

Rapport

Projet de réseau L3

LIN Christian
C.linu_u8@hotmail.fr
21702806
Université Paris Diderot

Sommaire :

I. Guide d'utilisation

1.Démarrage.....	3
2.Diagramme de commande.....	4
3.Interface de console.....	5
4.Interface de réceptions.....	7
5.Interface de publications.....	8
6.Publication.....	9
7.Suppression.....	10
8.Édition.....	11
9.Fermeture.....	12

II. Code et structure

1.Structuration.....	13
2.Adresses multiples.....	16
3.Publication Rapide.....	17
4.Démarrage fiable.....	18
5.Packet pacing.....	18
6.Suppression de données.....	19
7.Temps monotone.....	19
8.Interface utilisateur.....	20

I. Guide d'utilisation

Ce petit guide a pour objectif d'établir une démonstration claire de l'utilisation et des possibilités que ce logiciel permet de faire, sans avoir besoin de consulter le code.

Elle vous guidera de l'installation des librairies nécessaire à son utilisation la plus totale.

Elle permettra, par ailleurs, de simplifier l'analyse du code en montrant les idées générales de son implémentation.

1. Démarrage

Ce logiciel a été conçu pour être utilisé sous Linux, elle utilise C ainsi que des librairies annexes liés aux sockets. Pour installer les librairies essentielles de C, suivez ses lignes de commandes (sous Linux).

```
$sudo apt update
```

```
$sudo apt upgrade
```

```
$sudo apt install build-essential
```

Cela permet d'installer le compilateur GNU/GCC ainsi que ses outils utilitaires tels que make, dpkg-dev ...

Le logiciel fournit aussi un Makefile où vous pourrez :

-installer les librairies nécessaires :

```
$make install
```

-compiler le projet :

```
$make compile
```

Cela a pour effet de générer un fichier binaire dans bin/netP

-obtenir le path vers le bin :

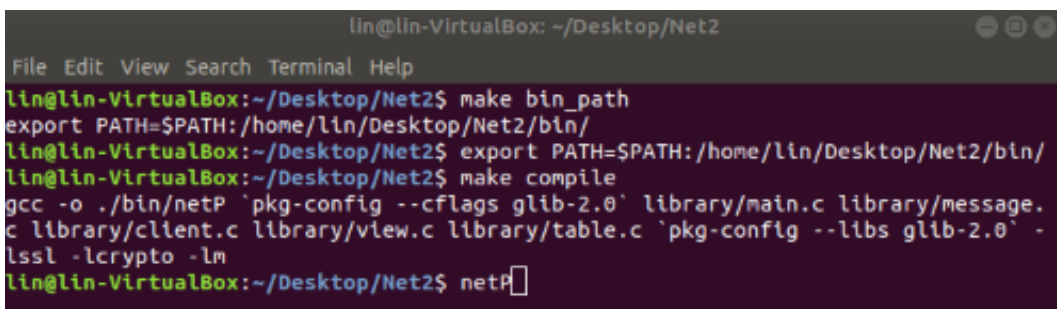
```
$make bin_path
```

Cela a pour effet de print sur le terminal la commande nécessaire pour enregistrer (temporairement) le logiciel sur le path. Pour l'enregistrer, il vous faudra copier ce qui sera print (*ctrl+MAJ+c*) et coller sur le terminal (*ctrl+MAJ+v*)

Pour lancer le projet, il faudra exécuter le fichier :

```
netP
```

Voici un exemple de lancement du logiciel:



```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help
lin@lin-VirtualBox:~/Desktop/Net2$ make bin_path
export PATH=$PATH:/home/lin/Desktop/Net2/bin/
lin@lin-VirtualBox:~/Desktop/Net2$ export PATH=$PATH:/home/lin/Desktop/Net2/bin/
lin@lin-VirtualBox:~/Desktop/Net2$ make compile
gcc -o ./bin/netP `pkg-config --cflags glib-2.0` library/main.c library/message.
c library/client.c library/view.c library/table.c `pkg-config --libs glib-2.0` -
lssl -lcrypto -lm
lin@lin-VirtualBox:~/Desktop/Net2$ netP
```

2. Diagramme de commande

Le logiciel utilise des lignes de commandes afin de réaliser des actions.

Au lancement du programme, l'utilisateur est dans l'interface de #console.

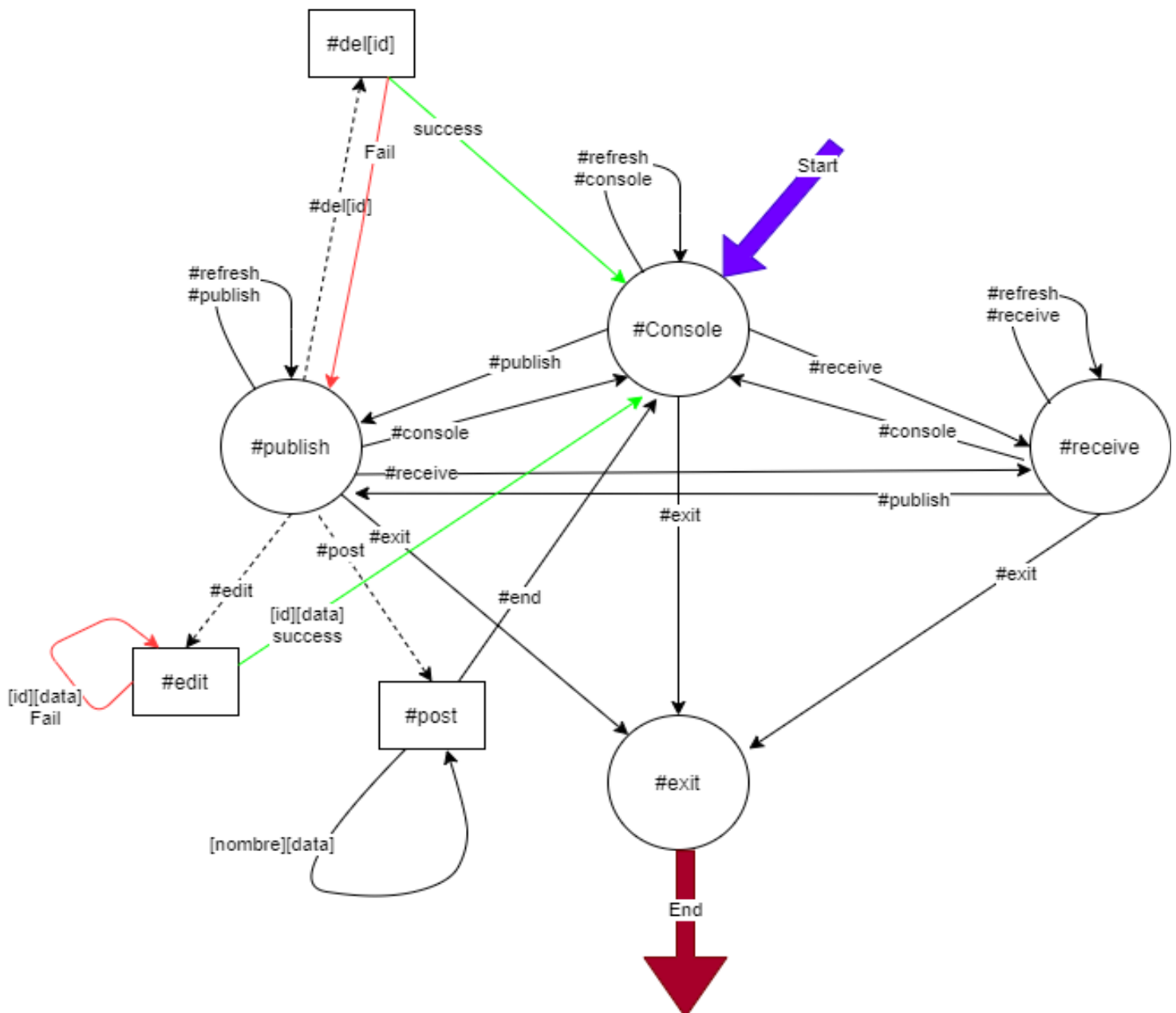
L'utilisateur peut changer d'interface à tous moment en respectant le diagramme des commandes ci-dessous.

--Extention----->

[id]=char[8]

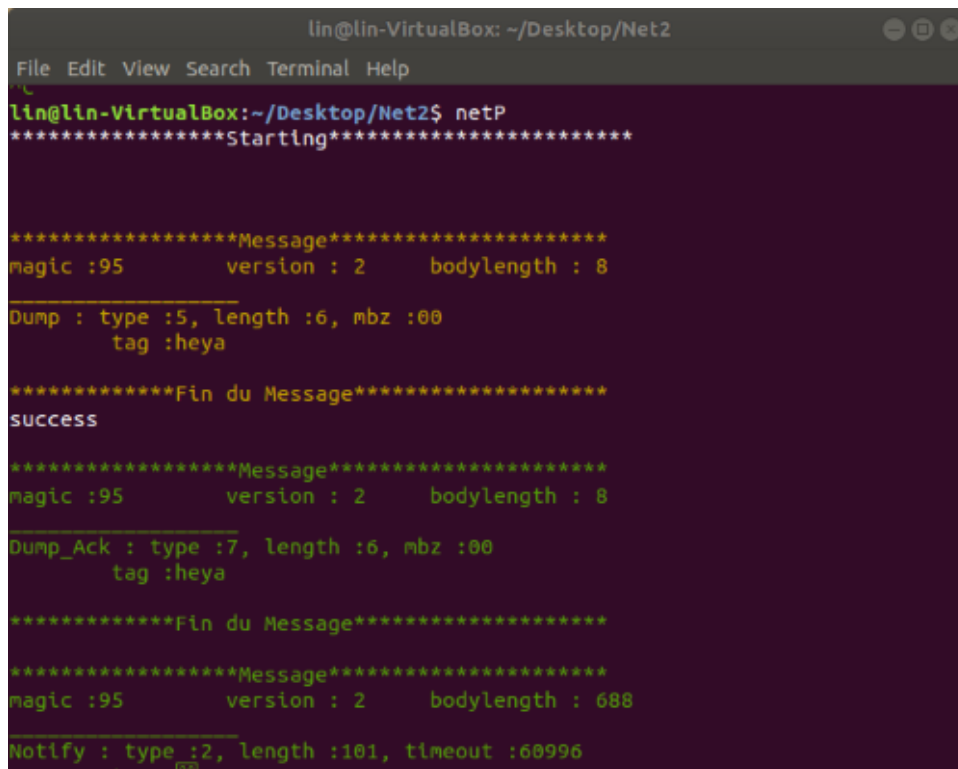
[nombre]=uint16_t

[data]=char*



3.Interface de console

Au lancement du programme `netP`, l'utilisateur commence sur l'interface de console.



```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help

lin@lin-VirtualBox:~/Desktop/Net2$ netP
*****Starting*****

*****Message*****
magic :95      version : 2      bodylength : 8
Dump : type :5, length :6, mbz :00
      tag :heya
*****Fin du Message*****
success

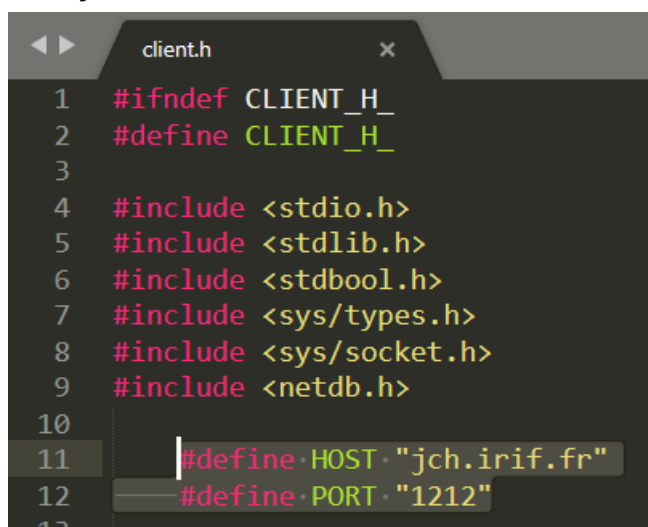
*****Message*****
magic :95      version : 2      bodylength : 8
Dump_Ack : type :7, length :6, mbz :00
      tag :heya
*****Fin du Message*****

*****Message*****
magic :95      version : 2      bodylength : 688
Notify : type :2, length :101, timeout :60996
```

*Les messages en jaune sont les paquets envoyés par l'utilisateur vers le serveur.

*Les messages en vert sont les paquets reçus du serveur.

L'adresse du serveur peut être configurée dans le fichier : `library/client.h`



```
client.h
1  #ifndef CLIENT_H_
2  #define CLIENT_H_
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <stdbool.h>
7  #include <sys/types.h>
8  #include <sys/socket.h>
9  #include <netdb.h>
10
11  #define HOST "jch.irif.fr"
12  #define PORT "1212"
13
```

L'utilisateur peut à tout moment modifier le destinataire des paquets envoyés en modifiant `HOST` et `PORT`, il faudra néanmoins

recompiler et relancer le programme.
L'utilisateur peut, à tout moment, entrer une ligne de commande.
Si l'utilisateur entre une ligne de commande vide, alors un message va être print indiquant l'interface actuel de l'utilisateur (en jaune) et les possibilités de lignes de commandes en bleu.

```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help

Notify_Ack : type :3, length :14, mbz :00
tag :[00]
id :^ ^

Notify_Ack : type :3, length :14, mbz :00
tag :[00]
id :7<>J@CML

Notify_Ack : type :3, length :14, mbz :00
tag :[00]
id :K@<1HHSB

Notify_Ack : type :3, length :14, mbz :00
tag :[00]
id :MSK2;>N;

*****Fin du Message*****

Vous êtes dans la console

[#console] pour visionner la console
[#receive] pour visionner l'interface des données reçus
[#publish] pour aller au mode de publication
[#refresh] pour rafraîchir le mode actuel
[#exit] pour quitter le programme
```

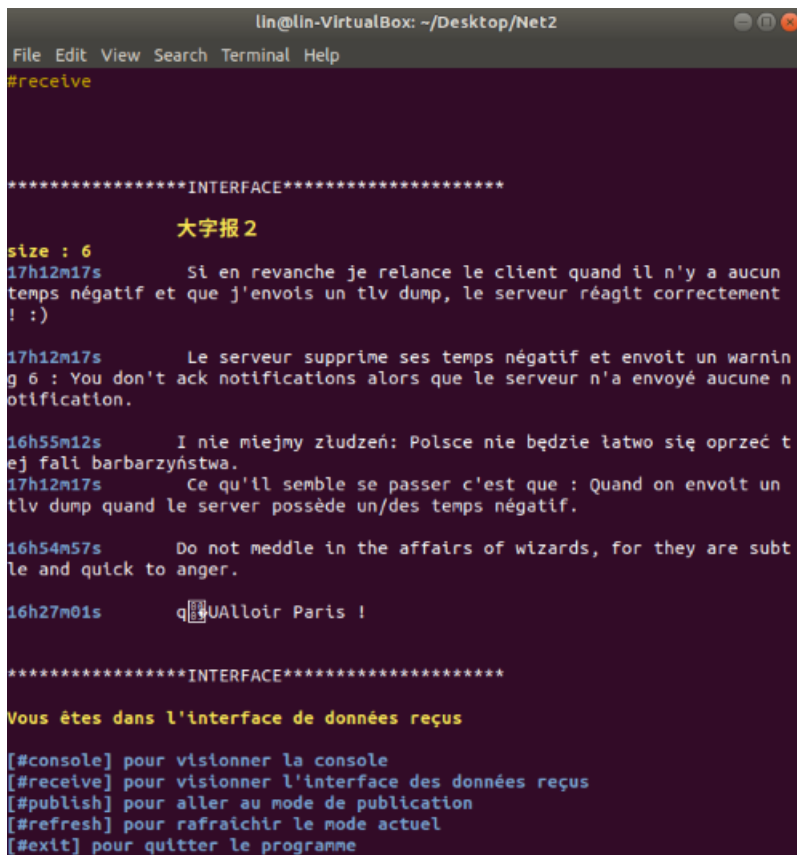
4.Interface de réception

L'interface de réception présente les données reçus par le serveur.

Elle indique le temps d'expiration en bleu et le message associé sans couleur.

Elle est accessible en entrant:

`#receive`



```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help
#receive

*****INTERFACE*****

          大字报 2
size : 6
17h12m17s      Si en revanche je relance le client quand il n'y a aucun
temps négatif et que j'envoie un tlv dump, le serveur réagit correctement
! :)

17h12m17s      Le serveur supprime ses temps négatif et envoie un warnin
g 6 : You don't ack notifications alors que le serveur n'a envoyé aucune n
otification.

16h55m12s      I nie miejmy złudzeń: Polsce nie będzie łatwo się oprzeć t
ej fali barbarzyństwa.
17h12m17s      Ce qu'il semble se passer c'est que : Quand on envoie un
tlv dump quand le server possède un/des temps négatif.

16h54m57s      Do not meddle in the affairs of wizards, for they are subt
le and quick to anger.

16h27m01s      q[?]ualloir Paris !

*****INTERFACE*****

Vous êtes dans l'interface de données reçus

[#console] pour visionner la console
[#receive] pour visionner l'interface des données reçus
[#publish] pour aller au mode de publication
[#refresh] pour rafraichir le mode actuel
[#exit] pour quitter le programme
```

L'utilisateur peut changer d'interface en suivant les instructions présentes en bleu.

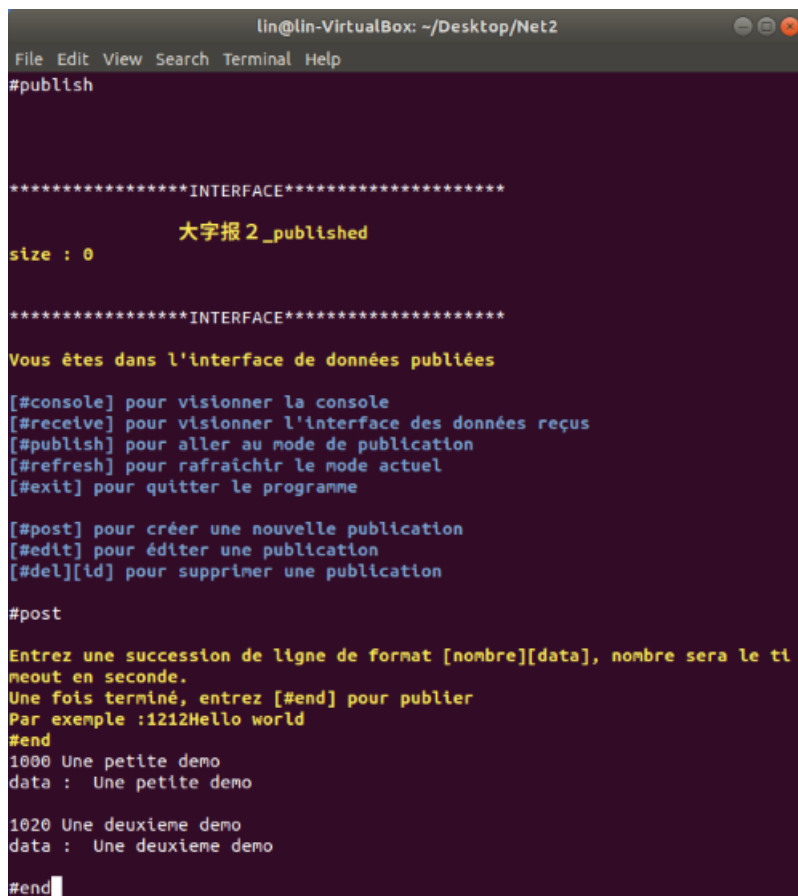
5.Interface de publication

L'interface de publication présente les données publiées par le client.

Elle indique le temps d'expiration en bleu et l'id et le message en Cyan.

Elle est accessible en entrant:

`#receive`



```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help
#publish

*****INTERFACE*****

          大字报 2 _published
size : 0

*****INTERFACE*****

Vous êtes dans l'interface de données publiées

[#console] pour visionner la console
[#receive] pour visionner l'interface des données reçus
[#publish] pour aller au mode de publication
[#refresh] pour rafraichir le mode actuel
[#exit] pour quitter le programme

[#post] pour créer une nouvelle publication
[#edit] pour éditer une publication
[#del][id] pour supprimer une publication

#post

Entrez une succession de ligne de format [nombre][data], nombre sera le timeout en seconde.
Une fois terminé, entrez [#end] pour publier
Par exemple :1212Hello world
#end
1000 Une petite demo
data : Une petite demo

1020 Une deuxieme demo
data : Une deuxieme demo

#end
```

Ici aucune donnée est présente car l'utilisateur n'a pas encore publié de donnée.

L'utilisateur peut changer d'interface en suivant les instructions présentes en bleu.

6.Publication

L'utilisateur peut publier une ou plusieurs donnée en entrant **#post** à partir de l'interface de publication.

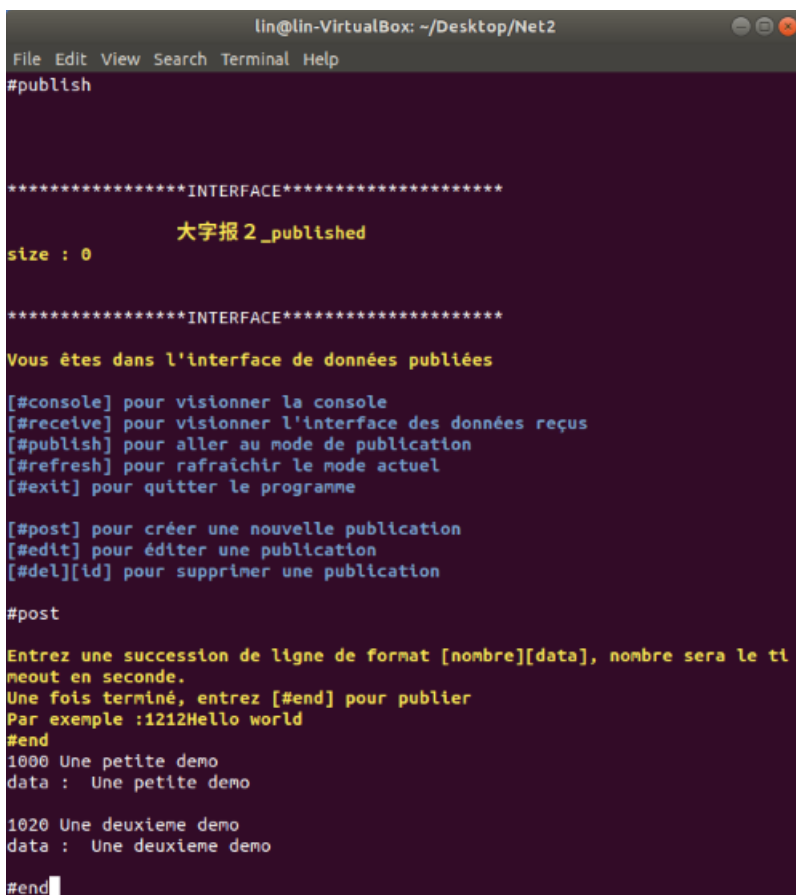
L'utilisateur devra ensuite entrer une série de lignes de format : [nombre][data] où [nombre] représente le temps de survie de la donnée en seconde, et [data] représente la donnée à publier. Une fois fini, l'utilisateur devra entrer **#end** pour confirmer la fin du paquet et l'envoyer vers le serveur.

Par exemple :

```
1000 Une petite demo
1020 Une deuxieme demo
#end
```

Attention : Si l'utilisateur n'entre pas de [nombre], le message aura une durée de survie de 0 seconde.

L'exemple est illustrée dans l'image ci-dessous.

A screenshot of a terminal window titled 'lin@lin-VirtualBox: ~/Desktop/Net2'. The terminal shows a menu-driven interface for publishing data. The user has entered '#publish' and is now in the 'INTERFACE' section. The interface displays 'size : 0' and a list of commands: [#console], [#receive], [#publish], [#refresh], [#exit], [#post], [#edit], and [#del][id]. The user has entered '#post' and is prompted to enter a line of format '[nombre][data]'. The user has entered '1000 Une petite demo' and '1020 Une deuxieme demo', and finally entered '#end' to confirm the publication. The terminal output shows the data being published: 'data : Une petite demo' and 'data : Une deuxieme demo'.

```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help
#publish

*****INTERFACE*****

          大字报 2 _published
size : 0

*****INTERFACE*****

Vous êtes dans l'interface de données publiées

[#console] pour visionner la console
[#receive] pour visionner l'interface des données reçus
[#publish] pour aller au mode de publication
[#refresh] pour rafraichir le mode actuel
[#exit] pour quitter le programme

[#post] pour créer une nouvelle publication
[#edit] pour éditer une publication
[#del][id] pour supprimer une publication

#post

Entrez une succession de ligne de format [nombre][data], nombre sera le tl
meout en seconde.
Une fois terminé, entrez [#end] pour publier
Par exemple :1212Hello world
#end
1000 Une petite demo
data : Une petite demo

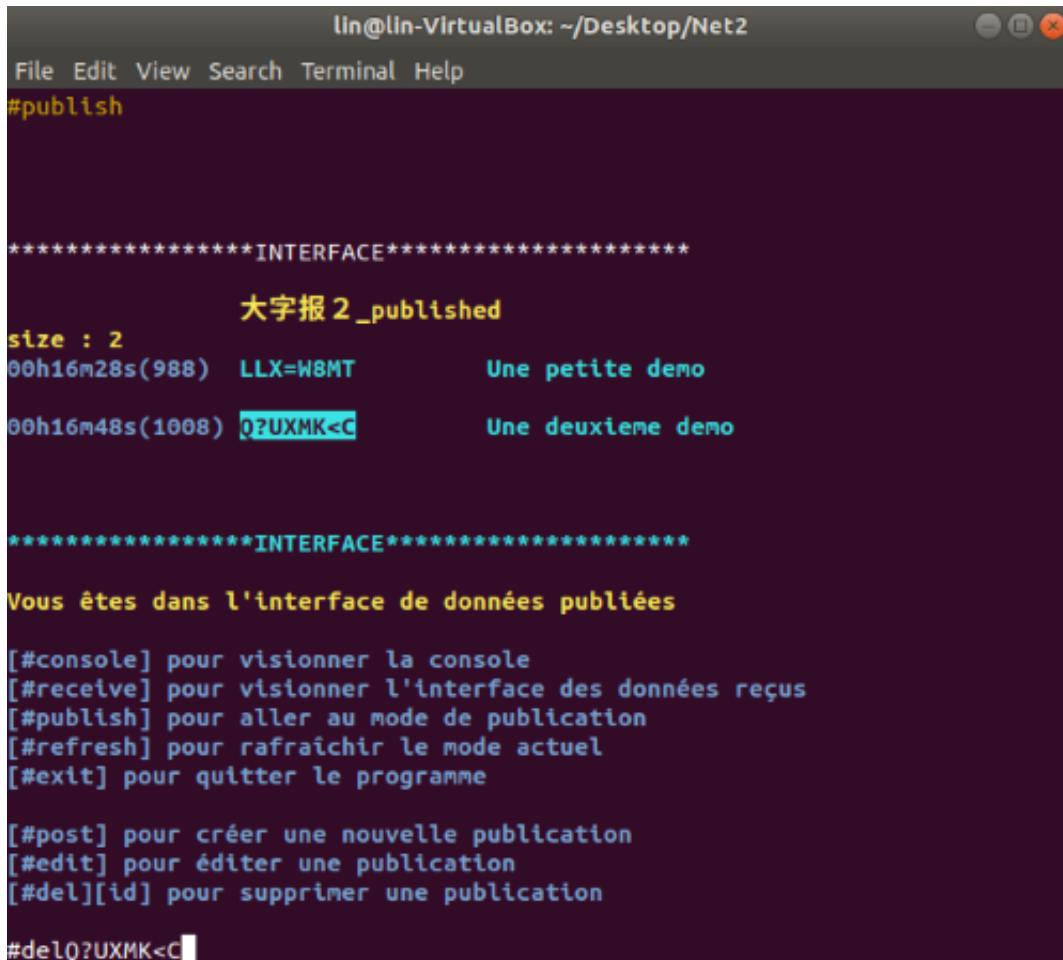
1020 Une deuxieme demo
data : Une deuxieme demo

#end
```

L'utilisateur retournera dans l'interface de console après avoir entré **#end**.

7. Suppression

L'utilisateur peut supprimer une donnée en entrant #del[id] à partir de l'interface de publication où id est l'identifiant du message publié. L'id est visible dans l'interface de publication.



```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help
#publish

*****INTERFACE*****

          大字报 2 _published
size : 2
00h16m28s(988)  LLX=W8MT          Une petite demo
00h16m48s(1008)  Q?UXMK<C        Une deuxieme demo

*****INTERFACE*****

Vous êtes dans l'interface de données publiées

[#console] pour visionner la console
[#receive] pour visionner l'interface des données reçus
[#publish] pour aller au mode de publication
[#refresh] pour rafraîchir le mode actuel
[#exit] pour quitter le programme

[#post] pour créer une nouvelle publication
[#edit] pour éditer une publication
[#del][id] pour supprimer une publication

#delQ?UXMK<C
```

L'exemple illustré par l'image montre la suppression de :

« Une deuxieme demo » dont l'id est Q?UXMK<C

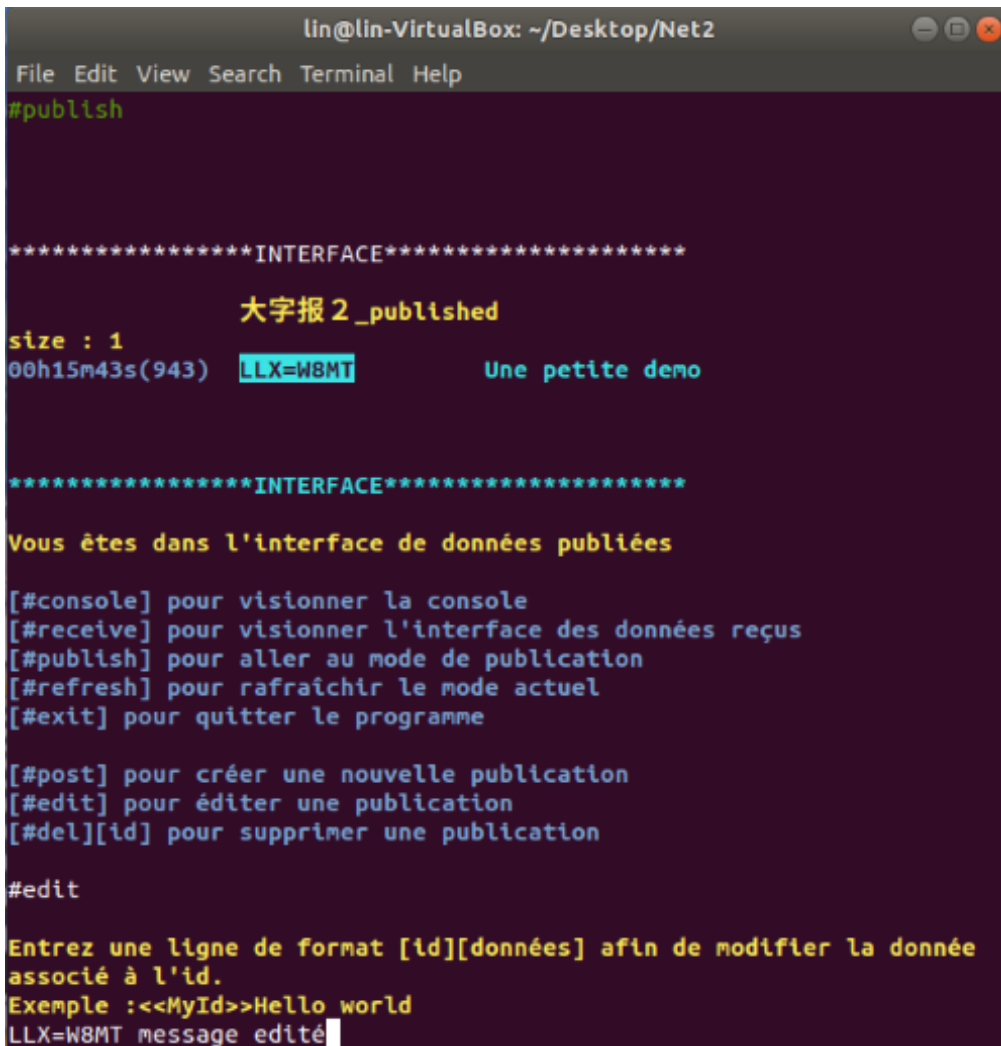
Ainsi, #delQ?UXMK<C supprimera la donnée associé à cette id.

Une fois la suppression réussie, l'utilisateur sera redirigé vers l'interface de console. Sinon il restera dans l'interface de publication.

NB : Sur le terminal, vous pouvez copier et coller une valeur avec ctrl+maj+c et ctrl+maj+v

8.Édition

L'utilisateur peut éditer une donnée en entrant **#edit** à partir de l'interface de publication.

A screenshot of a terminal window titled 'lin@lin-VirtualBox: ~/Desktop/Net2'. The window shows the output of the '#publish' command. It displays a list of data items, including '大字报 2_published' with a size of 1 and a timestamp of '00h15m43s(943)'. The item is indexed by 'LLX=W8MT' and has the value 'Une petite demo'. Below this, a menu of commands is shown, including '#edit'. The user has entered '#edit', and the terminal prompts for a line in the format '[id][données]' to modify the data. An example is given: 'Exemple :<<MyId>>Hello world'. The current line is 'LLX=W8MT message édité'.

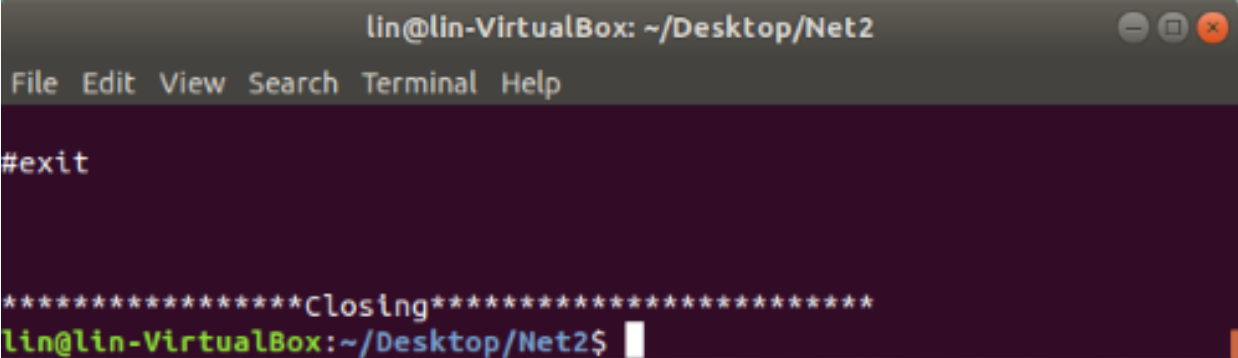
L'exemple illustré par l'image montre l'édition de:
«Une petite demo» indexé par l'id LLX=W8MT en
« message édité».

Une fois l'édition réussie, l'utilisateur sera redirigé vers l'interface de console. Sinon il restera dans l'interface de publication.

NB : Sur le terminal, vous pouvez copier et coller une valeur avec ctrl+maj+c et ctrl+maj+v

9. Fermeture

L'utilisateur peut fermer le programme en entrant :
#exit



```
lin@lin-VirtualBox: ~/Desktop/Net2
File Edit View Search Terminal Help

#exit

*****Closing*****
lin@lin-VirtualBox:~/Desktop/Net2$
```

Cette commande met fin au programme et libère les variables préalablement allouées.

II. Code et structure

Cette partie a pour objectif de comprendre le code, les algorithmes ainsi que les idées généraux à partir de l'analyse du code et sa structure.

1. Structuration

Structuration des fichiers :

```
Net2
|--bin----->netP
|      +----->netS
|
|--library-->client.c
|      +----->client.h
|      +----->main.c
|      +----->message.c
|      +----->message.h
|      +----->table.c
|      +----->table.h
|      +----->view.c
|      +----->view.h
|
|--server--->server.c
|
|--Makefile
|
|--README.txt
|
|--Rapport.pdf
|_____
```

bin : Contient tous les fichiers binaires

netP : Fichier binaire qu'il faudra exécuter.

netS : Veuillez ignorer ce fichier binaire provenant de server/server.c

library : Contient l'ensemble des codes permettant de faire fonctionner le programme.

Client.c: Fichier de compilation contenant le coeur du programme run() permettant de lancer le programme.

Client.h : Fichier header associé à client.c

Main.c : Fichier de compilation lançant run() de client.h

message.c : Fichier de compilation regroupant des fonctions permettant la création et la lecture de paquets.

message.h:Fichier header associé à message.c, elle contient toute les structures de tlv à envoyer par le client.

table.c:Fichier de compilation regroupant les fonction de gestion des tables de hashage, ainsi que les fonction permettant de créer des données associés à ses tables.

table.h:Fichier header associé *table.c*, contient les structures des données à publiées ou reçus.

view.c:Fichier de compilation, elle a pour vocation de mettre en forme les différente vue du programme.

view.h:Fichier header associé à *view.c*

Serveur : contient *server.c*, elle a été utilisé seulement en cas de panne du serveur destinataire. Elle a pour simple vocation de déboguer et vérifier si un paquet envoyé a bien été reçu. Veuillez donc ignorer ce fichier.

Le programme utilise les structures suivante pour les TLV.

```
typedef struct{
    uint8_t magic;
    uint8_t version;
    uint16_t bodylength;
}Head;

typedef struct{
    uint8_t type;
    uint8_t length;
    uint8_t mbz[];
}PadN;

typedef struct{
    uint8_t type;
    uint8_t length;
    uint16_t timeout;
    char tag [4];
    char id [8];
    char data[];
}Notify;

typedef struct{
    uint8_t type;
    uint8_t length;
    uint8_t mbz[2];
    char tag [4];
    char id [8];
}Notify_Ack;

typedef struct{
    uint8_t type;
    uint8_t length;
    uint16_t timeout;
    char id [8];
    char secret [8];
    char data[];
}Publish;

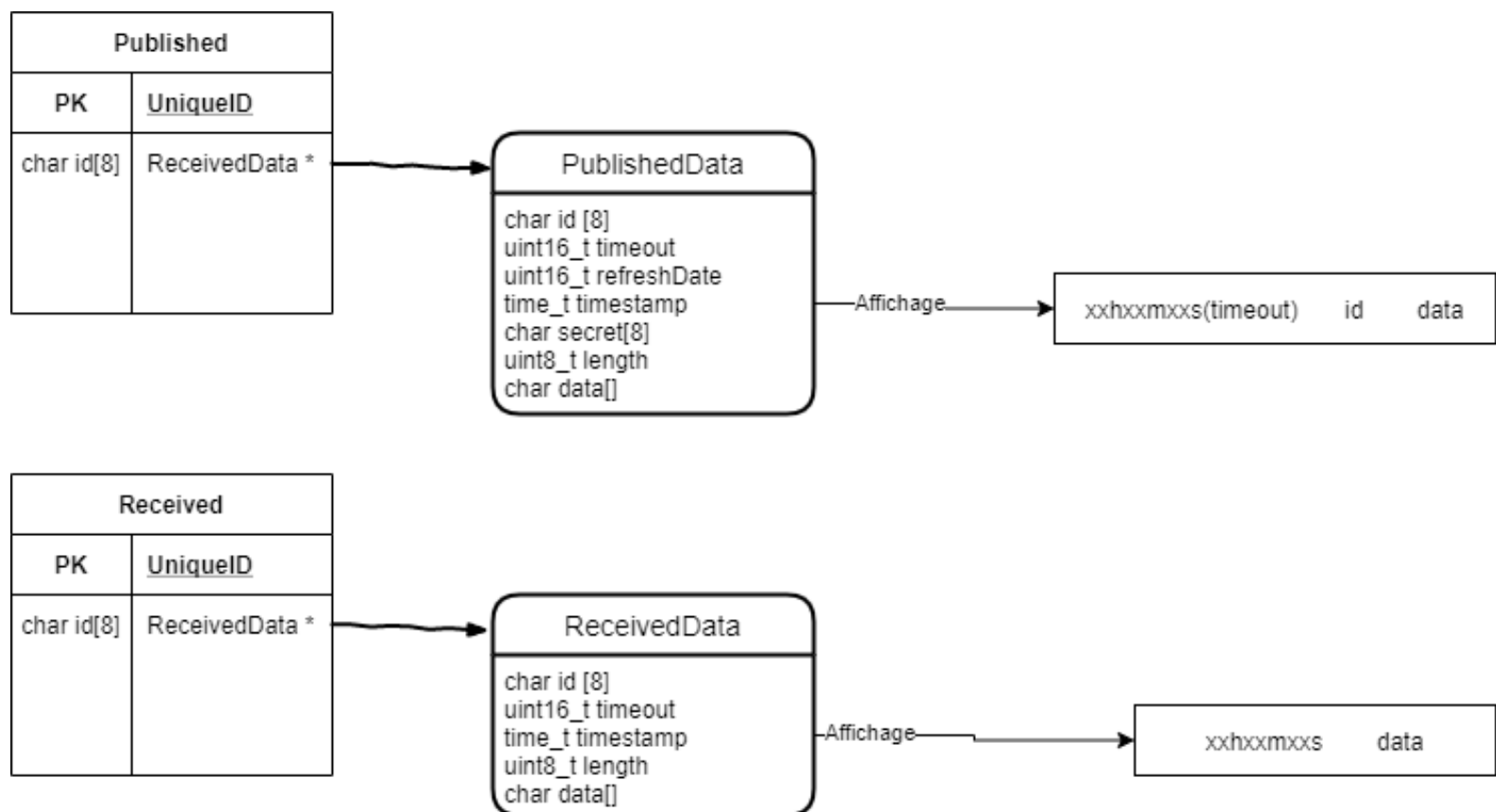
typedef struct{
    uint8_t type;
    uint8_t length;
    uint8_t mbz[2];
    char tag[4];
}Dump;

typedef struct{
    uint8_t type;
    uint8_t length;
    char message[];
}Warning;

typedef struct{
    uint8_t type;
    uint8_t length;
    uint8_t mbz[2];
    char tag[4];
}Dump_Ack;
```

Le programme utilise les structures suivantes pour la gestion des tables :

Les tables sont des GhashTable provenant de la librairie Glib.



Le programme n'utilise pas de base de donnée, ainsi lorsque l'utilisateur quitte le programme, les données ne seront pas sauvegardés.

2.Adresses multiples

Le programme permet de contacter le serveur sur toutes ses adresses, puis sélectionner celle qui fonctionne.

Cela est rendue possible grâce au paramètre AF_UNSPEC de addrinfo, ainsi, getaddrinfo va pouvoir générer une liste d'addrinfo dont le 'family' peut être un AF_INET (ipv4) ou AF_INET6 (ipv6)

```
37  int sockfd;
38  int rc;
39  int res;
40  addrinfo * hints = setAddrInfo(AI_PASSIVE, AF_UNSPEC, SOCK_DGRAM, 0);
41
42  addrinfo * result, * rp;
43
44  if(sockfd<0) {perror("socket error"); exit(1);}
45
46  if(rc=getaddrinfo(HOST, PORT, hints, &result) != 0) {perror("invalid_address"); exit(1);}
47  for(rp = result; rp!=NULL; rp=rp->ai_next){
48      if(rp->ai_family == AF_INET){
49          printf("IPV4\n");
50          sockfd = socket(AF_INET, SOCK_DGRAM, 0);
51          break;
52      }else if(rp->ai_family == AF_INET6){
53          printf("IPV6\n");
54          sockfd = socket(AF_INET6, SOCK_DGRAM, 0);
55          break;
56      }else{
57          printf("UNKNOWN, continue la recherche\n");
58      }
59  }
```

Si aucune adresse est trouvée, le programme se termine.

3.Publication Rapide

Les publications liés aux lignes de commandes sont instantanés. Ainsi, lorsqu'un client publie une nouvelle donnée, ou lorsqu'il change la valeur d'une donnée existante, le client n'a pas besoin d'attendre le prochain cycle (de 8 secondes) de publication.

La fonction permettant la gestion des lignes de commandes est : « onLineInserted » de client.c.

```
while(!quit){
    if(res = recvfrom(sockfd, buf, 1024, MSG_DONTWAIT, rp->ai_addr, &(rp->ai_addrlen))>0){
        printMsg(buf, GREEN);
        onReceive(sockfd, rp, buf, published, received);
        //break;
    }
    if(poll(&mypoll, 1, 20)){
        .....fgets(cmdLine, 1024, stdin);
        .....onLineInserted(sockfd, rp, cmdLine, publishBuf, padding, published, received);
        .....}
    if(difftime(time(NULL), cycle)>8){
        cycle=time(NULL);
        if(tableSize(published)>0){
            //printf("checking publish\n");
            if(update_tables(sockfd, rp, received, published)){
                assocTime = time(NULL);
            }
        }
        g_hash_table_foreach_remove(received, isExpiredReceived, NULL);
    }
    if(difftime(time(NULL), assocTime)>50){
        if(sendHead(sockfd, rp)){
            assocTime=time(NULL);
        }
    }
}
```

4. Démarrage fiable

L'utilisateur peut utiliser soit la forme courte du tlv Dump, soit la forme longue. Par défaut, elle est initialisée avec la forme longue.

Le code permettant de faire cela est :

```
if(sendDump(sockfd, result, "heya")){
    printf("success\n");
}
```

où sendDump va envoyer un paquet contenant un TLV Dump_Ack au serveur avec le tag « heya ». Si le tag est NULL, un TLV Dump sera envoyé.

5. Packet Pacing

L'algorithme update_table retourne true quand un paquet a été envoyé et false quand aucun paquet a été envoyé. Ce qui permet d'échelonner une publication automatique tous les 8 secondes. Ce qui permet de moins surcharger le réseau .

```
while(!quit){
    if(res = recvfrom(sockfd, buf, 1024, MSG_DONTWAIT, rp->ai_addr, &(rp->ai_addrlen))>0){
        printMsg(buf, GREEN);
        onReceive(sockfd, rp, buf, published, received);
        //break;
    }
    if( poll(&mypoll, 1, 20) ){
        fgets(cmdLine, 1024, stdin);
        onLineInserted(sockfd, rp, cmdLine, publishBuf, padding, published, received);
    }
    if(diffTime(time(NULL), cycle)>8){
        cycle=time(NULL);
        if(tableSize(published)>0){
            //printf("checking publish\n");
            if(update_tables(sockfd, rp, received, published)){
                assocTime = time(NULL);
            }
        }
        g_hash_table_foreach_remove(received, isExpiredReceived, NULL);
    }
    if(diffTime(time(NULL), assocTime)>50){
        if(sendHead(sockfd, rp)){
            assocTime=time(NULL);
        }
    }
}
```

6. Suppression de donnée

L'utilisateur peut, à tous moment, supprimer une donnée avec l'utilisation de lignes de commandes.

(Cf : I.7 : Suppression)

Lorsqu'un utilisateur poste une donnée, l'id de la donnée est générée aléatoirement parmi les « caractères printable ». Ainsi, la suppression #del[id] est possible.

Ainsi, Un client qui désire supprimer une donnée peut envoyer un TLV Publish avec un délai d'expiration égal à 0.

7. Temps monotone

Pour la gestion du temps, le programme gère les temps fixe avec `time_t` ;

Sachant que `time(NULL)` donne le temps actuel, alors `difftime(time(NULL), timestamp)` donne le temps écoulé.

Une simple soustraction permet donc d'obtenir le temps fixe restant.

```
133 time_t·timeLeft(time_t·timestamp,·uint16_t·timeout){
134     time_t·now = time(NULL);
135     return timeout-(now-timestamp);
136 }
137
```

Ce qui permet d'obtenir un « Temps monotone » sans soucis de décalage.

8.Interface utilisateur

Le programme fourni une interface utilisateur en texte (non graphique).

Elle utilise des codes couleurs afin de faciliter la compréhension du déroulement des messages. (Message jaune = message envoyé ; Message vert = message reçu).

Elle peut naviguer à travers divers mode, à l'aide de lignes de commandes comme illustré dans « I. Guide d'utilisation ».