

Devoir 1 DU COURS IFT **585**

*TITRE*

f**onctionnement d’un protocole bidirectionnel à reprises multiples (sélectives)**

|  |  |
| --- | --- |
| *PAR*  BILODEAU CEDRIC 15070012 CISSE OUSMANE 16154215 COMEAU SAMUEL 15077907 NGUIFFO MAMNO PATRICIA 16155893 ROY MICHAEL 14088275  **TABLE DES MATIÈRES**  1. Mse en Contexte 3  2. Contenu du zip 4  3. Utilisation 5  3.1 [Méthode de compilation et d’exécution 3](#_Toc484979412)  3.2 Le fichier de paramètres 6  3.3 Le fichier d'erreur 6  4. Etapes de conception Explication du programme 6  4.1 Main  4.2 fonction EmetteurRecepteur 6  4.3 Fonction supportTransmission…………………………………………………………………………6  4.4 Class TamponCirculaire  4.5 Class Trame  4.6 NameSpace Hamming  4.7 Class SortieFichier et Entree Fichier  5. Etapes de conception  6 Degre de Fonctionnement | *Professeur***:**  **Bessam Abdulrazak** |

# Mise en contexte

Dans le cadre de notre premier devoir de télématique portant sur le fonctionnement d’un protocole bidirectionnel a reprises multiples, nous avons implémenter en C++ le rejet sélectif comme Protocol de liaison, l’algorithme de Hamming pour la correction/détections des erreurs et enfin la gestion de l’insertion des erreurs par l’utilisateur.

# Contenue du zip

Notre archive zip contient l’intégralité de notre devoir ; il est principalement constitué de :

-**Constantes.h** contenant une définition des différents types de trames utilisés (ACK, NACK, DONNEE, VALIDATED) et une définition des principales constantes.

-**Hamming.h** et **Hamming.cpp** contiennent l’implémentation du code de Hamming en vue de la détection/correction d’erreur.

- **Trame.h** et **Trame.cpp** permettent la mise au point des trames qui feront l’objet de l’envoie et de la réception entre les couches liaisons.

-**TamponCirculaire.cpp** et **TamponCirculaire.h** matérialisent la classe contenant toutes les fonctionnalités nécessaires au fonctionnent du protocole de rejet sélectif.

-**parametres.txt** est un fichier contenant les paramètres nécessaires pour a l’exécution des protocoles implémentés dans ce devoir.

-**erreurs.txt** pour l’insertion d’erreurs

- **EntreeFichier.h** , **EntreeFichier.cpp**, **SortieFichier.h** et **SortieFichier.cpp** permettent l’ecriture et la lecture du fichier; nous les avons utilisés pour matérialiser la couche reseau.

-**IFT585-TP1.cpp** contient le main et les fonctions principales comme EmetteurRecepteur…

-**VectorCircilaire.h** et **VectorCirculaire.cpp**

-**Destination.txt** , **fichier.txt** et **Source.txt**

# Utilisation

## 3.1 Méthode de compilation et d’exécution

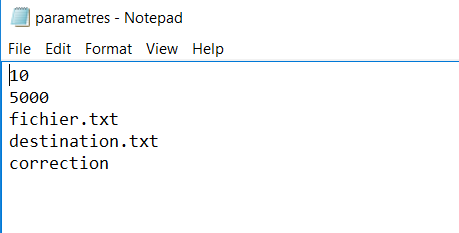
Ayant programmer en C++, nous avons utilisé Visual Studio comme environnement de développement. Ceci étant, nous fournissons un programme déjà compilé à l’aide de Visual Studio.

Pour ce qui est de l’exécution, il suffit simplement d’exécuter le .exe fournie.

## 3.2 Le fichier de paramètres

Ce fichier doit se nommer « parametres.txt ». Le fichier de paramètre est composé de 5 paramètres. Chaque paramètre prend une ligne. Le premier paramètre est la taille des tampons en nombre de trame qui peuvent prendre. Le second est le délai de temporisation en millisecondes. Le troisième est le chemin du fichier à copier. Le quatrième est le chemin de destination pour le fichier à copier. Le dernier est si le programme roule en mode détection d’erreur ou en mode correcteur d’erreur avec Hamming. Pour être en mode correction, la ligne doit exactement avoir les caractères suivant : « correction », sinon il est en mode détection automatiquement.

Exemple



## 3.4 Le fichier d’erreurs

Dans ce fichier, chaque ligne représente une nouvelle erreur. en effet, ce fichier utilise une syntaxe X : Y ou X représente le numéro de la trame et Y le numéro du bit qui requière une modification dans la trame.

# Explication du programme

## Main

Le main ne fait que lire le fichier de paramètre et faire partir les trois thread. Il y a deux thread qui roulent sur la fonction (emetteurRecepteur) et une autre qui représente la couche physique (supportTransmission).

## Fonction emetteurRecepteur

Cette fonction est l’implémentation concrète du rejet sélectif vue dans le cours. En effet, elle s’occupe de la gestion de l’envoie et de la réception des messages entre émetteur et récepteur ; elle permet principalement de faire fonctionner le rejet sélectif pour que l’envoie et la réception des messages se fassent correctement en utilisant les tampons.

## Fonction supportTransmission

C’est une fonction dont le rôle principal est de passer des données entre le thread émetteur et le thread récepteur ; elle matérialise la couche physique.

## Class TamponCirculaire

Class gérant l’ensemble des fonctionnalités du tampon nécessaire au rejet sélectif. Il utilise la class trame.

## Class Trame

Class représentant notre trame. Celle-ci est composée de 3 bits pour numéro de séquence, de 2 bits pour le type et de 16 bits pour les données. Il y a aussi 5 autre bits pour Hamming. Cette trame est contenue dans un uint32\_t, mais il y a quelques bits qui ne seront pas utilisés.

## Namepace Hamming

Le namespace Hamming comprend l’ensemble des fonctions nécessaires pour l’encodage, le décodage, la détection et la correction de la trame.

## Class SortieFichier et EntreeFichier

Ces deux class permettent l’écriture et la lecture de fichier ; ils matérialisent la couche réseau.

# Étapes de conception

Pour la conception de devoir, nous avons commencé par faire toutes les class qui seront utile à la conception du rejet sélectif. Nous avons implémenté Hamming au début aussi. Par la suite, nous avons intégré les différentes class. nous avons successivement essayé de faire la synchronisation, mais malheureusement celle-ci était mal faite. Nous avons continué en codant la fonction émetteur-récepteur. Finalement, nous avons fait beaucoup de débogages, car nous avions beaucoup de bug. Au début, les bugs étaient dans la synchronisation par la suite dans le code de l’émetteur-récepteur et dans le tampon.

# Degré de fonctionnement

Dans ce devoir, l’implémentation du code de Hamming fonctionne très bien, l’insertion d’erreur par l’utilisateur est également fonctionnelle cependant, le protocole de rejet sélectif ne fonctionne pas dans son intégralité