RAPPORT DE PROJET

DATE: 13/05/2020.

BINOME:

-HAMMAL ABDELMALEK

-HAMERLAIN SAMI

✓ PARTIE IMPLEMENTEES :

-Sujet de base.

-Vérification de la cohérence des Node State.

-Agrégation ('juste l'envoie d'une série de nodehash')

☒ *CHOSES INATTENDUES :*

- On pense qu'il y a un problème avec le nodehash qu'on ne comprend pas, il est toujours diffèrent de celui qu'on reçoit malgré que tous semblent bien justes.

-Quelques erreurs sur valgrind qu'on n'a pas compris d'où elle vient malgré s'ils plantent jamais l'exécution mais on n'a pas pu enlever certaines erreurs qui apparaissent que sur valgrind.

-Fonctionnement avec d'autres pères non testé.

✓ CE QUI MARCHE :

-Récupération des voisins permanents avec getaddrinfo (on récupère deux adresses permanentes une adresse ipv6 on la stock tell qu'elle est et une adresse ipv4 on la stock sous la forme mapped : : ffff: ipv4).

-Utilisation de socket polymorphe envoie et réception des paquets de la part des adresses ipv6 ainsi que des adresses ipv4.

-la boucle a événement (la boucle attend une entrée dans la socket pendant 20 secondes, s'il y a une entrée on gère le paquet sinon on passe au traitement qu'il faut faire chaque 20 secondes).

-teste du paquet reçu et parcours jusqu'au dernier TLV.

-identification et traitement de chaque TLV (la boucle parcourt chaque tlv dans le paquet jusqu'au dernier et fait chaque traitement propre au type du tlv et envoie une réponse à ce dernier).

-gestion des données et voisins. (On teste l'existence de chaque adresse reçue dans la liste des voisins et la possibilité de l'ajouté, ainsi chaque 20 secondes on teste les voisins à supprimer).

- Exécution de chaque traitement périodique de 20 secondes (S'il n'y a pas de données dans 20 secondes la boucle passe à envoyer des networkhash à chaque voisin ainsi que faire la mise à jour de la table de voisins et vérifier si le nombre de voisins est inférieure à 5 on envoie un tlv neighbor request à un voisin tiré au hasard).

✓ CE QUI NE MARCHE PAS :

- On pense qu'il y a un problème avec le nodehash qu'on ne comprend toujours pas :

Il est toujours différent de celui qu'on reçoit malgré que tous semblent bien justes.

❖ STRUCTURE DU PROGRAMME ET TECHNIQUES EMPLOYEES :

Utilisations de listes chainées pour les données et les voisins.

I. Structure d'un voisin :

-Un voisin est caractérisé par deux chaines pour son adresse Host et son port ainsi qu'un entier pour déterminer son type (permanent ou transitoire) et un champ date qui contient la date de l'émission du dernier paquet.

```
20
21  typedef struct voisins{
22      unsigned char *hostadresse;
23      unsigned char *port;
24      int type;//1 permannent 0 sinon
25      time_t date;
26
27  }voisins;
28
29  typedef struct Tvoisins{
30  voisins table[15];
31  int indice;
32  }Tvoisins;
```

II. Structure d’une donnée :

-une donnée est caractérisée par 3 chaines qui représentent successivement l’id, le numéro séquentiel et le contenu de la donnée.

```
7   #include <string.h>
8   #include <netinet/in.h>
9   #include<arpa/inet.h>
10  #include<time.h>
11  #include <openssl/sha.h>
12
13  typedef struct donnees {
14        unsigned char i[8];
15        unsigned char s[2];
16        unsigned char d[192];
17  }donnees;
18
19
20
```

- Boucle à événements pour gestion des paquets avec un Timeval égal à 20 secondes => donc la boucle attend 20 secondes s'il n'y a pas d'entrée on fait les traitements à faire chaque 20 secondes environ sinon on traite le paquet reçu.
- **P**arcours des TLV avec un entier qui pointe toujours vers la tête du TLV et on parcourt en ajoutant à ce dernier la taille du TLV en cours jusqu'à ce qu'on arrive à la taille du paquet reçue.
- **I**dentifications des TLV avec des simples tests "if" sur la tête de chaque TLV du paquet puis on procède au traitement du retour par a port au type du TLV.
- **O**n a fait des fonctions pour chaque type de TLV à rendre, la fonction reçoit un pointeur vers un paquet et les informations nécessaires pour remplir ce dernier Ces fonction retournent le paquet prêt à être envoyé.
- **U**tilisation de fonction pour ajouter des données, une fonction pour vérifier l'existence d'un id dans la table de données ;si il existe la fonction retourne un pointeur vers l'élément sinon elle renvoie NULL, une fonction pour ajouter un voisin, une fonction pour vérifier l'existence d'un voisin dans la liste; une fonction qui parcourt la liste des voisins et teste si le voisin est transitoire et n'a rien envoyé pendant plus de 70 secondes elle le surprime.
  (Ces fonctions reçoivent en argument un pointeur vers le nombre d'éléments dans la liste de données ou vers le nombre de voisins, ce dernier est soit augmenté ou décrémenté par à port au fonctionnement).

❖ REMARQUE :

-Pour chaque itération de la grande boucle While () on demande à l'utilisateur s'il veut changer sa donnée si la réponse est positive on met à jour sa donnée et on augmente le Seqno.

- Puisque on demande à chaque itération une entrée de la part de l'utilisateur, ça va prendre du temps vu au nombre de paquet rentrons, donc pour tester il faut juste enlever le if ou on demande à l'utilisateur de changer sa donnée et on garde la donnée Null du début et c'est ce qu'on a fait dans app2.c => on a enlevé la demande de changement de donnée aussi.

- Pour le code on a laissé deux fichier .c : app.c et app2.c qui sont les même c'est juste que app2.c a plus d'affichage pendant l'exécution que app.c .

-A cause des problème avec le hash , pour ne pas envoyer toujours des node state request a des node qu'on a déjà leur données , on teste l'existence de l'id dans notre liste si il existe je teste son seqno si il est le même que celui reçue on envoie pas de node state resquest pour ce dernier.

❖ Exemple d'exécution annotée :

-Pour la première exécution la boucle selecte ne reçoit rien pendant 20 secondes et elle passe à envoyer un network hash aux 2 adresses permanentes et on voit le paquet suivant sortir de Wireshark :

0000 60 06 cb 24 00 1e 11 40 20 01 00 00 53 aa 06 4c `..$...@ ...S..L
0010 00 90 06 fc a3 65 a2 ee 20 01 06 60 33 01 92 00 .....e.. ..`3...
0020 00 00 00 00 51 c2 1b 9b 04 bc 04 bc 00 1e f0 10 ....Q.....
0030 5f 01 00 12 04 10 8c 48 ba 7b 5c aa 60 ee a3 fb _.....H.{\.`...
0040 2e 7d 58 bd 52 db .}X.R.

-On constate que le paquet commence par 5f ce qui est égal à 95 (MAGIC) ensuite le deuxième octet est égale à 01 après on a 2 octets égaux à 00 12 ce qui est égale à 18 en décimale qui est la taille d'un tlv network hash complet ensuite on a le 4eme octet égale à 04 ce qui est

l'identifiant du tlv network hash et ensuite on a le champ lenght du tlv égal à 10 ce qui est égal à 16 en décimale ce qui est la taille des 16 octets qui suit qui représente le networkhash.

-On constate après qu'un paquet est reçu qui est de la forme qui suit :

0000 60 06 cb 24 00 1e 11 40 20 01 00 00 53 aa 06 4c `..$...@ ...S..L
0010 00 90 06 fc a3 65 a2 ee 20 01 06 60 33 01 92 00 .....e.. ..`3...
0020 00 00 00 00 51 c2 1b 9b 04 bc 04 bc 00 1e f0 10 ....Q.....
0030 5f 01 00 02 05 00 _...

-Qui représente un tlv network state request parce que le père forcement a trouvé que le networkhash reçue est différent de celui qui a calculé.
-Après on constate que notre père répond avec ce paquet sortant :

0000 60 08 b6 dc 00 28 11 40 20 01 00 00 53 aa 06 4c `....(.@ ...S..L
0010 34 c2 0a 87 a3 65 a2 ee 20 01 06 60 33 01 92 00 4....e.. ..`3...
0020 00 00 00 00 51 c2 1b 9b 04 bc 04 bc 00 28 21 70 ....Q.....(!p
0030 5f 01 00 1c 06 1a fe 0e 68 9b 4d 4c fc 7f 00 02 _..... $...
0040 71 5e 4e 9e b2 4b 25 bd c8 1a 86 7c 12 f2 72 df q^N..K%....|..r.

-C'est une série de node hash mais puisqu'il y a que la donnée au début il y a qu'un seul tlv et on constate que le type est égal à 06 au début après sur 8 octets on voit mon id après sur 2 octets mon seqno et finalement le node hash sur 16 octets.

-On constate après une réception d'un paquet nodestate request de la forme suivante :

0000 62 01 6d d4 04 06 11 33 20 01 06 60 33 01 92 00 b.m....3 ..`3...
0010 00 00 00 00 51 c2 1b 9b 20 01 00 00 53 aa 06 4c ....Q... ...S..L
0020 34 c2 0a 87 a3 65 a2 ee 04 bc 04 bc 04 06 9f a0 4....e.....
0030 5f 01 00 0a 07 08 fe 0e 68 9b 4d 4c fc 7f _.....

-Paquet envoyé car forcément le père n'a pas trouvé une entré pour l'id envoyé ou bien que le hash envoyé est différent du hash calculé.

-Notre père repend au node state request par un node state pour l'id reçu de la forme:

0000 60 08 b6 dc 00 35 11 40 20 01 00 00 53 aa 06 4c `....5.@ ...S..L
0010 34 c2 0a 87 a3 65 a2 ee 20 01 06 60 33 01 92 00 4....e.. ..`3...
0020 00 00 00 00 51 c2 1b 9b 04 bc 04 bc 00 35 f3 9f ....Q.....5..
0030 5f 01 00 28 08 26 fe 0e 68 9b 4d 4c fc 7f 00 01 _..).'.....
0040 c3 b6 1b ae 4d 04 91 36 b2 f9 89 c4 7d 70 96 65 ....M..6....}p.e
0050 74 65 73 74 5f 72 61 70 70 6f 72 74 test_rapport

-Après on voit apparaitre notre données sur http://jch.irif.fr:8082/ sous la forme :
- 14 May 06:29 fe0e689b4d4cfc7f 1 test_rapport.