TD liaison de données

1 Gestion d'une liaison de données

L'objectif est de construire de manière incrémentale les éléments de base d'un protocole de liaison de données orienté bit. On considère une liaison de donnés point-à-point avec transmission de messages utilisateurs uniquement d'un ETTD A vers un ETTD B. On ne s'occupera de plus que de la phase de transfert d'information (de A vers B) et pas des fonctions de gestion de la connexion (établissement, libération, maintien).

1.1 Etape 1

On suppose le circuit de données totalement fiable et une capacité de mémorisation illimitée sur l'ETTD B.

a. Quelles sont les fonctions à remplir par le protocole ?

b. Donner la structure la plus simple pour la trame.

c. Ecrire une procédure E1 pour l'émetteur A et une procédure R1 pour le récepteur B pour la mise en œuvre du protocole.

1.2 Etape 2

On lève maintenant l'hypothèse de fiabilité du circuit de données. Répondre aux mêmes questions. On validera de plus les 2 affirmations suivantes :

- Le récepteur B doit traiter de la même façon les trames reçues erronées et les trames non reçues : ne rien faire.
- Il est nécessaire de numéroter les trames (1 bit suffit) pour assurer le séquencement et la non duplication des informations utilisateurs.

a. Quelles sont les fonctions à remplir par le protocole ?

b. Donner la structure la plus simple pour l'unité de donnée manipulée par ces nouvelles fonctions.

c. Ecrire une procédure E2 et une procédure R2 pour A et B pour la mise en œuvre du protocole.

1.3 Etape 3

On lève maintenant l'hypothèse de capacité illimitée de l'ETTD B. On suppose que le récepteur ne peut stocker qu'une trame à la fois. Une trame arrivant en B est mémorisée, analysée puis remise à l'utilisateur mais avec un temps de traitement non négligeable et bien sûr inconnu de A.

a. Donner la structure la plus simple pour l'unité de donnée manipulée par ces nouvelles fonctions.

b. Ecrire les procédures E3 et R3 correspondantes.

1.4 Hints

Dans les 3 étapes, l'écriture des procédures se fera en utilisant des formes algorithmiques usuelles par exemple dans le style PASCAL (tant que faire, si alors sinon, repeter jusqu'a, Debut Fin ...).

On pourra supposer que la communication entre les procédures que l'on veut écrire et l'utilisateur qui veut envoyer des messages se fait par l'intermédiaire d'un buffer `User-Buffer` (rempli par l'utilisateur) qui est passé en paramètre. Ce buffer est rempli dans les procédures en réception (R1, R2, R3) et utilisé pour les procédures en émission (E1, E2, E3).

Pour l'intéraction entre le support physique (circuit de données) et les procédures que l'on écrit, on pourra supposer définies les procédures suivantes :

- `Envoyer(sequence-bit)` qui prend une séquence de bit pour la mettre sur le support physique.
- `Insere-Zero(sequence-bit)` qui retourne la séquence de bits après insertion d'un bit zéro après chaque sous-séquence de 5 bits consécutifs à 1. (de manière plus générale on pourrait prendre une fonction `Transparence` avec un algorithme différent)
- `Lecture_jusqu'a_flag` qui écoute le support physique et retourne la séquence de bits jusqu'au prochain Flag. Le Flag n'est pas retourné par la procédure. La séquence peut donc être vide.
- `Supprime_Zero(sequence_bit)` qui retourne la séquence après suppression des bits zéro de transparence. N.B. : dans la réalité cette fonction serait incluse dans `Lecture_jusqu'a_flag` car l'élimination de la transparence et la reconnaissance du Flag correspondent au même algorithme interne.

Par ailleurs on pourra supposer également définies les procédures suivantes :

- `Armer(Timer)` qui initialise le timer `Timer` pour une durée d'attente.
- `Desarmer(Timer)` qui arrête le timer `Timer`.
- `Expire(Timer)` qui indique si le timer `Timer` a expiré.
- `CRC(Suite_de_bits)` qui retourne la valeur du CRC16 calculée sur la suite de bits `Suite_de_bits`.
- `Extraire(Identifiant, Suite_de_bits)` qui retourne la suite de bits correspondant au champ `Identifiant` dans la suite de bits `Suite_de_bits`. `Identifiant` peut prendre les valeurs: Compteur, Type, Données, CRC, ...
- `Attendre()` qui met le processus en pause pour une seconde.