Δίκτυα Υπολογιστών ΙΙ (8ο εξάμηνο) Experimental Virtual Lab

Εργασία δικτυακού προγραμματισμού (Java socket programming)

Η εργασία που ζητείται να εκπονηθεί στο μάθημα Δίκτυα Υπολογιστών ΙΙ αποτελεί μία εφαρμογή δικτυακού προγραμματισμού (network programming). Η εργασία στοχεύει στην ανάπτυξη πειραματικής δικτυακής εφαρμογής με τη γλώσσα προγραμματισμού Java, την εξοικείωση με το πρωτόκολλο επικοινωνίας υπολογιστών UDP (user datagram protocol), την εισαγωγή στους μηχανισμούς μετάδοσης ψηφιακού ήχου σε πραγματικό χρόνο (audio streaming) διαμέσου δικτύων μεταγωγής πακέτων και τη συλλογή στατιστικών μετρήσεων τιμών ορισμένων παραμέτρων που συμβάλλουν μαζί με άλλες στη διαμόρφωση της ποιότητας της επικοινωνίας των υπολογιστών στο Internet.

Η εφαρμογή μπορεί να αναπτυχθεί αυτοτελώς σε υπολογιστές του εργαστηρίου του Τομέα ή αλλού. Ζητείται όμως ορισμένες μετρήσεις να πραγματοποιηθούν σε συνεργασία με τον server του πειραματικού εικονικού εργαστηρίου (experimental virtual lab) που αναπτύσσεται στα πλαίσια των μαθημάτων Δίκτυα Υπολογιστών Ι και ΙΙ. Η διεύθυνση http://ithaki.eng.auth.gr/netlab/index.html αποτελεί το σημείο αναφοράς των δραστηριοτήτων που ακολουθούν στην εργασία αυτή.

Δημιουργία περιβάλλοντος προγραμματισμού σε Java

Για την εργασία απαιτείται η διάθεση του περιβάλλοντος προγραμματισμού `Java Development Kit JDK` το οποίο παρέχεται δωρεάν από τη διεύθυνση http://java.sun.com/javase/downloads/index.jsp Η εγκατάσταση και η εξοικείωση με το περιβάλλον αυτό μπορεί να διευκολυνθεί από το ηλεκτρονικό εκπαιδευτικό υλικό που διατίθεται από τη διεύθυνση http://java.sun.com/docs/books/tutorial/index.html επίσης δωρεάν. Πλήρης τεκμηρίωση σε ηλεκτρονική μορφή για την Java μπορεί να βρεθεί στη διεύθυνση http://java.sun.com/javase/6/docs/index.html. Συμπληρωματικά τα βιβλία Learning Java, 3rd edition, O'Reilly Associates και Java Network Programming, 3rd edition, O'Reilly Associates αποτελούν χρήσιμα εκπαιδευτικά βοηθήματα ενώ το βιβλίο Java 2 in a Nutshell : A Quick Reference Guide, 5th edition, O'Reilly Associates αποτελεί μία συνοπτική αλλά εξαιρετικά περιεκτική αναφορά στη γλώσσα προγραμματισμού Java και τις βασικές βιβλιοθήκες της.

Εργαλεία προγραμματισμού σε Java

Κάθε εφαρμογή σε Java απαιτεί **(α)** τη δημιουργία κώδικα σε πηγαία μορφή (source code), **(β)** τη μεταγλώττισή του σε κώδικα μηχανής Java (Java virtual machine code JVM code) και **(γ)** την εκτέλεσή του με τον διερμηνευτή Java (Java interpreter). Στα πλαίσια της εργασίας αυτής προτείνεται για το στάδιο (α) η χρήση του επεξεργαστή κειμένου `notepad` ενώ για τα στάδια (β) και (γ) η χρήση των εργαλείων `javac` και `java` αντίστοιχα μέσα από παράθυρο γραμμών εντολών[1] (command line window). Τα

[1] Εναλλακτικά τα ενοποιημένα περιβάλλοντα ανάπτυξης Java IDEs (integrated development environments) NetBeans και Eclipse διατίθενται επίσης δωρεάν από τις διευθύνσεις http://www.netbeans.org και http://www.eclipse.org αντίστοιχα

εργαλεία `javac` και `java` περιλαμβάνονται στην εγκατάσταση του `J2SDK` που αναφέρθηκε παραπάνω. Το `notepad` είναι διαθέσιμο σε οποιοδήποτε υπολογιστή Win/98/Me/2K/XP/Vista.

Δικτυακές εφαρμογές σε Java

Η εργασία επικεντρώνεται στην εξοικείωση με τους μηχανισμούς επικοινωνίας υπολογιστών μέσω αποστολής και λήψης μεμονωμένων πακέτων πληροφορίας γνωστών ως πακέτων τύπου `datagram` (user datagram protocol UDP packets).

Οι βιβλιοθήκες οι οποίες κατεξοχήν θα χρησιμοποιηθούν στην εργασία είναι η `java.net` και η `java.io`. Η πρώτη περιλαμβάνει κλάσεις (Java classes) διαχείρισης δικτυακών πόρων (network resources) ενώ η δεύτερη κλάσεις διαχείρισης υπολογιστικών πόρων (computer resources).

Η βιβλιοθήκη `java.net` διαθέτει μεταξύ άλλων την κλάση `DatagramSocket` η οποία επιτρέπει τη δημιουργία συνδέσμων (`sockets`) μέσω των οποίων αποστέλλονται ή λαμβάνονται πακέτα τύπου UDP χωρίς όμως να εξασφαλίζεται η αξιόπιστη μετάδοση και παράδοσή τους στον προορισμό τους. Τα πακέτα UDP έχουν μία εξαιρετικά απλή δομή η οποία μεταφέρει ως έχουν τα bytes ενός πίνακα (byte array) κατά μήκος του δικτύου. Η μεταφορά γίνεται με μία μόνο ενέργεια αποστολής πακέτου αξιοποιώντας με τον καλύτερο δυνατό τρόπο τους πόρους του δικτύου (best effort transmission). Ο έλεγχος της άφιξης του πακέτου στον προορισμό του και η ένταξή του σε σύνολο παρόμοιων πακέτων πληροφορίας που διακινούνται μεταξύ δύο υπολογιστών είναι ευθύνη των ιδίων των δικτυακών εφαρμογών που τρέχουν στους υπολογιστές αυτούς.

Η μέθοδος `send()` αποστέλλει ένα πακέτο `datagram` μέσω της σύνδεσης `DatagramSocket`. Το πακέτο πρέπει να περιέχει τη διεύθυνση (`IP address`) του υπολογιστή προς τον οποίο αποστέλλεται καθώς επίσης και τον αριθμό της θύρας (`port number`) στην οποία απευθύνεται. Η μέθοδος `receive()` αναμένει, στα πλαίσια μίας εφαρμογής, την άφιξη ενός πακέτου. Μόλις το πακέτο παραληφθεί, η μέθοδος το καταχωρεί μαζί με τη διεύθυνση του υπολογιστή του αποστολέα. Ας σημειωθεί ότι η μέθοδος `receive()` προκαλεί αναστολή της εκτέλεσης των υπόλοιπων εντολών που ακολουθούν στην εφαρμογή σε όλη τη διάρκεια αναμονής της μέχρι την άφιξη του επόμενου πακέτου (blocking operation[2]). Η μέθοδος `setSoTimeout()` επιτρέπει τον ορισμό ενός μεγίστου χρόνου αναμονής έτσι ώστε αν για κάποιο λόγο δεν φτάσει πακέτο στο πέρας αυτού του χρόνου, η εκτέλεση της μεθόδου `receive()` παύει και η εκτέλεση της εφαρμογής συνεχίζει με τις αμέσως επόμενες εντολές.

Η κλάση `DatagramPacket` επιτρέπει τον ορισμό παραμέτρων που σχετίζονται με τη δομή των πακέτων `datagram` που διακινούνται από έναν υπολογιστή διαμέσου μίας σύνδεσης `DatagramSocket`. Οι μέθοδοι `setAddress()` και `setPort()` της

[2] Η νεότερη βιβλιοθήκη java.nio διαθέτει μεθόδους οι οποίες δεν αναστέλλουν την εκτέλεση της εφαρμογής (non-blocking operations)

κλάσης αυτής προσδιορίζουν τη διεύθυνση Internet και τον αριθμό της θύρας αντίστοιχα του υπολογιστή προς τον οποίο πρόκειται να αποσταλεί ένα πακέτο `datagram`.

Η βιβλιοθήκη `java.io` διαθέτει πλήθος κλάσεων για τη διαχείριση αρχείων. Για παράδειγμα η μέθοδος κατασκευής αντικειμένων `File()` επιτρέπει τη δημιουργία αντικειμένων τύπου `File` και τη συσχέτισή τους με αρχεία στο σκληρό δίσκο μέσω των ονομάτων τους όπως αυτά δίνονται από το `filesystem` του υπολογιστή. Οι κλάσεις `FileInputStream` και `FileOutputStream` παρέχουν μεθόδους για ανάγνωση ή εγγραφή bytes από/προς αρχεία δημιουργώντας αντίστοιχες ροές (byte streams) από/προς αυτά.

Μεταξύ των θεμελιωδών κλάσεων Java περιλαμβάνεται και η κλάση `java.lang.System` η οποία διαθέτει τη μέθοδο `currentTimeMillis()`. Η μέθοδος αυτή επιστρέφει σε μία εφαρμογή την τρέχουσα ώρα του συστήματος σε milliseconds. Καλούμενη σε διαφορετικά σημεία μιας εφαρμογής κατά την εκτέλεσή της, η μέθοδος παρέχει την δυνατότητα χρονομέτρησης γεγονότων (events) όπως αυτά των αναχωρήσεων και αφίξεων πακέτων. Η εργασία που ζητείται να εκπονηθεί στη συνέχεια του εικονικού εργαστηρίου στηρίζεται πρακτικά στη χρονομέτρηση τέτοιων γεγονότων.

Εγγραφή και αναπαραγωγή ψηφιακού ήχου (digital audio)

Η εργασία στοχεύει στην εξοικείωση με τους μηχανισμούς μετάδοσης ψηφιακού ήχου στο Internet σε πραγματικό χρόνο με τη μορφή ροών πακέτων ήχου (audio streaming). Η μετάδοση ήχου προϋποθέτει με τη σειρά της εξοικείωση με τους μηχανισμούς εγγραφής και αναπαραγωγής ψηφιακού ήχου από διάφορες πηγές (audio sources) και με διάφορες διατάξεις ήχου (audio devices) που συναντώνται στους υπολογιστές σήμερα.

Η βιβλιοθήκη `javax.sound.sampled` παρέχει το σύνολο των απαιτουμένων αντικειμένων Java για την εγγραφή και αναπαραγωγή ψηφιακού ήχου στον υπολογιστή. Ειδικότερα το αντικείμενο `AudioSystem` παρέχει τις μεθόδους προσπέλασης στις διατάξεις εισόδου/εξόδου ήχου προς/από τον υπολογιστή ενώ το αντικείμενο `AudioFormat` παρέχει τις μεθόδους μορφοποίησης του εγγραφόμενου ή αναπαραγόμενου ψηφιακού ήχου.

Για λόγους απλοποίησης της εφαρμογής, στην εργασία υιοθετείται η μονοφωνική παλμοκωδική διαμόρφωση ήχου PCM (pulse code modulation) με συχνότητα δειγματοληψίας 8000 samples/sec και ομοιόμορφη προσημασμένη γραμμική κβάντιση ακριβείας Q bits/sample. Για τη μορφοποίηση του ήχου με αυτά τα χαρακτηριστικά, η μέθοδος (audio format constructor) `AudioFormat(8000,Q,1,true,false)` επιστρέφει ένα αντικείμενο μορφοποίησης ήχου το οποίο χρησιμοποιείται σε επόμενα τμήματα της εφαρμογής για την εγγραφή και την αναπαραγωγή του ήχου. Στα πειράματα που ακολουθούν στην εργασία η παράμετρος Q έχει την τιμή 8 ή 16 bits/sample.

Για την αναπαραγωγή του ήχου προσδιορίζεται αρχικά μία έξοδος ήχου με τη βοήθεια της μεθόδου `AudioSystem.getSourceDataLine` που επιστρέφει στην εφαρμογή

ένα αντικείμενο τύπου `SourceDataLine`. Η έξοδος ενεργοποιείται με τη μέθοδο `open()` η οποία δέχεται δύο ορίσματα. Το πρώτο αναφέρεται στο αντικείμενο μορφοποίησης ήχου που υιοθετείται (audio format) ενώ το δεύτερο στο μέγεθος της εσωτερικής μνήμης (audio buffer) που εξασφαλίζει αδιάλειπτη αναπαραγωγή ήχου με σταθερό ρυθμό (bit rate) σε περιβάλλον πολλαπλών υπολογιστικών εργασιών (multi-tasking environment).

Η αναπαραγωγή ήχου γίνεται με τη μέθοδο write() η οποία συνοδεύει ένα ενεργοποιημένο αντικείμενο εξόδου ήχου τύπου `SourceDataLine`. Η μέθοδος δέχεται τρία ορίσματα. Το πρώτο αναφέρεται σε ένα διάνυσμα τύπου `byte[]` το οποίο φέρει τα δείγματα ήχου προς αναπαραγωγή ενώ το δεύτερο και το τρίτο αναφέρονται στον δείκτη αρχής και το μήκος αντίστοιχα ενός τμήματος του διανύσματος το οποίο αντιγράφεται στην εσωτερική μνήμη της εξόδου ήχου προς αναπαραγωγή.

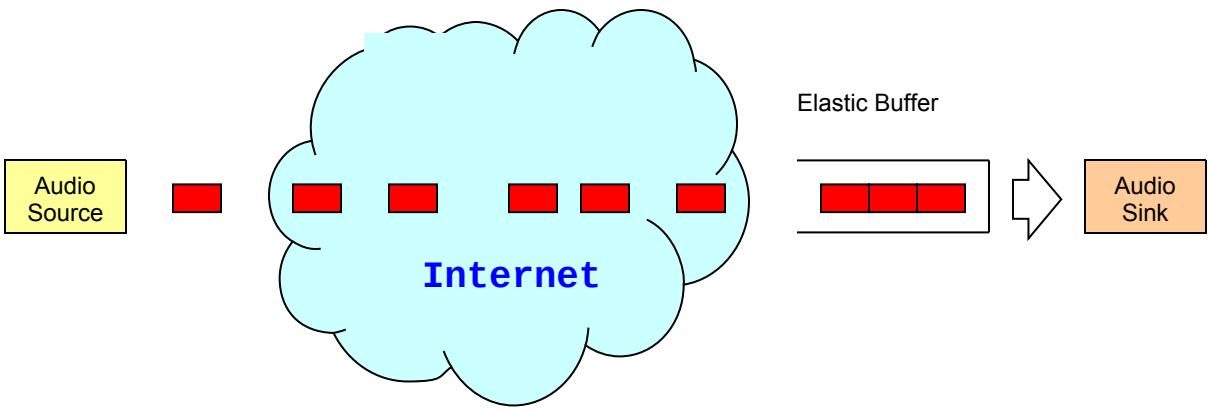
Ας σημειωθεί ότι στην Java μία μεταβλητή τύπου `byte` παριστάνει έναν προσημασμένο ακέραιο (signed integer) που λαμβάνει τιμές από -128 έως +127.

Τέλος, η λειτουργία ενός ενεργοποιημένου αντικειμένου εξόδου ήχου σταματά προσωρινά ή το ίδιο το αντικείμενο απενεργοποιείται πλήρως με τις μεθόδους `stop()`και `close()` αντίστοιχα.

Μετάδοση πακέτων ψηφιακού ήχου στο Internet (packet audio transmission)

Στην εργασία οι παραπάνω μέθοδοι αναπαραγωγής ήχου χρησιμοποιούνται σε συνδυασμό με τις μεθόδους αποστολής και λήψης πακέτων ήχου τύπου `datagram` για τη δημιουργία ad-hoc ροών πακέτων ψηφιακού ήχου σε πραγματικό χρόνο (audio streaming).

Επειδή η αναπαραγωγή ήχου θα πρέπει να γίνεται με σταθερό ρυθμό παρά το γεγονός ότι οι αφίξεις πακέτων ήχου στον δέκτη δεν γίνονται με σταθερό ρυθμό (λόγω αρχικής συμπίεσης και μετέπειτα στατιστικής πολυπλεξίας και μεταγωγής πακέτων) υιοθετείται ο μηχανισμός ελαστικής μνήμης (elastic buffer) στον δέκτη όπως φαίνεται στο σχήμα που ακολουθεί.



Η υλοποίηση της ελαστικής μνήμης στην εφαρμογή μπορεί να πάρει τη μορφή ενός πίνακα δύο διαστάσεων. Στον πίνακα αυτόν αποθηκεύονται τα πακέτα ήχου, ένα σε κάθε γραμμή, καθώς αυτά καταφτάνουν στον δέκτη σε τυχαίες χρονικές στιγμές. Από τον ίδιο πίνακα, πακέτα ήχου τα οποία ήδη είναι καταχωρημένα αποσπώνται μέσω της εφαρμογής του δέκτη με σταθερό ρυθμό προς αναπαραγωγή από τη διάταξη εξόδου ήχου του δέκτη.

Επειδή τόσο η μέθοδος `receive()` λήψης πακέτων τύπου `datagram` όσο και η μέθοδος `write()` αναπαραγωγής πακέτων ήχου αναστέλλουν την εκτέλεση του προγράμματος (blocking operations), η διαχείριση της ελαστικής μνήμης, σύμφωνα με μία προγραμματιστική προσέγγιση, μπορεί να ανατεθεί σε δύο διαφορετικές αλλά συνεργαζόμενες διαδικασίες με την μορφή προγραμματιστικών νημάτων (programming threads) σε περιβάλλον πολυ-νηματικού προγραμματισμού (multi-thread programming). Σε ένα τέτοιο περιβάλλον, ένα νήμα `rxThread` αναλαμβάνει την ενημέρωση της ελαστικής μνήμης με νέα πακέτα ήχου ό,ποτε αυτά καταφτάνουν στο δέκτη σε τυχαία χρονικά διαστήματα ενώ ένα δεύτερο νήμα `playThread` αναλαμβάνει την ανανέωση της εσωτερικής μνήμης (audio buffer) της διάταξης εξόδου ήχου αποσπώντας σε τακτά χρονικά διαστήματα πακέτα ήχου τα οποία ήδη βρίσκονται στην ελαστική μνήμη (elastic buffer) του δέκτη.

Είναι αυτονόητο ότι η απόσπαση πακέτων ήχου από την ελαστική μνήμη δεν μπορεί να προτρέχει της ενημέρωσης της ελαστικής μνήμης με νέα πακέτα που καταφτάνουν από το δίκτυο. Ο έλεγχος των διαδικασιών αυτών (buffer control) αποτελεί θεμελιώδες ζήτημα στη διαχείριση ροών πακέτων ήχου σε πραγματικό χρόνο (audio streaming) και απαιτεί σε πολλές περιπτώσεις τη συνεργασία μεταξύ πομπού και δέκτη (βλ. RealNetworks Streaming Server και Microsoft Media Player). Σε πολλές περιπτώσεις σήμερα διαφαίνεται επιπλέον η ανάγκη συνεργασίας με το ίδιο το δίκτυο (πχ MPEG-21) για την εξασφάλιση αποδεκτής ποιότητας ήχου από άκρη σε άκρη (network end-to-end quality of service E2E QoS) σε συνθήκες υψηλού φορτίου.

Ωστόσο, με σκοπό την ελαχιστοποίηση της πολυπλοκότητας της ζητούμενης εφαρμογής, στην εργασία υιοθετείται ένα σχήμα ελέγχου ανοικτού βρόγχου δηλαδή χωρίς τη συνεργασία πομπού και δέκτη όπως αναφέρεται στη συνέχεια.

Η εφαρμογή στο εικονικό εργαστήριο του μαθήματος

Ζητείται η ανάπτυξη εφαρμογής Java η οποία **(α)** επικοινωνεί με τον server του εργαστηρίου μέσω συνδέσεων `datagram` (Java `UDP socket` programming) και **(β)** επιτρέπει στατιστικές μετρήσεις τιμών ορισμένων παραμέτρων της επικοινωνίας αυτής. Τα χαρακτηριστικά της εφαρμογής και οι ζητούμενες μετρήσεις περιγράφονται στη συνέχεια.

[1] Όπως εμφανίζεται στα σχήματα που ακολουθούν, με την επιλογή της εργασίας Java socket programming από τη διεύθυνση http://ithaki.eng.auth.gr/netlab/index.html ο server θέτει αυτομάτως στην αποκλειστική διάθεση καθενός/καθεμίας φοιτητή/φοιτήριας ένα αντίγραφο (νήμα, thread) της εφαρμογής `echoServerThread`.

[2] Μεταξύ της επιλογής της εργασίας και της ενεργοποίησης της εφαρμογής `echoServerThread` μεσολαβεί χρόνος έως και 60 δευτερόλεπτα. Ο συνολικός χρόνος που διατίθεται το αντίγραφο αυτό της εφαρμογής online εμφανίζεται στην ιστοσελίδα απόκρισης του server μετά από επιτυχή επιλογή. Μετά το πέρας του χρόνου αυτού και εάν είναι επιθυμητή η συνέχεια, απαιτείται νέα επιλογή της ίδιας εργασίας.

[3] Η εφαρμογή `echoServerThread` δημιουργεί αυτόματα μία σύνδεση `UDP socket` η οποία τίθεται σε «ακρόαση» στη διεύθυνση `155.207.18.208` του server και στη θύρα (`socket port`) με αριθμό που προσδιορίζεται από την παράμετρο `server listening port` στην ίδια ιστοσελίδα απόκρισης του server.

[4] Η εφαρμογή `userApplication` που ζητείται να κατασκευαστεί θα πρέπει να επιτρέπει την αποστολή ενός μεγάλου αριθμού πακέτων `clientRequest datagram` διαδοχικά προς τον server. Σε κάθε επιτυχή λήψη ενός πακέτου `clientRequest datagram` ο server αποκρίνεται με ένα ή περισσότερα πακέτα `serverResponse datagram`.

[5] Τα πακέτα `serverResponse` αποστέλλονται από τον server στην εφαρμογή `userApplication` στη διεύθυνση `client address` και στη θύρα με αριθμό `client listening port` όπως εμφανίζονται στην ιστοσελίδα απόκρισης του server.

[6] Για την επικοινωνία `request/response`, τα πακέτα `clientRequest` πρέπει να φέρουν ως περιεχόμενο `info (payload)` τον κωδικό `echo_request_code` που δίνεται επίσης στην ιστοσελίδα απόκρισης του server. Με τη σειρά του ο server ενθυλακώνει σε κάθε πακέτο `serverResponse` την ημερομηνία και την ώρα του συστήματος ως εξης :

PSTART `DD-MM-YYYY HH:MM:SS` PSTOP

Μετά από κάθε λήψη πακέτου `clientRequest` και πριν από την αποστολή του πακέτου `serverResponse`, ο server καθυστερεί για χρονικό διάστημα του οποίου η διάρκεια σε milliseconds προσδιορίζεται τυχαία σύμφωνα με μία κατανομή πιθανότητας. Η μορφή και οι παράμετροί της προσδιορίζονται εκ νέου και αυτομάτως από τον server στην αρχή κάθε συνόδου (session) στην οποία προβαίνει ο/η φοιτητής/φοιτήτρια με τον server του εικονικού εργαστηρίου. Η παρεμβολή της καθυστέρησης αυτής αναιρείται μετά από κάθε πακέτο `clientRequest` που φέρει τον κωδικό `echo_request_code=E000`. Η αναίρεση ισχύει μόνο για το αμέσως επόμενο πακέτο `server Response` που ακολουθεί.

[7] Η χρήση του κωδικού `image_request_code` επιτρέπει την αποστολή μέσω διαδοχικών πακέτων `serverResponse` του τρέχοντος πλαισίου εικόνας από κωδικοποιητή εικόνας που φιλοξενείται στον ίδιο server του εικονικού εργαστηρίου στη διεύθυνση http://ithaki.eng.auth.gr/virtual/vlabProject.html και παρέχει real-time video με την κίνηση της Εγνατίας Οδού στο τμήμα εμπρός από την Πολυτεχνική Σχολή. Η αποστολή του πλαισίου γίνεται με διαδοχικά πακέτα UDP μήκους 128 bytes το καθένα εκτός από το τελευταίο. Το πλήθος των πακέτων αυτών καθώς και το μήκος του τελευταίου πακέτου ποικίλει ανάλογα με το συνολικό μήκος του τρέχοντος πλαισίου `imageFrame` σε bytes. Διαδοχικές αποστολές του κωδικού `image_request_code` στον server σε συνδυασμό με την απευθείας απεικόνιση[3] των λαμβανόμενων imageFrames στην οθόνη του τερματικού οδηγεί σε αναπαραγωγή κινούμενου video με ρυθμό ανανέωσης των imageFrames ανάλογα με τις τρέχουσες συνθήκες στο δίκτυο.

Η αποστολή του κωδικού `image_request_code` για τη λήψη εικόνας μπορεί να συνοδευτεί με την ένδειξη `CAM=1` ή `CAM=2` η οποία προσδιορίζει στην είσοδο του κωδικοποιητή την έξοδο από αντίστοιχη κάμερα που λειτουργεί online με τον server του εικονικού εργαστηρίου. Το γεγονός αυτό έχει ώς αποτέλεσμα τη λήψη στην εφαρμογή `userApplication` εικόνας από την προσδιοριζόμενη κάμερα. Σημειώνεται ότι **(α)** σε απουσία της παραμέτρου `CAM` ο server θεωρεί εκ προοιμίου την τιμή `CAM=2` (default value), **(β)** η πρώτη κάμερα παράγει εικόνες με ανάλυση 640x480 pixels ενώ η δεύτερη παράγει εικόνες με ανάλυση 320x240 pixels και **(γ)** η οπτική γωνία της πρώτης κάμερας παραμένει σταθερή ενώ της δεύτερης αλλάζει είτε κάθε 4 λεπτά της ώρας είτε σε τυχαία χρονικά διαστήματα αν εκείνη τη χρονική περίοδο η κάμερα βρίσκεται υπό τηλε-έλεγχο από επισκέπτη του εικονικού εραστηρίου μέσα από τη διεύθυνση Ιντερνετ του server.

[8] Η χρήση του κωδικού `sound_request_code` οδηγεί στην αποστολή από τον server μίας ακολουθίας πακέτων ήχου το καθένα τύπου `datagram`. Η παράμετρος `YXXX` που συνοδεύει τον κωδικό προσδιορίζει την πηγή ήχου `Y` που ενεργοποιείται καθώς και τον αριθμό `XXX` των πακέτων ήχου που αποστέλλονται. Η τιμή `Y=T` ενεργοποιεί εικονική γεννήτρια συχνοτήτων στην περιοχή 200 - 4000 Hz ενώ η τιμή `Y=F` ενεργοποιεί την αναπαραγωγή τμήματος ενός audio clip που επιλέγεται τυχαία από το πειραματικό ρεπερτόριο μουσικής που διαθέτει ο server του εικονικού εργαστηρίου.

Με σκοπό τη συμπίεση της πληροφορίας, τα πακέτα ήχου κωδικοποιούνται σύμφωνα με τις αρχές της απλής διαφορικής παλμοκωδικής διαμόρφωσης DPCM (differential pulse code modulation).

Η διαφορά δύο διαδοχικών δειγμάτων ήχου x_i και x_{i-1} κανονικοποιείται, κβαντίζεται με ομοιόμορφο μη προσαρμοζόμενο κβαντιστή και κωδικοποιείται με 4 bits από την τιμή -8 έως την τιμή +7. Στη συνέχεια, οι διαδοχικές τιμές διαφορών τοποθετούνται ανά δύο σε

[3] Η τεχνική αυτή οδηγεί σε ad-hoc υλοποίηση ενός πρωτοκόλλου μετάδοσης video τύπου motion jpeg (MJPEG) σύμφωνα με το οποίο κάθε imageFrame αποστέλλεται με τη μορφή αρχείου .jpeg. Η διαχείριση αρχείων εικόνων τύπου .jpeg (απεικόνιση, αποθήκευση, μετασχηματισμός format, κ.α.) δεν περιλαμβάνεται στις απαιτήσεις της εργασίας αυτής.

Πειραματική κωδικοποίηση DPCM και AQ-DPCM

$$\mu = \left[\sum_{i=1}^{256} x_i \right] * 256^{-1}$$

$$X_i = x_i - \mu \qquad i = 1,\ldots,256$$

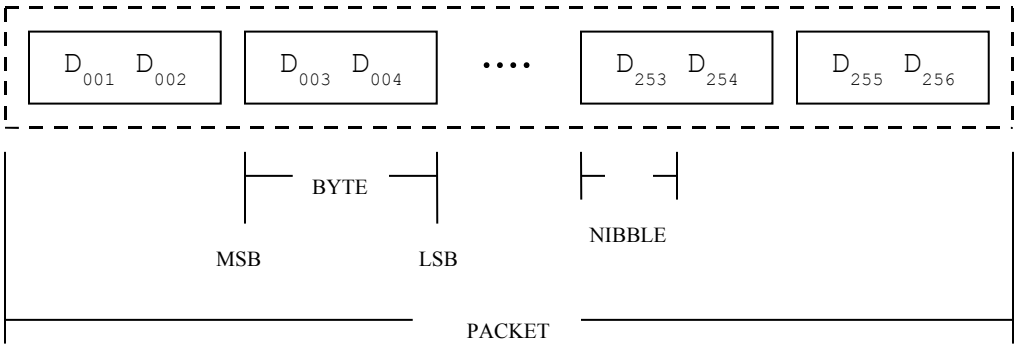
$$\Delta_i = X_i - X_{i-1} \qquad i = 1,\ldots,256$$

$$\sigma^2 = \left[\sum_{i=1}^{256} \Delta_i^2 \right] * 256^{-1} \qquad \beta = [\, Q(\sigma) \,]^{-1}$$

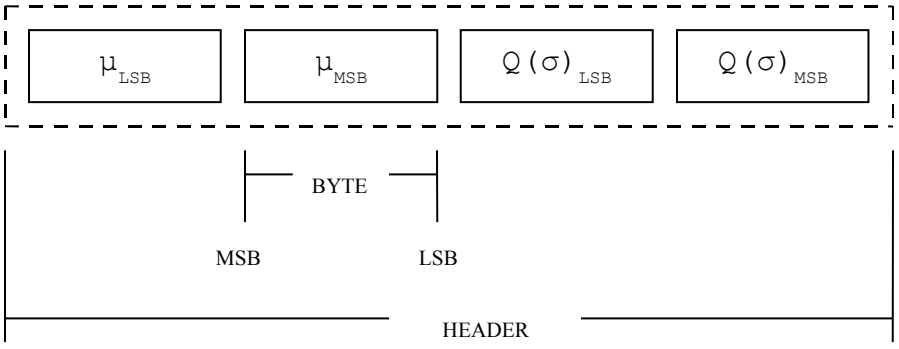
$$d_i = \Delta_i * \beta \qquad i = 1,\ldots,256$$

$$D_i = (d_i + 8) \text{ AND } 0xF \qquad i = 1,\ldots,256$$

Δομή πακέτου DPCM και AQ-DPCM



Δομή επικεφαλίδος πακέτου AQ-DPCM



ένα byte αφού προηγουμένως σε καθεμία τιμή διαφοράς προστεθεί, για λόγους απλοποίησης της εφαρμογής, η τιμή +8 ώστε το κάθε nibble του byte να παριστά θετικό ακέραιο αριθμό μεταξύ των τιμών 0 και +15 (unsigned nibbles).

Τα μεταδιδόμενα πακέτα αποτελούνται από 128 bytes ζευγών διαφορών τα οποία σύμφωνα με τα παραπάνω αντιστοιχούν σε 256 δείγματα ήχου.

Στον δέκτη, τα δείγματα ήχου 32 τέτοιων πακέτων αναπαραγόμενα[4] με σταθερό ρυθμό 8000 samples/sec αντιστοιχούν περίπου σε 1 δευτερόλεπτο ήχου. Επομένως ο μέγιστος αριθμός πακέτων ήχου που αποστέλλονται μετά από μία αίτηση μέσω του `sound_request_code` και της παραμέτρου `YXXX` αντιστοιχεί σε 30 δευτερόλεπτα ήχου.

Για λόγους πειραματισμού, ο server του εικονικού εργαστηρίου διαθέτει επιπλέον διάταξη προσαρμοζόμενου κβαντιστή ο οποίος οδηγεί σε προσαρμοζόμενο μηχανισμό κωδικοποίησης AQ-DPCM (adaptive quantiser - differential pulse code modulation).

Η προσαρμογή γίνεται κάθε 256 δείγματα ήχου προσδιορίζοντας το βήμα ενός βέλτιστου ομοιόμορφου κβαντιστή των 4 bits με βάση την υπόθεση κανονικής κατανομής των τιμών των διαφορών (Gaussian error distribution). Οι τιμές των διαφορών υπολογίζονται, κωδικοποιούνται και τοποθετούνται σε πακέτα των 128 bytes όπως παραπάνω. Η μέση τιμή καθώς και το βήμα του κβαντιστή των 256 δειγμάτων ήχου κωδικοποιούνται, ως προσαρμοζόμενες παράμετροι κωδικοποίησης, γραμμικά με ακρίβεια 16 bits και οι τιμές τους τοποθετούνται σε 4 bytes (overhead) τα οποία προηγούνται του πακέτου των 128 bytes των διαφορών. Το σύνολο αποστέλλεται ως ένα ενιαίο πακέτο UDP μήκους 132 bytes. Κατά το σχηματισμό των 4 bytes των παραμέτρων κωδικοποίησης (header) το λιγότερο σημαντικό byte προηγείται του πλέον σημαντικού byte (little_endian format).

Ο μηχανισμός AQ-DPCM ενεργοποιείται με την αποστολή (στο ίδιο UDP request packet) προς τον server του εικονικού εργαστηρίου της παραμέτρου-ένδειξης `AQ` αμέσως μετά τον κωδικό `sound_request_code` και πριν την παράμετρο `YXXX`.

Συλλογή μετρήσεων και παρουσίαση αποτελεσμάτων

[A] Ζητείται η παρουσίαση αποτελεσμάτων τουλάχιστον από δύο συνόδους με τον server του εικονικού εργαστηρίου που απέχουν μεταξύ τους τουλάχιστον 48 ώρες.

[B] Από κάθε σύνοδο ζητείται η παρουσίαση **(i)** ενός τουλάχιστον διαγράμματος `G1` το οποίο εμφανίζει, για χρονική διάρκεια τουλάχιστον 4 λεπτών, το χρόνο απόκρισης του συστήματος σε milliseconds για κάθε πακέτο `echo` που έχει αποσταλεί στη διάρκεια αυτήν, **(ii)** ενός τουλάχιστον διαγράμματος `G2` το οποίο εμφανίζει, για χρονική διάρκεια τουλάχιστον 4 λεπτών, τη ρυθμαπόδοση (throughput) του συστήματος υπολογιζόμενη με την τεχνική του κινούμενου μέσου όρου κάθε δευτερόλεπτο για τα 8 ή 16 ή 32 (ενδεικτικά) πλέον πρόσφατα δευτερόλεπτα κάθε φορά, **(iii)** δύο τουλάχιστον επιπλέον

[4] Κατά την αναπαραγωγή, κάθε προσημασμένη διαφορά μπορεί να πολλαπλασιάζεται με ένα συντελεστή β>1 ο οποίος σχετίζεται με τον συντελεστή κανονικοποίησης. Τι παρατηρείτε στην ποιότητα του αναπαραγομένου ήχου των διαφόρων audio clips για διάφορες τιμές του συντελεστή β ?

διαγραμμάτων G3 και G4 αντίστοιχα των περιπτώσεων (i) και (ii) παραπάνω με απενεργοποιημένη την καθυστέρηση που παρεμβάλει ο server και **(iv)** τεσσάρων ιστογραμμάτων G5, G6, G7 και G8 της συχνότητας ή προσεγγιστικά της πιθανότητας εμφάνισης των τιμών που καταγράφονται αντίστοιχα στις μετρήσεις (i) έως (iii) παραπάνω.

[Γ] Από τις μετρήσεις της παραγράφου [Β] ή από μετρήσεις πέραν αυτών, ζητείται η εκτίμησή σας για **(i)** το είδος της κατανομής του χρόνου καθυστέρησης που παρενέβαλε ο server μεταξύ των πακέτων που έστειλε, **(ii)** τη μέση τιμή της καθώς και τη διασπορά την οποία καταγράψατε στην εφαρμογή σας και **(iii)** τιμές συχνοτήτων και τίτλους audio clip που προέρχονται αντιστοιχα από την εικονική γεννήτρια συχνοτήτων και το πειραματικό ρεπερτόριο μουσικής του server του εικονικού εργαστηρίου.

[Δ] Επίσης ζητείται η παρουσίαση **(α)** δύο διαγραμμάτων G9 και G10 τμημάτων κυματομορφών που προέρχονται αντιστοιχα από την εικονική γεννήτρια συχνοτήτων και το πειραματικό ρεπερτόριο μουσικής του server του εικονικού εργαστηρίου, **(β)** τεσσάρων διαγραμμάτων G11, G12, G13 και G14 κατανομών των τιμών των διαφορών[5] αλλά και των τιμών των ίδιων των δειγμάτων των κυματομορφών ήχου που αναπαράγονται κατά την αποκωδικοποίηση σημάτων DPCM και AQ-DPCM και τέλος **(γ)** τεσσάρων διαγραμμάτων G15, G16, G17 και G18 τμημάτων των ακολουθιών τιμών της μέσης τιμής και του βήματος του προσαρμοζόμενου κβαντιστή που λαμβάνονται κατά τη διάρκεια λήψης σήματος AQ-DPCM από την εικονική γεννήτρια συχνοτήτων και από το πειραματικό ρεπερτόριο μουσικής του server του εικονικού εργαστηρίο .

Σημειώνεται ότι, επειδή οι τεχνικές ανάπτυξης πολυ-νηματικών εφαρμογών, όπως αναφέρθηκε νωρίτερα, δεν περιλαμβάνονται στις απαιτήσεις της εργασίας αυτής, η αναπαραγωγή του ήχου, σύμφωνα με μία άλλη προγραμματιστική προσέγγιση, μπορεί να γίνει σε δύο ανεξάρτητα διαδοχικά στάδια. Στο πρώτο στάδιο λαμβάνονται όλα τα πακέτα ήχου που φτάνουν σε τυχαία χρονικά διαστήματα και αποθηκεύονται ακολουθιακά στην ελαστική μνήμη του δέκτη. Στο δεύτερο στάδιο η ελαστική μνήμη τροφοδοτεί σε τακτά χρονικά διαστήματα την εσωτερική μνήμη της διάταξης εξόδου ήχου για την αναπαραγωγή του ήχου με σταθερό ρυθμό αναπαραγωγής. Σύμφωνα με την προσέγγιση αυτήν και με βάση τις παραμέτρους του εικονικού εργαστηρίου που αναφέρθηκαν νωρίτερα η ελαστική μνήμη θα πρέπει να έχει χωρητικότητα έως και 30 δευτερόλεπτα ήχου.

[Ε] Τα παραπάνω αποτελέσματα ζητείται να συνοδεύονται από **(α)** σύντομα σχόλια ή παρατηρήσεις σας, **(β)** μία σύντομη βιβλιογραφική τεχνική αναφορά στο πρωτόκολλο UDP και **(γ)** μία επίσης σύντομη βιβλιογραφική τεχνική αναφορά σε διεθνή πρότυπα audio streaming.

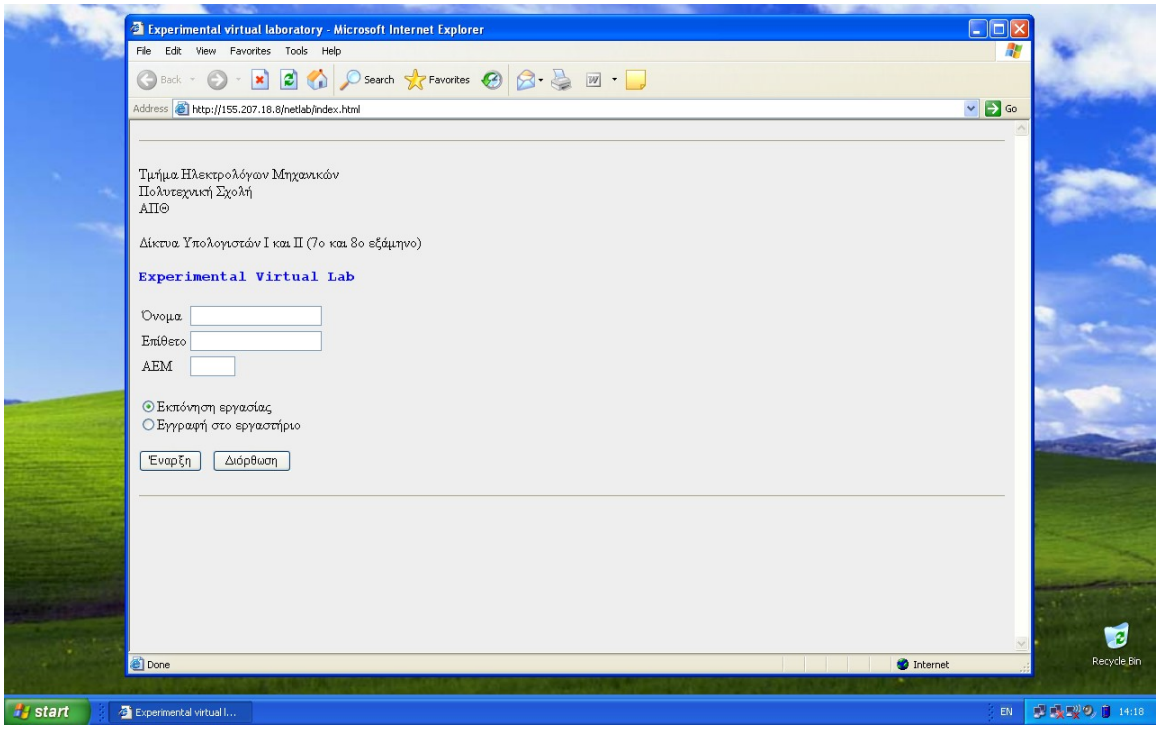
[Ζ] Τα αποτελέσματα μαζί με τα σχόλιά σας, την τεχνική αναφορά και τον πηγαίο κώδικα της εφαρμογής ζητείται να υποβληθούν μέσω ηλεκτρονικού ταχυδρομείου στην ηλεκτρονική διεύθυνση που δίνεται παρακάτω με τις εξής απαραίτητα προδιαγραφές μορφοποίησης **(i)** τα γραφήματα G1 έως G18 μαζί με τα στοιχεία της παραγράφου [Γ] θα

[5] Οι κατανομές που καταγράφονται προσεγγίζουν την κατανομή Gauss, την κατανομή Laplace , άλλη ?

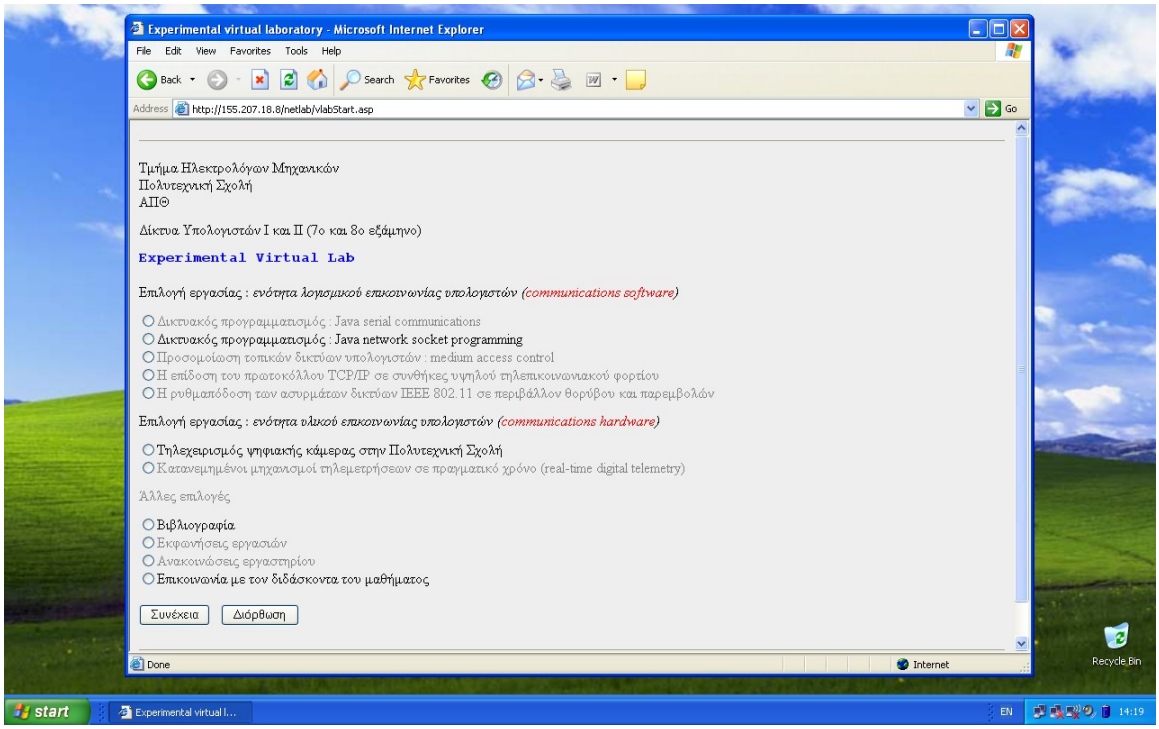
αποτελούν ένα διακριτό αρχείο WORD με όνομα *session1.doc* για την πρώτη σύνοδο και ένα δεύτερο διακριτό αρχείο με όνομα *session2.doc* για τη δεύτερη σύνοδο, **(ii)** στον τίτλο κάθε γραφήματος θα φαίνονται απαραίτητα η ημέρα και η ωρα που έγιναν οι μετρήσεις μαζί με τον κωδικό `{echo,image,sound}_request_code` που χρησιμοποιήθηκε στη διάρκεια των μετρήσεων αυτών, **(iii)** τα κείμενα των σημείων (α), (β) και (γ) της παραγράφου [Ε] θα αποτελούν ένα τρίτο διακριτό αρχείο WORD με όνομα *report.doc,* **(iv)** ο πηγαίος κώδικας της εφαρμογής Java θα αποτελεί ένα τέταρτο διακριτό αρχείο τύπου TEXT με όνομα *source.txt*, **(v)** τα τέσσερα αρχεία *session1.doc, session2.doc, report.doc* και *source.txt* θα αποτελούν τις μόνες συνιστώσες ενός συμπιεσμένου αρχείου τύπου ZIP ή RAR με όνομα *project.zip* ή *project.rar* και τέλος **(vi)** στη γραμμή `Subject:` του μηνύματος ηλεκτρονικού ταχυδρομείου που θα αποσταλεί και θα φέρει προσαρτημένο μόνο το αρχείο *project.zip* ή *project.rar,* θα εμφανίζεται απαραίτητα μόνο το όνομα, το επώνυμο και ο ΑΕΜ του/της φοιτητή/φοιτήτριας που υποβάλει την εργασία.

[Η] Η διεύθυνση επικοινωνίας με τον διδάσκοντα για το εικονικό εργαστήριο και το μάθημα Δίκτυα Υπολογιστών ΙΙ (8ο εξάμηνο) είναι mitrakos@eng.auth.gr

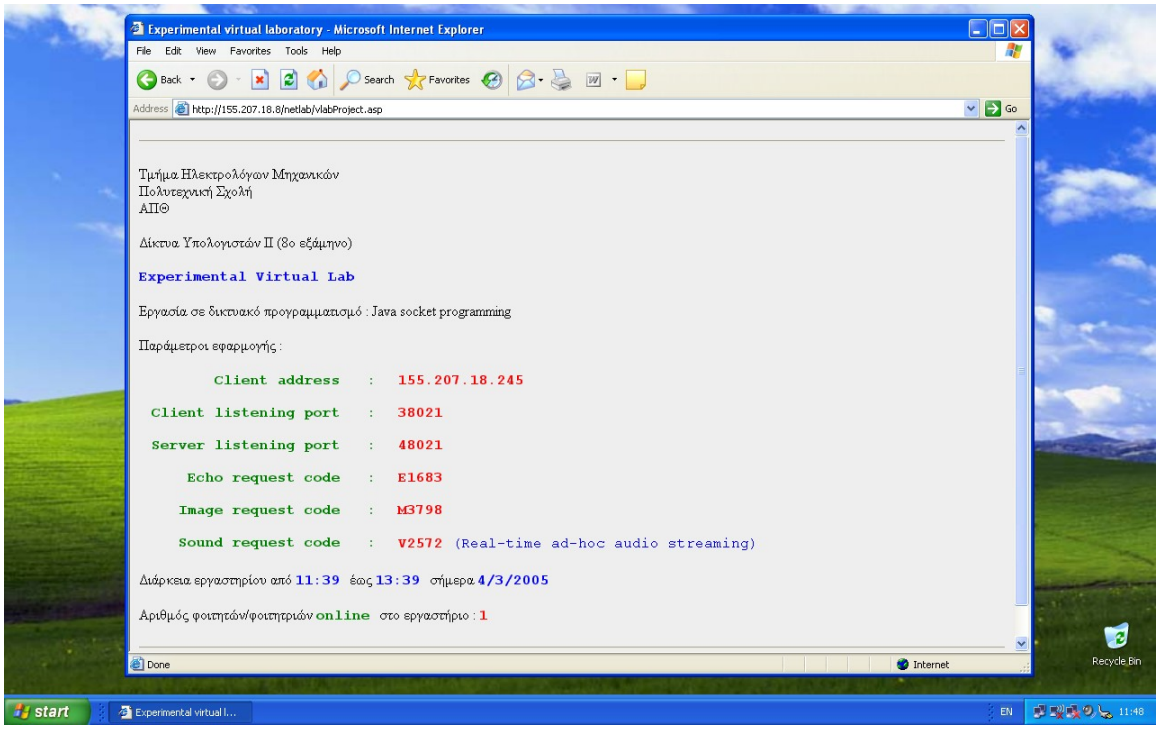
[Θ] Enjoy ☺



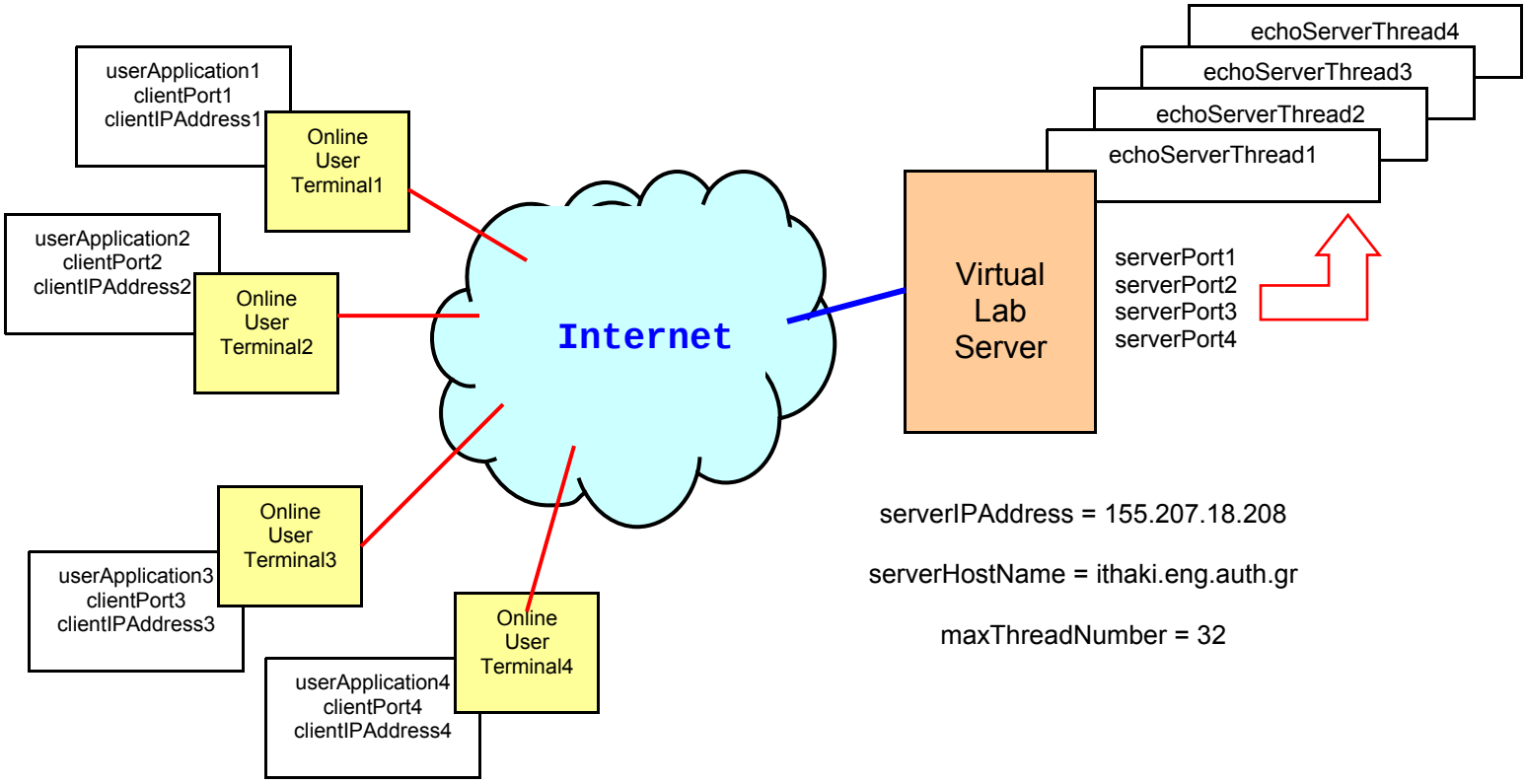
Εικόνα 1 : Η είσοδος στο εικονικό εργαστήριο http://ithaki.eng.auth.gr/netlab/index.html



Εικόνα 2 : Η επιλογή της εργασίας προς εκπόνηση και η έναρξη μίας συνόδου (session) με τον server του εικονικού εργαστηρίου του μαθήματος



Εικόνα 3 : Ενδεικτικές τιμές των προτεινομένων παραμέτρων της εφαρμογής κατά τη διάρκεια μίας συνόδου στο εικονικό εργαστήριο



Σχήμα 1 : Συνοπτικό διάγραμμα επικοινωνιών στο εικονικό εργαστήριο

```
DatagramSocket s = new DatagramSocket();

String packetInfo = "Hello there !"

byte[] txbuffer = packetInfo.getBytes();

int serverPort = 2008;

byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 }

InetAddress hostAddress = InetAddress.getByAddress(hostIP);

DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
                                                hostAddress,serverPort);
s.send(p)


int clientPort = 2004;

DatagramSocket r = new DatagramSocket(clientPort);

r.setSoTimeout(800);

byte[] rxbuffer = new byte[2048];

DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);

for (;;) {
  try {
    r.receive(q);
    message = new String(rxbuffer,0,q.getLength());
  } catch (Exception x) {
    System.out.println(x);
  }
}


AudioFormat linearPCM = new AudioFormat(8000,16,1,true,false);

SourceDataLine lineOut = AudioSystem.getSourceDataLine(linearPCM);

lineOut.open(linearPCM,32000);

lineOut.start();

byte[] audioBufferOut = new byte[8000];

lineOut.write(audioBufferOut,0,8000);

lineOut.stop();

lineOut.close();
```