Projet de Protocoles Internet

Guillermo Morón Usón & Steven Sailly

6 janvier 2023

1 Parties traitées

Ces trois points ont principalement été testés avec le pair *jch.irif.fr* :

- Capacités minimales du pair
- Bon comportement en cas de problèmes réseau (testé avec une connexion douteuse)
- Signatures cryptographiques

La traversée de NAT a été implémentée mais sans succès. Par conséquent, l'export d'une arborescence, aussi implémenté, n'a pas pu être testé. Aucune autre extension n'a été implémentée.

2 Choix d'implémentation

Après avoir été générée une première fois, la clé privée est stockée sur disque, puis réutilisée aux prochaines exécutions du client.

À la fin de l'enregistrement, on s'assure que le serveur ait reçu notre racine en envoyant une requête à l'API REST.

Après l'enregistrement au serveur, on lance deux goroutines pour :

- envoyer des keepalives au serveur et oublier les "anciens" pairs
- lire les requêtes des autres pairs.

La goroutine principale lit les entrées au clavier, et gère l'envoi des requêtes du pair et leurs réponses.

On représente les pairs comme une map associant le nom d'un pair à ses informations (addresses, clé publique...). On représente un paquet par ses champs *Id*, *Type* et *Body*, et les autres champs sont calculés lors de l'envoi du paquet. On représente un noeud avec toutes ses informations, dont son hash.

On construit un noeud à partir d'un fichier *récursivement* :

- s'il est de type *Chunk*, on construit le noeud directement

- s'il est de type *BigFile*, on coupe ses octets de données en 32, on construit les noeuds correspondant à ces données, puis on les ajoute à sa liste de fils
- s'il est de type *Directory*, on construit les noeuds de ses fils, puis on les ajoute à sa liste de fils.

Avant d'envoyer une requête *GetDatum* à un pair p, si le *handshake* avec p n'est pas effectué, on commence par tenter le *handshake*, puis on lance la procédure de traversée de NAT si le *handshake* n'a pas pu être effectué.

Lors d'un téléchargement, on écrit les noeuds sur disque seulement une fois que les données correspondant au hash initialement demandé sont *entièrement* récupérées. Autrement dit, on n'écrit pas au fur et à mesure qu'on récupère les données.

Lorsqu'un pair nous envoie une requête *GetDatum*, on cherche le hash demandé avec un BFS à partir de la racine.

Lorsqu'on souhaite effectuer une traversée de NAT avec un pair q, une goroutine lui envoie des messages *NoOp* afin de percer aussi le NAT de notre côté.

La plupart des envois de messages sont faits en utilisant un backoff exponentiel.

3 Autres informations

3.1 Compilation et exécution

```
go build
./dfsp [DIRECTORY_TO_EXPORT]
```

3.2 Traversée de NAT et signature

Lors des tests de la traversée de NAT, à la réception du premier *Hello*, on a eu des erreurs de vérification de signature. Pourtant, nos messages semblent correctement signés puisqu'ils sont acceptés par *jch.irif.fr*, et la signature des messages de *jch.irif.fr* nous semble valide lors de la vérification.