

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Πληροφορικής

Ψηφιακές Επικοινωνίες

## Προγραμματιστική εργασία - Υλοποίηση του αλγόριθμου CRC

Συντάκτης:

Αλέξανδρος Κόρκος

Καθηγητές:

Νικοπολιτίδης Πέτρος  
Παπαδημητρίου Γεώργιος

# Περιεχόμενα

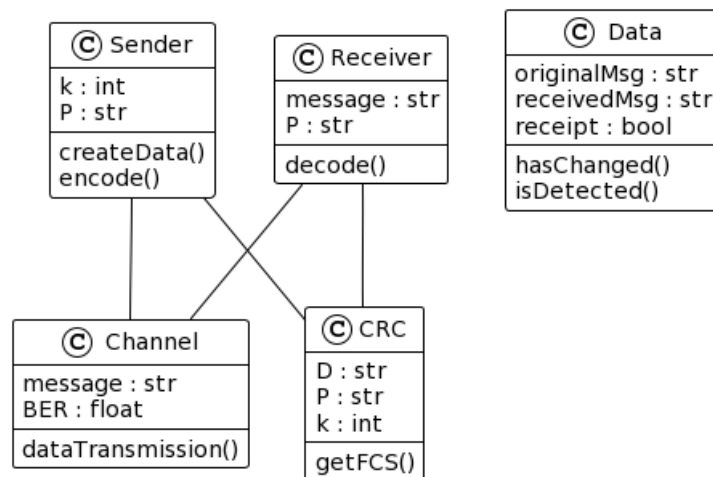
1	Υλοποίηση	2
2	Αξιολόγηση δεδομένων	6

# Κεφάλαιο 1

## Υλοποίηση

Η υλοποίηση του αλγορίθμου άλλα και του περιβάλλοντος προσομοίωσης έγινε στην γλώσσα Python.

Μια σύντομη περιγραφή των κλάσεων που χρησιμοποιήθηκαν φαίνεται παρακάτω.



Σχήμα 1.1: Αναπαράσταση UML

Η κλάση Data είναι μια βοηθητική κλάση για τον χειρισμό των δεδομένων από την προσομοίωση της αποστολής μηνυμάτων, για αυτό το λόγο δεν θα εξετασθεί σε αυτή την τεχνική αναφορά.

### Η κλάση CRC

Η υλοποίηση του αλγορίθμου CRC πραγματοποιείται στην μέθοδο getFCS() της κλάσης CRC.

```

1 def getFCS(self) -> str:
2     self.D += '0' * (self.k - 1)
3
4     copyOfD = self.D[:self.k]
5
6     def XOR(a, b) -> str:
7         ans = ""
8         for i in range(1, len(b)):
9             ans += str(int(a[i]) ^ int(b[i]))
10        return ans
11
12    i = self.k
13
14    while i < len(self.D):
15        if copyOfD[0] == '0':
16            copyOfD = XOR('0'*i, copyOfD)
17        else:
18            copyOfD = XOR(self.P, copyOfD)
19        copyOfD += self.D[i]
20        i += 1
21
22    if copyOfD[0] == '0':
23        copyOfD = XOR('0'*i, copyOfD)
24    else:
25        copyOfD = XOR(self.P, copyOfD)
26
27    return copyOfD

```

### Κώδικας 1.1: Υπολογισμός του FCS

Στην γραμμή 2 γίνεται η τοποθέτηση των  $k$  μηδενικών δεξιά του  $D$ .

Από την γραμμή 6 έως 10, υλοποιείται η συνάρτηση XOR όπου προχωράει στην πράξη xor μεταξύ δυο δυαδικών αριθμών και επιστρέφει το αποτέλεσμα της πράξης. Η συνάρτηση αυτή θα χρησιμοποιηθεί στην διαίρεση mod-2 παρακάτω.

Στις γραμμές 14 με 25, γίνεται η διαίρεση mod-2 μεταξύ του  $P$  (μήκους  $k$ ) και του αριθμού που δημιουργήθηκε στην γραμμή 2 (μήκους  $n$ ). Σε κάθε βήμα της διαίρεσης, γίνεται η πράξη xor μεταξύ ενός αντίγραφου του  $D$  (μήκους  $k$ ) και του αριθμού  $P$ . Το υπόλοιπο της πράξης, κρατιέται και του προστίθεται στο τέλος ένα bit το οποίο προέρχεται από τον αριθμό  $D$  (δηλ. κατεβαίνει ένα bit στο υπόλοιπο). Ο νέος αυτός αριθμός διαιρείται με την σειρά του με το  $P$ . Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να μην υπάρχουν άλλα ψηφία από τον αριθμό  $D$ , να κατέβουν στο υπόλοιπο.

Στο απόσπασμα του κώδικα, δεν αγνοείται  $k$  η περίπτωση όπου ο  $D$  θα έχει ένα ή παραπάνω μηδενικά στην αρχή του (βλ. γραμμή 15, 16).

## Η κλάση Sender

Στην κλάση Sender γίνεται η παράγωγή τυχαίων δεδομένων και η κωδικοποίηση τους.

```
1 def createData(self) -> str:
2     D = ""
3     for _ in range(self.k):
4         D += str(randint(0, 1))
5     return D
6
7 def encode(self) -> str:
8     D = self.createData()
9     fcs = CRC(D, self.P).getFCS()
10    return D + fcs
```

**Κώδικας 1.2:** Οι μέθοδοι της κλάσης Sender

Από την γραμμή 1 έως 5, γίνεται η παραγωγή ενός τυχαίου δυαδικού μηνύματος (μήκους k). Για αυτό το λόγο χρησιμοποιείται η μέθοδος randint για την τυχαία αναπαραγωγή είτε του 0 είτε του 1.

Στις σειρές 7 - 10, δημιουργούνται κάποια τυχαία δεδομένα (μέσω της createData()) και η ακολουθία FCS για τα δεδομένα αυτά, ενώ τέλος γίνεται η συνένωση τους και επιστρέφεται.

## Η κλάση Receiver

Η μοναδική μέθοδος αυτής της κλάσης, αποκωδικοποιεί τα δεδομένα και ελέγχει την ορθότητα τους.

```
1 def decode(self) -> bool:
2     crc = CRC(self.message, self.P)
3     if int(crc.getFCS()) == 0:
4         return True
5     return False
```

**Κώδικας 1.3:** Η μέθοδος decode()

Η παραπάνω μέθοδος εκτελεί διαίρεση mod-2 μεταξύ του κωδικοποιημένου μηνύματος (δηλ. δεδομένα + FCS) και του αριθμού P. Αν το αποτέλεσμα είναι 0 τότε σημαίνει πως το μήνυμα δεν έχει αλλοιωθεί.

## Η κλάση Channel

Η κλάση Channel αναπαριστά ενόρυβο κανάλι.

```
1 def dataTransmission(self) -> str:
2     msg = ""
3     for i in range(len(self.message)):
4         if uniform(0, 1) < self.BER:
5             if self.message[i] == "0":
6                 msg += "1"
7             else:
8                 msg += "0"
9         else:
10            msg += self.message[i]
11    return msg
```

**Κώδικας 1.4:** Η αλλοίωση των δεδομένων

Η μοναδική μέθοδος της Channel, αλλοιώνει κάποιο ή κάποια bit του μηνύματος με κάποια τυχαία συχνότητα και επιστρέφει το λανθασμένο μήνυμα.

## Κεφάλαιο 2

### Αξιολόγηση δεδομένων

Για την αξιολόγηση του αλγορίθμου CRC, χρησιμοποιήθηκε προσομοίωση με 10,000,000 τυχαίων μηνυμάτων. Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα της προσομοίωσης.

	Πληθάριθμος μηνυμάτων	Ποσοστιαίος αριθμός μηνυμάτων (%)
Αλλοιωμένα μηνύματα	247,567	2.47567
Αλλοιωμένα μηνύματα που εντοπίστηκαν	247,471	2.47471
Αλλοιωμένα μηνύματα που δεν εντοπίστηκαν	96	0.03877

**Πίνακας 2.1:** Αποτύπωση δεδομένων

Όπως φαίνεται από το πίνακα 2.1 ένα πάρα πολύ μικρός αριθμός μηνυμάτων που αλλοιώθηκαν δεν εντοπίστηκε (μόλις 96 στα 247,567). Συνεπώς, μπορεί να βγει το συμπέρασμα πως ο αλγόριθμος CRC είναι ιδιαίτερα αποδοτικός στο να εξασφαλίζει σφάλματα τα οποία συμβαίνουν κατά την μετάδοση ενός μηνύματος.

### Βιβλιογραφία

[1] Stallings W., *Επικοινωνίες Υπολογιστών και Δεδομένων*, Εκδόσεις Τζιόλα, 2011.