

Implémentation d'un Calculateur et Vérificateur de CRC

Lyes SEHILA

February 15, 2025

Contents

1	Rappel du sujet	2
2	Analyse du sujet	2
3	Choix techniques effectués	3
3.1	Langage et bibliothèques	3
3.2	Structure du programme	3
3.3	Implémentation de la division CRC	3
4	Résultats et tests	3
4.1	Captures d'écran de l'application	4
5	Conclusion	6
5.1	Difficultés rencontrées	6
5.2	Limites du programme	6
5.3	Perspectives d'évolution	6

1 Rappel du sujet

Le projet consiste à réaliser une application permettant de calculer et de vérifier un code *CRC* (Cyclic Redundancy Check). Le CRC est une méthode de détection d'erreurs largement utilisée dans le domaine des télécommunications et du stockage des données. L'objectif est de fournir une interface conviviale permettant :

- De calculer un CRC à partir d'une trame binaire et d'un polynôme générateur.
- De vérifier l'intégrité d'une trame (message + CRC) en utilisant le même polynôme générateur.
- D'afficher clairement les étapes de la division polynomiale (opération de XOR).

2 Analyse du sujet

Le sujet nécessite de comprendre les principes de base du CRC :

- **Représentation polynomiale** : Le message binaire et le polynôme générateur sont représentés sous forme de polynômes à coefficients binaires.
- **Division polynomiale en binaire** : La division polynomiale se fait via des opérations XOR (sans retenue) afin d'obtenir le reste, qui constitue le CRC.
- **Ajout de zéros** : Pour calculer un CRC, on ajoute autant de zéros que le degré du polynôme générateur à la fin de la trame avant la division.
- **Vérification** : La vérification consiste à recalculer un CRC sur le message reçu (trame + CRC). Si le reste obtenu est nul (tous les bits à 0), alors aucune erreur n'est détectée.

3 Choix techniques effectués

3.1 Langage et bibliothèques

- **Java Swing** : pour l'interface graphique (calcul et vérification du CRC, ainsi qu'une fonctionnalité de conversion d'une trame en polynôme).
- **Look & Feel Nimbus** : pour offrir une interface moderne et agréable.
- **ChatGPT** : utilisé pour améliorer le design de la fenêtre et proposer une organisation plus ergonomique du code.

3.2 Structure du programme

Le programme est divisé en deux classes principales :

- **CRC Calculator** : contient la logique de calcul et de vérification du CRC.
- **CRC Interface** : gère l'interface graphique. L'utilisateur peut y saisir la trame binaire, sélectionner un polynôme générateur ou en saisir un, et visualiser les résultats.

3.3 Implémentation de la division CRC

- La trame est convertie en une liste d'entiers (0 ou 1).
- On ajoute le nombre de zéros correspondant au degré du polynôme.
- On parcourt la liste pour trouver le premier 1, puis on applique l'opération XOR avec le polynôme générateur.
- Les différentes étapes de la division sont stockées et renvoyées pour être affichées à l'écran.

4 Résultats et tests

L'interface obtenue permet de :

- Calculer le CRC d'une trame : la fenêtre affiche la trame initiale, le polynôme utilisé, toutes les étapes de division et le **CRC** calculé.
- Vérifier un message contenant déjà un CRC : la fenêtre affiche de la même manière les étapes de division et indique si le message est correct ou erroné.

4.1 Captures d'écran de l'application

The screenshot shows a window titled "Calculateur et Vérificateur de CRC" with three tabs: "Calculer CRC", "Vérifier CRC", and "Calculer Polynôme". The "Calculer CRC" tab is active. It contains a text input field for "Message binaire" with the value "11100111". Below it, there is a dropdown menu for "Polynôme générateur (binaire)" set to "Custom", and a text input field for the polynomial value "10110". A blue button labeled "Calculer CRC" is positioned to the right of the polynomial input. The bottom section of the window displays the results of the calculation in a light blue box:

```
=== Calcul du CRC ===  
Message binaire : 11100111  
Polynôme générateur : 10110  
  
Étapes de division :  
Étape 1 : 111001110000  
Étape 2 : 010101110000  
Étape 3 : 000011110000  
Étape 4 : 000001000000  
Étape 5 : 000000011000  
Étape 6 : 000000001110  
  
CRC calculé : 1110  
Message à transmettre : 111001111110
```

Figure 1: Exemple de calcul du CRC

Calculateur et Vérificateur de CRC

Calculer CRC Vérifier CRC Calculer Polynôme

Message avec CRC : 111001111110

Polynôme générateur (binaire) : Custom
10110

Vérifier CRC

=== Vérification du CRC ===
Message avec CRC : 111001111110
Polynôme générateur : 10110

Étapes de division :
Étape 1 : 111001111110
Étape 2 : 010101111110
Étape 3 : 000011111110
Étape 4 : 000001001110
Étape 5 : 000000010110
Étape 6 : 000000000000

Résultat : Le message est correct, aucune erreur détectée.

Figure 2: Exemple de vérification du CRC

5 Conclusion

5.1 Difficultés rencontrées

- La mise en place d'une interface graphique en Java Swing demande une bonne connaissance de la disposition des composants et de la gestion des événements.
- La compréhension de la division polynomiale en binaire (basée sur l'opération XOR) peut être délicate pour les débutants.

5.2 Limites du programme

- L'interface ne gère pas la détection d'erreurs plus avancées (ex. insertion d'erreurs aléatoires).
- La gestion des trames très longues (plusieurs milliers de bits) pourrait ralentir le programme, bien qu'en pratique cela reste rarement bloquant.

5.3 Perspectives d'évolution

- Ajouter un module de **simulation de transmission** avec insertion d'erreurs aléatoires pour tester la robustesse du CRC.
- Permettre l'exportation des résultats (au format PDF, CSV, etc.).
- Améliorer encore l'interface graphique, par exemple en offrant un mode clair/sombre, ou en permettant une personnalisation avancée des couleurs.