Εργασία Δικτυακού Προγραμματισμού

Java Serial Communications Programming

Λιούπης Θεόδωρος
ΑΕΜ: 9733

ΑΝΑΦΟΡΑ ΓΙΑ ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΟΥ ΠΡΟΕΚΥΨΑΝ ΚΑΤΑ ΤΗΝ ΕΚΠΟΝΗΣΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
6ο εξάμηνο

Σχολιασμός αποτελεσμάτων

Παρακάτω παρουσιάζονται παρατηρήσεις και σχόλια που συνοδεύουν τα δεδομένα που προέκυψαν μετά από τις δύο συνόδους με τον server του εικονικού εργαστηρίου.

1) echoPackets

Κατά τις δύο συνόδους που πραγματοποιήθηκαν στις 21/04 και στις 23/04, χρησιμοποιήθηκε ο κωδικός echo_request_code για την λήψη πακέτων με το μορφή κειμένου από την Ιθάκη. Με σκοπό την εκτίμηση του χρόνου απόκρισης του συστήματος, η εφαρμογή virtualModem που αναπτύχθηκε στα πλαίσια της εργασίας, ζητάει διαρκώς echoPackets από τον server για χρόνο 5 λεπτών. Με τον τρόπο αυτό βγαίνει μία μέση εκτίμηση για τον χρόνο που χρειάζεται από την στιγμή που γίνεται κάθε φορά το συγκεκριμένο αίτημα στην Ιθάκη, μέχρι την λήψη ολόκληρου του πακέτου. Πιο συγκεκριμένα, παρατηρώντας τα γραφήματα G1 των δύο συνόδων (*session1.pdf και session2.pdf*) βλέπουμε ότι ο χρόνος απόκρισης για ένα echoPacket κυμαίνεται περίπου από 25 έως 95 milliseconds. Να σημειωθεί εδώ ότι το εικονικό μόντεμ έχει οριστεί να τρέχει σε ταχύτητες 80 Kbps (not fixed) με timeout στα 500 milliseconds. Τέλος, σε χρόνο 5 λεπτών, έγινε λήψη 8789 και 8383 πακέτων στις συνόδους ένα και δύο αντίστοιχα.

2) Εικόνες E1 και E2

Χρησιμοποιώντας την εφαρμογή virtualModem και τους αντίστοιχους image_requests_codes πραγματοποιήθηκε επιτυχώς η λήψη δύο εικόνων για κάθε σύνοδο, μία χωρίς σφάλματα και μία με σφάλματα μετάδοσης, από το server της Ιθάκης. Περισσότερες λεπτομέρειες για το τρόπο που γίνεται η λήψη και η αποθήκευση των εικόνων θα δοθούν στη συνέχεια, κατά την σύντομη περιγραφή του κώδικα της εφαρμογής.

3) Εικόνα M1 με ίχνη GPS

Για την λήψη της εικόνας που απεικονίζει τα διάφορα ίχνη GPS πάνω στο χάρτη χρειάστηκε πρώτα να προσδιοριστούν ποια ακριβώς ίχνη θέλαμε να προβάλουμε σε κάθε σύνοδο. Έτσι, πρώτα χρησιμοποιήθηκε ο κωδικός PxxxxR=XPPPPLLL για την επιστροφή ιχνών GPS από την Ιθάκη. Στη συνέχεια έγινε προσεκτική επιλογή των συντεταγμένων, ώστε τα ίχνη να έχουν απόσταση μεταξύ τους και να αποφευχθεί η επικάλυψη του ενός από το άλλο κατά την απεικόνιση τους. Κατόπιν, πραγματοποιήθηκε η κατάλληλη μετατροπή των συντεταγμένων κάθε ίχνους στην κατάλληλη μορφή και στάλθηκαν όλα μαζί στη σωστή σειρά, χρησιμοποιώντας την παράμετρο T=ΑΑΒΒΓΓΔΔΕΕΖΖ συνοδευτικά με το request code Pxxxx, ώστε η Ιθάκη να προβάλει τα αντίστοιχα pins στον χάρτη. Τέλος, η λήψη της εικόνας γίνεται με τον ίδιο τρόπο όπως στις E1 και E2.

4) Μηχανισμός ARQ

Τελικό στάδιο της εργασίας είναι η εξοικείωση με τη διαχείριση σφαλμάτων μετάδοσης που προκαλεί ο θόρυβος στο κανάλι επικοινωνίας. Ειδικότερα, καλούμαστε να διαχειριστούμε σφάλματα πακέτων χρησιμοποιώντας τον μηχανισμό ARQ (automatic repeat request). Όπως γνωρίζουμε, ο δέκτης απαντά με ACK ή NACK request code αναλόγως με την κατάσταση του ληφθέντος πακέτου (λανθασμένου ή σωστού). Ιδιαίτερα σημαντικό είναι να εξάγουμε συμπεράσματα για το χρόνο απόκρισης του συστήματος αλλά και την κατανομή πιθανότητας επανεκπομπής πακέτου. Για τον λόγο αυτό, χρησιμοποιήθηκε ο μηχανισμός ARQ για την συνεχόμενη λήψη πακέτων για χρόνο 4 λεπτών.

Παρατηρώντας τα γραφήματα G2 όπως παρουσιάζονται στα αρχεία *session1.pdf* και *session2.pdf*, το σύστημα χρειάζεται χρόνο να ανταποκριθεί που κυμαίνεται από 35 έως 380 milliseconds, ανάλογα τον αριθμό των επανεκπομπών κάθε φορά. Αναφορικά, για μηδενική επανεκπομπή ο χρόνος απόκρισης κυμαίνεται από 35 έως 45 milliseconds, για μονή επανεκπομπή από 70 μέχρι 90 milliseconds, για διπλή από 110 μέχρι 130 κλπ.

Ιδιαίτερο ενδιαφέρον έχει το αποτέλεσμα που προκύπτει, αν κανείς παρατηρήσει τα γραφήματα G3 στα αντίστοιχα αρχεία από τις δύο συνόδους. Αρχικά, στο πρώτο βλέπουμε τους αριθμούς των συνολικών πακέτων για κάθε i επανεκπομπή, όπου i = 0,1,2…9. Αν συγκρίνουμε τα πακέτα που χρειάστηκαν μηδενική επανεκπομπή με εκείνα που χρειάστηκαν ακόμη μία επανεκπομπή, θα δούμε ότι τα πρώτα είναι περισσότερα από τα δεύτερα. Αντίστοιχα και για τα υπόλοιπα. Δηλαδή, σε ένα μεγάλο αριθμό λήψης πακέτων, η πιθανότητα ένα πακέτο να χρειαστεί επανεκπομπή μειώνεται όσο αυξάνεται ο αριθμός των επανεκπομπών. Ας παρατηρήσουμε τώρα το γράφημα G3 που δείχνει την κατανομή πιθανότητας για τον αριθμό επανεκπομπής πακέτου. Βλέπουμε ότι η κατανομή παρουσιάζει εκθετική μείωση. Μπορούμε δηλαδή να πούμε ότι η κατανομή πιθανότητας αυτή, μπορεί να περιγραφεί από την γεωμετρική κατανομή πιθανότητας. Πράγματι, από τον ορισμό της γεωμετρικής πιθανότητας, «σε μία σειρά δοκιμών Bernoulli (ανεξάρτητες δοκιμές με σταθερή πιθανότητα *P* για την επιτυχία) η τυχαία μεταβλητή X, που ισούται με το πλήθος των δοκιμών (*επανεκπομπών*) μέχρι την πρώτη επιτυχία, είναι μία γεωμετρική τυχαία μεταβλητή».

Τέλος, απαραίτητος είναι ο υπολογισμός του Bit Error Rate (BER). Ο αριθμητικός υπολογισμός αυτής της ποσότητας περιγράφεται συνοπτικά στα αρχεία *session1.pdf* και *session2.pdf.* Αναφορικά γίνεται χρήση του τύπου $P = (1 - BER)^L$ όπου *P* η πιθανότητα επιτυχούς λήψης πακέτου και *L* το συνολικό μήκος του πακέτου σε bits.

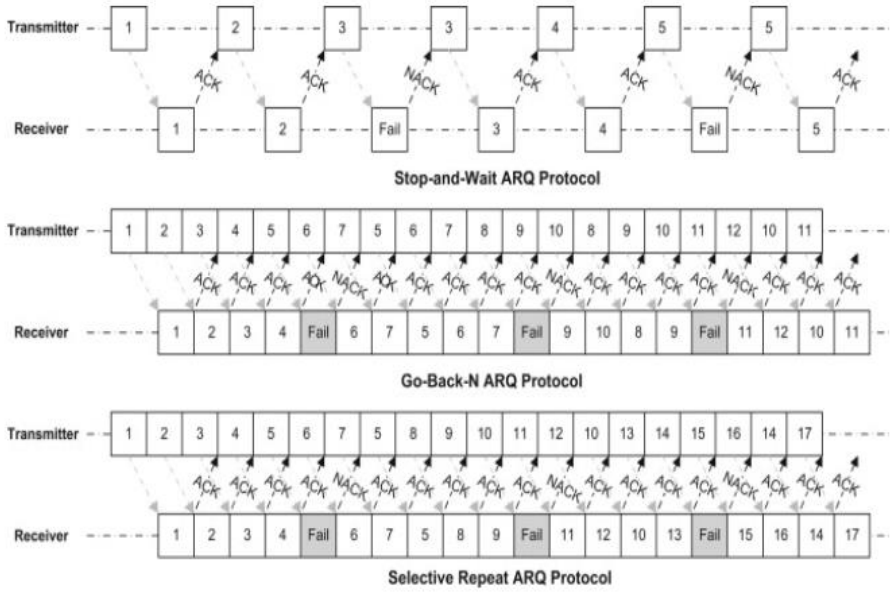
Παρατήρηση: η πιθανότητα *P* στον παραπάνω τύπο ισούται με την πιθανότητα λήψης επιτυχούς πακέτου δηλαδή με $\frac{numOfAcks}{numofAcks+numofNacks}$ που ταυτίζεται με την πιθανότητα μηδενικής επανεκπομπής.

Βιβλιογραφική αναφορά

1) Πρωτόκολλα ARQ

Όπως έχει συζητηθεί και στο μάθημα των Δικτύων 1, το ARQ (automatic repeat request) είναι ένας μηχανισμός ελέγχου των σφαλμάτων κατά την μετάδοση δεδομένων, ο οποίος χρησιμοποιεί μηνύματα επιβεβαίωσης λήψης (θετικά ή αρνητικά) και timeouts ώστε να πετύχει την αξιόπιστη μετάδοση πληροφορίας μέσα από ένα κανάλι θορύβου. Υπάρχουν 3 πρωτόκολλά ARQ:

a. Το πρωτόκολλο Stop and Wait ARQ (σταμάτα και περίμενε) είναι το βασικό πρωτόκολλο ARQ κατά το οποίο ο αποστολέας της πληροφορίας στέλνει ένα πακέτο την φορά και μετά περιμένει από τον δέκτη την θετική ένδειξη ACK για την σωστή λήψη του πακέτου πληροφορίας ή την αρνητική ένδειξη NACK για την λήψη πακέτου με σφάλματα. Σε περίπτωση που η ένδειξη ACK χρειαστεί περισσότερο χρόνο από χρόνο *timeout* για να φτάσει πίσω στον πομπό ή ο πομπός λάβει ένδειξη NACK από τον δέκτη, τότε ξαναστέλνει το ίδιο πακέτο.
b. Το πρωτόκολλο Go-Back-N-ARQ (πήγαινε πίσω N φορές) είναι μία μορφή ARQ που συνεχώς στέλνει πακέτα στον δέκτη, χωρίς να περιμένει για την ένδειξη ACK ή NACK μέσα σε ένα χρονικό «παράθυρο» μετάδοσης. Ο δέκτης καθώς δέχεται τα πακέτα, συγκρατεί την ακολουθία των αριθμών των πακέτων που έρχονται και έτσι «ξέρει» τον αριθμό του πακέτου που πρέπει να έρθει μετά. Για παράδειγμα, αν έχει λάβει τα πακέτα 1, 2, 3, 4 περιμένει το επόμενο πακέτο που θα έρθει να είναι το 5. Ο αριθμός αυτός στέλνεται κάθε φορά μαζί με το αναγνωριστικό ACK/NACK κάθε φορά πίσω στο πομπό. Ο δέκτης αγνοεί τα πακέτα που δεν έχουν τον αναμενόμενο αριθμό επειδή είτε έχει έρθει ένα διπλότυπο πακέτο, είτε πακέτο με μεγαλύτερο αριθμό. Μόλις ο πομπός στείλει όλα τα πακέτα στο χρονικό περιθώριο που είχε (αναλόγως τον αριθμό N), ελέγχει αν έχουν αναγνωριστεί όλα τα πακέτα που είχε στείλει. Αν για κάποιο πακέτο ήρθε η ένδειξη NACK ή δεν έγινε καμία αναγνώριση από τον δέκτη, τότε ο πομπός ξαναστέλνει όλα τα πακέτα ξεκινώντας από εκείνο για το οποίο έγινε η αρνητική ένδειξη αναγνώρισης μέχρι και το τελευταίο πακέτο το οποίο ο δέκτης έλαβε σωστά (σύμφωνα με την ακολουθία των αριθμών που καταγράφει ο δέκτης και τον αριθμό N).
c. Το πρωτόκολλο Selective Repeat ARQ (επανεκπομπή επιλεγμένου πακέτου) το οποίο μοιάζει με το Go-Back-N-ARQ απλά με μία βασική διαφορά. Ο πομπός επανεκπέμπει μόνο εκείνα τα πακέτα για τα οποία έχει λάβει NACK ή καθόλου αναγνώριση από τον δέκτη. Αυτό έρχεται σε αντίθεση με τον προηγούμενο πρωτόκολλο το οποίο απαιτεί την επανεκπομπή όλων των πακέτων που έχουν σταλεί από το λανθασμένο πακέτο και μετά.



2) Πρωτόκολλο MNP (Microcom Networking Protocol)

Το πρωτόκολλο δικτύωσης Microcom (MNP) είναι ένα πρωτόκολλο επικοινωνίας που αναπτύχθηκε αρχικά από την Microcom Inc για διορθώσεις σφαλμάτων και συμπιέσεις. Διορθώνει τις τροποποιήσεις που εισάγονται στα δεδομένα κατά τη διάρκεια εκπομπών μέσω παρεμβολών τηλεφωνικής γραμμής και προσφέρει διάφορα επίπεδα για διορθώσεις δεδομένων και συμπίεση.

Για την διαπίστωση σφαλμάτων, δεδομένα όπως CRC ή checksum προστίθενται σε κάθε πακέτο, το οποίο υποδεικνύει το αρχικό περιεχόμενο. Τα πακέτα ελέγχονται από το CRC για ανάλυση σφαλμάτων. Εάν δεν εντοπιστούν σφάλματα, αποστέλλεται ένα μήνυμα επιβεβαίωσης υπογράφοντας ένα αίτημα για το επόμενο πακέτο. Διαφορετικά, αποστέλλεται μια αρνητική επιβεβαίωση ζητώντας την αποζημίωση του πακέτου που υπέστη ζημία. Η διαδικασία αυτή όμως, καθώς είναι χρονοβόρα, για να επιταχυνθεί συνδυάστηκε με το πρωτόκολλο των ολισθηρών παραθύρων (sliding windows) που επιτρέπει το σύστημα να προχωρήσει στο επόμενο πακέτο χωρίς να έχει έρθει κάποιου είδους αναγνώριση για τα πακέτα που έχουν σταλθεί μέχρι στιγμής. Αυτό βέβαια, συμβαίνει για κάποιο χρονικό παράθυρο, για το οποίο αν δεν έχει γίνει θετική ή αρνητική αναγνώριση, τα πακέτα στέλνονται ξανά. Όπως παρατηρούμε η λογική είναι ίδια με εκείνη του Go-Back-N-ARQ πρωτοκόλλου.

Υπάρχουν 9 διαφορετικές εκδόσεις MNP (που ονομάζονται κλάσεις) με την καθεμία να εισάγει καινούργιες μεθόδους λειτουργίας, με το πέρασμα του χρόνου. Το MNP είναι το πιο διαδεδομένο πρωτόκολλο modem σήμερα, λόγω της ικανότητας του για γρήγορο εντοπισμό σφαλμάτων και διόρθωσης τους.

Σχολιασμός του κώδικα Java

Παρακάτω περιγράφεται με σύντομο τρόπο ο κώδικας που δημιουργήθηκε κατά την ανάπτυξη εφαρμογής εικονικού μόντεμ στα πλαίσια της εργασίας.

Η εφαρμογή *virtualModem* υλοποιείται σε ένα αρχείο *virtualModem.java* το οποίο τρέχει τοπικά στον υπολογιστή με την χρήση του τερματικού. Ο κώδικας αναπτύχθηκε με την χρήση του Visual Studio Code και βρίσκεται στο αρχείο *source.pdf.*

Για να τρέξει η εφαρμογή είναι απαραίτητη η κλάση *Modem.class* που παρέχεται από τον server της Ιθάκης και πρέπει να βρίσκεται στο ίδιο *file path* μαζί με τον πηγαίο κώδικα του αρχείο *virtualModem.java (source.pdf)*.

Η εφαρμογή virtualModem υλοποιείται μέσα από την αντίστοιχη κλάση virtualModem η οποία έχει τις εξής μεθόδους:

- *public static void message_request(Modem modem):* η συγκεκριμένη συνάρτηση χρησιμοποιείται για την απεικόνιση των πακέτων σε μορφή κειμένου που λαμβάνουμε από την Ιθάκη κατά την σύνδεση μας με τον server (*welcome message*) και κατά την λήψη echoPackets. Να σημειωθεί ότι η συγκεκριμένη συνάρτηση καλείται αφού πρώτα έχει γίνει το αντίστοιχο request στην Ιθάκη με την κλήση της μεθόδου modem.write().
- *public void echoPacketLoop(Modem modem, String requestCode):* η μέθοδος αυτή καλείται για την ανάλυση της απόκρισης του συστήματος κατά την λήψη πακέτων echoPackets. Η μέθοδος αυτή, όταν κληθεί, συλλέγει δεδομένα για χρόνο 5 λεπτών.
- *public void image_request(Modem modem, String requestCode, String imageName):* η συγκεκριμένη μέθοδος χρησιμοποιείται όταν είναι απαραίτητη η λήψη εικόνας από τον server της Ιθάκης. Δίνοντας τον κατάλληλο κωδικό στην συνάρτηση στο πλαίσιο *request code*, η μέθοδος επιστρέφει την εικόνα που ζητήθηκε (*gps εικόνα, εικόνα με ή χωρίς σφάλματα*) και την αποθηκεύει σε αρχείο με όνομα *imageName* που δίνεται ως όρισμα στην συνάρτηση.
- *public void gps_request(Modem modem, String requestCode):* Η συγκεκριμένη μέθοδος τυπώνει στο τερματικό τα ίχνη GPS που έχουν ζητηθεί από την Ιθάκη με τον κατάλληλο *requestCode.*
- *public void ARQ(Modem modem, String ackRequestCode, String nackRequestCode):* η συνάρτηση ARQ υλοποιεί την επαναλαμβανόμενη διαδικασία που έχουμε περιγράψει και παραπάνω για την διόρθωση σφαλμάτων κατά την μετάδοση πληροφορίας μέσα από ένα κανάλι με θόρυβο. Η χρόνος για τον οποίο συλλέγει δεδομένα είναι 4 λεπτά. Οι κωδικοί ACK και NACK δίνονται ως όρισμα κατά την κλήση της συνάρτησης.

- *public static void main(String[] param):* Εδώ υλοποιείται η εφαρμογή *virtualModem* με την σωστή χρήση των παραπάνω μεθόδων. Ο κώδικας της *main* έχει γραφτεί με τέτοιον τρόπο ώστε, κατά την κλήση του εικονικού μόντεμ, να ζητείται από τον χρήστη να εισάγει το επιθυμητό *requestCode* στο τερματικό του. Στη συνέχεια, μετά από έλεγχο του κωδικού, καλείται η κατάλληλη συνάρτηση. Να σημειωθεί εδώ ότι, μετά τον τερματισμό κάποιας μεθόδου, η εφαρμογή ζητάει ξανά από το χρήστη την εισαγωγή νέου *requestCode*. Η συγκεκριμένη διαδικασία συνεχίζει μέχρι ο χρήστης να τερματίσει την εφαρμογή πατώντας exit (*EXIT or Exit*) στο τερματικό του.