



UNIVERSITÉ DE POITIERS,

Master 1 Réseaux Télécommunications Multimédia et Automatique

Compte rendu de Travaux pratiques
Système de Transmission Multimedia
Du 26 Mars au 14 Avril 2015

Transmission Multimedia sur canaux extrêmement sensibles (Qualité et sécurité)



Groupe d'étudiants: Guy-Florent SADELER
Thomas LE BRIS

Responsable pédagogique: Clency PERRINE

Contents

List of Figures	iii
List of Tables	1
Introduction	2
1 Transmission d'un fichier binaire	3
1.1 Résultats de la transmission sans distorsion	3
1.2 Correction de la transmission sans distorsion	3
2 Conversion d'un fichier binaire en un fichier texte	5
3 Application end	8
3.1 Notions sur les formats étudiés	8
3.2 Transmission des fichiers	8
3.2.1 Transmission sans erreur	10
4 Tableau de synthèse des résultats obtenus	11
Conclusion	12

List of Figures

1.1	Illustration du retard des filtres	4
2.1	TEB = 0.0057	6
2.2	TEB = 0.5	6
3.1	Image jpg choisie	9
3.2	Image ppm choisie	9
3.3	Image jpg transmise sans erreur	10
3.4	Image bmp transmise sans erreur	10

List of Tables

Avant-propos

Objectifs

Les objectifs de cette séance de travaux pratiques est l'étude d'une chaîne de transmission en s'appropriant l'utilité de ses des différents blocs.

Il s'agit de transmettre une information quantifiée à travers une chaîne puis d'observer la distorsion.

Pour cette étude, nous avons utilisé **ADS** (Advanced Design System) et **Matlab**.

Introduction

Au cours de cette année académique, nous avons pu étudier certains concepts de transmission de données, de compression et quantification. La thématique étant: la Transmission Multimedia sur canaux extrêmement sensibles (Qualité et sécurité).

Cette étude est un excellent moyen pour renforcer ces connaissances acquises et avoir une meilleure vision de la conception des fichiers.

Chapter 1

Transmission d'un fichier binaire

On insère un fichier txt sur la chaîne de transmission dans des conditions favorables, c'est à dire sans distorsion d'information.

1.1 Résultats de la transmission sans distorsion

En observant les fichiers file.txt (de départ) et print.txt (d'arrivée), on constate :

- Les données obtenues en arrivée sont totalement déformées du fichier de base malgré qu'il s'agit d'une transmission sans bruit, les fichiers devant donc être identiques;
- Le fichier de départ transmis par l'émetteur comprend des « 0 » et des « 1 » (codage RZ) tandis que le fichier d'arrivée comprend des valeurs flottantes de -1.0 et 1.0 (codage NRZ en flottant);
- Il y a un retard (délai) de 16 bits;

1.2 Correction de la transmission sans distorsion

Afin d'obtenir des fichiers similaires pour poursuivre notre étude en sachant que tous les paramètres y sont convenablement fixés, nous procédons à une correction des erreurs du fichier d'arrivée.

- Dans un premier temps, nous avons décidé d'insérer un bloc après le démodulateur qui convertirait les valeurs flottantes obtenues en entiers. Malgré cela, nous avons un fichier de départ en RZ et un en arrivée en NRZ, on en déduit que cette erreur provient du démodulateur. Car, il est chargé de retranscrire le fichier envoyé avec des niveaux de tensions données. Pour corriger cela, nous changeons donc son seuil de codage;
- Toutefois, il subsiste un retard (délai) de 16 bits, retard dû aux deux filtres sur chaque ligne disposés en série, chacun d'eux insérant un retard de 8 bits à raison d'un bit pour 8 échantillons; 1 bit va être codé avec 8 échantillons ceci en sachant que nous avons un retard de 128 échantillons pour chacune des voies I et Q. Notons que la taille des filtres est représentatives de chaque retard, deux filtres de taille 64 sur chaque ligne, soient 128 échantillons de retard pour chaque voie, 16 bits.

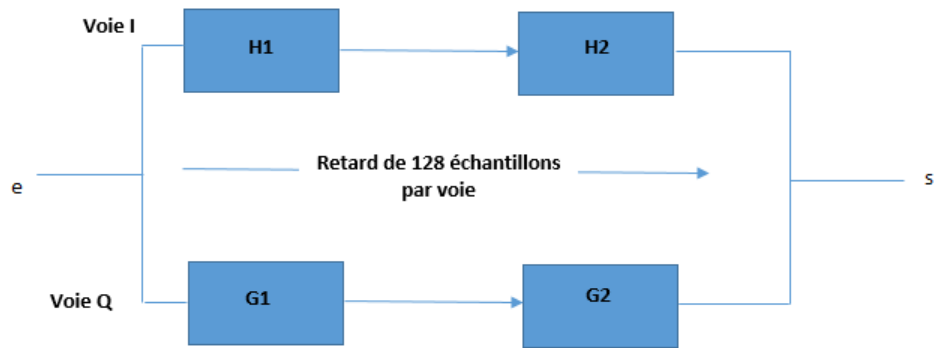


Figure 1.1: **Illustration du retard des filtres**

Le calcul du TEB est bien évidemment déterminé en prenant en compte ce retard, c'est ce qui nous a permis d'en arriver à cette déduction et à retrouver les bits correspondants à ceux du fichier de départ.

Chapter 2

Conversion d'un fichier binaire en un fichier texte

On souhaite dans cette partie transmettre un fichier binaire quelconque sur la chaîne de transmission.

Soit un fichier texte avec le nom "Thomas Le Bris" dedans, son code ASCII en décimal sera :

84 104 111 109 97 32 76 101 32 66 114 105 115

En ouvrant le fichier dans Matlab à l'aide de la fonction précédente nous obtenons bien le code prévu :

```
A =  
  
    84  
   104  
   111  
   109  
    97  
   115  
    32  
    76  
   101  
    32  
    66  
   114  
   105  
   115
```

Pour obtenir le code ASCII il nous suffit d'écrire Master 1 RTMA dans le fichier texte puis de l'ouvrir dans Matlab :

```
A =
    77
    97
   115
   116
   101
   114
    32
    49
    32
    82
    84
    77
    65
```

ADS nécessitant un fichier texte ne comportant que des '0' et '1' formatés en une seule colonne. Nous allons utiliser Matlab afin d'effectuer la conversion d'un fichier texte en binaire puis son formatage approprié.

cf fichier joint conv2ADS.m

ADS nous renvoyant en sortie un fichier formaté de la même façon il nous réutiliserons Matlab afin de reconvertir le fichier traité par la chaîne de transmission dans son format original.

cf fichier joint ADS2ASCII.m

En transmettant le fichier dans des conditions favorables nous obtenons bien un fichier identique mais en augmentant le bruit des erreurs apparaissent sur le fichier en sortie :

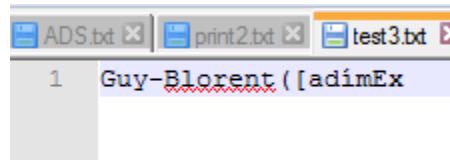


Figure 2.1: TEB = 0.0057

Jusqu'à devenir complètement différent du fichier original avec un TEB égal à 0.5 :

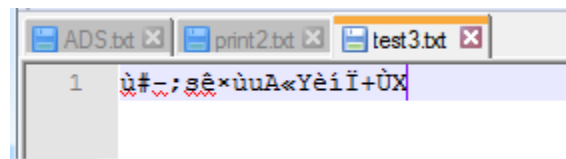


Figure 2.2: TEB = 0.5

Cela est dû au fait que le bruit transforme des 0 en 1 et des 1 en 0 ce qui fausse donc la conversion lors du retour en ASCII.

Chapter 3

Application end

Sommaire

.....	2
.....	2
.....	2
.....	2

3.1 Notions sur les formats étudiés

Nous choisissons de décrire les deux formats que nous allons utiliser par les définitions suivantes:

PPM est un format de fichier image selon (Wikipedia) « Ce format de fichier est utilisé pour des images couleur. Chaque pixel est codé par trois valeurs (rouge, vert et bleu). Comme le format PGM, en plus des caractéristiques de largeur et hauteur, une valeur maximale utilisée pour coder les niveaux de couleur ; cette valeur doit être inférieure à 65536. » Sa structuration binaire est toujours définie selon la même source comme comprenant un header pour un fichier binaire ou ASCII comme suit: « Un fichier ppm binaire a pour nombre magique P6. Chaque pixel est codé sur 1 ou 2 bytes selon que la valeur maximale soit inférieure ou supérieure à 256. » (bytes=octets) Le header est donc défini selon l'information transportée.

JPEG (acronyme de Joint Photographic Experts Group) est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe. (Wikipedia)

La différence entre JPEG et PPM est que le premier format utilise un codage entropique tandis que le second utilise un codage fixe pour la quantification.

3.2 Transmission des fichiers

Nous faisons transiter deux images, codées par les différents formats que nous avons précités, sur notre chaîne pour en évaluer la dégradation. Le choix de deux formats, nous permet de constater les erreurs de transmission typiques à chaque format ou du moins l'impact des erreurs de transmission sur chacun des formats(leur sensibilité).

Voici les images choisies pour chaque format:

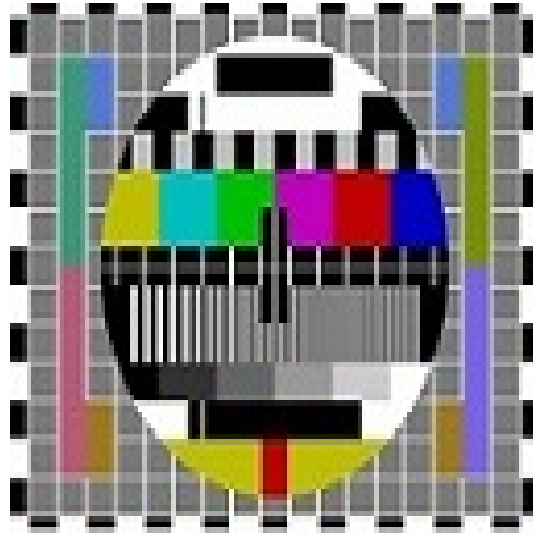


Figure 3.1: Image jpg choisie
<http://www.google.fr/>

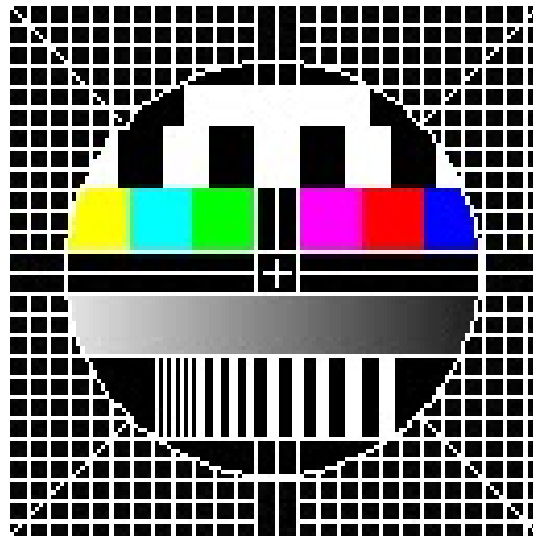


Figure 3.2: Image ppm choisie
<http://www.google.fr/>

On constate bien que les images ont une représentation différente selon chaque format. Par, exemple les dimensions (128 x 128 pour jpeg et 190 x 143 pour bmp) et tailles (14Ko pour jpeg et 10Ko pour bmp) de l'image sont différentes. En général, le codage bmp donne une image plus lourde car il s'agit d'un codage fixe, la conversion de l'une ou l'autre de ces images, nous aurait permis de vérifier cela. Ceci s'explique également par le fait que chaque représentation de format est inextricablement liée au codage qui lui est propre.

Notons que pour atténuer l'erreur de transmission, nous augmentons le paramètre AddNDensity du générateur de bruit. Nous présentons dans ces différentes sections, nos résultats de transmission obtenus.

3.2.1 Transmission sans erreur

On règle le générateur de bruit en fixant son paramètre AddNDensity à -65 dB et on arrive à minimiser toute l'erreur de transmission. De cette valeur à -70 dB, nous n'avons pas de dégradation et un faible taux d'erreur binaire (BER). On injecte les fichiers binaires obtenu pour chaque image puis on obtient :

Nous obtenons les images suivantes:

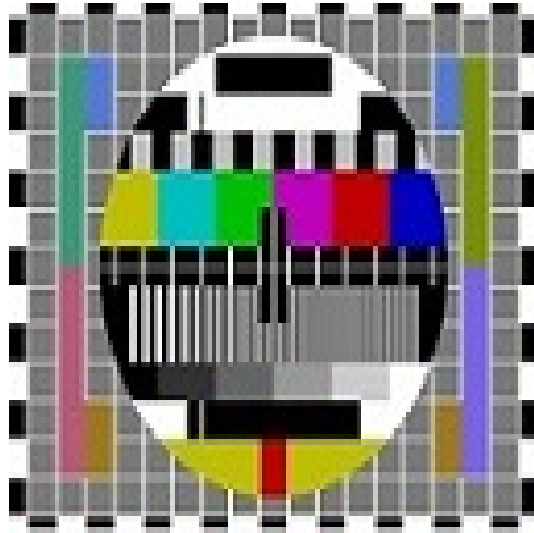


Figure 3.3: Image jpg transmise sans erreur

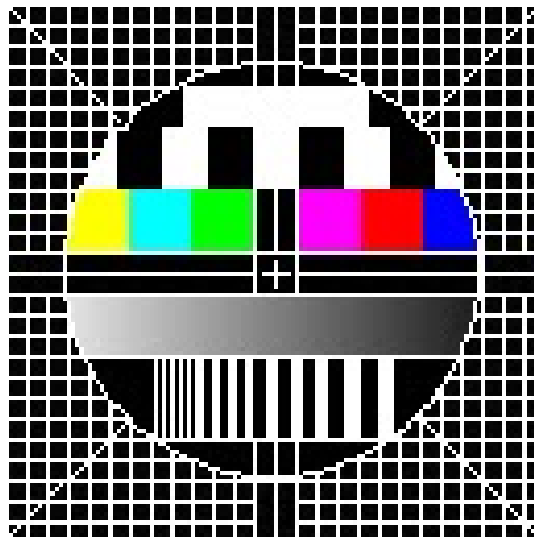


Figure 3.4: Image bmp transmise sans erreur




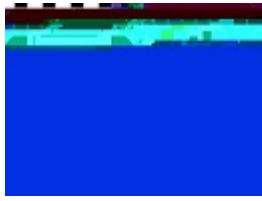


Comme attendu, il n'y a d'erreur visible sur cette image. Les images sont très identiques à celles de départ. Ceci avec un BER de $9.745 * 10^{-6}$ pour JPEG et $4.295 * 10^{-5}$ pour BMP. Le premier étant donc plus faible.

Chapter 4

Tableau de synthèse des résultats obtenus

Sommaire

1.1 Résultats de la transmission sans distorsion	3
1.2 Correction de la transmission sans distorsion	3
.	3

	JPEG	BMP	MP4
fichier original			
TEB_1	2.8E-5	1.5E-4	2.1E-5
TEB_2	1.5E-4	0.001	0.004
$TEB_1 < TEB < TEB_2$			
Robuste ?	non	oui	non
Type d'erreurs constatées	Effets blocs, désynchronisation	Erreurs pixels	Effets blocs, désynchronisation

Conclusion

Sommaire

1.1	Résultats de la transmission sans distorsion	3
1.2	Correction de la transmission sans distorsion	3
	3

A l'issu de cette séance, nous avons une meilleure vision de la caractérisation d'un fichier et de la transmission des données sur une chaine de transmission.

Ceci est un plus en terme de compétences à notre formation, en ce sens que nous apprenons par ces travaux à caractériser les données numériques qui sont binaires et mieux connaitre les formats et les erreurs inhérentes.