

Contrôle écrit
COMMUNICATIONS NUMERIQUES

--- corrigé ---

Partie A : Questionnaire (1 pt par question, ½ si incomplet)

1. *L'opération de codage numérique d'une information numérique crée une fonction discontinue dans le temps comme dans les valeurs. Toutefois, lors du stockage sur un support ou de la transmission sur un canal, cette fonction devient (ou doit devenir) continue à variations continues. Pourquoi ?*

Tout système physique possède une « vitesse » limite de réaction à une action extérieure, accompagnée d'un comportement particulier au système face à cette action (dit « réponse » du système). Cette propriété se traduit, dans le domaine temporel, par une réponse à des discontinuités sous la forme d'une fonction continue à variations continues. Dans le domaine fréquentiel, cette propriété se traduit par le phénomène dit de « filtrage ». Le stockage ou la transmission s'appuyant sur des systèmes physiques entraînent donc cet type de transformation.

2. *La transformation de cette fonction discontinue en fonction continue ne peut pas se faire sans condition(s). Pourquoi et quelle(s) condition(s) ?*

A la lecture (stockage) ou à la réception (transmission), il est nécessaire d'opérer à une régénération de manière à retrouver l'information numérique initiale. Il ne faut donc pas que la transformation (filtrage) puisse provoquer des erreurs de régénération. Ceci se traduit par une satisfaction aux critères de Nyquist (absence d'interférences inter symboles).

3. *Le codage numérique d'une information binaire peut amener à une fonction discontinue de base à 2 niveaux, trois ou un plus grand nombre de niveaux . Donner un exemple de chaque cas et expliquer les avantages ou inconvénients de chacune des configurations.*

Fonction à deux niveaux : exemple NRZ unipolaire ('0' = 0 et '1' = +V), 1 bit = 1 valeur, avantage de la relation directe car « codage de base », inconvénient de l'encombrement spectral non optimisé, inconvénient de la non distinction entre bit '0' transmis ou absence de transmission, inconvénient de l'horloge non transmise explicitement

Fonction à trois niveaux : exemple NRZ antipolaire ('0' = -V et '1' = +V), ou RZ bipolaire ('0' = -V puis 0 et '1' = +V puis 0), avantage de la distinction entre message et absence de transmission, avantage de transmission explicite de l'horloge, inconvénient de l'encombrement spectral non optimisé (pire pour RZ)

Fonction à quatre ou plus niveaux : exemple 2B1Q ou en général nB-mX (codage arithmétique), avantage de la réduction de débit de symbole ce qui optimise d'occupation spectrale, inconvénient de la multiplicité et la proximité des niveaux qui accroît la probabilité d'erreurs.

4. *Dans quelles situations est-on amené à effectuer une modulation numérique ? Quel est l'avantage de faire le choix d'une modulation combinée d'amplitude et de phase plutôt qu'une simple modulation d'amplitude ou qu'une simple modulation de phase, à nombre d'états comparables ?*

La modulation numérique multi-états peut être utilisée pour obtenir une fonction dont la bande de fréquence occupée est inférieure à la bande qui serait occupée si la fonction numérique n'était pas modulée (et donc rendre possible une transmission sur un canal donné)

La modulation numérique peut être utilisée aussi pour permettre de réaliser un multiplexage fréquentiel de plusieurs sources sur un même canal.

Une modulation combinée amplitude-phase permet d'éloigner les valeurs des états par rapport à une simple modulation d'amplitude seule ou une simple modulation de phase seule, en effet, une modulation à 32 états entraîne, soit 8 états de phase et 4 états d'amplitude, soit 32 états de phase ; cette dernière configuration impliquerait des états de phase éloignés de 11 degrés environ, donc « trop proches » en regard des perturbations qui risquent d'affecter les angles.

5. Lorsque l'on utilise un codage numérique de type NRZ, on fait toujours subir à la source binaire une opération préalable d'embrouillage. Pourquoi ? Cette opération modifie-t-elle le débit ?

Le codage numérique NRZ n'engendre pas de transition systématique pour chaque bit, ce qui fait qu'une longue suite de '0' ou de '1' entraîne une valeur constante locale de la fonction stockée ou transmise. Lors de la lecture ou de la réception, le rythme d'horloge ne peut plus donc être restitué (ou stabilisé) correctement. L'opération d'embrouillage consiste à faire une addition modulo 2 entre une séquence binaire de longueur L définie émise par la source et une séquence pseudo-aléatoire (donc connue !) de même longueur L. Le débit n'est donc aucunement modifié.

6. Lors de la visualisation en direct d'une séquence vidéo par Internet, le protocole de communication utilise une procédure « BEC » pour réaliser le traitement des erreurs. Pour quelle raison ? Quel est alors le processus indispensable à mettre en œuvre dans l'application du client qui visualise cette séquence (en plus de la procédure BEC) ?

La séquence vidéo est passée, avant son émission, par un processus de compression très efficace, mais avec perte d'information (mpeg par exemple), ce qui fait que des erreurs de transmission affectant naturellement l'information comprimée pourraient entraîner des erreurs très importantes lors de la décompression. Or, un système « FEC » souffre d'erreurs résiduelles indétectables, ce qui milite donc vers une procédure « BEC ».

En outre, Internet utilise, en règle générale, un protocole de communication (dit « TCP-IP ») basé sur un principe de transmission par paquets numérotés avec un processus de contrôle à l'arrivée puis, selon, acquittement ou demande de répétition. La visualisation d'une séquence vidéo utilise donc le même processus que pour un téléchargement de fichier ou pour une consultation de page web.

Le fait d'utiliser une procédure « BEC » impose que le client mette en œuvre une mémoire « tampon » (« buffer ») de manière à compenser les éventuels aller-retour de transmission inhérents à la procédure « BEC », ce tampon étant destiné à garantir une fluidité de la présentation des images successives, sans coupure, ni arrêt figé.

7. Le codage convolutif est-il utilisé dans une procédure « BEC » ou dans une procédure « FEC » ? (justifiez votre réponse)

Le codage convolutif est utilisé exclusivement dans une procédure « FEC » car c'est délibérément un système de correction d'erreurs, alors qu'une procédure BEC réclame un système de simple détection (on pourrait bien sûr utiliser une partie de l'algorithme de Viterbi pour effectuer une détection d'erreurs, mais ça serait du « gâchis » !)

8. Pourquoi une procédure d'entrelacement est-elle à même de réduire les erreurs résiduelles après traitement ?

Il se trouve que sur les supports de stockage aussi bien que dans les canaux de transmission, les erreurs arrivent par « paquets » (salissure ou déformation mécanique locales dans le premier cas, perturbation électromagnétique localisée dans le temps dans le deuxième cas). Ainsi, on remarque que, la plupart du temps, il n'y a pas d'erreurs, mais que, parfois, il arrive beaucoup d'erreurs groupées. La première situation ne nécessite aucune action du correcteur, mais la deuxième situation peut entraîner un dépassement de capacité de correction du correcteur entraînant à son tour la présence d'erreurs résiduelles.

La solution est de pouvoir « désorganiser » ou « disperser » ces erreurs. La procédure d'entrelacement réalise cette solution.

9. *Pourquoi n'utilise-t-on jamais seul un code convolutif comme moyen de correction d'erreurs ?*

Dans son principe, un correcteur de code convolutif peut engendrer, s'il y a dépassement de sa capacité de correction, de nombreuses erreurs « résiduelles » (en fait, il est à même d'engendrer un message totalement faux par rapport au message d'origine).

De ce fait, il est impératif de faire suivre ce correcteur par un autre, basé sur un principe différent (code linéaire par exemple) et qui sera à même de corriger les erreurs résiduelles du premier (après désentrelacement). Lors du codage, il faut donc prévoir, avant le codage convolutif, un premier codage linéaire, suivi d'un entrelacement (principe du codage « externe »).

10. *L'opération de multiplexage inverse permet de transmettre l'information d'une source dont le débit est supérieur à la capacité d'un seul canal ... mais une opération de compression de données a le même but. Comparer ces deux procédés.*

Ces deux procédés n'opèrent pas avec les mêmes ordres de grandeur.

La compression d'information (sans perte) permet de gagner dans des proportions relativement modestes (sans perte d'information : algorithme d'Huffman ou semblables), la limite étant fixée par le premier théorème de Shannon

Le multiplexage inverse n'a théoriquement aucune limite, car il s'agit de distribuer un débit sur autant de canaux que nécessaire. La limite est alors pratique (ces canaux existent-ils ?) et économique (combien puis-je accepter de mobiliser de canaux ?)

Partie B : Exercices (10 pts)

1. Multiplexage : (6 pts)

Le multiplexage de 32 voies de débits unitaires égaux à 64 kbit/s engendre un format dit « E1 » de débit égal à 2048 kbit/s. Le résultat de cette opération se nomme « affluent E1 » (prendre l'exemple du format vu dans le sujet N° 5 des TD)

1.1 *Proposer une solution permettant de multiplexer 4 affluents pour former un format « E2 » (faire un schéma du format « E2 ») (2 pts)*

Soient 4 affluents ABCD de formats identiques « E1 », donc cadencés au rythme binaire de 2048 kbit/s. On peut imaginer construire un format « E2 » en plaçant à la suite ces 4 affluents dans l'ordre et en adoptant donc un nouveau rythme binaire de $4 \times 2048 = 8192$ kbit/s.

Ce changement de format se fait aisément grâce à un jeu de registres : 4 registres chargés simultanément par les 4 affluents puis lus en série les uns derrière les autres, l'invariance du temps (!) impose donc une horloge 4 fois plus rapide pour la lecture du « E2 » que pour le chargement de chacun des 4 « E1 ».

AAAA.... / BBBB.... / CCCC / DDDDD -- $\rightarrow$ ABCDABCDABCDABCD.....

1.2 *La méthode de synchronisation prévue initialement dans le cadre du format « E1 » est-elle suffisante pour le format « E2 » ou faut-il la modifier ? Faites une proposition. (2 pts)*

Il faut la modifier car on retrouve pour « E2 » le même problème que pour « E1 » : quelle est le premier affluent « A » (même cas de figure pour « E1 » : quelle est la première voie ?).

Solution : on peut adopter exactement le même principe et donc, prévoir un cinquième affluent « T » : TABCDTABCDTABCD.....

Cette solution, radicale, serait très coûteuse en « sur-débit » ainsi occasionné !

Une autre solution est de modifier la séquence de synchronisation existante de l'affluent A.

Le format normal d'un affluent « E1 » est $Sy \ 1 \ 2 \ 3 \ 4 \ 5 \ \dots \ 30 \ 31$

Désormais les affluents « A » (début d'un format « E2 ») seront $S2y \ 1 \ 2 \ 3 \ 4 \ 5 \ \dots \ 30 \ 31$

La séquence binaire $S2y$ étant facilement identifiable, le démultiplexage est possible, car $S2y$ marque le début du multiplex « E2 » et, en même temps de début du multiplex « E1 », et ainsi de suite.

1.3 On itère l'opération pour réaliser des formats « En », toujours en opérant des multiplexages successifs de 4 affluents « En-1 ». Voyez-vous une limite théorique ? ou une limite pratique ? (2 pts)

L'opération précédente, en utilisant le même principe de modification de Sy pour marquer le début du multiplex « En » peut s'itérer sans limite théorique.

A chaque fois, le débit binaire est multiplié par 4.

Ainsi pour E3 on arrive à 32 Mbit/s, puis 131 Mbit/s (E4), puis 524 (E5), puis 2 Gbit/s (E6), puis 8.38 Gbit/s (E7) ... jusqu'à dépasser nos possibilités techniques actuelles de capacité de transmission (limites physiques des canaux, limites électroniques des machines, réglementations).

2. Code linéaire (4 pts)

On considère un code détecteur – correcteur d'erreurs ayant les caractéristiques suivantes (notations habituelles) :

$$m = 6 \qquad k = 3 \qquad n = m + k = 9$$

2.1 Est-ce un code de Hamming ? (1 pt)

Non, ce n'est pas un code Hamming, car n et k devraient satisfaire à $n = 2^k - 1$, ce qui n'est pas le cas.

2.2 Quel est le pouvoir détecteur de ce code ? (1 pt)

En imaginant qu'il est correctement construit, le pouvoir théorique de détection ce code vaut $k = 3$.

2.3 Quel est le pouvoir correcteur de ce code ? (1 pt)

Les mots à traiter font 9 bits, il y a donc, au minimum, 9 erreurs simples possibles.

Le fait que $k=3$ signifie qu'il y a eu 3 équations de redondance, ce qui va pouvoir engendrer un vecteur syndrome à 3 composantes binaires, soit $2^3 - 1 = 7$ syndromes distincts non nuls. La bijection entre l'ensemble de ces 7 syndromes non nuls et l'ensemble des 9 erreurs simples possibles est donc impossible.

Le pouvoir correcteur de ce code est nul = ce n'est pas un code correcteur.

2.4 Pour quelle procédure peut-on utiliser ce code (BEC, FEC, les deux, aucune) ? (1 pt)

On ne peut bien sûr pas l'utiliser pour une procédure « FEC » (il serait nécessaire d'avoir un code correcteur).

On peut l'utiliser pour une procédure « BEC » (détection d'erreurs et demande de répétition si erreur constatée), en sachant qu'il fera une bonne détection tant que le nombre d'erreurs ne dépassera pas 3 sur un mot de 9 bits, soit une probabilité d'erreurs de $1/3$, valeur très mauvaise en moyenne, mais atteignable (et dépassable) en cas de paquet d'erreurs ...