Projet de programmation réseau

Dazibao par inondation non-fiable

Antoine Berthel & Damien Herber

14 mai 2020

La partie fonctionnelle

- Les calculs du node hash.
- La gestion des paquets et des TLV.
- La table des voisins est fonctionnelle et mis à jours comme demandé (elle peut contenir un maximum de 100 entrées. Il est en effet envisageable de contenir plus de 15 entées comme demandé dans le sujet).
- La table des données publiées est fonctionnelle.
- Toute les erreurs vis-à-vis des paquets et des TLV sont gérées dans notre parser qui détecte et ignore paquets ou les TLV dont la taille annoncé est fausse. Nous ne renvoyons par contre pas de TLV Warning.
- En plus de notre parser nous avons une fonction arcParser permettant le processus inverse. Cette fonction tronque automatiquement la liste de TLV envoyée et s'arrête lorsque cette liste dépasse la taille autorisée pour un paquet.
- Nous avons ajouté des options permettant de changer les valeurs par défaut (voir README)
    - Une option de débogage en fait parti.

Les choses à corriger ou à améliorer

- Le network hash fonctionnait lors de nos testes en local, mais on ne sait pas s'il est correct vis-à-vis du serveur car il n'y a pas de warning.
- Les listes de TLV sont volontairement tronquées à l'envoi pour éviter de dépasser la taille maximale autorisée.

Structure du programme et techniques employée

Nous commençons par créer un fichier afin de conserver notre ID généré de façon aléatoire pour les utilisations suivantes.
Une fois l'ID récupéré nous vérifions si un message a été donné en option afin de l'ajouter dans la table des données publiée (il sera ainsi automatiquement publié lors de l'inondation).
Vient ensuite la phase d'initialisation du socket.

On ajoute ensuite le serveur de manière permanente dans la table des voisins.

Début du protocole d'inondation :

Nous utilisons une boucle while(true).
Toutes les 20 secondes, nous envoyons un TLV 4 à tous nos voisins, un TLV 2 à un voisin aléatoire (si il y a moins de 5 voisins) et vérifions si un voisin à été actif durant les dernières 70 secondes.
Nous avons 5 secondes pour recevoir un paquet et nous le parsons avec la fonction parser(). Nous mettons à jour l'expéditeur dans la table des voisins.
Le protocole fourni par le sujet est alors exécuté.

Parser

Nous avons créé des structures dans le fichier modele.h :

La structure « donnee » est utilisée pour stocker les informations concernant une ligne de la table des données : id est le node id encodé sur 64 bits, seqno est le numéro de séquence encodé sur 16 bits, node_hash est le hash encodé sur 128 bits correspondant au hash du nœud dans lequel il est stocké et data représente la chaîne de caractères de taille length qui permet de stocker un message, nous avons choisi d'utiliser la méthode flexible array.

La structure « addr » est utilisée pour stocker un socket : ip et port.

La structure « voisin » est utilisée pour stocker les information concernant une ligne de la table des voisins.

La structure « tlv » est utilisée pour stocker un tlv parsé : on peut y stocker une liste de pointeurs de donnée.

La structure « paquet » est utilisée pour stocker un paquet parsé : on utilise la même méthode flexible array pour y stocker une liste de pointeurs de tlv.

Exemple d'exécution entre serveur et client

(voir le dossier exemple_execution)

Idées d'extensions

Il est possible de faire un buffer pour stocker les tlv à envoyer à une même personne afin d'envoyer des paquets contenant plusieurs tlv et des tlv dans plusieurs paquets sans les tronquer.

Il est possible de stocker la table des données dans un fichier afin de seulement les mettre à jour lors des connexions suivantes.

Il est aussi possible de tenter de négocier avec les fournisseur Internet afin d'avoir un multicast sur leur réseau.