ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δίκτυα Υπολογιστών Ι

Εργασία δικτυακού υπολογισμού σε Java

Παναγιώτης Σαββίδης

8094

ΘΕΣΣΑΛΟΝΙΚΗ 2020

ΠΕΡΙΕΧΟΜΕΝΑ:

Εισαγωγικά

Αρχεία:

Στην εργασία περιλαμβάνονται 4 αρχεία pdf.

Το πρώτο ονομάζεται **report.pdf** και περιέχει τη συνολική περιγραφή του κώδικα και τη λειτουργία όλων των τμημάτων.

Το δεύτερο και τρίτο pdf ονομάζονται **session1.pdf** και **session2.pdf** και περιλαμβάνουν τις δυο επικοινωνίες με το τερματικό της Ιθάκης και μια αναλυτική παρουσίαση όλων των στοιχείων που παραλείφθηκαν από αυτή.

Τέλος, το **source.pdf** έχει τον κώδικα που χρησιμοποιήθηκε για τη σύνδεση με το modem της Ιθάκης.

Για την εργασία αυτή δημιουργήθηκε η εφαρμογή *userApplication* σε Java και χρησιμοποιήθηκε η κλάση modem.class που υλοποιεί τη σύνδεση με το modem της Ιθάκης. Για τη λειτουργία επιλέγουμε modem speed = 80000 και modem timeout = 2000.

Κώδικας Java

EchoPackets

Χρησιμοποιούμε τον κωδικό E9141 και λαμβάνουμε πακέτα που αποστέλνονται από την Ιθάκη. Σε κάθε κύκλο αποστολής/παραλαβής πακέτων λαμβάνουμε πληροφορίες για το χρόνο απόκρισης του συστήματος. Όπως γνωρίζουμε, χρόνος απόκρισης του συστήματος ορίζεται ως ο χρόνος που θα μεσολαβήσει από τη στιγμή που στέλνεται το αίτημα για την παραλαβή πακέτου μέχρι τη στιγμή που γίνεται η παραλαβή από τον χρήστη.

Τα πακέτα που θα παραλάβουμε θα αποθηκευτούν σε ένα FileOutputStream.

```
FileOutputStream fout=null;
fout=new FileOutputStream(”text.txt”);
```

Έχουμε μια for που τρέχει από το 1 μέχρι το 200 και έτσι καλούμε τη συνάρτηση modem.write `modem.write("E9141\r".getBytes());` που θα μας επιστρέψει έναν αριθμό (όχι μονάδα) για όσο λαμβάνει δεδομένα.

Ο χρόνο αποστολής αυτής της αίτησης πακέτων θα αποθηκευτεί σε μια μεταβλητή long ως εξής:
`long sTime=System.currentTimeMillis();`

Όταν λάβουμε το -1 θα σημειωθεί ο τρέχων χρόνος λήψης και στη συνέχεια θα αποθηκεύσουμε στο αρχείο fout τη διαφορά ανάμεσα στους δύο χρόνους. Τέλος, θα μετατρέψουμε τον χρόνο σε String μεταβλητή και μετά σε πίνακα byte.

ImagePackets

Η ImagePackets είναι η μέθοδος που θα μας βοηθήσει στην παραλαβή της εικόνας μέσω της κάμερας που είναι εγκατεστημένη στο Server του εικονικού εργαστηρίου της Ιθάκης. Με την αποστολή του κωδικού M2488 θα λάβουμε από την Ιθάκη την εικόνα την οποία και θα αποθηκεύσουμε. Η εικόνα αυτή θα αποθηκευτεί σε ένα binary αρχείο, όπως φαίνεται παρακάτω, και θα μεταφερθεί στη συνέχεια σε έναν πίνακα από byte, αφού γίνει η λήψη όλων των στοιχείων.

```
ByteArrayOutputStream array=new ByteArrayOutputStream();

byte [] byteArray = array.toByteArray();
```

Αφού οριστούν οι περιοχές της εικόνας θα δημιουργήσω ένα αρχείο jpeg και θα περάσω όλα τα δεδομένα εκεί, ώστε να δημιουργηθεί το τελικό απαιτούμενο αρχείο.

```
FileOutputStream fout=new FileOutputStream("image.jpeg");
        for (int i=0; i<byteArray.length; i++){
                fout.write(byteArray[i]);
                }
```

Λήψη δεδομένων GPS εικόνων.

Για την παραλαβή των εικόνων του GPS θα πρέπει να μας στείλει ο server της Ιθάκης ριπές πακέτων για να μπορέσουμε να μελετήσουμε τη ρυθμαπόδοση του συστήματος. Ρυθμαπόδοση είναι ο λόγος του αριθμού των bits που θα μας στείλει η Ιθάκη σε μια ριπή, προς τον χρόνο που θα μεσολαβήσει από τη στιγμή που θα αποσταλεί το αίτημά μας μέχρι τη στιγμή που θα ολοκληρωθεί η διαδικασία.

Το modem θα στείλει πακέτα GPS με τη μορφή προτάσεων. Οι τίτλοι των προτάσεων αυτών θα έχουν κάποια από τις παρακάτω μορφές: GPGSV, GPGSA, GPGGA, GPRMC. Οι πρώτες δύο δεν περιέχουν καμία πληροφορία σχετικά με το γεωγραφικό μήκος και γεωγραφικό πλάτος, πληροφορίες που υπάρχουν αποκλειστικά στις GPGGA και GPRMC.

Με τη χρήση του gps_request_code R=XXPPPPLL θα μπορέσουμε να λάβουμε μόνο την πρόταση GPGSA.

Με σκοπό την αποθήκευση των όσων έλαβα από την Ιθάκη θα δημιουργήσω ξανά ένα ByteOutputStream και στη συνέχεια θα το μετατρέψω σε string, ώστε να είναι δυνατή η επεξεργασία του.

```
ByteArrayOutputStream array=new ByteArrayOutputStream();

String gpsPackets=array.toString();
```

Θα χρειαστώ ένα Buffered Reader για να καταφέρω να προσπελάσω τα πακέτα ανά γραμμή.

```
BufferedReader reader = new BufferedReader(new
        StringReader(gpsPackets));
```

Από τα 200 GPS πακέτα που θα ληφθούν, θα χρειαστεί να αποθηκεύσω μόνο τα εννέα που θα διαφέρουν μεταξύ τους χρόνο μεγαλύτερο και ίσο από 4 δευτερόλεπτα ($t \geq 4$).

```
String tStrings[]=new String[9];
```

Τέλος, θα χρησιμοποιήσω κάποιες από τις πληροφορίες που έλαβα από το τερματικό της Ιθάκης για να δημιουργήσω τις παραμέτρους Τ.

Αυτές αντιστοιχούν σε συντεταγμένες των σημείων που επιθυμώ να λάβω εικόνα από το Google Maps.

Με την χρήση του

```
String  fRequest= ”P5007T=” + tStrings[0] + ”T=” + tStrings[1] +
”T=”+tStrings[2]+”T=”+tStrings[3]+”T=”+tStrings[4]+”T=”+tStrings[5]
    +”T=”+tStrings[6]+”T=”+tStrings[7]+”T=”+tStrings[8]+”\r”;
```

θα στείλω τον κωδικό μου, μαζί με τις παραμέτρους T και θα λάβω σαν απάντηση την εικόνα που επιθυμώ.

Εφαρμογή του μηχανισμού ARQ

Automatic Repeat Request, ή σε συντόμευση ARQ, είναι μια από τις μεθόδους ελέγχου για πιθανά σφάλματα. Ο τρόπους λειτουργίας του μπορεί να περιγραφεί ως εξής: Ο αποστολέας περιμένει την επιβεβαίωση από τον δέκτη ότι δεν υπήρξε κάποιο σφάλμα, για κάθε πακέτο που στάλθηκε. Αν εντοπιστεί σφάλμα σε κάποιο πακέτο, τότε ο αποστολέας πρέπει να ξαναστείλει το πακέτο αυτό και να ξαναγίνει ο έλεγχος για την ακεραιότητα του πακέτου.

Όταν επιβεβαιωθεί η ορθότητα του πακέτου τότε θα επιστραφεί στην Ιθάκη η ένδειξη ACK για να προχωρήσει στην αποστολή του επόμενου πακέτου. Στις περιπτώσεις που υπάρχει σφάλμα σε κάποιο πακέτο, θα επιστραφεί στην Ιθάκη η ένδειξη NACK.

Όπως και προηγουμένως, θα λάβουμε και θα αποθηκεύσουμε τα δεδομένα για να καταφέρουμε να υπολογίσουμε τον χρόνο απόκρισης του συστήματος και την πιθανότητα σφάλματος.

Μέσα σε μια for, που θα τρέξει μέχρι να υπάρξουν 200 πακέτα με σφάλμα, δημιουργώ ένα αρχείο που θα περιέχει τους χρόνους απόκρισης. Για άλλη μια φορά, με τη χρήση ενός Byte Array OutputStream, θα συλλέξω τα δεδομένα και θα δημιουργήσω έναν πίνακα 200 θέσεων όπου θα υπάρχει αποθηκευμένος ο αριθμός εκπομπής κάθε πακέτου. Στα δύο for που υπάρχουν θα γίνει ο διαχωρισμός των bytes από το πακέτο που έχουν πληροφορίες για τον κώδικα fcs και τα bytes που περιέχουν τον κώδικα των 16 bit.

```
char[] wPacket=packet.toCharArray();
```

```
char[] hex=new char[16];

char[] fcs=new char[3];

int p=0;

for (int i=0; i<=15; i++){

hex[p]=wPacket[i+31];

p++;

}

int z=0;

for (int i=0; i<=2; i++){

fcs[z]=wPacket[i+49];

z++;

}
```

Με την χρήση ενός xor τελεστή στον κώδικα των 16 bit θα πραγματοποιήσω τη σύγκριση με τον κώδικα fcs.

```
int a=hex[0];

for (int i=1;i<16;i++){

a=a^(hex[i]);

}
```

Πρέπει να συγκρίνω τον κωδικό fcs με το αποτέλεσμα που έχουμε λάβει από την xor, αφού πρώτα τα μετατρέψω σε μορφή String. Αν το αποτέλεσμα είναι 0, δηλαδή είναι ίσα, τότε γράφω στο αρχείο μου τον χρόνο απόκρισης του πακέτου και προωθώ τον ACK. Αν όμως η σύγκριση μας επιστρέψει 1, τότε προωθώ τον NACK κωδικό και προσθέτω το ένα στο συγκεκριμένο κελί, ώστε να δηλώσω ότι έγινε επανεκπομπή.

```
if (check==0){
count++;
if (count<200){
String timeACK= String.valueOf(dur);
byte[] ACK=timeACK.getBytes();
fout.write(ACK);
fout.write(System.getProperty("line.separator").getBytes());
sTimeACK=System.currentTimeMillis();
modem.write("Q1972\r".getBytes()); }
}
else {
modem.write("R5053\r".getBytes());
errors[count-1]++;
}
```

Πρωτόκολλα και μηχανισμοί λειτουργίας

Για την εργασία αυτή πρέπει να δημιουργήσουμε επικοινωνία μεταξύ δύο modem, του δικού μας και του server της Ιθάκης, μέσω της θύρας RS232.

Το πρωτόκολλο για σειριακή επικοινωνία που χρησιμοποιείται τις περισσότερες φορές είναι το *Universal Asynchronous Receiver Transmitter* (UART), το οποίο είναι ένα microchip που είναι υπεύθυνο για τη διασύνδεση του υπολογιστή μας με τις συσκευές που είναι συζευγμένες σε αυτόν σειριακά. Το UART δημιουργεί το *Data Terminal Equipment* (DTE) στον υπολογιστή μας, με σκοπό την επικοινωνία και την ανταλλαγή δεδομένων με άλλες συσκευές.

Στις περισσότερες περιπτώσεις το DTE (που είναι κατά πλειοψηφία modem) συνδέεται με το *Data Circuit-terminating Equipment* (DCE), το οποίο παρέχει clocking, σύνδεση και μετατροπή σημάτων σε συμβατή μορφή στον DCE.

Σε αυτό το σημείο, αξίζει να σημειωθούν επιγραμματικά οι λειτουργίες του microchip UART:

- Μετατρέπει τα bytes σε σειριακό ρεύμα από bits και το αντίστροφο.

- Προσθέτει κάποιο bit ισοτιμίας και ελέγχει την ισοτιμία των bytes εισόδου.

Για επικοινωνίες μεγάλων αποστάσεων, το UART δε μας προσφέρει την επιθυμητή αξιοπιστία και καταφεύγουμε στο RS232. Το RS232, εκτός από θύρα, είναι ένα πρωτόκολλο επικοινωνίας, όπου το λογικό ένα έχει τάση κάπου ανάμεσα στα +3 με +15 Volt και γι' αυτό είναι η πρώτη μας επιλογή για επικοινωνίες μεγαλύτερων αποστάσεων.

Πρωτόκολλα modem

Στην εποχή μας, σχεδόν όλοι γνωρίζουν τι είναι ένα modem. Ο επιστημονικός ορισμός του *modulator-demodulator* (modem) είναι η συσκευή εκείνη η οποία παραλαμβάνει και διαμορφώνει κάποιο αναλογικό σήμα, ώστε να κωδικοποιήσει την ψηφιακή πληροφορία και το αντίστροφο.

Τα πρωτόκολλα χωρίζονται σε δύο κύριες κατηγορίες, τα πρωτόκολλα αποσφαλμάτωσης και συμπίεσης δεδομένων και τα πρωτόκολλα ταυτόχρονης μετάδοσης σημάτων και συστημάτων. Στην πρώτη κατηγορία ανήκουν τα πρωτόκολλα V.42 και V.44, ενώ στη δεύτερη τα V.90 και V.92.

I. V.42

Ως V.42 ορίζουμε ένα *International Telephone and Telegraph Consultative Committee* (CCITT) V-series πρωτόκολλο το οποίο είναι υπεύθυνο για τη ρύθμιση και τον εντοπισμό σφαλμάτων σε υψηλής ταχύτητας modem.

Διαμέσου του V.42, το modem επικοινωνεί με αναλογικές και ψηφιακές τηλεφωνικές γραμμές, δηλαδή κάνει σύγχρονη και ασύγχρονη μετατροπή. Επίσης, δίνει τη δυνατότητα στον δέκτη να

ζητά την επανεκπομπή χαμένων packets, χωρίς όμως να παρέχει πληροφορίες για το χρόνο που θα χρειαστεί.

II. V.44

Το πρωτόκολλο V44 είναι υπεύθυνο για τη συμπίεση των δεδομένων μέσω ενός αλγορίθμου, μετατρέπει τους δυαδικούς κώδικες σε χαρακτήρες και τέλος, έχει τη δυνατότητα να καλείται αυτόματα σε περίπτωση που υπάρξουν ασυμπίεστα δεδομένα.

III. V.90

Το πρωτόκολλο V.90 είναι υπεύθυνο για τις ταχύτητες download και upload δεδομένων με τιμές 56kbps και 33.6kbps αντίστοιχα. Σχεδιάστηκε με πολλές υποσχέσεις, αλλά πολύ σύντομα εγκαταλείφθηκε και αντικαταστάθηκε από το V.92.

IV. V.92

Το πρωτόκολλο αυτό, όπως ανέφερα παραπάνω, αποτελεί βελτίωση του V.90, αφού επιτρέπει upload ταχύτητα μέχρι 48kbps και επίσης έχει πολύ καλή συμβατότητα με το V.44.

Βιβλιογραφία

Gallo, Michael και Hancock, William M., *Networking Explained* (2nd Edition), Digital Press, 2001.

Tanenbaum, Andrew S. και Wetherall, David J., *Computer Networks* (5th Edition), Pearson, 2011.

https://en.wikipedia.org/wiki/Modem

http://etymonline.com/index.php?allowed_in_frame=0&search=modem&searchmode=none

http://www.itu.int/rec/T-REC-V/en