Δίκτυα Υπολογιστών 2 (7ο εξάμηνο) Experimental Virtual Lab

Εργασία δικτυακού προγραμματισμού (Java socket programming)

Η εργασία που ζητείται να εκπονηθεί στο μάθημα Δίκτυα Υπολογιστών 2 αποτελεί μία εφαρμογή δικτυακού προγραμματισμού (network programming). Η εργασία στοχεύει στην ανάπτυξη πειραματικής δικτυακής εφαρμογής με τη γλώσσα προγραμματισμού Java, την εξοικείωση με τα πρωτόκολλα επικοινωνίας υπολογιστών UDP (user datagram protocol) και TCP (transmission control protocol), την εισαγωγή στους μηχανισμούς μετάδοσης ψηφιακού ήχου σε πραγματικό χρόνο διαμέσου δικτύων μεταγωγής πακέτων (audio packet transmission) 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 προσωπικό υπολογιστή του εργαστηρίου του Τομέα ή άλλου χώρου. Ζητείται όμως ορισμένες μετρήσεις να πραγματοποιηθούν σε συνεργασία με τον σέρβερ Ιθάκη του πειραματικού εικονικού εργαστηρίου (experimental virtual lab) που αναπτύσσεται στα πλαίσια των μαθημάτων Δίκτυα Υπολογιστών 1 και 2. 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 εξοικείωση με το περιβάλλον αυτό μπορεί να διευκολυνθεί με το ηλεκτρονικό εκπαιδευτικό υλικό που διατίθεται επίσης δωρεάν από τη διεύθυνση http:// java.sun.com/docs/books/tutorial/index.html. 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» και «Java Network Programming» των εκδόσεων O'Reilly Associates αποτελούν χρήσιμα εκπαιδευτικά βοηθήματα ενώ το βιβλίο «Java in a Nutshell : A Quick Reference Guide» επίσης των εκδόσεων 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 εικονικής μηχανής Java (Java virtual machine 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`[1] ενώ για τα στάδια (β) και (γ) η χρήση των εργαλείων `javac` και

[1] Εναλλακτικά μπορεί να χρησιμοποιηθεί το εργαλείο Programmer's Notepad που διατίθεται δωρεάν από τη διεύθυνση http://www.pnotepad.org ή κάποιο από τα ενοποιημένα περιβάλλοντα ανάπτυξης Java IDEs (integrated development environments) NetBeans ή Eclipse που διατίθενται επίσης δωρεάν από τις διευθύνσεις http://www.netbeans.org και http://www.eclipse.org αντίστοιχα

`java` αντίστοιχα μέσα από παράθυρο γραμμών εντολών (command line window). Τα εργαλεία `javac` και `java` περιλαμβάνονται στην εγκατάσταση του `JDK` που αναφέρθηκε παραπάνω. Το `notepad` είναι διαθέσιμο σε οποιονδήποτε υπολογιστή Win/98/Me /2K/XP/Vista/7.

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 ο αριθμός Q των bits του κβαντιστή 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.

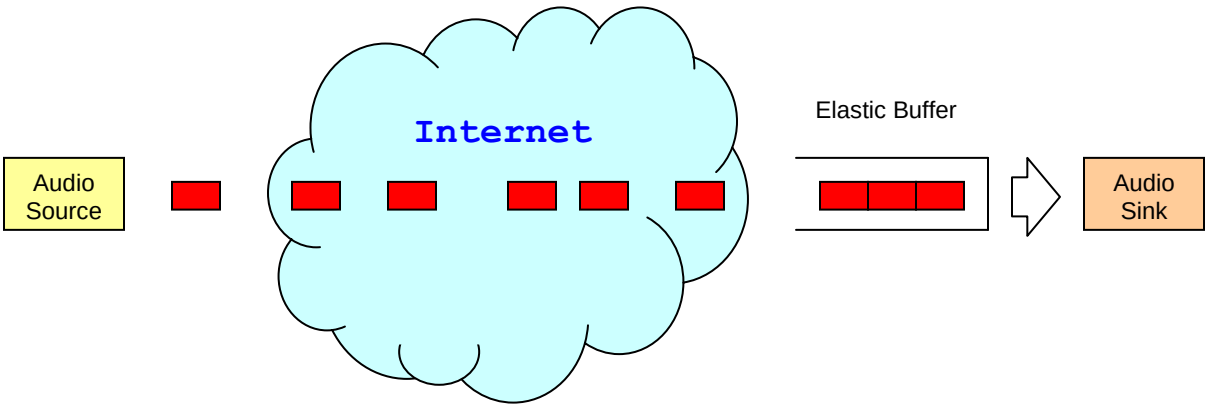
Ας σημειωθεί ότι στην Java μία μεταβλητή τύπου `byte` παριστάνει έναν προσημασμένο ακέραιο (signed integer) που λαμβάνει τιμές από -128 έως +127.

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[3] 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 όποτε 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. Real Networks 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 βρόχου 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.

[3] Ειδικότερα ας σημειωθεί ότι ενώ η μέθοδος `receive()` αναστέλλει πάντοτε την εκτέλεση του προγράμματος η μέθοδος `write()` αναστέλλει την εκτέλεση του προγράμματος ανάλογα με τις συνθήκες που επικρατούν στον εσωτερικό `buffer` του αντικειμένου εξόδου ήχου `SourceDataLine` όπως αυτές προσδιορίζονται από τη μέθοδο `available()` του ιδίου αντικειμένου. Η μέθοδος `available()` μπορεί να αξιοποιηθεί ώστε ο παραπάνω εσωτερικός `buffer` να αποτελέσει την βάση μίας απλής υλοποίησης του απαραίτητου `elastic buffer` για την ομαλή αναπαραγωγή του ήχου.

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 `serverThread`.

[2] Μεταξύ της επιλογής της εργασίας και της ενεργοποίησης της εφαρμογής `serverThread` 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 `serverThread` 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[4] 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 :

[4] Αν η σύνδεση με τον server του virtual lab γίνεται μέσω μηχανισμού NAT όπως για παράδειγμα συμβαίνει στις συνήθεις συνδέσεις ADSL, θα πρέπει να γίνει κατάλληλη ρύθμιση του μηχανισμού NAT ώστε να λειτουργεί ως virtual server. Με τον τρόπο αυτόν, πακέτα που αποστέλλονται από τον server στην εξωτερική διεύθυνση `client address` (public IP) και την θύρα `client listening port` (public port) θα προωθούνται (packet forwarding) στην εσωτερική διεύθυνση `client address` (private address) και την αντίστοιχη επίσης εσωτερική θύρα `client listening port` (private port) με τις οποίες λειτουργεί η εφαρμογή `userApplication` στο εσωτερικό τοπικό δίκτυο. Ακόμη θα πρέπει, αν απαιτείται, να γίνει κατάλληλη ρύθμιση του εσωτερικού firewall ώστε να επιτρέπεται η αμφίδρομη προώθηση UDP πακέτων (packet forwarding) μεταξύ του εσωτερικού τοπικού δικτύου (LAN) και του εξωτερικού εκτεταμένου δικτύου (WAN).

```
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=E0000`. Η αναίρεση ισχύει μόνο για το αμέσως επόμενο πακέτο `serverResponse` που ακολουθεί.

Ας σημειωθεί ότι όταν ο κωδικός `echo_request_code` συνοδεύεται από την ένδειξη `TXX` τα πακέτα που επιστρέφει ο server φέρουν ως περιεχόμενο ενδείξεις θερμοκρασιών που συλλέγονται σε σχεδόν πραγματικό χρόνο κατά μήκος του Ανατολικού Τομέα της Εγνατίας Οδού και από σημεία που βρίσκονται σε μικρή απόσταση από το οδόστρωμα. Το περιεχόμενο των πακέτων στην περίπτωση αυτήν έχει τη μορφή

```
PSTART DD-MM-YYYY HH:MM:SS TXX DD-MM HH:MM +ZZ C PSTOP
```

όπου `XX` ο κωδικός του σταθμού τηλεμετρήσεων[5] όπως εμφανίζεται στο χάρτη της επιλογής *«Θερμοκρασίες Εγνατίας Οδού»* του κεντρικού καταλόγου επιλογών του εικονικού εργαστηρίου, `+ZZ` η προσημασμένη τιμή της θερμοκρασίας και `DD-MM HH:MM` η ημερομηνία και ή ώρα καταγραφής της θερμοκρασίας από τον server του συστήματος Lightnet Digital Radio Telemetry Network στο Περιφερειακό Γραφείο Κομοτηνής της Εγνατίας Οδού.

[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/netlab/video.html και παρέχει real-time[6] video με την κίνηση της Εγνατίας Οδού στο τμήμα εμπρός από την Πολυτεχνική Σχολή. Η αποστολή του πλαισίου γίνεται τμηματικά με διαδοχικά πακέτα[7] UDP. Όλα τα πακέτα πλην του τελευταίου έχουν σταθερό μήκος. Το πλήθος των πακέτων καθώς και το μήκος του

[5] Με δεδομένο ότι οι κωδικοί των σταθμών τηλεμετρήσεων είναι 2-ψήφιοι αριθμοί, πόσοι εξ αυτών αντιστοιχούν σε σταθμούς που φέρουν αισθητήρια θερμοκρασίας και επομένως αποκρίνονται σε αιτήματα αποστολής μετρήσεων θερμοκρασιών ?

[6] Το ψηφιακό σήμα video κωδικοποιείται σύμφωνα με μία ad-hoc εκδοχή της τεχνικής Motion JPEG (MPEG) σε συνδυασμό με έναν απλό μηχανισμό (hack it!) ελέγχου ροής (flow control) που προσαρμόζεται στις συνθήκες του εκάστοτε διαύλου επικοινωνίας. Σε κάθε frame αναγράφεται η μέση εκτίμηση των πλέον πρόσφατων τιμών (moving average) του ρυθμού των frames (fps) καθώς και της ταχύτητας μετάδοσης (bps). Ποιες είναι οι εκτιμήσεις των παραμέτρων αυτών που λαμβάνετε στο τερματικό σας ?

[7] Επειδή αρκετές εργασίες αναμένεται (σ.σ. έτος 2008) να γίνουν μέσω συνδέσεων ADSL και με σκοπό την απλοποίηση του πρωτοκόλλου επικοινωνίας αλλά και των απαραίτητων ρυθμίσεων της σύνδεσης ADSL, ο ρυθμός των πακέτων UDP που στέλνει ο server μειώνεται κατά πολύ από τον ίδιο τον server ώστε η ακολουθία των πακέτων αυτών που καταφτάνει στον ADSL router και το ενσωματωμένο firewall να μην εκλαμβάνεται ως κακόβουλη επίθεση DoS attack τύπου UDP flood.

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[8] 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 συνοδεύεται από την ένδειξη `CAM=FIX` ή `CAM=PTZ` η οποία προσδιορίζει στην είσοδο του κωδικοποιητή το σήμα από αντίστοιχη κάμερα που λειτουργεί online με τον server του εικονικού εργαστηρίου. Το γεγονός αυτό έχει ως αποτέλεσμα τη λήψη μέσα από την εφαρμογή `userApplication` εικόνας από τη συγκεκριμένη κάμερα. Σε απουσία της παραμέτρου `CAM` ο server θεωρεί αυτομάτως την παράμετρο `CAM=FIX` (default value).

Σημειώνεται ότι η πρώτη κάμερα παράγει εικόνες με ανάλυση 640x480 pixels και σταθερή (`fixed`) γωνία λήψης ενώ η δεύτερη παράγει εικόνες με ανάλυση 320x240 pixels και οπτική γωνία η οποία αλλάζει σε τυχαία χρονικά διαστήματα αν εκείνη τη χρονική περίοδο η κάμερα βρίσκεται υπό τηλε-έλεγχο (`pan-tilt-zoom control`) από επισκέπτη του εικονικού εργαστηρίου μέσα από διεπαφή (user interface) που διατίθεται στο μενού επιλογών του server Ιθάκη ή απευθείας από τη διεύθυνση του `videoCoder` του εργαστηρίου http://ithaki.eng.auth.gr/netlab/video.html.

Η παράμετρος `CAM=PTZ` μπορεί να συνοδεύεται και από την παράμετρο `DIR=X` όπου `X` ένδειξη που προσδιορίζει την επιθυμητή κατεύθυνση μεταβολής της γωνίας λήψης της κάμερας σύμφωνα με το ρεπερτόριο επιλογών `L` *αριστερά*, `U` *πάνω*, `R` *δεξιά* και `D` *κάτω*. Επιπλέον οι ενδείξεις `M` *απομνημόνευση* και `C` *κέντρο* επιτρέπουν, η μεν πρώτη την απομνημόνευση της τρέχουσας γωνίας λήψης της κάμερας, η δε δεύτερη την επαναφορά της τελευταίας απομνημονευμένης γωνίας λήψης. Ας σημειωθεί ότι ο χρόνος απόκρισης της κάμερας στις επιλογές αυτές είναι μερικά δευτερόλεπτα επειδή μεσολαβεί μηχανισμός σερβοκινητήρα για την αλλαγή του προσανατολισμού της κάμερας.

Επίσης ο κωδικός `image_request_code` μπορεί να συνοδευτεί και με την ένδειξη `FLOW=ON` η οποία θέτει σε λειτουργία έναν απλό μηχανισμό ελέγχου ροής πακέτων κατά τη λήψη της εικόνας. Ο μηχανισμός υλοποιείται με την αποστολή ενός UDP πακέτου με περιεχόμενο την ένδειξη `NEXT` από την εφαρμογή `userApplication` προς τον server του εικονικού εργαστηρίου κάθε φορά που η εφαρμογή λαμβάνει τμήμα της εικόνας με τη μορφή ενός πακέτου UDP. Με τη σειρά του ο server μετά την αποστολή ενός πακέτου UDP τίθεται σε αναμονή της ένδειξης `NEXT` και προχωρά στη

[8] 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. Ωστόσο σημειώνεται ότι, με σκοπό τη διευκόλυνση της εφαρμογής κατά τη λήψη των εικόνων, ένα binary αρχείο jpeg αρχίζει με τα bytes `0xFF 0xD8` (start of image delimiter) και τελειώνει με τα bytes `0xFF 0xD9` (end of image delimiter).

μετάδοση του επόμενου πακέτου εικόνας μόλις λάβει την ένδειξη `NEXT`. Αν ο server δεν λάβει την ένδειξη `NEXT` εντός 1400 milliseconds τότε επαναλαμβάνει την εκπομπή του ιδίου πακέτου. Αν απαιτηθεί, η διαδικασία αυτή επαναλαμβάνεται συνολικά δύο φορές το πολύ (max retransmission count) σε κάθε πακέτο. Με τον τρόπο αυτόν ο server είναι σε θέση να προσαρμόσει το ρυθμό αποστολής πακέτων στο ρυθμό λήψης πακέτων από την εφαρμογή ανεξαρτήτως των συνθηκών που επικρατούν στο κανάλι μετάδοσης. Σημειώνεται ότι αν ο server διαπιστώσει την ανάγκη επανάληψης του ιδίου πακέτου για περισσότερες από δύο φορές[9], τότε, για λόγους απλοποίησης του πρωτοκόλλου επικοινωνίας, διακόπτει τη μετάδοση της τρέχουσας εικόνας και αμέσως τίθεται σε αναμονή της επόμενης εντολής `request` που πρόκειται να εκδοθεί από την εφαρμογή.

Ακόμη ο κωδικός `image_request_code` μπορεί να συνοδευτεί επιπλέον και με την ένδειξη `UDP=L` η οποία δηλώνει σε bytes το μήκος `L` του τμήματος `info` των πακέτων UDP που χρησιμοποιεί ο server για την αποστολή της εικόνας[10]. Η παράμετρος `L` μπορεί να λάβει μία από τις διακριτές τιμές `128`, `256`, `512` και `1024`. Σε απουσία της ένδειξης `UDP=L` ο server χρησιμοποιεί την τιμή `L=128`.

[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[11].

Με σκοπό τη συμπίεση της πληροφορίας, τα πακέτα ήχου κωδικοποιούνται σύμφωνα με τις αρχές της διαφορικής παλμοκωδικής διαμόρφωσης DPCM (differential pulse code modulation) η οποία μπορεί να είναι προσαρμοζόμενη (adaptive) ή μη (non adaptive).

Κατά την απλή μη προσαρμοζόμενη κωδικοποίηση, η διαφορά δύο διαδοχικών δειγμάτων ήχου x_i και x_{i-1} 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 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

[9] Ποιές συνθήκες επικοινωνίας οδηγούν κατά τη γνώμη σας στην παραπάνω διαπίστωση ?

[10] Ποιά είναι η επίδραση του μήκους `L` στην ταχύτητα (ρυθμαπόδοση) της μετάδοσης της εικόνας ?

[11] Για διευκόλυνση της εφαρμογής στα αρχικά στάδια ανάπτυξής της, η παράμετρος `LZZ` πριν από την παράμετρο `FXXX` οδηγεί τον server του εικονικού εργαστηρίου ντετερμινιστικά στην επιλογή του audio clip με αύξοντα αριθμό `ZZ`. Πόσα είναι τα πειραματικά audio clip που διαθέτει ο server σήμερα ?

Πειραματική κωδικοποίηση DPCM και AQ-DPCM

$$\mu = \frac{1}{256}\sum_{i=1}^{256} x_i \qquad (*)$$

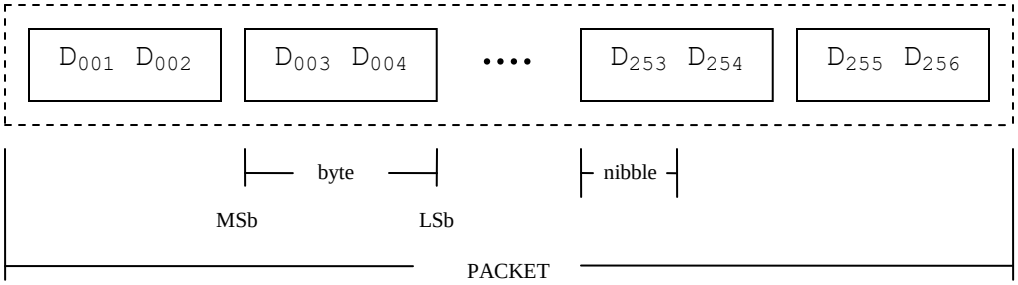
$$\Delta_i = x_i - x_{i-1} \qquad i = 1,\ldots,256$$

$$\sigma^2 = \frac{1}{256}\sum_{i=1}^{256} \Delta_i^2 \qquad \beta = f_{4bits}(\sigma) \qquad (*)$$

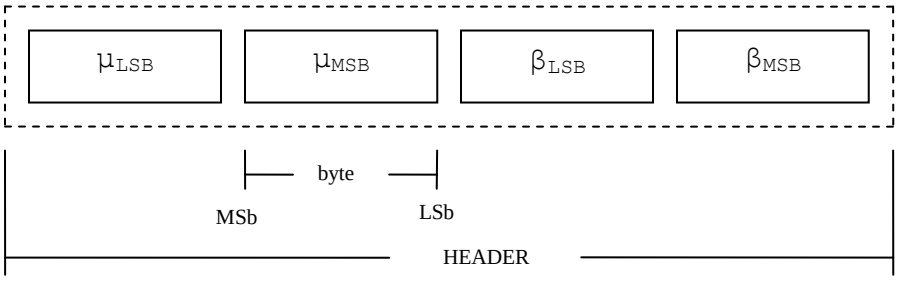
$$d_i = (\text{int})\frac{\Delta_i}{\beta} \qquad -8 \le d_i \le +7 \qquad i = 1,\ldots,256$$

$$D_i = (d_i + 8)\ \text{AND}\ 0xF \qquad 0 \le D_i \le 15 \qquad i = 1,\ldots,256$$

Πειραματική δομή πακέτου DPCM και AQ-DPCM



Πειραματική δομή επικεφαλίδας πακέτου AQ-DPCM



(*) Στην απλή κωδικοποίηση **DPCM** ο πομπός υπολογίζει τις τιμές μ και β για το σύνολο των πακέτων ήχου πριν την κωδικοποίηση και αποστολή τους. Κατά τη λήψη ο δέκτης θεωρεί μ=0 και β=1 για όλα τα πακέτα (γιατί ?) ή ακόμη β>1 όπως αναφέρεται στην υποσημείωση [12]. Στην προσαρμοζόμενη κωδικοποίηση **AQ-DPCM** ο πομπός υπολογίζει τις τιμές μ και β σε κάθε πακέτο και ενημερώνει τον δέκτη για τις τιμές αυτές μέσω της επικεφαλίδας του αντιστοίχου πακέτου. Σημειώνεται ότι η συνάρτηση f(σ) προσδιορίζει το βέλτιστο βήμα του κβαντιστή σύμφωνα με τον αλγόριθμο Max–Lloyd (optimal scalar quantisers).

ήχου 32 τέτοιων πακέτων αναπαραγόμενα[12] 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 περίπου σε 30 δευτερόλεπτα ήχου.

Σημειώνεται ότι στην περίπτωση της απλής μη προσαρμοζόμενης κωδικοποίησης DPCM ο αριθμός Q των bits του κβαντιστή που χρησιμοποιείται κατά την αναπαραγωγή του ήχου[13] από την εφαρμογή `userApplication` μπορεί να έχει την τιμή Q=8 ή Q=16.

Για λόγους πειραματισμού, ο σέρβερ του εικονικού εργαστηρίου διαθέτει επιπλέον μηχανισμό προσαρμοζόμενου κβαντιστή ο οποίος με τη σειρά του οδηγεί σε προσαρμοζόμενη κωδικοποίηση 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`.

Σημειώνεται ότι, στην περίπτωση της προσαρμοζόμενης κωδικοποίησης AQ-DPCM, ο αριθμός Q των bits του κβαντιστή που χρησιμοποιείται κατά την αναπαραγωγή του ήχου από την εφαρμογή `userApplication` θα πρέπει να έχει την τιμή Q=16 ώστε να διατηρείται η ακρίβεια των 16 bits κατά την ανασύνθεση του ήχου στον δέκτη σύμφωνα με τις τιμές των 16 bits μέσης τιμής και βήματος κβάντισης που αποστέλλει ο πομπός.

Το πρωτόκολλο TCP και ο μηχανισμός HTTP

Οι μηχανισμοί που κατ' εξοχήν χρησιμοποιούνται στη μεταφορά ροών πληροφορίας με τη μορφή μακρών ακολουθιών χαρακτήρων (character streams) ή γενικώς bytes (byte streams) στηρίζονται στο πρωτόκολλο μεταφοράς TCP (transmission control protocol). Όπως και το πρωτόκολλο UDP, το TCP ορίζει την έννοια του συνδέσμου (socket) με

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

[13] Τί διαφορές παρατηρείτε στον ήχο για Q=8 και Q=16 ? Ποιά είναι η επίδραση του συντελεστή β ?

βάση τη διεύθυνση δικτύου (IP address) του απομακρυσμένου υπολογιστή και τον αριθμό της θύρας (port number) της εφαρμογής στην οποία απευθύνεται η ροή πληροφορίας.

Η βιβλιοθήκη `java.net` παρέχει την κλάση `Socket()` η οποία διαθέτει μεθόδους κατασκευής και διαχείρισης συνδέσμων TCP. Μία από τις μεθόδους κατασκευής ενός συνδέσμου TCP δέχεται δύο ορίσματα, `inetAddress` και `portNumber`. Το πρώτο όρισμα προσδιορίζει τη διεύθυνση του απομακρυσμένου υπολογιστή με βάση τη μέθοδο `InetAddress.getByAddress(addressArray)` όπου `addressArray` είναι ένα διάνυσμα τύπου `byte[]` που περιέχει τη διεύθυνση δικτύου με συμβολισμό στο δεκαδικό σύστημα (dot-decimal notation) πχ

```
byte[] addressArray ={ (byte)155, (byte)207, 18, (byte)208 };
```

ενώ το δεύτερο όρισμα προσδιορίζει τον αριθμό της απομακρυσμένης θύρας με τη μορφή ακεραίου αριθμού πχ

```
int portNumber = 80;
```

Οι ροές εισόδου-εξόδου μιας εφαρμογής που χρησιμοποιεί το πρωτόκολλο TCP συσχετίζονται με ένα σύνδεσμο `s` μέσω αντίστοιχων μεθόδων διαχείρισης ροών ως εξής :

```
InputStream in = s.getInputSream();

OutputStream out = s.getOutputStream();
```

Είναι γνωστό ότι η συνεργασία ενός φυλλομετρητή Ιντερνετ πχ Chrome browser και ενός σερβερ πχ Apache web server στηρίζεται στο πρωτόκολλο HTTP (hyper-text transfer protocol). Με τη σειρά του, το HTTP, ως πρωτόκολλο χαρακτήρων (character based protocol) εκ κατασκευής, χρησιμοποιεί το πρωτόκολλο μεταφοράς TCP. Με βάση το γεγονός αυτό, μία μικρή πειραματική εφαρμογή που δημιουργεί ένα σύνδεσμο TCP προς τη θύρα 80 θα μπορούσε να χρησιμοποιηθεί για τη μελέτη της συμπεριφοράς[14] ενός web σέρβερ. Πράγματι, η αποστολή της ακολουθίας χαρακτήρων (HTTP request)

```
"GET /netlab/hello.html HTTP/1.0\r\nHost: ithaki.eng.auth.
gr:80\r\n\r\n"
```

με την εντολή `Java OutputStream`

```
out.write("GET /netlab/hello.html HTTP/1.0\r\nHost: ithaki.
eng.auth.gr:80\r\n\r\n".getBytes());
```

προς τον σέρβερ Ιθάκη του εικονικού εργαστηρίου του μαθήματος έχει ως αποτέλεσμα την επιστροφή από τον σέρβερ της παρακάτω ακολουθίας χαρακτήρων (HTTP response)

[14] Hacking, με την καλή έννοια ☺

```
HTTP/1.1 200 OK
Date: Wed, 13 Mar 2013 12:00:28 GMT
Server: Apache/2.4.2 (Win32) PHP/5.4.6
Last-Modified: Sun, 29 Apr 2012 17:03:57 GMT
Accept-Ranges: bytes
Content-Length: 56
Connection: close
Content-Type: text/html

<html>
<body>
Hello there !<br>
</body>
</html>
```

Η ακολουθία αυτή, εκτός από τα προθέματα (headers) του πρωτοκόλλου HTTP, περιλαμβάνει επιπλέον τον κώδικα HTML της ιστοσελίδας που θα απεικόνιζε ένας φυλλομετρητής (browser) σε κανονική λειτουργία. Η διάκριση του τμήματος των προθεμάτων από το τμήμα του κώδικα HTML γίνεται με βάση την κενή γραμμή που μεσολαβεί και δηλώνεται με τους χαρακτήρες `\r\n` που λειτουργούν ως οριοθέτες (HTTP delimiters).

Ας σημειωθεί ότι τα στοιχεία που ζητούνται κατά την παράδοση της εργασίας, όπως αναφέρεται στη συνέχεια, θα μπορούσαν να εμπλουτιστούν και με την παράθεση προθεμάτων από δύο-τρεις δημόσιους servers με ελεύθερη επιλογή από το Ιντερνετ.

Ο τηλεχειρισμός της ιπτάμενης μικρο-πλατφόρμας Ithakicopter

Συμπληρωματικά, τα πειράματα με το πρωτόκολλο TCP θα μπορούσαν περιλάβουν και την καταγραφή τηλε-μετρήσεων σχετικά με την τρέχουσα θέση και την ισχύ των μικρο-κινητήρων της ιπτάμενης μικρο-πλατφόρμας *Ithakicopter* που παρουσιάζεται στο URL http://ithaki.eng.auth.gr/netlab/ithakicopter.html.

Πράγματι, η ροή εξόδου από τον σέρβερ Ιθάκη ενός συνδέσμου TCP προς τη θύρα `38048`, μετά την αποστολή ενός αρχικού μηνύματος[15] (*preamble text*), παρέχει τέτοιες τηλε-μετρήσεις με τη μορφή

```
ITHAKICOPTER LMOTOR=LLL RMOTOR=RRR ALTITUDE=AAA ~

                ~ TEMPERATURE=+TT.TT PRESSURE=PPPP.PPTELEMETRY \r\n
```

όπου `LLL` και `RRR` 3-ψήφιοι αριθμοί στο 10-δικό σύστημα που δηλώνουν με ακρίβεια 8 bits τις τρέχουσες τιμές του duty-cycle των κυματομορφών PWM που ελέγχουν τις στροφές των δύο μικρο-κινητήρων της ιπτάμενης μικρο-πλατφόρμας *Ithakicopter*. Η τιμή `AAA` είναι επίσης 3-ψήφιος αριθμός στο 10-δικό σύστημα που δηλώνει την τρέχουσα θέση της μικρο-πλατφόρμας μετρούμενη σε αριθμό pixels από το επίπεδο αναφοράς `Ground Level` όπως αυτό εμφανίζεται στα frames του video πραγματικού χρόνου που

[15] Μεταβείτε με τον φυλλομετρητή σας στη διεύθυνση http://ithaki.eng.auth.gr:38048. Τι παρατηρείτε ?

παρέχεται από το URL http://ithaki.eng.auth.gr/netlab/ithakicopter.html. Οι τιμές `TT.TT` και `PPPP.PP` επίσης στο 10-δικό σύστημα δηλώνουν τις τρέχουσες τιμές της θερμοκρασίας και της ατμοσφαιρικής πίεσης στο περιβάλλον της ιπτάμενης μικρο-πλατφόρμας *Ithakicopter*.

Σημειώνεται ότι για λόγους συγχρονισμού αλλά και διαχείρισης της καθυστέρησης δικτύου η αποστολή των παραπάνω τηλε-μετρήσεων αποτελεί απόκριση (response) του σερβερ Ιθάκη μετά από αντίστοιχο αίτημα (request) το οποίο απαραιτήτως προηγείται. Για τον σκοπό αυτόν, μια εφαρμογή απομακρυσμένου ελέγχου `AUTOPILOT` θα πρέπει να αποστέλλει προς τον σερβερ Ιθάκη, σε κατάλληλα χρονικά διαστήματα μέσω της ροής εισόδου του ιδίου συνδέσμου TCP προς τη θύρα `38048`, αιτήματα της μορφής

```
AUTO FLIGHTLEVEL=FFF LMOTOR=LLL RMOTOR=RRR PILOT \r\n
```

όπου `FFF` δηλώνει το επιθυμητό επίπεδο πτήσης και `LLL` και `RRR` τις επιθυμητές τιμές `LMOTOR` και `RMOTOR`. Διευκρινίζεται ότι η τιμή `FFF` χρησιμοποιείται μόνο για λόγους απεικόνισης πάνω στο video κίνησης της μικρο-πλατφόρμας ενώ οι τιμές `LLL` και `RRR` χρησιμοποιούνται από το ίδιο το σύστημα για την καθοδήγηση της ιπτάμενης μικρο-πλατφόρμας *Ithakicopter*. Το γεγονός αυτό σε συνδυασμό με τη λήψη των ενδείξεων `ALTITUDE` μπορεί να οδηγήσει στην ανάπτυξη ενός πλήρους δικτυακού βρόχου ελέγχου[16] (networked control loop) της ιπτάμενης μικρο-πλατφόρμας *Ithakicopter* εξ αποστάσεως μέσω Ιντερνετ. Επαναλαμβάνοντας την ίδια διαδικασία *request-response*, η εφαρμογή `AUTOPILOT` μπορεί να ελέγξει εξ αποστάσεως τη μικρο-πλατφόρμα μέχρι 180 δευτερόλεπτα τα οποία αντιστοιχούν στο μέγιστο επιτρεπτό χρόνο αδιάλειπτης λειτουργίας του μηχανισμού αυτομάτου ελέγχου (single control session).

Εναλλακτικά σε μια απλή εφαρμογή `TELEMETRY` παρέχεται η δυνατότητα μόνο λήψης των τρεχουσών τιμών `LMOTOR`, `RMOTOR` και `ALTITUDE` με τη μορφή

```
ITHAKICOPTER LMOTOR=LLL RMOTOR=RRR ALTITUDE= AAA ~

                ~ TEMPERATURE=+TT.TT PRESSURE=PPPP.PPTELEMETRY \r\n
```

μέσω πακέτων `UDP` στη θύρα `48078` (listening port) παράλληλα με την λειτουργία της εφαρμογής «*Java JNLP application based Ithakicopter*» η οποία διατίθεται στην σελίδα http://ithaki.eng.auth.gr/netlab/ithakicopter.html υπό την προϋπόθεση ότι οι δύο εφαρμογές μοιράζονται την ίδια δικτυακή διεύθυνση. Σημειώνεται ότι, στην περίπτωση που η σύνδεση Ιντερνετ παρέχεται μέσω μηχανισμού CGN (Carrier Grade NAT), ειδική μέριμνα θα πρέπει να ληφθεί ώστε να επιτραπεί η διέλευση των telemetry UDP packets διαμέσου του μηχανισμού CGN (CGN traversal ή NAT hole punching) με την αποστολή ενός αρχικού πακέτου `UDP` (γιατί ?) προς την θύρα `48078` του σερβερ *Ithaki*.

[16] Θα μπορούσε η εφαρμογή *remote control* να λειτουργήσει σε περιβάλλον *smart phone* λαμβάνοντας τις εντολές εισόδου από τα αδρανειακά αισθητήρια της συσκευής ?

Συλλογή διαγνωστικών στοιχείων αναφορικά με την λειτουργία οχήματος

Επιπλέον, τα πειράματα με το πρωτόκολλο TCP θα μπορούσαν περιλάβουν και τη συλλογή στοιχείων σχετικά με την λειτουργία οχήματος μέσω του μηχανισμού OBD-II *(Onboard Diagnostics II)* σύμφωνα με το διεθνές πρότυπο SAE J1979. Το πρότυπο αυτό επιτρέπει την αποστολή σειράς κωδικών request προς μία ηλεκτρονική μονάδα ελέγχου *(electronic controller unit ECU)* που ελέγχει συγκεκριμένο υποσύστημα του οχήματος και στη συνέχεια την λήψη σειράς κωδικών response που μεταφέρουν στοιχεία για την λειτουργική κατάσταση του αντιστοίχου υποσυστήματος του οχήματος. Η επικοινωνία των μονάδων ECU όλων των υποσυστημάτων του οχήματος γίνεται με την βοήθεια τοπικού δικτύου επί του οχήματος (controller area network CAN) που ακολουθεί το διεθνές πρότυπο ISO 15765-4.

Ο σερβερ Ιθάκη διαθέτει πηγαίες εγγραφές πραγματικών στοιχείων OBD-II (real world source data) τα οποία καλύπτουν περισσότερο από μια ώρα λειτουργίας ενός οχήματος σε στάση ή σε κίνηση. Η λήψη των στοιχείων αυτών μπορεί να γίνει μέσω TCP socket[17] στη θύρα 29078. Ο πίνακας που ακολουθεί παρουσιάζει ένα μικρό μέρος των κωδικών OBD-II που μπορεί να χρησιμοποιηθούν για πειραματικές εφαρμογές σε συνεργασία με τον σερβερ Ιθάκη. Η αίτηση request περιλαμβάνει δύο παραμέτρους, η καθεμία με τη μορφή 2-ψήφιου 16-δικού αριθμού, που αφορούν στις τιμές MODE και PID (parameter ID) ανάλογα με τα στοιχεία που ζητούνται. Το πρωτόκολλο προβλέπει ένα κενό `SP`

OBD-II request	OBD-II response

Mode	PID	Description	1st byte	2nd byte	Units	Formula
01	1F	Engine run time	XX	YY	sec	256*XX+YY
01	0F	Intake air temperature	XX		°C	XX-40
01	11	Throttle position	XX		%	XX*100/255
01	0C	Engine RPM	XX	YY	RPM	((XX*256)+YY)/4
01	0D	Vehicle speed	XX		Km/h	XX
01	05	Coolant temperature	XX		°C	XX-40

`(space)` μεταξύ των δύο παραμέτρων και τον χαρακτήρα `CR` (`ASCII 13`) στο τέλος ως οριοθέτη. Η απόκριση response επαναλαβάνει τις δύο παραμέτρους MODE και PID επισυνάπτοντας κατά περίπτωση είτε μία μόνο 16-δική τιμή XX είτε δύο 16-δικές τιμές XX και YY. Οι τιμές αυτές κωδικοποιούν τα στοιχεία των επιμέρους λειτουργιών του οχήματος που ζητούνται σύμφωνα με τους τύπους του παραπάνω πίνακα. Ας σημειωθεί ότι κατά την απόκριση (α) το πλέον σημαντικό ψηφίο (most significant nimble) της παραμέτρου MODE αντικαθίσταται με την τιμή 0x4, (β) μεταξύ των διαδοχικών τιμών

[17] Εναλλακτικά μπορεί να χρησιμοποιηθεί και το πρωτόκολλο `UDP` με την αποστολή του κωδικού OBD-II request συνοδευόμενου από την παράμετρο `OBD=MODE PID` σύμφωνα με τον πίνακα OBD-II. Σημειώνεται ότι ο οριοθέτης `CR` ή `\r` δεν είναι απαραίτητος στο τέλος της παραμέτρου OBD κατά την επικοινωνία με το πρωτόκολλο `UDP`.

παρεμβάλλεται επίσης ένα κενό `SP` και (γ) στο τέλος προστίθεται ο χαρακτήρας `CR` και πάλι ως οριοθέτης (delimiter).

Συλλογή μετρήσεων και παρουσίαση αποτελεσμάτων

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

[Β] 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 διαγραμμάτων `G3` και `G4` αντίστοιχα των περιπτώσεων (i) και (ii) παραπάνω με απενεργοποιημένη την καθυστέρηση που παρεμβάλει ο server, **(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 και (**v**) ενός διαγράμματος `R1` το οποίο, θεωρώντας τις μετρήσεις (i) ως στιγμιαίες τιμές RTT (round trip time), με υπέρθεση παρουσιάζει τις τιμές των παραμέτρων SRTT (smooth round trip time, σ_{RTT} (round trip time variance) και RTO (retransmission timeout), με συντελεστές βαρύτητας α,β,γ δικής σας επιλογής, όπως θα τις προσδιόριζε μια δική σας υλοποίηση του λογισμικού TCP στην πλευρά του δικού σας τερματικού.

[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)** τις τιμές των συχνοτήτων και τους τίτλους[18] των audio clip που προέρχονται από την εικονική γεννήτρια συχνοτήτων και το πειραματικό ρεπερτόριο μουσικής του server του εικονικού εργαστηρίου αντίστοιχα.

[Δ] Επίσης ζητείται η παρουσίαση **(α)** δύο εικόνων `E1` και `E2` από τις κάμερες `CAM1` και `CAM2` αντίστοιχα, **(β)** οκτώ τιμών θερμοκρασιών `T1...T8` από αντίστοιχους σταθμούς τηλεμετρήσεων της Εγνατίας Οδού, **(γ)** δύο διαγραμμάτων `G9` και `G10` τμημάτων κυματομορφών που προέρχονται αντίστοιχα από την εικονική γεννήτρια συχνοτήτων και το πειραματικό ρεπερτόριο μουσικής του server του εικονικού εργαστηρίου, **(δ)** τεσσάρων διαγραμμάτων `G11`, `G12`, `G13` και `G14` κατανομών των τιμών των διαφορών[19] αλλά και των τιμών των ίδιων των δειγμάτων των κυματομορφών ήχου που αναπαράγονται κατά την αποκωδικοποίηση σημάτων DPCM και AQ-DPCM[20], **(ε)** τεσσάρων διαγραμμάτων `G15`, `G16`, `G17` και `G18` τμημάτων των ακολουθιών τιμών της

[18] Η εφαρμογή αναγνώρισης μουσικών τίτλων TrackID (Google it!) μπορεί να βοηθήσει στη γεφύρωση του «μουσικού χάσματος γενεών» που ενδεχομένως δημιουργεί το πειραματικό ρεπερτόριο της Ιθάκης ☺

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

[20] Η κωδικοποίηση AQ-DPCM εφαρμόζεται μόνο στις κυματομορφές που προέρχονται από το πειραματικό ρεπερτόριο μουσικής του server του εικονικού εργαστηρίου.

μέσης τιμής και του βήματος του προσαρμοζόμενου κβαντιστή που λαμβάνονται κατά τη διάρκεια λήψης σήματος AQ-DPCM από δύο audio clip του πειραματικού ρεπερτορίου μουσικής του server του εικονικού εργαστηρίου, **(στ)** δύο διαγραμμάτων `G19` και `G20` τηλε-μετρήσεων που λαμβάνονται κατά τη διάρκεια της κίνησης της μικρο-πλατφόρμας *Ithakicopter* μέσω του μηχανισμού[21] *udp telemetry* για δύο αντίστοιχα επιθυμητά επίπεδα *flight level* και τέλος **(ζ)** πέντε διακεκριμένα διαγράμματα ή ένα διάγραμμα με υπέρθεση τις επί μέρους καμπύλες αναφορικά με τις παραμέτρους του πίνακα OBD-II για τέσσερα (οποιαδήποτε συνεχόμενα) λεπτά λειτουργίας του οχήματος.

Σημειώνεται ότι, επειδή οι τεχνικές ανάπτυξης πολυ-νηματικών εφαρμογών όπως αναφέρθηκε νωρίτερα δεν περιλαμβάνονται στις απαιτήσεις της εργασίας αυτής[22], η αναπαραγωγή του ήχου μπορεί να γίνει σε δύο ανεξάρτητα διαδοχικά στάδια. Στο πρώτο στάδιο λαμβάνονται όλα τα πακέτα ήχου που φτάνουν σε τυχαία χρονικά διαστήματα και αποθηκεύονται[23] ακολουθιακά στην ελαστική μνήμη του δέκτη. Στο δεύτερο στάδιο η ελαστική μνήμη τροφοδοτεί σε τακτά χρονικά διαστήματα[24] 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. Εναλλακτικά μπορεί να υιοθετηθεί η απλή υλοποίηση με βάση τη μέθοδο `available()` όπως αναφέρεται στην υποσημείωση [3].

[Ε] Τα παραπάνω αποτελέσματα ζητείται να συνοδεύονται από **(α)** σύντομα σχόλια ή παρατηρήσεις σας, **(β)** μία σύντομη βιβλιογραφική τεχνική αναφορά στο πρωτόκολλο UDP, **(γ)** μία επίσης σύντομη βιβλιογραφική τεχνική αναφορά σε διεθνή πρότυπα audio streaming, **(δ)** μία ή περισσότερες εκτυπώσεις (screen shots) των ρυθμίσεων `IP`, `DNS`, `DHCP`, `NAT` και `PORT` της διάταξης ADSL που ενδεχομένως έχετε χρησιμοποιήσει στην εργασία για την επικοινωνία με τον σέρβερ Ιθάκη του εικονικού εργαστηρίου και **(ε)** για καθεμία εφαρμογή της εργασίας, μια ενδεικτική εκτύπωση (filtered screen shot) του live packet capture display όπως εμφανίζεται σε real-time με την βοήθεια του εργαλείου *wireshark* (https://www.wireshark.org/download.html) καθώς η αντίστοιχη εφαρμογή Java τρέχει και επικοινωνεί με την Ιθάκη. Σημειώνεται ότι σε κάθε εκτύπωση κάθε εφαρμογής του σημείου (ε) προηγείται ένας μικρός αριθμός echo packets από την εφαρμογή echo έτσι ώστε να δίνεται με σαφώς ευδιάκριτο και ευανάγνωστο τρόπο πλήρης χρονοσήμανση στις ζητούμενες εκτυπώσεις.

[Ζ] Τα αποτελέσματα μαζί με τα σχόλιά σας, την τεχνική αναφορά και τον πηγαίο κώδικα της εφαρμογής ζητείται να υποβληθούν με τη μορφή αρχείων PDF και τις εξής απαραίτητα προδιαγραφές μορφοποίησης : **(i)** οι εικόνες `E1` και `E2`, οι τιμές θερμοκρασιών

[21] Εναλλακτικά οι ενδιαφερόμενοι/τριες φοιτητές/τριες μπορούν να αναπτύξουν τη δική τους εφαρμογή *tcp/ip autopilot*. Σε μια τέτοια περίπτωση, με σημείο εκκίνησης τη θέση ηρεμίας, ο μηχανισμός *autopilot* που θα παρουσιάσει την ελάχιστη μέση τετραγωνική απόκλιση από ένα επιθυμητό *low flight level* και στη συνέχεια από ένα *high flight level* για χρονικό διάστημα μεγαλύτερο από ένα λεπτό της ώρας, σε on-line λειτουργία μέσα στην τάξη, θα προσθέσει τιμητικά έναν επιπλέον βαθμό στην αντίστοιχη εργασία ☺

[22] Στο τέλος του παρόντος κειμένου παρατίθενται ενδεικτικά δύο εκδόσεις μιας εφαρμογής Java (template) που μπορεί να χρησιμοποιηθούν πειραματικά σε εφαρμογές πολυ-νηματικού προγραμματισμού.

[23] Πόσο διαρκεί η ολοκλήρωση αυτού του σταδίου ?

[24] Πώς εξασφαλίζεται ο συγχρονισμός στο σημείο αυτό ?

`T1...T8` καθώς και τα γραφήματα `G1` έως `G20` μαζί με τα στοιχεία της παραγράφου [Δ] θα αποτελούν ένα διακριτό αρχείο με όνομα *session1.pdf* για την πρώτη σύνοδο και ένα δεύτερο διακριτό αρχείο με όνομα *session2.pdf* για τη δεύτερη σύνοδο, **(ii)** στον τίτλο κάθε εικόνας καθώς και κάθε γραφήματος θα φαίνονται απαραίτητα η ημέρα και η ώρα 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 με όνομα *report.pdf* ενώ τα στοιχεία των σημείων (δ) και (ε) θα ενσωματωθούν σε δύο επιπλέον διακριτά αρχεία με ονόματα *router.pdf* και *wireshark.pdf* αντίστοιχα, **(iv)** ο πηγαίος κώδικας της εφαρμογής Java θα αποτελεί ένα τελευταίο διακριτό αρχείο με όνομα *source.pdf* και **(v)** όλες οι σελίδες των αρχείων *session1.pdf, session2.pdf, report.pdf, router.pdf, wireshark.pdf* και *source.pdf* θα φέρουν κεφαλίδα με την ένδειξη *Δίκτυα Υπολογιστών 2* και το ονοματεπώνυμο με το ΑΕΜ του/της φοιτητή/τριας που εκπόνησε[25] την εργασία.

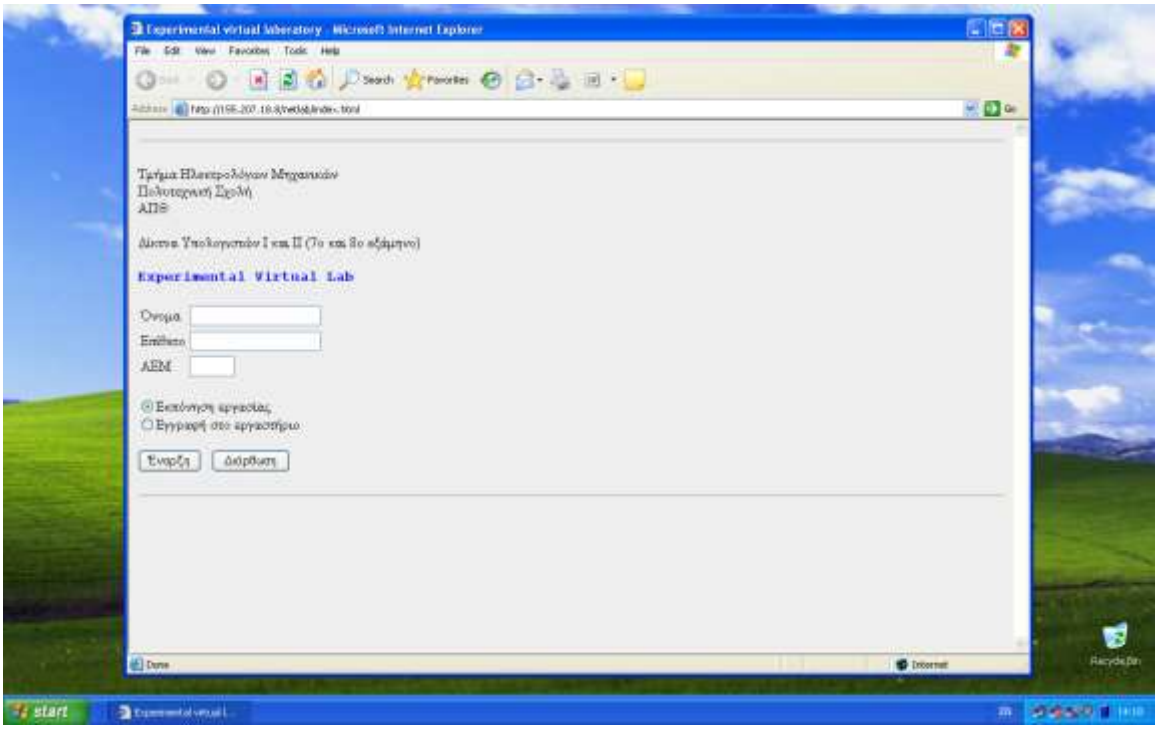
Τα παραπάνω αρχεία PDF θα υποβληθούν μέσω της επιλογής μεταφόρτωσης αρχείων *(file upload)* του σέρβερ Ιθάκη στον κοινό χώρο εναπόθεσης εργασιών *(dropbox[26])* του εικονικού εργαστηρίου.

[Η] Η διεύθυνση επικοινωνίας με τον διδάσκοντα για το εικονικό εργαστήριο και το μάθημα Δίκτυα Υπολογιστών 2 (7ο εξάμηνο) είναι ~~mitrakos@eng.auth.gr~~ dimitris.mitrakos@otenet.gr

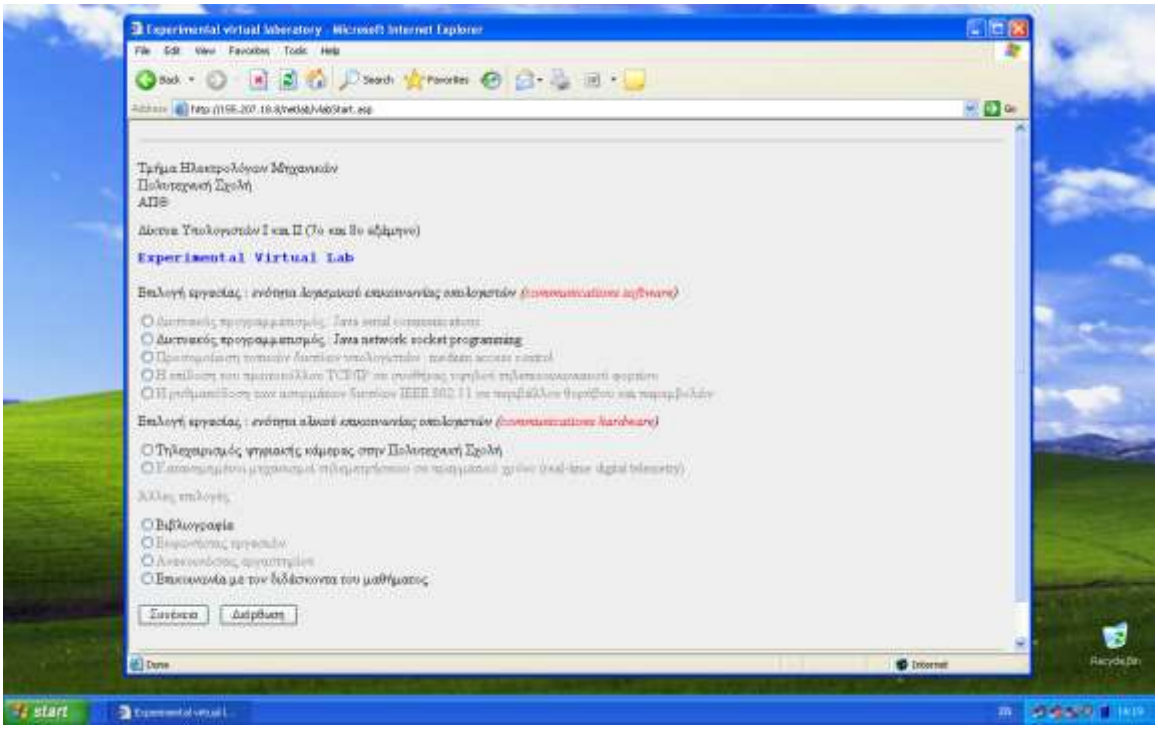
[Θ] Enjoy !

[25] Όχι το ονοματεπώνυμο και το ΑΕΜ του/της φοιτητή/τριας που υποβάλει την εργασία ☺

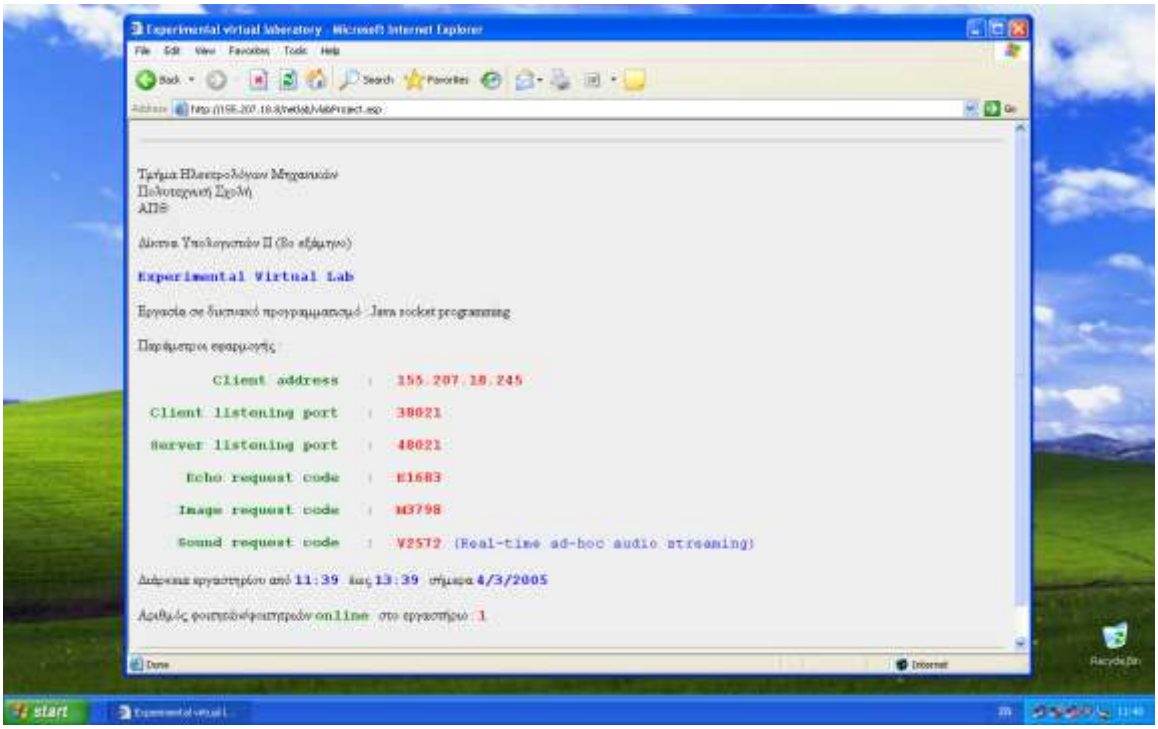
[26] Για λόγους ασφαλείας το μέγιστο μέγεθος κάθε αρχείου περιορίζεται στα 2 Mbytes ενώ για παρόμοιους λόγους ο μέγιστος αριθμός αρχείων περιορίζεται στα 8 αρχεία ανά ΑΕΜ.



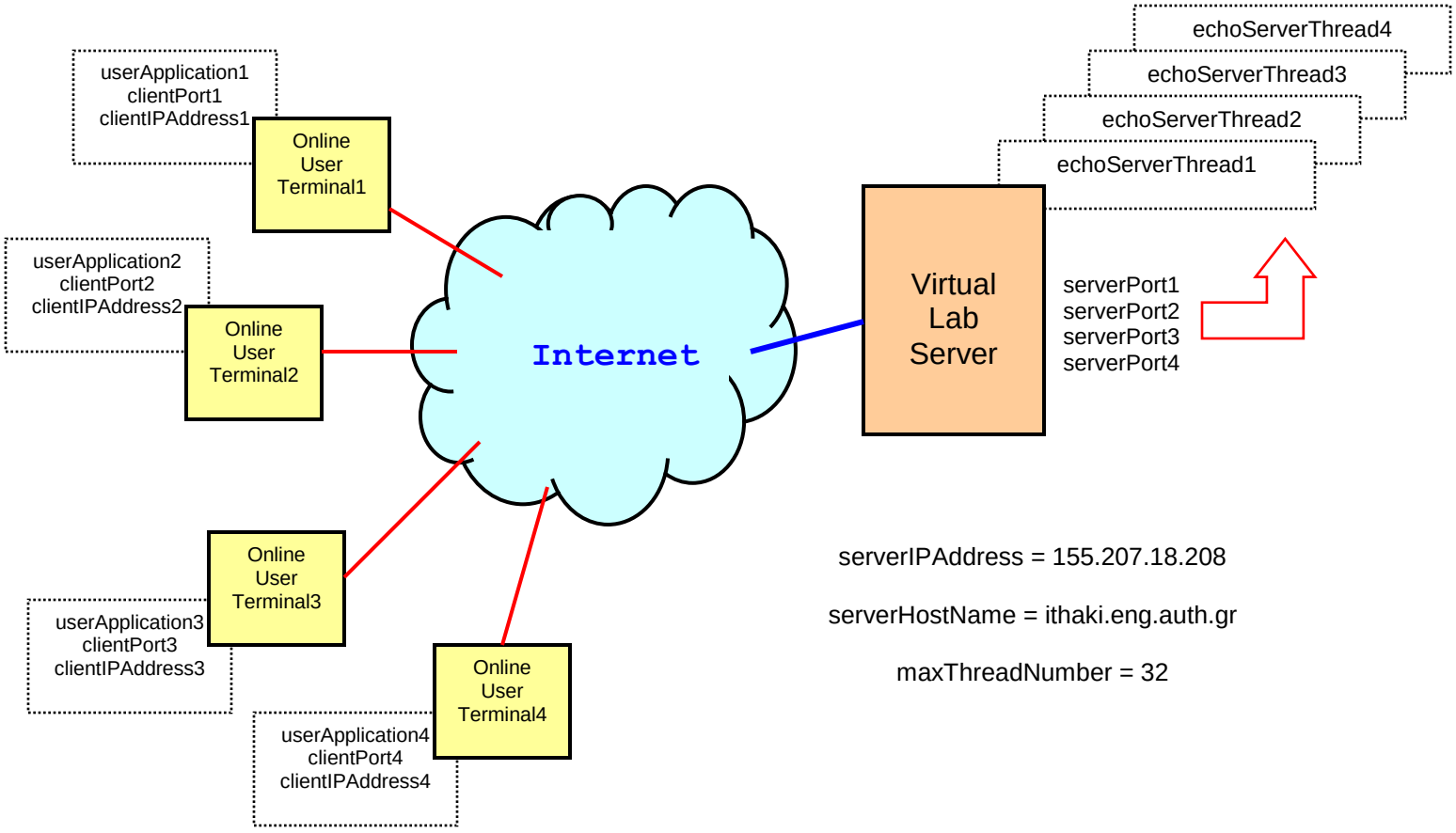
Εικόνα 1 : Η είσοδος στο εικονικό εργαστήριο http://ithaki.eng.auth.gr/netlab/index.html



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

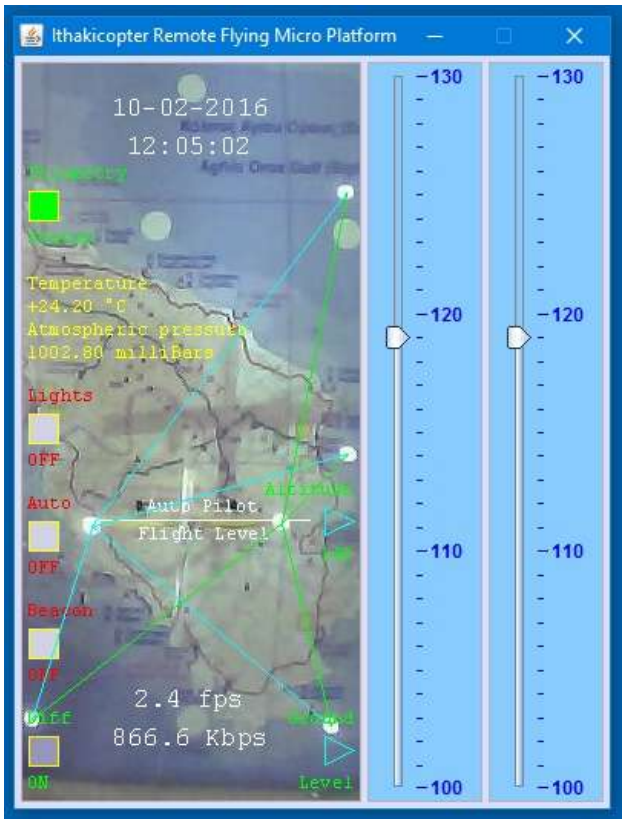


Εικόνα 3 : Ενδεικτικές τιμές των προτεινομένων παραμέτρων της εφαρμογής κατά τη διάρκεια μίας συνόδου στο εικονικό εργαστήριο



Σχήμα 1 : Συνοπτικό διάγραμμα επικοινωνιών στο εικονικό εργαστήριο

http://ithaki.eng.auth.gr/netlab/ithakicopter.html



```
/*
/* Local NAT virtual server - No ISP carrier grade NAT (CGN)
/*

DatagramSocket s = new DatagramSocket();

String packetInfo = "Hello there !"

byte[] txbuffer = packetInfo.getBytes();

int serverPort = 38004;

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 = 48004;

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();
```

```
/*
/* ISP carrier grade NAT (CGN) - No local NAT virtual server
/*

int clientPort = 48008;

InetAddress clientAddress = InetAddress.getLocalHost();

DatagramSocket s = new DatagramSocket(clientPort,clientAddress);

String packetInfo = "Hello there !"

byte[] txbuffer = packetInfo.getBytes();

int serverPort = 38008;

InetAddress hostAddress = InetAddress.getByName("155.207.18.208");

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

s.setSoTimeout(800);

byte[] rxbuffer = new byte[2048];

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

for (;;) {
  try {
    s.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();
```

```
/*
/* TCP socket input and output streams
/*

Socket s;

InetAddress serverAddress;

InetSocketAddress serverSocket;

InputStream in;

OutputStream out;

serverAddress=InetAddress.getByName("ithaki.eng.auth.gr");

serverSocket=new InetSocketAddress(serverAddress,80);

s=new Socket();

s.setSoTimeout(1800);

s.connect(serverSocket,2800);

in=s.getInputStream();

out=s.getOutputStream();

out.write("GET /netlab/hello.html HTTP/1.0\r\n\r\n".getBytes());

for (;;) {

  k=in.read();

  if (k==-1) break;

  response+=(char)k;

}
```

```
public class multiThreadApp {

  Thread A;

  Thread B;

  public multiThreadApp() {
    A = new alpha();
    B = new beta();
  }

  public static void main(String[] args) {
    multiThreadApp M = new multiThreadApp();
  }

  public class alpha extends Thread {

    public alpha() {
      this.start();
    }

    public void run() {
      while(true) {
        System.out.println("A");
        try{
          this.sleep(1000);
        } catch (Exception x) {
        }
      }
    }

  }

  public class beta extends Thread {

    public beta() {
      this.start();
    }

    public void run() {
      while(true) {
        System.out.println("B");
        try{
          this.sleep(2000);
        } catch (Exception x) {
        }
      }
    }

  }

}
```

```
public class multiThreadApp {

  public multiThreadApp() {
    alphaThread A = new alphaThread();
    betaThread B = new betaThread();
  }

  public static void main(String[] args) {
    multiThreadApp M = new multiThreadApp();
  }

  public class alphaThread implements Runnable {

    Thread alpha = null;

    public alphaThread() {
      alpha = new Thread(this);
      alpha.start();
    }

    public void run() {
      while(true) {
        System.out.println("A");
        try{
          alpha.sleep(1000);
        } catch (Exception x) {
        }
      }
    }

  }

  public class betaThread implements Runnable {

    Thread beta = null;

    public betaThread() {
      beta = new Thread(this);
      beta.start();
    }

    public void run() {
      while(true) {
        System.out.println("B");
        try{
          beta.sleep(2000);
        } catch (Exception x) {
        }
      }
    }

  }

}
```