Ψηφιακές Επικοινωνίες - Εργασία
Αλγόριθμος CRC

Δεληγιαννάκης Χαράλαμπος
4383

Ιούνιος 2024

Εισαγωγή

Στην εργασία παρουσιάζεται η υλοποίηση του αλγορίθμου CRC (Cyclic Redundancy Check), ο οποίος αποτελεί έναν από τους πιο γνωστούς κώδικες ανίχνευσης σφαλμάτων στη μετάδοση δεδομένων.

1 Πρώτο ζητούμενο

Ο αλγόριθμος γράφτηκε σε γλώσσα Python και τα αρχεία έχουν ονόματα *"main.py"* και *"package.py"*.

Η βασική κλάση του αρχείου είναι η κλάση **Package** όπου υλοποιείται στο *"package.py"* περιλαμβάνει όλες τις απαραίτητες πληροφορίες του πακέτου με τα δεδομένα (όπως τις μεταβλητές D, F, T, P της εκφώνησης) καθώς και όλες τις απαραίτητες λειτουργίες του αλγορίθμου CRC (modulo-2 binary division). Ουσιαστικά, η υλοποίηση του αλγορίθμου βρίσκεται μέσα σε αυτήν την κλάση. Ας αναλύσουμε κάθε μέθοδό της αναλυτικότερα:

- Κατασκευαστής κλάσης:
  Δέχεται ως παραμέτρους τις μεταβλητές D (η προς μετάδοση ακολουθία δεδομένων των k bits) και P (ο προκαθορισμένος αριθμός των $n-k+1$ bits με τον οποίο θα πρέπει να είναι διαιρέσιμη η ακολουθία των n bits που πρόκειται να μεταδοθεί).

- Μέθοδος send():
  Το πακέτο υποτίθεται ότι στάλθηκε και στη συγκεκριμένη μέθοδο καλούνται οι 4 παρακάτω υπομέθοδοι με τη σειρά για να υλοποιήσουν τον αλγόριθμο CRC.

  1. Μέθοδος calculateF():
     Υπολογίζει την ακολουθία FCS (F) των $n-k$ bits, εκτελώντας την modulo-2 διαίρεση του $2^{n-k}D$ με το P και χρησιμοποιώντας το υπόλοιπο αυτής.
  2. Μέθοδος calculateT():
     Υπολογίζει την ακολουθία T των n bits που πρόκειται να μεταδοθεί, εκτελώντας την πράξη $2^{n-k}D+F$.
  3. Μέθοδος errors():
     Προσομοιώνει την δημιουργία σφαλμάτων κατά τη μετάδοση. Στον συγκεκριμένο κώδικα το bit error rate είναι 10^{-3}.
  4. Μέθοδος calculateRes():
     Υπολογίζει το υπόλοιπο της modulo-2 διαίρεσης του T με το P, το οποίο δείχνει αν υπήρχε σφάλμα κατά τη μετάδοση. (Αν υπήρχε, είναι διάφορο του μηδενός)

- Μέθοδος xor():
  Υπολογίζει το αποτέλεσμα της γνωστής πράξης XOR μεταξύ δύο δυαδικών αριθμών

που είναι σε μορφή String.

- Μέθοδος binary_division():
  Υπολογίζει το υπόλοιπο της modulo-2 διαίρεσης δύο δωαδικών αριθμών.

Στο αρχείο *"main.py"*, για το πρώτο ζητούμενο δημιουργείται τυχαία αριθμός k μεταξύ 10 και 30 όπου αντιπροσωπεύει το μέγεθος της πληροφορίας σε bits. Έπειτα, ζητείται από τον χρήστη να εισάγει τον σταθερό αριθμό P και τέλος δημιουργούνται 10 τυχαία μηνύματα (μεγέθους k bits το καθένα) τα οποία στέλνονται και ελέγχονται από τον αλγόριθμο. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
Length of data messages is:  19  bits per message.
Give P: 10111010
Message  0001111100111001100  got sent correctly with FCS:  0110110  and result:  0000000
Message  1000000101101100110  got sent correctly with FCS:  1011100  and result:  0000000
Message  0111010011011110001  got sent correctly with FCS:  0100000  and result:  0000000
Message  0111000101010111010  got sent correctly with FCS:  0010110  and result:  0000000
Message  1010100001011010000  got sent correctly with FCS:  1110000  and result:  0000000
Message  1111100011011100101  got sent correctly with FCS:  0110000  and result:  0000000
Message  1110000001100100100  got sent incorrectly (!) with FCS:  1001100  and result:  0001000
Message  1010001111111010111  got sent correctly with FCS:  1101100  and result:  0000000
Message  1100110001001000011  got sent correctly with FCS:  0101010  and result:  0000000
Message  1100100111101110111  got sent correctly with FCS:  1001100  and result:  0000000
```

Figure 1: Έξοδος τυχαίας εκτέλεσης του προγράμματος για το πρώτο ζητούμενο

2 Δεύτερο ζητούμενο

Η εκτέλεση για τα δεδομένα του δεύτερου ζητουμένου υλοποιείται στη συνάρτηση *randomPackages*() του αρχείου *"main.py"*. Σε αυτήν, δημιουργούνται αντικείμενα της κλάσης Package, όπου σε κάθε πακέτο μεταδίδεται μια ακολουθία με συγκεκριμένο μέγεθος. Στη συνέχεια, για κάθε πακέτο καλείται η μέθοδος send(), και τυπώνονται τα κατάλληλα βοηθητικά μηνύματα του δεύτερου ζητούμενου. Τονίζεται πως για να τρέξει το δεύτερο ζητούμενο, πρέπει η γραμμή 36 στο main αρχείο να μην είναι commented, δηλαδή να γίνει η εξής αλλαγή: *#randomPackages() -> randomPackages()*

Ο παραπάνω αλγόριθμος έτρεξε για δεδομένα με μήκος πληροφορίας $k = 20$ bits, $P = 110101$ και bit error rate $BER = 10^{-3}$. Τα αποτελέσματα που παρήχθηκαν είναι τα εξής:

- Αριθμός πακέτων: 1000 πακέτα
- Το ποσοστό των μηνυμάτων που φθάνουν με σφάλμα (στο block δεδομένων ή στο CRC) στον αποδέκτη: 2.6%
- Το ποσοστό των μηνυμάτων που ανιχνεύονται ως εσφαλμένα από το CRC: 2.5%
- Το ποσοστό των μηνυμάτων που φθάνουν με σφάλμα στο αποδέκτη και δεν ανιχνεύονται από το CRC: 0.1%

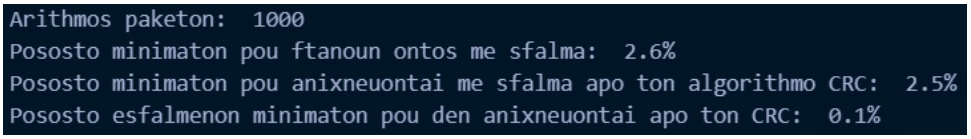

Figure 2: Έξοδος τυχαίας εκτέλεσης του προγράμματος για το δεύτερο ζητούμενο

Γενικά, ο αλγόριθμος CRC δεν ανιχνεύει όλα τα σφάλματα. Αυτό φαίνεται και στην παραπάνω εκτέλεση. Όμως, επειδή η παραγωγή των πληροφοριών είναι τυχαία, υπάρχει προφανώς μεγάλη περίπτωση να υπάρχουν εκτελέσεις όπου ο αλγόριθμος ανιχνεύει όλα τα σφάλματα. Πράγματι, αν ο αλγόριθμος τρέξει για 1000 πακέτα όπως και πάνω, σχεδόν όλες τις φορές ο αλγόριθμος θα ανιχνεύσει όλα τα σφάλματα. Μεγαλύτερος αριθμός πακέτων δίνει μεγαλύτερες πιθανότητες να μην ανιχνεύσει ο αλγόριθμος κάποιο υπάρχον σφάλμα στη μετάδοση ενός πακέτου.