CORRE Mélanie
CHOVELON Pierrick
FIP 2A
PROMO 2017

SIT 213
Atelier Logiciel

Rapport

Formation par partenariat

SIT 213 – S4

Année scolaire 2015-2016

TELECOM Bretagne

Sommaire

1. INTRODUCTION

2. TRAVAIL DEMANDE

2.1 ETAPE 1

Après avoir compris ce qui nous est demandé et après avoir étudié la structure du projet (classes, héritage, ...) nous avons commencé l'étape 1 du projet. Celle-ci consiste à développer la **transmission élémentaire « back-to-back »**. Celle-ci correspond au modèle de transmission basique qui comporte une source, un transmetteur logique parfait, une destination ainsi que deux sondes logiques. La source devra émettre une séquence booléenne fixe ou aléatoire. Le transmetteur quant à lui, ne fera que transmettre le signal à la destination sans le modifier (car transmetteur parfait). La destination recevra alors ce signal. Les deux sondes logiques permettent quant à elles de visualiser le signal en sortie de la source (sonde 1) et en sortie du transmetteur (sonde 2).

La méthode principale du projet est le **main** de la classe **Simulateur**. Dans celui-ci se trouve la déclaration d'un objet **Simulateur** et l'appelle aux différentes méthodes utilisées **(.execute()**, **.calculTauxErreurBinaire()**, ...).

Nous avons choisi de déclarer et d'instancier tous les objets de la chaîne de transmission dans le constructeur de **Simulateur**. Le constructeur appelle la méthode **analyseArguments** pour mettre à jour les attributs de l'objet selon les paramètres passés dans la ligne de commande. Selon la valeur de l'attribut **seed**, nous déclarons soit une **SourceFixe**, soit une **SourceAleatoire**. Les sondes sont ensuite connectées aux bons éléments.

Dans cette première étape, il nous était demandé de développer les deux types de sources (Fixe et Aléatoire), ainsi que d'écrire le code de la méthode qui permet de calculer le taux d'erreur binaire.

Pour la source fixe, le code est relativement simple. La suite de bits passée en paramètre est analysée par le constructeur de **SourceFixe**. Celui-ci, selon le caractère (1 ou 0), remplit les cases d'un tableau de **Boolean**. Ce tableau est ensuite utilisé lors de l'instanciation d'un objet **Information** qui correspond à l'attribut **informationGeneree** de la source.

Pour la source aléatoire, le code diffère légèrement. En effet, l'option **–mess** correspond maintenant au nombre d'éléments que la source aléatoire doit générer de manière aléatoire. Le constructeur de **SourceAleatoire** remplit, grâce à un objet **Random** et à la méthode **nextBoolean()**, un tableau de **Boolean** qui constituera les données à transmettre. L'objet **Random** utilise le **seed** (graine) passé en paramètre pour initialiser le générateur aléatoire. Le tableau est ensuite utilisé lors de l'instanciation d'un objet **Information** qui correspond à l'attribut **informationGeneree** de la source.

La méthode **calculTauxErreurBinaire** récupère l'information émise par la source et l'information reçue par la destination. Une boucle est alors utilisée pour comparer chacun des caractères envoyés et reçus. S'il y a une différence entre les deux, la variable **nbBitsErrones** est incrémentée. Le **TEB** est ensuite calculé en faisant le rapport entre le nombre de bits erronés et le nombre de bits total du message envoyé.

Vérifications bon fonctionnement:

Nous avons vérifié que les sondes affichaient de bonnes courbes avec plusieurs tests.

1. Ligne de commande : `java Simulateur -mess 10101010`

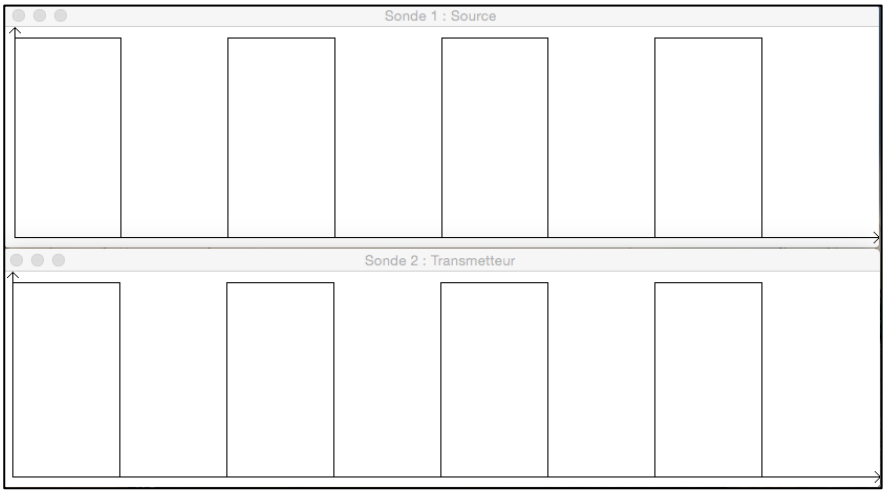

Figure 1 : Résultats obtenus (-mess 10101010)

Les courbes des sondes 1 et 2 sont identiques. Le caractère parfait du **Transmetteur** est bien mis en évidence.

2. Ligne de commande : `java Simulateur -mess 10 -seed 3`

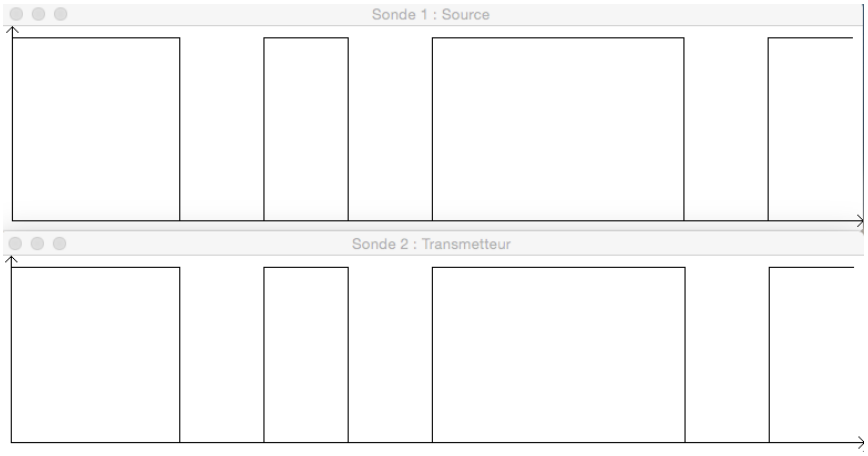

Figure 2 : Résultats obtenus (-mess 10 -seed 3)

Les courbes obtenues sont identiques. Le caractère parfait du **Transmetteur** est également visible dans le cas d'une génération aléatoire. Il y a bien 10 symbole de générés.

Dans les deux cas, le TEB était de 0. Ce qui correspond bien à un **Transmetteur** parfait.

```
java  Simulateur  -mess  10  -seed  3    =>   TEB : 0.0
```

Figure 3 : Résultat TEB du deuxième test

Enfin, il nous a été de créer deux scripts permettant :

1) de compiler le projet et de générer la javadoc
2) de lancer l'application sur un jeu de tests.

C'est ce que nous avons fait avec le script `compilejavadoc.sh` et `starttest.sh`. Un fichier README est également présent pour plus d'explications.

2.2 ETAPE 2

2.4 ETAPE 3

2.6 ETAPE 4

2.8 ETAPE 5

2.10 ETAPE FINALE

3. CONCLUSION

www.telecom-bretagne.eu

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
Tél. : + 33 (0)2 29 00 11 11
Fax : + 33 (0)2 29 00 10 00

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
Tél. : + 33 (0)2 99 12 70 00
Fax : + 33 (0)2 99 12 70 19

Campus de Toulouse
10, avenue Edouard Belin
BP 44004
31028 Toulouse Cedex 04
France
Tél. : +33 (0)5 61 33 83 65
Fax : +33 (0)5 61 33 83 75

TELECOM Bretagne