Elèves référents :

CARADO Anthony
COPPYN Florian
DUBOIS Jean-Marc

Professeur :

Christian TOINARD

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE

TD Réseau n°08 : Transmission fiable – Protocole 5

Vous allez programmer un protocole bidirectionnel simultané (full duplex) au niveau liaison. Il garantit une transmission fiable avec une anticipation en émission et accepte uniquement les trames en séquence.
Le protocole 5, utilisé dans ce TD, est un protocole qui attend 4 types d'événements possibles, dont network_layer_ready.

Vous rendrez le code p5.c complété ainsi que vos réponses au format PDF.

Questions Préliminaires :

QP1 : Donnez la taille de la fenêtre de réception.

QP2 : Donnez la taille de la fenêtre d'émission et déterminez un scénario démontrant la non fiabilité lorsque l'anticipation se fait sur tout l'espace de numérotation. **(Possibilité de faire un schéma)**

→ Ce scénario inclut la perte d'un acquittement ; montrez-le. Quel problème en découle ?

Question 1 :

Dans p5.c, expliquez le rôle des variables *next_frame_to_send*, *ack_expected* et *frame_expected*. (l 34-36).

Explicitez leur utilité pour les fenêtres d'émission et de réception.

Question 2 :

Complétez la fonction *send_data* qui permet de construire une trame, de l'envoyer et d'armer un délai de garde en utilisant `start_timer`, `inc` et `to_physical_layer`

➔ On pourra s'aider des fonctions décrites dans protocole.h

Question 3 :

Complétez le *case network_layer_ready* en utilisant les fonctions *send_data*, *from_network_layer* et *disable_network_layer.*

Question 4 :

Complétez la ligne *if (nbuffered <…)* permettant de déterminer s'il est possible d'accepter un nouveau paquet dans la fenêtre d'émission. Justifiez le choix de cette condition vis-à-vis de la fiabilité.

Question 5 :

Complétez le *case frame_arrival* en utilisant les fonctions *from_physical_layer*, *to_network_layer*, *inc* et *enable_network_layer* .

Bonus :

B1 Le case timeout peut être codé ainsi :

```
case timeout:    /* trouble; retransmit all outstanding frames */

next_frame_to_send = ack_expected;    /* start retransmitting here */
for (i = 1; i <= nbuffered; i++) {
        send_data(next_frame_to_send, frame_expected, buffer);    /* resend 1 frame */
         inc(next_frame_to_send);    /* prepare to send the next one */
}
```

Lorsqu'un délai tombe, pourquoi retransmettons nous toutes les trames en attente ?

B2 Définissez sur l'ensemble du code les lignes qui portent le rôle Emetteur, Récepteur ou Emetteur/Récepteur.