Eva SOUSSI
Paul FLORENCE
3MIC-E

Lorsque nous avons testé le protocole TCP pour l'envoi d'une vidéo, nous avons remarqué qu'à cause du système de reprise des pertes, toutes les images arrivaient côté récepteur mais qu'il fallait du temps pour toutes les récupérer. Nous avions donc une vidéo très saccadée. Notre œil ne voit pas toutes les images et c'est pourquoi il y a une possibilité de faire un protocole plus évolué. Il inclue un mécanisme de reprise des pertes mais elles sont paramétrées telles qu'un certain pourcentage de perte est déclaré acceptable. Il ne va pas nuire au bon visionnage de la vidéo et va permettre d'améliorer la fluidité de lecture.

Implémentation

Pour permettre l'utilisation de différents sockets nous avons choisi de les stocker dans un tableau global comme ceci :

```
#define MAX_SOCKET 1024
mic_tcp_sock binded_sockets[MAX_SOCKET]
```

Pour la gestion des descripteurs de fichiers, on utilise un compteur en variable global qui correspond à l'index dans le tableau et à la valeur de `mic_tcp_sock.fd`. Cela permet de retrouver facilement les données qui nous intérressent quand on nous passe un descripteur de fichier. De plus, pour optimiser le tableau de `mic_tcp_sock` on réutilise les socket qui ont été fermés (état CLOSED).

Nous avons rajouté un état `CONNECTED` dans `protocol_state` qui est l'état assigné à un socket à sa création. Cela sera amené à changer lors des prochains TP (IDLE).

Pour l'envoi de PDU, on ne s'est pas occupé du numéro de port de destination car il est geré par la couche IP lors de l'appel à `IP_send`.

Pour compiler on utilise la commande `make all`.