TD N° 2: protocoles SLIP et HDLC

Rémy Grünblatt – remy@grunblatt.org

23 octobre 2019

1 Protocole SLIP

Le protocole SLIP (pour *Serial Line Internet Protocol*) est un protocole d'encapsulation qui permet à deux ordinateurs reliés entre eux par une liaison série d'échanger des paquets IP, en point-à-point. Il est défini dans la RFC 1055 (Requests For Comments) et est souvent surnommé « Rick Adams' SLIP », du nom de son créateur et en raison du grand nombre de protocoles portant aussi le nom de SLIP.

On considère la topologie suivante :

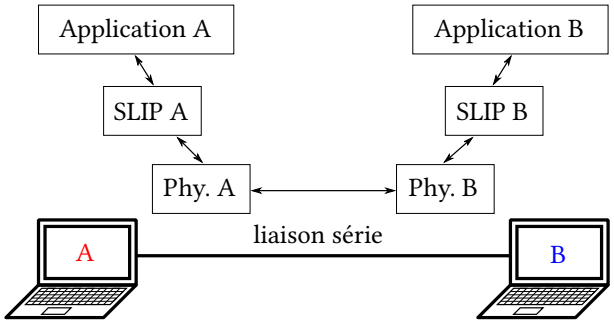

FIGURE 1 – Représentation d'une connection via SLIP

1. De haut en bas puis de bas en haut (de A à B), on a :
   - paquets IP (datagramme) ;
   - trames SLIP ;
   - signal électrique ;
   - trames SLIP ;
   - paquets IP (datagramme).
2. On a en rouge les caractères 'END', et en bleu les caractères d'échappement, ainsi qu'en vert les caractères suivant les caractères d'échappement :

   0x6C 0xC0 0x6E 0x6F 0x75 0x76 0x65 0x6C 0x6C 0x65 0x73 0x20 0x64 0x75 0xDB 0xDC 0x20 0xC0 0xAB 0xC0 0x6D 0x6F 0x6E 0x64 0x65 0xC0

   Le récepteur cherche le caractère END=0xC0 pour repérer le début d'un message. Une fois détecté, il cherche les caractères 0xC0 ou 0xDB pour détecter soit la fin du message, soit un traitement de transparence, respectivement. Il transmet donc les messages suivants à la couche *Application A* : 0x6E 0x6F 0x75 0x76 0x65 0x6C 0x6C 0x65 0x73 0x20 0x64 0x75 0xC0 0x20 puis 0x6D 0x6F 0x6E 0x64 0x65.
3. La couche *SLIP A* transmet les données 0xC0 0x63 0x65 0x63 0x69 0x20 0x63 0x65 0x74 0xDB 0xDC 0x20 0x75 0x6E 0x20 0x74 0x65 0x74 0x65 0xDB 0xDC 0xC0. On préfixe et on suffixe le message par 0xC0 (END), et on effectue un traitement de transparence pour les octets 0xC0 (afin de ne pas les confondre avec les octets END).
4. Cette couche transmet les données initiale à *Application B* (sinon, on a un mauvais protocole).
5. Un commentaire possible est celui disponible dans la RFC originale : `https://tools.ietf.org/html/rfc1055#page-4`

2 Protocole HDLC

On repère en **gras** les fanions de débuts et les fanions de fin, codés sur 8 bits. On a donc trois trames dont le contenu est situé entre ces fanions. On effectue ensuite le traitement de transparence binaire, en enlevant les '0' après chaque

suite de 5 ‘1‘ (en rouge). En bleu, on représente les adresses de destination (codées sur 8 bits). En vert, on représente la commande, codée sur 8 bits. Enfin, en orange, on représente le CRC, codé sur 16 bits.

01111110 01011111 0 00101000 10111110**1** 0101000011101011 **01111110**
01111110 01011111 0 10100101 1110010100110000 **01111110**
01111110 01011111 0 10001101 1100110011100101 **01111110**

L’échange est donc le suivant :

- **Trame n° 1** : La commande est ‘00101000‘. Le premier bit étant ‘0‘, il s’agit d’une trame d’information. On identifie $N(s) = 010$, $P = 1$ (puisque la trame provient de l’initiateur A de l’échange), et enfin $N(R) = 000$. On observe donc une trame de numéro de trame $N(S)$ égal à 2, trame qui attend une réponse (puisque $P = 1$), et qui acquitte toutes les trames antérieures à la trame n° 0. Les *données* contenues dans la trame sont ‘10111111‘.
- **Trame n° 2** : La commande est ‘10100101‘. Le premier bit étant ‘1‘, il ne s’agit pas d’une trame d’information. Le second bit étant ‘0‘, il s’agit donc d’une trame de supervision de données. On identifie $SS = 10$, $P = 0$, et $N(R) = 101$. On a donc affaire à un message de type Receive Not Ready ($SS = 10$), qui n’attend pas de réponse ($P = 0$) et qui acquitte toutes les trames de numéro inférieur (ou égal) à $N(R) = 101 = 5$. A demande donc la suspension des envois après les trames de numéro 5.
- **Trame n° 3** : La commande est ‘10001101‘. Le premier bit étant ‘1‘, il ne s’agit pas d’une trame d’information. Le second bit étant ‘0‘, il s’agit donc d’une trame de supervision de données. On identifie $SS = 00$, $P = 1$, et $N(R) = 101$. On a donc affaire à un message de type Receive Ready ($SS = 00$), qui attend une réponse ($P = 1$), qui signifie que A, l’initiateur, est prêt à recevoir des trames de son correspondant de numéro de trame supérieur à $N(R) = 101 = 5$.

Le déroulement global de l’échange est donc le suivant : A, initiateur, commence par envoyer une requête à B. Cette requête nécessite une réponse de la part de B, puisqu’on a $P = 1$. Ainsi, B envoie des données, données qui sont ensuite acquittées par A dans la trame 2 (A acquite 5 trames, donc B a au moins envoyé 5 trames). A envoie ensuite une trame pour demander une interruption de l’envoi de trame par B (dans le cas de buffers pleins, par exemple). Si B n’a pas fini son envoi, alors il met en pause sa communication et n’envoie pas les trames suivantes. Si B a fini son envoi, alors il n’envoie aucune requête puisqe $P = 0$. Enfin, A informe B qui peut à nouveau recevoir des trames, et la suite (si elle existe) de la réponse de B peut être transmise.