Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Πληροφορικής
Μάθημα ΠΠΣ : Ψηφιακές Επικοινωνίες (4ο εξάμηνο)
Εργασία : Υλοποίηση Αλγορίθμου CRC
Ονοματεπώνυμο Φοιτητή : Ηλίας Τερζής
ΑΕΜ Φοιτητή : 3170

Εισαγωγή

Η εργασία αυτή αφορά στην υλοποίηση του κώδικα ανίχνευσης σφαλμάτων CRC (Cyclic Redundancy Check- Κυκλικός Έλεγχος Πλεονασμού) ο οποίος επινοήθηκε και δημοσιεύθηκε από τον Γουέσλεϊ Πέτερσον (W. Wesley Peterson) το 1961. Είναι ένας από τους πιο συνηθισμένους , και πιο ισχυρούς κώδικες ανίχνευσης σφαλμάτων και περιγράφεται ως ακολούθως:
Δοθέντος ενός μπλοκ δεδομένων των k bits , ο μεταδότης δημιουργεί μια ακολουθία των $n<k$ bits (Frame Check Sequence, FCS) ,τέτοια ώστε η συνολική ακολουθία των k+n bits που προκύπτει να διαιρείται ακριβώς με κάποιον προκαθορισμένο αριθμό P των n+1 bits. Όταν η ακολουθία των k +n bits φθάσει στον αποδέκτη, τότε η ορθότητά της ελέγχεται διαιρώντας την με τον προκαθορισμένο αριθμό P. Αν από τη διαίρεση αυτή δεν προκύψει υπόλοιπο, τότε το πακέτο γίνεται αποδεκτό. Αν προκύψει υπόλοιπο, τότε συνάγεται ότι το πακέτο έχει αλλοιωθεί και ζητείται η επαναμετάδοσή του.

Στην 1η ενότητα της αναφοράς αυτής γίνεται η περιγραφή του κώδικα του προγράμματος.Στην 2η ενότητα , παρατίθενται τα αποτελέσματα από δύο εκτελέσεις του προγράμματος ως ενδεικτικά παραδείγματα και στην 3η και τελευταία ενότητα τα αποτελέσματα από ακόμα μία εκτέλεση για εισόδους όμως που ορίζονται στην εκφώνηση της εργασίας.

1.Περιγραφή του κώδικα του προγράμματος

Για την υλοποίηση του αλγορίθμου χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java SE 12 και ως ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) το IntelliJ IDEA (educational edition) .

Το πρόγραμμα απαρτίζουν τρεις διαφορετικές κλάσεις : η κλάση Main, η κλάση Cli και η κλάση Message. Η Main είναι η κλάση από την οποία ξεκινάει το πρόγραμμα. Εδώ δημιουργούνται τα αντικείμενα των δύο άλλων κλάσεων για να εκτελεσθούν οι απαραίτητες λειτουργίες. Η Cli είναι υπεύθυνη για να ζητάει από το χρήστη τα δεδομένα και να εμφανίζει τα αποτελέσματα. Τέλος , στην Message γίνονται όλες οι απαραίτητες διεργασίες για την υλοποίηση του αλγορίθμου.

Πιο συγκεκριμένα μέσω της κλάσης Cli από το χρήστη ζητείται τόσο το πλήθος n των μηνυμάτων , όσο και το μέγεθός τους (k bits) .Ακόμη , ο χρήστης καλείται να εισάγει τον προκαθορισμένο διαιρέτη P και τον εκθέτη BER (Bit Error Rate) για την πιθανότητα να συμβεί σφάλμα καθώς τα μηνύματα μεταδίδονται μέσα από το ενθόρυβο κανάλι.

Στην κλάση Messages γίνεται – με τυχαίο τρόπο – η δημιουργία των μηνυμάτων , τα οποία αποθηκεύονται σε έναν διδιάστατο πίνακα ακεραίων messages[n][k]. Στη συνέχεια υπολογίζεται ο αριθμός των bits του P - ο οποίος μετατρέπεται σε μονοδιάστατο πίνακα ακεραίων – και προστίθενται τα n+1 bits αυτού του αριθμού για να υλοποιηθεί ο πολλαπλασιασμός:

$$T = 2^{n} M ,$$

όπου T η ακολουθία bit που θα μεταδοθεί, και M το αρχικό μήνυμα. Αυτό γίνεται στη μέθοδο addZerosToMess().
Στην μέθοδο applyCRCforMessages() καλείται επαναληπτικά για κάθε μήνυμα η μέθοδος crc(int[] t) η οποία δέχεται ένα αντίγραφο μηνύματος ως πίνακα int[] t . Στην μέθοδο crc(int[] t) έχουμε αντίστοιχα επαναληπτική κλήση της μεθόδου xor(int[] t, long[] divisor,int pos) στην οποία κάθε φορά στέλνουμε την θέση pos από την οποία πρέπει να εφαρμοστεί η λογική πράξη xor (αριθμητική modulo-2). Η μέθοδος xor επίσης δέχεται τον πίνακα t και τον πίνακα με τα n+1 ψηφία του P. Έτσι , κάνει ανά bit την σύγκριση για το αποκλειστικό-Ή (xor) για τα πρώτα n+1 ψηφία και αποθηκεύει το αποτέλεσμα στην κάθε θέση του t (δηλαδή απλώς αλλάζει το περιεχόμενό του). Στην επόμενη κλήση της , θα πάει στα επόμενα n+1 ψηφία . Αυτό γίνεται για k φορές μέσω του βρόχου στην crc(int[] t) , δηλαδή τόσες φορές όσα είναι τα ψηφία του μηνύματος πριν την προσθήκη των μηδενικών. (Στην crc(int[] t) γίνεται έλεγχος κάθε φορά ώστε η xor να καλείται μόνο από το σημείο που τα bits ξεκινούν με 1.) Στο τέλος ως αποτέλεσμα έχουμε τον τροποποιημένο πίνακα t , του οποίου τα n-1 ψηφία τοποθετούμε στο τέλος του μηνύματος (εκεί όπου πριν υπήρχαν μηδενικά). Δηλαδή εκτελείται η πράξη :

$$T = 2^{n} M + F ,$$

όπου n το μέγεθος της ακολουθίας FCS, M το αρχικό μήνυμα και F η ακολουθίας FCS που υπολογίστηκε από τον αλγόριθμο CRC.

Αυτή η εργασία γίνεται για όλα τα μηνύματα και ως αποτέλεσμα μετά από την κλήση της μεθόδου applyCRCforMessages() έχουμε τροποποιήσει τον πίνακα messages, και πλέον περιλαμβάνει τις ακολουθίες T που θα σταλούν .

Στη συνέχεια γίνεται η κλήση της μεθόδου errorGenerator για να αναπαρασταθείι η μετάδοση των μηνυμάτων μέσα από το ενθόρυβο κανάλι. Ουσιαστικά για κάθε bit του κάθε μηνύματος , γίνεται η σύγκριση ενός ψευδοτυχαίου αριθμού με το 1 . Εάν αυτός ο αριθμός τύχει να είναι το 1 , γίνεται αλλαγή στο bit (από $1 \rightarrow 0$ ή $0 \rightarrow 1$). Ο αριθμός αυτός δημιουργείται από μια γεννήτρια Random ψευδοτυχαίων αριθμών από το 0 έως το $10^{|BER|}$-1 (επειδή υπολογίζεται η απόλυτη τιμή του BER δεν έχει σημασία εάν ο χρήστης εισάγει εδώ θετικό ή αρνητικό αριθμό). Έτσι, εάν για παράδειγμα ο χρήστης για BER εισάγει το -3 , ο ψευδοτυχαίος αριθμός θα είναι από το 0 έως το 999. Μέσα στην μέθοδο errorGenerator μετριέται ο αριθμός των μηνυμάτων , στα οποία έγινε αλλοίωση .

Μετά από αυτό, πρέπει να γίνει ο έλεγχος στον αποδέκτη με τον αλγόριθμο CRC για να βρει ποια μηνύματα αλλοιώθηκαν. Τυπικά, το σύστημα του αποδέκτη μόλις ανιχνεύσει σφάλμα ζητάει την επαναμετάδοση του μηνύματος που το περιέχει. Στο πρόγραμμα αυτό, όμως, απλώς μετριέται ο αριθμός των αλλοιωμένων μηνυμάτων. Για τον έλεγχο του εάν μια ακολουθία bit που ελήφθη αποτελεί αλλοιωμένο μήνυμα , καλείται για αυτό το μήνυμα ο αλγόριθμος crc . Μετά από την διαίρεση με τον ίδιο προκαθορισμένο αριθμό P , θα πρέπει να μην υπάρχει υπόλοιπο (όλα τα bit να είναι 0) για να συμπεράνει ο αποδέκτης ότι έλαβε ακέραιο το μήνυμα.

Τέλος , το πρόγραμμα ολοκληρώνει τις λειτουργίες του, εμφανίζοντας τον αριθμό των μηνυμάτων τα οποία αλλοιώθηκαν αλλά και τον αριθμό των μηνυμάτων που ο έλεγχος CRC στον αποδέκτη μέτρησε ως αλλοιωμένα. Μπορεί ο 2ος αυτός αριθμός να είναι μικρότερος, γιατί αν και με μικρή πιθανότητα, υπάρχει η περίπτωση ο αλγόριθμος να μην ανιχνεύσει μερικά σφάλματα που δημιουργήθηκαν (όσο πιο μεγάλος ο αριθμός P βέβαια, τόσο πιο απίθανο να συμβεί κάτι τέτοιο).

Στη συνέχεια , παρουσιάζονται δύο παραδείγματα εκτέλεσης του προγράμματος.

2.Παραδείγματα λειτουργίας του προγράμματος

α) Ως ένα πρώτο παράδειγμα , παρατίθεται εδώ η εκτέλεση του προγράμματος με τις ακόλουθες εισόδους:

- Αριθμός μηνυμάτων: 1.000.000
- Μέγεθος του κάθε μηνύματος: 10 bits
- Bit Error Rate: 10^{-2}
- Διαιρέτης P : 1011

```
Number of messages will you send :
1000000
Number of bits for every message :
10
Bit Error Rate has  base  10 , give the exponent :
2
The P number for CRC algorithm is :
1011
Transmission completed.
In this session 1000000 messages were transmitted.
CRC detected as incorrect 63777 messages.
In fact they were sent with error 63935 messages
```

1ο παράδειγμα λειτουργίας

Παρατηρούμε ότι 63.935 μηνύματα αλλοιώθηκαν ενώ ο αλγόριθμος CRC ανίχνευσε 63.777, δηλαδή “αποδέχτηκε” ως ορθά 158 μηνύματα ενώ αυτά είχαν αλλοιωθεί. Γι’αυτά τα μηνύματα , ο αποδέκτης δεν θα κάνει αίτηση επαναποστολής και θα τα δεχθεί σαν να ήταν ορθά.

β) Στο 2ο παράδειγμα λειτουργίας , δίνουμε στο πρόγραμμα τις ίδιες εισόδους εκτός από τον διαιρέτη P που τον θέτουμε σε 101111.

```
Number of messages will you send :
1000000
Number of bits for every message :
10
Bit Error Rate has  base  10 , give the exponent :
2
The P number for CRC algorithm is :
101111
Transmission completed.
In this session 1000000 messages were transmitted.
CRC detected as incorrect 73005 messages.
In fact they were sent with error 73006 messages
```

2ο παράδειγμα λειτουργίας

Παρατηρούμε ότι 73.006 μηνύματα αλλοιώθηκαν ενώ ο αλγόριθμος CRC ανίχνευσε 73.005, δηλαδή “αποδέχτηκε” μόνο 1 μήνυμα ως ορθό ενώ αυτό είχε σφάλμα! Η σημείωση που αξίζει να κάνουμε εδώ, αφορά τον διαιρέτη P : επαληθεύεται το γεγονός ότι όσο μεγαλύτερος είναι, τόσο πιο απίθανη είναι η περίπτωση να περάσει τον CRC έλεγχο του αποδέκτη ένα αλλοιωμένο μήνυμα. Μάλιστα , στο παράδειγμα , αυξήσαμε μόνο κατά δύο bit το μέγεθός του. Στην πράξη χρησιμοποιούνται πολύ μεγαλύτεροι αριθμοί ως προκαθορισμένοι διαιρέτες (π.χ. CRC-32) και εκεί οι πιθανότητες να “αποτύχει” στον έλεγχο ο αλγόριθμος είναι ελάχιστες.

3.Παράδειγμα λειτουργίας από την εκφώνηση της εργασίας

Σε αυτή την ενότητα , παρουσιάζεται το αποτέλεσμα εκτέλεσης του προγράμματος και γίνεται ο υπολογισμός των ζητούμενων ποσοστών για τις καθορισμένες από την εκφώνηση εισόδους :

- Αριθμός μηνυμάτων : δεν δηλώνεται στην εκφώνηση οπότε επιλέγεται ο ίδιος με τα προηγούμενα παραδείγματα.
- Μέγεθος του κάθε μηνύματος: 10
- Bit Error Rate: 10^{-3}
- Διαιρέτης P : 110101

```
Number of messages will you send :
1000000
Number of bits for every message :
10
Bit Error Rate has  base  10 , give the exponent :
-3
The P number for CRC algorithm is :
110101
Transmission completed.
In this session 1000000 messages were transmitted.
CRC detected as incorrect 7596 messages.
In fact they were sent with error 7596 messages
```

Εκτέλεση του προγράμματος για τις δοσμένες τιμές από την εκφώνηση.

Ποσοστά για τα αποτελέσματα:

- Το ποσοστό των μηνυμάτων που φθάνουν με σφάλμα στον αποδέκτη είναι : 0.7596% .
- Το ποσοστό των μηνυμάτων που ανιχνεύονται ως εσφαλμένα από τον CRC αλγόριθμο στον αποδέκτη είναι : 0.7596% .
- Το ποσοστό των μηνυμάτων που φθάνουν με σφάλμα στον αποδέκτη αλλά δεν ανιχνεύονται ως εσφαλμένα από τον CRC αλγόριθμο είναι : 0% .

Βιβλιογραφία

Για την εισαγωγή αντλήθηκε υλικό από την Βικιπαίδεια , το κεφ. 6.5 του βιβλίου “*Επικοινωνίες Υπολογιστών και Δεδομένων ,10η έκδοση*” του William Stallings και από τις διαφάνειες της εκφώνηση της εργασίας.