάζεται (IP-πολύπλεξη) μαζί με ψηφιακό τηλεοπτικό πρόγραμμα σε ένα ενιαίο ρεύμα μεταφοράς που είναι η είσοδος σε ένα COFDM (κωδικοποιημένη ορθογώνια πολύπλεξη συχνότητας) [11] διαμορφωτή και ένα ενισχυτή ισχύος και μεταδίδεται μέσα στη UHF ζώνη σύμφωνα με το πρότυπο DVB-T [5]. Κάθε χρήστης λαμβάνει τις κατάλληλες απαντήσεις IP έμμεσα μέσω του αντιστοιχισμένου του CMN.

Το ρεύμα DVB-T χρησιμοποιείται σε μια θεμελιώδη τοπολογία και διασυνδέει όλους τους CMN που βρίσκονται μέσα στην περιοχή εκπομπής. Ως εκ τούτου ένα εικονικό κοινή θεμελιώδες Ethernet είναι παρόν σε κάθε CMN παρέχοντας την IP κίνηση από το ρεύμα DVB-T. Επιπλέον κάθε CMN αποτελεί μια φυσική διεπαφή για αυτό το δίκτυο ενώ οι χρήστες μπορούν να έχουν πρόσβαση σε αυτό το δίκτυο μέσω των CMN. Μια τέτοια υποδομή μπορεί να χρησιμοποιηθεί για την παροχή ετερογενών υπηρεσιών / εφαρμογών από παροχείς υπηρεσιών και χειριστές τηλεπικοινωνιών ή ακόμα και από ανεξάρτητους χρήστες που επιθυμούν να δημιουργούν, να χειρίζονται και να παρέχουν τις υπηρεσίες / εφαρμογές τους σε ολόκληρο το δίκτυο που συμβάλλει στη μεταξύ του σύμπραξης (καλύτερα από τη σύγκλιση).



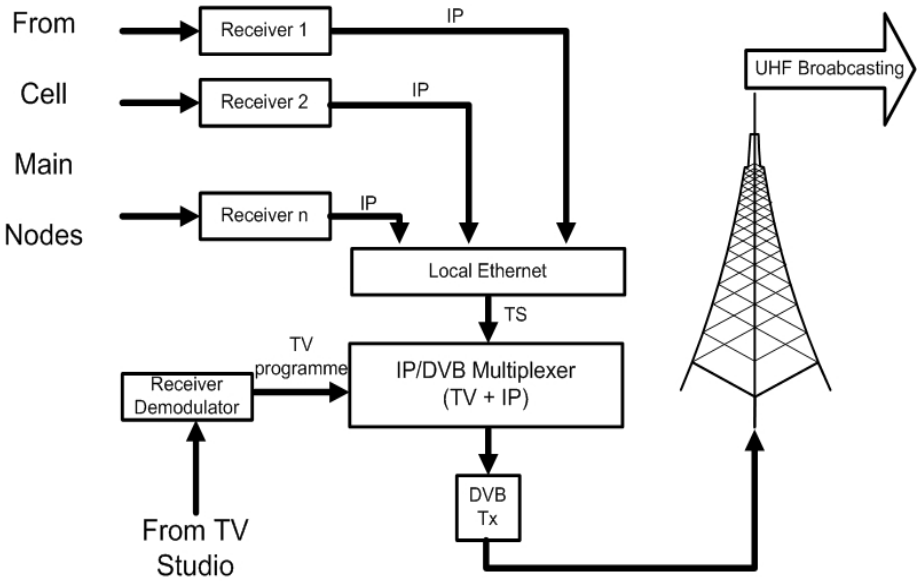
Σχήμα 4.1: Η γενική αρχιτεκτονική συστήματος DVB-T

4.2 Κεντρικό σημείο εκπομπής

Η αρχιτεκτονική κεντρικού σημείου εκπομπής (σχήμα 4.2) ακολουθεί την αναπαραγωγική έννοια DVB-T και χρησιμοποιείται από όλους τους συμμετέχοντες (παροχείς, χειριστές τηλεπικοινωνιών και ανεξάρτητοι προμηθευτές περιεχομένου) στο ρεύμα μεταφοράς. Σε αυτό το πλαίσιο η αναπαραγωγική πλατφόρμα DVB-T είναι ικανή:

1. Λήψη την IP κίνηση χρηστών (μέσω των CMN) μέσω επίγειων uplinks.
2. Λήψη οποιουδήποτε ψηφιακού τηλεοπτικού προγράμματος που προέρχεται από παροχείς τηλεοπτικού στούντιο.
3. Εκπομπή IP δεδομένων απευθυνόμενων σε όλα τα CMN (που βρίσκονται μέσα στην περιοχή εκπομπής) μαζί με το ψηφιακό τηλεοπτικό πρόγραμμα μέσω ενός κοινού UHF downlink..

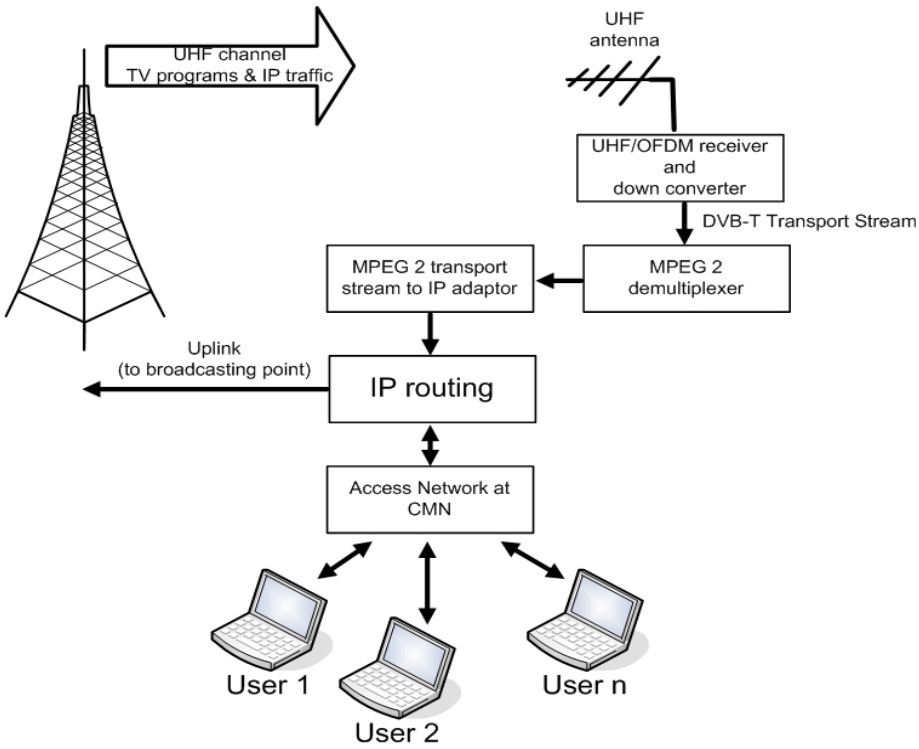
Από αυτή την άποψη η συσκευή πολύπλεξης λαμβάνει οποιοδήποτε τύπο IP δεδομένων ή / και τηλεοπτικού προγράμματος, προσαρμόζει οποιαδήποτε δεδομένα IP και τηλεοπτικό πρόγραμμα MPEG-2 σε ένα ενιαίο ρεύμα μεταφοράς DVB-T (IP στην ενθυλάκωση MPEG-2) και εκπέμπει το ρεύμα DVB-T σύμφωνα με το πρότυπο DVB-T .



Σχήμα 4.2: Γενική διαμόρφωση του αναπαραγωγικού DVB-T

4.3 Ενδιάμεσος κόμβος διανομής (CMN)

Όπως αναφέρθηκε οι χρήστες αποκτούν την πρόσβαση στις υπηρεσίες IP μέσω των αντιστοιχισμένων τους CMN τους. Επιπλέον κάθε CMN μπορεί να αντιμετωπισθεί ως κοινή εικονική φυσική διεπαφή που επιτρέπει την πρόσβαση (στους χρήστες) σε ένα θεμελιώδες δίκτυο (DVB-T TS). Η επικοινωνία μεταξύ ενός CMN και των φιλοξενημένων χρηστών του εκτελείται μέσω point-to-multipoint συνδέσεων (δηλ. WLAN). Η γενική αρχιτεκτονική ενός CMN απεικονίζεται στο σχήμα 4.3. Ένα τέτοιο CMN λαμβάνει και διαβιβάζει όλη την κίνηση IP προερχόμενη από τους φιλοξενημένους χρήστες του στο σημείο εκπομπής. Επιπλέον το CMN διαβιβάζει όλη την κίνηση IP που προορίζεται για τους τελικούς χρήστες και είναι διαθέσιμη από το ρεύμα μεταφοράς DVB.



Σχήμα 4.3: Γενική διαμόρφωση του CMN

5 Αρχιτεκτονική Middle-ware

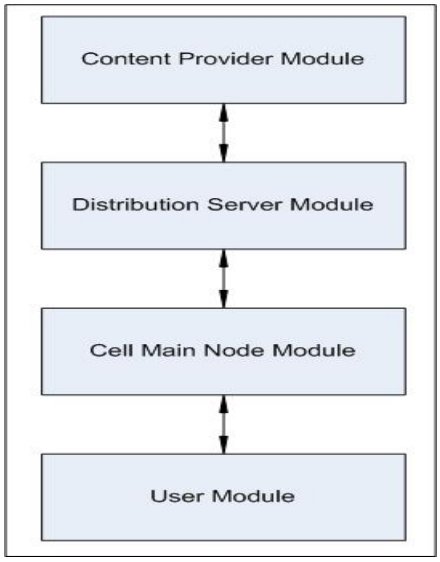
Το Middle-ware για την ψηφιακή τηλεόραση δεν είναι μια πρόσφατη έννοια, OpenTV, MediaHighway και πολλές άλλες επιχειρήσεις είναι στην αμφίδρομη αγορά τηλεοπτικού Middle-ware για πολύ καιρό ενώ οι τυποποιημένοι οργανισμοί έχουν αναπτύξει ανοικτές λύσεις Middle-ware για σχεδόν τόσο πολύ. Η χρησιμοποίηση Middle-ware για αμφίδρομη τηλεόραση έχει διάφορα πλεονεκτήματα όπως ότι κρύβει τις υποστρωματικές διαφορές υλικού και επιτρέπει τη φορητότητα σχετικά με τις προδιαγραφές υλικού και λειτουργικού συστήματος. Επιπλέον η χρησιμοποίηση Middle-ware για την ψηφιακή τηλεόραση επιτρέπει τη διαφανή πρόσβαση πέρα από μια σειρά υπηρεσιών πολυμέσων [12]. Αυτή η εργασία υιοθετεί την έννοια μιας εναλλακτικής τεχνολογίας Middle-ware που, όπως τις περισσότερες από τις τεχνολογίες Middle-ware, επιτρέπει στους χρήστες να αποκτήσουν διαφανώς πρόσβαση σε υπηρεσίες μέσω της στοίβας πρωτοκόλλου IP. Αυτό το κεφάλαιο περιγράφει την αρχιτεκτονική της τεχνολογίας Middle-ware που σχεδιάζεται και υλοποιείται προκειμένου να προσφερθεί διαφανής πρόσβαση (στο χρήστη) σε οποιοδήποτε είδος υπηρεσιών μέσω της στοίβας πρωτοκόλλου IP.

5.1 Γενική αρχιτεκτονική Middle-ware

Όπως αναφέρθηκε η χρησιμοποίηση Middle-ware για αμφίδρομη τηλεόραση επιτρέπει στους χρήστες να έχουν πρόσβαση διαφανώς στις υπηρεσίες άσχετα από το υλικό και το λειτουργικό σύστημα που χρησιμοποιούνται. Με βάση αυτήν την έννοια αυτό το κεφάλαιο επεξεργάζεται το σχέδιο και την αρχιτεκτονική ενός εναλλακτικού Middle-ware το οποίο ουδέτερα και διαφανώς συνδέει τις ενότητες υλικού με τις παρεχόμενες υπηρεσίες επιτρέποντας στους χρήστες να έχουν πρόσβαση στις υπηρεσίες IP που παρέχονται από μια πλατφόρμα DVB-T. Η γενική αρχιτεκτονική Middle-ware αποτελείται από τέσσερις ενότητες (σχήμα 5.1).

1. Η ενότητα Παροχέας Υπηρεσιών (Content Provider)
2. Η ενότητα Εξυπηρετητής Διανομής (Distribution Server)
3. Η ενότητα CMN (Cell Main Node)
4. Η ενότητα Χρήστη (Interactive or Active User)

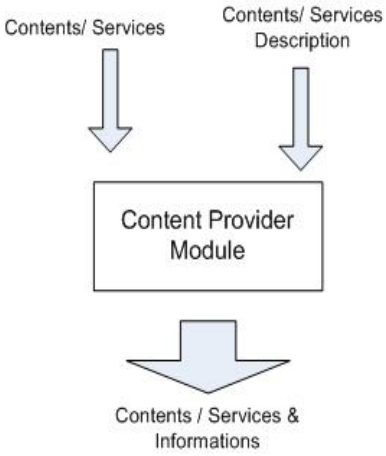
Κάθε παροχέας υπηρεσιών είναι εξοπλισμένος με μια ενότητα λογισμικού που επιτρέπει τη διαφανή παροχή υπηρεσιών στους τελικούς χρήστες. Η ενότητα παροχέας υπηρεσιών επιτρέπει εκτός από την παροχή υπηρεσιών (π.χ. πολλαπλή διανομή βίντεο) τη διανομή πληροφοριών επιτρέποντας σε έναν παροχέα υπηρεσιών να κοινοποιήσει την οντότητά της και τις παρεχόμενες υπηρεσίες της. Οι πληροφορίες υπηρεσιών που προέρχονται από τον παροχέα υπηρεσιών χρησιμοποιούνται από το CMN και τις ενότητες χρηστών προκειμένου να επιτραπεί η πρόσβαση στις παρεχόμενες υπηρεσίες. Η ενότητα CMN εγκαθίσταται στο CMN και χρησιμοποιείται για την ανίχνευση υπηρεσιών και τις υπηρεσίες που διαβιβάζονται στο Middle-ware χρήστη. Κατ' αυτό τον τρόπο ενημερώνονται οι τελικοί χρήστες για και έχουν πρόσβαση στις διαθέσιμες υπηρεσίες έμμεσα από το αντιστοιχισμένο τους CMN.



Σχήμα 5.1: Γενική αρχιτεκτονική του προτεινόμενου Middle-ware

5.2 Ενότητα Παροχέας Υπηρεσιών

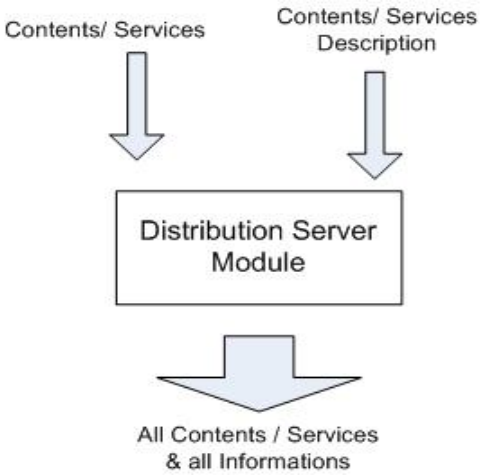
Όπως αναφέρθηκε κάθε παροχέας υπηρεσιών είναι εξοπλισμένος με μια ενότητα λογισμικού (σχήμα 5.2) που επιτρέπει εκτός από την παροχή υπηρεσιών, τη διανομή πληροφοριών σχετικά με την οντότητα παροχέα υπηρεσιών και τις παρεχόμενες υπηρεσίες. Αυτό το είδος πληροφοριών έχει δύο σκοπούς i)δίνει την δυνατότητα στους παροχείς υπηρεσιών να κοινοποιήσει την ύπαρξή τους και ii) επιτρέπει τη μοναδική διάκριση των υπηρεσιών. Από αυτή την άποψη κάθε παροχέας υπηρεσιών μπορεί να θεωρηθεί ως διαφορετική πηγή περιεχομένου και υπηρεσιών ενώ κάθε υπηρεσία μπορεί να περιγραφεί και να προσεγγιστεί σύμφωνα με τις πληροφορίες περιγραφής που συνοδεύονται με αυτόν κάτι που με τη σειρά του επιτρέπει την παροχή ετερογενών υπηρεσιών από έναν ενιαίο παροχέα υπηρεσιών.



Σχήμα 5.2: Γενική αρχιτεκτονική ενότητας παροχέα υπηρεσιών

5.3 Ενότητα Εξυπηρετητής Διανομής

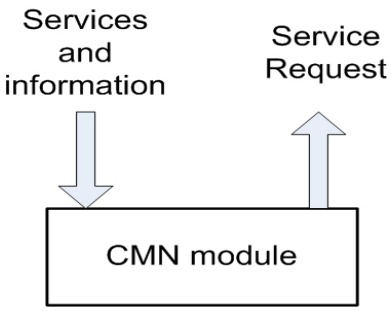
Η ενότητα αυτή αντιπροσωπεύει τον ενιαίο παροχέα υπηρεσιών (σχήμα 5.3) δηλαδή δέχεται όλες τις πληροφορίες των διαθέσιμων υπηρεσιών, τις ενοποιεί και στην συνέχεια τις στέλνει σαν ένα στοιχείο στην ενότητα CMN. Επιπλέον παρέχεται η δυνατότητα για έλεγχο των παροχέων υπηρεσιών. Η ενότητα αυτή εγκαθίσταται σε κάθε σύστημα εκπομπής DVB-T.



Σχήμα 5.3: Γενική αρχιτεκτονική ενότητας Εξυπηρετητής Διανομής

5.4 Ενότητα CMN

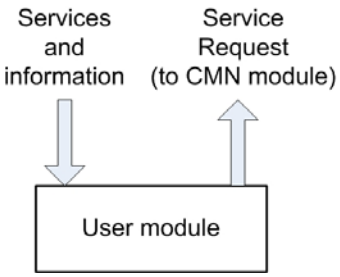
Όπως αναφέρθηκε στο κεφάλαιο 4 κάθε CMN είναι μια κοινή εικονική φυσική διεπαφή που επιτρέπει την πρόσβαση (στους χρήστες) σε ένα θεμελιώδες δίκτυο (DVB-T TS). Προκειμένου να ανιχνευθεί και να επιτραπεί η πρόσβαση στις διαθέσιμες υπηρεσίες που παρέχονται από εκείνο το δίκτυο χρησιμοποιείται μια ενότητα λογισμικού (σχήμα 5.4). Από αυτή την άποψη αυτή η ενότητα εκτελεί δύο λειτουργίες i) ανιχνεύει τις διαθέσιμες υπηρεσίες που παρέχονται από ένα κοινό θεμελιώδες δίκτυο και ii) επιτρέπει την πρόσβαση σε εκείνες τις υπηρεσίες. Η ανίχνευση υπηρεσιών εκτελείται με τη λήψη των πληροφοριών που στέλνονται από τον Εξυπηρετητή Διανομής. Αυτό το είδος πληροφοριών επιτρέπει στην ενότητα CMN να προσδιορίσει μεμονωμένα τις παρεχόμενες υπηρεσίες που στέλνονται από κάθε παροχέα υπηρεσιών στον Εξυπηρετητή Διανομής. Επιπλέον αυτές οι πληροφορίες χρησιμοποιούνται επίσης προκειμένου να επιτραπεί η πρόσβαση στις παρεχόμενες υπηρεσίες.



Σχήμα 5.4: Η γενική αρχιτεκτονική της ενότητας CMN

5.5 Ενότητα Χρήστη

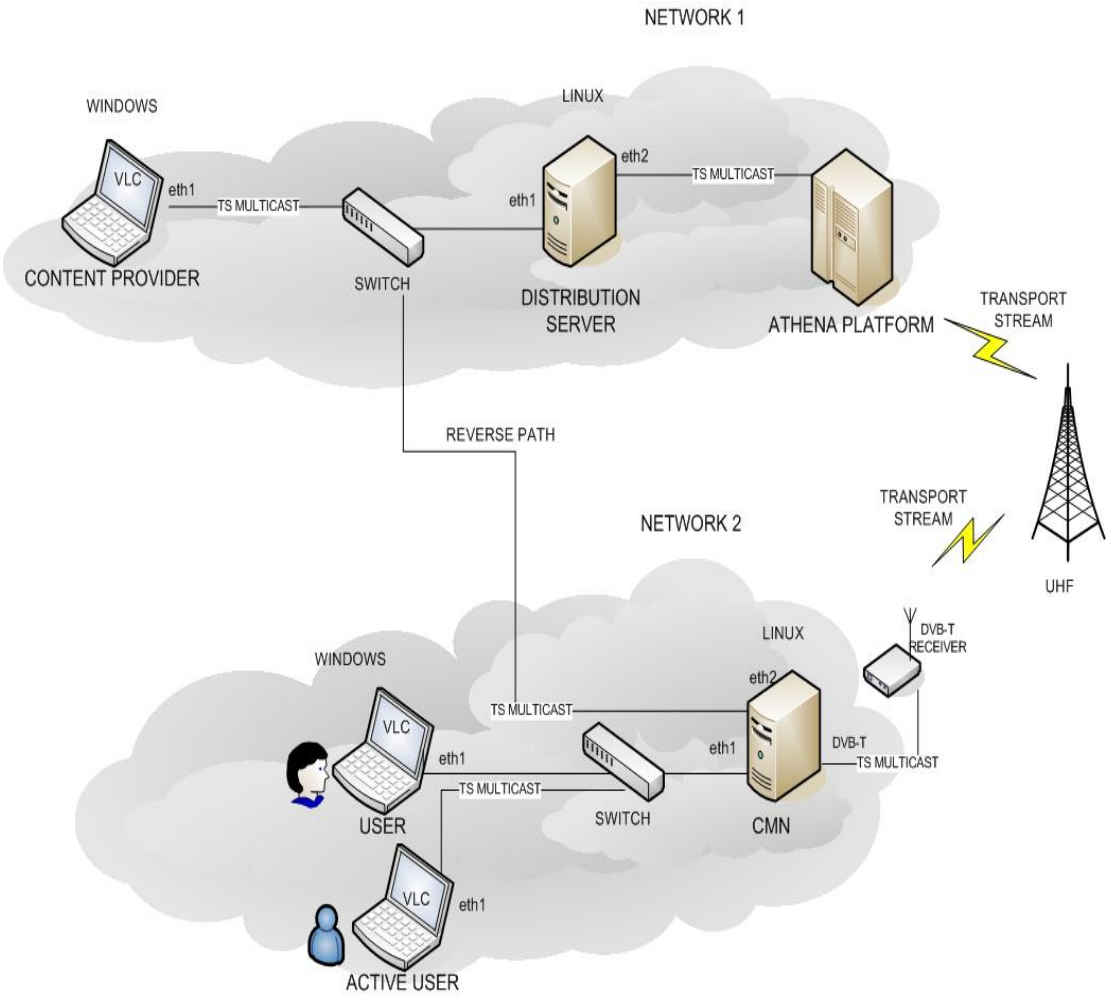
Το τελευταίο συστατικό του Middle-ware που σχεδιάζεται και εφαρμόζεται σε αυτή την εργασία είναι η ενότητα χρήστη Αυτή η ενότητα εγκαθίσταται στον τελικό χρήστη και συνδέει την ενότητα χρήστη με την ενότητα CMN. Επιπλέον η ενότητα χρήστη επιτρέπει στους χρήστες να ενημερωθούν και να έχουν πρόσβαση σε υπηρεσίες έμμεσα από την ενότητα CMN. Από αυτή την άποψη όλες οι πληροφορίες σχετικά με τις διαθέσιμες υπηρεσίες μπορούν να διαβιβαστούν από την ενότητα CMN στην ενότητα χρήστη και διατίθενται στους τελικούς χρήστες. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν από τους τελικούς χρήστες που επιλέγουν και έχουν πρόσβαση στις υπηρεσίες με την αποστολή ενός αιτήματος υπηρεσίας στην ενότητα CMN.



Σχήμα 5.5: Η γενική αρχιτεκτονική της ενότητας χρήστη

6 Διαμόρφωση, εφαρμογή και ολοκλήρωση συστήματος

Μετά από την αρχιτεκτονική δικτύου και middle-ware που περιγράφτηκε στα προηγούμενα κεφάλαια (κεφάλαια 4 και 5) αυτό το κεφάλαιο παρουσιάζει τη διαμόρφωση, την υλοποίηση και την ολοκλήρωση μιας δοκιμής πρωτοτύπου που επιτρέπει τη διαφανή πρόσβαση στις υπηρεσίες μέσω της στοίβας πρωτοκόλλου IP. Συγκεκριμένα αυτό το κεφάλαιο περιγράφει την ολοκλήρωση ενός Middle-ware και ενός συστήματος DVB-T, τη διαμόρφωση του ενσωματωμένου συστήματος και επεξηγεί την εφαρμογή του τελικού συστήματος μέσω μιας ζωντανής δοκιμής.



Σχήμα 6.1 Προτεινόμενη αρχιτεκτονική δικτύου

6.1 Διαμόρφωση και υλοποίηση συστήματος DVB-T

Όπως αναφέρθηκε αυτή η εργασία υιοθετεί μια ήδη προτεινόμενη και επεκταμένη υποδομή συστήματος DVB-T . Από αυτή την άποψη σχεδιάζουμε και επεκτείνουμε ένα τέτοιο σύστημα DVB-T που επιτρέπει την πρόσβαση εκτός από ψηφιακό τηλεοπτικό πρόγραμμα σε υπηρεσίες IP. Το επεκταμένο σύστημα αποτελείται από ένα σημείο εκπομπής και ένα CMN.

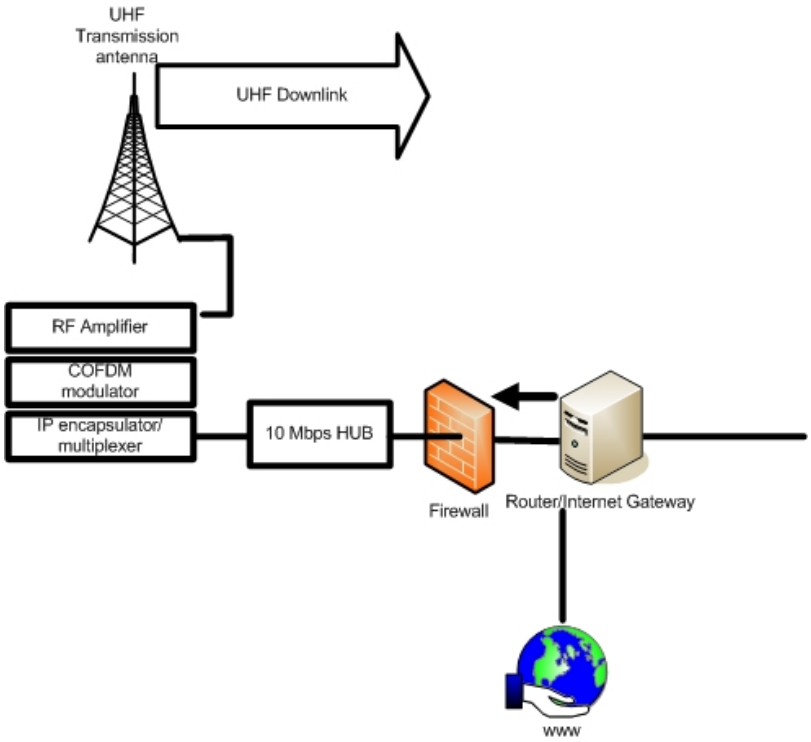
6.1.1 Διαμόρφωση σημείου εκπομπής

Στο σημείο εκπομπής ένας πομπός DVB-T ρυθμίστηκε για να διαβιβάσει στο κανάλι 40 του UHF (ένα 8MHZ κανάλι με κεντρική συχνότητα 626 MHz) χρησιμοποιώντας 16 QAM πλάνο διαμόρφωσης 1/32 διάστημα προστασίας και αναλογία κώδικα 7/8. Αυτές οι παράμετροι αντιστοιχούν σε χρήσιμο bit rate περίπου 21 Mbps.

Επιπλέον και προς την υλοποίηση του σημείου εκπομπής εισάγουμε και ενσωματώνουμε τις ακόλουθες ενότητες λογισμικού και υλικού.

1. Έναν IP-to-DVB encapsulator (που χρησιμοποιεί το πρωτόκολλο MPE) με έναν πολυπλέκτη
2. Έναν διαμορφωτής COFDM
3. Έναν ενισχυτή RF
4. Μια UHF κεραία μετάδοσης
5. Μια σύνδεση (αμφίδρομη) 10 Mbps ((10Mbps hub) (uplink) για την επικοινωνία μεταξύ του CMN με το σημείο εκπομπής
6. Ένα τοπικό LAN (ένα 10 Mbps hub) για την επικοινωνία μεταξύ των παροχέων υπηρεσιών και του σημείου εκπομπής και τη διασύνδεση μεταξύ του σημείου εκπομπής και του δρομολογητή.
7. Ένας μηχανισμός δρομολόγησης που εκτελείται από μια μηχανή Linux (Kernel 2.6.17.3)

Η τελευταία ενότητα (ο μηχανισμός δρομολόγησης) εκτελεί τις ακόλουθες λειτουργίες i) εξασφαλίζει τη μονόδρομη επικοινωνία μεταξύ του CMN και του σημείου εκπομπής και ii)δρα ως πύλη Διαδικτύου. Μια τέτοια τοπολογία δικτύων (σχήμα 6.2) επιτρέπει την ασυμμετρική επικοινωνία μεταξύ του CMN και του σημείου εκπομπής επιτρέποντας στα αιτήματα/επιβεβαιώσεων των IP δεδομένων να στέλνονται στο σημείο εκπομπής μέσω ενός αφοσιωμένου uplink ενώ οι απαντήσεις διαβιβάζονται μέσω του ρεύματος DVB-T .Επιπλέον ένα τέτοιο σύστημα DVB-T επιτρέπει στους παροχείς υπηρεσιών (όπως επίσης και στους τελικούς χρήστες) να διανέμουν περιεχόμενό τους (π.χ. πολλαπλή διανομή βίντεο) μέσω του σημείου εκπομπής

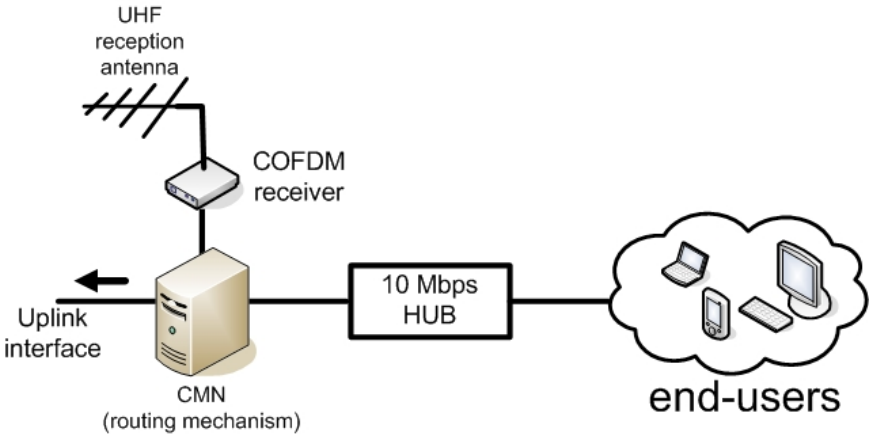


Σχήμα 6.2: Διαμόρφωση του σημείου εκπομπής DVB-T

6.1.2 Διαμόρφωση CMN

Όπως αναφέρθηκε οι χρήστες μπορούν να αποκτήσουν την πρόσβαση στις υπηρεσίες που παρέχονται από ένα θεμελιώδες δίκτυο (DVB-T TS) έμμεσα μέσω των αντιστοιχισμένων CMN τους. Για τη διευκόλυνση της πρόσβασης στις παρεχόμενες υπηρεσίες αυτή η εργασία σχεδιάζει και εφαρμόζει ένα τέτοιο CMN. Η διαμόρφωση ενός υλοποιημένου CMN απεικονίζεται στο σχήμα 6.3. Οι ενότητες που χρησιμοποιήθηκαν για την υλοποίηση του CMN είναι οι ακόλουθες
Ένα 100 Mbps Ethernet NIC που παρέχει την επικοινωνία μεταξύ του CMN και του σημείου εκπομπής (ο δρομολογητής/η πύλη Διαδικτύου) και χρησιμοποιείται για την αποστολή IP αιτημάτων/επιβεβαιώσεων ή/και του περιεχομένου από τους χρήστες στο σημείο εκπομπής.
Ένα 100 Mbps Ethernet NIC που παρέχει την επικοινωνία μεταξύ του CMN και των τελικών χρηστών και χρησιμοποιείται για τη λήψη IP αιτημάτων/επιβεβαιώσεων ή/και του περιεχομένου από τους χρήστες.
Ένας δέκτης DVB-T που χρησιμοποιείται για τη λήψη IP δεδομένων (απαντήσεις IP, πολλαπλή διανομή βίντεο) από το downlink DVB-T
Ενα 10Mbps hub που διασυνδέει το CMN με τους χρήστες
Ένας μηχανισμός δρομολόγησης που εκτελείται από μια μηχανή Linux (το CMN)

Η τελευταία ενότητα (ο μηχανισμός δρομολόγησης) εξασφαλίζει ότι όλη η κίνηση IP που προέρχεται από τους χρήστες διαβιβάζεται στο σημείο εκπομπής μέσω αφοσιωμένου uplink ενώ όλη η κίνηση IP που προορίζεται για τους τελικούς χρήστες (που είναι διαθέσιμοι από το ρεύμα DVB-T) διαβιβάζεται σε αυτούς.



Σχήμα 6. 3: Διαμόρφωση του σημείου εκπομπής DVB-T

6.2 Middle-ware υλοποίηση

Αυτή η εργασία σχεδιάζει και υλοποιεί μια πλατφόρμα Middle-ware που προσφέρει τη διαφανή πρόσβαση (στους τελικούς χρήστες) σε οποιοδήποτε είδος υπηρεσιών για τη στοίβα πρωτοκόλλου IP. Όπως αναφέρθηκε στο κεφάλαιο 5 το Middle-ware που υλοποιήθηκε αποτελείται από τέσσερις ενότητες που είναι η ενότητα Παροχέας Υπηρεσιών, η ενότητα Εξυπηρετητής Διανομής,η ενότητα CMN και η ενότητα Χρήστη.

6.2.1 Υλοποίηση ενότητας παροχέα υπηρεσιών

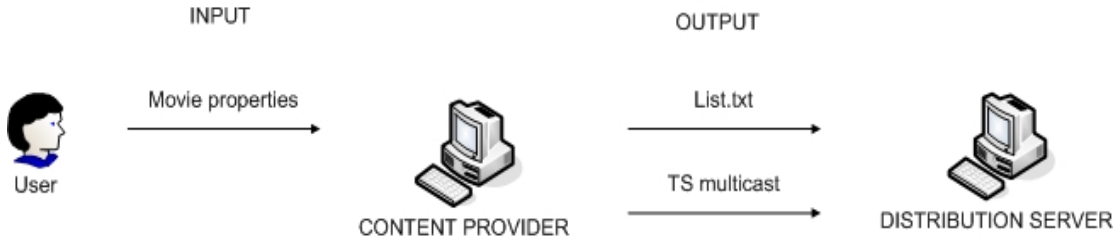
Η ενότητα παροχέας υπηρεσιών επιτρέπει σε έναν παροχέα υπηρεσιών εκτός από τη διανομή του περιεχομένου της να κοινοποιήσει την ύπαρξή της και τις παρεχόμενες υπηρεσίες της. Για την πραγματοποίηση αυτού, κάθε παροχέας υπηρεσιών είναι εξοπλισμένος με το λογισμικό που εκτελεί τις ακόλουθες διαδικασίες:

1. Επιτρέπει σε έναν παροχέα υπηρεσιών να επιλέξει και να περιγράψει το περιεχόμενο που επιθυμεί να διανείμει και
2. Αποστολή του περιεχομένου του παροχέα μαζί με τις πληροφορίες.

Αυτές οι δύο λειτουργίες μπορούν να πραγματοποιηθούν από τον παροχέα υπηρεσιών μέσω ενός Graphic User Interface (GUI) υλοποιημένο σε JAVA .

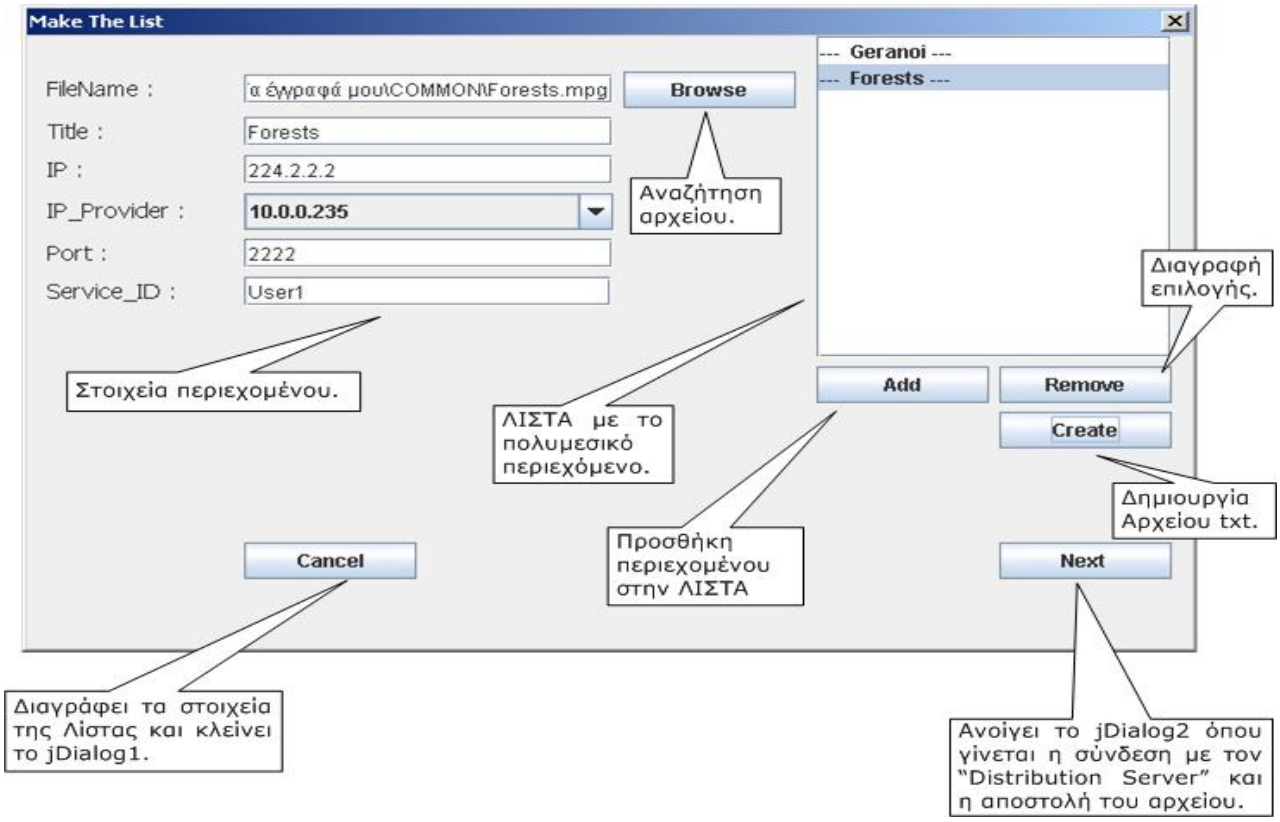
Η διεπαφή “Content Provider” αντιπροσωπεύει τον παροχέα υπηρεσιών. Σκοπός της είναι να συλλέξει όλες τις απαραίτητες πληροφορίες για την κατασκευή ενός αρχείου που θα στέλνεται στον “Distribution Server” (Εξυπηρετητή Διανομής) και θα περιέχει τα στοιχεία από τις πολυμεσικές υπηρεσίες που πρόκειται να στείλει ο παροχέας. Η διαδικασία αυτή γίνεται στο jDialog1. Στην συνέχεια στο jDialog2 πραγματοποιείται η σύνδεση με τον “ Distribution Server” μέσω socket όπου στέλνεται και το αρχείο. Τελευταίο βήμα είναι η αποστολή των πολυμεσικών

υπηρεσιών ως Transport Stream μέσω multicast με την χρήση του προγράμματος VLC.



Σχήμα 6.4. Γενική περιγραφή διεπαφής Content Provider

Η διεπαφή χωρίζεται σε 3 κλάσεις. **Provider, Connection, Stream**. Η κλάση Provider.java είναι η κύρια κλάση όπου γίνεται η δημιουργία του αρχείου. Επίσης περιέχει και το γραφικό κομμάτι της διεπαφής. Οι κλάσεις Conection.java και Stream.java είναι βοηθητικές. Η πρώτη ασχολείται με την σύνδεση με τον "Distribution Server" και την αποστολή αρχείου, ενώ η δεύτερη με την αποστολή του αρχείου ως Transport Stream μέσω multicast με την βοήθεια του προγράμματος VLC. Παρακάτω ακολουθεί μια περιγραφή του γραφικού περιεχομένου της διεπαφής.



Σχήμα 6.5 Περιγραφή του jDialog1 Content Provider

Κουμπί **"Browse"**→ Ανοίγει ένα dialog και επιλέγεται το επιθυμητό αρχείο βίντεο. Εμφανίζονται συγκεκριμένοι τύποι αρχείων που έχουν οριστεί. Επιστρέφει ολόκληρο το path του αρχείου και το τοποθετεί σε textfield δίπλα από label "Filename". Το path του αρχείου θα το χρησιμοποίηση του πρόγραμμα VLC ώστε να μπορεί να το στείλει multicast.

Κουμπί **“Cancel”**→ Διαγράφει τα στοιχεία της Λίστας και κλείνει το jDialog1.

Κουμπί **”Add”**→ Παίρνει όλες τις πληροφορίες από τα textfields και τις τοποθετεί σε ένα HashMap (δυναμική λίστα αποθήκευσης). Ο τίλτος του στοιχείου προστίθεται στην Λίστα και στην συνέχεια όλα τα πεδία καθαρίζονται ώστε να υπάρχει η δυνατότητα για καινούργιο στοιχείο. Σε περίπτωση που χρειάζεται να αλλάξει κάποια παράμετρος επιλέγεται το επιθυμητό αρχείο, εμφανίζονται οι πληροφορίες στα textfields, γίνονται οι αλλαγές, κουμπί “Remove” και τέλος κουμπί “Add”.

Κουμπί **“Remove”**→ Διαγράφει το επιλεγμένο αρχείο από την Λίστα.

Κουμπί **”Create”**→ Διαβάζει το HashMap και τοποθετεί τις πληροφορίες σε ένα αρχείο List.txt. Σε κάθε textfield προστίθεται και το ανάλογο xml tag. Το xml tag θα χρειαστεί για να φτιαχτεί το τελικό xml αρχείο στον “Distribution Server”.

Κουμπί **“Next”**→ Ανοίγει το jDialog2 όπου γίνεται η σύνδεση με τον “Distribution Server” , η αποστολή του αρχείου με την λίστα και η μετάδοση του TS μέσω multicast.

FileName: Πλήρες όνομα του αρχείου. Συμπληρώνεται αυτόματα

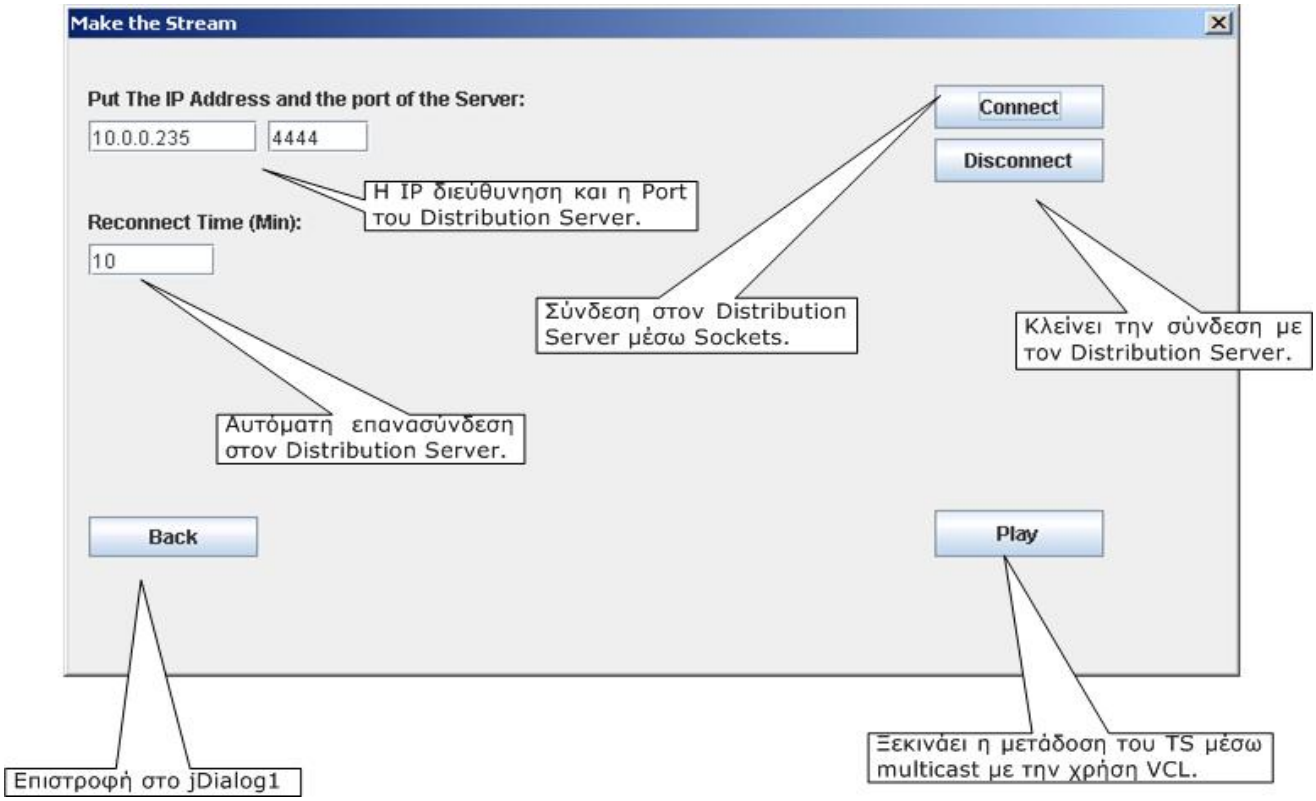
Title: Τίτλος αρχείου. Το συμπληρώνει ο χρήστης.

IP: Διεύθυνση που θα γίνει multicast. Πρέπει να αλλάζει ο χρήστης την διεύθυνση για κάθε νέο αρχείο ώστε να ξεχωρίζουν αφού έχουν το ίδιο port.

IP_Provider: Διεύθυνση του παροχέα υπηρεσιών. Φορτώνεται αυτόματα σε dropdown list. Χρειάζεται για το routing στον CMN για να αναγνωρίζει τον παροχέα σε περίπτωση δύο ίδιων διευθύνσεων multicast.

Port: Πόρτα διεύθυνσης. Η τιμή παραμένει σταθερή. Για κάθε provider αντιστοιχεί μια πόρτα. Αρχικά συμπληρώνεται από τον χρήστη.

Service_ID: Όνομα του παροχέα υπηρεσιών. Παραμένει σταθερό. Αρχικά συμπληρώνεται από τον χρήστη.



Σχήμα 6.6 Περιγραφή του jDialog2 Content Provider

Κουμπί **"Back"**→ Επιστροφή στο jDialog1.

Κουμπί **"Connect"**→ Ανοίγει socket για την δημιουργία σύνδεσης με τον "Distribution Server". Δέχεται ορίσματα το IP, Port του Server.

Κουμπί **"Disconnect"**→ Κλείνει το socket της σύνδεσης με τον "Distribution Server".

Κουμπί **"Play"**→ Ανοίγει το πρόγραμμα VLC, διαβάζει το αρχείο που έχει φτιαχτεί και στέλνει ένα Transport Stream μέσω multicast για κάθε στοιχείο της Λίστας ανάλογα με τις παραμέτρους.

Το **Reconnect Time** δηλώνει το χρόνο που θα συνδέεται αυτόματα στον "Distribution Server". Καθορίζεται από τον χρήστη.

Τα πεδία **IP Address,Port** καθορίζονται από τον χρήστη και αναφέρονται στην σύνδεση με τον "Distribution Server".

Όλη η κίνηση που προέρχεται από έναν παροχέα υπηρεσιών στέλνεται στο σημείο εκπομπής όπου ένας IP encapsulator την προσαρμόζει χρησιμοποιώντας το PID που ορίζεται για τον συγκεκριμένο multicast. Αυτή η κίνηση περιέχει τις ακόλουθες υπηρεσίες i) την πολλαπλής διανομής κίνηση των υπηρεσιών βίντεο που παρέχει ο παροχέας υπηρεσιών, και ii) τον παροχέα υπηρεσιών και υπηρεσιών πληροφοριών με μορφή πολλαπλής διανομής.

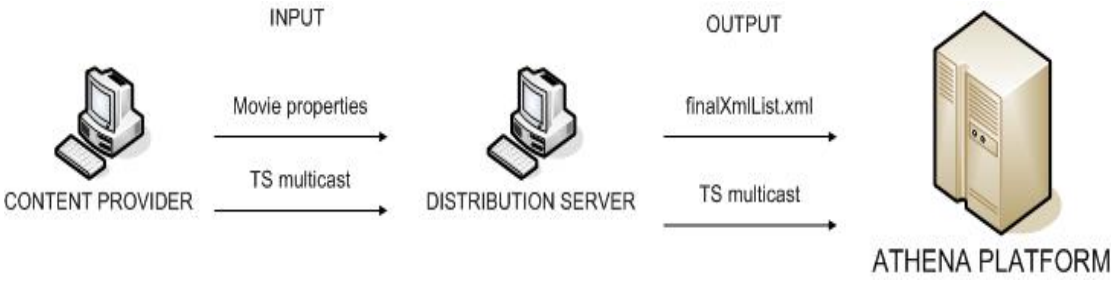
6.2.2 Υλοποίηση ενότητας Εξυπηρετητή Διανομής

Η ενότητα Εξυπηρετητής Διανομής επιτρέπει την διαχείριση όλων των παροχέων υπηρεσιών καθώς και την κοινοποίηση - ενοποίηση όλων των διαθέσιμων υπηρεσιών. Για την πραγματοποίηση αυτού, κάθε παροχέας υπηρεσιών είναι εξοπλισμένος με το λογισμικό που εκτελεί τις ακόλουθες διαδικασίες:

1. Διαχείριση των παροχέων υπηρεσιών.
2. Προώθηση των υπηρεσιών στο σημείο εκπομπής
3. Ενοποίηση των πληροφοριών των υπηρεσιών

Αυτές οι τρεις λειτουργίες μπορούν να πραγματοποιηθούν από τον Εξυπηρετητή Διανπμής μέσω ενός Graphic User Interface (GUI) υλοποιημένο σε JAVA .

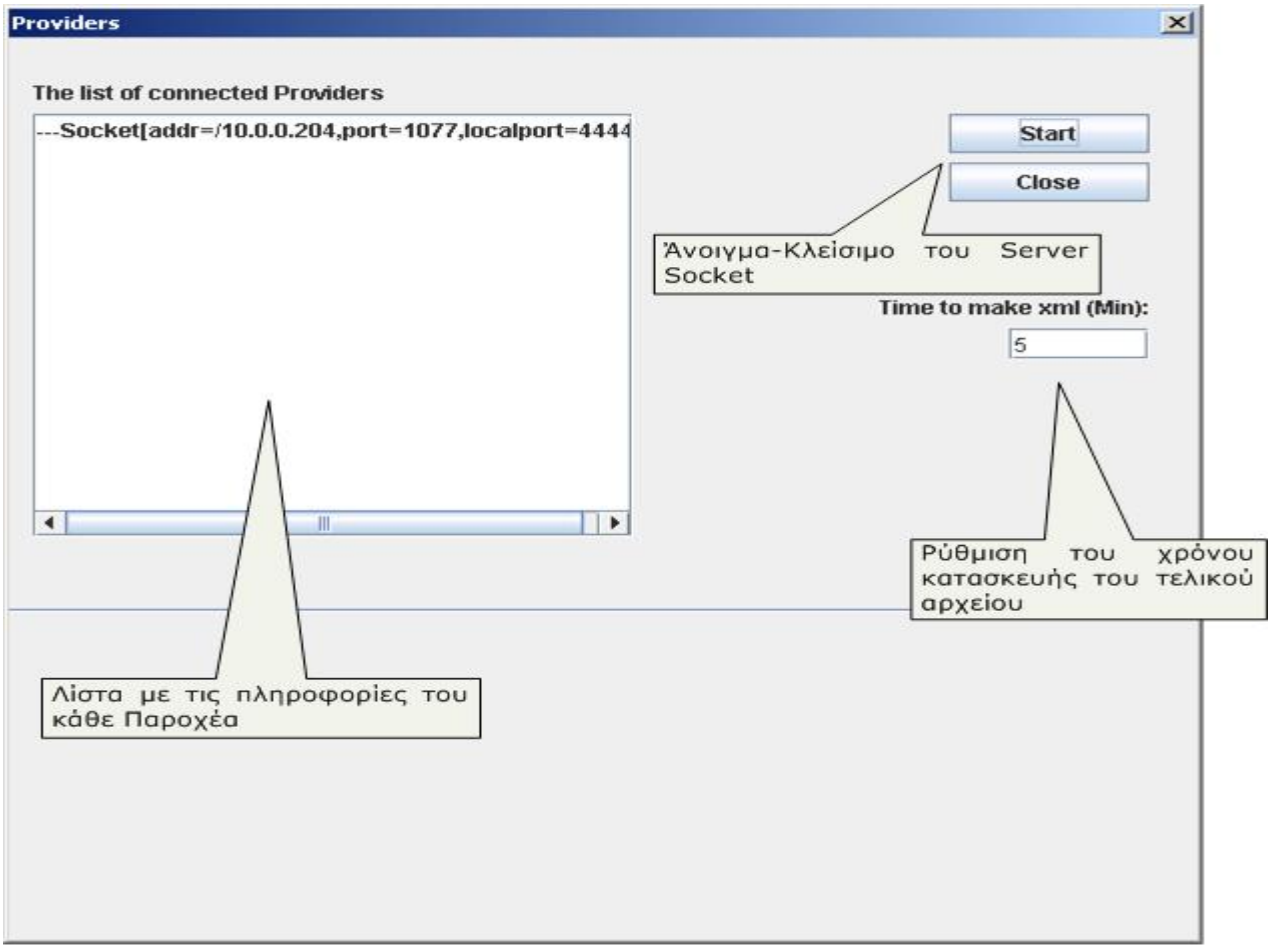
Η διεπαφή "Distribution Server" αντιπροσωπεύει τον Εξυπηρετητή Διανομής. Σκοπός της είναι η διαχείριση στις συνδέσεις των Content Providers. Ο κάθε παροχέας υπηρεσιών εκτός από την μετάδοση του πολυμεσικού περιεχομένου ,στέλνει στον Distribution Server ένα αρχείο που περιέχει πληροφορίες για το πολυμεσικό περιεχόμενο που πρόκειται να μεταδώσει. Ο Distribution Server συλλέγει το κάθε αρχείο και τα ενοποιεί σε ένα αρχείο xml το οποίο στην συνέχεια στέλνεται στην πλατφόρμα της ψηφιακής τηλεόρασης. Επίσης κάνει προώθηση τα ip πακέτα στην πλατφόρμα ATHENA. Στην πραγματικότητα είναι ένας ενδιάμεσος σταθμός μεταξύ των Providers και της πλατφόρμας DVB-T. Βοηθάει στην καλύτερη διαχείριση των συνδέσεων.



Σχήμα 6.7. Γενική περιγραφή διεπαφής Distribution Server

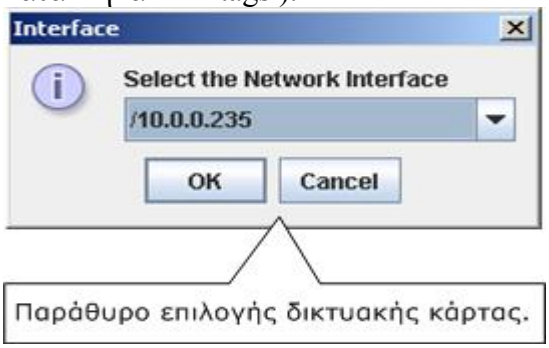
Η διεπαφή περιέχει 5 κλάσεις. Η **DServer** δημιουργεί το γραφικό περιβάλλον και δημιουργεί τις συνδέσεις πάνω στις οποίες θα συνδεθούν οι Providers. Η κλάση **ProviderConnection** διαχειρίζεται τις συνδέσεις των Providers. Επίσης λαμβάνει το αρχείο που στέλνει ο κάθε Provider και το αποθηκεύει σε ένα συνολικό αρχείο. Η κλάση **MakeXml** τοποθετεί τα xml tags που απαιτούνται στο συνολικό αρχείο το οποίο το αντιγράφει σε ένα τελικό αρχείο. Στην συνέχεια το συνολικό αρχείο ξαναφτιάχνεται από την αρχή, δηλαδή δέχεται τις ενημερώσεις των Providers. Το τελικό αρχείο xml αφού διαβαστεί ξεκινάει η διαδικασία προώθησης των εισερχόμενων ip πακέτων ανάλογα με τις πληροφορίες που περιέχει. Σε κάθε νέο ενημερωμένο αρχείο που έρχεται ελέγχεται αν υπάρχουν αλλαγές στα στοιχεία. Διαγράφει τα παλιά και σταματάει την προώθηση ενώ προσθέτει τα καινούργια. Η κλάση **ForwardingDS** ανάλογα με τα ορίσματα που δέχεται διαχειρίζεται τις προωθήσεις. Αυτό γίνεται μέσω linux με την βοήθεια του smcroute. Οι λειτουργίες

του smcroute είναι άνοιγμα – κλείσιμο του smcroute, άνοιγμα κλείσιμο συγκεκριμένης προώθησης ανάλογα τα ορίσματα (εισερχόμενη δικτυακή κάρτα, Ip Provider, Ip multicast, εξερχόμενη δικτυακή κάρτα). Τέλος η κλάση **SendXml** στέλνει μέσω multicast σε συγκεκριμένη ip διεύθυνση το τελικό αρχείο xml σαν udp. Παρακάτω ακολουθεί η περιγραφή της διεπαφής.



Σχήμα 6.8. Διαχείριση των Providers

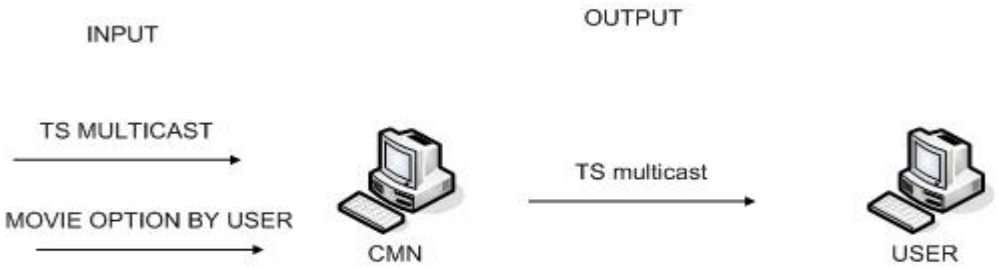
Στο σχήμα 1 περιγράφει την γραφική διεπαφή του Distribution Server. Πριν το άνοιγμα του Server Socket , εμφανίζεται παράθυρο όπου επιλέγεται η επιθυμητή δικτυακή κάρτα (Σχήμα 2). Οι πληροφορίες από κάθε Provider εμφανίζονται στην Λίστα. Επίσης ο χειριστης της διεπαφής πρέπει να ορίσει τον χρόνο που θα κατασκευάζεται το τελικό αρχείο (συλλέγει το προσωρινό αρχείο, βάζει τα κατάλληλα xml tags).



Σχήμα 6.9. Επιλογή της δικτυακής κάρτας

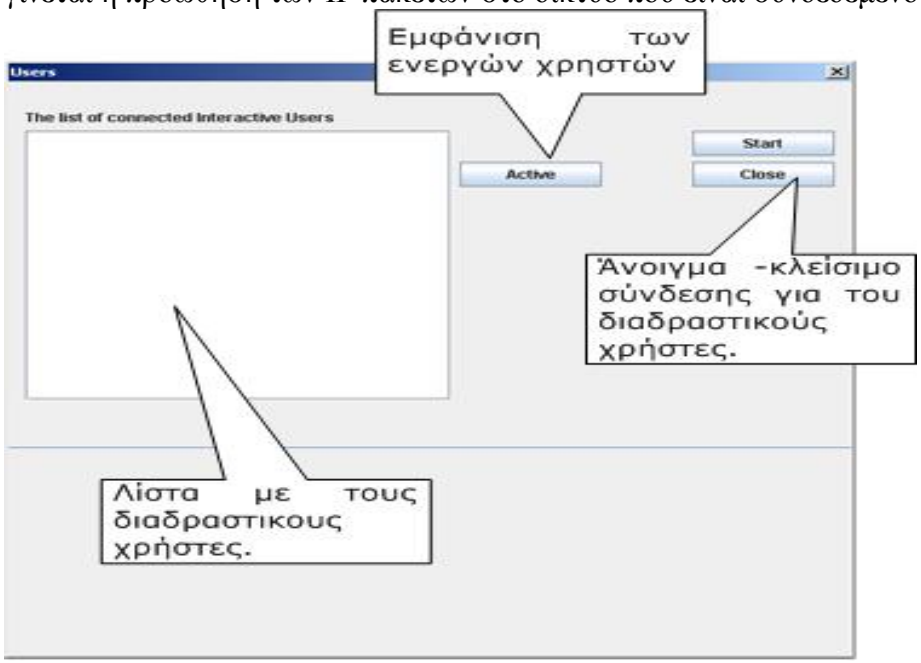
6.2.3 Υλοποίηση ενότητας CMN

Η διεπαφή CMN ασχολείται με την διασύνδεση των Users με τους Providers. Αναλαμβάνει μέσω του dvb-t receiver να διαβάσει τα TS πακέτα που εκπέμπει η πλατφόρμα ATHENA. Στην συνέχεια ανοίγει socket για να συνδεθούν οι χρήστες. Σε κάθε σύνδεση στέλνει και την συνολική λίστα με όλα τα πολυμεσικά περιεχόμενα. Επιπλέον όταν δεχτεί κάποια επιλογή από κάποιον χρήστη αναλαμβάνει τα προωθήσει τα κατάλληλα IP πακέτα στο δίκτυο σύνδεσης με τους Users. Τέλος γίνεται έλεγχος για δυναμική διαχείριση εύρους ζώνης του δικτύου.

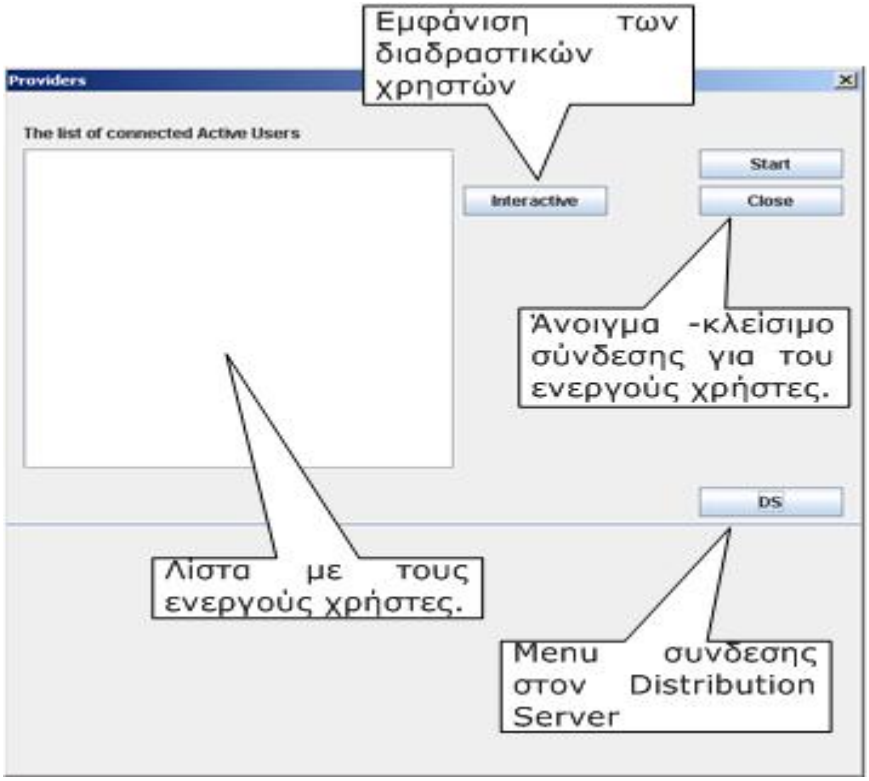


Σχήμα 6.10. Γενική περιγραφή της διεπαφής CMN

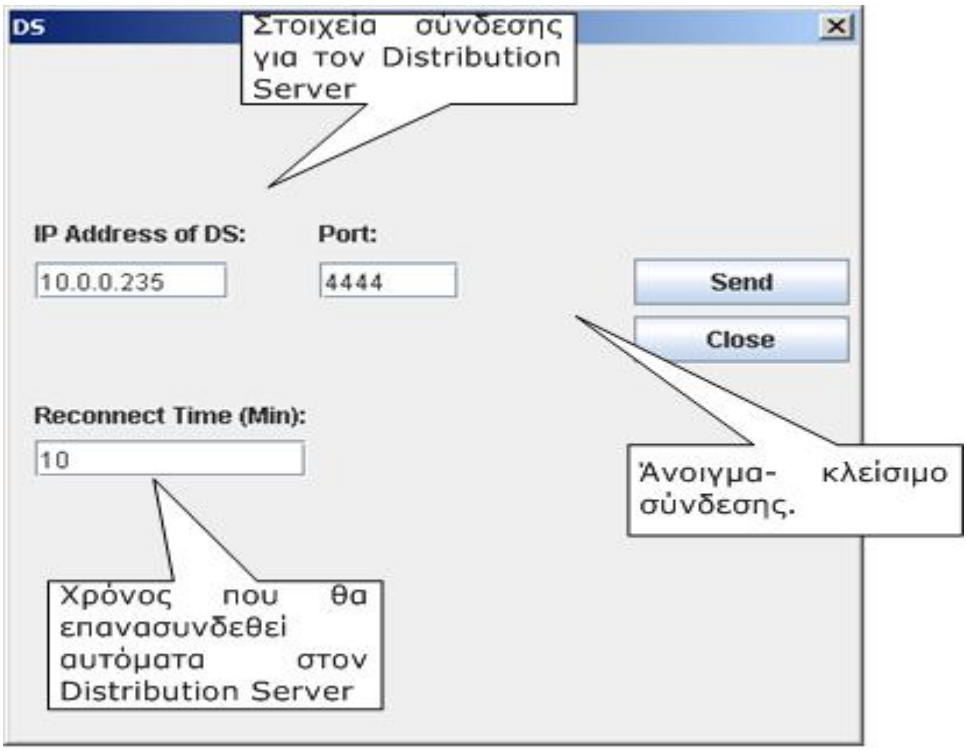
Η διεπαφή CMN χωρίζεται σε 8 κλάσεις. Είναι η πιο σημαντική διεπαφή ολόκληρης της πτυχιακής. Η κλάση **CMN** περιέχει τα γραφικά κομμάτια της διεπαφής. Η κλάση **AUserConnection** διαχειρίζεται τις συνδέσεις των ενεργών χρηστών. Η κλάση **IUserConnection** διαχειρίζεται τις συνδέσεις των διαδραστικών χρηστών. Η κλάση **ReceiveXml** αναλαμβάνει να συνδεθεί στην κάρτα dvb-t receiver ώστε να διαβάσει το αρχείο xml με το πολυμεσικό περιεχόμενο. Η κλάση **SendList** έχει ως σκοπό την αποστολή του αρχείου στον Distribution Server με τα περιεχόμενα των ενεργών χρηστών που έχουν συνδεθεί στον CMN. Η κλάση **Routing** διαχειρίζεται τις αιτήσεις των διαδραστικών χρηστών και αποθηκεύει τις πληροφορίες στην κλάση **RequestList**. Με την βοήθεια την κλάσης **Forwarding** γίνεται η προώθηση των IP πακέτων στο δίκτυο που είναι συνδεδεμένοι οι χρήστες.



Σχήμα 6.11. Περιγραφή του menu συνδέσεων των διαδραστικών χρηστών



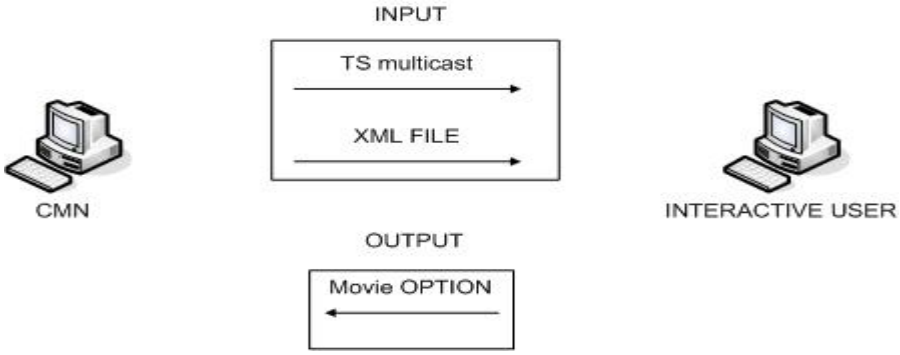
Σχήμα 6.12. Περιγραφή του menu συνδέσεων των ενεργών χρηστών



Σχήμα 6.13. Περιγραφή του menu σύνδεσης με τον Distribution Server

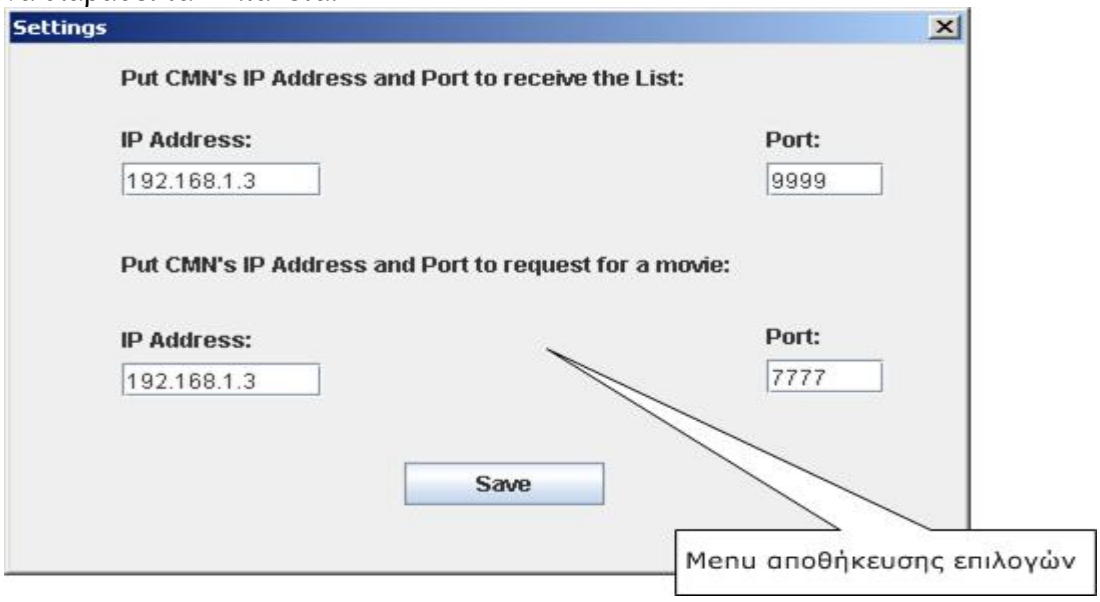
6.2.4 Υλοποίηση ενότητας Χρήστης

Η διεπαφή "Interactive User" αντιπροσωπεύει τον διαδραστικό χρήστη. Σκοπός της είναι να παρουσιάσει στον τελικό χρήστη τις διαθέσιμες πολυμεσικές υπηρεσίες. Ο χρήστης συνδέεται στον αντιστοιχισμένο CMN και αποδέχεται το αρχείο με το πολυμεσικό περιεχόμενο. Στην συνέχεια επιλέγει το επιθυμητό στοιχείο του περιεχομένου και στέλνει την επιλογή του πίσω στον CMN. Ο CMN αναλαμβάνει να προωθήσει τα κατάλληλα ip πακέτα στο δίκτυο. Ο χρήστης με την βοήθεια του προγράμματος VLC διαβάζει τα πακέτα και τα οποία παρουσιάζονται στην οθόνη του.

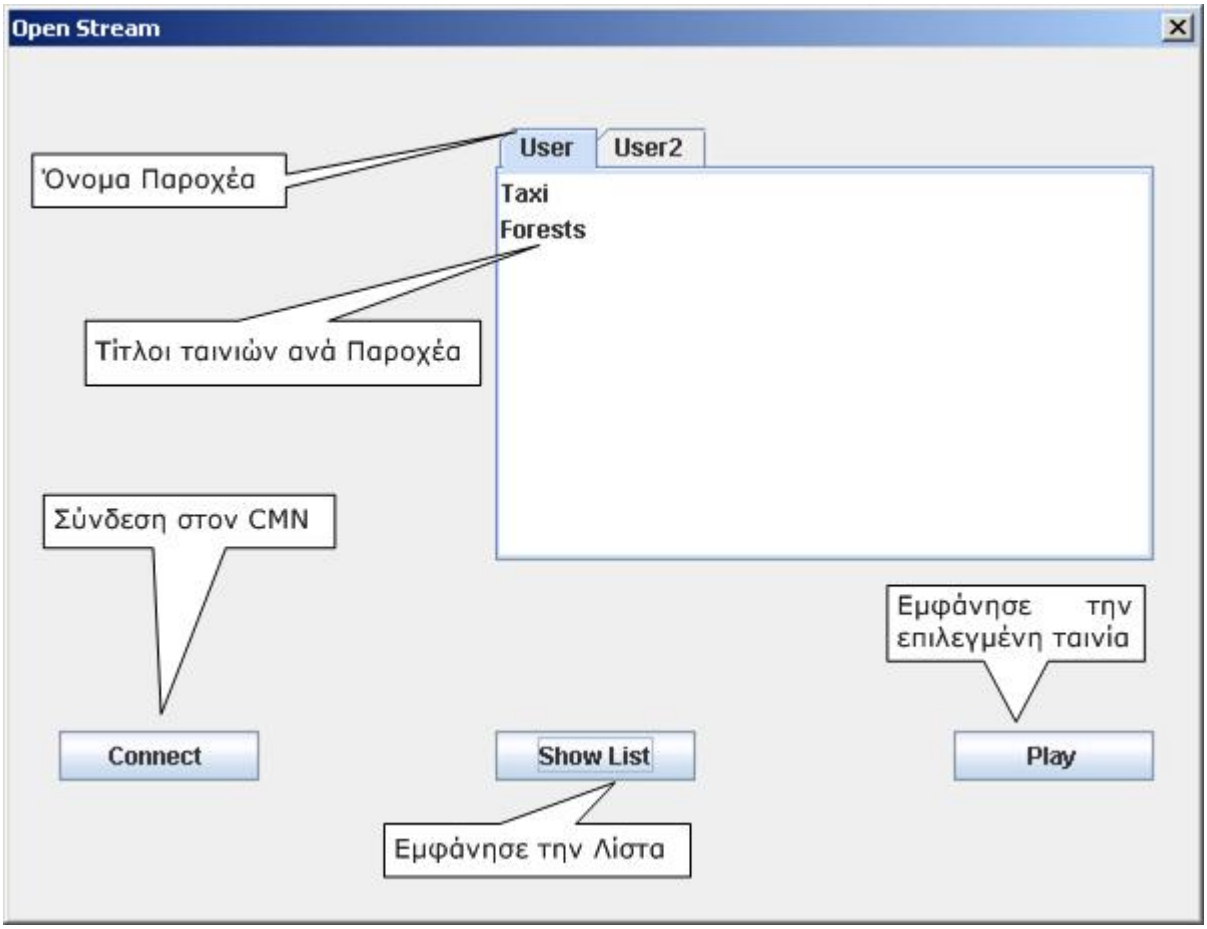


Σχήμα 6.14. Γενική περιγραφή της διεπαφής User

Η διεπαφή "User" χωρίζεται σε 3 κλάσεις. Η **InteractiveUser** ασχολείται με το γραφικό κομμάτι της διεπαφής. Επίσης διαχειρίζεται τις ρυθμίσεις δικτύου δηλαδή τις IP διευθύνσεις του CMN ώστε να μπορεί αυτόματα να συνδέεται. Οι ρυθμίσεις αποθηκεύονται σε ένα αρχείο. Επίσης αποδέχεται το xml αρχείο και το διαβάζει ώστε να εμφανιστούν τα διαθέσιμα πολυμεσικά περιεχόμενα. Η κλάση **Movies** διαχειρίζεται τα στοιχεία του περιεχομένου. Η κλάση **Play** αναλαμβάνει να στείλει την επιλογή του στον CMN και την συνέχεια με την βοήθεια του προγράμματος VLC να διαβάσει τα IP πακέτα.



Σχήμα 6.15. Περιγραφή του μενού Settings του User



Σχήμα 6.16. Περιγραφή του jDialog1 του User

Κουμπί "**Connect"**→ Συνδέεται μέσω sockets στον αντιστοιχισμένο CMN.

Κουμπί "**Show List"**→ Εμφανίζει την λίστα με το πολυμεσικό περιεχόμενο. Για κάθε νέο παροχέα δημιουργείται και ένα νέο tab. Κάθε tab περιέχει όλες τις διαθέσιμες ταινίες του παροχέα.

Κουμπί "**Play"**→ Στέλνει την επιλογή του περιεχομένου στον CMN και στην συνέχεια ανοίγει το πρόγραμμα VLC και διαβάζει τα πακέτα που προωθεί ο CMN.

Παράρτημα

Ακολουθεί ο κώδικας από κάθε διεπαφή.

Παροχέας Υπηρεσιών

```
 package ContentProvider;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.swing.DefaultListModel;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

/*
```

* Provider.java

```
 * Created on 7 November 2006
 */

/**
 *
 * @author  DIMITRIS PAPADOGIANNIS
 */

public class Provider extends JFrame{

    /** Creates new form Provider */
    public Provider() {
        initComponents();

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        jDialog1 = new javax.swing.JDialog();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
```

```
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jScrollPane1 = new javax.swing.JScrollPane();
        jList1 = new javax.swing.JList();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jButton5 = new javax.swing.JButton();
        jButton6 = new javax.swing.JButton();
        jLabel7 = new javax.swing.JLabel();
        jComboBox2 = new javax.swing.JComboBox();
        jTextField9 = new javax.swing.JTextField();
        jDialog2 = new javax.swing.JDialog();
        jButton7 = new javax.swing.JButton();
        jTextField6 = new javax.swing.JTextField();
        jLabel6 = new javax.swing.JLabel();
        jButton9 = new javax.swing.JButton();
        jButton10 = new javax.swing.JButton();
        jButton8 = new javax.swing.JButton();
        jTextField7 = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();
        jTextField8 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        jMenuItem2 = new javax.swing.JMenuItem();


jDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
        jDialog1.setTitle("Make The List");
        jDialog1.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        jDialog1.setModal(true);
        jDialog1.setResizable(false);
        jDialog1.getAccessibleContext().setAccessibleName("");
        jDialog1.getAccessibleContext().setAccessibleDescription("");
        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel1.setText("FileName : ");

        jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel2.setText("Title : ");

        jLabel3.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel3.setText("IP : ");

        jLabel4.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel4.setText("Port : ");

        jLabel5.setFont(new java.awt.Font("Tahoma", 0, 14));
```

```
        jLabel5.setText("Service_ID : ");

        jTextField1.setToolTipText("File's Path");

        jTextField2.setToolTipText("File's Title");

        jTextField3.setToolTipText("Multicast IP Address");

        jTextField4.setToolTipText("Multicast Port");


jList1.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        jList1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jList1MouseClicked(evt);
            }
        });

        jScrollPane1.setViewportView(jList1);

        jButton1.setText("Browse");
        jButton1.setToolTipText("Selects the video file");
        jButton1.setMaximumSize(new java.awt.Dimension(70, 20));
        jButton1.setMinimumSize(new java.awt.Dimension(70, 20));
        jButton1.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton1.getAccessibleContext().setAccessibleName("");
        jButton1.getAccessibleContext().setAccessibleDescription("");

        jButton2.setText("Add");
        jButton2.setToolTipText("Adds a file to the list");
        jButton2.setMaximumSize(new java.awt.Dimension(70, 20));
        jButton2.setMinimumSize(new java.awt.Dimension(70, 20));
        jButton2.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton2.getAccessibleContext().setAccessibleName("");
        jButton2.getAccessibleContext().setAccessibleDescription("");

        jButton3.setText("Create");
        jButton3.setToolTipText("Creates the xmlList ");
        jButton3.setMaximumSize(new java.awt.Dimension(70, 20));
        jButton3.setMinimumSize(new java.awt.Dimension(70, 20));
        jButton3.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
```

```
        });

        jButton3.getAccessibleContext().setAccessibleName("");
        jButton3.getAccessibleContext().setAccessibleDescription("");

        jButton4.setText("Remove");
        jButton4.setToolTipText("Removes a file from the list");
        jButton4.setMaximumSize(new java.awt.Dimension(70, 20));
        jButton4.setMinimumSize(new java.awt.Dimension(70, 20));
        jButton4.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton4.getAccessibleContext().setAccessibleName("");
        jButton4.getAccessibleContext().setAccessibleDescription("");

        jButton5.setText("Cancel");
        jButton5.setToolTipText("Cancel all");
        jButton5.setMaximumSize(new java.awt.Dimension(70, 20));
        jButton5.setMinimumSize(new java.awt.Dimension(70, 20));
        jButton5.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jButton6.setText("Next ");
        jButton6.setToolTipText("Goes to the Connection");
        jButton6.setMaximumSize(new java.awt.Dimension(70, 20));
        jButton6.setMinimumSize(new java.awt.Dimension(70, 20));
        jButton6.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });

        jLabel7.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel7.setText("IP_Provider :");

        jComboBox2.setToolTipText("Provider's IP Address");

        jTextField9.setToolTipText("Provider's Name");

        org.jdesktop.layout.GroupLayout jDialog1Layout = new
org.jdesktop.layout.GroupLayout(jDialog1.getContentPane());
        jDialog1.getContentPane().setLayout(jDialog1Layout);
        jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog1Layout.createSequentialGroup()
                .addContainerGap()
```

```
.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                .add(jDialog1Layout.createSequentialGroup()

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG, false)
                        .add(org.jdesktop.layout.GroupLayout.LEADING,    jLabel1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                        .add(org.jdesktop.layout.GroupLayout.LEADING,    jLabel2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .add(org.jdesktop.layout.GroupLayout.LEADING,    jLabel3,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .add(jLabel7)
                    .add(jLabel4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 102, Short.MAX_VALUE)
                    .add(jLabel5,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 102, Short.MAX_VALUE))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog1Layout.createSequentialGroup()
                        .add(jTextField9,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 213, Short.MAX_VALUE)
                        .add(1, 1, 1))
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,   jTextField4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 214, Short.MAX_VALUE)
                    .add(jComboBox2, 0, 214, Short.MAX_VALUE)
                    .add(jTextField1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 214, Short.MAX_VALUE)
                    .add(jTextField2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 214, Short.MAX_VALUE)
                    .add(jTextField3,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 214, Short.MAX_VALUE)
                    .add(jButton5,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG)
                    .add(jDialog1Layout.createSequentialGroup()
                        .add(jButton1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(12, 12, 12)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                            .add(jDialog1Layout.createSequentialGroup()
```

```
                                .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .add(jButton4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                            .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 206, Short.MAX_VALUE)))
                        .add(jButton6,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .add(org.jdesktop.layout.GroupLayout.TRAILING,          jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addContainerGap())
        );

        jDialog1Layout.linkSize(new java.awt.Component[] {jButton1, jButton2,
jButton3, jButton4}, org.jdesktop.layout.GroupLayout.HORIZONTAL);

        jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog1Layout.createSequentialGroup()

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                .add(jDialog1Layout.createSequentialGroup()
                    .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                219,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(jButton4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(jDialog1Layout.createSequentialGroup()
                    .add(24, 24, 24)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jLabel1)
                        .add(jTextField1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
```

```
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(jButton1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jLabel2)
                        .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jLabel3)
                        .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jLabel7)
                        .add(jComboBox2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jTextField4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(jLabel4))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                        .add(jLabel5)
                        .add(jTextField9,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))))
                .add(64, 64, 64)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                    .add(jButton6,  org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
```

```
                .add(jButton5,  org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(47, Short.MAX_VALUE))
        );

        jDialog1Layout.linkSize(new  java.awt.Component[]  {jButton1,  jButton2,
jButton3, jButton4}, org.jdesktop.layout.GroupLayout.VERTICAL);


jDialog2.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
        jDialog2.setTitle("Make the Stream");
        jDialog2.setModal(true);
        jDialog2.setResizable(false);
        jButton7.setText("Connect");
        jButton7.setToolTipText("Connects to the Server");
        jButton7.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

        jTextField6.setText("10.0.0.235");
        jTextField6.setToolTipText("IP Address");

        jLabel6.setText("Put The IP Address and the port of the Server:");

        jButton9.setText("Play ");
        jButton9.setToolTipText("Sends multicast");
        jButton9.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton9ActionPerformed(evt);
            }
        });

        jButton10.setText("Back");
        jButton10.setToolTipText("Goes to previous menu");
        jButton10.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton10.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton10ActionPerformed(evt);
            }
        });

        jButton8.setText("Disconnect");
        jButton8.setToolTipText("Closes the Connection");
        jButton8.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton8ActionPerformed(evt);
            }
        });

        jTextField7.setText("4444");
        jTextField7.setToolTipText("Port");
```

```
        jLabel8.setText("Reconnect Time (Min):");

        jTextField8.setText("10");
        jTextField8.setToolTipText("Minutes to Reconnect to Server for Update");

        org.jdesktop.layout.GroupLayout jDialog2Layout = new
org.jdesktop.layout.GroupLayout(jDialog2.getContentPane());
        jDialog2.getContentPane().setLayout(jDialog2Layout);
        jDialog2Layout.setHorizontalGroup(

jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog2Layout.createSequentialGroup()
                .addContainerGap()

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG, false)
                    .add(jTextField8,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    75,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(jDialog2Layout.createSequentialGroup()

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG, false)
                            .add(org.jdesktop.layout.GroupLayout.LEADING,       jLabel6,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .add(org.jdesktop.layout.GroupLayout.LEADING,
jDialog2Layout.createSequentialGroup()

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG, false)
                                    .add(jTextField6)
                                    .add(jButton10,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE))

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                                .add(jTextField7,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    60,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                            .add(org.jdesktop.layout.GroupLayout.LEADING, jLabel8))
                        .add(242, 242, 242)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG)
                                .add(jButton8,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                .add(org.jdesktop.layout.GroupLayout.LEADING,   jButton7,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                            .add(jButton9,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                    100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
```

```
                .add(78, 78, 78))))
        );
        jDialog2Layout.setVerticalGroup(

jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog2Layout.createSequentialGroup()
                .add(28, 28, 28)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog2Layout.createSequentialGroup()
                        .add(jButton7,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
25, Short.MAX_VALUE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jButton8,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
25, Short.MAX_VALUE)
                        .add(231, 231, 231))
                    .add(jDialog2Layout.createSequentialGroup()
                        .add(262, 262, 262)
                        .add(jButton9,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
25, Short.MAX_VALUE))
                    .add(jDialog2Layout.createSequentialGroup()
                        .add(jLabel6)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                            .add(jTextField6,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(jTextField7,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                        .add(34, 34, 34)
                        .add(jLabel8)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jTextField8,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(148, 148, 148)
                        .add(jButton10,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,                                 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .add(71, 71, 71))
        );

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Content Provider");
        jTextField5.setEditable(false);
        jTextField5.setText("Status:");

        jMenu1.setText("File");
        jMenuItem1.setText("Make Stream");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem1);

        jMenu1.add(jSeparator1);

        jMenuItem2.setText("Exit");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem2);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jTextField5, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 792,
Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap(532, Short.MAX_VALUE)
                .add(jTextField5, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        );
        setBounds(50, 10, 800, 600);
    }// </editor-fold>


    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton8--> It changes the value of the "connectFlag" to "false" so the
connection
    // between ContentProvider and Distribution Server will close.
    // The class "Connection" uses this variable.

        connectFlag=false;
        jTextField5.setText("Status: The Connection is closed...");
    }
```

```
    private void jList1MouseClicked(java.awt.event.MouseEvent evt) {

    // When i click in the jList1 all the information appears in the jTextFields.
    // Takes the selection of the jList1 , puts the selection in the Iterator
    //and returns  all the information from the hashmap.

        int counter =-1;
        Iterator listIter = lista.listIterator();
        int selection=jList1.getSelectedIndex();
        while (counter != selection){
            tmpHash = (HashMap)listIter.next();
            jTextField1.setText(""+tmpHash.get(1));
            jTextField2.setText(""+tmpHash.get(2));
            jTextField3.setText(""+tmpHash.get(3));
            jTextField4.setText(""+tmpHash.get(4));
            jComboBox2.setSelectedItem(tmpHash.get(6));
            jTextField9.setText(""+tmpHash.get(5));
            counter++;
        }
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Connect" --> It makes a new Thread which makes a new class
(Connection) which
    // connects the Provider with the Distribution Server.

        Thread thrdConnect = new Thread(new Connection());
        thrdConnect.start();
    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Play" --> It sends the video.

        Iterator listIter1 = lista.listIterator();    // ListIterator is the pointer of the
ArrayList -->lista.
        while(listIter1.hasNext()){         // Plays all the movies.
            tmpHash = (HashMap)listIter1.next();      // Puts the next ListIterator's
element in tmpHash as HashMap.

            // Multithreading! It needs 3 inputs Path,Ip,Port.
            try {
                // Makes a new class "Stream" which plays the files witl vlc program.A
new thread is created for each file.
                Thread thrdMulti = new Thread(new
Stream((String)tmpHash.get(1),(String)tmpHash.get(3),(String)tmpHash.get(4))
);
                thrdMulti.start();
                jTextField5.setText("Status: The Video Stream is sending...");
            }catch(NullPointerException e){
              jTextField5.setText("Status: "+  e.getMessage());
            }
        }
    }
```

```
    private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Back" --> Goes to jDialog1.

        jTextField5.setText("Status: ");    //It clears the Status Bar.
        jDialog2.setVisible(false);
        jDialog1.setVisible(true);
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Next" --> Goes to jDialog2.

        jTextField5.setText("Status: Set up the Connection...");
        jDialog1.setVisible(false);
        jDialog2.setBounds(160,160,500,500);
        jDialog2.pack();
        jDialog2.setVisible(true);
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton " Cancel" --> Deletes everything.

        jTextField5.setText("Status: ");
        lista.clear();
        listModel.clear();
        jList1.removeAll();
        jDialog1.dispose();
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Remove" --> Removes an element both from the jList1 and  the
ArrayList --> lista.

        if ( jList1.getSelectedIndex()== -1){      // If you dont select a value from
jList1 an error message will show up.
            JOptionPane.showMessageDialog(jDialog1,"Please  select  a  movie  from
the list.","List error",JOptionPane.ERROR_MESSAGE);}
        else {
            int tempSelection=jList1.getSelectedIndex();
            listModel.removeElementAt(tempSelection);
            lista.remove(tempSelection);
            jTextField5.setText("Status: The selection was erased...");
            }
    }


    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Create" --> Makes the xmlList and the stream.

        String tempStr = ""; // xmlList uses the tempStr.
        jTextField5.setText("Status: ");
        Iterator listIter = lista.listIterator();
```

```
        if ( listIter.hasNext()== false)
    // If no movie has been added then a showMessage will show up.

          JOptionPane.showMessageDialog(jDialog1,"Please add a movie in the
list.","Warning",JOptionPane.WARNING_MESSAGE);
        else {
          while(listIter.hasNext()){              // Looks for added movies.
            tmpHash = (HashMap)listIter.next();       // Uses the listIterator as
HashMap

            // tempStr is used for the xmlList and collects all the tags-values in
one string.
            tempStr=tempStr + "\n\t<Movie>";
            tempStr=tempStr + "\n\t\t<Title>" + tmpHash.get(2) + "</Title>";
            tempStr=tempStr + "\n\t\t<IP>" + tmpHash.get(3) + "</IP>";
            tempStr=tempStr + "\n\t\t<IPPro>" + tmpHash.get(6) + "</IPPro>"
;
            tempStr=tempStr + "\n\t\t<Port>" + tmpHash.get(4) + "</Port>";
            tempStr=tempStr  +  "\n\t\t<ServiceID>"  +  tmpHash.get(5)  +
"</ServiceID>" ;
            tempStr=tempStr + "\n\t</Movie>\n";
          }

          // Makes the List
          try {
            File outputFile = new File("..\\videoList.txt");         // Makes a txtfile.
            FileWriter out = new FileWriter(outputFile);
 // The file can be written by  "out" which is FileWriter.

            // Writes the tempStr to the videoList txtfile.The opening and closing
tags
            //of the xml file will be added in DistributionServer.

            out.write(tempStr);
            out.close();
            jTextField5.setText("Status: The List is ready...");
          }catch (IOException ex){
            jTextField5.setText("Status: " + ex.getMessage());
          }
        }
    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {

    // JMenuItem "Exit" -->  The application closes.

        setVisible(false);
        dispose();
    }


    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "ADD" --> Adds elements to JList1.

        jTextField5.setText("Status: ");
```

```
        Map map = new HashMap();          // Puts values in the Map.
        map.put(1,jTextField1.getText());
        map.put(2,jTextField2.getText());
        map.put(3,jTextField3.getText());
        map.put(4,jTextField4.getText());
        map.put(5,jTextField9.getText());
        map.put(6,jComboBox2.getSelectedItem());


        lista.add(map);
 // Puts the Map in the ArrayList -->lista which controls the Map's elements dynamically.

        // Puts the value Title in the jList1
        listModel.addElement("---  " + jTextField2.getText()+ " ---");
   // Adds elements to the ListModel.
        jList1.setModel(listModel);                                    // Sets the ListModel in the jList1.

        jTextField1.setText("");                                       // All the jTextFields are cleared.
        jTextField2.setText("");
        jTextField3.setText("");
        jTextField5.setText("Status: The Video file was added...");
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Browse" --> selects the video file and returns the path.

        // It makes the filter for specific files.
        javax.swing.filechooser.FileFilter filter = new javax.swing.filechooser.FileFilter() {
            public boolean accept(File f) {
                return f.isDirectory()          // Returns the Directory path if the file ends with these endings.
                || f.getName().toLowerCase().endsWith(".avi")
                || f.getName().toLowerCase().endsWith(".mpg")
                || f.getName().toLowerCase().endsWith(".mpeg")
                || f.getName().toLowerCase().endsWith(".wmv")
                || f.getName().toLowerCase().endsWith(".mov");
            }
            public String getDescription() {          // In this dialog "Movie Files" is displayed.
                return "Movie Files";
            }
        };

        jFileChooser1.setFileFilter(filter);
        jFileChooser1.showOpenDialog(this);
        fileName=""+jFileChooser1.getSelectedFile();          // Returns the file's path.
        jTextField1.setText(fileName);
    }
```

```
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

    // JMenuItem1 "Make Stream" --> It shows the jDialog1.

        // Finds all the active network interfaces and display them in the
jComboBox2.
        // The Content's Provider IP is needed for the xmlList.
        try {
            tmp1                                                                =
InetAddress.getAllByName(InetAddress.getLocalHost().getHostName());
            for (int i = 0; i < tmp1.length; i++) {
                String s;
                s=tmp1[i].toString();
                String[] t=null;
                t = s.split("/");
                jComboBox2.addItem(t[1]);
            }
        } catch (UnknownHostException ex) {
            jTextField5.setText("Status: " + ex.getMessage());
        }
        jTextField5.setText("Status: Make the video list...");
        jDialog1.setBounds(160,160,500,500);
        jDialog1.pack();
        jDialog1.setVisible(true);
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Provider().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton10;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JComboBox jComboBox2;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JDialog jDialog2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
```

```
    private javax.swing.JList jList1;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JSeparator jSeparator1;
    protected javax.swing.JTextField jTextField1;
    protected javax.swing.JTextField jTextField2;
    protected javax.swing.JTextField jTextField3;
    protected javax.swing.JTextField jTextField4;
    protected static javax.swing.JTextField jTextField5;
    protected static javax.swing.JTextField jTextField6;
    protected static javax.swing.JTextField jTextField7;
    protected static javax.swing.JTextField jTextField8;
    protected javax.swing.JTextField jTextField9;
    // End of variables declaration
    private String fileName;    // JDialog's "Browse" result.
    // First puts the values in the listModel and then the listModel  in the jList1.
    private DefaultListModel listModel= new DefaultListModel();
    private Thread thrdMulti;         // This thread is used for the multicast.
    private java.util.List lista = new ArrayList();
    // Dynamic administration of the movies.
    final JFileChooser jFileChooser1 = new JFileChooser();    // It is used for Path.
    private HashMap tmpHash;
    protected static Thread thrdConnect; // This thread  is used for the connection.
    protected static boolean connectFlag=true;
     // while connectFlag is true the connection between DS,CP is alive.
    private InetAddress[] tmp1;
}
```

```
package ContentProvider;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
/*
```

* Connection.java

```
 * Created on 5 January 2007.
 */

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
```

```
public class Connection implements Runnable {

    /**
     * Creates a new instance of Connection
     *
     *This class creates a connection between Content Provider and Distribution Server in order to send      the
     *file "videoList.txt" which contains all the information about the multimedia content.After the file arrives
     *the connection closes and after specific time(user determines it) it connects again automatically in order to send
     *the updated file.If there aren't any changes in the file it will send the previous.
     *
     */
    public Connection() {
    }

    public void run() {

        while ( Provider.connectFlag==true){
          try {
             // New Socket(Ip,Port).
             sConnection = new Socket(Provider.jTextField6.getText(),Integer.parseInt(Provider.jTextField7.getText()));
             Provider.jTextField5.setText("Status: The Connection was established...");

          } catch (UnknownHostException ex) {
             Provider.jTextField5.setText("Status: Error...1" + ex.getMessage());
             break;
          } catch (IOException ex) {
             Provider.jTextField5.setText("Status: Error...2" + ex.getMessage());
             break;
          }
          // sendfile
          File xmlFile = new File ("..\\videoList.txt");                // Finds the file.
          byte [] tempArray  = new byte [(int)xmlFile.length()+1];
     // Makes a temp array with the file's length.
          FileInputStream fis = null;

          try {
             fis = new FileInputStream(xmlFile);
                  // Puts the file into InputStream.
             BufferedInputStream bis = new BufferedInputStream(fis);
     // Puts the InputStream into Buffer.
             bis.read(tempArray,0,tempArray.length);
                // Reads the file.
             OutputStream os = null;
             os = sConnection.getOutputStream();
           // Sends the file.
             os.write(tempArray,0,tempArray.length);
            // Makes the file.
             os.flush();
             os.close();                                          // Closes the file.
             Provider.jTextField5.setText("Status: The List was sent...");
```

```
            // Agrument in milliseconds.
            //1 sec =1000 ms *60 ms = 60000 ms = 1 min
            //Determine the reconnect time.It will send the file again.


Provider.thrdConnect.sleep(Integer.parseInt(Provider.jTextField8.getText())*600
00);

            if(Provider.connectFlag==false){

               sConnection.close();
               Provider.jTextField5.setText("Status: Connection is closed...");
            }
         } catch (FileNotFoundException ex) {
            Provider.jTextField5.setText("Status: Error...3" + ex.getMessage());

         } catch (IOException ex) {
             Provider.jTextField5.setText("Status: Error...4" + ex.getMessage());

         } catch (InterruptedException ex) {
             Provider.jTextField5.setText("Status: Error...5" + ex.getMessage());
         }
      }
   }
   private Socket sConnection;
}
```

```
package ContentProvider;
/*
```

* Stream.java

```
 *
 * Created on 5 December 2006
 */
import java.io.IOException;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 *
 *This class creates a new process in order to open the VLC program.VLC will
send the multimedia content as Trasport stream via multicast.
 *The arguments which VLC needs are:The path of the file, The IP address and
the Port. ttl is 10.Warning:the vlc program must be intalled
 *in  specific directory."C:\Program Files\VideoLAN\VLC"
 */
```

```
public class Stream  implements Runnable {

   Stream(String a ,String b ,String c){

      //Connects the arguments with local Strings

      vlcPath=a;
      vlcIp=b;
      vlcPort=c;
   }

   public void run(){

      ProcessBuilder tmp = new ProcessBuilder( );   //New Process

      try {
         //tmp.command(program,arguments).   program = VLC (path)   , VLC
arguments to play the video(vlc command line).

         tmp.command("C:\\Program Files\\VideoLAN\\VLC\\vlc"  , " "  +  vlcPath
+
               "\"                                                              \"--
sout=#duplicate{dst=display,dst=std{access=udp,mux=ts,dst=\"\""  +  vlcIp  +
"\"\":\"\"" + vlcPort+ "\"\"}} --tll 10");
         tmp.start();

      } catch(IOException ex) {
         Provider.jTextField5.setText("Status: " + ex.getMessage());
         }
   }
   private String vlcPath , vlcIp , vlcPort;
}
```

Εξυπηρετητής Διανομής

```
package Server;

import java.awt.Component;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.ServerSocket;
import java.net.SocketException;
import java.net.UnknownHostException;
```

```
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Iterator;
import javax.swing.JOptionPane;
/*
```

* DServer.java

```
 *
 * Created on 14 January 2007.
 */

/**
 *
 * @author  DIMITRIS PAPADOGIANNIS
 */
public class DServer extends javax.swing.JFrame {

    /**
     * Creates new form DServer
     *
     *This class administrates the incoming connections.Collects the Providers' files
and makes a final xml file
     *with all the movies.Also searchs for the availiable network inteface to
establish the ip forwarding in order to send the
     *ip traffic into the Amber.
     */
    public DServer() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */

    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        jDialog1 = new javax.swing.JDialog();
        jScrollPane1 = new javax.swing.JScrollPane();
        jList1 = new javax.swing.JList();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jSeparator2 = new javax.swing.JSeparator();
        jLabel2 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField1 = new javax.swing.JTextField();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();


jDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
```

```
        jDialog1.setTitle("Providers");
        jDialog1.setResizable(false);
        jScrollPane1.setPreferredSize(new java.awt.Dimension(260, 140));
        jList1.setRequestFocusEnabled(false);
        jScrollPane1.setViewportView(jList1);

        jButton2.setText("Start");
        jButton2.setToolTipText("Open The Connection");
        jButton2.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton3.setText("Close");
        jButton3.setToolTipText("Close The Connection");
        jButton3.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jLabel1.setText("The list of connected Providers");

        jLabel2.setText("Time to make xml (Min):");

        jTextField2.setToolTipText("Determines the time which the program will
make the final xml file.");

        org.jdesktop.layout.GroupLayout jDialog1Layout = new
org.jdesktop.layout.GroupLayout(jDialog1.getContentPane());
        jDialog1.getContentPane().setLayout(jDialog1Layout);
        jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog1Layout.createSequentialGroup()
                .addContainerGap()

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog1Layout.createSequentialGroup()
                        .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 300,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
153, Short.MAX_VALUE)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                                .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
```

```
                        .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .add(jLabel2)
                    .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 70,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .add(jLabel1))
            .add(24, 24, 24))
        .add(jSeparator2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 600,
Short.MAX_VALUE)
    );
    jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(jDialog1Layout.createSequentialGroup()
            .add(26, 26, 26)
            .add(jLabel1)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 270,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jDialog1Layout.createSequentialGroup()
                    .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(59, 59, 59)
                    .add(jLabel2)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
            .add(46, 46, 46)
            .add(jSeparator2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
10, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(228, Short.MAX_VALUE))
    );

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Distribution Server");
    jTextField1.setEditable(false);
    jTextField1.setText("Status:");

    jMenu1.setText(" Administration");
    jMenuItem1.setText("Providers");
    jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem1ActionPerformed(evt);
        }
```

```
        });

        jMenu1.add(jMenuItem1);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, jTextField1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 892, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap(632, Short.MAX_VALUE)
                .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        );
        setBounds(50, 10, 900, 700);
    }// </editor-fold>

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton " Close" --> It closes the ServerSocket of Providers.

        flagP=false;
        try {
            serverP.close();
        } catch (IOException ex) {
            jTextField1.setText("Status: "+ ex.getMessage());
        }
        //Stops forwarding by giving argument 0 in smcroute.

        Thread thrdForward =new Thread(new
ForwardingDS((String)null,(String)null,0));
        thrdForward.start();

    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Start" --> It opens a connection for the Providers.
    // Firstly finds all the available interfaces and then shows them in a
JOptionPane in which user selects the inteface.
    // Two Threads will start.The first for the Providers and the second for the xml
file.

        jTextField1.setText("Status: Searching for availiable interfaces...");
        flagP=true;
        Enumeration e,f;
            try {
```

```
            int ni = 0;
            e = NetworkInterface.getNetworkInterfaces();    //Finds the avaliable
network Interfaces.
            while (e.hasMoreElements()) {
               f = ((NetworkInterface)e.nextElement()).getInetAddresses();
//Takes the InetAddress of the Interfaces.
               while (f.hasMoreElements()) {
                  ((InetAddress)f.nextElement()).getHostName();
                  ++ni;
               }
            }
            in = new InetAddress[ni];
            ni = 0;
            e = NetworkInterface.getNetworkInterfaces();
            while (e.hasMoreElements()) {
               f = ((NetworkInterface)e.nextElement()).getInetAddresses();
               while (f.hasMoreElements())
                  in[ni++] = (InetAddress)f.nextElement();
            }
            for (int i = 0; i < in.length; i++) {
                 in[i].getCanonicalHostName();
            }
         } catch (SocketException ex) {
           jTextField1.setText("Status: "+ ex.getMessage());
         }

      //Opens a JOptionPane Dialog.
      InetAddress value=(InetAddress)JOptionPane.showInputDialog
      (null,"Select the Network
Interface","Interface",JOptionPane.INFORMATION_MESSAGE,null,in,in[0]);

      //Starts a thread for the connection with Providers.
      Thread threadP = new Thread(new ProviderConnection(jList1));     //The
jList1 argument used for change the jList1 from another class.
      threadP.start();

      //Starts a thread for the xml file.Finds the file,puts the closing tags and
copy into another file.
      Thread threadRead= new Thread(new MakeXml());
      threadRead.start();
   }


   private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

   // JMenuItem " Providers" --> opens the jDialog1 and operates the
connections of the Providers.

      jDialog1.setBounds(30,30,600,600);
      jDialog1.setVisible(true);

      //Starts the smcroute.
      Thread thrdForward =new Thread(new
ForwardingDS((String)null,(String)null,1));
      thrdForward.start();
   }
```

```
    protected static void copyFile(File inputFile,File outputFile) throws Exception {

        //Copies a file into another one.

        FileReader in = new FileReader(inputFile);
        FileWriter out = new FileWriter(outputFile);
        int c;

        while ((c = in.read()) != -1)
          out.write(c);

        in.close();
        out.close();
        }

    protected static void readList(){

        //Reads the finalXmlList.xml.If a movie doesnt exist anymore
        //this method deletes it from the list.Otherwise it starts forwarding.

      Iterator listIter = lista.listIterator();
      while(listIter.hasNext()){
        tmpHash = (HashMap)listIter.next();
        if (tmpHash.get(3)=="false"){
            Thread thrdForward =new Thread(new
ForwardingDS((String)tmpHash.get(1),(String)tmpHash.get(2),-1));
            thrdForward.start();
            lista.remove(tmpHash);
        }else{
            Thread thrdForward =new Thread(new
ForwardingDS((String)tmpHash.get(1),(String)tmpHash.get(2),2));
            thrdForward.start();
        }
    }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new DServer().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JList jList1;
```

```
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JSeparator jSeparator2;
    protected static javax.swing.JTextField jTextField1;
    protected static javax.swing.JTextField jTextField2;
    // End of variables declaration
    protected static boolean flagP=true;    // Uses for the connection between
Distribution Server and Content Provider.
    protected static boolean flagSend=true;
    protected static Thread threadRead;
    protected static ServerSocket serverP = null;   //The ServerSocket of the
Distribution Server.
    protected static InetAddress value;
    private InetAddress[] in;       //Table with all the availiable IP Address.
    protected static java.util.List lista = new ArrayList();     // Dynamic
administration of the movies.
    private static HashMap tmpHash;
}
```

```
*
```

* ProviderConnection.java

```
 *
 * Created on 19 January 2007.
 *
 */

package Server;

import java.io.BufferedOutputStream;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import javax.swing.DefaultListModel;
import javax.swing.JList;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 *
```

```
 *This class creates connections between Server and Provider(Server side).Also
receives the files.
 */
public class ProviderConnection implements Runnable{              //Makes the
connection between Server and Provider.

   /**
    * Creates a new instance of ProviderConnection
    */
   public ProviderConnection(JList tmp) {                        //The argument jList
uses for administration of the list.
      jListP=tmp;
   }

   public void run(){
      try{
         DServer.serverP = new ServerSocket(4444,10,DServer.value);
//Opens a new ServerSocket(Port,Max providers,interface)
      }catch(IOException e){
          DServer.jTextField1.setText("Status: "+ e.getMessage());
      }
      while (DServer.flagP==true) {
         DServer.jTextField1.setText("Status: Waiting...");
         Socket providerSocket = null;
         try {
            providerSocket = DServer.serverP.accept();               //Accepts the
Provider.

            //Shows the properties of Provider in the jList1.
            if (providerSocket!=null){
               DServer.jTextField1.setText("Status: Accepted connection : " +
providerSocket);
               listModelSP.addElement("---" + providerSocket);
               jListP.setModel(listModelSP);
            }
            else
               break;

            //Makes a file in which all the Proiders' files will be copy.
            //Writes the initial tags.
            BufferedWriter out = new BufferedWriter(new
FileWriter("../xmlList.xml"));
            out.write("<?xml version=\"1.0\" encoding=\"ISO-8859-
1\"?>\n<Catalog>\n");
            out.close();
         } catch (IOException ex) {
         DServer.jTextField1.setText("Status: "+ ex.getMessage());
         }

         int filesize=6022386; // filesize temporary hardcoded
         int bytesRead = 0;
         int current = 0;

         // Receives Providers' file and put it in the xmlList.xml file.
         byte [] tempArray  = new byte [filesize];
         InputStream is = null;
```

```
          try {
             is = providerSocket.getInputStream();   //Reads the inputStream
             FileOutputStream fos = null;
             fos = new FileOutputStream("../xmlList.xml",true);

             BufferedOutputStream bos = new BufferedOutputStream(fos);

             bytesRead = is.read(tempArray,0,tempArray.length);
             current = bytesRead;

             while (bytesRead > -1) {                              //Writes into the
file.
                bytesRead = is.read(tempArray, current, (tempArray.length-
current));
                current += bytesRead;
                bos.write(tempArray, 0 , current);
             }
             bos.close();

          } catch (FileNotFoundException ex) {
           DServer.jTextField1.setText("Status: "+ ex.getMessage()); ;
          } catch (IOException ex) {
           DServer.jTextField1.setText("Status: "+ ex.getMessage());
          }
       }
    }
    private DefaultListModel listModelSP= new DefaultListModel();
       //Name listModel  Server for the Providers.
    private JList jListP;                                        // jlist for the Providers.
    private BufferedWriter out;
}
```

---

*

* MakeXml.java

```
*
* Created on 8 February 2007.
*
*/

package Server;

import java.io.BufferedInputStream;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStream;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */

public class MakeXml implements Runnable{

    /**
     * Creates a new instance of MakeXml
     *
     *This class makes the xml file by puting the opening and the closing xml tags
into the file.
     *Also copies the file into another file in order to use it again. The new file is
read and
     *put their contents in an ArrayList.After that the ArrayList is read and start
the forwarding
     */
    public MakeXml() {
    }

    public void run() {
        while(DServer.flagP=true){

            try {
                //Takes the temp file by specific time in minutes 60000milliseconds =
1Min

DServer.threadRead.sleep(Integer.parseInt(DServer.jTextField2.getText())*6000
0);
            } catch (InterruptedException ex) {
                DServer.jTextField1.setText("Status: "+ ex.getMessage());
            }
            // Closes the xml temp file.
            try {
                BufferedWriter out = new BufferedWriter(new
FileWriter("../xmlList.xml",true));
                out.write("\n</Catalog>");
                out.close();
            }catch(IOException e){
                 DServer.jTextField1.setText("Status: "+ e.getMessage());
                }
            //Copies the temp file into a final file.This file is taken by the Users.
            try {
```

```
            DServer.copyFile(new File("../xmlList.xml"),new
File("../finalXmlList.xml"));

          } catch (Exception ex) {
              DServer.jTextField1.setText("Status: "+ ex.getMessage());
          }
           Thread sendXml= new Thread(new SendXml());
           sendXml.start();

          //Makes the new temp File
          try {
              BufferedWriter out = new BufferedWriter(new
FileWriter("../xmlList.xml"));
              out.write("<?xml version=\"1.0\" encoding=\"ISO-8859-
1\"?>\n<Catalog>\n");
              out.close();
          }catch(IOException e){
              DServer.jTextField1.setText("Status: "+ e.getMessage());
              }
      }
      //The following code reads a xml file.Finds the values by the appropriate
tags.
      //Reads every tag and in every movie starts a new thread in order to start
routing.
       try {

          DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
          DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
          Document doc = docBuilder.parse (new File("../finalXmlList.xml"));

          // normalize text representation
          doc.getDocumentElement ().normalize ();
              doc.getDocumentElement().getNodeName();


          NodeList listOfPersons = doc.getElementsByTagName("Movie");
          int totalPersons = listOfPersons.getLength();

          movies = new String[20][2];
          int i = 0;

          for(int s=0; s<listOfPersons.getLength() ; s++){


              Node firstPersonNode = listOfPersons.item(s);
              if(firstPersonNode.getNodeType() == Node.ELEMENT_NODE){


                  Map map = new HashMap();

                  Element firstPersonElement = (Element)firstPersonNode;

                  //Gets the elements by specific tag name.
                  NodeList titleList =
firstPersonElement.getElementsByTagName("Title");
                  Element titleElement = (Element)titleList.item(0);
```

```
                NodeList textTitleList = titleElement.getChildNodes();
                    ((Node)textTitleList.item(0)).getNodeValue().trim();


                NodeList ipList = firstPersonElement.getElementsByTagName("IP");
                Element ipElement = (Element)ipList.item(0);

                NodeList textIpList = ipElement.getChildNodes();
                    ((Node)textIpList.item(0)).getNodeValue().trim();


                NodeList portList =
firstPersonElement.getElementsByTagName("Port");
                Element portElement = (Element)portList.item(0);

                NodeList textPortList = portElement.getChildNodes();
                    ((Node)textPortList.item(0)).getNodeValue().trim();


                NodeList idList =
firstPersonElement.getElementsByTagName("ServiceID");
                Element idElement = (Element)idList.item(0);

                NodeList textIdList = idElement.getChildNodes();
                    ((Node)textIdList.item(0)).getNodeValue().trim();


                NodeList ipProList =
firstPersonElement.getElementsByTagName("IPPro");
                Element ipProElement = (Element)ipProList.item(0);

                NodeList textIPProList = ipProElement.getChildNodes();
                    ((Node)textIPProList.item(0)).getNodeValue().trim();


               //Collects the values of IP,IP_Provider in order to use them as
arguments in the Smcroute(routing in Linux).
                map.put(1,((Node)textIpList.item(0)).getNodeValue().trim());
                map.put(2,((Node)textIPProList.item(0)).getNodeValue().trim());
                map.put(3,true);
                movies[i][0] = ((Node)textIpList.item(0)).getNodeValue().trim();
                movies[i][1] =
((Node)textIPProList.item(0)).getNodeValue().trim();

                Iterator listIter1 = DServer.lista.listIterator();

                //Checks if the movie has already existed in list.
                while(listIter1.hasNext()){
                  tmpHash1 = (HashMap)listIter1.next();
                  if(tmpHash1.get(1)==map.get(1) &&
tmpHash1.get(2)==map.get(2)){
                    map.put(3,false);
                  }
                }
                //If the movie doesnt exist,puts it into the map.
                if(map.get(3)=="true")
```

```
                DServer.lista.add(map);

              }//end of if clause
             i++;
           }//end of for loop with s var

         }catch (SAXParseException err) {
         DServer.jTextField1.setText("Status: " +"** Parsing error" + ", line "
            + err.getLineNumber () + ", uri " + err.getSystemId ());
         DServer.jTextField1.setText("Status: " + err.getMessage ());

        }catch (SAXException e) {
        Exception x = e.getException ();
        ((x == null) ? e : x).printStackTrace ();

        }catch (Throwable t) {
         DServer.jTextField1.setText("Status: " +t.getMessage());
        }

        //Checks if a movie doesnt exist anymore.If exists --> true  else --> false.
        Iterator listIter2 = DServer.lista.listIterator();
         while(listIter2.hasNext()){
              tmpHash2 = (HashMap)listIter2.next();
              for (int i = 0; i < movies.length; i++) {

                  if((tmpHash2.get(1)==movies[i][0]) &&
tmpHash2.get(2)==movies[i][1]){
                        tmpHash2.put(3,true);
                         break;
                  } else
                     tmpHash2.put(3,false);
              }
         }

        DServer.readList();      //Reads the list and administrates the elements.Only
the files with true will be forwarding.
    }
    private HashMap tmpHash1;
    private HashMap tmpHash2;
    private String[][] movies;
}
```

```
/*
 * SendXml.java
 *
 * Created on 9 March 2007
 *
 */

package Server;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class SendXml implements Runnable{



    /** Creates a new instance of SendXml
     * This class sends the xml file via multicast in specific port  */
    public SendXml() {

          try {
             FileInputStream fis;

             File xmlFile = new File ("../finalXmlList.xml");                //find the
file.
             byte [] tempArray  = new byte [(int)xmlFile.length()];      //Makes a
temp array with the file's length.

             fis = new FileInputStream(xmlFile);                           //Puts the file
into InputStream.
             BufferedInputStream bis = new BufferedInputStream(fis);         //Puts
the InputStream into Buffer.
             bis.read(tempArray,0,tempArray.length);                       //Reads the
file.


             sock = new MulticastSocket();
             pack = new DatagramPacket( tempArray, tempArray.length,
                InetAddress.getByName( IPM_ADDRESS ), IPM_PORT );
          } catch( Exception e ) {
             DServer.jTextField1.setText("Status: Exception in creating socket or
packet: "
                   + e.getMessage() );
          }
       }

    public void run() {
```

```
        try {
            sock.setTimeToLive(10);
            sock.send( pack );              //sends the file as udp packets.
            DServer.jTextField1.setText("Status: XmlList sent." );
        } catch( Exception e ) {
            DServer.jTextField1.setText("Status: Exception in sending packet: "
                  + e.getMessage() );
        }
    }
   /** Port to use for IP Multicast sending. */
    public static final int IPM_PORT = 5678;
    /** Address to use for IP Multicast sending. */
    public static final String IPM_ADDRESS = "224.2.2.1";
    /** Time to wait between sending messages, in milliseconds. */
    public static final int SLEEP_MILLISECS = 10000;
    /** UDP Socket object to use for networking. */
    private MulticastSocket sock;
    /** UPD Packet object to use for sending message. */
    private DatagramPacket pack;
}
```

```
/*
```

* ForwardingDS.java

```
 *
 * Created on 27 March 2007.
 *
 */
package Server;

import java.io.IOException;
import java.net.ServerSocket;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class ForwardingDS implements Runnable{

    /**
     * Creates a new instance of ForwardingDS
     *This class administrates the routing.Uses the smcroute.
     */
    public ForwardingDS(String a, String b, int c) {
        ipMulti= a;
        ipPro =b;
        smcroute=c;
    }

    public void run() {
        ProcessBuilder tmp = new ProcessBuilder( );   //New Process
```

```
       try {
          //tmp.command(program,arguments).  program = smcroute (routing in
Linux)  , Smcroute routing arguments.
          if(smcroute==1){
             tmp.command("/usr/sbin/smcroute","-d");            //Starts Routing.
             tmp.start();
          }


          if(smcroute==2){
             tmp.command("/usr/sbin/smcroute" ,"-a","eth0",
ipPro,ipMulti,"eth1");     //Starts the specific route.
             tmp.start();
          }

          if(smcroute==0){
             tmp.command("/usr/sbin/smcroute","-k");          //Closes Routing.
             tmp.start();
          }

          if(smcroute==-1){
             tmp.command("/usr/sbin/smcroute", "-r","eth0", ipPro,ipMulti);
//Stops the specific route.
             tmp.start();
          }

       } catch(IOException ex) {
           DServer.jTextField1.setText("Status: "+ ex.getMessage());
          }
    }
   private  String ipMulti;
   private  String ipPro;
   private  int smcroute;
}
```

CMN

```
package CMN;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Enumeration;
import javax.swing.JOptionPane;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

/*
```

* CMN.java

```
 *
 * Created on 14 December 2007.
 */

/**
 *
 * @author  DIMITRIS PAPADOGIANNIS
 */
public class CMN extends javax.swing.JFrame {

    /**
     * Creates new form CMN

     */
    public CMN() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */

    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        jDialog1 = new javax.swing.JDialog();
        jScrollPane1 = new javax.swing.JScrollPane();
        jList1 = new javax.swing.JList();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
```

```
        jButton5 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jSeparator2 = new javax.swing.JSeparator();
        jDialog2 = new javax.swing.JDialog();
        jLabel2 = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        jList2 = new javax.swing.JList();
        jButton7 = new javax.swing.JButton();
        jButton8 = new javax.swing.JButton();
        jButton9 = new javax.swing.JButton();
        jSeparator1 = new javax.swing.JSeparator();
        jDialog3 = new javax.swing.JDialog();
        jButton11 = new javax.swing.JButton();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jButton12 = new javax.swing.JButton();
        jLabel5 = new javax.swing.JLabel();
        jTextField4 = new javax.swing.JTextField();
        jTextField1 = new javax.swing.JTextField();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();


jDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
        jDialog1.setTitle("Providers");
        jDialog1.setResizable(false);
        jScrollPane1.setPreferredSize(new java.awt.Dimension(260, 140));
        jList1.setRequestFocusEnabled(false);
        jScrollPane1.setViewportView(jList1);

        jButton2.setText("Start");
        jButton2.setToolTipText("Open The Connection");
        jButton2.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton3.setText("Close");
        jButton3.setToolTipText("Close The Connection");
        jButton3.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton4.setText("Interactive");
        jButton4.setToolTipText("See the Users");
        jButton4.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton4.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton5.setText("DS");
        jButton5.setToolTipText("Make the final xmlLIst");
        jButton5.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jLabel1.setText("The list of connected Active Users");

        org.jdesktop.layout.GroupLayout jDialog1Layout = new
org.jdesktop.layout.GroupLayout(jDialog1.getContentPane());
        jDialog1.getContentPane().setLayout(jDialog1Layout);
        jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog1Layout.createSequentialGroup()
                .addContainerGap()

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog1Layout.createSequentialGroup()

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                            .add(jDialog1Layout.createSequentialGroup()
                                .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 300,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                                .add(jButton4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 78,
Short.MAX_VALUE)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                                    .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                    .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                            .add(jLabel1))
                        .addContainerGap())
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog1Layout.createSequentialGroup()
```

```
                    .add(jButton5,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())))
            .add(jSeparator2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 604,
Short.MAX_VALUE)
        );
        jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog1Layout.createSequentialGroup()
                .add(26, 26, 26)
                .add(jLabel1)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog1Layout.createSequentialGroup()
                        .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                            .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(jButton4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                    .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 270,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(15, 15, 15)
                .add(jButton5, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jSeparator2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
10, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(228, Short.MAX_VALUE))
        );

jDialog2.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
        jDialog2.setTitle("Users");
        jDialog2.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        jDialog2.setResizable(false);
        jLabel2.setText("The list of connected Interactive Users");

        jScrollPane2.setPreferredSize(new java.awt.Dimension(260, 140));
        jList2.setRequestFocusEnabled(false);
        jScrollPane2.setViewportView(jList2);

        jButton7.setText("Start");
        jButton7.setToolTipText("Open The Connection");
```

```
        jButton7.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

        jButton8.setText("Close");
        jButton8.setToolTipText("Close The Connection");
        jButton8.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton8ActionPerformed(evt);
            }
        });

        jButton9.setText("Active");
        jButton9.setToolTipText("See the Providers");
        jButton9.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton9ActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout jDialog2Layout = new
org.jdesktop.layout.GroupLayout(jDialog2.getContentPane());
        jDialog2.getContentPane().setLayout(jDialog2Layout);
        jDialog2Layout.setHorizontalGroup(

jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog2Layout.createSequentialGroup()
                .addContainerGap()

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog2Layout.createSequentialGroup()
                        .add(jScrollPane2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 300,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jButton9,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 78,
Short.MAX_VALUE)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG)
                            .add(jButton7,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(jButton8,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                    .add(jLabel2))
                .addContainerGap())
```

```
            .add(jSeparator1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 604,
Short.MAX_VALUE)
        );
        jDialog2Layout.setVerticalGroup(

jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog2Layout.createSequentialGroup()
                .add(27, 27, 27)
                .add(jLabel2)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog2Layout.createSequentialGroup()
                        .add(jButton7,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                            .add(jButton8,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(jButton9,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                    .add(jScrollPane2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 270,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(47, 47, 47)
                .add(jSeparator1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
10, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(226, 226, 226))
        );
        jDialog3.setTitle("DS");
        jDialog3.setResizable(false);
        jButton11.setText("Send");
        jButton11.setToolTipText("Connect to Server and send the list");
        jButton11.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton11.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton11ActionPerformed(evt);
            }
        });

        jTextField2.setText("localhost");

        jTextField3.setText("4444");

        jLabel3.setText("IP Address of DS:");

        jLabel4.setText("Port:");

        jButton12.setText("Close");
        jButton12.setToolTipText("Close the connection");
        jButton12.setPreferredSize(new java.awt.Dimension(70, 20));
```

```
        jButton12.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton12ActionPerformed(evt);
            }
        });

        jLabel5.setText("Reconnect Time (Min):");

        jTextField4.setToolTipText("Set the time to reconnect with the Server");

        org.jdesktop.layout.GroupLayout jDialog3Layout = new
org.jdesktop.layout.GroupLayout(jDialog3.getContentPane());
        jDialog3.getContentPane().setLayout(jDialog3Layout);
        jDialog3Layout.setHorizontalGroup(

jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog3Layout.createSequentialGroup()

.add(jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jDialog3Layout.createSequentialGroup()
                        .addContainerGap()

.add(jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                            .add(jLabel3)
                            .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 90,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                        .add(34, 34, 34)

.add(jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                            .add(jLabel4)
                            .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog3Layout.createSequentialGroup()
                                .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 65,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 91,
Short.MAX_VALUE)
                                .add(jButton11,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))))
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog3Layout.createSequentialGroup()
                        .add(290, 290, 290)
                        .add(jButton12,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .add(jDialog3Layout.createSequentialGroup()
                        .addContainerGap()

.add(jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILI
NG, false)
                            .add(org.jdesktop.layout.GroupLayout.LEADING, jTextField4)
```

```
                    .add(org.jdesktop.layout.GroupLayout.LEADING, jLabel5,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
            .addContainerGap())
        );
        jDialog3Layout.setVerticalGroup(

jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog3Layout.createSequentialGroup()
            .addContainerGap(26, Short.MAX_VALUE)

.add(jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                .add(jLabel3)
                .add(jLabel4))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jDialog3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jButton11,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 25,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(jButton12, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(20, 20, 20)
            .add(jLabel5)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(jTextField4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(142, 142, 142))
        );

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Cell Main Node");
        jTextField1.setEditable(false);
        jTextField1.setText("Status:");

        jMenu1.setText(" Administration");
        jMenuItem1.setText("Active Users");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem1);
```

```
        jMenuItem2.setText("Interactive Users");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem2);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, jTextField1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 892, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(632, Short.MAX_VALUE)
                .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        );
        setBounds(50, 10, 900, 700);
    }// </editor-fold>

    private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "Close"--> Closes the connection between DS and CMN.

        flagSend=false;
    }

    private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "Send" --> Sends the file with the movies to the DS.

        flagSend=true;
        Thread threadSend = new Thread(new SendList());
        threadSend.start();
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "DS"--> Opens the jDialog3 which makes the connection between
DS and CMN.

        jDialog3.setBounds(300,40,400,400);
        jDialog3.setVisible(true);
    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // JButton "Close" --> Closes the socket for the IU.

        flagU=false;
        flagReceive=false;

        //Close the routing.
        Thread thrdForward =new Thread(new ForwardingCMN(null,null,0));
        thrdForward.start();
    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "Active"--> Shows the jDialog1 --> The Active Users.

        jDialog1.setBounds(30,30,600,600);
        jDialog1.setVisible(true);
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "Interactive"--> Shows the jDialog2 --> The Interactiveactive
Users.

       jDialog2.setBounds(650,30,600,600);
       jDialog2.setVisible(true);
       Thread xmlList = new Thread(new ReceiveXml());
       xmlList.start();
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "Close" --> Closes the socket for the AU.

        flagP=false;
        AUserConnection.socketP=null;
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // JButton "Start"  --> It opens a connection for the Users, a thread will start.
    // Firstly finds all the available interfaces and then shows them in a
JOptionPane in which user selects the inteface.
    // Two Threads will start.The first for the Users and the second for the Routing.

        flagU=true;

        //Shows the JOptionPane to select the availiable interface.
        InetAddress value=(InetAddress)JOptionPane.showInputDialog
        (null,"Select the Network
Interface","Interface",JOptionPane.INFORMATION_MESSAGE,null,in,in[0]);

        //     Creates a new Thread to open a ServerSocket for the movie's
selection.
        Thread threadU = new Thread(new IUserConnection(jList2,value));
//The jList2 argument is used to change the jList2 from another class.
        threadU.start();

        //Creates a new Thread to open a ServerSocket for the Users.
        Thread threadRouting = new Thread(new Routing(value));     //The value
argument is used to determine the interface for the ServerSocket.
        threadRouting.start();
```

```
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
     // JButton "Start" --> It opens a connection for the Providers, a thread will
start.
     // Firstly finds all the available interfaces and then shows them in a
JOptionPane in which user selects the inteface.

        flagP=true;

        //Shows the JOptionPane to select the availiable interface.
        InetAddress value=(InetAddress)JOptionPane.showInputDialog
        (null,"Select the Network
Interface","Interface",JOptionPane.INFORMATION_MESSAGE,null,in,in[0]);

        Thread threadP = new Thread(new AUserConnection(jList1,value));     //The
jList1 argument is used to change the jList1 from another class.
        threadP.start();

    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    // JMenuItem " Users" --> opens the jDialog2 and operates the connections of
the Users.

        jDialog2.setBounds(650,30,600,600);
        jDialog2.setVisible(true);

    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // JMenuItem " Providers" --> opens the jDialog1 and operates the
connections of the Providers.

        jDialog1.setBounds(30,30,600,600);
        jDialog1.setVisible(true);
    }
    protected static void copyFile(File inputFile,File outputFile) throws Exception {
        //Copies a file into another one.

        FileReader in = new FileReader(inputFile);
        FileWriter out = new FileWriter(outputFile);
        int c;

        while ((c = in.read()) != -1)
          out.write(c);

        in.close();
        out.close();
        }
    //Searchs for aviliable interfaces.
    private static void interfaces(){
        Enumeration e,f;
            try {
                int ni = 0;
                e = NetworkInterface.getNetworkInterfaces();
```

```
            while (e.hasMoreElements()) {
                f = ((NetworkInterface)e.nextElement()).getInetAddresses();
                while (f.hasMoreElements()) {
                    ((InetAddress)f.nextElement()).getHostName();
                    ++ni;
                }
            }
            in = new InetAddress[ni];
            ni = 0;
            e = NetworkInterface.getNetworkInterfaces();
            while (e.hasMoreElements()) {
                f = ((NetworkInterface)e.nextElement()).getInetAddresses();
                while (f.hasMoreElements())
                    in[ni++] = (InetAddress)f.nextElement();
            }
            for (int i = 0; i < in.length; i++) {
                 in[i].getCanonicalHostName();
            }
        } catch (SocketException ex) {
            ex.printStackTrace();
        }
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new CMN().setVisible(true);
            interfaces();
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton jButton11;
private javax.swing.JButton jButton12;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JButton jButton9;
private javax.swing.JDialog jDialog1;
private javax.swing.JDialog jDialog2;
private javax.swing.JDialog jDialog3;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JList jList1;
private javax.swing.JList jList2;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem2;
```

```
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    protected static javax.swing.JTextField jTextField1;
    protected static javax.swing.JTextField jTextField2;
    protected static javax.swing.JTextField jTextField3;
    protected static javax.swing.JTextField jTextField4;
    // End of variables declaration
    protected static boolean flagP=true;
  //Uses for the connection between CMN and Providers.
    protected static boolean flagU=true;
  //Uses for the connection between CMN and Users.
    protected static boolean flagSend=true;
 //Uses for the connection between CMN and Server.
    protected static boolean flagReceive=true;
 //Uses for the connection between CMN and Server.
    protected static Thread threadSend;
    private static InetAddress[] in;     //Table with all the availiable interfaces.
    protected static java.util.List lista = new ArrayList();
   // Dynamic administration of the movies.
}
```

```
/*
 * AUserConnection.java
 *
 * Created on 19 December 2007
 *
 */

package CMN;

import java.io.BufferedOutputStream;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import javax.swing.DefaultListModel;
import javax.swing.JList;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class AUserConnection implements Runnable{
```

```
   /**
    * Creates a new instance of AUserConnection
    *This class creates the connections between CMN and Active Users.Shows the
properties of the user in a jList.
    *In each connection it receives the txt file to each Active user.
    */
   public AUserConnection(JList tmp, InetAddress tmp2) {
     jListP=tmp;
     value=tmp2;
   }
   public void run(){
      try{
         serverP = new ServerSocket(8888,10,value);
     //New ServerSocket for ActiveUsers.
      }
      catch(IOException ex) {
         ex.printStackTrace();
      }


       while (CMN.flagP == true) {
         CMN.jTextField1.setText("Status: Waiting...");
         Socket socketP = null;
         try {
            socketP = serverP.accept();
 // Name Server Socket -->socketP contains the acceptance of the connection.

         } catch (IOException ex) {
           CMN.jTextField1.setText("Status: " + ex.getMessage());
         }
         //Writes the proerties of each socketP in the jList1.
         if (socketP!=null){
            CMN.jTextField1.setText("Accepted connection : " + socketP);
            listModelSP.addElement("---" + socketP);
            jListP.setModel(listModelSP);
         }
         else
          break;

         int filesize=6022386; // filesize temporary hardcoded
         int bytesRead = 0;
         int current = 0;

         //Receives the txt file from each ActiveUser.
         byte [] tempArray  = new byte [filesize];
         InputStream is = null;
         try {
            is = socketP.getInputStream();
            FileOutputStream fos = null;
            fos = new FileOutputStream("../videoList.txt",true);

            BufferedOutputStream bos = new BufferedOutputStream(fos);
            bytesRead = is.read(tempArray,0,tempArray.length);
            current = bytesRead;
            while (bytesRead > -1) {
               bytesRead = is.read(tempArray, current, (tempArray.length-
current));
```

```
                current += bytesRead;
                bos.write(tempArray, 0 , current);
             }
              bos.close();

             if (CMN.flagP==false){              //Checks for the value of flagP.
                serverP.close();
             }

          } catch (FileNotFoundException ex) {
          CMN.jTextField1.setText("Status: " + ex.getMessage());
          } catch (IOException ex) {
          CMN.jTextField1.setText("Status: " + ex.getMessage());
          }
       }
    }
    private DefaultListModel listModelSP= new DefaultListModel();         //Name
listModel  CMN for the Providers.
    private JList jListP;                                          // jlist for the Providers.
    protected static ServerSocket serverP = null;
    private BufferedWriter out;
    protected static Socket socketP;
    private InetAddress value;
}
```

```
/*
```

* IUserConnection.java

```
 *
 * Created on 19 December 2007.
 *
 */

package CMN;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import javax.swing.DefaultListModel;
import javax.swing.JList;


/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class IUserConnection implements Runnable {
```

```
   /**
    * Creates a new instance of IUserConnection
    *This class creates the connections between CMN and Interactive Users.Shows
the properties of the user in a jList.
    *In each connection it sends the xml file to each user.
    */
   public IUserConnection(JList tmp, InetAddress tmp2) {
      jListU=tmp;
      value=tmp2;
   }
   public void run(){
       ServerSocket serverU = null;
      try{
         serverU = new ServerSocket(9999,10,value);           //New
ServerSocket for Users.
      }
      catch(IOException ex) {
         ex.printStackTrace();
      }

      //With these two arguments we can start Forwarding.A new Thread will
start.
            Thread thrdForward =new Thread(new ForwardingCMN(null,null,1));
            thrdForward.start();

      while (CMN.flagU == true) {
         CMN.jTextField1.setText("Status: Waiting...");
         Socket socketU = null;
         try {
            socketU = serverU.accept();      // Name Server Socket -->socketU
contains the acceptance of the connection.

         } catch (IOException ex) {
            ex.printStackTrace();
         }
         //Writes the proerties of each socketU in the jList2.
         CMN.jTextField1.setText("Status: Accepted connection : " + socketU);
         listModelSU.addElement("---" + socketU);
         jListU.setModel(listModelSU);


        // Sends the xml file(contains all the movies) to each User when he
connects to CMN.
         File xmlFile = new File ("../finalXmlList.xml");
         byte [] tempArray  = new byte [(int)xmlFile.length()+1];
         FileInputStream fis = null;

         try {
            fis = new FileInputStream(xmlFile);
            BufferedInputStream bis = new BufferedInputStream(fis);
            bis.read(tempArray,0,tempArray.length);
            OutputStream os = null;
            os = socketU.getOutputStream();
            os.write(tempArray,0,tempArray.length);
            os.flush();
            os.close();
```

```
            CMN.jTextField1.setText("Status: The final xmlList was sent...");

          } catch (FileNotFoundException ex) {
              CMN.jTextField1.setText("Status: Error..." + ex.getMessage());

          } catch (IOException ex) {
              CMN.jTextField1.setText("Status: Error..." + ex.getMessage());

          }

          if (CMN.flagU==false){
             try {
                serverU.close();
             } catch (IOException ex) {
                ex.printStackTrace();
             }
          }
       }

    }
private JList jListU;       // jlist for the Clients.
private DefaultListModel listModelSU= new DefaultListModel();
private ServerSocket serverU = null;
private Socket socketU;
private InetAddress value;
}
```

```
/*
 * SendList.java
 *
 * Created on 20 March 2007.
 *
 */

package CMN;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class SendList implements Runnable{

    /** Creates a new instance of SendList
```

```
    This class sends the list with the movies to Distribution Server via reverse
path.
     */
    public SendList() {
    }

    public void run() {
        while ( CMN.flagSend==true){
            try{
                aUConnectionS = new
Socket(CMN.jTextField2.getText(),Integer.parseInt(CMN.jTextField3.getText()));
            }catch (UnknownHostException ex) {
                CMN.jTextField1.setText("Status: " + ex.getMessage());
            } catch (IOException ex) {
                CMN.jTextField1.setText("Status: " + ex.getMessage());
            }
           // sendfile
                    File xmlFile = new File ("../videoList.txt");               //find the
file.
                    byte [] tempArray  = new byte [(int)xmlFile.length()+1];
//Makes a temp array with the file's length.
                    FileInputStream fis = null;

                    try {
                        fis = new FileInputStream(xmlFile);                         //Puts
the file into InputStream.
                        BufferedInputStream bis = new BufferedInputStream(fis);
//Puts the InputStream into Buffer.
                        bis.read(tempArray,0,tempArray.length);
//Reads the file.
                        OutputStream os = null;
                        os = aUConnectionS.getOutputStream();
//Sends the file.
                        os.write(tempArray,0,tempArray.length);
//Makes the file
                        os.flush();
                        os.close();

                        CMN.jTextField1.setText("Status: The List was sent...");


CMN.threadSend.sleep(Integer.parseInt(CMN.jTextField4.getText())*60000);
                        //Stops the Thread in order to send the list again in specific
time(milliseconds).
                        //60000 milliseconds = 1Min
                       if(CMN.flagSend==false){

                            aUConnectionS.close();
                            CMN.jTextField1.setText("Status: Connection is closed...");
                        }
                    } catch (InterruptedException ex) {
                          CMN.jTextField1.setText("Status: Error...2" +
ex.getMessage());

                    } catch (FileNotFoundException ex) {
                          CMN.jTextField1.setText("Status: Error...3" +
ex.getMessage());
```

```
                } catch (IOException ex) {
                    CMN.jTextField1.setText("Status: Error...4" +
ex.getMessage());
                }
            }
    }
    private Socket  aUConnectionS;
}
```

```
/*
 * ReceiveXml.java
 *
 * Created on 9 March 2007
 *
 */

package CMN;

import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.util.Enumeration;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class ReceiveXml implements Runnable{

    /** Creates a new instance of ReceiveXml
     This class receives the xml File by reading the dvb-T receiver in specific IP
Address.
     */
    public ReceiveXml() {
        receiveBuffer = new byte[BUFFER_SIZE];

        try {

           sock = new MulticastSocket( IPM_PORT );
           sock.setInterface(in[1]);
           pack = new DatagramPacket( receiveBuffer, receiveBuffer.length );

           sock.joinGroup(InetAddress.getByName(IPM_ADDRESS));
        } catch( Exception e ) {
```

```
        CMN.jTextField1.setText("Status: Exception in creating Socket or Packet:
"
                + e.getMessage() );
      }
   }

   public void run() {

      while(CMN.flagReceive==true){
         try {

            sock.receive( pack );
            FileOutputStream fos = null;
            fos = new FileOutputStream("../finalXmlList.xml");
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            bos.write(receiveBuffer, 0 , pack.getLength());
            bos.close();

            CMN.jTextField1.setText("Status: XmlList received: "  );

            pack.setLength( receiveBuffer.length );
         } catch( Exception e ) {
            CMN.jTextField1.setText("Status: Exception in receiving packet: "
                  + e.getMessage() );
         }
      }
   }
   /** Port to use for IP Multicast receiving. */
   public static final int IPM_PORT = 5678;
   /** IP Multicast address to receive on. */
   public static final String IPM_ADDRESS = "224.2.2.1";
   /** Size of receive buffer, in bytes. */
   public static final int BUFFER_SIZE = 1024;
   /** IP Multicast Socket object to use. */
   private MulticastSocket sock;
   /** UDP Packet object to use for receiving message. */
   private DatagramPacket pack;
   /** Buffer to store received text in. */
   private byte[] receiveBuffer;
   private InetAddress[] in;
}
```

```
/*
 * RequestList.java
 *
 * Created on 11 March 2007,
 *
 */

package CMN;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class RequestList {

    /** Creates a new instance of RequestList
     This class saves the properties of each movie.
     */
    public RequestList(String a,String b) {
        IpMulticast=a;
        IpProvider=b;
        counter=1;
    }

    public String IpMulticast,IpProvider;
    public int counter;
}
```

```
/*
 * Routing.java
 *
 * Created on 26 March 2007
 *
 */

package CMN;

import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
```

```
 */
public class Routing implements Runnable{
   //This Class makes the routing by taking the request from each User(IP
Multicast,IP Provider)
   //and puts the arguments in the Class Forwarding.Creates a new ServerSocket
for the requests.

   /** Creates a new instance of Routing */
   public Routing(InetAddress tmp2) {
    tmp2=value;
   }

   public void run() {

       int bytesRead = 0;
       byte [] tempArray  = new byte [20];

       ServerSocket serverRouting = null;
       try{
          serverRouting = new ServerSocket(7777,10,value);
       }
       catch(IOException ex) {
       CMN.jTextField1.setText("Status: " + ex.getMessage());
       }

      while (CMN.flagU == true) {
          CMN.jTextField1.setText("Status: Waiting...");
          Socket socketR = null;
          try {
              socketR = serverRouting.accept();
  // Name Server Socket -->socketR contains the acceptance of the connection.

           InputStream is = socketR.getInputStream();
        //Reads the InputStream.
           bytesRead = is.read( tempArray,0,tempArray.length);
           byte [] tempArray1  = new byte [bytesRead];
      //Saves them as bytes.
           for (int i = 0; i < bytesRead; i++) {
             tempArray1[i] = tempArray[i];
           }

            String str = new String(tempArray1);
       //Casts the bytes into String-->Takes the initial request.

            String[] t=null;
                //Seperate the string into two pieces (IP Multicast,IP Provider)
            t = str.split("/");
            System.out.println(""+t[0]+"\n");
            System.out.println(""+t[1]+"\n");


            Iterator listIter1 = CMN.lista.listIterator();

            boolean exists = false;
             while(listIter1.hasNext()){                // Looks for  added movies.
               RequestList tmp = (RequestList)listIter1.next();
               if (tmp.IpProvider.equals(t[1]) &&
```

```
                    tmp.IpMulticast.equals(t[0])) {
                exists = true;
                ++tmp.counter;
            }
          }
          if (!exists)
            CMN.lista.add(new RequestList(t[0],t[1]));

          if(t[0]=="224.2.2.2"){
            Thread thrdForward =new Thread(new
ForwardingCMN(t[0],t[1],3));
            thrdForward.start();
          }else {

          //With these two arguments we can start Forwarding.A new Thread
will start.
          Thread thrdForward =new Thread(new ForwardingCMN(t[0],t[1],2));
          thrdForward.start();
          }
        } catch (IOException ex) {
        CMN.jTextField1.setText("Status: " + ex.getMessage());
        }
      }
   }
   private InetAddress value;
}
```

```
/*
 * ForwardingCMN.java
 *
 * Created on 27 March 2007.
 *
 */

package CMN;

import java.io.IOException;
import java.net.ServerSocket;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class ForwardingCMN implements Runnable{          //This class
administrate the Routing between CMN and Users.
```

```
   /**
    * Creates a new instance of ForwardingCMN
    */
   public ForwardingCMN(String a, String b, int c) {
   ipMulti= a;
   ipPro =b;
   smcroute = c;
   }

   public void run() {
      ProcessBuilder tmp = new ProcessBuilder( );   //New Process

     try {
         //tmp.command(program,arguments).  program = smcroute (routing in
Linux)  , Smcroute routing arguments.
         if(smcroute==1){
            tmp.command("/usr/sbin/smcroute","-d");            //Starts Routing.
            tmp.start();
         }

         if(smcroute==2){
            tmp.command("/usr/sbin/smcroute" ,"-a","dvb0_0",
ipPro,ipMulti,"eth1");     //Start specific route.
            tmp.start();
         }

         if(smcroute==0){
            tmp.command("/usr/sbin/smcroute","-k");          //Closes Routing.
            tmp.start();
         }

         if(smcroute==-1){
            tmp.command("/usr/sbin/smcroute", "-r","eth0", ipPro,ipMulti);
//Removes Routing.
            tmp.start();
         }

      } catch(IOException ex) {
         CMN.jTextField1.setText("Status: " + ex.getMessage());
         }
   }
  private  String ipMulti;  //IP Multicast.
  private  String ipPro;    //IP Provider.
  private int smcroute;
}
```

Χρήστης

```
package User;

import java.io.BufferedOutputStream;
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.HashMap;
import java.util.Map;
import javax.swing.DefaultListModel;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

/*
```

* InteractiveUser.java

```
 *
 * Created on 13 December 2006.
 */

/**
 *
 * @author  DIMITRIS PAPADOGIANNIS
 */
public class InteractiveUser extends JFrame {

    /**
     * Creates new form InteractiveUser
     *
     *This class creates a user interface in order to see the availiable multimedia
content.Firstly connects with CMN
     *and download the file with all the elements.Then chooses the file and sends
the option back to CMN.After that CMN starts
     *ip forwarding.The User receives it by using the program VLC.
     */
    public InteractiveUser() {

        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
```

```
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        jDialog1 = new javax.swing.JDialog();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jDialog2 = new javax.swing.JDialog();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jButton4 = new javax.swing.JButton();
        jTextField1 = new javax.swing.JTextField();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        jMenuItem2 = new javax.swing.JMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem3 = new javax.swing.JMenuItem();


jDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
        jDialog1.setTitle("Open Stream");
        jDialog1.setModal(true);
        jDialog1.setResizable(false);
        jButton1.setText("Connect ");
        jButton1.setToolTipText("Connect to the  server");
        jButton1.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Show List");
        jButton2.setToolTipText("Shows the movie list");
        jButton2.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton3.setText("Play ");
        jButton3.setToolTipText("Play the selected movie");
        jButton3.setPreferredSize(new java.awt.Dimension(70, 20));
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            jButton3ActionPerformed(evt);
        }
    });

    org.jdesktop.layout.GroupLayout jDialog1Layout = new
org.jdesktop.layout.GroupLayout(jDialog1.getContentPane());
    jDialog1.getContentPane().setLayout(jDialog1Layout);
    jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog1Layout.createSequentialGroup()
            .add(25, 25, 25)
            .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
100, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 118,
Short.MAX_VALUE)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                .add(org.jdesktop.layout.GroupLayout.TRAILING, jTabbedPane1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 329, Short.MAX_VALUE)
                .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog1Layout.createSequentialGroup()
                    .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(129, 129, 129)
                    .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 100,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
            .add(22, 22, 22))
    );
    jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog1Layout.createSequentialGroup()
            .add(38, 38, 38)
            .add(jTabbedPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 218,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 85,
Short.MAX_VALUE)

.add(jDialog1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELI
NE)
                .add(jButton3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jButton1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jButton2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .add(69, 69, 69))
    );
```

```
jDialog2.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);
        jDialog2.setTitle("Settings");
        jDialog2.setModal(true);
        jDialog2.setResizable(false);
        jLabel1.setText("IP Address:");

        jLabel2.setText("Port:");

        jLabel3.setText("Put CMN's IP Address and Port to request for a movie:");

        jLabel4.setText("Put CMN's IP Address and Port to receive the List:");

        jLabel5.setText("IP Address:");

        jLabel7.setText("Port:");

        jButton4.setText("Save");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout jDialog2Layout = new
org.jdesktop.layout.GroupLayout(jDialog2.getContentPane());
        jDialog2.getContentPane().setLayout(jDialog2Layout);
        jDialog2Layout.setHorizontalGroup(

jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jDialog2Layout.createSequentialGroup()
                .add(56, 56, 56)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                    .add(jLabel4)
                    .add(jDialog2Layout.createSequentialGroup()

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                            .add(jLabel1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                            .add(jTextField2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                            .add(jTextField4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE))
                        .add(jLabel5))
                    .add(222, 222, 222)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG, false)
                        .add(jLabel7)
                        .add(jLabel2)
```

```
                    .add(jTextField5,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 59, Short.MAX_VALUE)
                    .add(jTextField3)))
                .add(jLabel3))
            .add(40, 40, 40))
        .add(org.jdesktop.layout.GroupLayout.TRAILING,
jDialog2Layout.createSequentialGroup()
            .addContainerGap(197, Short.MAX_VALUE)
            .add(jButton4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
100, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(180, 180, 180))
    );
    jDialog2Layout.setVerticalGroup(

jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(jDialog2Layout.createSequentialGroup()
            .addContainerGap()
            .add(jLabel4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 14,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(21, 21, 21)

.add(jDialog2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
                .add(jDialog2Layout.createSequentialGroup()
                    .add(jLabel2)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(73, 73, 73)
                    .add(jLabel7)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jTextField5,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(jDialog2Layout.createSequentialGroup()
                    .add(jLabel1)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(31, 31, 31)
                    .add(jLabel3)
                    .add(28, 28, 28)
                    .add(jLabel5)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jTextField4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 38,
Short.MAX_VALUE)
            .add(jButton4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
25, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
```

```
            .add(38, 38, 38))
        );

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Interactive User");
        jTextField1.setEditable(false);
        jTextField1.setText("Status:");

        jMenu1.setText("File");
        jMenuItem1.setText("Open Network Stream");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem1);

        jMenu1.add(jSeparator1);

        jMenuItem2.setText("Exit");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });

        jMenu1.add(jMenuItem2);

        jMenuBar1.add(jMenu1);

        jMenu2.setText("Settings");
        jMenuItem3.setText("View");
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });

        jMenu2.add(jMenuItem3);

        jMenuBar1.add(jMenu2);

        setJMenuBar(jMenuBar1);

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jTextField1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 792,
Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
```

```
            .addContainerGap(532, Short.MAX_VALUE)
            .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
      );
      setBounds(10, 10, 800, 600);
   }// </editor-fold>

   private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

   //JButton"Save"-->Saves the parameters of the connections in a txt file.

      settings=settings+"IP:"+jTextField2.getText()+"\n";
      settings=settings+"Port:"+jTextField3.getText()+"\n";
      settings=settings+"IP2:"+jTextField4.getText()+"\n";
      settings=settings+"Port2:"+jTextField5.getText()+"\n";
      File outputFile = new File("..\\parameters.txt");
      FileWriter out;

      try {
         out = new FileWriter(outputFile);
         out.write(settings);
         out.close();
      } catch (IOException ex) {
      jTextField1.setText("Status: " +ex.getMessage());
      }
   }

   private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
   //Loads the file "parameters.txt" which contains all the parameters.

      loadParameters();  //Loads the values in the jTextFields.

      jDialog2.setBounds(100,100,100,100);
      jDialog2.pack();
      jDialog2.setVisible(true);

   }

   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
   // JButton "Play" --> It reads the xml file with the movies and play them with
vlc. New thread.

      Thread thrdPlay = new Thread(new Play());
      thrdPlay.start();
   }

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

   // JButton "Download List" --> Receives the xmlList.
   //The following code opens a xml file ,reads it and shows the titles of each
movie in the jList1.
   //On each different Service_Id(provider) makes a new tab in the jList1.

       try {

          JList xmlJList = new JList();
```

```
        jListModel= new DefaultListModel();
        jTabbedPane1.removeAll();

        DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
        Document doc = docBuilder.parse (new File("..\\xmlListU.xml"));

        // normalize text representation
        doc.getDocumentElement ().normalize ();
            doc.getDocumentElement().getNodeName();


        NodeList listOfPersons = doc.getElementsByTagName("Movie");
        int totalPersons = listOfPersons.getLength();

        for(int s=0; s<listOfPersons.getLength() ; s++){


           Node firstPersonNode = listOfPersons.item(s);
           if(firstPersonNode.getNodeType() == Node.ELEMENT_NODE){


              Element firstPersonElement = (Element)firstPersonNode;

              NodeList titleList =
firstPersonElement.getElementsByTagName("Title");
              Element titleElement = (Element)titleList.item(0);

              NodeList textTitleList = titleElement.getChildNodes();
                   ((Node)textTitleList.item(0)).getNodeValue().trim();


              NodeList ipList = firstPersonElement.getElementsByTagName("IP");
              Element ipElement = (Element)ipList.item(0);

              NodeList textIpList = ipElement.getChildNodes();
                   ((Node)textIpList.item(0)).getNodeValue().trim();


              NodeList portList =
firstPersonElement.getElementsByTagName("Port");
              Element portElement = (Element)portList.item(0);

              NodeList textPortList = portElement.getChildNodes();
                   ((Node)textPortList.item(0)).getNodeValue().trim();


              NodeList idList =
firstPersonElement.getElementsByTagName("ServiceID");
              Element idElement = (Element)idList.item(0);

              NodeList textIdList = idElement.getChildNodes();
                   ((Node)textIdList.item(0)).getNodeValue().trim();
```

```
                NodeList ipProList =
firstPersonElement.getElementsByTagName("IPPro");
                Element ipProElement = (Element)ipProList.item(0);

                NodeList textIPProList = ipProElement.getChildNodes();
                      ((Node)textIPProList.item(0)).getNodeValue().trim();


                tabName1=((Node)textIdList.item(0)).getNodeValue().trim();

                //Saves the elements of each movie in the class "Movies"
                //If another Service_Id comes then a new tab is created.
                if (!tabName.equals(tabName1)){


                   xmlJList= makeTab((new
Movies(((Node)textTitleList.item(0)).getNodeValue().trim(),
                            ((Node)textIpList.item(0)).getNodeValue().trim(),
                            ((Node)textPortList.item(0)).getNodeValue().trim(),
                            ((Node)textIdList.item(0)).getNodeValue().trim(),
                            ((Node)textIPProList.item(0)).getNodeValue().trim())));
                   jTabbedPane1.addTab(tabName1,xmlJList);

                }else {
                   jListModel.addElement(new
Movies(((Node)textTitleList.item(0)).getNodeValue().trim(),
                            ((Node)textIpList.item(0)).getNodeValue().trim(),
                            ((Node)textPortList.item(0)).getNodeValue().trim(),
                            ((Node)textIdList.item(0)).getNodeValue().trim(),
                            ((Node)textIPProList.item(0)).getNodeValue().trim()));

                   xmlJList.setModel(jListModel);
                   jTabbedPane1.addTab(tabName1,xmlJList);
                }
                tabName=tabName1;

             }//end of if clause
           }//end of for loop

       }catch (SAXParseException err) {
      jTextField1.setText("Status: " +"** Parsing error" + ", line "
           + err.getLineNumber () + ", uri " + err.getSystemId ());
       jTextField1.setText("Status: " + err.getMessage ());

       }catch (SAXException e) {
       Exception x = e.getException ();
       ((x == null) ? e : x).printStackTrace ();

       }catch (Throwable t) {
       jTextField1.setText("Status: " +t.getMessage());
       }
    }
    private JList makeTab(Movies movie){

     //Make a new  Tab each time a different Service_ID come.

       jListModel= new DefaultListModel();
```

```
        jListModel.addElement(movie);
        JList xmlJList =  new JList();
        xmlJList.setModel(jListModel);
        return xmlJList;
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    // JButton "Connect to Server" --> connect with the server

        try {
            sConnection = new
Socket(jTextField2.getText(),Integer.parseInt(jTextField3.getText()));
     //new Socket(Ip,Port).
        } catch (UnknownHostException ex) {
          jTextField1.setText("Status: " +ex.getMessage());
        } catch (IOException ex) {
            jTextField1.setText("Status: " +ex.getMessage());
        }

        int filesize=6022386; // filesize temporary hardcoded
        int bytesRead = 0;
        int current = 0;

        // receives file
        byte [] tempArray  = new byte [filesize];
        InputStream is = null;
        try {
            is = sConnection.getInputStream();
            FileOutputStream fos = null;
            fos = new FileOutputStream("..\\xmlListU.xml");
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            bytesRead = is.read(tempArray,0,tempArray.length);

            current = bytesRead;

            while (bytesRead > -1){
               bytesRead = is.read(tempArray, current, (tempArray.length-current));
               current += bytesRead;
            }

            bos.write(tempArray, 0 , current);
            bos.close();
            sConnection.close();
        } catch (FileNotFoundException ex){
          jTextField1.setText("Status: " +ex.getMessage());
        } catch (IOException ex){
          jTextField1.setText("Status: " +ex.getMessage());
        }


    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

    // JMenuItem "Open Network Stream" --> open the jDialog2 first if the file
with parameters doesn't exist
    //then call the "loadParameters" and opens the jDialog2.
```

```
        jTextField1.setText("Status: ");
        File f = new File("..\\parameters.txt");
         if(!f.exists()){
            jDialog2.setBounds(100,100,100,100);
            jDialog2.pack();
            jDialog2.setVisible(true);
        }
        loadParameters();
        jDialog1.setBounds(60,60,500,500);
        jDialog1.pack();
        jDialog1.setVisible(true);
    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {

    //  JMenuItem "Exit" --> close the application.

        setVisible(false);
          dispose();
    }
    private void loadParameters(){
        //Finds the file ,reads it and load the values in the jTextFields.
        try {
            FileReader f =new FileReader("..\\parameters.txt");
            BufferedReader bf =new BufferedReader(f);
            String s ="";
            int a=0;
            while((s = bf.readLine())!= null){
                a++;
                String[] t=null;                        //Seperate the string into two
pieces (Name,Value)
                t = s.split(":");
                if (a==1)
                jTextField2.setText(""+t[1]);
                if (a==2)
                jTextField3.setText(""+t[1]);
                if (a==3)
                jTextField4.setText(""+t[1]);
                if (a==4)
                jTextField5.setText(""+t[1]);
             }

            } catch (IOException ex) {
                jTextField1.setText("Status: " +ex.getMessage());
            }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new InteractiveUser().setVisible(true);
            }
        });
    }
```

```
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JDialog jDialog2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JSeparator jSeparator1;
    protected static javax.swing.JTabbedPane jTabbedPane1;
    protected static javax.swing.JTextField jTextField1;
    protected static javax.swing.JTextField jTextField2;
    protected static javax.swing.JTextField jTextField3;
    protected static javax.swing.JTextField jTextField4;
    protected static javax.swing.JTextField jTextField5;
    // End of variables declaration
    private Socket sConnection = null;  //Users's Connection with the  Server
    private int i = 0;              //Reading xml file uses this.
    private  String tabName="0";
    private  String tabName1=null;
    private static int list;
    private String settings="";    //This String saves the values in the file.
    private static JList xmlJList = new JList();
    private static DefaultListModel jListModel= new DefaultListModel();
}
```

```
/*
 * Movies.java
 *
 * Created on 12 March 2007
 *
 */

package User;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
```

```
 */
public class Movies {

   /** Creates a new instance of Movies */
   public Movies(String a,String b,String c,String d, String e) {
      title=a;
      ipMulticast=b;
      port=c;
      serviceId=d;
      ipProvider=e;

   }

   public String toString() {
      return title;
   }

   public String ipMulticast,ipProvider,port,title,serviceId;
}
```

```
/*
```

* Play.java

```
 *
 * Created on 10 January 2007
 *
 */

package User;

import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import javax.swing.JList;
import javax.swing.JOptionPane;

/**
 *
 * @author DIMITRIS PAPADOGIANNIS
 */
public class Play implements Runnable{

   /**     * Creates a new instance of Play
    */
   public Play() {
   }

   public void run(){
```

```
        ProcessBuilder playC = new ProcessBuilder( );   //New Process

        try {
           //Gets the selection of the jTabbedPane.


          tmpJList = (JList)InteractiveUser.jTabbedPane1.getSelectedComponent();
          Movies tmpMovies=(Movies)tmpJList.getSelectedValue();
          System.out.println(""+tmpMovies.title);

           if ( tmpJList.getSelectedIndex()== -1){     //If you don't select a value
from jList1 an error message will show up
           JOptionPane.showMessageDialog(InteractiveUser.jTabbedPane1,
                  "Please select a movie from the list.",  "List error",
JOptionPane.ERROR_MESSAGE);}
           else {

              //Sends to CMN the movie which the User want to watch.
               try {
                  cmnSocket = new
Socket(InteractiveUser.jTextField4.getText(),Integer.parseInt(InteractiveUser.jTe
xtField5.getText()));
                  //new Socket(Ip,Port).
                  OutputStream os = null;
                  os = cmnSocket.getOutputStream();
                   //Sends the file.
                  os.write((tmpMovies.ipMulticast + "/" +
tmpMovies.ipProvider).getBytes());       //Makes the file
                  os.flush();
                  os.close();

               } catch (UnknownHostException ex) {
               InteractiveUser.jTextField1.setText("Status: "+ex.getMessage());
               } catch (IOException ex) {
               InteractiveUser.jTextField1.setText("Status: "+ex.getMessage());
               }

     //         Uses the VLC program to receive the multicast transmition.
(Program,Arguments)

               playC.command("C:\\Program Files\\VideoLAN\\VLC\\vlc"  , "udp://@"
+ tmpMovies.ipMulticast  + ":" + tmpMovies.port );
               playC.start();
               InteractiveUser.jTextField1.setText("Status: " +playC.command());

           };

           } catch (IOException ex) {
              InteractiveUser.jTextField1.setText("Status: "+ex.getMessage());
              }
     }
     private  JList tmpJList = new JList();
     private  int currentTab;
     private Socket cmnSocket;
}
```

Βιβλιογραφία:

[1] http://middleware.objectweb.org/
[2] http://www.sei.cmu.edu/str/descriptions/middleware.html
[3] http://www.ist-athena.org/
[4] http://www.dvb.org/
[5] ETS 300 744, Digital Video Broadcasting (DVB): Framing structure, channel coding and modulation for Digital Terrestrial Television (DVB-T), ETSI, 1997
[6] http://www.mhp.org
[7] ETS EN 301 192, Digital Video Broadcasting (DVB): DVB Specification for data broadcasting, ETSI, 1999
[8] ETSI 102 812 Digital Video Broadcasting (DVB):Multimedia Home Platform (MHP) Specification 1.1.1
[9] Java™ Technologies for Interactive Television
[10] ETS 300 802: Digital Video Broadcasting (DVB): Network-independent protocols for DVB interactive services
[11] J.H Scott “The how and why of COFDM”
[12] J.-P. Evain, The Multimedia Home Platform – an overview, EBU Technical Review, Spring 1998
[13] http://www.videolan.org/
[14] ETSI 302 304, Digital Video Broadcasting (DVB): Transmission System for Handheld Terminals (DVB-H)
[15] ETSI: Draft EN 302 307: Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2).
[16] http://www.etsi.org/
[17] ISO/IEC DIS 13818-1: Generic coding of moving pictures and associated audio information: Systems
[18] www.sun.com