TP 5 : détection et correction d'erreurs
Réseau M1

Ce TP est optionnel. Il est à faire chez soi et ne nécessite pas l'aide d'un enseignant car les exécutables utilisés sont exclusivement éducatifs. La note de ce TP ne sera prise en compte que si elle apporte des points supplémentaires à la moyenne.

Dans ce TP, nous vous proposons de simuler les méthodes de détections et de corrections des erreurs de transmission à partir d'exécutables de test. Des messages binaires sont générés aléatoirement puis codés. Des erreurs peuvent être simulées avant une phase de détection puis de corrections des erreurs et enfin le décodage du signal.
Les exécutables sont à télécharger sur le serveur pédagogique. Décompressez le fichier téléchargé. Les exécutions sont à réaliser à partir d'une console où vous visualiserez les résultats.

1 Contrôle de parité

Le codage par contrôle de parité ajoute un bit de parité à la fin de chaque bloc. On étudie ici une parité paire. C'est à dire que le poids de Hamming du mot de code généré est toujours paire. La commande `./parityCheck k nb_block` permet de simuler le codage et le décodage avec une dimension k d'un message de nb_block blocs.

1. Tester le codage avec plusieurs blocs pour une longueur du code de 1 octet. Quel est le rendement ?
2. Introduisez une erreur sur un bloc. L'erreur est-elle détectée ? Est-elle corrigée ?
3. Faire de même en introduisant deux puis trois erreurs dans le même bloc.
4. Que dire du message décodé ?

2 Parité longitudinale et transversale

Le codage LRC calcule la parité sur chaque ligne et colonne d'une matrice de données. La commande `./LRC width height nb_block` permet de simuler le codage et le décodage de nb_block blocs avec une matrice de *width* colonnes et *height* lignes.

1. Testez le codage avec plusieurs tailles de matrice. Que dire du rendement ?
2. Introduisez une erreur dans un bloc. L'erreur est-elle détectée ? Est-elle corrigée ? Donnez un exemple et expliquez.
3. Introduisez plusieurs erreurs dans un même bloc. Les erreurs sont-elles détectées ? Sont-elles corrigées ?
4. Que dire du message décodé lorsqu'il y a une erreur ?

3 Code de Hamming

Le codage de Hamming place les bits de redondance aux positions relatives à des puissance de deux. La commande `./Hamming n-k nb_block` permet de simuler le codage et le décodage de nb_block blocs en ajoutant $n - k$ bits de redondance.

1. Testez le codage avec plusieurs valeurs de $n-k$. Que dire du rendement ? Comparez avec le codage LRC. Que vaut la longueur du code ? Pourquoi ?
2. Pour la suite faire les tests avec $n-k=3$. Prenez un exemple et expliquez les valeurs des bits de redondance.
3. Introduisez une erreur dans un bloc. L'erreur est-elle détectée ? Est-elle corrigée ? Donnez un exemple et expliquez.
4. Introduisez plusieurs erreurs dans un même bloc. Les erreurs sont-elles détectées ? Sont-elles corrigées ?
5. Que dire du message décodé lorsqu'il y a une erreur ?

4 Code polynomial

Le codage polynomial consiste à considérer les bits d'une séquence comme les coefficients d'un polynôme. Un mot du code est ensuite reconnu lorsqu'il est multiple d'un polynôme générateur. L'émetteur et le récepteur doivent connaître le polynôme générateur.
La commande `./polynomial n k` permet de simuler le codage et le décodage d'un message de dimension d avec un code polynomial de longueur n.

1. Testez d'abord le codage avec un polynôme générateur de petite dimension (par exemple $G(x)=x^3+1$). Expliquez la méthode de codage et de détection des erreurs.
2. Pour la suite testez des messages de dimensions élevées. Prenez un polynôme générateur ayant plus d'un coefficient non nul. Introduisez une erreur. Cette erreur est-elle détectée ? Cette erreur est-elle corrigée ?
3. Prenez un polynôme générateur ayant un facteur irréductible de trois termes. Vérifiez que le code détecte une erreur double.
4. Prenez un polynôme générateur ayant (x+1) comme facteur. Vérifiez que le code détecte les erreurs d'ordre impair.

5 Délimitation de trames

Nous vous proposons ici de simuler une autre des fonctions principales de la couche liaison : la délimitation de trames. L'exécutable `delimitationTrames` simule le comportement du protocole HDLC. Le flot de données à envoyer se trouve dans le fichier `trames`.

1. Lancez la simulation avec bits de transparence. Expliquez le principe du protocole HDLC. Pourquoi et comment rajouter des bits de transparence ? Les trames reçues sont-elles bien celles envoyées ?
2. Lancez la simulation sans bits de transparence. Combien de trames sont envoyées ? Combien de trames sont reçues ? Combien de bits de transparence étaient nécessaire dans le point précédent. Expliquez.