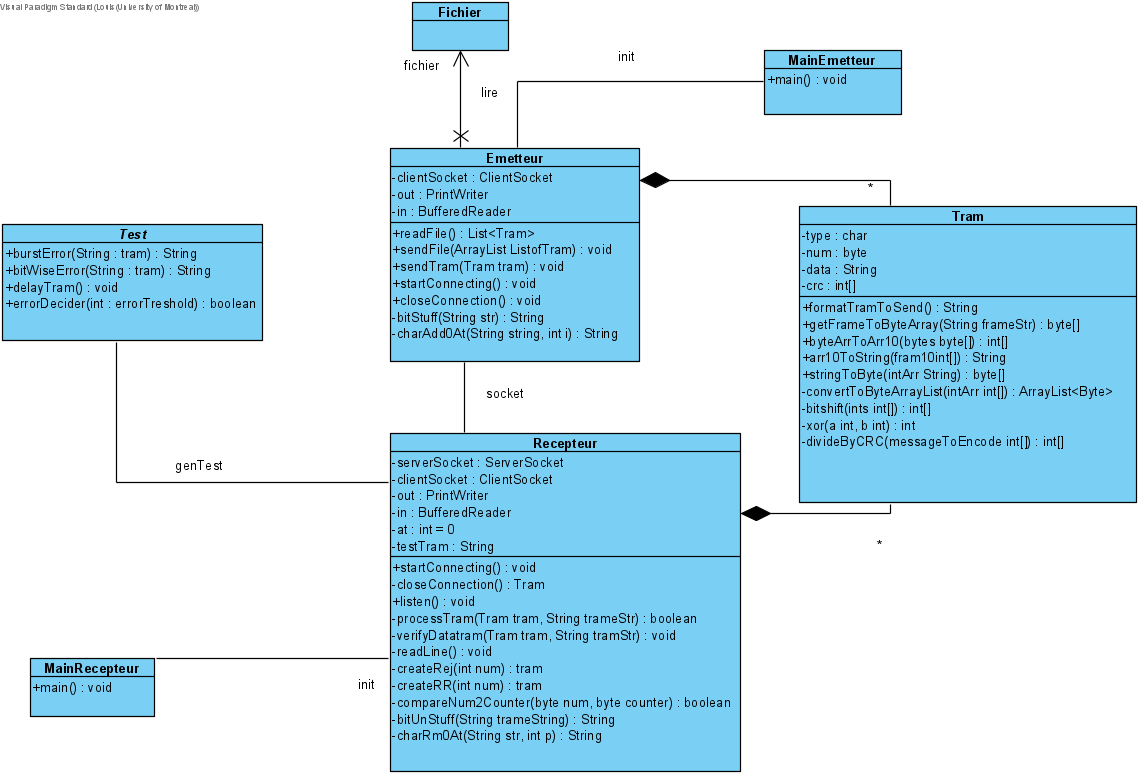
IFT3325-Téléinformatique-TP2

Louis Bertrand : 20141254

Ronald Gedeon : 20142399

Diagramme de classes :



Classes)

Classe : Emetteur

Emetteur est la classe qui lit le fichier et le transforme en trame pour l’envoyer vers la classe Recepteur

Attributs :

clientSocket : Le socket utilisé pour communiquer avec le Recepteur.

out : Le PrintWriter utilisé pour écrire les trames.

in : Le BufferedReader utilisé pour lire les trames envoyées par Recepteur.

Méthodes :

readFile() :

Prends un String comme nom de fichier à ouvrir et retourne une liste de trames correspondant aux lignes du fichier.

sendFile(ArrayList ListofTram) :

Prends une liste de trame comme paramètre et les envoie une par une à Recepteur en appelant sendTram().

sendTram(Tram tram):

Prends une trame en paramètre et l’envoie à Recepteur.

startConncetion() :

Établit la connection entre Emetteur et Recepteur.

closeConnection() :

Ferme la connection entre Emetteur et Recepteur.

bitStuff(String str):

Prends un string de 0 et de 1 et introduit du bit stuffing en ajoutant un 0 aux suites de cinq 1 consécutifs en appelant charAdd0Att().

charAdd0At(String string, int i):

Prends un String et un int et ajoute un 0 à la position indiquée et retourne le String modifié.

Classe : Recepteur

Recepteur est la classe qui s’occupe de faire la réception des trames produites et envoyées par Emetteur.

Attributs :

serverSocket : Le socket utilisé pour établir la connexion avec Emetter

clientSocket : Le socket utilisé pour communiquer avec Recepteur.

out : Le PrintWriter utilisé pour écrire les trames.

in : Le BufferedReader utilisé pour lire les trames envoyées par emetteur.

at : L’Integer utilisé pour savoir le numéro de la trame reçu. Il ne s’agit pas de la position de la trame dans la fenêtre, mais bien du numéro absolue de la trame dans l’ordre d’envoie.

testTram : Un String indiquant le type de test qui doivent être effectués sur la communication. Peut prendre 5 valeurs différentes entre “BIT”, “BURST”, “DELAY”, “LOSS” et “ERRORLESS”.

Méthodes :

startConnecting():

Établit la connexion avec Emetteur.

closeConnection() :

Ferme la connexion avec Emetteur.

listen():

Cette fonction est le cœur de Recepteur. Elle prend les Strings envoyés par Emetteur une par une, enlève le bit stuffing, introduit des erreurs si nécessaire, les transforme en objets de type Tram et les passe à processTram() pour les traiter. Elle s’arrête lorsque processTram() retourne false.

processtram(Tram tram, String trameStr) :

Prend une trame et son string associé en paramètre. Elle prend une trame, et dépendamment du type de trame reçue, fait une opération différente :

‘C’: Envoie une trame confirmant l’utilisation de Go-Back-N

‘I’: Vérifie l’intégrité de la trame d’information et la traite.

‘P’: Retourne une trame RR à Emetteur pour lui indiquer quelle trame

il devrait lui envoyer.

‘F’: Ferme la connexion entre Emetteur et Recepteur.

verifyDataTram(Tram tram, String tramStr) :

Vérifie l’intégrité de la trame d’information passée en paramètre. Si le CRC de la Trame est invalide, une trame REJ est envoyée à Emetteur. Si le numéro de la trame n’est pas le successeur immédiat de la dernière trame acceptée, une trame REJ est aussi envoyée. Si le nombre de trames reçues et acceptées remplie la fenêtre, alors une trame RR est envoyée.

readLine():

Lit les string envoyées par Emetteur par le BufferedReader et les ignores jusqu’à ce qu’on arrive à la dernière trame acceptée. Cela permet d’ignorer les trames qui suivent une trame rejetée à cause d’une erreur de CRC ou une trame perdue.

createRej(int num) :

Crée une trame de rejet dont le numéro de rejet est la variable num.

CreateRR(int num) :

Crée une trame accusé de réception dont le numéro de trame est la variable num.

compareNum2Counter(byte num, byte counter) :

Compare un numéro de trame et un compteur de position de trame. Retourne true si ils sont égaux, false sinon.

bitUnStuff(String trameString) :

Prends un string de 0 et de 1 enlève le bitstuffing en enlevant les 0 qui suivent une série de cinq 1. Il s’agit de l’inverse de la méthode bitStuff() de la classe Emeteur.

charRm0At(String str, int p) :

Prends un string de 0 et de 1 et enlève la charactère à la position p et retourne le nouveau string. Il s’agit de l’inverse de la méthode charAdd0At() de la classe Emetteur().

Classe : Tram

Tram est la classe qui sert à représenter les trames qui seront envoyées entre Emetteur et Recepteur.

Attributs :

type : Un charactère représentant le type de la trame. Il peut prendre les valeurs suivantes :

‘I’: Frame d’information

‘C’: Demande de connexion

‘A’: Accusé de réception (REJ)

‘R’: Rejet de trame (RR)

‘F’: Fin de communication

‘P’: Ping

num : Un byte indiquant le numéro de la trame.

data : L’information contenue dans la trame. Elle provient du fichier lu par Emetteur

CRC : Le CRC utilisé par la trame pour sa vérification d’intégrité. Stockée dans un tableau de int.

Méthodes :

formatTramToSend() :

Retourne un String qui représente le contenu de la trame.

getFrameToByteArray(String frameStr) :

Prends un string représentant la frame et la convertit en un tableau de bytes, chacun représentant un charactère du string.

byteArrToArr10(byte[]) :

Prends un tableau de bytes et le transforme en un tableau de int prenant tous soit la valeur 0 ou 1 puis le retourne.

arr10ToString(fram10int[]) :

Prends un tableau de int prenant tous la valeur 0 ou 1, le convertit en en tableau de bytes, puis le retourne.

stringToByte(intArr String) :

Prends un string composé de 0 et de 1 et le convertit en un tableau de bytes qui leurs sont associés puis le retourne.

convertToByteArrayList(intArr int[]) :

Prends un tableau de int prenant tous la valeur 0 ou 1 et le convertit en un tableau de bytes qui lui sont associés puis le retourne.

bitshift(ints int[]):

Prends un tableau de int et les décalent vers la gauche et rajoute un 0 à l’extrême gauche puis retourne le nouveau tableau de int.

xor(a int, b int) :

Prends deux int prenant les valeurs 0 et 1 et retourne le resultat de l’opérande XOR sur ces deux valeurs.

divideByCRC(messageToEncode int[]) :

Prends un tableau de int prenant les valeurs 0 ou1 et retourne le résultat de la division du message par le CRC en utilisant la division bit à bit.

Classe : Test

La classe Test est connectée au Recepteur et est appelée pour générer des erreurs dans la transmission des données en introduisant des erreurs sur les bits, en perdant et en délayant des trames.

Méthodes :

burstError(String : tram) :

Prend un string se 1 et de 0 et inverse une suite de bits parmi ceux-ci pour simuler une erreur en rafale puis retourne le string modifié.

bitWiseError(String : tram) :

Prend un string de 0 et de 1 et inverse un nombre aléatoire de bits puis retourne le string modifié.

delayTram() :

Délaie l’arrivée d’une trame pour cinq secondes

errorDecider(int : errorTreshold) :

Prend une valeur entre 1 et 10 et détermine aléatoirement si une erreur devrait être introduite à la trame. Retourne true si une erreur devrait être introduite, false sinon.

Classe : MainRmetteur

C’est la classe à appeler en premier. Elle permet la décision du type de tests effectués et établit la connexion entre Recepteur et Emeteur avant d’appeler la méthode listen() de Recepteur.

Classe : MainEmetteur

C’est la classe à appeler en deuxième. Elle établit la connexion entre Recepteur et Emetteur et c’est à partir de là que la trame d’accord sur l’utilisation de Go-back-N est envoyée.