ΤΕΙ ΗΠΕΙΡΟΥ (ΑΡΤΑΣ)
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ & ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Υλοποίηση Πλατφόρμας συζητήσεων (chatting) βασισμένη στο μοντέλο Πελάτη-Εξυπηρέτησης (Client-Server)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Στεφανάτος Βασίλειος

Επιβλέπων Καθηγητής: Δρ. Καββαδίας Χριστόφορος

Πίνακας περιεχομένων

ΓΛΩΣΣΑΡΙ

QT	*Cross-platform* application development framework
MFC	Microsoft Foundation Class Library
TCP	Transmission Control Protocol
ΑΠ	Αντικειμενοστραφής Προγραμματισμός
LWPs	LightWeight Processes
HTTP	Hyper-text Transfer Protocol
HTML	Hyper-text Markup Language
MSN	MicroSoft Network
ICQ	Internet Messaging Computer Programm (“I seek you”)
IRC	Internet Relay Chat client
IDE	Integrated Drive Electronics
GUI	Graphical User Interface
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations
XHTML	Extensible Hypertext Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface

1. Εισαγωγή

Στα πλαίσια της πτυχιακής αυτής εργασίας μελετήθηκε το μοντέλο επικοινωνίας *Client/Server*. Πελάτες μπορούν να συνδέονται σε *server* της εφαρμογής και μέσα από αυτή να δημιουργούν δωμάτια επικοινωνίας (*chat rooms*), να συζητούν δημόσια αλλά και ιδιωτικά. Οι εφαρμογές που αναπτύχθηκαν περιέχουν *Rich User Interface* και υποστηρίζουν εικονίδια *emoticons*. Τέλος, χρησιμοποιήθηκαν οι κλάσεις *Sockets* για την επικοινωνία, η οποία βασίζεται σε πρωτόκολλο που αναπτύχθηκε, και η κλάση *Threads,* για να υποστηρίζονται ταυτόχρονα πολλαπλοί πελάτες.

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το *Microsoft Visual Studio 2008*. Για το γραφικό περιβάλλον έγινε έρευνα σε όλες τις διαθέσιμες βιβλιοθήκες, όπως *QT, wxWidgets, MFC* και σε άλλες τεχνολογίες και τελικά επιλέχθηκε η MFC.

Για την επικοινωνία μεταξύ του πελάτη και του εξυπηρετητή ορίζονται ορισμένες εντολές, ένα πρωτόκολλο επικοινωνίας, δηλαδή, για τη δημιουργία δωματίων επικοινωνίας, για αποστολή ιδιωτικών μηνυμάτων σε χρήστη, την είσοδο σε συγκεκριμένο δωμάτιο, την έξοδο από αυτό και άλλες λειτουργίες.

2. Παρουσίαση θεωρίας

Στην ενότητα αυτή θα παρουσιάσουμε κάποια βασικά στοιχεία του αντικειμενοστραφούς προγραμματισμού. Στη συνέχεια θα αναφερθούμε στα νήματα και την επικοινωνία με TCP sockets, καθώς αποτελούν τη βάση της εφαρμογής.

2.1 Αντικειμενοστραφής προγραμματισμός

Στην επιστήμη υπολογιστών **αντικειμενοστρεφή προγραμματισμό** (object-oriented programming), ή ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται *αντικείμενο* και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων ονόματι *κλάση*. Η κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους· αυτή υπήρξε η πρωταρχική καινοτομία του ΑΠ.

Έτσι μπορεί να οριστεί μία προδιαγραφή δομής αποθήκευσης (π.χ. μία κλάση "τηλεόραση") η οποία να περιέχει τόσο ιδιότητες (π.χ. μία μεταβλητή "τρέχον κανάλι") όσο και πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία "άνοιγμα της τηλεόρασης"). Στο εν λόγω παράδειγμα κάθε υλική τηλεόραση (κάθε αντικείμενο αποθηκευμένο πραγματικά στη μνήμη) αναπαρίσταται ως ξεχωριστό, "φυσικό" στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Επομένως μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς "καλούπια". Οι αιτίες

που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), όμως τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.

Οι περισσότερες αντικειμενοστρεφείς έννοιες εμφανίστηκαν αρχικά στη γλώσσα προγραμματισμού Simula 67, η οποία ήταν προσανατολισμένη στην εκτέλεση προσομοιώσεων του πραγματικού κόσμου. Οι ιδέες της Simula 67 επηρέασαν κατά τη δεκαετία του '70 την ανάπτυξη της Smalltalk, της γλώσσας που εισήγαγε τον όρο αντικειμενοστρεφής προγραμματισμός. Η Smalltalk αναπτύχθηκε από τον Άλαν Κέι της εταιρείας Xerox στο πλαίσιο μίας εργασίας με στόχο τη δημιουργία ενός χρήσιμου, αλλά και εύχρηστου, προσωπικού υπολογιστή. Όταν η τελική έκδοση της Smalltalk έγινε διαθέσιμη το 1980 η έρευνα για την αντικατάσταση του δομημένου προγραμματισμού με ένα πιο σύγχρονο υπόδειγμα ήταν ήδη εν εξελίξει. Στη γλώσσα αυτή όλοι οι τύποι δεδομένων ήταν κλάσεις (δεν υπήρχαν δηλαδή πιο παραδοσιακές δομές δεδομένων παρά μόνο αντικείμενα).

Την ίδια περίπου εποχή, και επίσης με επιρροές από τη Simula, ολοκληρωνόταν η ανάπτυξη της C++ ως μίας ισχυρής επέκτασης της δημοφιλούς γλώσσας προγραμματισμού C στην οποία είχαν "μεταμοσχευθεί" αντικειμενοστρεφή χαρακτηριστικά. Η επιρροή της C++ καθ' όλη της δεκαετία του '80 ήταν καταλυτική με αποτέλεσμα τη σταδιακή κυκλοφορία αντικειμενοστρεφών εκδόσεων πολλών γνωστών διαδικαστικών γλωσσών προγραμματισμού. Κατά το πρώτο ήμισυ της δεκαετίας του '90 η βαθμιαία καθιέρωση στους μικροϋπολογιστές των γραφικών διασυνδέσεων χρήστη (GUI), για την ανάπτυξη των οποίων ο ΑΠ φαινόταν ιδιαιτέρως κατάλληλος, και η επίδραση της C++ οδήγησαν στην επικράτηση της αντικειμενοστρέφειας ως βασικού προγραμματιστικού υποδείγματος.

Το 1995 η εμφάνιση της Java, μίας ιδιαίτερα επιτυχημένης, πλήρως αντικειμενοστρεφούς γλώσσας που έμοιαζε συντακτικώς με τη C/C++ και προσέφερε πρωτοποριακές για την εποχή δυνατότητες, έδωσε νέα ώθηση

στον ΑΠ. Παράλληλα εμφανίστηκαν ποικίλες άτυπες βελτιώσεις στο βασικό προγραμματιστικό υπόδειγμα, όπως οι αντικειμενοστρεφείς γλώσσες μοντελοποίησης λογισμικού, τα σχεδιαστικά πρότυπα κλπ. Το 2001 η Microsoft εστίασε την προσοχή της στην πλατφόρμα .NET, μία ανταγωνιστική της Java πλατφόρμα ανάπτυξης και εκτέλεσης λογισμικού η οποία ήταν εξολοκλήρου προσανατολισμένη στην αντικειμενοστρέφεια.

Βασικές έννοιες

Κεντρική ιδέα στον αντικειμενοστρεφή προγραμματισμό είναι η **κλάση** (class), μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού. Πρακτικώς είναι ένας τύπος δεδομένων, ή αλλιώς το προσχέδιο μίας δομής δεδομένων με δικά της περιεχόμενα, τόσο μεταβλητές όσο και διαδικασίες. Τα περιεχόμενα αυτά δηλώνονται είτε ως *δημόσια* (public) είτε ως *ιδιωτικά* (private), με τα ιδιωτικά να μην είναι προσπελάσιμα από κώδικα εκτός της κλάσης. Οι διαδικασίες των κλάσεων συνήθως καλούνται *μέθοδοι* (methods) και οι μεταβλητές τους *γνωρίσματα* (attributes) ή *πεδία* (fields). Μία κλάση πρέπει ιδανικά να είναι εννοιολογικά αυτοτελής, να περιέχει δηλαδή μόνο πεδία τα οποία περιγράφουν μία κατηγορία αντικειμένων και δημόσιες μεθόδους οι οποίες επενεργούν σε αυτά όταν καλούνται από το εξωτερικό πρόγραμμα, χωρίς να εξαρτώνται από άλλα δεδομένα ή κώδικα εκτός της κλάσης, και επαναχρησιμοποιήσιμη, να αποτελεί δηλαδή μαύρο κουτί δυνάμενο να λειτουργήσει χωρίς τροποποιήσεις ως τμήμα διαφορετικών προγραμμάτων.

Αντικείμενο (object) είναι το στιγμιότυπο μίας κλάσης, δηλαδή αυτή καθαυτή η δομή δεδομένων (με αποκλειστικά δεσμευμένο χώρο στη μνήμη) βασισμένη στο «καλούπι» που προσφέρει η κλάση. Παραδείγματος χάρη, σε μία αντικειμενοστρεφή γλώσσα προγραμματισμού θα μπορούσαμε να ορίσουμε κάποια κλάση ονόματι BankAccount, η οποία αναπαριστά έναν τραπεζικό λογαριασμό, και να δηλώσουμε ένα αντικείμενο της με όνομα MyAccount. Το

αντικείμενο αυτό θα έχει δεσμεύσει χώρο στη μνήμη με βάση τις μεταβλητές και τις μεθόδους που περιγράψαμε όταν δηλώσαμε την κλάση. Έτσι, στο αντικείμενο θα μπορούσε να περιέχεται ένα γνώρισμα Balance (υπόλοιπο) και μία μέθοδος GetBalance (επέστρεψε το υπόλοιπο). Ακολούθως θα μπορούσαμε να δημιουργήσουμε ακόμα ένα ή περισσότερα αντικείμενα της ίδιας κλάσης τα οποία θα είναι διαφορετικές δομές δεδομένων (διαφορετικοί τραπεζικοί λογαριασμοί στο παράδειγμα). Ας σημειωθεί εδώ πως τα αντικείμενα μίας κλάσης μπορούν να προσπελάσουν τα ιδιωτικά περιεχόμενα άλλων αντικειμένων της ίδιας κλάσης.

Ενθυλάκωση δεδομένων (data encapsulation) καλείται η ιδιότητα που προσφέρουν οι κλάσεις να «κρύβουν» τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμμα και να εξασφαλίζουν πως μόνο μέσω των δημόσιων μεθόδων τους θα μπορούν αυτά να προσπελαστούν. Αυτή η τακτική παρουσιάζει μόνο οφέλη καθώς εξαναγκάζει κάθε εξωτερικό πρόγραμμα να φιλτράρει το χειρισμό που επιθυμεί να κάνει στα πεδία μίας κλάσης μέσω των ελέγχων που μπορούν να περιέχονται στις δημόσιες μεθόδους της κλάσης.

Αφαίρεση δεδομένων καλείται η ιδιότητα των κλάσεων να αναπαριστούν αφαιρετικά πολύπλοκες οντότητες στο προγραμματιστικό περιβάλλον. Μία κλάση αποτελεί ένα αφαιρετικό μοντέλο κάποιας κατηγορίας αντικειμένων. Επίσης οι κλάσεις προσφέρουν και αφαίρεση ως προς τον υπολογιστή, εφόσον η καθεμία μπορεί να θεωρηθεί ένας μικρός και αυτάρκης υπολογιστής (με δική του κατάσταση, μεθόδους και μεταβλητές).

Κληρονομικότητα ονομάζεται η ιδιότητα των κλάσεων να επεκτείνονται σε νέες κλάσεις, ρητά δηλωμένες ως κληρονόμους (*υποκλάσεις* ή 'θυγατρικές κλάσεις'), οι οποίες μπορούν να επαναχρησιμοποιήσουν τις μεταβιβάσιμες μεθόδους και ιδιότητες της γονικής τους κλάσης αλλά και να προσθέσουν δικές τους. Στιγμιότυπα των θυγατρικών κλάσεων μπορούν να χρησιμοποιηθούν όπου απαιτούνται στιγμιότυπα των γονικών (εφόσον η θυγατρική είναι κατά κάποιον τρόπο μία πιο εξειδικευμένη εκδοχή της γονικής), αλλά το αντίστροφο δεν ισχύει. Παράδειγμα κληρονομικότητας είναι μία γονική κλάση Vehicle (Όχημα) και οι δύο πιο εξειδικευμένες υποκλάσεις της Car (Αυτοκίνητο) και Bicycle (Ποδήλατο), οι οποίες λέμε ότι

"κληρονομούν" από αυτήν. Πολλαπλή κληρονομικότητα είναι η δυνατότητα που προσφέρουν ορισμένες γλώσσες προγραμματισμού μία κλάση να κληρονομεί ταυτόχρονα από περισσότερες από μία γονικές. Από μία υποκλάση μπορούν να προκύψουν νέες υποκλάσεις που κληρονομούν από αυτήν, με αποτέλεσμα μία ιεραρχία κλάσεων που συνδέονται μεταξύ τους "ανά γενιά" με σχέσεις κληρονομικότητας.

Υπερφόρτωση μεθόδου (method overloading) είναι η κατάσταση κατά την οποία υπάρχουν, στην ίδια ή σε διαφορετικές κλάσεις, μέθοδοι με το ίδιο όνομα και πιθανώς διαφορετικά ορίσματα. Αν πρόκειται για μεθόδους της ίδιας κλάσης διαφοροποιούνται μόνο από τις διαφορές τους στα ορίσματα και στον τύπο επιστροφής.

Υποσκέλιση μεθόδου (method overriding) είναι η κατάσταση κατά την οποία μία θυγατρική κλάση και η γονική της έχουν μία μέθοδο ομώνυμη και με τα ίδια ορίσματα. Χάρη στη δυνατότητα του *πολυμορφισμού* ο μεταγλωττιστής «ξέρει» πότε να καλέσει ποια μέθοδο, βασισμένος στον τύπο του τρέχοντος αντικειμένου. Δηλαδή πολυμορφισμός είναι η δυνατότητα των αντικειμενοστρεφών μεταγλωττιστών να αποφασίζουν δυναμικά ποια είναι η κατάλληλη να κληθεί μέθοδος σε συνθήκες υποσκέλισης.

Αφηρημένη κλάση (abstract class) είναι μία κλάση που ορίζεται μόνο για να κληρονομηθεί σε θυγατρικές υποκλάσεις και δεν υπάρχουν δικά της στιγμιότυπα (αντικείμενα). Η αφηρημένη κλάση ορίζει απλώς ένα "συμβόλαιο" το οποίο θα πρέπει να ακολουθούν οι υποκλάσεις της όσον αφορά τις υπογραφές των μεθόδων τους (όπου ως υπογραφή ορίζεται το όνομα, τα ορίσματα και η τιμή επιστροφής μίας διαδικασίας). Μία αφηρημένη κλάση μπορεί να έχει και μη αφηρημένες μεθόδους οι οποίες υλοποιούνται στην ίδια την κλάση (αν και φυσικά μπορούν να υποσκελίζονται σε υποκλάσεις). Αντιθέτως οι αφηρημένες μέθοδοί της είναι απλώς ένας ορισμός της υπογραφής τους και εναπόκειται στις υποκλάσεις να τις υλοποιήσουν. Μία αφηρημένη κλάση που δεν έχει γνωρίσματα και όλες οι μέθοδοί της είναι αφηρημένες και δημόσιες καλείται **διασύνδεση** (interface). Οι κλάσεις που κληρονομούν από μία διασύνδεση λέγεται ότι την "υλοποιούν".

2.2 Εισαγωγή στα νήματα

Η διεργασία είναι κυρίως μια λογική έννοια, η οποία στην πράξη υλοποιείται με τον μηχανισμό της διεργασίας σε επίπεδο Λειτουργικού Συστήματος (ΛΣ). Μια λογική διεργασία μπορεί να αντιστοιχεί σε μια (ή περισσότερες) διεργασίες του ΛΣ. Τι γίνεται όμως αν μια λογική διεργασία χρειάζεται να υποστηρίξει «ταυτόχρονη» εκτέλεση κώδικα; Μπορεί να δημιουργηθούν πολλές διεργασίες (ΛΣ) που συνεργάζονται για τον κοινό σκοπό (της λογικής διεργασίας). Όμως ακόμα και αν οι διεργασίες δημιουργηθούν έτσι ώστε να επικοινωνούν αποδοτικά μέσω κοινής μνήμης, το σύστημα επιβαρύνεται λόγω των επιπρόσθετων εναλλαγών περιβάλλοντος λειτουργίας (context switch) που πραγματοποιούνται.

Στον προγραμματισμό, αρκετές φορές χρειάζεται να εκτελεστούν από μια εφαρμογή δύο ή περισσότερες λειτουργίες ταυτόχρονα. Οι περισσότερες γλώσσες προγραμματισμού, όπως και η C++ επιτρέπουν τον «παράλληλο προγραμματισμό». Η εκτέλεση δύο ή περισσοτέρων λειτουργιών παράλληλα επιτυγχάνεται με χρονομερισμό. Το λειτουργικό σύστημα καθορίζει χρονομερίδια και η κάθε λειτουργία μπορεί να αποκτά πρόσβαση στον επεξεργαστή και στα υπόλοιπα εξαρτήματα του υπολογιστή διαδοχικά για συγκεκριμένο χρόνο.

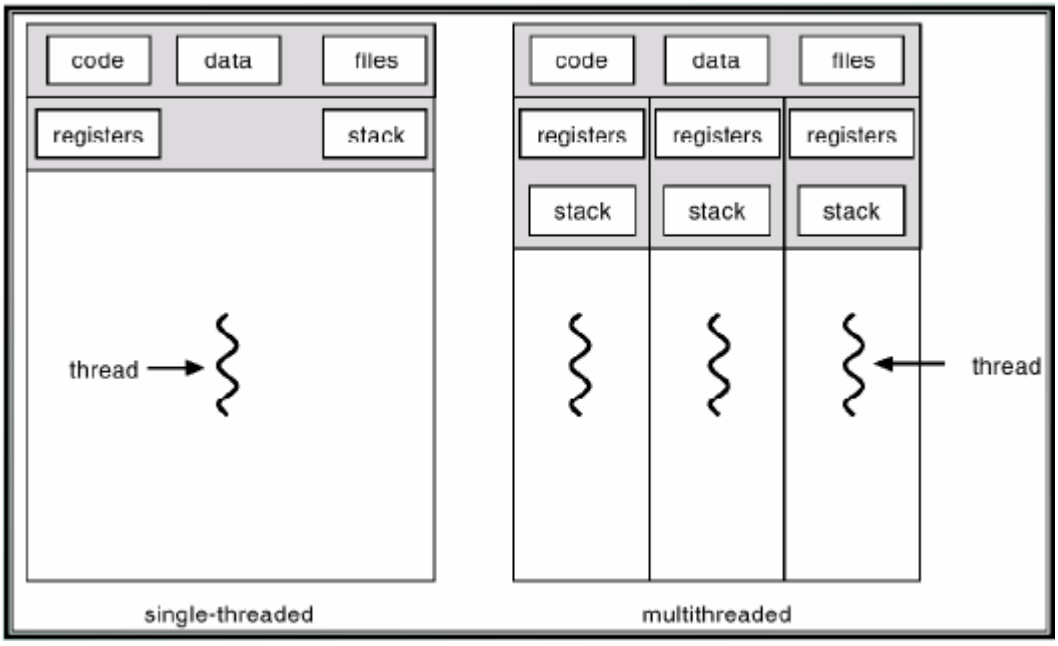
Στο λειτουργικό σύστημα τα νήματα και οι διεργασίες είναι οι μηχανισμοί που επιτρέπουν σε ένα πρόγραμμα να εκτελέσει ταυτόχρονα περισσότερες λειτουργίες. Στις διεργασίες, κάθε μια από αυτές έχει το δικό της χώρο διευθύνσεων και η θυγατρική διεργασία μπορεί να τροποποιήσει τα περιεχόμενα της μνήμης, να κλείσει περιγραφείς αρχείων κ.λπ. χωρίς να επηρεαστεί η μητρική διεργασία και το αντίστροφο.

Τα νήματα (νήματα ελέγχου), ονομάζονται και ελαφρές διεργασίες (lightweight processes – LWPs), είναι ξεχωριστές ροές του ίδιου προγράμματος που εκτελούνται μέσα σε μια λογική διεργασία. Τα νήματα μιας διεργασίας ανήκουν πάντα στον ίδιο χρήστη και λειτουργούν στην ουσία όπως και οι διεργασίες. Δεν είναι όμως πραγματικά ανεξάρτητες εκτελέσεις, καθώς:

- Έχουν τα ίδια δικαιώματα πρόσβασης σε πόρους
- Μοιράζονται τους ίδιους πόρους (δεδομένα, κώδικα, ανοικτά αρχεία, σήματα, ακόμα και τον χρόνο της ΚΜΕ)
- Μοιράζονται την ίδια μνήμη

Αν δηλαδή ένα νήμα τροποποιήσει την τιμή μιας μεταβλητής, το άλλο νήμα θα βλέπει τη νέα τιμή της μεταβλητής αυτής. Παρομοίως και για τα αρχεία: αν ένα νήμα ανοίξει ένα αρχείο, τότε όλα τα υπόλοιπα νήματα της εφαρμογής αποκτούν αμέσως πρόσβαση σε αυτό. Αυτό δε σημαίνει ότι κάθε νήμα δεν εκτελεί το δικό του κώδικα. Συνήθως κάθε νήμα περιέχει διαφορετικό κώδικα και για αυτό και έχει και τη δική του στοίβα.

Για κάθε νήμα διατηρείται ξεχωριστή κατάσταση εκτέλεσης, δηλαδή μετρητής προγράμματος, τιμές των καταχωρητών και στοίβα. Η εναλλαγή μεταξύ νημάτων (thread switch) ισοδυναμεί στην πράξη με απλή εναλλαγή κατάστασης ΚΜΕ (register set switch) και όχι με πραγματική εναλλαγή περιβάλλοντος λειτουργίας (context switch). Κατά το thread switch δεν πραγματοποιούνται λειτουργίες διαχείρισης μνήμης και δικαιωμάτων πρόσβασης, με αποτέλεσμα το thread switch να είναι πολύ πιο γρήγορο από το context switch. Οι εφαρμογές που είναι πολυνηματικές λέγονται και *multithreaded.*



Απεικόνιση Μονονηματικής και Πολυνηματικής εφαρμογής

Υλοποίηση Νημάτων σε Επίπεδο Χρήστη (User Threads)

Η διαχείριση των νημάτων γίνεται από βιβλιοθήκες που εκτελούνται σε κατάσταση χρήστη (user mode). Αποφεύγεται η επικοινωνία με το σύστημα/πυρήνα (δεν πραγματοποιούνται κλήσεις συστήματος). Τα νήματα εκτελούνται μέσα από μια κοινή διεργασία. Ένα νήμα αφήνει την ΚΜΕ με κλήση της ρουτίνας εναλλαγής, με λειτουργία που προκαλεί έμμεσα εναλλαγή ή με την λήξη ενός μετρητή. Συνήθως δεν υποστηρίζεται κατανομή του χρόνου εκτέλεσης της διεργασίας στα διάφορα νήματα που έχει μέσα της. Παραδείγματα τέτοιων νημάτων είναι τα: POSIX *pthreads,* Mach *C-threads,* Solaris *threads.*

> ***Πλεονεκτήματα:*** Τα νήματα χρήστη εναλλάσσονται (θεωρητικά) γρηγορότερα από τα νήματα πυρήνα. Ωστόσο, στην πράξη, καλές υλοποιήσεις νημάτων σε επίπεδο πυρήνα [Linux] εναλλάσσονται με συναφή απόδοση.
>
> ***Μειονεκτήματα:*** Ένα νήμα μπορεί να μονοπωλήσει τον χρόνο εκτέλεσης μιας διεργασίας (λιμοκτονία (starvation) των υπολοίπων νημάτων). Δεν γίνεται εκμετάλλευση της συμμετρικής πολυεπεξεργασίας. Επίσης, όταν μπλοκαριστεί ένα νήμα μέσα σε κάποια λειτουργία I/O τότε μπλοκάρονται όλα τα νήματα που εκτελούνται μέσα στην ίδια διεργασία (γιατί το ΛΣ δεν «γνωρίζει» την ύπαρξη τους).

Υλοποίηση Νημάτων σε Επίπεδο Συστήματος/Πυρήνα (Kernel Threads)

Τα νήματα πυρήνα υλοποιούνται στον πυρήνα του ΛΣ, και οι αντίστοιχες βιβλιοθήκες χρησιμοποιούν κλήσεις συστήματος. Σε αυτή την περίπτωση ο πυρήνας χρονοδρομολογεί κάθε νήμα εντός της μονάδας χρόνου που αναλογεί στην διεργασία μέσα στην οποία εκτελούνται τα νήματα. Υπάρχει περισσότερος φόρτος στο σύστημα λόγω της εναλλαγής κατάστασης χρήστη

– σύστημα (user mode - system mode) και την διαχείριση πιο πολύπλοκων περιβαλλόντων, αλλά οι αρχικές μετρήσεις απόδοσης δείχνουν αμελητέα αύξηση στο χρόνο. Παραδείγματα τέτοιων νημάτων είναι : Windows 95/98/NT/2000, Solaris, Tru64 UNIX, BeOS, Linux

Πλεονεκτήματα: Αποτροπή της μονοπώλησης της μονάδας χρόνου μιας διεργασίας από ένα νήμα (ενώ υπάρχουν και άλλα προς εκτέλεση). Το μπλοκάρισμα ενός νήματος σε I/O δεν συνεπάγεται μπλοκάρισμα και των άλλων νημάτων που εκτελούνται μέσα στην ίδια διεργασία. Μια διεργασία μπορεί να εκμεταλλευτεί πιο εύκολα τη συμμετρική πολυεπεξεργασία του ΛΣ και να τρέχει γρηγορότερα με κάθε ΚΜΕ που προστίθεται στο σύστημα.

Μειονεκτήματα: Μικρή ταχύτητα εναλλαγής των νημάτων.

2.3 Χρήση προγραμματιστικών νημάτων

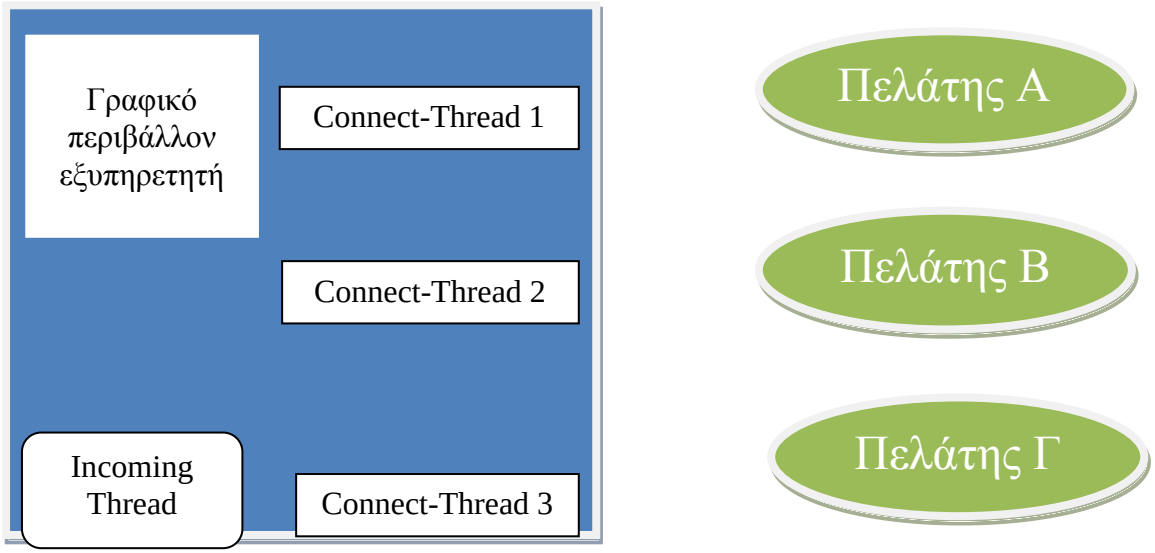
Θεωρώντας ότι η εφαρμογή του πελάτη όσο και η εφαρμογή του εξυπηρετητή θα διαθέτουν πλούσιο γραφικό περιβάλλον (*rich user interface*) και παράλληλα θα εκτελούν κώδικα για την μεταξύ τους επικοινωνία, αμέσως καταλαβαίνουμε ότι και οι δύο εφαρμογές θα είναι *multihtreaded*.

Ας πάρουμε για αρχή την εφαρμογή του εξυπηρετητή. Πρώτα από όλα μια τέτοια εφαρμογή πρέπει να υλοποιεί ένα κύριο «νήμα», ένα thread δηλαδή που διαρκώς «ακούει» μια συγκεκριμένη θύρα (μια «πόρτα») για εισερχόμενες κλήσεις. Όταν ένας πελάτης προσπαθεί να συνδεθεί με τον εξυπηρετητή, τότε το κύριο αυτό νήμα θα πρέπει να αντιλαμβάνεται την επιθυμία του και δημιουργεί ένα νέο νήμα για κάθε πελάτη.

Δίνοντας ένα παράδειγμα, όταν ο εξυπηρετητής τρέξει για πρώτη φορά και εμφανιστεί το γραφικό περιβάλλον τότε υπάρχουν σε λειτουργία δύο νήματα. Ένα για το γραφικό περιβάλλον ώστε να ανταποκρίνεται στο χειρισμό του χρήστη, το οποίο θα προέρχεται από τη βιβλιοθήκη που θα χρησιμοποιηθεί για το γραφικό περιβάλλον και ένα που διαρκώς «ακούει» για νέες συνδέσει.

Όταν υπάρχει μια εισερχόμενη τότε δημιουργείται νέο νήμα για να εξυπηρετεί το συγκεκριμένο πελάτη.

Η εφαρμογή δηλαδή είναι multi-threaded και έχει αριθμό threads που τρέχουν ταυτόχρονα. Ας ονομάσουμε το thread των γραφικών *UserInterface-Thread*, το *thread* που ακούει για εισερχόμενες *Incoming-Thread* και τα thread για εξυπηρέτηση πελατών *Connect-Thread*.



Απόδοση thread σε κάθε πελάτη (client)

Στην παραπάνω εικόνα ο εξυπηρετητής έχει τρείς πελάτες συνδεδεμένους και τρέχει 5 νήματα ταυτόχρονα. Ένα *Connect-Thread* για να εξυπηρετεί κάθε πελάτη, ένα Incoming-Thread για να «ακούει» για νέους πελάτες και ένα *UserInteface-Thread* για να κρατάει ζωντανό το User Interface της εφαρμογής.

2.4 Εισαγωγή στην TCP επικοινωνία

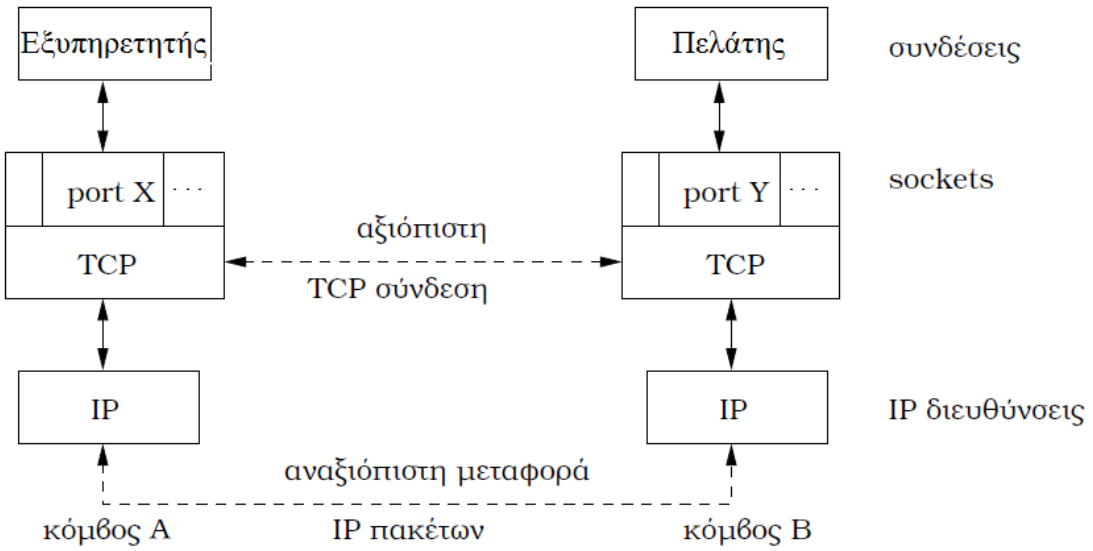
Το Πρωτόκολλο Ελέγχου Μεταφοράς TCP (Transmission Control Protocol), σχεδιάστηκε ειδικά για να προσφέρει έναν αξιόπιστο συρμό byte από άκρο σε άκρο, μέσω ενός αναξιόπιστου διαδικτύου (IP network layer). Το TCP επιτρέπει σε δύο δικτυακές οντότητες να δημιουργήσουν μεταξύ τους ένα

αμφίδρομο κανάλι επικοινωνίας για να μπορούν να διαβάζουν και να γράφουν ακολουθίες από bytes. Για παράδειγμα ένας web browser επικοινωνεί με έναν web server μέσω του TCP με σκοπό να στείλει το αίτημα του για μία συγκεκριμένη ιστο-σελίδα και να λάβει το περιεχόμενο της. Το πρωτόκολλο εγγυάται ότι τα bytes θα παραδοθούν στον αποδέκτη με την σειρά που εστάλησαν ή καθόλου. Αυτό απλοποιεί την λειτουργία των εφαρμογών αφού αν υπάρχει πρόβλημα στο δίκτυο θα έχουν να κάνουν με μία κατάσταση παντελούς έλλειψης επικοινωνίας και όχι με μερικά κατεστραμμένα ή εκτός σειράς παραδοτέα δεδομένα. Η δομή και το νόημα των bytes που ανταλλάζονται καθορίζεται από το στρώμα εφαρμογής και στην περίπτωση του web browser από το πρωτόκολλο HTTP (Hyper-Text Transfer Protocol).

Το TCP παρέχει δύο πολύ σημαντικές δικτυακές λειτουργίες:

α) *Έλεγχο Συμφόρησης (congestion control*). Με αυτή την λειτουργία ελέγχει τον ρυθμό με τον οποίο τα δεδομένα δίνονται στο δίκτυο ανάλογα με την «κατάσταση» του δικτύου. Ο κατανεμημένος αλγόριθμος που χρησιμοποιείται εγγυάται ότι δύο ανταγωνιστικά TCP streams θα έχουν ίση μεταχείριση του εύρους ζώνης του δικτύου.

β) *Έλεγχο Ροής (flow control*). Με αυτή την λειτουργία επιτρέπει στον αποδέκτη κάθε αμφίδρομης επικοινωνίας να ελέγχει-ρυθμίζει τον ρυθμό με τον οποίο ο αποστολέας στέλνει τα δεδομένα.



Ένα διαδίκτυο διαφέρει από ένα απλό δίκτυο, επειδή τα διαφορετικά μέρη του μπορεί να έχουν διαφορετικές τοπολογίες, διαφορετικό εύρος ζώνης, διαφορετικές καθυστερήσεις, διαφορετικά μεγέθη πακέτων και άλλες διαφορετικές παραμέτρους. Το TCP χτίζεται πάνω από το IP και έχει σχεδιαστεί έτσι ώστε να προσαρμόζεται δυναμικά στις ιδιότητες του διαδικτύου και να είναι ανθεκτικό ως προς πολλά είδη προβλημάτων μετάδοσης-λήψης πακέτων. Έτσι ενώ τα IP πακέτα μπορεί να χαθούν, να φτάσουν εκτός σειράς ή να έχουν σωστό header αλλά κατεστραμμένο payload, το TCP λύνει όλα τα παραπάνω προβλήματα με τις εξής λειτουργίες:

- *Stream segmentation:* Επειδή το IP υποστηρίζει την μεταφορά πακέτων περιορισμένου μεγέθους, το TCP σπάει το stream σε segments κατάλληλα για μετάδοση.

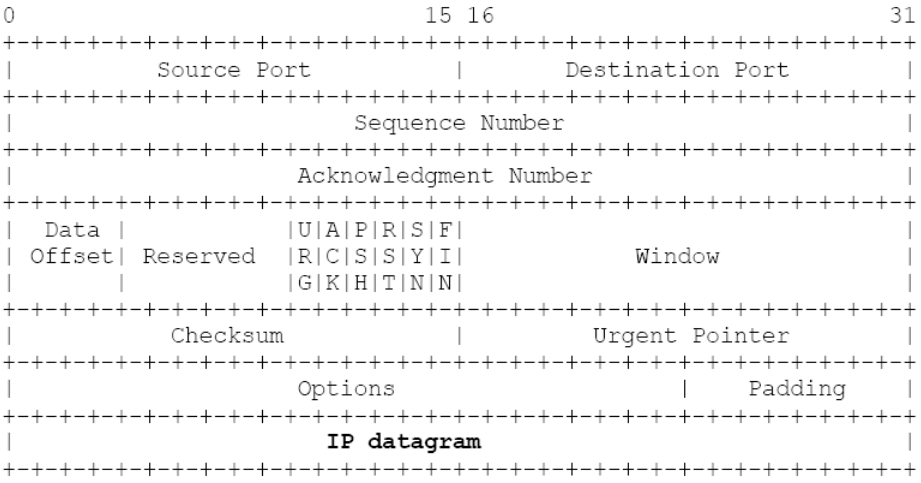
- *Stream reassembly:* Τα TCP stream segments μεταδίδονται σαν IP πακέτα τα οποία μπορεί να φτάσουν στον προορισμό τους με διαφορετική σειρά από αυτή που μεταδόθηκαν. Το TCP πρέπει να είναι ικανό να χειριστεί τα εκτός σειράς πακέτα και ακόμα να συναρμολογεί τα δεδομένα με τη σειρά που στάλθηκαν. Το TCP επιλύει αυτό το πρόβλημα μετρώντας τον αριθμό των bytes που στάλθηκαν στο συγκεκριμένο stream και αριθμώντας κάθε TCP/IP πακέτο.

- *Packet loss handling:* Τα TCP/IP πακέτα κατά την διάρκεια της μετάδοσης τους μπορεί να χαθούν. Το TCP πρέπει να είναι ικανό να διακρίνει ότι ένα πακέτο χάθηκε και να κανονίσει την αναμετάδοση του. Το TCP επιλύει αυτό το θέμα με το να στέλνει ο δέκτης θετικά πακέτα επιβεβαίωσης.

- *Data corruption detection:* Το IP προστατεύει μονό το δικό του header και δεν παρέχει καμία εγγύηση για τα δεδομένα. Το TCP επιλύει αυτό το πρόβλημα υπολογίζοντας ένα άθροισμα των δεδομένων του TCP πακέτου και το αποθηκεύει στο TCP header.

- *Throughput efficiency:* Το TCP χρησιμοποιεί την τεχνική sliding-window ώστε ο δέκτης να πληροφορεί τον αποστολέα σχετικά με το μέγιστο αριθμό ανεπιβεβαίωτων bytes που μπορούν να μεταδοθούν (το window). Μόλις ο αποστολέας στείλει δεδομένα ίσο με το μέγεθος του window πρέπει να περιμένει για επιβεβαίωση. Όταν η επιβεβαίωση παραληφθεί ο αποστολέας μπορεί να στείλει τόσα δεδομένα όσα επιβεβαιώθηκαν. Αυτός ο μηχανισμός επιτρέπει στους δέκτες να ελέγχουν την ροή της μετάδοσης.

Για να μπορέσουν να υποστηριχτούν οι παραπάνω λειτουργίες μίας TCP σύνδεσης, τα δύο άκρα πρέπει να διατηρούν διάφορους μετρητές. Επιπλέον η κατάσταση ορισμένων μετρητών απαιτεί την συνεργασία των δύο άκρων. Συνεπώς τα δύο άκρα ανταλλάσουν πληροφορίες ελέγχου χρησιμοποιώντας την καθορισμένου μεγέθους TCP header (20-byte) η οποία προσαρτάται σε κάθε IP πακέτο που μεταφέρει TCP δεδομένα.



Διάγραμμα IP

Σε μία TCP σύνδεση για να υποστηριχτούν η συναρμολόγηση ενός stream καθώς και τα πακέτα επιβεβαίωσης, τα δύο άκρα πρέπει να συμφωνήσουν για το initial sequence number με τον οποίο γίνεται διακριτή η εκάστοτε σύνδεση από προηγούμενες ενεργές ή όχι συνδέσεις.

Η διαδικασία εγκατάστασης μιας TCP σύνδεσης γίνεται με την μέθοδο της τριπλής χειραψίας(three-way handshake). Ο πελάτης στέλνει το αίτημα σύνδεσης στον εξυπηρετητή (server) μέσω ενός TCP/IP πακέτου **SYN**

(synchronization), το οποίο δεν περιλαμβάνει δεδομένα αλλά μόνο τα TCP, IP headers και το initial sequence number. Εν συνεχεία ο εξυπηρετητής στέλνει το αντίστοιχο SYN πακέτο καθώς και το πακέτο επιβεβαίωσης. Η διαδικασία τερματισμού μιας TCP σύνδεσης γίνεται από κάθε άκρο ξεχωριστά στέλνοντας ένα TCP πακέτο το FIN όπου ακολουθείται από το πακέτο επιβεβαίωσης.

2.5 TCP Sockets

Για να αποκτηθεί η υπηρεσία TCP, ο αποστολέας και ο παραλήπτης δημιουργούν ακραία σημεία που αποκαλούνται υποδοχές (sockets). Η υποδοχή είναι μια αφηρημένη προγραμματιστική έννοια και παρέχει στα προγράμματα το κατάλληλο interface για να γράψουν/διαβάσουν δεδομένα. Σαν αποτέλεσμα τα προγράμματα δεν ενδιαφέρονται για το πώς τα δεδομένα πραγματικά μεταδίδονται στο δίκτυο.

Κάθε socket έχει έναν αριθμό υποδοχής (διεύθυνση) που αποτελείται από την διεύθυνση IP του host και από έναν τοπικό αριθμό 16 bit για κάθε host, αποκαλούμενο θύρα (port).Κάθε υπηρεσία δικτύου συνδέεται (is bound to) με μια ή περισσότερες θύρες σε ένα ή περισσότερα IP interfaces της κάθε συσκευής του host. Το πρωτόκολλο TCP χρησιμοποιεί αυτόν τον μηχανισμό ώστε να γίνονται διακριτές οι υπηρεσίες δικτύου σε κάθε host .

Stream Sockets

Το παρακάτω διάγραμμα δείχνει την χρήση των sockets ώστε να επιτευχθεί μία αμφίδρομη byte-streaming επικοινωνία. Από την δεξιά μεριά του διαγράμματος υπάρχει ένα πρόγραμμα πελάτης που επιθυμεί να συνδεθεί με τον server που βρίσκεται στα αριστερά. Για να μπορέσει ο server να δεχτεί απομακρυσμένες συνδέσεις πρέπει πρώτα να δημιουργήσει ένα server socket. Ένα server socket είναι ένας δέκτης για εισερχόμενες αιτήσεις σύνδεσης.

Τα στάδια είναι τα εξής :

1) Το πρόγραμμα server δημιουργεί ένα TCP server socket και δεσμεύει (bound) την θύρα 4567.

2) Σε αυτό το σημείο ο server ενεργοποιεί την μέθοδο accept() από το TCP server socket και περιμένει-ακούει για συνδέσεις πελατών.

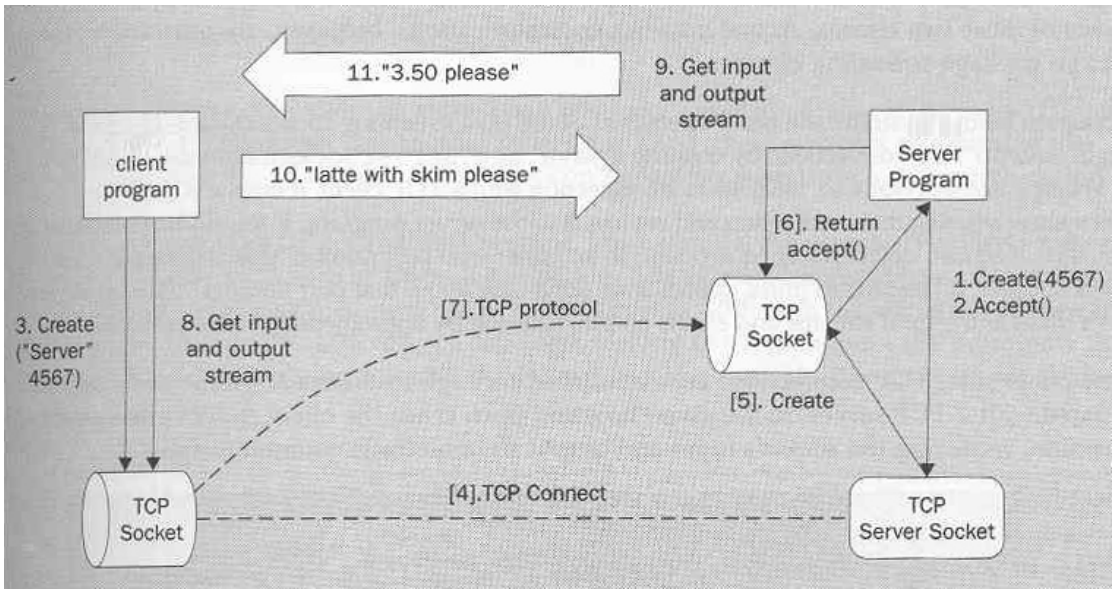
3) Από την άλλη μεριά ο πελάτης δημιουργεί ένα socket με παραμέτρους το όνομα ή την διεύθυνση του server καθώς και την θύρα που δέχεται ο server. Ο τύπος του πρωτόκολλου του client socket πρέπει να ταιριάζει με αυτόν του server socket. Στη συγκριμένη περίπτωση είναι το TCP.

4) Μετά την δημιουργία του το αντικείμενο socket προσπαθεί να δημιουργήσει-εγκαταστήσει σύνδεση μεταξύ των δύο άκρων χρησιμοποιώντας το TCP πρωτόκολλο.

5) Μόλις η TCP χειραψία ολοκληρωθεί, το server socket δημιουργεί ένα socket που προσομοιώνει την πλευρά του server στην TCP επικοινωνία .

6) Αυτό είναι το αποτέλεσμα της μπλοκαρισμένης μεθόδου accept()

Σε αυτή την φάση ο client και ο server κατέχουν από ένα socket αντικείμενο ο καθένας. Η ασυμμετρία της προηγούμενης φάσης για προσπάθεια σύνδεσης τελείωσε και τώρα οι δυο πλευρές μπορούν να επικοινωνήσουν πάνω από μία αμφίδρομη διασύνδεση. Για να μπορέσουν να ανταλλάξουν/μεταδώσουν πληροφορίες πρέπει πρώτα το κάθε άκρο να αποκτήσει ένα input και output stream από το socket αντικείμενο (Βήματα 8 και 9). Το περιεχόμενο της μετέπειτα συνομιλίας είναι σχετικό με την εκάστοτε εφαρμογή.



Χρήση Sockets για byte-streaming επικοινωνία

2.6 HTML Tags

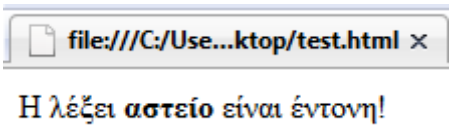
Η HTML (HyperText Markup Language) είναι μια περιγραφική γλώσσα που χρησιμοποιώντας ορισμένα tags μπορούμε να ορίσουμε πως θα εμφανίζετε κάποιο κείμενό μέσα στον φυλλομετρητή (Web Browser) ενός επισκέπτη της ιστοσελίδας.

Τα HTML tags αποτελούν σύντομες «εντολές» οι οποίες τοποθετούνται στο κείμενο μιας σελίδας HTML. Οι εντολές αυτές αποτελούνται από ζευγάρια. Το πρώτο tag του ζευγαριού εμπεριέχεται μέσα στα σύμβολα < και >, ενώ το δεύτερο ανάμεσα στα σύμβολα </ και >.

Για παράδειγμα για να γίνει χρήση του HTML tag **B** το οποίο μετατρέπει το κείμενο που εμπεριέχει σε Bold – έντονη μορφή θα χρησιμοποιηθεί η παρακάτω σύνταξη.

Η λέξει ****αστείο**** είναι έντονη!

Ένα αρχείο HTML που περιέχει το παραπάνω tag θα εμφανιστεί στο χρήστη ως εξής:



Η κάθε σελίδα HTML έχει μια ορισμένη δομή. Συγκεκριμένα tag πρέπει να χρησιμοποιηθούν σε συγκεκριμένα σημεία. Ο σκελετός δηλαδή ενός HTML αρχείου είναι ο εξής

```
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Μέσα στο tag “head” τοποθετούνται πληροφορίες σχετικά με το περιεχόμενο του αρχείου, όπως σε ποια γλώσσα είναι γραμμένο το κείμενο κ.α. Μέσα στο tag “body” τοποθετείται το κύριο σώμα του εγγράφου.

Τα διαθέσιμα HTML tags που υποστηρίζονται είναι τα εξής:

ADDRESS - Address information	**INPUT - Input field, button, etc.**
APPLET - Java applet	**ISINDEX - Primitive search**
AREA - Hotzone in imagemap	**I - Italics**
A - Anchor	**KBD - Keyboard input**
BASE - Document location	**LINK - Site structure**
BASEFONT - Default font size	**LI - List item**
BIG - Larger text	**MAP - Client-side imagemap**
BLOCKQUOTE - Large quotation	**MENU - Menu item list**
BODY - Document body	**META - Meta-information**
	DIV - Logical division

BR - Line break

B - Bold

CAPTION - Table caption

CENTER - Centered division

CITE - Short citation

CODE - Code fragment

DD - Definition

DFN - Definition of a term

DIR - Directory list

DL - Definition list

DT - Definition term

EM - Emphasized text

FONT - Font modification

FORM - Input form

H1 - Level 1 header

H2 - Level 2 header

H3 - Level 3 header

H4 - Level 4 header

H5 - Level 5 header

H6 - Level 6 header

HEAD - Document head

HR - Horizontal rule

HTML - HTML Document

IMG - Images

OL - Ordered list

OPTION - Selection list option

PARAM - Parameter for Java applet

PRE - Preformatted text

P - Paragraph

SAMP - Sample text

SCRIPT - Inline script

SELECT - Selection list

SMALL - Smaller text

STRIKE - Strikeout

STRONG - Strongly emphasized

STYLE - Style information

SUB - Subscript

SUP - Superscript

TABLE - Tables

TD - Table cell

TEXTAREA - Input area

TH - Header cell

TITLE - Document title

TR - Table row

TT - Teletype

UL - Unordered list

U - Underline

VAR - Variable

3. Μελέτη και σχεδιασμός του συστήματος

Στην πτυχιακή αυτή εργασία, λέγοντας **Chat Server** θα εννοούμε έναν πολύνυματικό (*multithreaded*) εξυπηρετητή που διαχειρίζεται τις συνδέσεις των πελατών και τη μεταφορά δεδομένων ανάμεσά τους. Ως **Chat Client** θα εννοούμε την εφαρμογή του πελάτη που θα χρησιμοποιείται για τη σύνδεση στο εξυπηρετητή και την ομιλία με άλλους χρήστες της πλατφόρμας.

Σχεδιαστικά η πιο απλοποιημένη μορφή *Client/Server chat* θα ήταν η διασύνδεση των χρηστών στον εξυπηρετητή και κατόπιν η μεταφορά των κειμένων που στέλνει ο κάθε χρήστης σε όλους τους υπολοίπους. Με τη σταδιακή εξοικείωση με των κώδικα και τις απαραίτητες βιβλιοθήκες, πάρθηκε η απόφαση να προστεθούν περαιτέρω δυνατότητες στο αρχικό μοντέλο.

3.1 Πρωτόκολλο Internet Relay Chat

Κατόπιν μελέτης του πρωτοκόλλου *Internet Relay Chat* (IRC) [1] καθώς και χρήση λογισμικών όπως το *MIRC* [2] πέρα από την απλή επικοινωνία των χρηστών, παρέχεται η δυνατότητα για εκτέλεση εντολών. Ο αριθμός των εντολών και των δυνατοτήτων του πρωτοκόλλου αυτού είναι πραγματικά μεγάλος και η πλήρης υλοποίησή του θα ήταν πραγματικά δύσκολη και χρονοβόρα. Επιλέχθηκε, λοιπόν, ο ορισμός ενός *subset* του πρωτοκόλλου και η υποστήριξη ορισμένων εντολών από τη πλατφόρμα που θα αναπτυχθεί.

Ενδεικτικά, το πρωτόκολλο IRC ορίζει την υποστήριξη δωματίων επικοινωνίας. Κάθε χρήστης μπορεί να δημιουργήσει ένα δωμάτιο στο οποίο ορίζει το όνομά του και τη θεματολογία του. Χρήστες που επιθυμούν να συνομιλήσουν με άλλους χρήστες, για το συγκεκριμένο θέμα, μπορούν να μπούν στο δωμάτιο αυτό, να συνομιλήσουν και όταν επιθυμούν να βγούν από το δωμάτιο αυτό. Μάλιστα, ένας χρήστης είναι δυνατό να βρίσκεται ταυτόχρονα σε περισσότερα από ένα δωμάτια. Παράλληλα μπορεί να συνομιλήσει ιδιωτικά με κάποιον άλλο χρήστη, αποστέλλοντας μόνο σε αυτόν συγκεκριμένα μηνύματα. Οι παραπάνω δυνατότητες κρίθηκαν αρκετά

ενδιαφέρουσες και αποφασίστηκε να υλοποιηθούν και στην πλατφόρμα που θα παρουσιαστεί στη συνέχεια.

Βέβαια το πρωτόκολλο IRC ορίζει πολλές ακόμα δυνατότητες για τους χρήστες του, οι σημαντικότερες από τις οποίες είναι:

- Μεταφορά αρχείων μεταξύ χρηστών
- Διατήρηση *NickName* (ονόματος χρήστη) με χρήση κωδικού πρόσβασης
- Ύπαρξη ενός ή περισσοτέρων διαχειριστών για κάθε δωμάτιο. Οι διαχειριστές μπορούν να έχουν διαφορετικά δικαιώματα και να χρησιμοποιούν μια ιεραρχία. Έχουν δικαιώματα να διώξουν χρήστες από ένα δωμάτιο (*kick*), αν για παράδειγμα κρίνουν ότι δεν συνομιλούν με πολιτισμένο τρόπο. Έχουν δικαίωμα να απαγορέψουν την είσοδο χρηστών για ορισμένο χρόνο, τιμωρώντας τους για λίγα λεπτά ή ακόμα και αρκετές ώρες (*ban*), για τυχόν παραβάσεις των κανόνων που ορίζουν. Μπορούν να προάγουν απλούς χρήστες σε διαχειριστές ή να αφαιρέσουν το δικαίωμα ενός διαχειριστή, που είναι χαμηλότερα από αυτούς στην ιεραρχία του δωματίου συζήτησης.
- Εκτέλεση εντολών για τον έλεγχο της ποιότητας επικοινωνίας σε δικτυακό επίπεδο ενός πελάτη με τον εξυπηρετητή (*ping*).
- Εκτέλεση εντολών για την ανεύρεση του τρόπου σύνδεσης του πελάτη (από πιο *Internet IP Address* συνδέεται με τον εξυπηρετητή) όπως η *whois*.

Επίσης στο πρωτόκολλο του IRC προβλέπεται επικοινωνία μεταξύ *servers* και μηχανισμοί για την ασφάλεια του συστήματος και προάσπισή του από κακόβουλες επιθέσεις.

3.2 Επιπρόσθετες δυνατότητες

Emoticons

Εκτός του πρωτοκόλλου, που δε θα περιγραφεί στο πλήρες μέγεθός του, επιπρόσθετες δυνατότητες παρέχουν τα λογισμικά που έχουν αναπτυχθεί για τη χρήση του. Κάτι που παρατηρείται τόσο σε *Chat Clients* του IRC όσο και το *MSN Messenger* [3] και το ICQ [4], είναι η γραφική αναπαράσταση ορισμένων συντομογραφιών σε αντίστοιχα *Icons*. Τα σύμβολα αυτά ονομάζονται *emoticons*, και ενώ μεταφέρονται από το δίκτυο ως απλό κείμενο, ο *Chat Client* τα μεταφράζει σε εικόνες. Το παραπάνω *feature* αποφασίστηκε να υλοποιηθεί και στις εφαρμογές που αναπτύξαμε. Συγκεκριμένα τα παρακάτω **emoticons** υλοποιήθηκαν, και η προσθήκη περισσοτέρων θα είναι δυνατή.

- :)
- :d ή :D
- :P
- :(
- :s
- ;)
- :o
- :|

Χρωματισμένο κείμενο

Πέρα από τα emoticons, μια ακόμα ενδιαφέρον δυνατότητα είναι η αποστολή και παρουσίαση **χρωματισμένου κειμένου**. Η τεχνική αυτή μπορεί να υλοποιηθεί προγραμματιστικά στέλνοντας HTML tags που χαρακτηρίζουν την υφή και το χρώμα του κειμένου.

Γνωρίζουμε για παράδειγμα ότι στην HTML ο κώδικας

Κάποιο κείμενο

Μεταφράζεται ως έντονη γραφή ή αλλιώς bold από τους φυλλομετρητές. Αντίστοιχα μεταφράζονται tag για την υπογράμμιση (Underlying) με τα tags <u>..</u>, η πλάγια γραμμή (Italics) με τα tags ... Η χρήση αντίστοιχων tags για μορφοποίηση κειμένου ή για χρωματισμό τους θα μπορούσε να χρησιμοποιηθεί επίσης.

3.3 Πρωτόκολλο συστήματος

Μετά από τη μελέτη του πρωτοκόλλου του IRC αλλά και εφαρμογών πελάτη τόσο για το IRC όσο άλλων δημοφιλών *chat clients* όπως το MSN και το ICQ αποφασίστηκε να υλοποιηθούν ορισμένες εντολές που θα μπορεί να χρησιμοποιεί ο πελάτης, για να εκμεταλλευτεί ιδιαίτερες λειτουργίες της εφαρμογής. Επιγραμματικά οι εντολές αυτές παρουσιάζονται παρακάτω:

Εντολή	Σύντομη περιγραφή
/Room_List	Εμφανίζει στο χρήστη τα υπάρχοντα δωμάτια επικοινωνίας.
/Create_Room RoomName	Δημιουργεί ένα νέο δωμάτιο επικοινωνίας με την ονομασία RoomName όπου χρήστες μπορούν να μπούνε για να συνομιλήσουν με άτομα κοινού ενδιαφέροντος.
/RoomTopic topic	Με την εντολή αυτή ορίζεται η θεματολογία του δωματίου.
/Join_Room RoomName	Τοποθετεί το χρήστη μέσα στο δωμάτιο επικοινωνίας με την ονομασία RoomName. Προϋποθέτει το δωμάτιο αυτό να έχει ήδη δημιουργηθεί. Το νέο δωμάτιο επικοινωνίας θα εμφανίζεται σε μια νέα καρτέλα στο γραφικό περιβάλλον της εφαρμογής.
/Leave_Room	Με την εντολή αυτή ο χρήστης θα μπορεί να βγαίνει από ένα δωμάτιο επικοινωνίας. Η εντολή αυτή θα πρέπει να δοθεί στην καρτέλα του αντίστοιχου δωματίου στο γραφικό περιβάλλον της εφαρμογής.

Εντολή	**Σύντομη περιγραφή**
/msg UserName	Με την εντολή αυτή δημιουργείται μια ιδιωτική συνομιλία μεταξύ του χρήστη που εκτελεί την εντολή αυτή και του χρήστη με την ονομασία UserName. Προϋποθέτει την ύπαρξη χρήστη με το όνομα αυτό. Στο γραφικό περιβάλλον θα ανοίγει μια νέα καρτέλα - tab για να περιέχει τα μηνύματα που θα ανταλλαχτούν στη νέα αυτή ιδιωτική συνομιλία.

Προγραμματιστικά, οι παραπάνω πέντε εντολές πρέπει να αναλυθούν και να ορισθεί ο τρόπος με τον οποίο θα μεταφέρονται, ώστε να είναι κατανοητές τόσο από την εφαρμογή των πελατών όσο και από την εφαρμογή του εξυπηρετητή. Παρακάτω ακολουθεί μια αναλυτικότερη παρουσίαση του πρωτοκόλλου:

3.3.1 Room_List

Για την εντολή των χρηστών :

/Room_List

Θα αποστέλλεται με χρήση TCP Socket οι ακριβείς παρακάτω χαρακτήρες:

<CS>/Room_List<CE>

3.3.2 Join_Room

Για την εντολή ενός χρήστη:

/Join_Room room

Θα αποστέλλεται με χρήση TCP Socket το παρακάτω κείμενο, που θα περιέχει πέρα από τις παραπάνω πληροφορίες και το όνομα χρήστη - nick name.

<CS>/Join_Room:user:RoomName<CE>

Επίσης κατά την εκκίνηση της εφαρμογής του πελάτη και κατά την πρώτη σύνδεση θα αποστέλλεται το μήνυμα

<CS>/Join_Room:user:admin<CE>

Αυτό συμβαίνει ώστε ο εξυπηρετητής να μπορεί να καταλάβει και να αποθηκεύσει σε κάποια μεταβλητή ότι ο χρήστης με τη συγκεκριμένη ονομασία αντιστοιχεί σε συγκεκριμένο thread ID.

3.3.3 Create_Room

Για την εντολή ενός χρήστη:

/Create_Room room

Θα αποστέλλεται το:

<CS>/Create_Room:room<CE>

Ο εξυπηρετητής λαμβάνοντας το μήνυμα αυτό θα δημιουργεί ένα νέο νήμα (Thread), θα το εκτελεί και θα αποθηκεύει το mapping του ονόματος δωματίου με το αντίστοιχο index(m_threadIDs).

3.3.4 RoomTopic

Ο χρήστης του Chat Client θα μπορεί να ορίζει τη θεματολογία του δωματίου με την παρακάτω εντολή:

/RoomTopic topic

Το μήνυμα που θα αποστέλλεται θα είναι το

<CS>/RoomTopic:room:topic<CE>

Και ο εξυπηρετητής θα φροντίζει να αλλάξει το όνομα του δωματίου επικοινωνίας στο εσωτερικό του mapping, αλλά και το κάθε chat client θα αλλάζει την ονομασία της καρτέλας που περιέχει το συγκεκριμένο δωμάτιο.

3.3.5 Msg

Όταν ένας χρήστης επιθυμεί να ξεκινήσει μια ιδιωτική συζήτηση με ένα άλλο χρήστη θα χρησιμοποιεί την παρακάτω εντολή

/msg user

Η εντολή αυτή θα μεταφέρεται ως

<CS>/msg User<CE>

Και θα δημιουργείται ένα mapping στον εξυπηρετητή μεταξύ δύο χρηστών στο server. Στο γραφικό περιβάλλον των εφαρμογών θα ανοίγει μια νέα καρτέλα.

Όταν οι δύο αυτοί χρήστες ανταλλάσσουν μηνύματα, αυτά θα μεταφέρονται με το παρακάτω format:

<CS>PMSG:ToUser:private message<CE>

Έτσι ώστε ο εξυπηρετητής να καταλαβαίνει ότι το μήνυμα πρέπει να αποσταλεί μόνο σε συγκεκριμένο παραλήπτη.

3.3.6 Leave_Room

Όταν ένας χρήστης θέλει να εγκαταλείψει ένα δωμάτιο επικοινωνίας θα χρησιμοποιεί την εντολή:

/Leave_Room room

Η οποία θα μεταφέρεται στον εξυπηρετητή ως:

<CS>/Leave_Room:Room<CE>

Ως αποτέλεσμα, ο εξυπηρετητής θα αφαιρεί από το συγκεκριμένο threadID από τη λίστα των παραληπτών των μηνυμάτων και στη συνέχεια θα αποστέλλει το παρακάτω μήνυμα σε κάθε συνδεδεμένο πελάτη ώστε και αυτός να ενημερωθεί για την αποχώρηση του συγκεκριμένου χρήστη.

<CS>USRLFT:UserName<CE>

3.3.7 Μηνύματα επικοινωνίας

Τα Chat μηνύματα που στέλνει ένας χρήστης σε ένα συγκεκριμένο δωμάτιο θα μεταφέρονται ως εξής:

<DS>ROOM:User:Message<DE>

Όταν λοιπόν ένας χρήστης **User** στείλει έναν μήνυμα **Message** σε ένα δωμάτιο **Room**, το αντίστοιχο μήνυμα θα αποστέλλεται από τον πελάτη στον εξυπηρετητή με τον τρόπο που προβλέπεται. Ο εξυπηρετητής στη συνέχεια θα αποστέλλει το παραπάνω μήνυμα στο client-thread. Το νήμα αυτό θα αναλαμβάνει να το μεταδώσει σε κάθε πελάτη που βρίσκεται μέσα σε αυτό το δωμάτιο επικοινωνίας.

Όταν ο χρήστης κάνει /Room_list τότε ο εξυπηρετητής τους απαντά με το εξής μήνυμα :

<CS>RLIST:comma seperated room list<CE>
<CS>RLST:RoomName:room_user_list<CE>
<CS>TPC:ROOMNAME:TOPIC<CE>
<CS>PMSG:FROMUSER:private message<CE>
<CS>NSR:room name<CE> //no such room(whose /leave room was sent)
<CS>LFT:room name<CE> //following room was successfully left
<CS>CON:room name<CE> //connected to the room (in response to

/Join_Room) so open a new tab for it

<CS>msg:Requesting user<CE> //open a new tab upon receiving this

//will be received by the tab on the server and the client whenever a new user connects to that room

<CS>USRLFT:Room name:UserName<CE>

//will be received by the server tabs and the clients

//since the user needs to know which user sent this private message

//since a single client might be having private chats with more than one user

Chat messages sent to the client

<DS>ROOM:User:Message<DE>

3.4 Περιπτώσεις χρήσης και διαγράμματα δραστηριοτήτων

Οι περιπτώσεις χρήσης της εφαρμογής μας παρουσιάζονται ονομαστικά στον παρακάτω πίνακα:

A/A	Όνομα	Εξαρτήσεις
1	Σύνδεση με το server	-
2	Αποστολή μηνύματος	1
3	Δημιουργία chat room	1
4	Είσοδος σε chat room	1
5	Έξοδος από chat room	1, 4
6	Έναρξη ιδιωτικής συνομιλίας	1
7	Αποστολή emoticon	1
8	Αποστολή έγχρωμου κειμένου	1
9	Προβολή διαθέσιμων chat room	1
10	Ορισμός θέματος chat room	1, 3

Οι εξαρτήσεις σημειώνουν την προϋπόθεση εκτέλεσης μιας περίπτωσης χρήσης πριν από μια άλλη.

1. Σύνδεση με το server

a) Περιγραφή σε φυσική γλώσσα

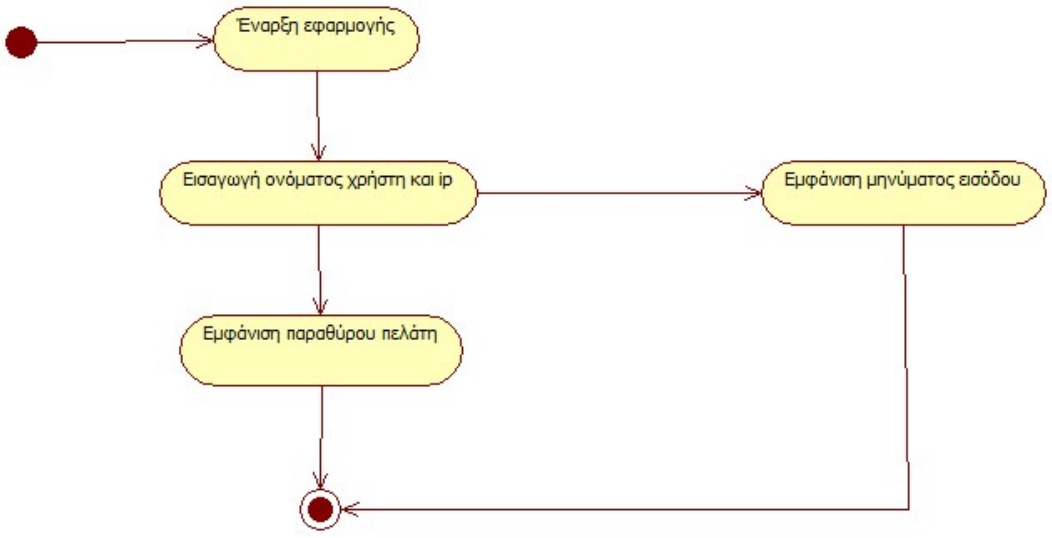
Ο πελάτης θέλει να ανταλλάξει μηνύματα με άλλους χρήστες της εφαρμογής. Ξεκινά την εφαρμογή του πελάτη και εισάγει το όνομα χρήστη και την ip διεύθυνση του. Τότε ανοίγει το παράθυρο επικοινωνίας του πελάτη και ο server εμφανίζει το μήνυμα εισόδου του χρήστη.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Έναρξη εφαρμογής**
2. **Εισαγωγή ονόματος χρήστη και ip**
3. **Εμφάνιση παραθύρου πελάτη**
4. **Εμφάνιση μηνύματος εισόδου**



2. *Αποστολή μηνύματος*

a) Περιγραφή σε φυσική γλώσσα

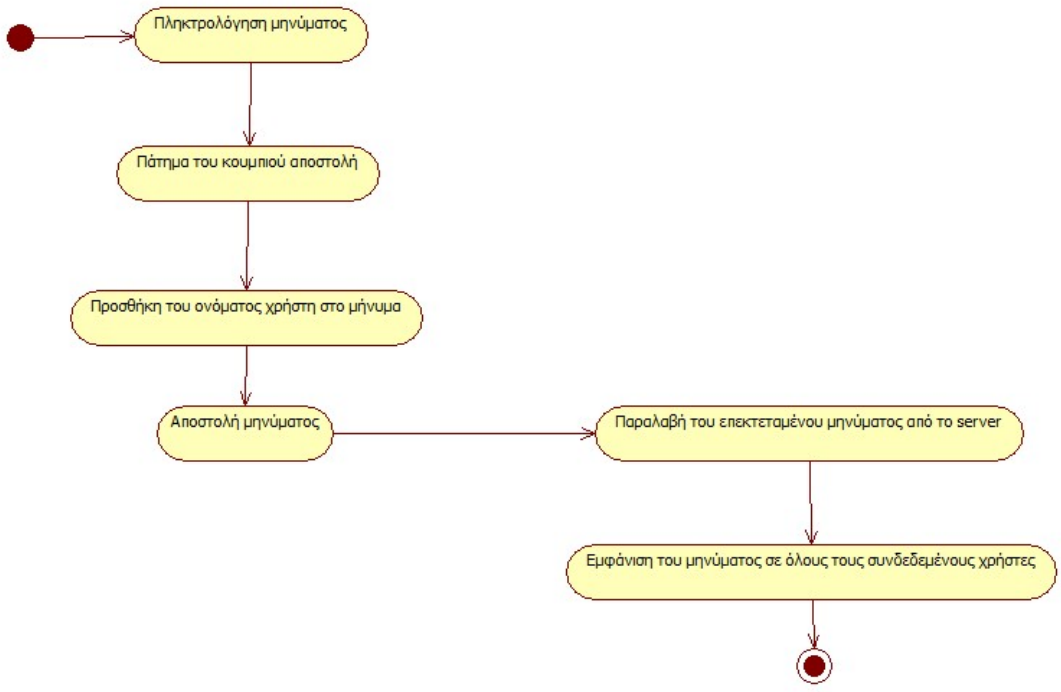
Ο πελάτης θέλει να στείλει ένα μήνυμα σε όλους τους χρήστες που είναι συνδεδεμένοι στην εφαρμογή. Αφού πληκτρολογήσει το μήνυμα, πατάει το κουμπί «αποστολή». Στο αρχικό μήνυμα προστίθεται το όνομα του χρήστη που το έστειλε. Ο server λαμβάνει το μήνυμα και το εμφανίζει στα παράθυρα όλων των χρηστών, μαζί με το όνομα του χρήστη που το έστειλε.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση μηνύματος**
2. **Πάτημα του κουμπιού αποστολή**
3. **Προσθήκη του ονόματος χρήστη στο μήνυμα**
4. **Παραλαβή του επεκτεταμένου μηνύματος από το server**
5. **Εμφάνιση του μηνύματος σε όλους τους συνδεδεμένους χρήστες.**



3. Δημιουργία chat room

a) Περιγραφή σε φυσική γλώσσα

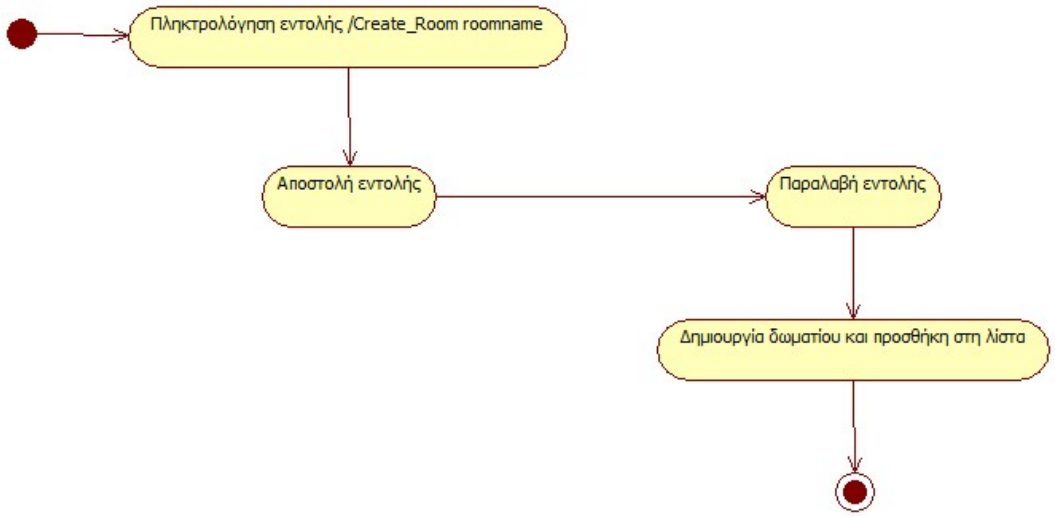
Ο πελάτης θέλει να δημιουργήσει ένα chat room. Στο χώρο εισαγωγής μηνυμάτων πληκτρολογεί την εντολή /Create_Room ακολουθούμενη από το όνομα του δωματίου. Το νέο δωμάτιο εμφανίζεται στη λίστα δωματίων του server.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση εντολής /Create_Room roomname**
2. **Δημιουργία δωματίου και προσθήκη στη λίστα του server.**



4. Είσοδος σε chat room

a) Περιγραφή σε φυσική γλώσσα

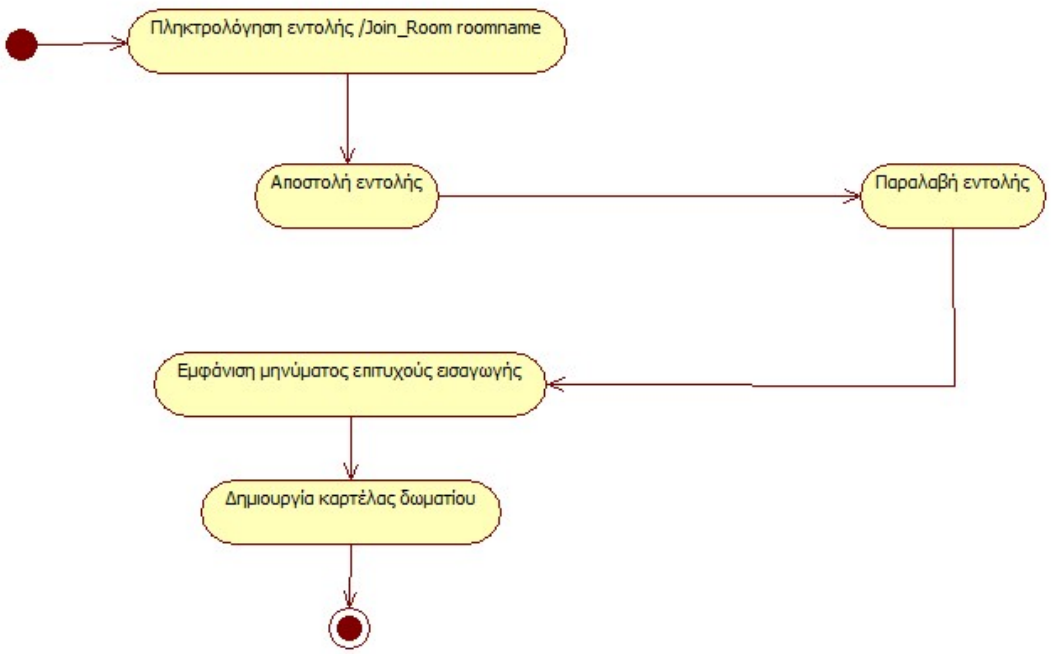
Ο πελάτης θέλει να μπει σε ένα chat room. Στο χώρο εισαγωγής μηνυμάτων πληκτρολογεί την εντολή /Join_Room ακολουθούμενη από το όνομα του δωματίου. Εμφανίζεται στο παράθυρο το μήνυμα επιτυχούς εισόδου στο chat room και δημιουργείτε μια νέα καρτέλα, όπου ο χρήστης θα επικοινωνεί με το συγκεκριμένο δωμάτιο χρηστών (και μόνο).

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση εντολής /Join_Room roomname**
2. **Εμφάνιση μηνύματος επιτυχούς εισαγωγής**
3. **Δημιουργία καρτέλας δωματίου**



5. Έξοδος από chat room

a) Περιγραφή σε φυσική γλώσσα

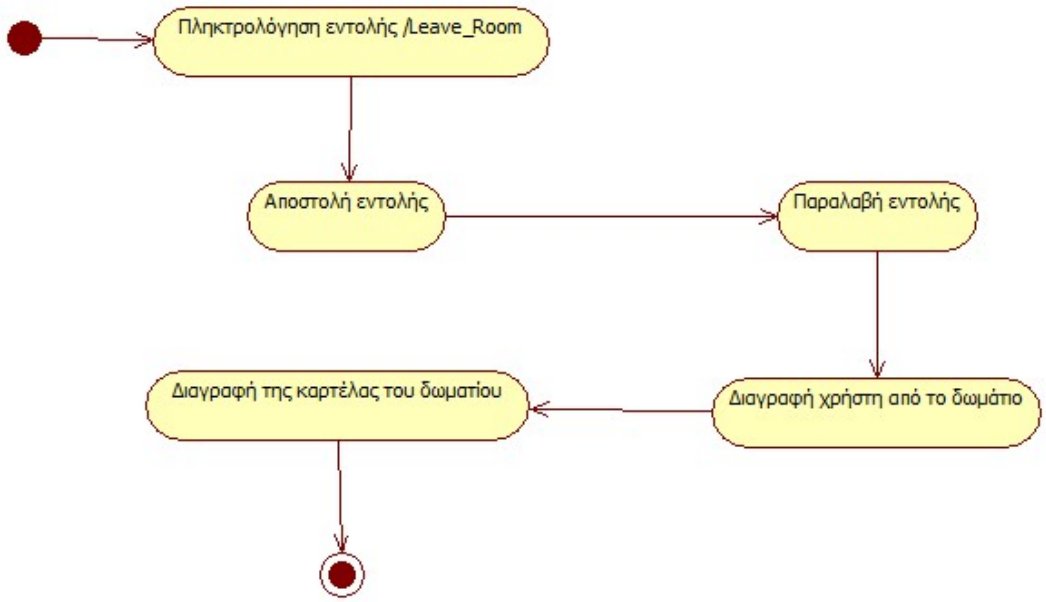
Ο πελάτης θέλει να βγει από ένα chat room. Στο χώρο εισαγωγής μηνυμάτων του chat room πληκτρολογεί την εντολή /Leave_Room. Τότε διαγράφεται η καρτέλα του chat room.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση εντολής /Leave_Room**
2. **Διαγραφή της καρτέλας του δωματίου**



6. Έναρξη ιδιωτικής συνομιλίας

a) Περιγραφή σε φυσική γλώσσα

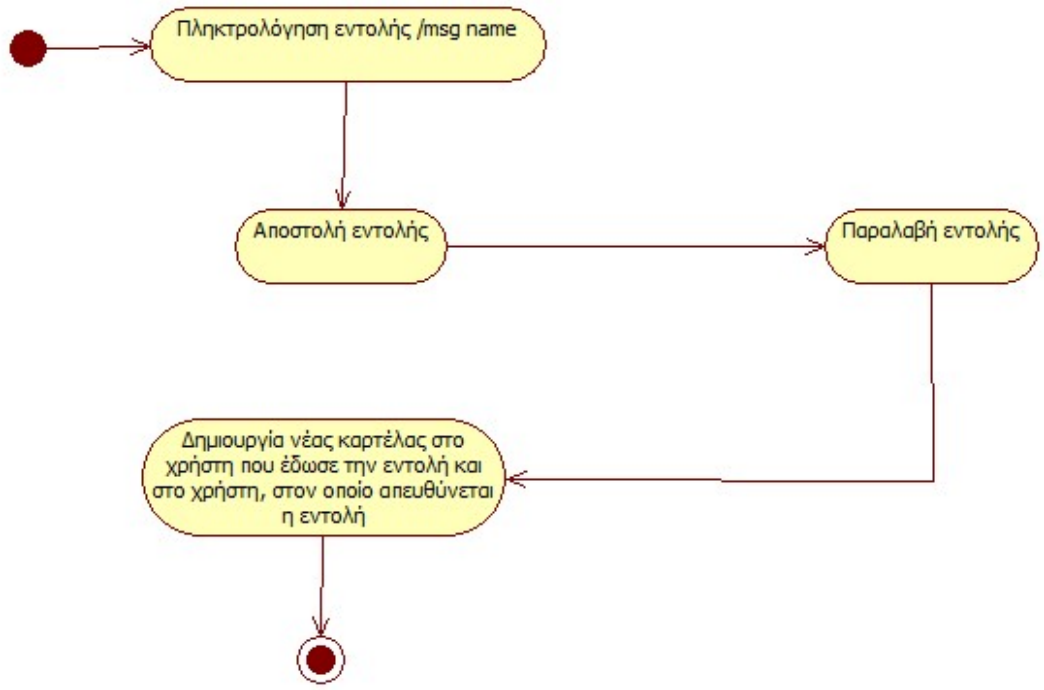
Ο πελάτης θέλει να ξεκινήσει μια ιδιωτική συνομιλία με κάποιο άλλο χρήστη. Στο χώρο εισαγωγής μηνυμάτων πληκτρολογεί την εντολή /msg ακολουθούμενη από το όνομα του άλλου χρήστη. Τότε και στους δύο χρήστες δημιουργείται μια νέα καρτέλα με το όνομα του έτερου χρήστη.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση εντολής /msg name**
2. **Δημιουργία νέας καρτέλας στο χρήστη που έδωσε την εντολή και στο χρήστη, στον οποίο απευθύνεται η εντολή.**



7. *Αποστολή emoticon*

a) Περιγραφή σε φυσική γλώσσα

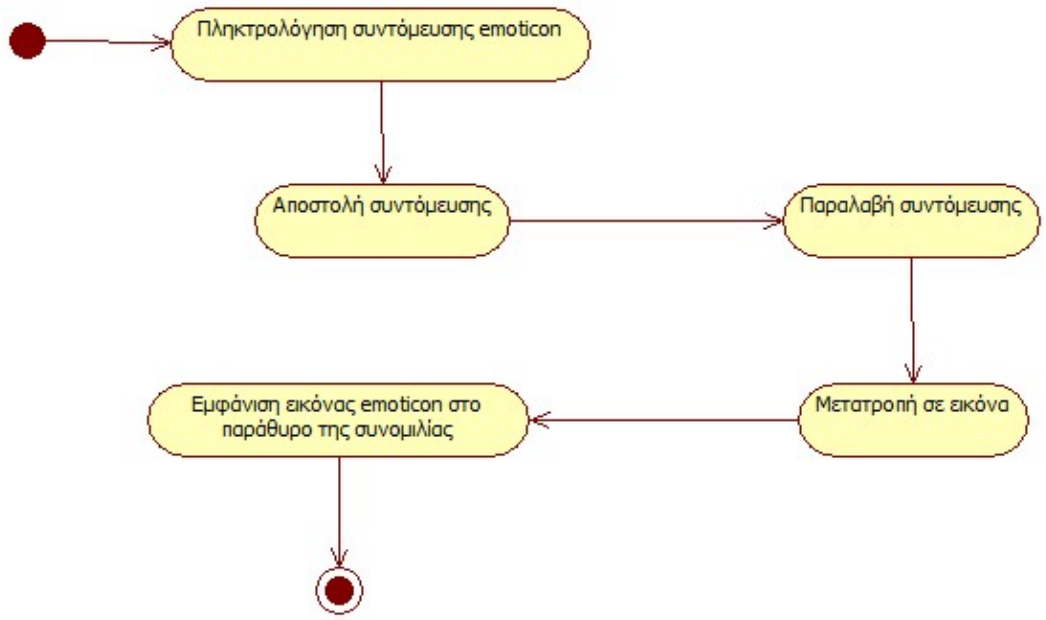
Ο πελάτης θέλει να προσθέσει ένα emoticon στο μήνυμά του. Στο χώρο εισαγωγής μηνυμάτων πληκτρολογεί μια από τις συντομεύσεως των emoticons [:) , ;) , p) κλπ..]. Όταν στείλει το μήνυμα, εμφανίζεται στην οθόνη το emoticon που αντιστοιχεί στην συντόμευση που πληκτρολογήθηκε.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση συντόμευσης emoticon**
2. **Εμφάνιση εικόνας emoticon στο παράθυρο της συνομιλίας**



8. *Αποστολή έγχρωμου κειμένου*

a) Περιγραφή σε φυσική γλώσσα

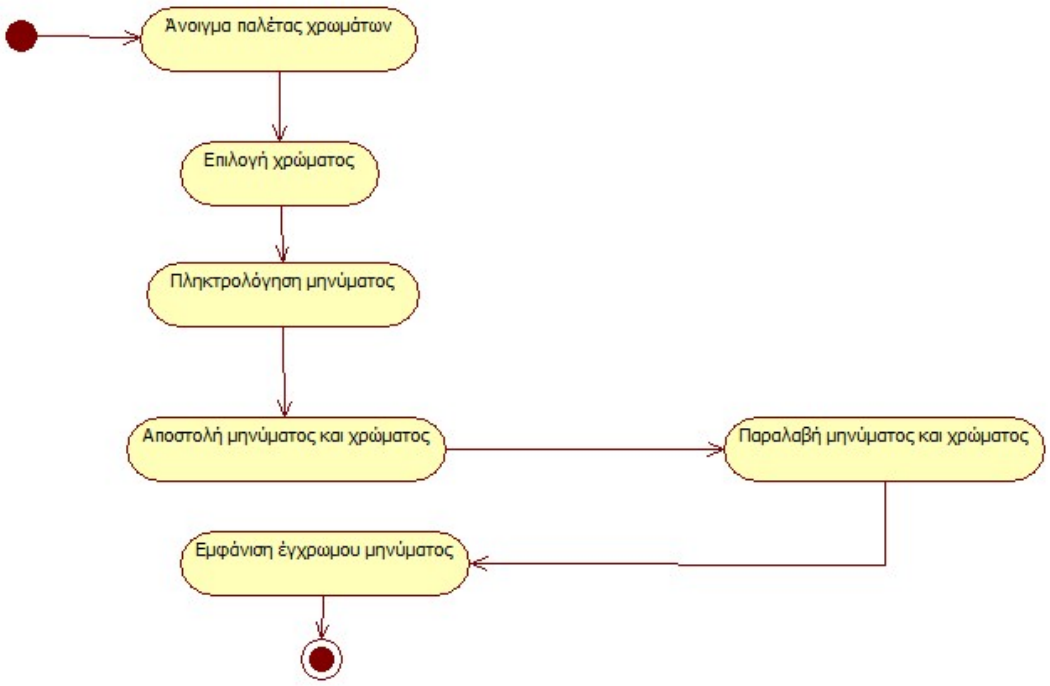
Ο πελάτης θέλει να στείλει έγχρωμο κείμενο. Πατώντας το κουμπί «color» δίπλα στην περιοχή εισαγωγής μηνυμάτων, εμφανίζεται μια παλέτα χρωμάτων. Ο πελάτης επιλέγει το χρώμα που επιθυμεί. Στη συνέχεια πληκτρολογεί ένα μήνυμα και το στέλνει. Το μήνυμα εμφανίζεται με το επιλεγμένο χρώμα.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Άνοιγμα παλέτας χρωμάτων**
2. **Επιλογή χρώματος**
3. **Πληκτρολόγηση μηνύματος**
4. **Αποστολή κειμένου και χρώματος**
5. **Εμφάνιση έγχρωμου κειμένου**



9. *Προβολή διαθέσιμων chat room*

a) Περιγραφή σε φυσική γλώσσα

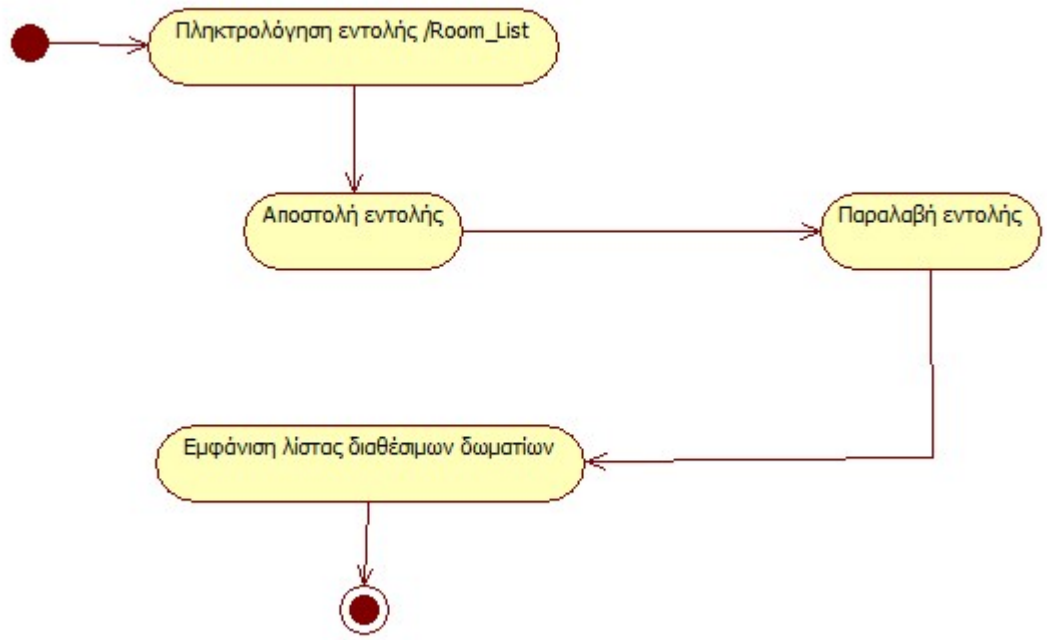
Ο πελάτης θέλει να δει όλα τα διαθέσιμα chat rooms. Στο χώρο εισαγωγής μηνυμάτων πληκτρολογεί την εντολή /Room_List. Τότε στο παράθυρο συνομιλιών εμφανίζεται η λίστα με τα διαθέσιμα chat rooms.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση εντολής /Room_List**
2. **Εμφάνιση λίστας διαθέσιμων δωματίων**



10. Ορισμός θέματος chat room

a) Περιγραφή σε φυσική γλώσσα

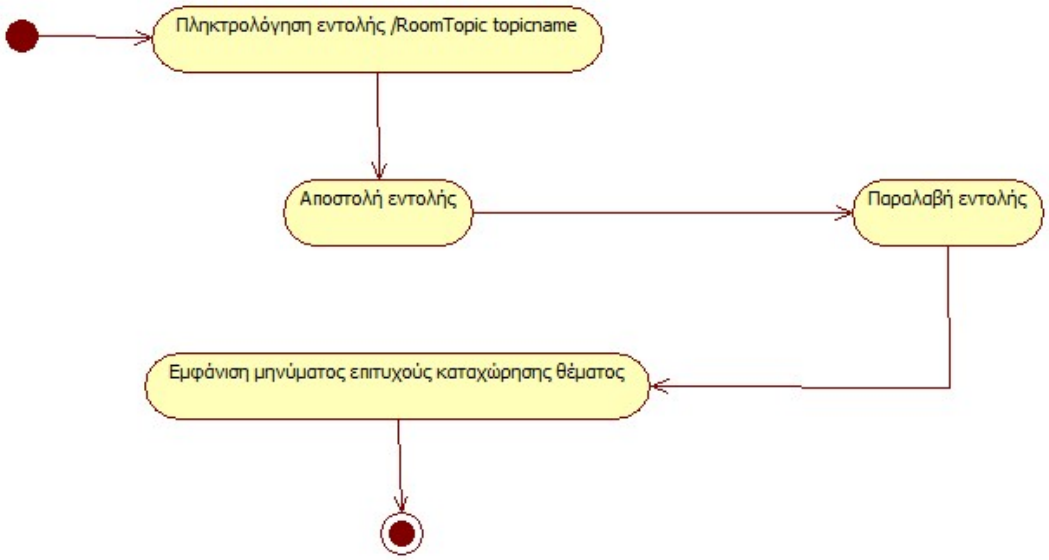
Ο πελάτης θέλει να ορίσει το θέμα του chat room που Δημιούργησε. Στο χώρο εισαγωγής μηνυμάτων πληκτρολογεί την εντολή /RoomTopic ακολουθούμενη από το όνομα του θέματος του δωματίου. Στο παράθυρο συνομιλιών εμφανίζεται ένα μήνυμα επιτυχούς καταχώρησης του θέματος.

b) Δράστης

Χρήστης - πελάτης

c) *Σενάριο*

1. **Πληκτρολόγηση εντολής /RoomTopic topicname**
2. **Εμφάνιση μηνύματος επιτυχούς καταχώρησης θέματος.**



4. Εργαλεία, πρότυπα και γλώσσα υλοποίησης

4.1 Η γλώσσα C++

Η **C++** (C Plus Plus) είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ. Θεωρείται μέσου επιπέδου γλώσσα, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου. Είναι μια δακτυλογραφούμενη, ελεύθερης μορφής, πολλαπλών παραδειγμάτων, μεταφράσιμη γλώσσα όπου η μετάφρασή της (compilation) δημιουργεί κώδικα μηχανής για ένα συγκεκριμένο τύπο υλικού. Υποστηρίζει δομημένο, αντικειμενοστραφή και γενικό προγραμματισμό.

Η γλώσσα αναπτύχθηκε από τον Bjarne Stroustrup το 1979 στα εργαστήρια Bell της AT&T, ως βελτίωση της ήδη υπάρχουσας γλώσσας προγραμματισμού C, και αρχικά ονομάστηκε "C with Classes", δηλαδή C με Κλάσεις. Μετονομάστηκε σε C++ το 1983. Οι βελτιώσεις ξεκίνησαν με την προσθήκη κλάσεων, και ακολούθησαν, μεταξύ άλλων, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα, πρότυπα κ.α.

Η γλώσσα ορίστηκε παγκοσμίως, το 1998, με το πρότυπο ISO/IEC 14882:1998. Η τρέχουσα έκδοση αυτού του προτύπου είναι αυτή του 2003, η ISO/IEC 14882:2003. Μια καινούρια έκδοση είναι υπό ανάπτυξη, γνωστή ανεπίσημα με την ονομασία C++0x.

Η C++ αποτέλεσε τη γλώσσα υλοποίησης των εφαρμογών.

4.2 Microsoft Visual Studio 2008

Το Microsoft Visual Studio είναι το κύριο ενσωματωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) που έχει δημιουργήσει η Microsoft. Μπορεί να χρησιμοποιηθεί για την ανάπτυξη

εφαρμογών κονσόλας ή γραφικής διεπαφής χρήστη, όπως επίσης Windows εφαρμογών, ιστοχώρων, εφαρμογών διαδικτύου και υπηρεσιών διαδικτύου.

Το Visual Studio περιλαμβάνει έναν συντάκτη κώδικα που υποστηρίζει το IntelliSense καθώς επίσης και αναπαραγωγή κώδικα. Ο ενσωματωμένος διορθωτής (debugger) λειτουργεί και ως διορθωτής επιπέδου πηγαίου κώδικα και ως διορθωτής επιπέδου μηχανής. Άλλα ενσωματωμένα εργαλεία που περιλαμβάνονται είναι ο σχεδιαστής φορμών για την δημιουργία GUI εφαρμογών, ο σχεδιαστής εφαρμογών Ιστού, ο σχεδιαστής κλάσεων και ο σχεδιαστής σχημάτων βάσεων δεδομένων. Επιτρέπει την προσθήκη plug-ins για την ενίσχυση της λειτουργίας σχεδόν σε κάθε επίπεδο, συμπεριλαμβανομένης της προσθήκης υποστήριξης για συστήματα ελέγχου πηγής (όπως το Supervision και το Team Foundation Server), την προσθήκη νέων συνόλων εργαλείων, όπως οι συντάκτες και οι οπτικοί σχεδιαστές για τις εξαρτώμενες από το πεδίο γλώσσες ή εργαλεία για άλλες πτυχές του κύκλου ζωής της ανάπτυξης λογισμικού (όπως ο Team Explorer).

Το Visual Studio υποστηρίζει διάφορες γλώσσες με τη βοήθεια των γλωσσικών υπηρεσιών, οι οποίες επιτρέπουν σε οποιαδήποτε γλώσσα προγραμματισμού να υποστηριχθεί (σε διαφορετικό βαθμό) από το συντάκτη και το διορθωτή κώδικα. Οι ενσωματωμένες γλώσσες περιλαμβάνουν τη C, τη C++, τη Visual Basic .NET και τη Visual C#. Η υποστήριξη για άλλες γλώσσες, όπως η Chrome, η F#, η Python, και η Ruby, μεταξύ άλλων, μπορεί να υπάρξει μέσω γλωσσικών υπηρεσιών, που πρέπει να εγκατασταθούν ξεχωριστά. Υποστηρίζει, επίσης, την XML/XSLT, την HTML/XHTML, τη JavaScript και τα CSS.

Το Microsoft Visual Studio στην έκδοση 2008 αποτέλεσε το εργαλείο ανάπτυξης όλων των εφαρμογών.

4.3 Microsoft Foundation Class Library (MFC)

Η Microsoft Foundation Class Library (MFC) είναι μια βιβλιοθήκη που τυλίγει μέρος του Windows API σε C++ κλάσεις, συμπεριλαμβανομένης της λειτουργίας που τους επιτρέπει να χρησιμοποιήσουν ένα προεπιλεγμένο πλαίσιο εφαρμογής. Οι κλάσεις καθορίζονται για πολλά από τα ρυθμισμένα αντικείμενα των Windows και επίσης για προκαθορισμένα παράθυρα και κοινά controls.

Η MFC εισήχθη το 1992 με το μεταγλωττιστή C/C++ 7.0 της Microsoft, για χρήση με τις δεκαεξάμπιτες εκδόσεις των Windows, ως ένα εξαιρετικά λεπτό αντικειμενοστρεφές C++ περιτύλιγμα για το Windows API. Η C++ είχε αρχίσει τότε να αντικαθιστά τη C στην ανάπτυξη των εμπορικών προγραμμάτων εφαρμογών. Έτσι, έγινε επίσης η πρώτη αντικατάσταση ενός παλαιότερου, αλφαριθμητικού IDE, που ονομαζόταν PWB.

Μια ενδιαφέρουσα ιδιορρυθμία της MFC είναι η χρήση του «Afx» ως πρόθεμα για πολλές λειτουργίες, μακροεντολές και το τυποποιημένο όνομα «stdafx.h» για τα header files. Κατά τη διάρκεια της αρχικής ανάπτυξης, αυτό που αργότερα ονομάστηκε MFC, αναφερόταν ως «Επεκτάσεις Πλαισίου Εφαρμογής (Application Framework Extensions)» που συντομογραφόταν ως «Afx». Το όνομα MFC υιοθετήθηκε πολύ αργά στον κύκλο απελευθέρωσης, για να αλλάξουν αυτές οι αναφορές.

Όταν η MFC εισήχθη, παρείχε C++ μακροεντολές για τη διαχείριση των μηνυμάτων των Windows (μέσω χαρτών μηνυμάτων), εξαιρέσεις (exeptions), τον προσδιορισμό τύπων χρόνου εκτέλεσης (RTTI), serialization και δυναμικά στιγμιότυπα κλάσεων. Οι μακροεντολές για τη διαχείριση των μηνυμάτων είχαν ως σκοπό να μειώσουν την κατανάλωση μνήμης, με την αποφυγή της υπέρμετρης χρήσης εικονικών πινάκων και, επιπλέον, να παρέχουν μια πιο συγκεκριμένη δομή για τα διάφορα Visual c++ εργαλεία, ώστε να μπορούν να χειριστούν τον κώδικα, χωρίς να επεξεργάζονται ολόκληρη τη γλώσσα. Οι μακροεντολές διαχείρισης των μηνυμάτων αντικατέστησαν τον εικονικό μηχανισμό λειτουργίας που παρείχε η C++.

Στις 7 Απριλίου 2008, η Microsoft απελευθέρωσε μια αναπροσαρμογή στις MFC κλάσεις ως εκτός ζώνης αναπροσαρμογή του Visual Studio 2008 και της MFC 9. Η αναπροσαρμογή χαρακτηρίζεται από την κατασκευή της νέας διεπαφής με τον χρήστη, συμπεριλαμβανομένης της διεπαφής Ribbon του Microsoft Office 2007 και σχετικά UI widgets, πλήρως εξατομικεύσιμες μπάρες εργαλείων και καρτέλες εγγράφων. Η νέα λειτουργικότητα παρέχεται σε νέες κλάσεις, έτσι ώστε οι παλαιές εφαρμογές να συνεχίζουν να τρέχουν.

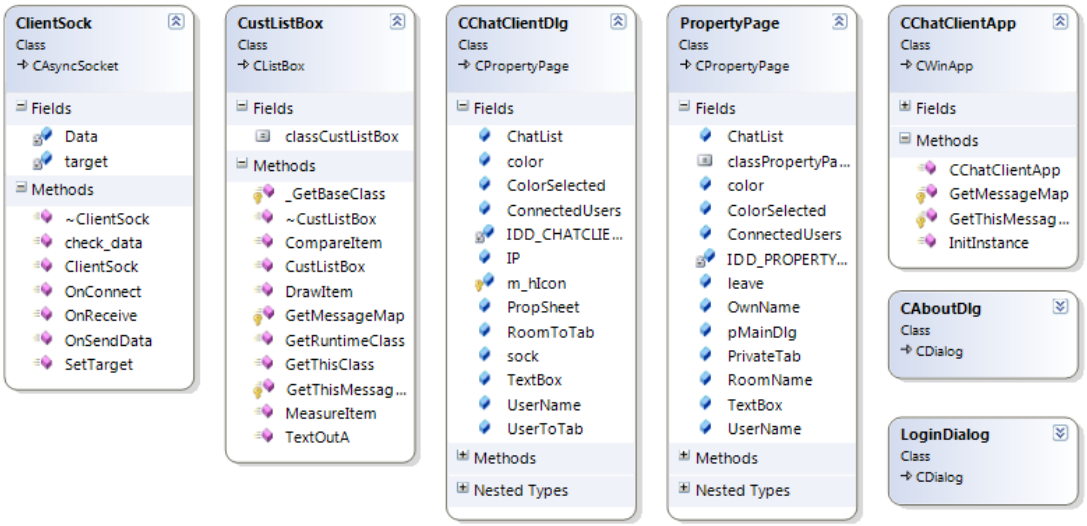
Η Microsoft Foundation Class Library (MFC) είναι η βιβλιοθήκη γραφικών που χρησιμοποιήθηκε στην εφαρμογή.

5. Σχεδίαση και Υλοποίηση

Στο κεφάλαιο αυτό θα παρουσιαστούν, αρχικά, τα αρχεία που υλοποιήθηκαν για τις εφαρμογές της πτυχιακής αυτής εργασίας. Μπορεί κανείς να παρατηρήσει ότι ορισμένα από τα αρχεία είναι κοινά και στις δύο εφαρμογές. Για παράδειγμα η δυνατότητα να εμφανίζεται χρωματισμένο κείμενο απαιτείται και στις δύο εφαρμογές και για αυτό υλοποιήθηκε ένα αρχείο και στη συνέχεια χρησιμοποιήθηκε και στις δύο εφαρμογές. Ο κώδικας υλοποίησης παρατίθεται στο Παράρτημα Α.

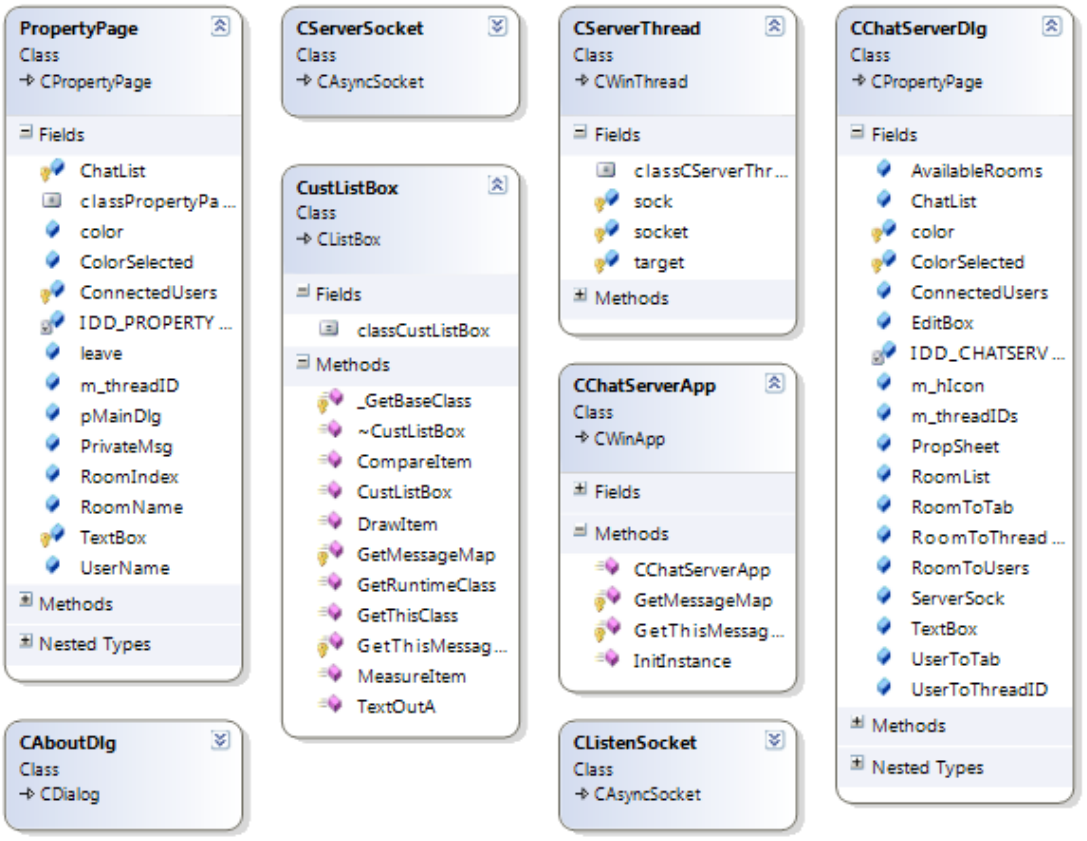
5.1 Class Diagrams

High Level Class Diagram για την εφαρμογή Client.



Αναλυτικότερο Class Diagram για την εφαρμογή Client.

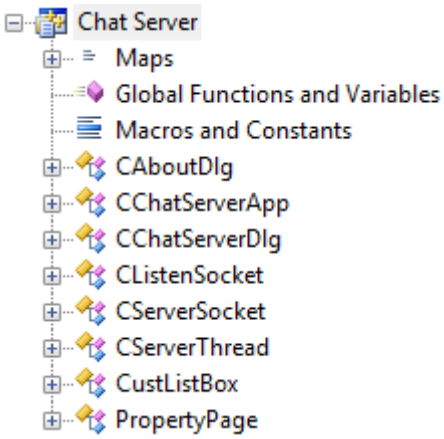
High Level Class Diagram για την εφαρμογή Server.



Αναλυτικότερο Class Diagram για την εφαρμογή Server.

5.2 Αρχεία του εξυπηρετητή

Η εφαρμογή του εξυπηρετητή αποτελείται από τα παρακάτω οκτώ (8) αρχεία:



Αναλυτικότερα, η χρήση του κάθε αρχείου παρουσιάζεται παρακάτω:

- **CChatServerApp.h/ CChatServerApp.cpp** :- Περιέχει το inheritance, το declaration και το definition της εφαρμογής που είναι μια C Windows Application.

- **CChatServerDlg.h/ CChatServerDlg.cpp** :- Περιέχει μια class που προέρχεται από την CPropertyPage και χρησιμοποιείται ως κύρια σελίδα property. Αυτή η σελίδα δεν κλείνει μέχρι να κάνει έξοδο ο χρήστης από την εφαρμογή.

- **CServerSocket.h/CServerSocket.cpp** :- Περιέχει την υλοποίηση της CServerSocket κλάσης, η οποία με τη σειρά της προέρχεται από την CAsyncSocket. Αντικείμενο της κλάσης αυτής περιέχεται στο CServerThread. Το αρχείο αυτό επίσης περιέχει την υλοποίηση της μεθόδου check_data(CString) που χρησιμοποιείται για να ελέγξει την κατάσταση (status) των δεδομένων που ελήφθησαν επιστρέφοντας μια από τις παρακάτω τιμές:

- ***DATA_IN_CONTINUATION:*** Εάν το δεδομένα που ελήφθησαν δεν είναι πλήρη και η συνέχειά τους αναμένεται να έρθει σε επόμενο TCP μήνυμα.
- ***COMMAND_DATA***: Εαν τα δεδομένα που ελήφθησαν αποτελούν εντολή για τον εξυπηρετητή.
- ***MESSAGE_DATA***: Εάν τα δεδομένα που ελήφθησαν είναι κανονικό μήνυμα επικοινωνίας μεταξύ των χρηστών της εφαρμογής

- **CListenSocket.h/CListenSocket.cpp** :- Περιέχει την υλοποίηση του CListenSocket που προέρχεται από το CAsyncSocket**.** Το αντικείμενο της κλάσης αυτής χρησιμοποιείται ως *Listener Socket* και «ακούει» για εισερχόμενες συνδέσεις TCP στην **θύρα (port) 4945**. Όταν ένας χρήστης συνδέεται στον εξυπηρετητή, τότε η μέθοδος OnAccept() αποδέχεται τη σύνδεση αυτή και ξεκινά ένα νέο νήμα (thread) που αναλαμβάνει από εκεί και πέρα την επικοινωνία του εξυπηρετητή με το συγκεκριμένο πελάτη.

- **CServerThread.cpp/ ServerThread.h :-** Περιέχει την υλοποίηση της κλάσης CServerThread, η οποία προέρχεται από την CWinThread. Είναι το νήμα που ξεκινά το αρχείο που παρουσιάστηκε παραπάνω (το CListenSocket) για κάθε εισερχόμενη σύνδεση. Αναλαμβάνει το νέο socket (που είναι τύπου CAsyncSocket) και την επικοινωνία μέσω αυτού με τον πελάτη..

- **CustListBox.h/ CustListBox.cpp** :- Περιέχει την υλοποίηση της κλάσης CustListBox, η οποία προέρχεται από την CListBox. Σκοπός του αρχείου αυτού είναι να παρέχει στο σύστημα τη δυνατότητα να εμφανίζει χρωματισμένο κείμενο και τα emoticons. Η κλάση CListBox που παρέχεται από τη γλώσσα προγραμματισμού δεν δίνει αυτές τις δυνατότητες. Ως αποτέλεσμα το αρχείο αυτό παρέχει τις παραπάνω αυτές επιθυμητές δυνατότητες στην εφαρμογή, αναγνωρίζοντας

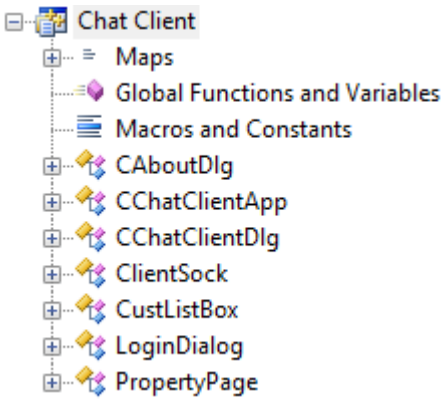
WM_DRAWITEM μηνύματα και κάνοντας customize τη συμπεριφορά και την εκτύπωση του κειμένου.

- **Messages.h**:- Το αρχείο αυτό, τόσο στον εξυπηρετητή όσο και στον πελάτη, ορίζει ορισμένα μηνύματα τα οποία μπορούν να μεταδοθούν στην Client/Server εφαρμογή της πτυχιακής αυτής εργασίας.

- **PropertyPage.h/ PropertyPage.cpp :-** Περιέχει την υλοποίηση της κλάσης PropertyPage που προέρχεται από την CPropertyPage που χρησιμοποιείται για τα ιδιωτικά και τα δημόσια δωμάτια επικοινωνίας.

5.3 Αρχεία του πελάτη

Η εφαρμογή του πελάτη αποτελείται από τα παρακάτω επτά (7) αρχεία:



- **CChatClientApp.h/ CChatClientApp.cpp** :- Περιέχει το inheritance, το declaration και το definition της εφαρμογής που είναι μια C Windows Application.

- **CChatClientDlg.h/ CChatClientDlg.cpp** :- Περιέχει μια class που προέρχεται από την CPropertyPage και χρησιμοποιείται ως κύρια σελίδα property. Αυτή η σελίδα δεν κλείνει μέχρι να κάνει έξοδο ο χρήστης από την εφαρμογή.

- **ClientSock.h/ ClientSock.cpp** :- Περιέχει την υλοποίηση της κλάσης ClientSocket που προέρχεται από την CAsyncSocket και χρησιμοποιείται για να συνδεθεί ο πελάτης στον εξυπηρετητή σε συγκεκριμένη IP διεύθυνση και θύρα (port) με συγκεκριμένο όνομα χρήστη (nick name). Το αρχείο αυτό περιέχει την υλοποίηση μεθόδου με την ονομασία check_data(CString) που ελέγχει την κατάσταση (status) των δεδομένων που λαμβάνονται επιστρέφοντας μια από τις παρακάτω τιμές:

  - ***DATA_IN_CONTINUATION*** : Εάν το δεδομένα που ελήφθησαν δεν είναι πλήρη και η συνέχειά τους αναμένεται να έρθει σε επόμενο TCP μήνυμα.
  - ***COMMAND_DATA*** : Εαν τα δεδομένα που ελήφθησαν αποτελούν εντολή για τον εξυπηρετητή.
  - ***MESSAGE_DATA*** : Εάν τα δεδομένα που ελήφθησαν είναι κανονικό μήνυμα επικοινωνίας μεταξύ των χρηστών της εφαρμογής.

- **CustListBox.h/ CustListBox.cpp** :- Περιέχει την υλοποίηση της κλάσης CustListBox, η οποία προέρχεται από την CListBox. Σκοπός του αρχείου αυτού είναι να παρέχει στο σύστημα τη δυνατότητα να εμφανίζει χρωματισμένο κείμενο και τα emoticons. Η κλάση CListBox που παρέχεται από τη γλώσσα προγραμματισμού δεν δίνει αυτές τις δυνατότητες. Ως αποτέλεσμα το αρχείο αυτό παρέχει τις παραπάνω αυτές επιθυμητές δυνατότητες στην εφαρμογή, αναγνωρίζοντας WM_DRAWITEM μηνύματα και κάνοντας customize τη συμπεριφορά και την εκτύπωση του κειμένου.

- **LoginDialog.h/ LoginDialog.cpp** :- Περιέχει την υλοποίηση του CDialogBox και δημιουργεί ένα dialog box για την είσοδο του χρήστη (login). Στο γραφικό περιβάλλον ο χρήστης μπορεί να τοποθετήσει το επιθυμητό όνομα χρήστη (nick name) και την IP διεύθυνση του εξυπηρετητή στον οποίο επιθυμεί να συνδεθεί.

- **PropertyPage.h/ PropertyPage.cpp :-** Περιέχει την υλοποίηση της κλάσης PropertyPage που προέρχεται από την CPropertyPage που χρησιμοποιείται για τα ιδιωτικά και τα δημόσια δωμάτια επικοινωνίας.

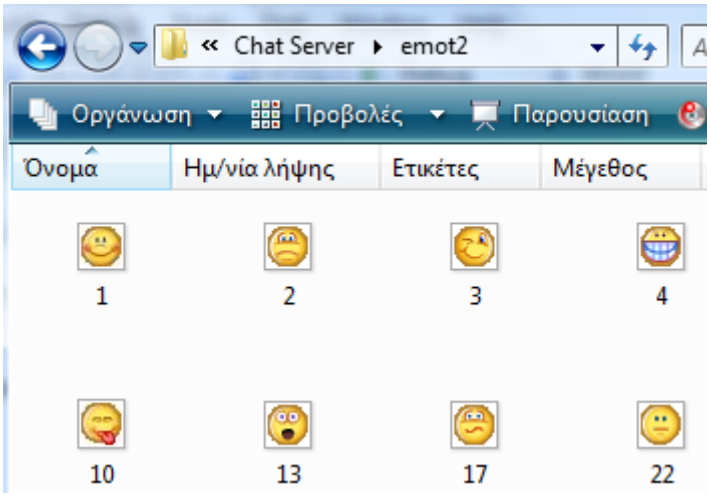
- **Stdafx.h/stdafx.cpp:-** Αυτό το αρχείο παρέχεται από το standard MFC framework και χρησιμοποιείται για να κάνει include διάφορα header files τόσο στην εφαρμογή του πελάτη όσο και του εξυπηρετητή.

5.4 Custom List Box

Τόσο στην εφαρμογή του πελάτη όσο και του εξυπηρετητή, έχει χρησιμοποιηθεί μια τροποποιημένη έκδοση του List Box component που προσθέτει στο αρχικό component δυνατότητες εκτύπωσης εικόνων (emoticons) και χρωματισμένου κειμένου. Τα αρχεία που υλοποιούν το παραπάνω είναι τα CustListBox.h και CustListBox.cpp.

Emoticons

Τα αρχεία με τις εικόνες των emoticons τοποθετούνται σε φάκελο /emot2



Στην υλοποίηση ο αλγόριθμος που εκτελείται είναι ο εξής. Για κάθε γραμμή κειμένου ξεκινά μια αναζήτηση διαδοχικά από τον πρώτο μέχρι και το

τελευταίο χαρακτήρα της γραμμής. Όταν αναγνωρίζονται διαδοχικά δύο συγκεκριμένοι χαρακτήρες όπως :P ή :D. τότε :

1. Εμφανίζουμε το κείμενο που προηγήθηκε του emoticon χρησιμοποιώντας την TextOut() μέθοδο στη θέση (indent.y). Η θέση αυτή είναι η αρχική θέση στην οποία πρέπει να εμφανιστεί ο πρώτος χαρακτήρας του κειμένου που πρέπει να εμφανιστεί.

2. Υπολογίζεται το πλάτος (width) του κειμένου που μόλις εμφανίστηκε. Αυτό γίνεται με τη μέθοδο GetTextExtent().cx, και προσθέτουμε σε αυτή την τιμή το indent (indent+=dc-> GetTextExtent(str,str.GetLength ()).cx;).

3. Επειδή η εικόνα για το emoticon (π.χ. για το :P) εμφανίζεται αμέσως μετά το κείμενο που μόλις εμφανίστηκε, η εικόνα εμφανίζεται στη θέση (indent.y). Για την εμφάνιση του emoticon η μέθοδος DrawState() καλείται. Το μέγεθος κάθε emoticon είναι 15x15 pixels. Για το λόγο αυτό προσθέτουμε 15 στη μεταβλητή indent. Αυτό συμβαίνει με το να φορτώνουμε στη μεταβλητή hBmp την εικόνα που αντιστοιχεί στο συγκεκριμένο emoticon (π.χ. IDB_TONGUE_SMILE).

Χρωματισμός κειμένου

Ένα μη χρωματισμένο κείμενο μεταφέρεται στη μορφή:

<DS>RoomName:UserName: Θέλω το μπλε χρώμα <DE>

Για να επιτευχθεί η μεταφορά της πληροφορίας του χρώματος το παραπάνω μήνυμα θα πρέπει να μεταφέρεται ως εξής.

<DS>RoomName:UserName: (c:0,0,255)Θέλω το μπλε χρώμα <DE>

Η κωδικοποίηση του χρωματισμού λοιπόν είναι η

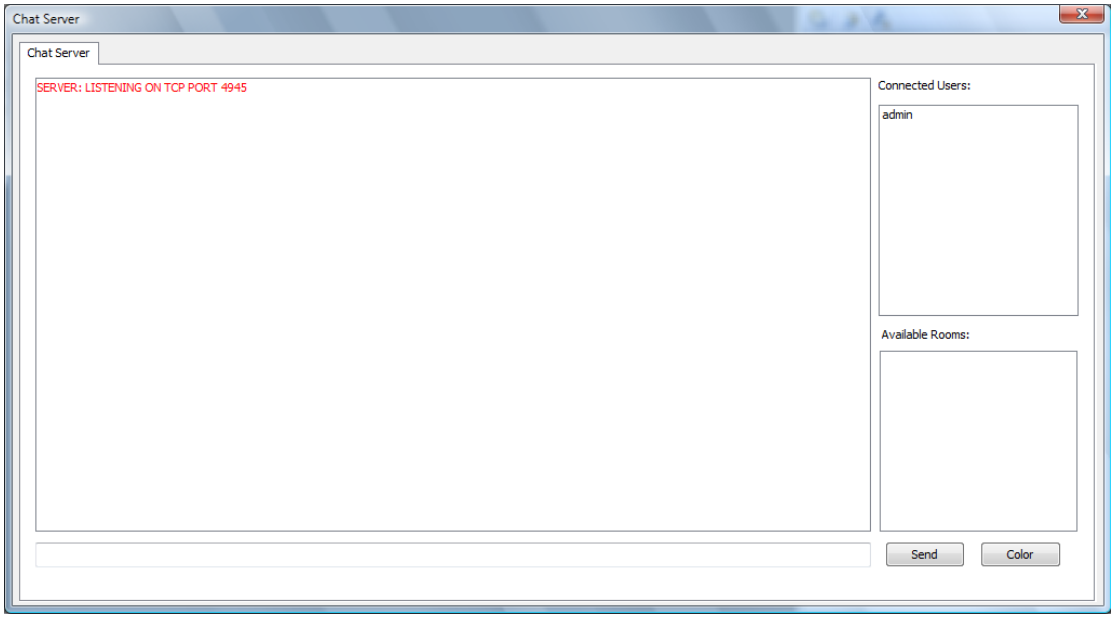
(c:R,G,B)

Όπου R,G,B είναι οι τιμές των βασικών χρωμάτων Red, Green, Blue εκφρασμένες από το 0 μέχρι το 255.

6. Παρουσίαση Εφαρμογής

6.1 Αρχική οθόνη εφαρμογής εξυπηρετητή

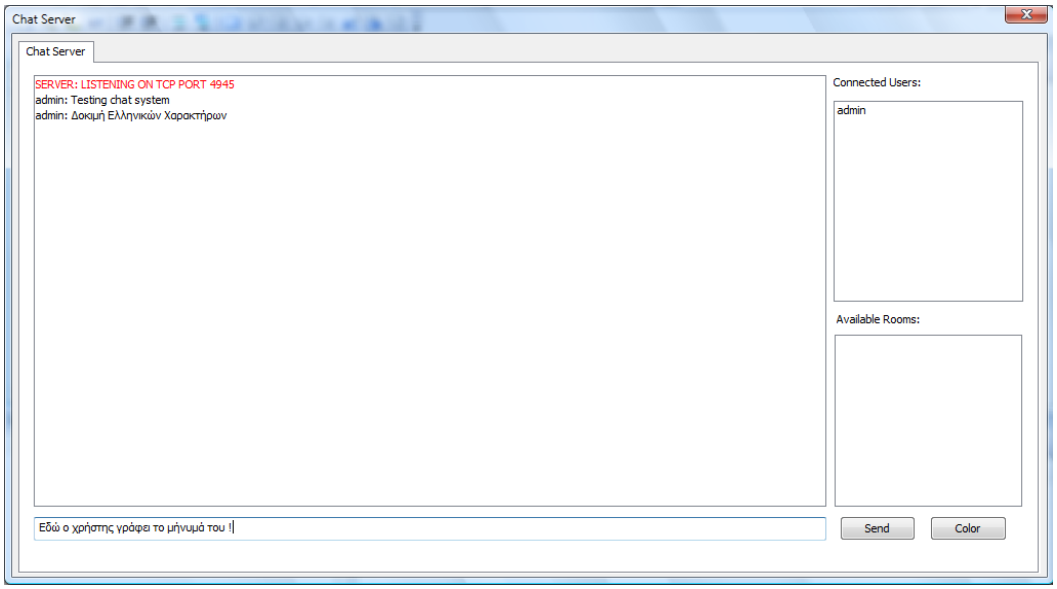
Αρχικά ξεκινά η εφαρμογή του εξυπηρετητή. Στο κεντρικό παράθυρο υπάρχει μια καρτέλα ανοικτή με την ονομασία Chat Server. Το μήνυμα που αρχικά εμφανίζεται, ενημερώνει ότι ο εξυπηρετητής λειτουργεί κανονικά ακούγοντας στην θύρα 4945. Στο πάνω δεξιά μέρος της εφαρμογής εμφανίζονται οι χρήστες που είναι συνδεδεμένοι. Προς το παρόν, μόνο ο διαχειριστής του εξυπηρετητή, με το όνομα χρήστη **admin** είναι συνδεδεμένος. Κάτω δεξιά εμφανίζονται τα δημόσια δωμάτια επικοινωνίας που έχουν δημιουργηθεί από τους χρήστες της πλατφόρμας. Στο κάτω μέρος της εφαρμογής υπάρχει ένα κουμπί **Send** για αποστολή μηνυμάτων και ένα κουμπί **Color** με το οποίο ο χρήστης μπορεί να χρωματίσει το μήνυμά του.



Εικόνα 1: Αρχική εικόνα - γραφικό περιβάλλον εξυπηρετητή. Μόνο ο διαχειριστής βρίσκεται στο σύστημα.

Τα μηνύματα που πληκτρολογούνται στο *Text Box* και αποστέλλονται, εμφανίζονται στην καρτέλα η οποία είναι επιλεγμένη. Ανοίγοντας η εφαρμογή,

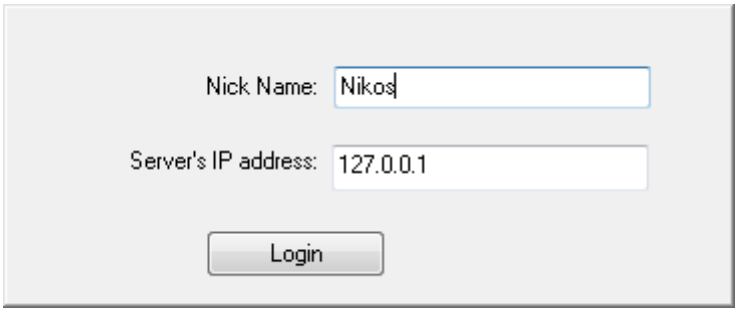
η μοναδική καρτέλα που είναι ανοικτή είναι αυτή του κεντρικού δημοσίου δωματίου επικοινωνίας, που όλοι οι χρήστες μπαίνουν κατά την εισαγωγή τους.



Εικόνα 2: Δοκιμή αποστολής Αγγλικών και Ελληνικών χαρακτήρων. Κανένας χρήστης δεν είναι συνδεδεμένος στο σύστημα.

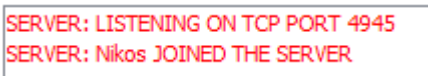
6.2 Σύνδεση πελάτη με το server

Για το σκοπό της παρουσίασης, ένας πελάτης - χρήστης θα συνδεθεί στο σύστημα και θα συνομιλήσει με το διαχειριστή. Η αρχική εικόνα που εμφανίζεται στην εφαρμογή *Chat Client* είναι το παρακάτω *Dialog* όπου ο χρήστης εισάγει το *Nick Name* του (το επιθυμητό όνομα χρήστη) και την IP διεύθυνση του εξυπηρετητή. Η δοκιμή γίνεται στον ίδιο υπολογιστή, οπότε το IP του εξυπηρετητή είναι το 127.0.0.1

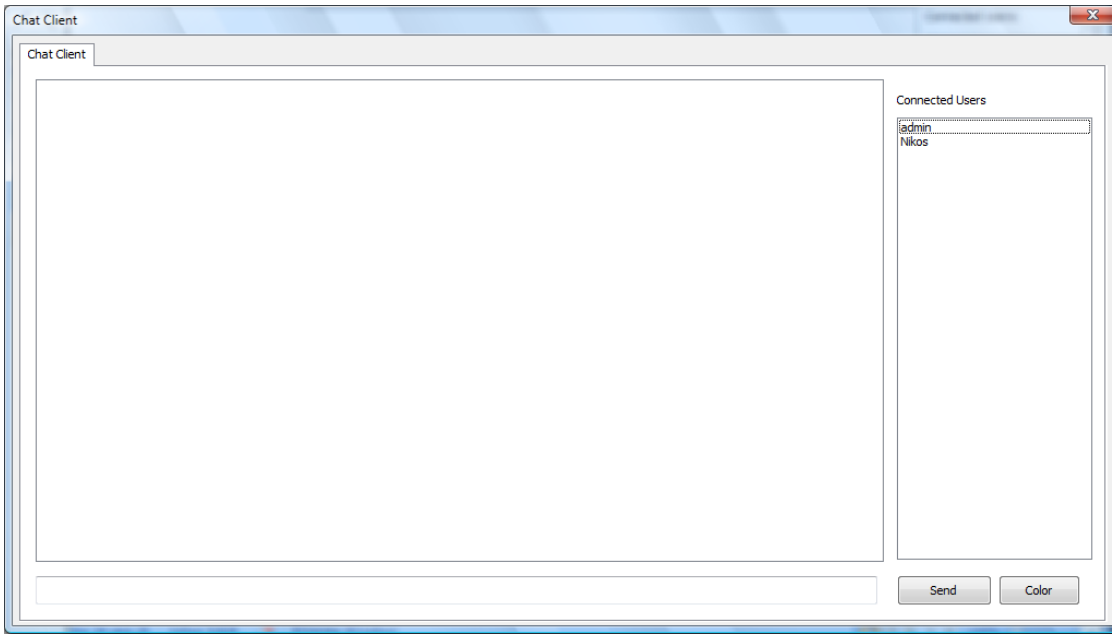


Εικόνα 3: Σύνδεση εφαρμογής πελάτη με όνομα χρήστη «Nikos»

Πατώντας **Login** ο χρήστης *Nikos* συνδέεται με επιτυχία στον εξυπηρετητή, στον οποίο εμφανίζεται μήνυμα ότι ο συγκεκριμένος χρήστης συνδέθηκε.



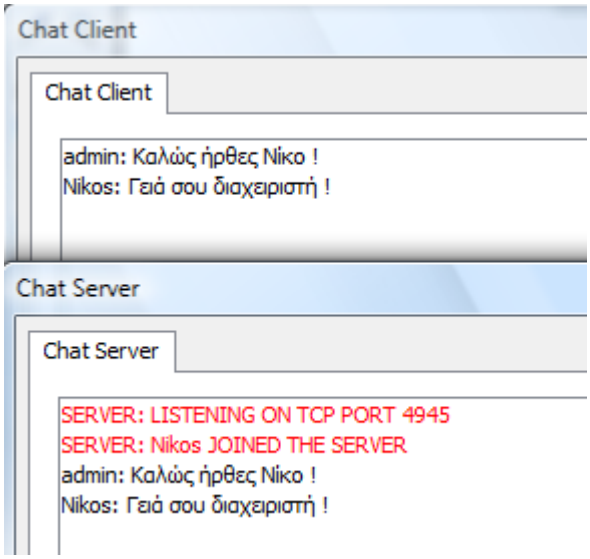
Εικόνα 4: Μήνυμα εξυπηρετητή αποδοχής σύνδεσης



Εικόνα 5: Το γραφικό περιβάλλον του πελάτη. Ο χρήστης Nikos βρίσκεται στο κεντρικό δωμάτιο επικοινωνίας μαζί με το διαχειριστή του συστήματος.

6.3 Αποστολή μηνύματος

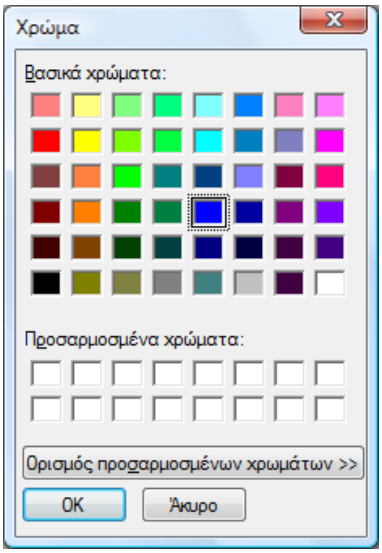
Ο διαχειριστής θέλει να καλωσορίσει το νέο χρήστη και του αποστέλλει το μήνυμα «Καλώς ήρθες Νίκο !». Ο χρήστης με τη σειρά του μπορεί να απαντήσει. Το κείμενο που στέλνουν έχει το προεπιλεγμένο μαύρο χρώμα.



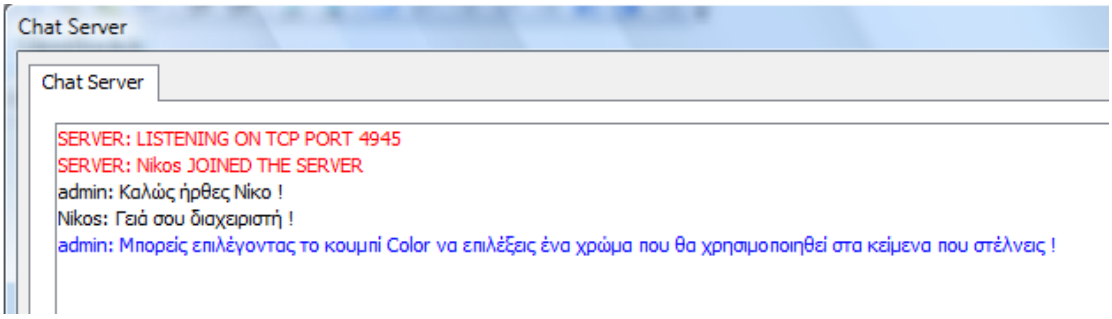
Εικόνα 6: Ο χρήστης και ο διαχειριστής συνομιλούν. Στο πάνω παράθυρο εικόνα από το *Chat Client*, στο κάτω από το *Chat Server*.

6.4 Αποστολή έγχρωμου κειμένου

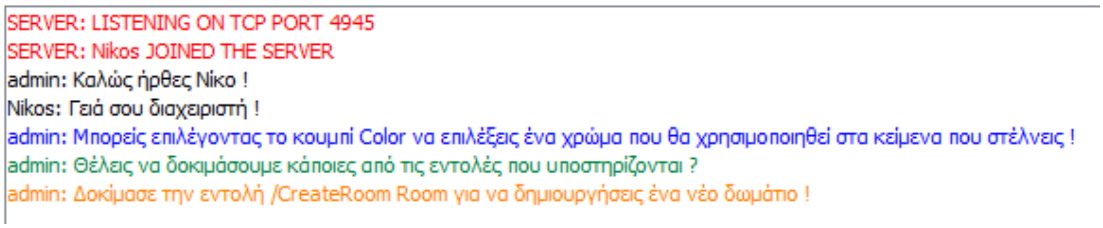
Κάθε χρήστης μπορεί να χρωματίσει το κείμενο που αποστέλλει χρησιμοποιώντας το κουμπί **Color**. Με το πάτημα του κουμπιού εμφανίζεται η παρακάτω παλέτα χρωμάτων:



Εικόνα 7: Ο διαχειριστής επιλέγει το κουμπί Color και το παραπάνω παράθυρο εμφανίζεται. Από τα βασικά χρώματα επιλέγει το μπλε.



Εικόνα 8: Το κείμενο που στέλνεται εμφανίζεται σε μπλε χρώμα τόσο στον πελάτη όσο και στο γραφικό περιβάλλον του διαχειριστή.



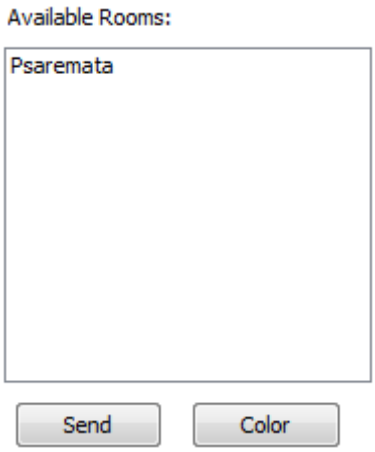
Εικόνα 9: Αποστολή μηνυμάτων σε διάφορα χρώματα

6.5 Δημιουργία chat room

Ο χρήστης Νίκος θέλει να δημιουργήσει ένα νέο δωμάτιο επικοινωνίας. Εκτελεί την εντολή /Create_Room και δημιουργεί ένα δωμάτιο επικοινωνίας με όνομα *Psaremata*.

Εικόνα 10: Ο χρήστης δημιουργεί ένα νέο δωμάτιο επικοινωνίας

Στο γραφικό περιβάλλον του διαχειριστή εμφανίζεται στη λίστα των δωματίων το νέο αυτό δωμάτιο επικοινωνίας:



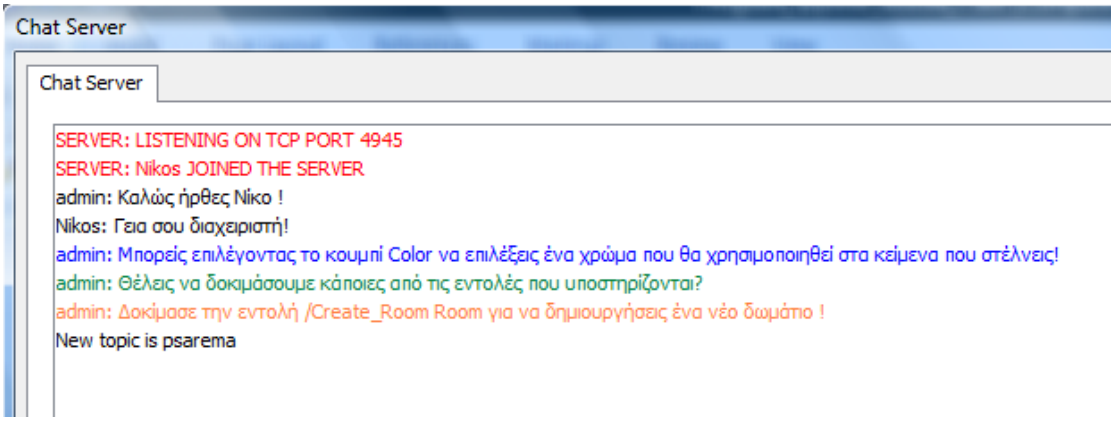
Εικόνα 11: Εμφάνιση νέου δωματίου στην εφαρμογή του εξυπηρετητή

6.6 Ορισμός θέματος chat room

Για να ορίσει το θέμα του δωματίου, ο χρήστης Νίκος πληκτρολογεί την εντολή /RoomTopic psarema.

Εικόνα 12: Ο χρήστης ορίζει το θέμα του δωματίου

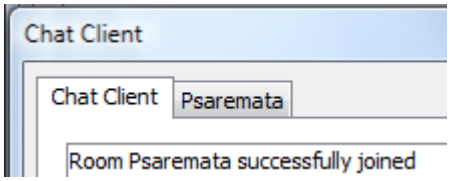
Στα παράθυρα των χρηστών εμφανίζεται το μήνυμα για το νέο θέμα.



Εικόνα 13: Μήνυμα για το θέμα του δωματίου συνομιλιών

6.7 Είσοδος σε chat room

Τόσο ο χρήστης όσο και ο διαχειριστής μπορούν να μπουν στο νέο αυτό δωμάτιο επικοινωνίας με την εντολή /Join_Room. Σε περίπτωση που ένας χρήστης προσπαθήσει να μπει σε ένα δωμάτιο το οποίο δεν έχει πρώτα δημιουργηθεί τότε του εμφανίζεται το μήνυμα: Room Name not found . Διαφορετικά, εμφανίζεται μήνυμα ότι με έγινε εισαγωγή με επιτυχία στο δωμάτιο αυτό. Μάλιστα αυτομάτως δημιουργείται μια νέα καρτέλα για να περιέχει τα μηνύματα που αφορούν το συγκεκριμένο δωμάτιο.

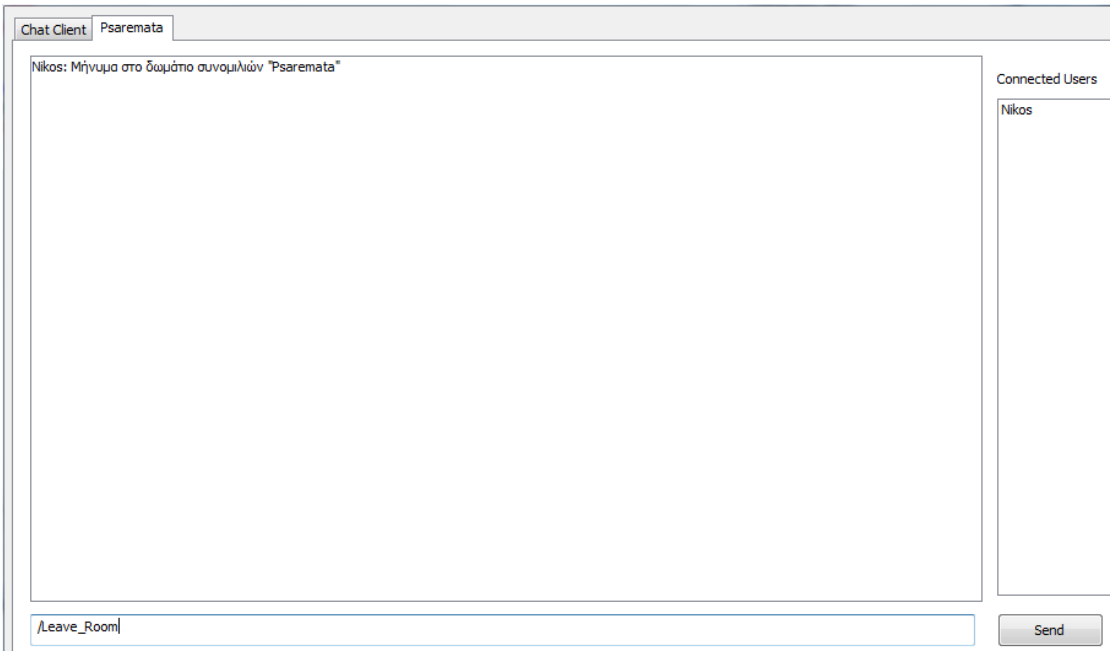


Εικόνα 14: Εισαγωγή στο νέο δωμάτιο με θεματολογία ψαρέματα.

Από εδώ και πέρα στέλνοντας μηνύματα, τα μηνύματα αυτά μεταφέρονται στο αντίστοιχο δωμάτιο με την καρτέλα που είναι επιλεγμένη. Μπορούν δηλαδή να εξελίσσονται παράλληλα πολλές διαφορετικές συζητήσεις.

6.8 Έξοδος από chat room

Οι χρήστες ενός chat room μπορούν να επιλέξουν να εξέλθουν από αυτό, χρησιμοποιώντας την εντολή /Leave_Room. Η εντολή αυτή πρέπει να δοθεί στην καρτέλα συνομιλιών του δωματίου.



Εικόνα 15: Ο χρήστης βγαίνει από το δωμάτιο συνομιλιών με τίτλο «Psaremata»

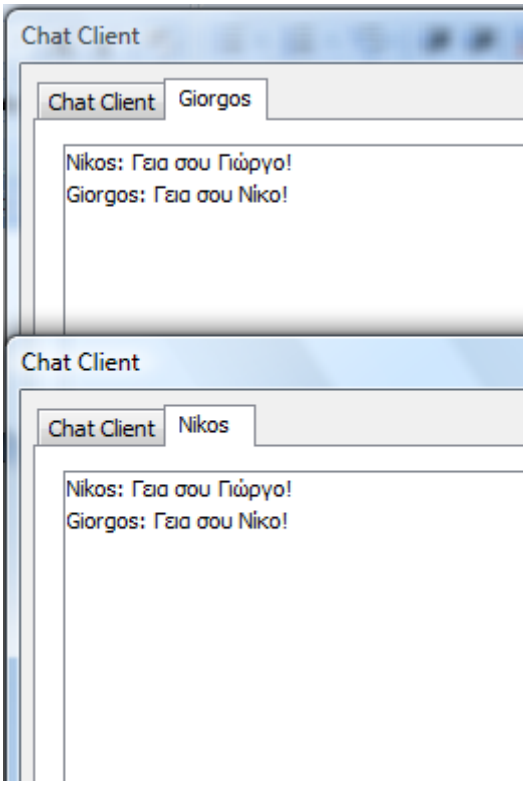
Μετά την εκτέλεση αυτής της εντολής, η καρτέλα του δωματίου κλείνει.

6.9 Έναρξη ιδιωτικής συνομιλίας με άλλο χρήστη

Ο χρήστης Nikos επιθυμεί να ξεκινήσει μια ιδιωτική συνομιλία με ένα άλλο χρήστη, με όνομα «Giorgos». Για να το κάνει αυτό πληκτρολογεί την εντολή "/msg Giorgos".

Εικόνα 16: Έναρξη ιδιωτικής συνομιλίας

Στο παράθυρο και των δύο χρηστών εμφανίζεται μια νέα καρτέλα με το όνομα του έτερου συνομιλητή.



Εικόνα 17: Δημιουργία νέας καρτέλας για την ιδιωτική συνομιλία.

Τα μηνύματα που στέλνονται σε αυτή την καρτέλα τα βλέπουν μόνο οι δύο συνομιλητές.

6.10 Αποστολή emoticon

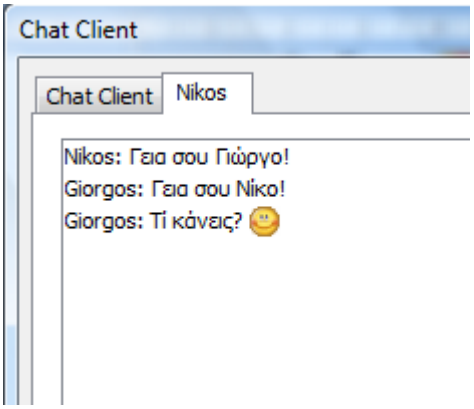
Ένας χρήστης μπορεί να θέλει να συμπεριλάβει στο μήνυμά του κάποιο από τα εικονίδια emoticon. Για να το κάνει αυτό, εισάγει στο μήνυμά του κάποια από τις συντομεύσεις των εικονιδίων.

- :)
- :d ή :D
- :P

- :(
- :s
- ;)
- :o
- :|

Εικόνα 18: Αποστολή μηνύματος με emoticon

Στο παράθυρο συνομιλιών, αντί για την συντόμευση, εμφανίζεται το αντίστοιχο εικονίδιο.



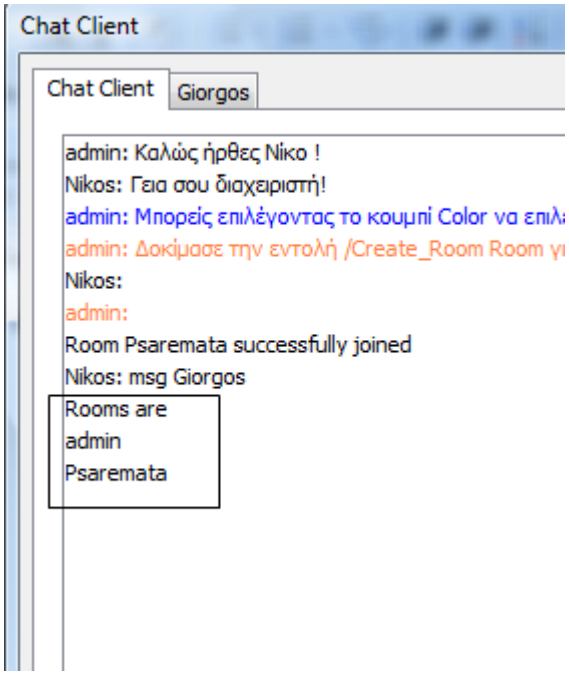
Εικόνα 19: Εμφάνιση emoticon

6.11 Προβολή διαθέσιμων chat room

Εάν ο χρήστης επιθυμεί να δει όλα τα διαθέσιμα chat room, πληκτρολογεί την εντολή /Room_List.

Εικόνα 20: Ο χρήστης επιθυμεί να δει τα διαθέσιμα chat rooms.

Στο παράθυρο συνομιλιών εμφανίζεται η λίστα με τα διαθέσιμα δωμάτια.



Εικόνα 21: Εμφάνιση λίστας διαθέσιμων δωματίων

7. Συμπεράσματα

Στα πλαίσια της πτυχιακής αυτής εργασίας έγινε συστηματική χρήση Socket programming και νημάτων (threads). Το πρωτόκολλο επικοινωνίας του IRC (Internet Relay Chat) μελετήθηκε και δημιουργήθηκε ένα υποσύνολό του το οποίο και χρησιμοποιήθηκε. Επίσης επειδή, ειδικά στις εφαρμογές Instant Messaging, συχνά γίνεται χρήση των emoticons, αποφασίστηκε να υποστηριχτούν οι εικόνες με τα χαμογελαστά προσωπάκια κ.α.

Σε γενικές γραμμές η πτυχιακή αυτή εργασία ήταν απαιτητική και ενδιαφέρουσα. Έγινε μια σχετική μελέτη σχετικά με την αρχιτεκτονική που θα ακολουθηθεί και πάρθηκαν ορισμένες αποφάσεις. Κώδικας socket και thread μελετήθηκε και αναπτύχθηκε. Απαίτησε μάλιστα και αρκετό debugging. Το γραφικό περιβάλλον που δημιουργήθηκε είναι ευχάριστο για τους χρήστες (υποστηρίζει χρώματα, emoticons κτλ) και γενικότερα η πτυχιακή εργασία ολοκληρώθηκε με επιτυχία.

8. Βιβλιογραφία

[1] Internet Relay Chat Request for Comments 1459

http://www.irchelp.org/irchelp/text/rfc1459.txt

[2] Internet Relay Chat Client - MIRC

http://www.mirc.com/

[3] MSN Messenger

http://download.live.com/messenger

[4] ICQ Client

http://www.icq.com

[5] Νήματα - συγχρονισμός

http://el.wikipedia.org/wiki/%CE%9D%CE%AE%CE%BC%CE%B1_(%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CE%AD%CF%82)

http://eclass.di.uoa.gr/D244/

https://faculty.teilam.gr/send.php?file_id=30

[6] TCP επικοινωνία - sockets

http://en.wikipedia.org/wiki/Transmission_Control_Protocol

http://dtps.unipi.gr/files/notes/2004-2005/eksamino_6/texnologies_diadiktyoy/tcp_programming.pdf

http://web.teipir.gr/new/ecs/pelab_1/tcp/inter3.htm

http://www.medialab.ntua.gr/education/MultimediaTechnology/JavaNotes/files/11.2%20Sockets.pdf

http://netlab.teiath.gr/JSPWiki/attach/NetLabEx/Exercise4-Theory.pdf

[7] Visual studio

http://en.wikipedia.org/wiki/Visual_studio

[8] C++

http://en.wikipedia.org/wiki/C%2B%2B

http://msdn.microsoft.com/en-us/visualc/default.aspx

http://www.cppreference.com/wiki/

[9] Microsoft Foundation Class Library - MFC

http://en.wikipedia.org/wiki/Microsoft_Foundation_Class_Library

8. Παράρτημα Α – Κώδικας υλοποίησης

Παρακάτω παρουσιάζεται ο κώδικας υλοποίησης της εφαρμογής.

8.1 Κώδικας υλοποίησης Chat Client

Chat Client.h

```
// Chat Client.h : main header file for the PROJECT_NAME application
//

#pragma once

#ifndef __AFXWIN_H__
	#error "include 'stdafx.h' before including this file for PCH"
#endif

#include "resource.h"		// main symbols


// CChatClientApp:
// See Chat Client.cpp for the implementation of this class
//

class CChatClientApp : public CWinApp
{
public:
	CChatClientApp();

// Overrides
	public:
	CPropertySheet PropSheet;
	virtual BOOL InitInstance();

// Implementation

	DECLARE_MESSAGE_MAP()
};

extern CChatClientApp theApp;
```

Chat Client.cpp

```
// Chat Client.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Chat Client.h"
#include "Chat ClientDlg.h"
#include "LoginDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif
```

```
// CChatClientApp

BEGIN_MESSAGE_MAP(CChatClientApp, CWinApp)
	ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
END_MESSAGE_MAP()


// CChatClientApp construction

CChatClientApp::CChatClientApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}


// The one and only CChatClientApp object

CChatClientApp theApp;


// CChatClientApp initialization

BOOL CChatClientApp::InitInstance()
{
// InitCommonControlsEx() is required on Windows XP if an application
// manifest specifies use of ComCtl32.dll version 6 or later to enable
// visual styles.  Otherwise, any window creation will fail.
	INITCOMMONCONTROLSEX InitCtrls;
	InitCtrls.dwSize = sizeof(InitCtrls);
// Set this to include all the common control classes you want to use
// in your application.
	InitCtrls.dwICC = ICC_WIN95_CLASSES;
	InitCommonControlsEx(&InitCtrls);

	CWinApp::InitInstance();

	if (!AfxSocketInit())
	{
		AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
		return FALSE;
	}

	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	// of your final executable, you should remove from the following
	// the specific initialization routines you do not need
	// Change the registry key under which our settings are stored
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));

	//start the login dialog first
	LoginDialog ldlg;
	ldlg.DoModal();
```

```
	CChatClientDlg* dlg = new CChatClientDlg;
	dlg->UserName = ldlg.UserName;
	dlg->IP = ldlg.IP;
	CPropertySheet* sheet = new CPropertySheet;
	sheet->AddPage(dlg);
	sheet->SetTitle("Chat Client");
	sheet->Create();
	m_pMainWnd = sheet;
	sheet->ShowWindow(SW_SHOWDEFAULT);
	return TRUE;
}
```

Chat ClientDlg.h

```
	// Chat ClientDlg.h : header file
//

#pragma once
#include "afxwin.h"
#include <map>
#include "ClientSock.h"
#include "CustListBox.h"
using namespace std;

// CChatClientDlg dialog
class CChatClientDlg : public CPropertyPage
{
// Construction
public:
	CChatClientDlg(/*CWnd* pParent = NULL*/);	// standard constructor
	CString UserName;
// Dialog Data
	enum { IDD = IDD_CHATCLIENT_DIALOG };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV
support


// Implementation
protected:
	HICON m_hIcon;
	// Generated message map functions
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	DECLARE_MESSAGE_MAP()
public:
	ClientSock sock;
	CString IP;
	map<CString,CPropertyPage*> UserToTab,RoomToTab;
	CPropertySheet PropSheet;
	bool ColorSelected;
```

```
	COLORREF color;
	LRESULT OnNetworkData(WPARAM,LPARAM);
	LRESULT OnSendData(WPARAM,LPARAM);
public:
	CString TextBox;
	CListBox ConnectedUsers;
	CustListBox ChatList;
	afx_msg void OnBnClickedSend();
	afx_msg void OnBnClickedColor();
	LRESULT OnConnectionMade(WPARAM,LPARAM);
	BOOL PreTranslateMessage(MSG* pMsg);
};
```

Chat ClientDlg.cpp

```
// Chat ClientDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Client.h"
#include "Chat ClientDlg.h"
#include "PropertyPage.h"
#include "messages.h"
#include "ClientSock.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#endif

#define PORTNO 4945
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	enum { IDD = IDD_ABOUTBOX };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}
```

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CChatClientDlg dialog

CChatClientDlg::CChatClientDlg(/*CWnd* pParent =NULL*/)
	: CPropertyPage(CChatClientDlg::IDD)
	, TextBox(_T(""))
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	ColorSelected = false;
}
LRESULT CChatClientDlg::OnNetworkData(WPARAM data,LPARAM)
{
	//all the data to a client will be recieved here
	//and then the private messages or room messages will
	//be forwarded to tabs accordingly
	CString *Data = (CString *)data;
	//<CS>RLST:RoomName:room_list<CE>
	if(Data->GetAt(1)=='C')
	{
		if(Data->Mid(4,strlen("RLIST")) == "RLIST")
		{
			//extract the room list
			CString RoomList;
			for(int i=strlen("<CS>RLIST:");Data->GetAt(i)!='<';i++)
				RoomList+=Data->GetAt(i);
			ChatList.AddString("Rooms are");
			CString temp;
			for(int i=0;i<RoomList.GetLength();i++)
			{
				if(RoomList[i] == ',')
				{
					CString *OwnStr = new CString;
					*OwnStr = temp;
					ChatList.AddString(*OwnStr);

					temp = "";
				}
				else temp += RoomList[i];
			}
			CString* OwnStr = new CString;
			*OwnStr = temp;
			ChatList.AddString(*OwnStr);
			CString* msg = new CString;
			for(int i=4;Data->GetAt(i)!='<';i++)
				*msg+=Data->GetAt(i);
			//also send rlist to every room
			map<CString,CPropertyPage*>::iterator iter;
			for(iter=RoomToTab.begin();iter!=RoomToTab.end();iter++)
				iter->second->PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
		}
		else if(Data->Mid(4,strlen("RLST")) == "RLST")
		{
			//and send it to the specific tab
			//check if this is for admin
			CString RoomName("");
			for(int i=strlen("<CS>RLST:");Data->GetAt(i)!=':';i++)
```

```
                        RoomName+=Data->GetAt(i);
                    if(RoomName == "admin")
                    {
                        //then since this is the tab for admin
                        //extract the list
                        int loc = Data->ReverseFind(':') + 1;
                        CString UserList("");
                        for(;Data->GetAt(loc)!='<';loc++)
                            UserList+=Data->GetAt(loc);
                        //extract the users from the list and display them
                        CString User("");
                        for(int i=0;i<UserList.GetLength();i++)
                        {
                            if(UserList[i]==',')
                            {
                                //update the users list
                                ConnectedUsers.AddString(User);
                                User="";
                            }
                            else User+=UserList[i];
                        }
                        ConnectedUsers.AddString(User);
                    }
                    else //if it was not for admin then send to tab
                    {
                        CString *msg =  new CString;
                        //extract the room list
                        int loc = Data->ReverseFind(':') + 1;
                        CString UserList("");
                        for(;Data->GetAt(loc)!='<';loc++)
                            UserList+=Data->GetAt(loc);

                        *msg = CString("RLST:") + UserList;
                        ((CPropertySheet *)GetParent())-
>SetActivePage(RoomToTab[RoomName]);
                        RoomToTab[RoomName]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                    }
                }
                //<CS>TPC:ROOMNAME:TOPIC<CE>
                else if(Data->Mid(4,strlen("TPC")) == "TPC")
                {
                    //extract the room name
                    int loc;
                    CString RoomName;
                    for(loc=strlen("<CS>TPC:");Data->GetAt(loc)!=':';loc++)
                        RoomName+=Data->GetAt(loc);
                    CString Topic;
                    for(++loc;Data->GetAt(loc)!='<';loc++)
                        Topic+=Data->GetAt(loc);

                    //check if this was for admin room
                    if(RoomName == "admin")
                    {
                        CString* OwnStr = new CString;
                        *OwnStr = CString("New Topic is ") + Topic;
                        ChatList.AddString(*OwnStr);
                        ChatList.PostMessage(WM_LBUTTONDOWN);
```

```
                        ChatList.PostMessage(WM_LBUTTONUP);
                }
                else
                {
                        CString *msg =  new CString;
                        *msg = CString("TPC:")+Topic;
                        RoomToTab[RoomName]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                }
        }
        //<CS>PMSG:FROMUSER:private message<CE>
        else if(Data->Mid(4,strlen("PMSG")) == "PMSG")
        {
                //send the message to the tab
                //after reducing it to "<DS>user:msg<DE>"
                CString *msg =  new CString;
                int loc = Data->Find(':') + 1;
                CString User;
                for(int i=loc;Data->GetAt(i)!=':';i++)
                        User+=Data->GetAt(i);
                for(;Data->GetAt(loc)!='<';loc++)
                        *msg+=Data->GetAt(loc);
                *msg = "<DS>" + *msg + "<DE>";
                UserToTab[User]->PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
        }
        //<CS>NSR:room name<CE>
        else if(Data->Mid(4,strlen("NSR")) == "NSR")
        {
                //send to active tab
        }
        //<CS>LFT:room name<CE>
        else if(Data->Mid(4,strlen("LFT")) == "LFT")
        {
                //data will be processed in admin room(this)
                CString RoomName;
                int loc = Data->Find(':') + 1;
                for(;*Data[loc]!='<';loc++)
                        RoomName+=*Data[loc];
                CString* OwnStr = new CString;
                *OwnStr = CString("Room ") + RoomName + CString(" successfully
left");
                ChatList.AddString(*OwnStr);

        }
        //<CS>CON:room name<CE>
        else if(Data->Mid(4,strlen("CON")) == "CON")
        {
                //data will be processed in admin room(this)
                CString RoomName;
                int loc = Data->Find(':') + 1;
                for(;Data->GetAt(loc)!='<';loc++)
                        RoomName+=Data->GetAt(loc);
                CString* OwnStr = new CString;
                *OwnStr = CString("Room ") + RoomName + CString(" successfully
joined");
                ChatList.AddString(*OwnStr);

                //create a new tab
```

```
                        PropertyPage* page = new PropertyPage;
                        page->m_psp.pszTitle = RoomName;
                        page->m_psp.dwFlags |=PSP_USETITLE;
                        page->RoomName = RoomName;
                        page->OwnName = UserName;
                        page->pMainDlg = this;
                        page->PrivateTab = false;
                        RoomToTab[RoomName] = page;
                        ((CPropertySheet *)GetParent())->AddPage(page);
                }
                //<CS>msg:Requesting user<CE>
                else if(Data->Mid(4,strlen("msg")) == "msg")
                {
                        //open a new tab
                        PropertyPage* page = new PropertyPage;
                        //extract the user name
                        CString ReqUserName;
                        int loc = Data->Find(':') + 1;
                        for(;Data->GetAt(loc)!='<';loc++)
                                ReqUserName+=Data->GetAt(loc);
                        page->m_psp.pszTitle = ReqUserName;
                        page->m_psp.dwFlags |=PSP_USETITLE;
                        page->UserName = ReqUserName;
                        page->OwnName = UserName;
                        page->pMainDlg = this;
                        page->PrivateTab = true;
                        UserToTab[ReqUserName] = page;
                        ((CPropertySheet *)GetParent())->AddPage(page);
                        ((CPropertySheet *)GetParent())->SetActivePage(page);
                }
                //<CS>USRLFT:RoomName:UserName<CE>
                else if(Data->Mid(4,strlen("USRLFT")) == "USRLFT")
                {
                        //extract the room name and send it as "USRLFT:UserName" to the
tab
                        CString UserName,Room_Name;
                        int loc = Data->Find(':') + 1;
                        for(;Data->GetAt(loc)!=':';loc++)
                                Room_Name+=Data->GetAt(loc);
                        for(++loc;Data->GetAt(loc)!='<';loc++)
                                UserName+=Data->GetAt(loc);
                        //remove the user from this room (admin) and send the message to
all the tabs
                        if(Room_Name=="admin")
                        {
                             loc = ConnectedUsers.FindStringExact(0,UserName);
                        if(loc!=LB_ERR)
                                ConnectedUsers.DeleteString(loc);
                        }
                        //tabs will remove the user if it exists in them
                        //and also remove it from UserToTab mapping
                        //and close the private chat if one user leaves
                        //send the messge to tabs
                        map<CString,CPropertyPage*>::iterator iter;
                        for(iter = RoomToTab.begin();iter!=RoomToTab.end();iter++)
                        {
                                CString *msg = new CString;
                                *msg = "USRLFT:" + Room_Name + ":" + UserName;
```

```
                        //*msg = *Data;
                        iter->second-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg,NULL);
                }
                if(Room_Name=="admin")
                {
//if a user has left the admin room it means he has closed the chat window
//and so the private tabs that correspond to ti must also get closed
                        map<CString,CPropertyPage*>::iterator itere;
                        for(itere =
UserToTab.begin();itere!=UserToTab.end();itere++)
                        {
                                if(itere->first == UserName) //this user's private
chat room
                                {
                                        CString *msg = new CString;
                                        *msg = "USRLFT:"+ Room_Name + ":" + UserName;
                                        //*msg = *Data
                                        itere->second-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg,NULL);
                                        UserToTab.erase(itere);
                                        break;
                                }
                        }
                }
        }
        }
        else //it is chat message
        {
                if(Data->GetAt(1)=='D')
                {
                        //check the message is for which room
                        //and display it accordingly
                        CString RoomName;
                        //<DS>ROOM:User:Message<DE>
                        int i;
                        for(i=strlen("<DS>");Data->GetAt(i)!=':';i++)
                                RoomName+=Data->GetAt(i);
                        if(RoomName=="admin")
                        {
                                        //filter out the room name
                                        CString msg;
                                        for(int i=strlen("<DS>admin:");Data-
>GetAt(i)!='<';i++)
                                                msg+=Data->GetAt(i);
                                        CString *OwnStr = new CString;
                                        *OwnStr = msg;
                                        ChatList.AddString(*OwnStr);

                        }
                        else
                        {
                                //filter out the room name
                                CString *msg = new CString;
                                for(int i=Data->Find(':') + 1;Data-
>GetAt(i)!='<';i++)
                                        *msg+=Data->GetAt(i);
                                *msg = CString("<DS>") + *msg + CString("<DE>");
```

```
                                    RoomToTab[RoomName]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                                }
                        }
                }
                return 0;
}

void CChatClientDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        DDX_Text(pDX, IDC_TEXT, TextBox);
        DDX_Control(pDX, IDC_CONNECTED_USERS, ConnectedUsers);
        DDX_Control(pDX, IDC_CHATLIST, ChatList);
}

BEGIN_MESSAGE_MAP(CChatClientDlg, CDialog)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_MESSAGE(UWM_NETWORK_DATA,OnNetworkData)
        ON_MESSAGE(UWM_SEND_DATA,OnSendData)
        ON_MESSAGE(UWM_CONNECTIONMADE,&CChatClientDlg::OnConnectionMade)
        //}}AFX_MSG_MAP
        ON_BN_CLICKED(IDC_SEND, &CChatClientDlg::OnBnClickedSend)
        ON_BN_CLICKED(IDC_COLOR, &CChatClientDlg::OnBnClickedColor)
END_MESSAGE_MAP()


// CChatClientDlg message handlers

BOOL CChatClientDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Add "About..." menu item to system menu.

        // IDM_ABOUTBOX must be in the system command range.
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu != NULL)
        {
                CString strAboutMenu;
                strAboutMenu.LoadString(IDS_ABOUTBOX);
                if (!strAboutMenu.IsEmpty())
                {
                        pSysMenu->AppendMenu(MF_SEPARATOR);
                        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
                }
        }

        if (!AfxSocketInit())
        {
                AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
                return FALSE;
        }
```

```
	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);// Set small icon

	if(!sock.Create())
		MessageBox("Create failed");
	sock.SetTarget(this);
	//try about ten times to connect
	int i=0;

	int Error;
	//while(!(Error=sock.Connect(IP,PORTNO)) && i++<10);
	Error = sock.Connect(IP,PORTNO);
	if(Error==0)
	{

	}

	ConnectedUsers.AddString(UserName);
	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CChatClientDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CChatClientDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
```

```
			dc.DrawIcon(x, y, m_hIcon);
		}
		else
		{
			CDialog::OnPaint();
		}
}

// The system calls this function to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CChatClientDlg::OnQueryDragIcon()
{
		return static_cast<HCURSOR>(m_hIcon);
}

LRESULT CChatClientDlg::OnSendData(WPARAM data,LPARAM)
{
		//send the data
		CString *Data = (CString *)data;
		sock.Send(Data->GetBuffer(),Data->GetLength());
		return 0;
}

void CChatClientDlg::OnBnClickedSend()
{
		UpdateData(TRUE);
		TextBox.Trim();
		CString *data = new CString;
		if(TextBox[0]=='/')
		{
			if(TextBox.Mid(0,strlen("/Room_List")) == "/Room_List")
				*data = CString("<CS>") + TextBox + CString("<CE>");
			else if(TextBox.Mid(0,strlen("/Join_Room")) == "/Join_Room")
			{
				//extract room name
				CString RoomName;
				for(int i=strlen("/Join_Room") + 1;i<TextBox.GetLength();i++)
					RoomName+=TextBox[i];
//build the command string,admin since this is the admin room
				*data = CString("<CS>/Join_Room:") + UserName + ":" + RoomName +
CString("<CE>");
			}
			else if(TextBox.Mid(0,strlen("/Create_Room")) == "/Create_Room")
				*data = CString("<CS>") + TextBox + CString("<CE>");
			else if(TextBox.Mid(0,strlen("/RoomTopic")) == "/RoomTopic")
			{
				//extract topic
				CString topic;
				for(int i=strlen("/RoomTopic") + 1;i<TextBox.GetLength();i++)
					topic+=TextBox[i];
				//package as <CS>/RoomTopic:room:topic<CE>
				//display the topic in this room
				//ChatList.AddString("New Room topic is" + topic);
				*data = CString("<CS>/RoomTopic:admin:") + topic +
CString("<CE>");
			}
			else if(TextBox.Mid(0,strlen("/msg")) == "/msg")
			{
```

```
                    //extract the user name
                    CString ReqUserName;
                    for(int i=strlen("/msg") + 1;i<TextBox.GetLength();i++)
                          ReqUserName+=TextBox[i];
                    *data = CString("<CS>/msg ") + ReqUserName + CString("<CE>");
                    //open a new tab
                    PropertyPage* page = new PropertyPage;
                    page->PrivateTab = true;
                    page->m_psp.pszTitle = ReqUserName;
                    page->m_psp.dwFlags |=PSP_USETITLE;
                    page->UserName = ReqUserName;
                    page->OwnName = UserName;
                    page->pMainDlg = this;
                    UserToTab[ReqUserName] = page;
                    ((CPropertySheet *)GetParent())->AddPage(page);
                    ((CPropertySheet *)GetParent())->SetActivePage(page);
                    //PostMessage(UWM_NETWORK_DATA,(WPARAM)data);
              }
              else if(TextBox.Mid(0,strlen("/Leave_Room")) == "/Leave_Room")
              {
                    //extract the user name
                    CString RoomName;
                    for(int i=strlen("/Leave_Room") + 1;i<TextBox.GetLength();i++)
                          RoomName+=TextBox[i];
                    *data = CString("<CS>/Leave_Room:") + RoomName + CString("<CE>");
              }
          }
              else //<DS>ROOM:User:Message<DE>
              {
                    //display in this dialog
                    //add the user name to it
                    CString ColorString;
                    if(ColorSelected)
ColorString.Format("(c:%d,%d,%d)",GetRValue(color),GetGValue(color),GetBValue(color));
                    CString *OwnStr = new CString;
                    *OwnStr = UserName + ": " + ColorString +TextBox;
                    ChatList.AddString(*OwnStr);
                    //ChatList.AddString(UserName + ": " + ColorString +TextBox);
                    //ChatList.PostMessage(WM_LBUTTONDOWN);
                    //ChatList.PostMessage(WM_LBUTTONUP);
                    *data = CString("<DS>admin:") + UserName + CString(": ") +
ColorString + TextBox + CString("<DE>");
              }
       if(PostMessage(UWM_SEND_DATA,(WPARAM)data,NULL)){}
       else MessageBox("PostMessage failed");
       TextBox="";
       UpdateData(FALSE);
}



void CChatClientDlg::OnBnClickedColor()
{
        CColorDialog cdlg;
        if(cdlg.DoModal()==IDOK)
        {
              ColorSelected = true;
```

```
                color = cdlg.GetColor();
        }
        else ColorSelected = false;
}

LRESULT CChatClientDlg::OnConnectionMade(WPARAM,LPARAM)
{
       CString* com = new CString;
       *com = CString("<CS>/Join_Room:") + UserName + CString(":admin<CE>");
       int i=0;
       int ErrorState;
       sock.Send(com->GetBuffer(),com->GetLength());

       return 0;
}
BOOL CChatClientDlg::PreTranslateMessage(MSG* pMsg)
{
       if (pMsg->message == WM_KEYDOWN && pMsg->wParam == VK_RETURN)
              OnBnClickedSend();
       return CPropertyPage::PreTranslateMessage(pMsg);
}
```

ClientSock.h

```
#pragma once

// ClientSock command target
#define DATA_IN_CONTINUATION 1
#define MESSAGE_DATA 2
#define COMMAND_DATA 3
#define DATA_START 4
#define DATA_END 5
#include "messages.h"
class ClientSock : public CAsyncSocket
{
public:
       ClientSock();
       private:
              CString Data;
              CWnd *target;
       public:
              void SetTarget(CWnd* h){target = h;}
              int check_data(CString data)
              {
                     //RETURNS:
                     /*
                      DATA_IN_CONTINUATION
                      MESSAGE_DATA
                      COMMAND_DATA
                     */
                     //check if the data is complete
                     int ret = 0;
                     int sz = data.GetLength();

                     if((data[0]=='<'&&data[1]=='C'&&data[2]=='S'&&data[3]=='>')||
                            (data[0]=='<'&&data[1]=='D'&&data[2]=='S'&&data[3]=='>'))
```

```
                    {
                          ret = DATA_IN_CONTINUATION;

                          //check if ending tag is also present
                          if(data[sz-4]=='<'&&data[sz-3]=='C'&&data[sz-2]=='E'&&data[sz-
1]=='>')
                                ret = COMMAND_DATA;
                          else if (data[sz-4]=='<'&&data[sz-3]=='D'&&data[sz-
2]=='E'&&data[sz-1]=='>')
                                ret = MESSAGE_DATA;
                    }
                    return ret;
             }
             void OnReceive(int nErrorCode)
             {
                    if(nErrorCode==0)
                    {
                          char data[100];
                          int nRead = Receive(data,100);
                          if(nRead != SOCKET_ERROR)
                          {
                                data[nRead] = '\0';

      if(check_data(CString(data))==MESSAGE_DATA||check_data(CString(data))==COMMAND_DATA)
                                {
                                      Data+=data;
                                      CString *DataToBeSent = new CString;
                                      *DataToBeSent = Data;
                                      //sent the message to dialog box
                                      if(!target->PostMessage(UWM_NETWORK_DATA,
(WPARAM)DataToBeSent, (LPARAM)::GetCurrentThreadId()))
                                      { /* failed to send */
                                            ASSERT(FALSE);
                                            delete DataToBeSent;
                                      } /* failed to send */
                                      //clear the internal data store
                                      Data="";
                                }
                                else if(check_data(CString(data))==DATA_IN_CONTINUATION)
                                      //Store the data
                                      Data+=data;
                          }
                    }
                    CAsyncSocket::OnReceive(nErrorCode);
             }
      void OnSendData(WPARAM data)
      {
             CString *Data = new CString;
             *Data = *(CString *)data;
             Send(Data->GetBuffer(),Data->GetLength());
             //7return 0;
      }

      void OnConnect(int nErrorCode)   // CMyAsyncSocket is
// derived from CAsyncSocket
{
      //AfxMessageBox("OnConnect");
   if (0 != nErrorCode)
```

```
   {
      switch( nErrorCode )
      {
         case WSAEADDRINUSE:
            AfxMessageBox("The specified address is already in use.\n");
            break;
         case WSAEADDRNOTAVAIL:
            AfxMessageBox("The specified address is not available from the local
machine.\n");
            break;
         case WSAEAFNOSUPPORT:
            AfxMessageBox("Addresses in the specified family cannot be used with this
socket.\n");
            break;
         case WSAECONNREFUSED:
            AfxMessageBox("The attempt to connect was forcefully rejected.\n");
            break;
         case WSAEDESTADDRREQ:
            AfxMessageBox("A destination address is required.\n");
            break;
         case WSAEFAULT:
            AfxMessageBox("The lpSockAddrLen argument is incorrect.\n");
            break;
         case WSAEINVAL:
            AfxMessageBox("The socket is already bound to an address.\n");
            break;
         case WSAEISCONN:
            AfxMessageBox("The socket is already connected.\n");
            break;
         case WSAEMFILE:
            AfxMessageBox("No more file descriptors are available.\n");
            break;
         case WSAENETUNREACH:
            AfxMessageBox("The network cannot be reached from this host at this time.\n");
            break;
         case WSAENOBUFS:
            AfxMessageBox("No buffer space is available. The socket cannot be
connected.\n");
            break;
         case WSAENOTCONN:
            AfxMessageBox("The socket is not connected.\n");
            break;
         case WSAENOTSOCK:
            AfxMessageBox("The descriptor is a file, not a socket.\n");
            break;
         case WSAETIMEDOUT:
            AfxMessageBox("The attempt to connect timed out without establishing a
connection. \n");
            break;
         default:
            TCHAR szError[256];
            wsprintf(szError, "OnConnect error: %d", nErrorCode);
            AfxMessageBox(szError);
            break;
      }
      AfxMessageBox("Please close the application");
   }
   else
```

```
    {
        target->PostMessage(UWM_CONNECTIONMADE);
    }
    CAsyncSocket::OnConnect(nErrorCode);
}

    virtual ~ClientSock();
};
```

ClientSock.cpp

```
// lientSock.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Client.h"
#include "ClientSock.h"


// ClientSock

ClientSock::ClientSock()
{
}

ClientSock::~ClientSock()
{
}


// ClientSock member functions
```

CustListBox.h

```
#pragma once

#include<vector>
using namespace std;
// CustListBox

class CustListBox : public CListBox
{
    DECLARE_DYNAMIC(CustListBox)

public:
    CustListBox();
    virtual ~CustListBox();
    virtual int CompareItem(LPCOMPAREITEMSTRUCT lpCompareItemStruct);
    virtual void MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct);
    virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
    void TextOut(int x,int y,CDC* dc,CString txt);
```

```
protected:
	DECLARE_MESSAGE_MAP()
};
```

CustListBox.cpp

```
// ustListBox.cpp : implementation file
//

#include "stdafx.h"
#include "CustListBox.h"
#include "resource.h"

// CustListBox

IMPLEMENT_DYNAMIC(CustListBox, CListBox)

CustListBox::CustListBox()
{

}

CustListBox::~CustListBox()
{
}

int CustListBox::CompareItem(LPCOMPAREITEMSTRUCT lpCompareItemStruct){return 0;}
void CustListBox::MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
{
	//lpMeasureItemStruct->itemHeight = 15;
}
void CustListBox::TextOut(int x,int y,CDC* dc,CString txt)
{
	CString str("");
	//display the string part anf then if a image occurs then display it
	/*
	color formatted string
	(c:R,G,B) string
	*/
	bool PenSel = false;
	int i = txt.Find(':') + 2,ind;
	TRACE("\ntxt = %s i = %d",txt,i);
	//due to the format being user: message
	//select the pen according to the color
	if(txt.GetLength()>3&&(txt[i]=='(' && txt[i+1]=='c' && txt[i+2]==':'))
	{
		ind = i;
		//extract the R,G,B values
		BYTE R = 0,G = 0,B = 0;

		for(i+=3;isdigit(txt[i]);i++)
			R=R*10+(txt[i]-'0');
```

```
			for(++i;isdigit(txt[i]);i++)
				G=G*10+(txt[i]-'0');
			for(++i;isdigit(txt[i]);i++)
				B=B*10+(txt[i]-'0');
			//TRACE("\nR=%d  G=%d  B=%d",R,G,B);
			dc->SetTextColor(RGB(R,G,B));
			PenSel=true;
			//now erase color command from the string
			txt.Delete(ind,(i+1)-ind);
		}
		int indent = x;
		for(i=0;i<txt.GetLength();i++)
		{
			//check for emoticons
				if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='P'))
				{
					dc->TextOut(indent,y,str,str.GetLength());
					indent+=dc->GetTextExtent(str,str.GetLength()).cx;
					HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_TONGUE_SMILE),IMAGE_BITMAP,15,
					dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
					indent+=15;
					str="";
					i++;
				}
				else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]==')'))
				{
					dc->TextOut(indent,y,str,str.GetLength());
					indent+=dc->GetTextExtent(str,str.GetLength()).cx;
					HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_SMILE),IMAGE_BITMAP,15,15,LR_D
					dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
					indent+=15;
					str="";
					i++;
				}
				else if(i+1<txt.GetLength()&&(txt[i]==':'&&(txt[i+1]=='D'||txt[i+1]=='d'
				{
					dc->TextOut(indent,y,str,str.GetLength());
					indent+=dc->GetTextExtent(str,str.GetLength()).cx;
					HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_GRIN),IMAGE_BITMAP,15,15,LR_DE
					dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
					indent+=15;
					str="";
					i++;
				}
				else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='|'))
				{
					dc->TextOut(indent,y,str,str.GetLength());
					indent+=dc->GetTextExtent(str,str.GetLength()).cx;
					HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_STRAIGHT_FACE),IMAGE_BITMAP,15
					dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
					indent+=15;
					str="";
					i++;
				}
```

```
                    else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='('))
                    {
                           dc->TextOut(indent,y,str,str.GetLength());
                           indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                           HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_SAD),IMAGE_BITMAP,15,15,LR_DEF
                           dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                           str="";
                           indent+=15;
                           i++;
                    }
                    else if(i+1<txt.GetLength()&&(txt[i]==';'&&txt[i+1]==')'))
                    {
                           dc->TextOut(indent,y,str,str.GetLength());
                           indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                           HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_WINK),IMAGE_BITMAP,15,15,LR_DE
                           dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                           str="";
                           indent+=15;
                           i++;
                    }
                    else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='o'))
                    {
                           dc->TextOut(indent,y,str,str.GetLength());
                           indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                           HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_SURPRISE),IMAGE_BITMAP,15,15,L
                           dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                           str="";
                           indent+=15;
                           i++;
                    }
                    else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='s'))
                    {
                           dc->TextOut(indent,y,str,str.GetLength());
                           indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                           HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_WORRIED),IMAGE_BITMAP,15,15,LR
                           dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                           str="";
                           indent+=15;
                           i++;
                    }
                    else str+=txt[i];
      }
      if(str!="")
            dc->TextOut(indent,y,str,str.GetLength());
      //restore the black color
      dc->SetTextColor(RGB(0,0,0));
}
void CustListBox::DrawItem(LPDRAWITEMSTRUCT lp)
{
      CDC *dc = CDC::FromHandle(lp->hDC);
      //ASSERT(lp->itemData!=NULL);
      CString str = (char *)lp->itemData;
      TRACE("\nDRAWITEM STR=%s",str);
      TextOut(lp->rcItem.left,lp->rcItem.top,dc,str);
```

```
}


BEGIN_MESSAGE_MAP(CustListBox, CListBox)
END_MESSAGE_MAP()



// CustListBox message handlers
```

LoginDialog.h

```
#pragma once


// LoginDialog dialog

class LoginDialog : public CDialog
{
	DECLARE_DYNAMIC(LoginDialog)

public:
	LoginDialog(CWnd* pParent = NULL);   // standard constructor
	virtual ~LoginDialog();
// Dialog Data
	enum { IDD = IDD_LOGINDIALOG };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
	BOOL OnInitDialog();
	DECLARE_MESSAGE_MAP()
public:
	CString UserName;
	afx_msg void OnBnClickedLogin();
	CString IP;
};
```

LoginDialog.cpp

```
// LoginDialog.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Client.h"
#include "LoginDialog.h"


// LoginDialog dialog

IMPLEMENT_DYNAMIC(LoginDialog, CDialog)
```

```
LoginDialog::LoginDialog(CWnd* pParent /*=NULL*/)
	: CDialog(LoginDialog::IDD, pParent)
	, UserName(_T(""))
	, IP(_T(""))
{

}

LoginDialog::~LoginDialog()
{
}
BOOL LoginDialog::OnInitDialog()
{
	CDialog::OnInitDialog();
	IP = "127.0.0.1";
	UserName = "sd";
	UpdateData(FALSE);
	return TRUE;
}
void LoginDialog::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_USERNAME, UserName);
	DDX_Text(pDX, IDC_IP, IP);
}


BEGIN_MESSAGE_MAP(LoginDialog, CDialog)
	ON_BN_CLICKED(IDC_LOGIN, &LoginDialog::OnBnClickedLogin)
END_MESSAGE_MAP()


// LoginDialog message handlers

void LoginDialog::OnBnClickedLogin()
{
	UpdateData(TRUE);
	EndDialog(0);
}
```

PropertyPage.h

```
#pragma once
#include "afxwin.h"
#include "ClientSock.h"
#include "Chat ClientDlg.h"
#include "CustListBox.h"
// PropertyPage dialog

class PropertyPage : public CPropertyPage
{
	DECLARE_DYNAMIC(PropertyPage)

public:
	COLORREF color;
```

```
	bool ColorSelected;
	bool leave;
	bool PrivateTab;  //True if private chat tab ,False is Room Chat tab
	CString UserName; //in case of PrivateTab
	CString RoomName; //in case of Room chat
	CString OwnName; //own user name
	CChatClientDlg *pMainDlg;

	PropertyPage();
	virtual ~PropertyPage();
	LRESULT OnNetworkData(WPARAM,LPARAM);
	BOOL PreTranslateMessage(MSG* pMsg);
// Dialog Data
	enum { IDD = IDD_PROPERTYPAGE };

protected:
	BOOL OnInitDialog();
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

	DECLARE_MESSAGE_MAP()
public:
	CString TextBox;
	CListBox ConnectedUsers;
	CustListBox ChatList;
	afx_msg void OnBnClickedSend();
	afx_msg void OnBnClickedColor();
};
```

PropertyPage.cpp

```
// PropertyPage.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Client.h"
#include "PropertyPage.h"
#include "messages.h"
//#include "ClientSock.h"
// PropertyPage dialog

IMPLEMENT_DYNAMIC(PropertyPage, CPropertyPage)

PropertyPage::PropertyPage()
	: CPropertyPage(PropertyPage::IDD)
	, TextBox(_T(""))
{
	leave = false;
	PrivateTab = false;
	ColorSelected = false;
}

PropertyPage::~PropertyPage()
{
}
```

```
BOOL PropertyPage::OnInitDialog()
{
	CPropertyPage::OnInitDialog();
	ConnectedUsers.AddString(OwnName);
	if(PrivateTab)
		ConnectedUsers.AddString(UserName);
	return TRUE;
}
void PropertyPage::DoDataExchange(CDataExchange* pDX)
{
	CPropertyPage::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_TEXT, TextBox);
	DDX_Control(pDX, IDC_CONNECTED_USERS, ConnectedUsers);
	DDX_Control(pDX, IDC_CHATLIST, ChatList);
}
LRESULT PropertyPage::OnNetworkData(WPARAM data,LPARAM)
{
	/* messages recvd here are;
	"RLIST:room list"
	"TPC:Topic"
	"FROMUSER:private message"
	"USRLFT:UserName"*/
	CString *Data = (CString *)data;
	//hanlde RLIST here
	if(Data->Mid(0,strlen("RLIST")) == "RLIST")
	{
		ChatList.AddString("Room are");
	//	ChatList.PostMessage(WM_LBUTTONDOWN);
	//	ChatList.PostMessage(WM_LBUTTONUP);
		CString temp;
		for(int i=strlen("RLIST:");i<Data->GetLength();i++)
		{
			if(Data->GetAt(i) == ',')
			{
				CString *OwnStr = new CString;
				*OwnStr = temp;
				ChatList.AddString(*OwnStr);
				//ChatList.PostMessage(WM_LBUTTONDOWN);
				//ChatList.PostMessage(WM_LBUTTONUP);
				temp = "";
			}
			else temp += Data->GetAt(i);
		}
		CString *OwnStr = new CString;
		*OwnStr = temp;
		ChatList.AddString(*OwnStr);
		//ChatList.PostMessage(WM_LBUTTONDOWN);
		//ChatList.PostMessage(WM_LBUTTONUP);
	}
	else if(Data->Mid(0,strlen("RLST")) == "RLST")
	{
		CString User("");
		for(int i=strlen("RLST:");i<Data->GetLength();i++)
		{
			if(Data->GetAt(i)==',')
			{
				//update the users list
				ConnectedUsers.AddString(User);
```

```
                        User="";
                    }
                    else User+=Data->GetAt(i);
                }
                ConnectedUsers.AddString(User);
        }
        else if(Data->Mid(0,strlen("TPC")) == "TPC")
        {
                //extract the topic and display
                int loc = Data->Find(':') + 1;
                CString Topic;
                for(;loc<Data->GetLength();loc++)
                        Topic+=Data->GetAt(loc);
                CString *OwnStr = new CString;
                *OwnStr = CString("New Topic is ") + Topic;
                ChatList.AddString(*OwnStr);
                //ChatList.PostMessage(WM_LBUTTONDOWN);
                //ChatList.PostMessage(WM_LBUTTONUP);
        }
        else if(Data->Mid(0,strlen("USRLFT")) == "USRLFT")
        {
                //USRLFT:room name:username
                CString User_Name,Room_Name;
                int loc = Data->Find(':') + 1;
                for(;Data->GetAt(loc)!=':';loc++)
                        Room_Name+=Data->GetAt(loc);
                for(++loc;loc<Data->GetLength();loc++)
                        User_Name+=Data->GetAt(loc);
                CString *OwnStr = new CString;
                *OwnStr = User_Name + CString(" left this room.");
                ChatList.AddString(*OwnStr);
                ChatList.PostMessage(WM_LBUTTONDOWN);
                ChatList.PostMessage(WM_LBUTTONUP);
                //remove from list if the user left this room or the admin room
                if(Room_Name=="admin" || Room_Name==RoomName)
                {
                        loc = ConnectedUsers.FindStringExact(0,User_Name);
                        if(loc!=LB_ERR) ConnectedUsers.DeleteString(loc);
                }
        }
        else //this is general chat message display it as it is
                    //then display in this room
        {
                //extract the message and display it
                //<DS>ROOM:User:Message<DE>
                CString msg("");
                for(int i = 4;Data->GetAt(i)!='<';i++)
                        msg+=Data->GetAt(i);
                CString *OwnStr = new CString;
                *OwnStr = msg;
                ChatList.AddString(*OwnStr);
                ChatList.PostMessage(WM_LBUTTONDOWN);
                ChatList.PostMessage(WM_LBUTTONUP);
        }
        return 0;
}

BEGIN_MESSAGE_MAP(PropertyPage, CPropertyPage)
```

```
	ON_MESSAGE(UWM_NETWORK_DATA,OnNetworkData)
	ON_BN_CLICKED(IDC_SEND, &PropertyPage::OnBnClickedSend)
	ON_BN_CLICKED(IDC_COLOR, &PropertyPage::OnBnClickedColor)
END_MESSAGE_MAP()


// PropertyPage message handlers

void PropertyPage::OnBnClickedSend()
{
	UpdateData(TRUE);
	TextBox.Trim();
	CString* data = new CString;
	if(TextBox[0]=='/')
	{
		if(TextBox.Mid(0,strlen("/Room_List")) == "/Room_List")
			*data = CString("<CS>") + TextBox + CString("<CE>");
		else if(TextBox.Mid(0,strlen("/Join_Room")) == "/Join_Room")
		{
			//extract room name
			CString Room_Name;
			for(int i=strlen("/Join_Room") + 1;i<TextBox.GetLength();i++)
				Room_Name+=TextBox[i];
			//build the command string
			*data = CString("<CS>/Join_Room:") + OwnName +":"+ Room_Name +
CString("<CE>");
		}
		else if(TextBox.Mid(0,strlen("/Create_Room")) == "/Create_Room")
			*data = CString("<CS>") + TextBox + CString("<CE>");
		else if(TextBox.Mid(0,strlen("/RoomTopic")) == "/RoomTopic")
		{
			//extract topic
			CString topic;
			for(int i=strlen("/RoomTopic") + 1;i<TextBox.GetLength();i++)
				topic+=TextBox[i];
			//package as <CS>/RoomTopic:room:topic<CE>
			*data = CString("<CS>/RoomTopic:")+ RoomName + ":" + topic +
CString("<CE>");
		}
		else if(TextBox.Mid(0,strlen("/msg")) == "/msg")
		{
			//extract the user name
			CString ReqUserName;
			for(int i=strlen("/msg") + 1;i<TextBox.GetLength();i++)
				ReqUserName+=TextBox[i];
			*data = CString("<CS>/msg ") + ReqUserName + CString("<CE>");
			PropertyPage* page = new PropertyPage;
			page->m_psp.pszTitle = ReqUserName;
			page->m_psp.dwFlags |=PSP_USETITLE;
			page->UserName = ReqUserName;
			page->OwnName = OwnName;
			page->pMainDlg = pMainDlg;
			page->PrivateTab = true;
			pMainDlg->UserToTab[ReqUserName] = page;
			((CPropertySheet *)GetParent())->AddPage(page);
			((CPropertySheet *)GetParent())->SetActivePage(page);
		}
		else if(TextBox.Mid(0,strlen("/Leave_Room")) == "/Leave_Room")
```

```
		{
			//extract the user name
			/*CString Room_Name;
			for(int i=strlen("/Leave_Room") + 1;i<TextBox.GetLength();i++)
				Room_Name+=TextBox[i];*/
			if(!PrivateTab)
				*data = CString("<CS>/Leave_Room:") + RoomName +
CString("<CE>");
			leave = true;
		}
	}
		else
		{
			//this was a general user message
			//<DS>ROOM:User:Message<DE>
			//<CS>PMSG:Touser:private message<CE>
			CString ColorString;
			if(ColorSelected)
ColorString.Format("(c:%d,%d,%d)",GetRValue(color),GetGValue(color),GetBValue(color));
			if(PrivateTab)
				*data = CString("<CS>PMSG:") + UserName + ": "+ ColorString
+ TextBox + CString("<CE>");
			else *data = CString("<DS>") + RoomName + ":" + OwnName + ": " +
ColorString + TextBox + "<DE>";
			CString *OwnStr = new CString;
			*OwnStr = OwnName + ": " + ColorString + TextBox;
			ChatList.AddString(*OwnStr);
			//ChatList.PostMessage(WM_LBUTTONDOWN);
			//ChatList.PostMessage(WM_LBUTTONUP);
	    }

	if(pMainDlg->PostMessage(UWM_SEND_DATA,(WPARAM)data,NULL)){}
	else MessageBox("PostMessage failed");
	TextBox="";
	UpdateData(FALSE);
	if(leave) ((CPropertySheet *)GetParent())->RemovePage(this);
}
void PropertyPage::OnBnClickedColor()
{
	 CColorDialog cdlg;
	 if(cdlg.DoModal()==IDOK)
	 {
		ColorSelected = true;
		color = cdlg.GetColor();
	 }
	 else ColorSelected = false;
}
BOOL PropertyPage::PreTranslateMessage(MSG* pMsg)
{
	if (pMsg->message == WM_KEYDOWN && pMsg->wParam == VK_RETURN)
		OnBnClickedSend();
	return CPropertyPage::PreTranslateMessage(pMsg);
}
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently,
// but are changed infrequently

#pragma once
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls'version='6.0.0.0'
processorArchitecture='x86' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#ifndef _SECURE_ATL
#define _SECURE_ATL 1
#endif

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows
headers
#endif

// Modify the following defines if you have to target a platform prior to
the ones specified below.
// Refer to MSDN for the latest info on corresponding values for different
platforms.
#ifndef WINVER                          // Allow use of features specific to
Windows XP or later.
#define WINVER 0x0501           // Change this to the appropriate value to
target other versions of Windows.
#endif

#ifndef _WIN32_WINNT            // Allow use of features specific to
Windows XP or later.
#define _WIN32_WINNT 0x0501     // Change this to the appropriate value to
target other versions of Windows.
#endif

#ifndef _WIN32_WINDOWS          // Allow use of features specific to
Windows 98 or later.
#define _WIN32_WINDOWS 0x0410 // Change this to the appropriate value to
target Windows Me or later.
#endif

#ifndef _WIN32_IE               // Allow use of features specific to IE
6.0 or later.
#define _WIN32_IE 0x0600  // Change this to the appropriate value to target
other versions of IE.
#endif

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS    // some CString constructors
will be explicit

// turns off MFC's hiding of some common and often safely ignored warning
messages
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions


#include <afxdisp.h>        // MFC Automation classes
```

```
#ifndef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>           // MFC support for Internet Explorer 4
Common Controls
#endif
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>             // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


#include <afxsock.h>            // MFC socket extensions
#include <afxdlgs.h>


#ifdef _UNICODE
#if defined _M_IX86
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='x86' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#elif defined _M_IA64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='ia64' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#elif defined _M_X64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#else
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='*' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#endif
#endif
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// Chat Client.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

messages.h

```
/*****************************************************************************
*                                UWM_THREADSTART
* Inputs:
*       WPARAM: unused
*       LPARAM: (WPARAM)(DWORD) Thread ID
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main GUI thread that the child thread has started
*****************************************************************************/
#define UWM_THREADSTART  (WM_APP+200)

/*****************************************************************************
*                                UWM_THREADCLOSE
* Inputs:
*       WPARAM: unused
*       LPARAM: (LPARAM)(DWORD) thread ID of closing thread
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that a secondary thread has terminated
*****************************************************************************/

#define UWM_THREADCLOSE  (WM_APP+201)

/*****************************************************************************
*                                 UWM_NETWORK_DATA
* Inputs:
*       WPARAM: (WPARAM)(CByteArray *) Pointer to the data that was received
*       LPARAM: (LPARAM)(DWORD) thread ID of thread that received the data
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Passes data to the main thread to be processed and/or displayed
* Notes:
*       It is the responsibility of the recipient to delete the CString
*****************************************************************************/

#define UWM_NETWORK_DATA    (WM_APP+202)

/*****************************************************************************
*                                UWM_TERM_THREAD
* Inputs:
*       WPARAM: unused
*       LPARAM: unused
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Sent via PostThreadMessage to the child thread to shut it down
*****************************************************************************/

#define UWM_TERM_THREAD  (WM_APP+203)


/*****************************************************************************
*                              UWM_CONNECTIONCLOSE
* Inputs:
```

```
*       WPARAM: unused
*       LPARAM lParam: (LPARAM)(DWORD) thread ID of thread that closed
connection
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that the connection has closed
********************************************************************************/

#define UWM_CONNECTIONCLOSE (WM_APP + 204)


/********************************************************************************
*                              UWM_CONNECTIONMADE
* Inputs:
*       WPARAM: unused
*       LPARAM: unused
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that a connection has succeeded
********************************************************************************/

#define UWM_CONNECTIONMADE (WM_APP + 205)

/********************************************************************************
*                               UWM_NETWORK_ERROR
* Inputs:
*       WPARAM: (WPARAM)(DWORD) error code
*       LPARAM: (LPARAM)(DWORD) thread ID of thread that had the error
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that a network error has occurred
********************************************************************************/

#define UWM_NETWORK_ERROR (WM_APP+207)

/********************************************************************************
*                               UWM_SEND_COMPLETE
* Inputs:
*       WPARAM: unused
*       LPARAM: (LPARAM)(DWORD) thread ID of thread that completed sending
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the owner that the last Send has completed
********************************************************************************/

#define UWM_SEND_COMPLETE (WM_APP+208)

/********************************************************************************
*                                 UWM_SEND_DATA
* Inputs:
*       WPARAM: (WPARAM)(CByteArray*) Reference to a CByteArray to be sent
*       LPARAM: unused
* Result: LRESULT
*       Logically void, 0, always
```

```
* Effect:
*       Enqueues a send request
* Notes:
*       This is sent to the thread via PostThreadMessage
*       The sender must allocate the CByteArray on the heap.  The network
*       layers will delete this object when it has been sent
*****************************************************************************/

#define UWM_SEND_DATA (WM_APP + 209)

/*****************************************************************************
*                            UWM_START_NEXT_PACKET
* Inputs:
*       WPARAM: unused
*       LPARAM: unused
* Result: void
*
* Effect:
*       This is sent via PostThreadMessage from the network thread to itself
*       to initiate the asynchronous dequeue of the next pending packet
*****************************************************************************/

#define UWM_START_NEXT_PACKET (WM_APP + 210)

/*****************************************************************************
*                            UWM_NEWTARGET
* Inputs:
*       WPARAM: (WPARAM)(CWnd*) address of the new property page
*       LPARAM: unused
* Result: void
*
* Effect:
*       This is sent via PostThreadMessage from the network thread to itself
*       to initiate the asynchronous dequeue of the next pending packet
*****************************************************************************/

#define UWM_NEWTARGET (WM_APP + 211)
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Chat Client.rc
//
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_CHATCLIENT_DIALOG           102
#define IDP_SOCKETS_INIT_FAILED         103
#define IDD_PROPERTYPAGE                105
#define IDD_LOGINDIALOG                 106
#define IDR_MAINFRAME                   128
#define IDB_SMILE                       130
#define IDB_SAD                         131
#define IDB_WINK                        132
#define IDB_GRIN                        133
#define IDB_TONUGE_SMILE                134
```

```
#define IDB_TONGUE_SMILE                134
#define IDB_SURPRISE                    135
#define IDB_WORRIED                     136
#define IDB_STRAIGHT_FACE               137
#define IDC_CONNECTED_USERS             1001
#define IDC_TEXT                        1002
#define IDC_SEND                        1003
#define IDC_USERNAME                    1004
#define IDC_LOGIN                       1005
#define IDC_CHATLIST                    1006
#define IDC_IP                          1010
#define IDC_COLOR                       1011

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        138
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1013
#define _APS_NEXT_SYMED_VALUE           107
#endif
#endif
```

8.2 *Κώδικας υλοποίησης Chat Server*

Chat Server.h

```
// Chat Server.h : main header file for the PROJECT_NAME application
//

#pragma once

#ifndef __AFXWIN_H__
	#error "include 'stdafx.h' before including this file for PCH"
#endif

#include "resource.h"		// main symbols


// CChatServerApp:
// See Chat Server.cpp for the implementation of this class
//

class CChatServerApp : public CWinApp
{
public:
	CChatServerApp();

// Overrides
	public:
	CPropertySheet PropSheet;
	virtual BOOL InitInstance();

// Implementation
```

```
	DECLARE_MESSAGE_MAP()
};

extern CChatServerApp theApp;
```

Chat Server.cpp

```
// Chat Server.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Chat Server.h"
#include "Chat ServerDlg.h"
#include "PropertyPage.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CChatServerApp

BEGIN_MESSAGE_MAP(CChatServerApp, CWinApp)
	ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
END_MESSAGE_MAP()


// CChatServerApp construction

CChatServerApp::CChatServerApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}


// The one and only CChatServerApp object

CChatServerApp theApp;


// CChatServerApp initialization

BOOL CChatServerApp::InitInstance()
{
	// InitCommonControlsEx() is required on Windows XP if an
application
	// manifest specifies use of ComCtl32.dll version 6 or later to
enable
	// visual styles.  Otherwise, any window creation will fail.
	INITCOMMONCONTROLSEX InitCtrls;
	InitCtrls.dwSize = sizeof(InitCtrls);
	// Set this to include all the common control classes you want to
use
	// in your application.
	InitCtrls.dwICC = ICC_WIN95_CLASSES;
	InitCommonControlsEx(&InitCtrls);
```

```
	CWinApp::InitInstance();

	if (!AfxSocketInit())
	{
		AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
		return FALSE;
	}

	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	// of your final executable, you should remove from the following
	// the specific initialization routines you do not need
	// Change the registry key under which our settings are stored
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));

	//PropertyPage* page = new PropertyPage;
	CChatServerDlg* dlg = new CChatServerDlg;
	CPropertySheet* sheet = new CPropertySheet;
	//PropSheet.AddPage(dlg);
	//PropSheet.DoModal();
	sheet->AddPage(dlg);
	sheet->SetTitle("Chat Server");
	sheet->Create();
	m_pMainWnd = sheet;
	sheet->ShowWindow(SW_SHOWDEFAULT);
    return TRUE;
}
```

Chat ServerDlg.h

```
// Chat ServerDlg.h : header file
//

#pragma once
#include "afxwin.h"
#include "ListenSocket.h"
#include "CustListBox.h"
#include <map>
#include <vector>
using namespace std;

// CChatServerDlg dialog
class CChatServerDlg : public CPropertyPage
{
// Construction
public:
	CChatServerDlg(/*CWnd* pParent = NULL*/);	// standard constructor

// Dialog Data
	enum { IDD = IDD_CHATSERVER_DIALOG };
```

```
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

	LRESULT OnThreadStart(WPARAM,LPARAM);
    LRESULT OnThreadClose(WPARAM,LPARAM);
    LRESULT OnNetworkData(WPARAM,LPARAM);

	bool ColorSelected;
	COLORREF color;

// Implementation
public:
	CListenSocket ServerSock;
	vector<vector<DWORD> >  m_threadIDs;
	map<CString,DWORD> UserToThreadID;
	map<CString,int> RoomToThreadIDIndex;
	map<CString,CString> RoomToUsers;
	map<CString,CPropertyPage*> UserToTab,RoomToTab;
	CString RoomList; //comma seperated room list
	CPropertySheet PropSheet;

	HICON m_hIcon;
	// Generated message map functions
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnBnClickedSend();
	CString TextBox;
	CustListBox ChatList; //using the customized listbox for colored
text and emoticons
	CListBox AvailableRooms;
	CListBox ConnectedUsers;
	afx_msg void OnBnClickedColor();
	BOOL PreTranslateMessage(MSG* pMsg);
	CEdit EditBox;
};
```

Chat ServerDlg.cpp

```
// Chat ServerDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Server.h"
#include "Chat ServerDlg.h"
#include "messages.h"
#include "PropertyPage.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
```

```
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	enum { IDD = IDD_ABOUTBOX };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()


// CChatServerDlg dialog



CChatServerDlg::CChatServerDlg()
	: CPropertyPage(CChatServerDlg::IDD)
	, TextBox(_T(""))
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	ColorSelected = false;
}

void CChatServerDlg::DoDataExchange(CDataExchange* pDX)
{
	CPropertyPage::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_TEXTBOX, TextBox);
	DDX_Control(pDX, IDC_CHAT_LIST, ChatList);
	DDX_Control(pDX, IDC_ROOMS, AvailableRooms);
	DDX_Control(pDX, IDC_CONNECTED, ConnectedUsers);
	DDX_Control(pDX, IDC_TEXTBOX,EditBox);
}

BEGIN_MESSAGE_MAP(CChatServerDlg, CPropertyPage)
	//ON_WM_SYSCOMMAND()
	//ON_WM_PAINT()
```

```
	//ON_WM_QUERYDRAGICON()
	//}}AFX_MSG_MAP
	ON_MESSAGE(UWM_THREADSTART, OnThreadStart)
    ON_MESSAGE(UWM_THREADCLOSE, OnThreadClose)
	ON_MESSAGE(UWM_NETWORK_DATA, OnNetworkData)
	ON_BN_CLICKED(IDC_SEND, &CChatServerDlg::OnBnClickedSend)
	ON_BN_CLICKED(IDC_COLOR, &CChatServerDlg::OnBnClickedColor)

END_MESSAGE_MAP()


// CChatServerDlg message handlers

BOOL CChatServerDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.
/*M_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon*/

	//start listening
	AfxSocketInit();
	if(ServerSock.Create(4945))
	{
		if(ServerSock.Listen())
		{
			ChatList.AddString("SERVER: (c:255,0,0)LISTENING ON TCP PORT
4945");
			ChatList.PostMessage(WM_LBUTTONDOWN);
			ChatList.PostMessage(WM_LBUTTONUP);
			ServerSock.SetTarget(this);
		}
		else AfxMessageBox("listen failed");
	}
	//add a admin room
//	Rooms.Add("ADMIN");
	vector<DWORD> admin;
	admin.push_back(0);
	m_threadIDs.push_back(admin);
```

```
	//admin is in the admin room
	RoomToUsers["admin"] = "admin";
	ConnectedUsers.AddString("admin");
	RoomList = "admin";
	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CChatServerDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CChatServerDlg::OnPaint()
{
/*	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()),
0);
/// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CPropertySheet::OnPaint();
	}*/
}

// The system calls this function to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CChatServerDlg::OnQueryDragIcon()
{
	return static_cast<HCURSOR>(m_hIcon);
}

LRESULT CChatServerDlg::OnThreadStart(WPARAM,LPARAM lp)
```

```
{
	//add the thread(new connected user) to the default admin rooom
	 DWORD ThreadID = (DWORD)lp;
	 m_threadIDs[0].push_back(ThreadID);
	 return 0;
}

LRESULT CChatServerDlg::OnNetworkData(WPARAM data,LPARAM ThreadID)
{
	/*
	/Create_Room "Room"
/Join_Room "Room"
/Leave_Room "Room"
/msg User "private message"
/RoomTopic "topic"
*/
	/*
	 check for the server related commands
	 1. /Room_List :- send the list of rooms to the requesting client formatted as
	    <DS><LIST>comma seperated list<DE>

	 2. /Join_Room:user:RoomName :- search the particular room attach it to the new
room and send <DS>connected:Room topic<DE>and
	       create a new property sheet containing two list boxes and a edit box to
type the message and also send a message
	       to the thread indicating the new parent.
	    and if no such room exists send <DS>**No Such room<DS>

	 3. /Create_Room room :- create a new room named room and insert the requesting
thread in this room

	 4. /Leave_Room :- This command is to be given the room, server will remove te
threadID from that
	                  particular room threadID list.(this command will work in
every room except this admin room)

	 5. /RoomTopic topic :-  set the topic of the room but this will work only if
given by the room owner thread
	                       room owner threadIDs will  be maintained in a seperate
array

	 */
	//every command will be enclosed between <CS>command<CE>
	//every message data will be enclosed between <DS>message<DE>

	//check if this is a command message
	CString *Data = (CString *)data;
	DWORD threadID = (DWORD) ThreadID;
	CString com,msg;
	if(Data->GetAt(1)=='C')
	{
		int loc = Data->Find("<CE>");
		//extract the command
		com = Data->Mid(4,loc-4);
		//parse the command <CS>/Room_List<CE>
		if(com.Mid(0,strlen("/Room_List")) == "/Room_List")
		{
			if(threadID!=NULL)
```

```
                    {
                        //send the room list
                        //<CS>RLIST:comma seperated room list<CE>
                        CString *msg = new CString;
                        *msg = CString("<CS>RLIST:") + RoomList + "<CE>";
                        ::PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)
msg,NULL);
                    }
                    else //the command was from server itself
                    {
                        //display the output in the list box
                        ChatList.AddString("Rooms are");
                        ChatList.PostMessage(WM_LBUTTONDOWN);
                        ChatList.PostMessage(WM_LBUTTONUP);
                        CString temp;
                        for(int i=0;i<RoomList.GetLength();i++)
                        {
                            if(RoomList[i] == ',')
                            {
                                CString *OwnStr = new CString;
                                *OwnStr = temp;
                                ChatList.AddString(*OwnStr);
                                ChatList.PostMessage(WM_LBUTTONDOWN);
                                ChatList.PostMessage(WM_LBUTTONUP);
                                temp = "";
                            }
                            else temp += RoomList[i];
                        }
                        CString *OwnStr = new CString;
                        *OwnStr = temp;
                        ChatList.AddString(*OwnStr);
                        ChatList.PostMessage(WM_LBUTTONDOWN);
                        ChatList.PostMessage(WM_LBUTTONUP);
                    }
                }
                // /Join_Room:user:RoomName
                else if(com.Mid(0,strlen("/Join_Room")) == "/Join_Room")
                {
                    //extract the room name
                    int loc = com.ReverseFind(':');
                    CString Room_name;
                    Room_name = com.Mid(loc + 1 , com.GetLength());
                    //then extract the name of the user name
                    int loc2 = com.Find(':');
                    CString UserName = com.Mid(loc2 +1 ,loc - loc2-1);
                    //check if a new user has connected by check for "admin"
                    if(Room_name == "admin")
                    {
                        UserToThreadID[UserName] = threadID;
                        ConnectedUsers.AddString(UserName);
                        //display a log message about this new user
                        CString *OwnStr = new CString;
                        *OwnStr = "SERVER: (c:255,0,0)" + UserName + " JOINED THE
SERVER";
                        ChatList.AddString(*OwnStr);
                        //ChatList.PostMessage(WM_LBUTTONDOWN);
                        //ChatList.PostMessage(WM_LBUTTONUP);
                        //also send list of users connected to this room
```

```
                        //generate a comma seperated list of the users and send
                        //to this newly joined user and just this user's name to
                        //all the other clients in this room
                        //<CS>RLST:RoomName:room_list<CE>
                        CString *msg = new CString;
                        *msg = CString("<CS>RLST:admin:") + RoomToUsers[Room_name] +
CString("<CE>");
                        PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)msg,NULL);

                        //TWICE SEND THE DATA

      //PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)msg,NULL);

                        CString *msg2 = new CString;
                        *msg2 = CString("<CS>RLST:admin:") + UserName +
CString("<CE>");
                        for(int i=0;i<(int)m_threadIDs[0].size();i++)
                        {
                              if(m_threadIDs[0][i]!=threadID)

      PostThreadMessage(m_threadIDs[0][i],UWM_SEND_DATA,(WPARAM)msg2,NULL);
                        }
                        if(RoomToUsers[Room_name]!="")
                              RoomToUsers[Room_name]+= "," + UserName;
                        else RoomToUsers[Room_name] = UserName;
                  }

                  else
                  {
                        //search the room
                        int RoomIndex;
                        map<CString,int>::iterator iter;
                        iter  = RoomToThreadIDIndex.find(Room_name);
                        if(iter == RoomToThreadIDIndex.end()) //room name not found
                        {
                              if(threadID!=NULL)
                              {
                                    //notify the client that the requested room
name was not found
                                    CString *data = new CString;
                                    *data = "<CS>NSR:" + Room_name  + "<CE>";

      PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)data,NULL);
                              }
                              else //the message was from server itself
                              {
                                    ChatList.AddString("Room Name not found");
                                    //ChatList.PostMessage(WM_LBUTTONDOWN);
                                    //ChatList.PostMessage(WM_LBUTTONUP);
                              }
                        }
                        else
                        {
                              RoomIndex = iter->second;
                              if(threadID!=NULL)
                              {
                                    //else send connected
                                    CString *data = new CString;
```

```
                                        *data = "<CS>CON:" + Room_name  +"<CE>";

      PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)data,NULL);
                                        //and insert the threadID into the m_threadIDs
                                        m_threadIDs[RoomIndex].push_back(threadID);
                                        //also send list of users connected to this
room
                                          //generate a comma seperated list of the
users and send
                                        //to this newly joined user and just this
user's name to
                                        //all the other clients in this room
                                        CString *msg = new CString;
                                        if(RoomToUsers[Room_name]!="")//send the room
users list only if it is not empty
                                        {
                                              //consecutive send calls on the same
socket BEWARE!!!
                                              Sleep(300);
                                              *msg = CString("<CS>RLST:") + Room_name
+ CString(":") +  RoomToUsers[Room_name] + CString("<CE>");

      PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)msg,NULL);
                                        }
                                        CString *msg2 = new CString;
                                        *msg2 = CString("<CS>RLST:") + Room_name +
CString(":") +  UserName + CString("<CE>");
                                        for(int
i=0;i<(int)m_threadIDs[RoomIndex].size();i++)
                                        {
                                              //dont send it to thread that has joined
this room
                                              if(m_threadIDs[RoomIndex][i]!=threadID)

      PostThreadMessage(m_threadIDs[RoomIndex][i],UWM_SEND_DATA,(WPARAM)msg2,NULL);
                                          if(m_threadIDs[RoomIndex][i]==NULL)
                                            {
                                                  CString *msg =  new CString;
                                                  *msg = CString("<CS>RLST:") +
UserName + CString("<CE>");
                                                  //make the tab active
                                                  ((CPropertySheet *)GetParent())-
>SetActivePage(RoomToTab[Room_name]);
                                                  RoomToTab[Room_name]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                                            }
                                        }
                                        if(RoomToUsers[Room_name]!="")
                                              RoomToUsers[Room_name]+= "," + UserName;
                                        else RoomToUsers[Room_name] = UserName;
                                  }
                                  else //the message was from server itself
                                  {
                                        //display a new property page and add the
admin(NULL) to that room's threadID list
                                        m_threadIDs[RoomIndex].push_back(0);
                                        //Create a new property sheet and add a new
page
```

```
                                        PropertyPage *page = new PropertyPage;
                                        page->m_psp.pszTitle = Room_name;
                                        page->m_psp.dwFlags |= PSP_USETITLE;
                                        page->pMainDlg = this;
                                        page->UserName = "admin";
                                        //<CS>RLST:RoomName:room_user_list<CE>
                                        //send the users list to this new created room
tab
                                        CString *msg = new CString;//this will be sent
to the tab in this server
                                        bool RoomEmpty = true;
                                        if(RoomToUsers[Room_name]!="")
                                        {
                                               RoomEmpty = false;
                                               *msg = CString("<CS>RLST:")  +
RoomToUsers[Room_name] + CString("<CE>");
                                        }
                                        //send to all other users
                                        CString *msg2 = new CString;
                                        *msg2 = CString("<CS>RLST:") + Room_name + ":"
+ CString("admin") + CString("<CE>");
                                        for(int
i=0;i<(int)m_threadIDs[RoomIndex].size();i++)

      PostThreadMessage(m_threadIDs[RoomIndex][i],UWM_SEND_DATA,(WPARAM)msg2,NULL);
                                        if(RoomToUsers[Room_name]=="")
                                               RoomToUsers[Room_name] = "admin";
                                        else RoomToUsers[Room_name]+= CString(",") +
CString("admin");

                                        page->RoomName = Room_name;
                                        /*PropSheet.AddPage(page);
                                        PropSheet.DoModal();*/
                                        //page->PostMessage(UWM_SEND_DATA,(WPARAM)msg);
                                        ((CPropertySheet *)GetParent())->AddPage(page);
                                        CPropertyPage* Page = ((CPropertySheet
*)GetParent())->GetPage(((CPropertySheet *)GetParent())->GetPageIndex(page));
                                        ((CPropertySheet *)GetParent())-
>SetActivePage(page);
                                        if(!RoomEmpty)
                                               Page-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                                        //add the room name to page address mapping
                                        RoomToTab[Room_name] = Page;
                                }
                        }
                }
        }
        else if(com.Mid(0,strlen("/Create_Room")) == "/Create_Room")
        {
                //create a new room
                //extract the room name
                CString Room_name;
                Room_name = com.Mid(strlen("/Create_Room") + 1 , com.GetLength());
//              Rooms.Add(Room_name);
                if(!RoomList.GetLength())
                        RoomList = Room_name;
                else RoomList += "," + Room_name;
```

```
                    //add this thread to the newly created room
                    vector<DWORD> room;
                    m_threadIDs.push_back(room);
                    RoomToThreadIDIndex[Room_name] = (int)m_threadIDs.size() - 1;
//last one since the room was just added at the end
                    //so the first threadID in the room is of the room owner

                    //dont create a new property sheet , only display the new created
user in a list
                    //code to display room name in a list
                    AvailableRooms.AddString(Room_name);
                    //also change the target of the thread to this newly created
property page
                    //by sending a message UWM_NEWTARGET(WPARAM target,LPARAM) to the
thread to change its target
                    //and handle the message UWM_NETWORK_DATA in the property dialog
             }
             else if(com.Mid(0,strlen("/RoomTopic")) == "/RoomTopic")
             {
                    //<CS>/RoomTopic:room:topic<CE>
                    //extract the room name
                    CString RoomName("");
                    for(int i=strlen("/RoomTopic") + 1;com[i]!=':';i++)
                           RoomName += com[i];
                    int RoomIndex = RoomToThreadIDIndex[RoomName];
                    //extract the room topic
                    CString* topic = new CString;
                    *topic = com.Mid(com.ReverseFind(':') + 1 , com.GetLength());

                    if(RoomName == "admin")
                    {
                           //then display the new topic in the admin room
                           CString *OwnStr = new CString;
                           *OwnStr = "New topic is " + *topic;
                           ChatList.AddString(*OwnStr);
                           //ChatList.PostMessage(WM_LBUTTONDOWN);
                           //ChatList.PostMessage(WM_LBUTTONUP);
                    }

                    //send the topic to all the threads in this room
                    //<DS>TPIC:ROOM:TOPIC<DE>
                    *topic = "<CS>TPC:" + RoomName + ":" + *topic + "<CE>";
                    for(int i=0;i<(int)m_threadIDs[RoomIndex].size();i++)
                    {
                           if(m_threadIDs[RoomIndex][i]==NULL) //admin is also present
in that room
                           {
                                  if(RoomName!="admin")
                                  //send the message to the tab
                                  RoomToTab[RoomName]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)topic);
                           }
                           else //send to the other clients in the room

      PostThreadMessage(m_threadIDs[RoomIndex][i],UWM_SEND_DATA,(WPARAM)topic,NULL);
                    }
             }
```

```
		// /Leave_Room cannot be given in the admin room
		//<CS>/Leave_Room:Room<CE>
		else if(com.Mid(0,strlen("/Leave_Room")) == "/Leave_Room")
		{
			//get the user name from UserToThreadID
			CString User_Name("");
			map<CString,DWORD>::iterator iter;
			for(iter=UserToThreadID.begin();iter!=UserToThreadID.end();iter++)
			{
				if(iter->second == threadID)
				{
					User_Name = iter->first;
					break;
				}
			}
			if(threadID==NULL) //admin is leaving a room
				User_Name= "admin";
			//extract the room name
			CString RoomName("");
			for(int i=com.Find(':') + 1;i<com.GetLength();i++)
				RoomName += com[i];
			int RoomIndex = RoomToThreadIDIndex[RoomName];
			if(RoomIndex == 0)
			{
				//since a user cannot leave the admin room
				CString *ErrorMsg = new CString;
				*ErrorMsg = "<DS>admin:Sorry you cannot leave admin
room<DE>";

	PostThreadMessage(threadID,UWM_SEND_DATA,(WPARAM)ErrorMsg,NULL);
			}
			else
			{
				//remove the user from the room list
				for(int i=0;i<(int)m_threadIDs[RoomIndex].size();i++)
				{
					if(m_threadIDs[RoomIndex][i] != threadID)
					{//send USRLFT message to all other users
						CString* msg = new CString;
						*msg = "<CS>USRLFT:" + RoomName + ":" +
User_Name + "<CE>";

	PostThreadMessage(m_threadIDs[RoomIndex][i],UWM_SEND_DATA,(WPARAM)msg,NULL);
					}
				}
				for(int i=0;i<(int)m_threadIDs[RoomIndex].size();i++)
				{
					if(m_threadIDs[RoomIndex][i] == threadID)//remove
this requesting user

	m_threadIDs[RoomIndex].erase(m_threadIDs[RoomIndex].begin() + i);
				}
				//also remove from the user from RoomToUserList and
				//send a USRLFT:Roomname:username message to all the clients
in that room and
				//check if this room tab exists in the server than send the
message to that tab only
				//check in RoomToTab
```

```
                        if(User_Name!="admin")
                        {
                                if(RoomToTab.find(RoomName)!=RoomToTab.end())
                                {
                                        CString* msg = new CString;
                                        *msg = "<CS>USRLFT:" + RoomName + ":" +
User_Name + "<CE>";
                                        RoomToTab[RoomName]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                                }
                                if(UserToTab.find(User_Name)!=UserToTab.end())
                                {
                                        CString* msg = new CString;
                                        *msg = "<CS>USRLFT:" + RoomName + ":" +
User_Name + "<CE>";
                                        UserToTab[User_Name]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg);
                                }
                        }
                        else //if admin is leaving the room then remove the mapping
from RoomToTab or UserToTab
                        {
                                map<CString,CPropertyPage*>::iterator iter;
                                iter = RoomToTab.find(RoomName);
                                if(iter!=RoomToTab.end()) RoomToTab.erase(iter);
                        }
                        map<CString,CString>::iterator iter2;
                        for(iter2 =
RoomToUsers.begin();iter2!=RoomToUsers.end();iter2++)
                        {
                                if(iter2->first == RoomName || RoomName == "admin")
                                {
                                        //may be the user name is first in the room
                                        if(iter2->second.Replace(","+User_Name,"")==0)
                                        {
                                                //then try this(assuming the user is
first in this room)
                                                if(iter2->second.Replace(User_Name +
",","")==0)
                                                        //if even this fails then the user
might be alone in the room
                                                        iter2-
>second.Replace(User_Name,"");
                                        }
                                }
                        }

                }
        }
        //private message init(this message is sent before starting the private
coinversation)
        else if(com.Mid(0,strlen("/msg")) == "/msg")
        {
                //get the username of the requesting threadID
                CString RequestingUser;
                map<CString,DWORD>::iterator iter = UserToThreadID.begin();
                for(;iter!=UserToThreadID.end();iter++)
                {
```

```
                        if(threadID == iter->second)
                        {
                                RequestingUser = iter->first;
                                break; //found the user corresponding to the threadID
                        }
                }

                //get the username of the reuqested user
                CString RequestedUser;
                RequestedUser = com.Mid(strlen("/msg") + 1,com.GetLength());

                //check if the message is for the admin
                if(RequestedUser == "admin")
                {
                        //create a new tab
                        PropertyPage *page = new PropertyPage;
                        page->m_psp.pszTitle = RequestingUser;
                        page->m_psp.dwFlags |=PSP_USETITLE;
                        page->pMainDlg = this;
                        page->PrivateMsg = true;
                        page->UserName = RequestingUser;
                        ((CPropertySheet *)GetParent())->AddPage(page);
                        ((CPropertySheet *)GetParent())->SetActivePage(page);
                        //create a mapping of the requesting user to the property
page address
                        UserToTab[RequestingUser] = page;
                }
                else
                {
                        //create a mapping(what is the need to create a mapping?? ,
since everytime a user sents a private message to the
                        //other user he will do so by including the name of the
other user in the private message)
                        //send the message to the client in the form /msg:requesting
user

                        //check if the message is from the admin itself then open a
new tab
                        CString *msg = new CString;
                        if(threadID==NULL)
                        {
                                PropertyPage *page = new PropertyPage;
                                page->m_psp.pszTitle = RequestedUser;
                                page->m_psp.dwFlags |=PSP_USETITLE;
                                page->pMainDlg = this;
                                page->PrivateMsg = true;
                                page->UserName = RequestedUser;
                                ((CPropertySheet *)GetParent())->AddPage(page);
                                ((CPropertySheet *)GetParent())->SetActivePage(page);
                                //RequestedUser will be used in this case since admin
has requested to connect to a user
                                //and so admin will need to keep a mapping of that
user to the tab
                                UserToTab[RequestedUser] = page;
                                *msg = "<CS>msg:admin<CE>";
                        }
                        else *msg = "<CS>msg:" + RequestingUser + "<CE>";
```

```
	PostThreadMessage(UserToThreadID[RequestedUser],UWM_SEND_DATA,(WPARAM)msg,NULL);

			}
		}
		//private message from a user to the other user
		else if(com.Mid(0,strlen("PMSG")) == "PMSG")
		{
			//send the private message to the other user
			//get the username of the requesting threadID
			//<CS>PMSG:user:private message<CE>
			CString RequestingUser;
			map<CString,DWORD>::iterator iter = UserToThreadID.begin();

			for(;iter!=UserToThreadID.end();iter++)
			{
				if(threadID == iter->second)
				{
					RequestingUser = iter->first;
					break; //found the user corresponding to the threadID
				}
			}
			if(threadID == NULL) RequestingUser = "admin";
			//get the username of the requested user
			CString RequestedUser;
			int loc = com.Find(':') + 1;
			for(;com[loc]!=':';loc++)
				RequestedUser+=com[loc];
			if(RequestedUser == "admin") //send the message to property sheet
			{
				//extract the message from <CS>PMSG:From user:message<CE>
				//int loc = com.ReverseFind(':');
				//send the message to the page corresponding to the property
page
				//in the format <DS>User:Message<DE>
				CString *msg = new CString;
				*msg = "<DS>"+ RequestingUser + ":" + com.Mid(loc +
1,com.GetLength() - 4)+"<DE>";
				((CPropertySheet *)GetParent())-
>SetActivePage(UserToTab[RequestingUser]);
				UserToTab[RequestingUser]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg,NULL);
			}
			else
			{
				//send the message to RequestedUser
<CS>PMSG:FROMUSER:private message<CE>
				//client will be responsible for displaying the message in
the correct property
				//page according to the mapping.
				CString *msg = new CString;
				*msg = "<CS>PMSG:" + RequestingUser + ":" + com.Mid(loc +
1,com.GetLength())+"<CE>";

	PostThreadMessage(UserToThreadID[RequestedUser],UWM_SEND_DATA,(WPARAM)msg,NULL);
			}
		}
		}
	if(Data->GetAt(1)=='D')
```

```
	{
		//MessageBox("message data recieved");
		//extract the chat message
		//then if it was for the admin room then display in the admin room
		//else send the message to the room's tab
		//server also needs to check if the admin is in a particular room or not
		//a client will not need to check this since the server will send the
message to the client thread
		//if that particular client is there in the room

		//extract the room name and user name
		//<DS>ROOM:User:Message<DE>
		int i;
		CString RoomName(""),UserName(""),Message("");
		for(i=4;Data->GetAt(i)!=':';i++)
			RoomName += Data->GetAt(i);
		//extract the user name
		for(++i;Data->GetAt(i)!=':';i++)
			UserName += Data->GetAt(i);
		for(++i;Data->GetAt(i)!='<';i++)
			Message += Data->GetAt(i);

		//send the message to all the users in that room
		//if admin is in that room even then the message will be sent to everyone
		//so the user sending the message also recieves it
		CString* msg = new CString;
		*msg = "<DS>" + UserName + ":" + Message + "<DE>";
		int RoomIndex = RoomToThreadIDIndex[RoomName];
		for(int i=0;i<(int)m_threadIDs[RoomIndex].size();i++)
		{
			if(m_threadIDs[RoomIndex][i]!=NULL) //admin is not in the room
			{
				//and also the msg should not be sent to the sender itself
				if(m_threadIDs[RoomIndex][i]!=threadID)

	PostThreadMessage(m_threadIDs[RoomIndex][i],UWM_SEND_DATA,(WPARAM)Data,NULL);
			}
			else
			{
			//admin is there in the room
			//send the message to the room's tab in the server
				//but check if the message is for admin room itself
				if(RoomName=="admin")
				{
					//check if admin himself have not sent it
					if(UserName!="admin")
					{
						//filter out the room name
						CString msg;
						for(int i=strlen("<DS>admin:");Data-
>GetAt(i)!='<';i++)
							msg+=Data->GetAt(i);

						CString *OwnStr = new CString;
						*OwnStr = msg;
						ChatList.AddString(*OwnStr);
//						ChatList.PostMessage(WM_LBUTTONDOWN);
//						ChatList.PostMessage(WM_LBUTTONUP);
```

```
                                }
                            }
                        else
                            {
                                ((CPropertySheet *)GetParent())-
>SetActivePage(RoomToTab[RoomName]);
                                RoomToTab[RoomName]-
>PostMessage(UWM_NETWORK_DATA,(WPARAM)msg,NULL);
                            }
                    }
            }
        }
        return 0;
}
void CChatServerDlg::OnBnClickedSend()
{
        //commands given at the server will cause the message to be sent with NULL as
threadID
        UpdateData(TRUE);
        //properly package the data and commands and then send
        CString *data = new CString;
        /*
        <CS>/Room_List<CE>
(User command: /Room_List)
<CS>// /Join_Room:user:RoomName<CE>
(User Command: /Join_Room room)
<CS>/Create_Room room<CE>
(User Command: /Create_Room room)
<CS>/RoomTopic:room:topic<CE>
(User Command: /RoomTopic topic)
<CS>/msg User<CE>
(User Command: /msg user)
<CS>/Leave_Room:Room<CE>
(User Command: /Leave_Room room)
Chat Messages
<DS>ROOM:User:Message<DE>
(User Command: string starting without forward slash)*/
        TextBox.Trim();
        if(TextBox[0]=='/')
        {
                if(TextBox.Mid(0,strlen("/Room_List")) == "/Room_List")
                        *data = CString("<CS>") + TextBox + CString("<CE>");
                else if(TextBox.Mid(0,strlen("/Join_Room")) == "/Join_Room")
                {
                        //extract room name
                        CString RoomName;
                        for(int i=strlen("/Join_Room") + 1;i<TextBox.GetLength();i++)
                                RoomName+=TextBox[i];
                        //build the command string,admin since this is the admin room
                        //// /Join_Room:user:RoomName
                        *data = CString("<CS>/Join_Room:admin:") + RoomName +
CString("<CE>");
                }
                else if(TextBox.Mid(0,strlen("/Create_Room")) == "/Create_Room")
                        *data = CString("<CS>") + TextBox + CString("<CE>");
                else if(TextBox.Mid(0,strlen("/RoomTopic")) == "/RoomTopic")
                {
                        //extract topic
```

```
                    CString topic;
                    for(int i=strlen("/RoomTopic") + 1;i<TextBox.GetLength();i++)
                          topic+=TextBox[i];
                    //package as <CS>/RoomTopic:room:topic<CE>
                    *data = CString("<CS>/RoomTopic:admin:") + topic +
CString("<CE>");
             }
             else if(TextBox.Mid(0,strlen("/msg")) == "/msg")
             {
                    //extract the user name
                    CString UserName;
                    for(int i=strlen("/msg") + 1;i<TextBox.GetLength();i++)
                          UserName+=TextBox[i];
                    *data = CString("<CS>/msg ") + UserName + CString("<CE>");
             }
             else if(TextBox.Mid(0,strlen("/Leave_Room")) == "/Leave_Room")
             {
                    //extract the user name
                    CString RoomName;
                    for(int i=strlen("/Leave_Room") + 1;i<TextBox.GetLength();i++)
                          RoomName+=TextBox[i];
                    *data = CString("<CS>/Leave_Room:") + RoomName + CString("<CE>");
             }
             }
             else //this was a general user message
                    //<DS>ROOM:User:Message<DE>
             {
                    //display in own window
                    CString ColorString;
                    if(ColorSelected)
ColorString.Format("(c:%d,%d,%d)",GetRValue(color),GetGValue(color),GetBValue(color));
                    CString *OwnStr = new CString;
                    *OwnStr = CString("admin: ")+ ColorString + TextBox;
                    ChatList.AddString(*OwnStr);
                    Sleep(100);
                    //ChatList.PostMessage(WM_LBUTTONDOWN);
                    //ChatList.PostMessage(WM_LBUTTONUP);
                    *data = CString("<DS>")  + "admin:" + "admin: " + ColorString +
TextBox + "<DE>";
             }
       if(PostMessage(UWM_NETWORK_DATA,(WPARAM)data,NULL)){}
       else MessageBox("PostMessage failed");
       TextBox = "";
       UpdateData(FALSE);
}

 LRESULT CChatServerDlg::OnThreadClose(WPARAM ,LPARAM ThreadID)
 {
        DWORD threadID = (DWORD)ThreadID;
        //search the name of the user in UserToThreadID
        CString UserName;
        CString RequestingUser;
        map<CString,DWORD>::iterator ite = UserToThreadID.begin();
        for(;ite!=UserToThreadID.end();ite++)
        {
             if(threadID == ite->second)
             {
                    UserName = ite->first;
```

```
                    break; //found the user corresponding to the threadID
               }
          }
          if(UserName!="")
          {
          //remove the user from the list box
          int loc = ConnectedUsers.FindStringExact(0,UserName);
          ConnectedUsers.DeleteString(loc);
          //remove the threadID from all the rooms and also from the user list
          //send the message to all the room tabs they will remove if that user exists in
them

          //<CS>USRLFT:admin:UserName<CE>
          //since ThreadClose() means the user has left the admin room itself
          //send the msg to all the users
           for(int i=0;i<(int)m_threadIDs.size();i++)
           {
                 for(int j=0;j<(int)m_threadIDs[i].size();j++)
                 {
                             CString *msg = new CString;
                             *msg = "<CS>USRLFT:admin:" + UserName + "<CE>";

        PostThreadMessage(m_threadIDs[i][j],UWM_SEND_DATA,(WPARAM)msg,NULL);
                 }
           }

           //send the messge to tabs
           map<CString,CPropertyPage*>::iterator iter;
           for(iter = RoomToTab.begin();iter!=RoomToTab.end();iter++)
           {
                  CString *msg = new CString;
                  *msg = "<CS>USRLFT:admin:" + UserName + "<CE>";
                  iter->second->PostMessage(UWM_NETWORK_DATA,(WPARAM)msg,NULL);
           }
           map<CString,CPropertyPage*>::iterator itere;
           for(itere = UserToTab.begin();itere!=UserToTab.end();itere++)
           {
                  CString *msg = new CString;
                  *msg = "<CS>USRLFT:admin:" + UserName + "<CE>";
                  itere->second->PostMessage(UWM_NETWORK_DATA,(WPARAM)msg,NULL);
           }
          for(int i=0;i<(int)m_threadIDs.size();i++)
          {
                 for(int j=0;j<(int)m_threadIDs[i].size();j++)
                 {
                        if(m_threadIDs[i][j] == (DWORD) threadID)
                              m_threadIDs[i].erase(m_threadIDs[i].begin() + j);
                 }
          }
          //also remove this user from the RoomToUsers
          map<CString,CString>::iterator iter2;
          for(iter2 = RoomToUsers.begin();iter2!=RoomToUsers.end();iter2++)
          {
                 //may be the user name is first in the room
                 if(iter2->second.Replace(","+UserName,"")==0)
                 {
                             //then try this(assuming the user is first in this room)
                             if(iter2->second.Replace(UserName + ",","")==0)
```

```
                                        //if even this fails then the user might be alone in
the room
                                                iter2->second.Replace(UserName,"");
                }

        }
        //remove from UserToTab if present
         map<CString,CPropertyPage*>::iterator iter3;
         for(iter3 = UserToTab.begin();iter3!=UserToTab.end();iter3++)
         {
                if(iter3->first == UserName)
                {
                        UserToTab.erase(iter3);
                        break;
                }
        }
        }
        return 0;
 }
 void CChatServerDlg::OnBnClickedColor()
 {
        CColorDialog cdlg;
        if(cdlg.DoModal()==IDOK)
        {
                ColorSelected = true;
                color = cdlg.GetColor();
        }
        else ColorSelected = false;
 }
BOOL CChatServerDlg::PreTranslateMessage(MSG* pMsg)
{
       if (pMsg->message == WM_KEYDOWN && pMsg->wParam == VK_RETURN)
       {
              OnBnClickedSend();
              //EditBox.SendMessage(WM_LBUTTONDOWN);
              //EditBox.SendMessage(WM_LBUTTONUP);
       }
       return CPropertyPage::PreTranslateMessage(pMsg);
}
```

CustListBox.h

```
#pragma once

#include<vector>
using namespace std;
// CustListBox

class CustListBox : public CListBox
{
       DECLARE_DYNAMIC(CustListBox)

public:
       CustListBox();
       virtual ~CustListBox();
       virtual int CompareItem(LPCOMPAREITEMSTRUCT lpCompareItemStruct);
```

```
	virtual void MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct);
	virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
	void TextOut(int x,int y,CDC* dc,CString txt);

protected:
	DECLARE_MESSAGE_MAP()
};
```

CustListBox.cpp

```
// ustListBox.cpp : implementation file
//

#include "stdafx.h"
#include "CustListBox.h"
#include "resource.h"

// CustListBox

IMPLEMENT_DYNAMIC(CustListBox, CListBox)

CustListBox::CustListBox()
{

}

CustListBox::~CustListBox()
{
}

int CustListBox::CompareItem(LPCOMPAREITEMSTRUCT lpCompareItemStruct){return 0;}
void CustListBox::MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
{
	//lpMeasureItemStruct->itemHeight = 15;
}
void CustListBox::TextOut(int x,int y,CDC* dc,CString txt)
{
	CString str("");
	//display the string part anf then if a image occurs then display it
	/*
	color formatted string
	(c:R,G,B) string
	*/
	bool PenSel = false;
	int i = txt.Find(':') + 2,ind;
	TRACE("\ntxt = %s i = %d",txt,i);
	//due to the format being user: message
	//select the pen according to the color
	if(txt.GetLength()>3&&(txt[i]=='(' && txt[i+1]=='c' && txt[i+2]==':'))
	{
		ind = i;
		//extract the R,G,B values
		BYTE R = 0,G = 0,B = 0;

		for(i+=3;isdigit(txt[i]);i++)
```

```
                R=R*10+(txt[i]-'0');
            for(++i;isdigit(txt[i]);i++)
                G=G*10+(txt[i]-'0');
            for(++i;isdigit(txt[i]);i++)
                B=B*10+(txt[i]-'0');
            //TRACE("\nR=%d  G=%d  B=%d",R,G,B);
            dc->SetTextColor(RGB(R,G,B));
            PenSel=true;
            //now erase color command from the string
            txt.Delete(ind,(i+1)-ind);
      }
      int indent = x;
      for(i=0;i<txt.GetLength();i++)
      {
            //check for emoticons
                if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='P'))
                {
                      dc->TextOut(indent,y,str,str.GetLength());
                      indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                      HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_TONGUE_SMILE),IMAGE_BITMAP,15,
                      dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                      indent+=15;
                      str="";
                      i++;
                }
                else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]==')'))
                {
                      dc->TextOut(indent,y,str,str.GetLength());
                      indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                      HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_SMILE),IMAGE_BITMAP,15,15,LR_D
                      dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                      indent+=15;
                      str="";
                      i++;
                }
                else if(i+1<txt.GetLength()&&(txt[i]==':'&&(txt[i+1]=='D'||txt[i+1]=='d'
                {
                      dc->TextOut(indent,y,str,str.GetLength());
                      indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                      HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_GRIN),IMAGE_BITMAP,15,15,LR_DE
                      dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                      indent+=15;
                      str="";
                      i++;
                }
                else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='|'))
                {
                      dc->TextOut(indent,y,str,str.GetLength());
                      indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                      HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_STRAIGHT_FACE),IMAGE_BITMAP,15
                      dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                      indent+=15;
                      str="";
                      i++;
```

```
                        }
                        else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='('))
                        {
                                dc->TextOut(indent,y,str,str.GetLength());
                                indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                                HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_SAD),IMAGE_BITMAP,15,15,LR_DEF
                                dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                                str="";
                                indent+=15;
                                i++;
                        }
                        else if(i+1<txt.GetLength()&&(txt[i]==';'&&txt[i+1]==')'))
                        {
                                dc->TextOut(indent,y,str,str.GetLength());
                                indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                                HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_WINK),IMAGE_BITMAP,15,15,LR_DE
                                dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                                str="";
                                indent+=15;
                                i++;
                        }
                        else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='o'))
                        {
                                dc->TextOut(indent,y,str,str.GetLength());
                                indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                                HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_SURPRISE),IMAGE_BITMAP,15,15,L
                                dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                                str="";
                                indent+=15;
                                i++;
                        }
                        else if(i+1<txt.GetLength()&&(txt[i]==':'&&txt[i+1]=='s'))
                        {
                                dc->TextOut(indent,y,str,str.GetLength());
                                indent+=dc->GetTextExtent(str,str.GetLength()).cx;
                                HBITMAP hBmp =
(HBITMAP)LoadImage(AfxGetResourceHandle(),MAKEINTRESOURCE(IDB_WORRIED),IMAGE_BITMAP,15,15,LR
                                dc->DrawState(CPoint(indent,y),CSize(15,15),hBmp,DST_BITMAP);
                                str="";
                                indent+=15;
                                i++;
                        }
                        else str+=txt[i];
        }
        if(str!="")
                dc->TextOut(indent,y,str,str.GetLength());
        //restore the black color
        dc->SetTextColor(RGB(0,0,0));
}
void CustListBox::DrawItem(LPDRAWITEMSTRUCT lp)
{
        CDC *dc = CDC::FromHandle(lp->hDC);
        //ASSERT(lp->itemData!=NULL);
        CString str = (char *)lp->itemData;
        TRACE("\nDRAWITEM STR=%s",str);
```

```
	TextOut(lp->rcItem.left,lp->rcItem.top,dc,str);
}


BEGIN_MESSAGE_MAP(CustListBox, CListBox)
END_MESSAGE_MAP()



// CustListBox message handlers
```

ListenSocket.h

```
#pragma once

// CListenSocket command target

class CListenSocket : public CAsyncSocket
{
protected:
	CWnd* target;
public:
	CListenSocket();
	void SetTarget(CWnd * w) { target = w; }
	virtual void OnAccept(int nErrorCode);
	virtual ~CListenSocket();
};
```

ListenSocket.cpp

```
// ListenSocket.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Server.h"
#include "ListenSocket.h"
#include "messages.h"
#include "ServerThread.h"
// CListenSocket

CListenSocket::CListenSocket()
{
}

CListenSocket::~CListenSocket()
{
}

void CListenSocket::OnAccept(int nErrorCode)
```

```
{
      TRACE(_T("%s: CListensoc::OnAccept(%d)\n"), AfxGetApp()-
>m_pszAppName, nErrorCode);
    if(nErrorCode != ERROR_SUCCESS)
       { /* had an error */
        ASSERT(FALSE);
        ASSERT(target != NULL);
        if(target != NULL)
           target->PostMessage(UWM_NETWORK_ERROR, (WPARAM)nErrorCode,
(LPARAM)::GetCurrentThreadId());
        return;
       } /* had an error */
    // New connection is being established

    CAsyncSocket soc;

    // Accept the connection using a temp CAsyncSocket object.
    if(!Accept(soc))
       { /* accept failed */
        DWORD err = ::GetLastError();
        ASSERT(target != NULL);
        if(target != NULL)
           target->PostMessage(UWM_NETWORK_ERROR, (WPARAM)err,
(LPARAM)::GetCurrentThreadId());
        return;
       } /* accept failed */

    // Create a thread to handle the connection. The thread is created
suspended so that we can
    // set variables in CServerThread before it starts executing.
    CServerThread* pThread =
(CServerThread*)AfxBeginThread(RUNTIME_CLASS(CServerThread),

THREAD_PRIORITY_NORMAL,
                                                          0,

CREATE_SUSPENDED);
    if (pThread == NULL)
        {
         DWORD err = ::GetLastError();
         ASSERT(FALSE);  // help in debugging
         soc.Close();
         TRACE(_T("%s: CListenSoc::OnAccept: Could not create thread:
%d\n"), AfxGetApp()->m_pszAppName, err);
         ASSERT(target != NULL);
         if(target != NULL)
            target->PostMessage(UWM_NETWORK_ERROR, (WPARAM)err,
(LPARAM)::GetCurrentThreadId());
         return;
        }

    // Pass the socket to the thread by passing the socket handle. You
cannot pass
    // a CSocket object across threads.
    pThread->SetSocket(soc.Detach());
    pThread->SetTarget(target);
    // Now start the thread.
    pThread->ResumeThread();
```

```
    CAsyncSocket::OnAccept(nErrorCode);
}

// CListenSocket member functions
```

PropertyPage.h

```
#pragma once
#include "Chat ServerDlg.h"
#include "afxwin.h"
#include "CustListBox.h"
// PropertyPage dialog

class PropertyPage : public CPropertyPage
{
	DECLARE_DYNAMIC(PropertyPage)

public:
	bool ColorSelected;
	bool leave;
	bool PrivateMsg;  //true in case of private chat tab , false in case
of room tab
	CString UserName; //self name
	CString RoomName; //in case of Room chat
	DWORD m_threadID;
	COLORREF color;
	PropertyPage();
	virtual ~PropertyPage();

// Dialog Data
	enum { IDD = IDD_PROPERTYPAGE };

protected:
	BOOL OnInitDialog();
	LRESULT OnNetworkData(WPARAM,LPARAM);
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
public:
	int RoomIndex;
	CChatServerDlg* pMainDlg;

	DECLARE_MESSAGE_MAP()
	CString TextBox;
	CListBox ConnectedUsers;
	CustListBox ChatList;
	afx_msg void OnBnClickedSend();
	afx_msg void OnBnClickedColor();
	BOOL PreTranslateMessage(MSG* pMsg);
};
```

PropertyPage.cpp

```
// PropertyPage.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Server.h"
#include "PropertyPage.h"
#include "messages.h"

// PropertyPage dialog

IMPLEMENT_DYNAMIC(PropertyPage, CPropertyPage)

PropertyPage::PropertyPage()
	: CPropertyPage(PropertyPage::IDD)
{
	PrivateMsg = false;
	leave = false;
	ColorSelected = false;
}

PropertyPage::~PropertyPage()
{
}
BOOL PropertyPage::OnInitDialog()
{
	CPropertyPage::OnInitDialog();
	ConnectedUsers.AddString("admin");
	if(PrivateMsg)
		ConnectedUsers.AddString(UserName);
	return TRUE;
}
void PropertyPage::DoDataExchange(CDataExchange* pDX)
{
	CPropertyPage::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_TEXTBOX, TextBox);
	DDX_Control(pDX, IDC_CONNECTED, ConnectedUsers);
	DDX_Control(pDX, IDC_CHAT_LIST, ChatList);
}
LRESULT PropertyPage::OnNetworkData(WPARAM data,LPARAM ThreadID)
{
	/*
	Following message will be recieved here
	<CS>PMSG:user:private message<CE>
	<CS>TPIC:ROOM:TOPIC<CE>
	<DS>ROOM:User:Message<DE>
	*/

	//check for commands
	CString *Data = (CString *)data;
	if(Data->GetAt(1) == 'C')
	{
		//check for topic
		if(Data->Mid(4,strlen("TPIC")) == "TPIC")
		{
			//extract the topic and display
			CString Topic("");
			for(int i = Data->ReverseFind(':') + 1;Data->GetAt(i)!=':';i++)
```

```
                        Topic += Data->GetAt(i);
                    //display the string
                    CString *OwnStr = new CString;
                    *OwnStr = Topic;
                    ChatList.AddString(*OwnStr);
                    //ChatList.PostMessage(WM_LBUTTONDOWN);
                    //ChatList.PostMessage(WM_LBUTTONUP);
                }

                //check for Message
                else if(Data->Mid(4,strlen("PMSG")) == "PMSG")
                {
                        //extract the message
                        CString Message("");
                        for(int i=strlen("<CS>PMSG:");Data->GetAt(i)!='<';i++)
                                Message += Data->GetAt(i);
                        //display the string
                        CString *OwnStr = new CString;
                        *OwnStr = Message;
                        ChatList.AddString(*OwnStr);
//                      ChatList.PostMessage(WM_LBUTTONDOWN);
//                      ChatList.PostMessage(WM_LBUTTONUP);
                }
                //check for connected user's list
                //<CS>RLST:connected user list(comma seperated)<CE>
                else if(Data->Mid(4,strlen("RLST")) == "RLST")
                {
                        CString List("");
                        for(int i = strlen("<CS>RLST:");Data->GetAt(i)!='<';i++)
                                List += Data->GetAt(i);
                        //display the users in the connected user's list
                        CString User("");
                        for(int i = 0;i<List.GetLength();i++)
                        {
                                if(List[i]==',')
                                {
                                        ConnectedUsers.AddString(User);
                                        User="";
                                }
                                else User+=List[i];
                        }
                        ConnectedUsers.AddString(User);
                }
                //<CS>USRLFT:RoomName:UserName<CE>
                //user has left the room
                else if(Data->Mid(4,strlen("USRLFT")) == "USRLFT")
                {
                        //get the user name and room name
                        CString UserName(""),Room_Name("");
                        int loc = Data->Find(':') + 1;
                        for(;Data->GetAt(loc)!=':';loc++)
                                Room_Name+=Data->GetAt(loc);
                        for(++loc;Data->GetAt(loc)!='<';loc++)
                                UserName+= Data->GetAt(loc);
                        //remove the user from this room only if the user has left the
admin room or this room
                        if(Room_Name=="admin" || Room_Name==RoomName)
                        {
```

```
                        loc = ConnectedUsers.FindStringExact(0,UserName);
                        ConnectedUsers.DeleteString(loc);
                    }
                }
        }
        else
        {
                CString Message("");
                //extract message recieved for this user
                for(int i = Data->Find('>') + 1;Data->GetAt(i)!='<';i++)
                        Message += Data->GetAt(i);
                //display the message
                CString *OwnStr = new CString;
                *OwnStr = Message;
                ChatList.AddString(*OwnStr);
                //ChatList.PostMessage(WM_LBUTTONDOWN);
                //ChatList.PostMessage(WM_LBUTTONUP);
        }
        return 0;
}

BEGIN_MESSAGE_MAP(PropertyPage, CPropertyPage)
        ON_MESSAGE(UWM_NETWORK_DATA, OnNetworkData)
        ON_BN_CLICKED(IDC_SEND, &PropertyPage::OnBnClickedSend)
        ON_BN_CLICKED(IDC_COLOR, &PropertyPage::OnBnClickedColor)
END_MESSAGE_MAP()


// PropertyPage message handlers

void PropertyPage::OnBnClickedSend()
{
        UpdateData(TRUE);
        //properly package the data and send it to the main dlg (or server in case of
client)
        //check for command data
        TextBox.Trim();
        CString* data = new CString;
        if(TextBox[0]=='/')
        {
                if(TextBox.Mid(0,strlen("/Room_List")) == "/Room_List")
                        *data = CString("<CS>") + TextBox + CString("<CE>");
                else if(TextBox.Mid(0,strlen("/Join_Room")) == "/Join_Room")
                {
                        //extract room name
                        CString RoomName;
                        for(int i=strlen("/Join_Room") + 1;i<TextBox.GetLength();i++)
                                RoomName+=TextBox[i];
                        //build the command string,admin since this is the admin room
                        *data = CString("<CS>/Join_Room:admin:") + RoomName +
CString("<CE>");
                }
                else if(TextBox.Mid(0,strlen("/Create_Room")) == "/Create_Room")
                        *data = CString("<CS>") + TextBox + CString("<CE>");
                else if(TextBox.Mid(0,strlen("/RoomTopic")) == "/RoomTopic")
                {
                        //extract topic
                        CString topic;
```

```
                    for(int i=strlen("/RoomTopic") + 2;i<TextBox.GetLength();i++)
                          topic+=TextBox[i];
                    //package as <CS>/RoomTopic:room:topic<CE>
                    *data = CString("<CS>/RoomTopic:") + RoomName + ":" + topic +
CString("<CE>");
             }
             else if(TextBox.Mid(0,strlen("/msg")) == "/msg")
             {
                    //extract the user name
                    CString UsrName;
                    for(int i=strlen("/msg") + 1;i<TextBox.GetLength();i++)
                          UsrName+=TextBox[i];
                    *data = CString("<CS>/msg ") + UsrName + CString("<CE>");
             }
             else if(TextBox.Mid(0,strlen("/Leave_Room")) == "/Leave_Room")
             {
                    //extract the user name
                    /*CString RoomName;
                    for(int i=strlen("/Leave_Room") + 1;i<TextBox.GetLength();i++)
                          RoomName+=TextBox[i];*/
                    if(!PrivateMsg)
                          *data = CString("<CS>/Leave_Room:") + RoomName +
CString("<CE>");
                    //close this page
                    leave = true;
             }
      }
             else //this was a general user message
                    //<DS>ROOM:User:Message<DE>
             {
                    CString ColorString;
                    if(ColorSelected)
ColorString.Format("(c:%d,%d,%d)",GetRValue(color),GetGValue(color),GetBValue(color));
                    if(PrivateMsg)
                    {
                          *data = CString("<CS>PMSG:") + UserName + ": " +
ColorString + TextBox + CString("<CE>");
                          CString *OwnStr = new CString;
                          *OwnStr = "admin: " + ColorString + TextBox;
                          ChatList.AddString(*OwnStr);
                          //ChatList.PostMessage(WM_LBUTTONDOWN);
                          //ChatList.PostMessage(WM_LBUTTONUP);
                    }
                    else *data = CString("<DS>") + RoomName + ":" + UserName +": "+
ColorString + TextBox + "<DE>";
             //     ChatList.AddString("admin:" + TextBox);
          }
      if(pMainDlg->PostMessage(UWM_NETWORK_DATA,(WPARAM)data,NULL)){}
      else MessageBox("PostMessage failed");
      TextBox="";
      UpdateData(FALSE);
      if(leave)
             ((CPropertySheet *)GetParent())->RemovePage(this);
}


void PropertyPage::OnBnClickedColor()
{
```

```
        CColorDialog cdlg;
        if(cdlg.DoModal()==IDOK)
        {
               ColorSelected = true;
               color = cdlg.GetColor();
        }
        else ColorSelected = false;
}
BOOL PropertyPage::PreTranslateMessage(MSG* pMsg)
{
       if (pMsg->message == WM_KEYDOWN && pMsg->wParam == VK_RETURN)
              OnBnClickedSend();
       return CPropertyPage::PreTranslateMessage(pMsg);
}
```

ServerSocket.h

```
#pragma once

// CServerSocket command target
//check_data() return values
#define DATA_IN_CONTINUATION 1
#define MESSAGE_DATA 2
#define COMMAND_DATA 3
#define DATA_START 4
#define DATA_END 5

class CServerSocket : public CAsyncSocket
{
protected:
       CWnd* target;
       CString Data;
public:
       DWORD threadID;
       CServerSocket();
       void SetTarget(CWnd * w) { target = w; }
       virtual void OnClose(int nErrorCode);
    virtual void OnReceive(int nErrorCode);
       int check_data(CString str);
       virtual ~CServerSocket();
};
```

ServerSocket.cpp

```
// ServerSocket.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Server.h"
#include "ServerSocket.h"
#include "messages.h"

// CServerSocket
```

```
CServerSocket::CServerSocket():Data("")
{
//     state = DATA_START;
}

CServerSocket::~CServerSocket()
{
}

void CServerSocket::OnClose(int nErrorCode)
{
       //close the thread which is handling this socket
       PostThreadMessage(threadID,UWM_THREADCLOSE,NULL,NULL);
       CAsyncSocket::OnClose(nErrorCode);
}
int CServerSocket::check_data(CString data)
{
       //RETURNS:
       /*
          DATA_IN_CONTINUATION
          MESSAGE_DATA
          COMMAND_DATA
   */
       //check if the data is complete
       int ret = 0;
       int sz = data.GetLength();

       if((data[0]=='<'&&data[1]=='C'&&data[2]=='S'&&data[3]=='>')||
              (data[0]=='<'&&data[1]=='D'&&data[2]=='S'&&data[3]=='>'))
       {
              ret = DATA_IN_CONTINUATION;

              //check if ending tag is also present
              if(data[sz-4]=='<'&&data[sz-3]=='C'&&data[sz-2]=='E'&&data[sz-1]=='>')
                           ret = COMMAND_DATA;
              else if (data[sz-4]=='<'&&data[sz-3]=='D'&&data[sz-2]=='E'&&data[sz-1]=='>')
                           ret = MESSAGE_DATA;
       }
       return ret;
}


void CServerSocket::OnReceive(int nErrorCode)
{
       //when the data is completely recived send th data to the dialog
       //check for commands
       //every command will be enclosed between <CS>command<CE>
       //every message data will be enclosed between <DS>message<DE>

       if(nErrorCode==0)
       {
       char data[100];
       int nRead = Receive(data,100);
       if(nRead != SOCKET_ERROR)
       {
              data[nRead] = '\0';
```

```
	if(check_data(CString(data))==MESSAGE_DATA||check_data(CString(data))==COMMAND_DATA)
		{
			Data+=data;
			CString *DataToBeSent = new CString;
			*DataToBeSent = Data;
			//sent the message to dialog box
			if(!target->PostMessage(UWM_NETWORK_DATA, (WPARAM)DataToBeSent,
(LPARAM)::GetCurrentThreadId()))
		{ /* failed to send */
		 ASSERT(FALSE);
		 delete DataToBeSent;
		} /* failed to send */
			//clear the internal data store
			Data="";
		 }
		 else if(check_data(CString(data))==DATA_IN_CONTINUATION)
			//Store the data
			Data+=data;
	}
	CAsyncSocket::OnReceive(nErrorCode);
}
// CServerSocket member functions
}
```

ServerThread.h

```
#pragma once

#include "ServerSocket.h"

// CServerThread

class CServerThread : public CWinThread
{
	DECLARE_DYNCREATE(CServerThread)

protected:
	CServerThread();		// protected constructor used by dynamic
creation
	virtual ~CServerThread();
	SOCKET socket;
	CWnd* target;
	CServerSocket sock;
public:
	void OnThreadClose(WPARAM,LPARAM);
	void OnSendData(WPARAM,LPARAM);
	void SetSocket(SOCKET h){socket = h;}
	void SetTarget(CWnd *t){target = t;}
	virtual BOOL InitInstance();
	virtual int ExitInstance();

protected:
	DECLARE_MESSAGE_MAP()
};
```

ServerThread.cpp

```
// ServerThread.cpp : implementation file
//

#include "stdafx.h"
#include "Chat Server.h"
#include "ServerThread.h"
#include "messages.h"

// CServerThread

IMPLEMENT_DYNCREATE(CServerThread, CWinThread)

CServerThread::CServerThread()
{
}

CServerThread::~CServerThread()
{
}

BOOL CServerThread::InitInstance()
{
	target->PostMessage(UWM_THREADSTART, 0, (LPARAM)m_nThreadID);

	ASSERT(target != NULL);
    if(target == NULL)
	 return FALSE;
    AfxSocketInit();
	sock.SetTarget(target);
	sock.threadID = m_nThreadID;
    if(socket != NULL)
	 { /* attach socket */
	  sock.Attach(socket);
	 } /* attach socket */
	return TRUE;
}

int CServerThread::ExitInstance()
{
	ASSERT(target != NULL);
    if(target != NULL)
	 target->PostMessage(UWM_THREADCLOSE, 0, (LPARAM)m_nThreadID);
	return CWinThread::ExitInstance();
}
void CServerThread::OnSendData(WPARAM Data,LPARAM)
{
	//send the data
	CString* data = (CString *)Data;
	sock.Send(data->GetBuffer(),data->GetLength());
}
void CServerThread::OnThreadClose(WPARAM,LPARAM)
{
```

```
        //end the thread
        ASSERT(target != NULL);
     if(target != NULL)
        target->PostMessage(UWM_THREADCLOSE, 0, (LPARAM)m_nThreadID);
        AfxEndThread(0);
}
BEGIN_MESSAGE_MAP(CServerThread, CWinThread)
        ON_THREAD_MESSAGE(UWM_SEND_DATA,OnSendData)
        ON_THREAD_MESSAGE(UWM_THREADCLOSE,OnThreadClose)
END_MESSAGE_MAP()


// CServerThread message handlers
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently,
// but are changed infrequently

#pragma once
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls'version='6.0.0.0'
processorArchitecture='x86' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#ifndef _SECURE_ATL
#define _SECURE_ATL 1
#endif

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows
headers
#endif

// Modify the following defines if you have to target a platform prior to
the ones specified below.
// Refer to MSDN for the latest info on corresponding values for different
platforms.
#ifndef WINVER                          // Allow use of features specific to
Windows XP or later.
#define WINVER 0x0501           // Change this to the appropriate value to
target other versions of Windows.
#endif

#ifndef _WIN32_WINNT            // Allow use of features specific to
Windows XP or later.
#define _WIN32_WINNT 0x0501     // Change this to the appropriate value to
target other versions of Windows.
#endif

#ifndef _WIN32_WINDOWS          // Allow use of features specific to
Windows 98 or later.
#define _WIN32_WINDOWS 0x0410 // Change this to the appropriate value to
target Windows Me or later.
#endif
```

```
#ifndef _WIN32_IE               // Allow use of features specific to IE
6.0 or later.
#define _WIN32_IE 0x0600  // Change this to the appropriate value to target
other versions of IE.
#endif

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS    // some CString constructors
will be explicit

// turns off MFC's hiding of some common and often safely ignored warning
messages
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions


#include <afxdisp.h>        // MFC Automation classes



#ifndef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>           // MFC support for Internet Explorer 4
Common Controls
#endif
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>             // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


#include <afxsock.h>            // MFC socket extensions

#include <afxcoll.h>

#include <afxdlgs.h>

#ifdef _UNICODE
#if defined _M_IX86
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='x86' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#elif defined _M_IA64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='ia64' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#elif defined _M_X64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#else
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='*' publicKeyToken='6595b64144ccf1df'
language='*'\"")
```

```
#endif
#endif
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// Chat Server.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

messages.h

```
/*****************************************************************************
*                                UWM_THREADSTART
* Inputs:
*       WPARAM: unused
*       LPARAM: (WPARAM)(DWORD) Thread ID
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main GUI thread that the child thread has started
*****************************************************************************/
#define UWM_THREADSTART  (WM_APP+200)

/*****************************************************************************
*                                UWM_THREADCLOSE
* Inputs:
*       WPARAM: unused
*       LPARAM: (LPARAM)(DWORD) thread ID of closing thread
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that a secondary thread has terminated
*****************************************************************************/

#define UWM_THREADCLOSE  (WM_APP+201)

/*****************************************************************************
*                                 UWM_NETWORK_DATA
* Inputs:
*       WPARAM: (WPARAM)(CByteArray *) Pointer to the data that was received
*       LPARAM: (LPARAM)(DWORD) thread ID of thread that received the data
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Passes data to the main thread to be processed and/or displayed
* Notes:
*       It is the responsibility of the recipient to delete the CString
*****************************************************************************/
```

```
#define UWM_NETWORK_DATA    (WM_APP+202)

/*************************************************************************
*                              UWM_TERM_THREAD
* Inputs:
*       WPARAM: unused
*       LPARAM: unused
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Sent via PostThreadMessage to the child thread to shut it down
*************************************************************************/

#define UWM_TERM_THREAD  (WM_APP+203)


/*************************************************************************
*                            UWM_CONNECTIONCLOSE
* Inputs:
*       WPARAM: unused
*       LPARAM lParam: (LPARAM)(DWORD) thread ID of thread that closed
connection
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that the connection has closed
*************************************************************************/

#define UWM_CONNECTIONCLOSE (WM_APP + 204)


/*************************************************************************
*                             UWM_CONNECTIONMADE
* Inputs:
*       WPARAM: unused
*       LPARAM: unused
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that a connection has succeeded
*************************************************************************/

#define UWM_CONNECTIONMADE (WM_APP + 205)

/*************************************************************************
*                             UWM_NETWORK_ERROR
* Inputs:
*       WPARAM: (WPARAM)(DWORD) error code
*       LPARAM: (LPARAM)(DWORD) thread ID of thread that had the error
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the main thread that a network error has occurred
*************************************************************************/

#define UWM_NETWORK_ERROR (WM_APP+207)

/*************************************************************************
```

```
*                              UWM_SEND_COMPLETE
* Inputs:
*       WPARAM: unused
*       LPARAM: (LPARAM)(DWORD) thread ID of thread that completed sending
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Notifies the owner that the last Send has completed
********************************************************************************/

#define UWM_SEND_COMPLETE (WM_APP+208)

/********************************************************************************
*                                UWM_SEND_DATA
* Inputs:
*       WPARAM: (WPARAM)(CByteArray*) Reference to a CByteArray to be sent
*       LPARAM: unused
* Result: LRESULT
*       Logically void, 0, always
* Effect:
*       Enqueues a send request
* Notes:
*       This is sent to the thread via PostThreadMessage
*       The sender must allocate the CByteArray on the heap.  The network
*       layers will delete this object when it has been sent
********************************************************************************/

#define UWM_SEND_DATA (WM_APP + 209)

/********************************************************************************
*                             UWM_START_NEXT_PACKET
* Inputs:
*       WPARAM: unused
*       LPARAM: unused
* Result: void
*
* Effect:
*       This is sent via PostThreadMessage from the network thread to itself
*       to initiate the asynchronous dequeue of the next pending packet
********************************************************************************/

#define UWM_START_NEXT_PACKET (WM_APP + 210)

/********************************************************************************
*                             UWM_NEWTARGET
* Inputs:
*       WPARAM: (WPARAM)(CWnd*) address of the new property page
*       LPARAM: unused
* Result: void
*
* Effect:
*       This is sent via PostThreadMessage from the network thread to itself
*       to initiate the asynchronous dequeue of the next pending packet
********************************************************************************/

#define UWM_NEWTARGET (WM_APP + 211)
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Chat Server.rc
//
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_CHATSERVER_DIALOG           102
#define IDP_SOCKETS_INIT_FAILED         103
#define IDD_PROPERTYPAGE                105
#define IDR_MAINFRAME                   128
#define IDB_SMILE                       129
#define IDB_SAD                         130
#define IDB_WINK                        131
#define IDB_GRIN                        132
#define IDB_TOUNGUE_SMILE               133
#define IDB_TONGUE_SMILE                133
#define IDB_SURPRISE                    134
#define IDB_WORRIED                     135
#define IDB_STRAIGHT_FACE               136
#define IDC_CHAT_LIST                   1001
#define IDC_TEXTBOX                     1002
#define IDC_CONNECTED                   1003
#define IDC_ROOMS                       1004
#define IDC_SEND                        1005
#define IDC_COLOR                       1006

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        138
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1008
#define _APS_NEXT_SYMED_VALUE           105
#endif
#endif
```