Protocole 802.3, on retransmet les trames jusqu'à réception d'un acquittement.

Il y a plusieurs version de la procédure 802.3.send(). Pour la méthode 802.3.receive(trame) il n'y a qu'une seule version

```
802.3.receive(trame){
        couche.liaison.receive(trame) ;
}
```

C'est la couche physique qui est responsable de détecter et de gérer une collision (c'est-à-dire transmettre assez longtemps pour s'assurer que toutes les stations détectent la collision). La procédure ***boolean*** *physique.send(trame)* retourne une variable d'état **status**=true s'il n'y a pas eu de collision et false sinon.

La procedure 802.3.send(trame) est blocante, le thread appelant est libéré seulement si la trame à été transmise. En pratique, le nombre de retransmission est limté. Comme il n'y a qu'un seul canal il n'y a pas de raison qu'un autre thread appelle cette méthode. Ce n'est pas le cas pour les protocole ARQ (Go-Back_N et Selective Repeat) qui doivent nécessairement utiliser un signal (ou une interruption). La méthode du protocole ALOHA, ALOHA.send, peut aussi s'implémenter sans handler d'interruption (signal).

Version 1 :

Comme dans le cours, on commence à transmettre dès que le canal est libre. S'il y a une collision alors on retransmet mais cette fois on attend un temps aléatoire qui est une variable uniforme dans [1,M]. La valeur de M est multipliée par 2 après chaque nouvelle collisions. La valeur de M doit être contenu dans [Mmin, Mmax] ;

```
802.3.send(trame) {
    M=Mmin;
    while(canal ocupé) {}  attente que le canal se libère
    boolean status = physique.send(trame + entete couche liaison de donnees) ;  encapsulation
    while( !status){
        wait(1,M) ;   attend un temps aléatoire uniformément distribué dans [1,M]
        status = physique.send(trame + entete couche liaison de donnees) ;
        if(2*M <= Mmax)
                M=2*M ;
        end ;
    }
}
```

Version 2 :

Avant de transmettre on attend un temps fixe pour permettre à une trame d'acquittement (ou une autre trame de contrôle) d'être prioritaire.

```
802.3.send(trame) {
    M=Mmin ;
    repeat{
        while(canal ocupé) {}  attente que le canal se libère
        wait(délai fixe)
    until( !canal occupé)
    boolean status = physique.send(trame + entete couche liaison de donnees) ;  encapsulation
    while( !status){
        wait(1,M) ;   attend un temps aléatoire uniformément distribué dans [1,M]
        status = physique.send(trame + entete couche liaison de donnees) ;
        if(2*M <= Mmax)
                M=2*M ;
        end ;
    }
}
```

Version 3 :

Comme Version 2 avec attente aléatoire avant chaque transmission

```
802.3.send(trame) {
    M=Mmin ;
    repeat{
        while(canal ocupé) {}  attente que le canal se libère
        wait(délai fixe)
    until( !canal occupé)
    repeat{
        wait(1,M) ;   attend un temps aléatoire uniformément distribué dans [1,M]
        boolean status = physique.send(trame + entete couche liaison de donnees) ;
        if(2*M <= Mmax)
                M=2*M ;
        end;
    until(!status) ;
}
```