Rapport sur le projet UGEGreed

DEBATS Julien - LY-IENG Steven

UgeGreed Ly-Ieng Debats

1 Introduction . . . 2
1.1 Présentation du projet . . . 2
1.2 Présentation de l'application . . . 2
1.3 Détails sur le fonctionnement de l'application . . . 2
2 Architecture . . . 3
2.1 Transmissions des données . . . 3
3 UGEGreed . . . 4
3.1 RFC . . . 4
3.2 Code . . . 4
4 Utilisation . . . 4
4.1 Connexion . . . 4
4.2 Déconnexion . . . 5
4.3 Lancement . . . 5
4.4 Default . . . 5
5 Évolutions depuis la Beta . . . 6
5.1 Maniere de traiter . . . 6
6 Fonctionnalités . . . 15
6.1 Fonctionnel . . . 15
6.2 Fonctionnel Partiellement ou pas entièrement implémenté . . . 15
6.3 Non Fonctionnel . . . 15
7 Difficulté rencontré . . . 16
7.1 Difficulté sur la RFC . . . 16
7.2 Difficulté sur le code . . . 16
8 Division du travail . . . 18
8.1 RFC . . . 18
8.2 Code . . . 18
8.3 Rapport . . . 18
9 Annexe . . . 18
9.1 Commentaires . . . 18
9.2 Remerciement . . . 18

1 Introduction

1.1 Présentation du projet

Le but du projet UGEGreed est de réaliser un système de calcul distribué au dessus du protocol TPC. L'idée est d'aider les chercheurs qui veulent tester des conjectures sur un très grand nombre de cas en distribuant leurs calculs sur plusieurs machines.

1.2 Présentation de l'application

Typiquement, un chercheur va vouloir tester une conjecture sur tous les nombres de 1 à 1 000 000 000. Pour cela, il va écrire une fonction Java qui teste cette conjecture pour un nombre n donné. Il ne lui reste qu'à exécuter cette fonction sur tous les nombres de 1 à 1 000 000 000. Cela peut prendre beaucoup de temps et on voudrait pouvoir accélérer le processus en partageant la vérification sur plusieurs machines. Par exemple, si l'on dispose de 10 machines, une machine peut vérifier les nombres de 1 à 100 000 000, une autre de 100 000 001 à 200 000 000, etc...

Le but des applications que vous allez développer sera de pouvoir se connecter les unes aux autres pour se répartir les tâches à faire. Ensuite, les applications téléchargeront le Jar correspondant et exécuteront le code pour chacune des valeurs qui leur ont été attribuées et renverront les réponses vers l'application qui a proposé la tâche initiale. C'est elle qui créera le fichier contenant toutes les réponses.

Nous vous donnerons le code nécessaire pour instancier une classe contenue dans un Jar. Ce n'est pas une difficulté du projet.

1.3 Détails sur le fonctionnement de l'application

Quand on démarre une application, on lui donne un port d'écoute sur lequel elle acceptera la connexion d'autres applications. On peut, en plus, donner l'adresse d'une autre application. Dans ce cas, l'application va commencer par se connecter à l'autre application. Si l'application est démarrée sans l'adresse d'une autre application, on dit qu'elle est démarrée en mode ROOT.

Par exemple, on peut démarrer une application en mode ROOT à l'adresse A sur le port 6666, puis une application à l'adresse B sur le port 7777 en lui disant de se connecter à A sur le port 6666, et enfin une autre application à l'adresse C sur le port 8888 en lui disant de se connecter sur l'application à l'adresse B sur le port 7777. Les trois applications forment un réseau qui vont pouvoir s'échanger des informations.

Une fois qu'une application est démarrée, l'utilisateur va pouvoir demander le test d'une conjecture en donnant l'url du Jar, le nom qualifié de la classe contenue dans le Jar et la plage des valeurs à tester. Les calculs à faire doivent être répartis entre les différents membres du réseau et les réponses doivent être collectées par l'application qui a fait la demande. De plus, les clients doivent pouvoir moduler la charge de travail qu'ils acceptent. Par exemple, si le client fait déjà beaucoup de calculs pour le réseau, il doit pouvoir refuser.

De plus votre protocole doit permettre à une application qui n'a pas été démarrée en ROOT de se déconnecter du réseau sans impacter le reste du réseau, ni perdre des calculs. En particulier, si cette application s'était engagée à faire des calculs, ces calculs devront être réalisés par d'autres applications du réseau après son départ.

Sauf en cas de déconnexion d'une autre application du réseau, les applications ne doivent pas initier d'autres connexions que la connexion qu'elles établissent au démarrage

2 Architecture

Notre architecture se résume surtout a l'envoi de plusieurs types de trames pour pouvoir tout faire.

2.1 Transmissions des données

Pour nous la transmission des données se fait par plusieurs types de trames

Connexion

Lors de la connexion d'une application au réseau, l'application parente a cette dernière (celle a qui elle s'est connecté) transmet une trame NEW LEAF qui stocke l'adresse de l'application et son adresse, et on fait remonter cette trame jusqu'à la ROOT tout en ajoutant les applications par lesquels la trame est passé, en plus de cela, chaque application devra en même temps mettre à jour sa table de routage en conséquence.

La ROOT transmettra aussi cette trame vers ses applications enfants qui ne lui ont pas envoyé cette trame pour pouvoir mettre à jour leurs table de routage tout en ajoutant aussi à leurs tours leurs adresse à la fin de la trame.

Déconnexion

Lors d'une demande de déconnexion l'application va envoyer une trame signalant à toutes les applications du réseau sa mise en déconnexion pour qu'elles suppriment cette application de leurs table de routage mais aussi pour que les applications connexes se connectent entre eux pour délaisser l'application qui se déconnecte. En même temps avant de se déconnecter l'application répartie et envoi ses taches a ses enfant si c'est un enfant sinon il l'envoi a son père.

Lancement de la tache

L'application sur laquelle on initie la tâche va pinger toute les applications du réseau pour savoir lesquels sont libre, celles qui seront libres enverrons un ping réponse disant si elles sont libre ou non. Celle qui seront libre se feront attribuer une partie de la tache.

Lorsque l'on reçoit une tâche, via une trame, on regarde l'URL ou bien le chemin par lequel on doit passer pour accéder au jar et aussi on prend en compte le fichier dans lequel doit être sauvegardé les données.

A chaque fois que l'on doit fait une conjecture, le résultat sera directement mis dans le fichier. Une fois la tache fini l'application se met en attente d'une nouvelle tache

3 UGEGreed

3.1 RFC

Vous pouvez consulter notre RFC à tout moment ici

3.2 Code

Nous avons en majorité utilisé la RFC pour faire la partie code même si il y a certains moment ou nous avons décidé de dévier légèrement de la RFC. Mais globalement cela reste la même chose que ce qu'on avait prévu au départ. Il y a des choses que nous avons ajouté ou supprimé selon le besoin.

4 Utilisation

4.1 Connexion

TODO

Pour pouvoir lancer le code, il faudra se rendre au niveau du //jar// et lancer la commande

Pour la root

```
java -jar UGEGreed.jar <Host> <Port>
```

ou bien

```
java bin.fr.uge.greed.Main <Host> <Port>
```

Pour la une autre application

```
java -jar UGEGreed.jar <Host> <Port> <ServerHost> <ServerPort>
```

ou bien

```
java bin.fr.uge.greed.Main <Host> <Port> <ServerHost> <ServerPort>
```

Connexion établis

Lorsqu'une connexion s'etablis la table de routage et la table de connexion se met a jour et nous affiche

```
-------------Table of connexions--------------
Connected To : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:60610 remote=localhost/127.0.0.1:1111]

-----RouteTable------
localhost/127.0.0.1:1111 : localhost/127.0.0.1:1111
```

Apres une nouvelle connexion d'une application à l'application qui se trouve a l'adresse 4444

```
-------------Table of connexions--------------
Conneted from : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:2222 remote=/127.0.0.1:60612]
Connected To : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:60610 remote=localhost/127.0.0.1:1111]

-----RouteTable------
localhost/127.0.0.1:1111 : localhost/127.0.0.1:1111,
/127.0.0.1:3333 : /127.0.0.1:3333
```

(Ceci est un exemple lorsque l'on connecte deux application à une autre)

- le "connected from" s'affiche pour présenter les applications qui se sont connecté à cette application

- le "connected to" s'affiche lorsqu'on se connecte à une autre application, il ne peut y avoir que un ou 0 "connected to"

4.2 Déconnexion

Pour pouvoir se déconnecter "correctement sans "ctrl+c" il y aura un scanner sur chaque applications qui lorsqu'il lira

```
DISCONNECT
```

déconnectera l'application

Sur le terminal qui se déconnecte on aura

```
DISCONNECT
mai 07, 2023 11:57:21 PM fr.uge.greed.Application processCommands
INFO: ----------------------
Disconnecting the node ...
.
.
.
mai 07, 2023 11:57:21 PM fr.uge.greed.Application analyseur
INFO: Disconnected Succesfully
----------------------
```

et ça nous renverra sur la ligne de commande

Pour les autres terminaux

```
Connexion closed >>>>>>>>>>>>>>>>>>>>>>>>>> /127.0.0.1:60612

-------------Table of connexions--------------
Connected To : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:60610 remote=localhost/127.0.0.1:1111]

-----RouteTable------
localhost/127.0.0.1:1111 : localhost/127.0.0.1:1111
```

4.3 Lancement

Le lancement d'une tache se fera en utilisant la commande:

```
START <URL/JAR> <Nom classe> <Debut> <Fin> <Fichier resultat>
```

4.4 Default

Si dans le terminal il y est marqué autre chose que ces deux dernières commandes, il nous sera envoyé un message d'usage

5 Évolutions depuis la Beta

Ce qu'il manquait

- Lors de la soutenance Beta notre code contenait pas encore les tout les reader, nous et nous partions sans le savoir sur un serveur bloquant. et nous n'avions non plus pas le lanceur de tâches
- La table de routage était utilisé avec des InetSocketAddress, l'enseignant nous a dit qu'il était préférable que ce soit des contextes.

Ce qui à été ajouté

- Les methodes ne sont pas bloquantes
- Nous avons les readers qui sont fonctionnels
- Nous avons aussi ajouté les trames qui se lancent bien.
- La console permetant de Deconnecter et de Lancer une tache est présente (mais on ne peux pas encore lancer de tache (voir point suivant))
- Le launcher de conjectures est présent mais pas implémenté dans le projet (voir README)
- La deconnexion n'est pour l'instant possible que pour les feuilles

Ce qui n'a pas pu être ajouté

- Nous n'avons pas changé les clés valeurs de la table de routage par les Contextes comme demandé car nous n'en n'avons pas eu le temps, nous préférions nous concentrer sur les autres aspect du code que nous trouvions bien plus importante
- La déconnexion d'une application interne et re-connexion de ses fils vers l'application supérieur

Ce qui à été supprimé depuis la Beta car plus d'utilité

- Trame First Root : Comme lorsque l'on se connecte on envoi un buffer vers la Root pour prévenir d'update la table de routage on a plus besoin de demander a la Root de ping pour recevoir les données pour update la table de routage.
- Trame New Leaf : La trame First Leaf s'occupe de faire ce qui etait sensé fait avec la Trame New Leaf
- placer les données a traiter dans un buffer, on a mis dans des champs à la place

5.1 Maniere de traiter

L'analyseur de trames (dans la classe Application)

```
void analyseur(Trame tramez) throws IOException {
        Objects.requireNonNull(tramez);
        switch (tramez.getOp()) {
        case 3 -> {
                var tmp3 = (TrameAnnonceIntentionDeco) tramez;
                var appDeco = tmp3.dda().AddressSrc();
                var daronApp = tmp3.dda().AddressDst();
                table.deleteRouteTable(appDeco);
                table.removeKeyFromValue(appDeco);
                if(daronApp.equals(localInet)) {
                        System.out.println("JE SUIS TON PERE");
                                var doa = new DataOneAddress(5,appDeco);
                                TrameSuppression supp = new
                                    TrameSuppression(doa);
                                broadCastWithoutFrom((InetSocketAddress)
                                    recu.getRemoteAddress(), supp);

                                var dda = new DataDoubleAddress(4,
                                    localInet,appDeco);
```

```
                        var trameConfirmation  = new
                            TramePingConfirmationChangementCo(dda)
                            ;
                        var context = getContextFromSocket(recu);
                        context.queueTrame(trameConfirmation);
                }
                else {
                        if(table.get(appDeco).equals(scDaron.
                            getRemoteAddress())) {

                                var beauDaron = scDaron.getRemoteAddress
                                    ();
                                scDaron.close();
                                scDaron = SocketChannel.open();
                                System.out.println("open");
                                selector = Selector.open();
                                scDaron.configureBlocking(false);
                                scDaron.register(selector, SelectionKey.
                                    OP_CONNECT);

                                scDaron.connect(daronApp);
                                System.out.println("MON␣DARON:␣" +
                                    scDaron.getRemoteAddress());
                                                //table.
                                                    removeKeyFromValue((
                                                    InetSocketAddress)
                                                    beauDaron);
                                var doa = new DataALotAddress(7,new
                                    ArrayList<InetSocketAddress>(Arrays.
                                    asList(localInet)));
                                var dda = new DataDoubleAddress(4,
                                    localInet,(InetSocketAddress)
                                    beauDaron);
                                var trm = new
                                    TramePingConfirmationChangementCo(dda)
                                    ;
                                var trme = new TrameFirstLeaf(doa);
                                daronContext.queueTrame(trm);
                                selector.wakeup();

                                if(!isroot) {
                                        daronContext.queueTrame(trm);
                                        selector.wakeup();
                                        System.out.println("ENVOI");
                                }
                        }
                }
        }
        case 4 -> {
                var tmp4 = (TramePingConfirmationChangementCo) tramez;
                var addressDeco = tmp4.dda().AddressDst();
                var addressChangement = tmp4.dda().AddressSrc();
                if(addressDeco.equals(localInet)) { //c'est nous qui se
                    barrons
                        if(connexions.size() == 1) {
                                silentlyClose(daronContext.key);
                                Thread.currentThread().interrupt();
                                logger.info("Disconnected␣Succesfully\n
                                    ----------------------");
                                System.exit(0);
                        }
                }
        }
        case 5 -> {
                var tmp5 = (TrameSuppression) tramez;
                var addressDeco = tmp5.doa().Address();
                table.removeKeyFromValue(addressDeco);
                table.deleteRouteTable(addressDeco);
                broadCastWithoutFrom((InetSocketAddress) recu.
                    getRemoteAddress(),tramez);
                if(isroot) {
                        var list = table.getAllAddress();
                        list.add(localInet);
                        var ndla = new DataALotAddress(8, list);
                        var caca = new TrameFullTree(ndla);
                        broadCast(caca);
                }
        }
        case 7 -> {
                var tmp7 = (TrameFirstLeaf) tramez;
```

```
                var listo = tmp7.dla().list();
                var route = listo.get(listo.size()-1);
                for(int i = 0; i != listo.size();i++) {
                        if(!listo.get(i).equals(localInet)) {
                                table.addToRouteTable(listo.get(i), route
                                    );
                                if(!reseau.contains(listo.get(i))) {
                                        reseau.add(listo.get(i));
                                }
                        }
                }
                if(!isroot) {//pas root
                        if(connexions.size()!=1){
                                System.out.println( listo.removeIf(e -> e
                                    .equals(localInet)));
                                listo.add(localInet);
                                var ndla = new DataALotAddress(7,listo);
                                var trm = new TrameFirstLeaf(ndla);
                                daronContext.queueTrame(trm);
                        }
                }
                else {//root
                        var list = new ArrayList<>(reseau);
                        list.add(localInet);
                        var ndla = new DataALotAddress(8, list);
                        var caca = new TrameFullTree(ndla);
                        broadCast(caca);
                }
                selector.wakeup();
        }
        case 8 -> {
                var tmp8 = (TrameFullTree) tramez;
                var listz = tmp8.dla().list();
                table.removeKeyIf(listz);
                for(int i = 0; i != listz.size(); i++){
                        if(!listz.get(i).equals(localInet)) {
                                table.addToRouteTable(listz.get(i),(
                                    InetSocketAddress) recu.
                                    getRemoteAddress());
                                if(reseau.contains(listz.get(i))) {
                                        reseau.add(listz.get(i));
                                }
                        }
                }
                if(!isroot && connexions.size() > 1){
                        listz.removeIf(e -> e.equals(localInet));
                        listz.add(localInet);
                        var ndla = new DataALotAddress(8, listz);
                        var trm = new TrameFullTree(ndla);
                        broadCastWithoutFrom((InetSocketAddress)scDaron.
                            getRemoteAddress(),trm); //A verifier avec le
                            broadcast
                }
        }
        case 10 -> {
                TramePingEnvoi tmp = (TramePingEnvoi) tramez; // A verif
                var address = tmp.doa().Address();
                if(connexions.size() > 1) {
                        broadCastWithoutFrom((InetSocketAddress) recu.
                            getRemoteAddress(),tramez);
                        selector.wakeup();
                }
                DataResponse dr;
                if(bufferDonnee.position() != 0 || dispo!=null) {
                        dispo = address;
                        dr = new DataResponse(11,localInet,address,false)
                            ;
                } else {
                        dr = new DataResponse(11,localInet,address,true);
                }
                var trm = new TramePingReponse(dr);
                var con = getContextFromSocket(recu);
                con.queueTrame(trm);
                selector.wakeup();
        }
        case 11 -> {
                var tmp11 = (TramePingReponse) tramez;
                var addressSrc = tmp11.dr().addressSrc();
                var addressDest = tmp11.dr().addressDst();
                var resp = tmp11.dr().boolByte();
```

```
                if(!addressDest.toString().equals(localInet.toString().
                    replace("localhost", ""))) {
                        broadCastWithoutFrom((InetSocketAddress) recu.
                            getRemoteAddress(),tramez);
                }else {
                        if(resp ==false) {
                                if(commande.computeIfPresent(addressSrc,
                                    (k,v) -> v = false)==null){
                                        commande.put(addressSrc,false);
                                }
                        }
                        if(resp == true) {
                                if(commande.computeIfPresent(addressSrc,
                                    (k,v) -> v = true)==null){
                                commande.put(addressSrc,true);
                                }
                        }
                }
        }
        default -> {
                return;
        }
        }
}
```

Classe Pour lire une addresse

```
public class AddressReader implements Reader<InetSocketAddress>{
        private enum State{
                DONE,WAITING_IP,WAITING_TYPE,WAITING_HOST,ERROR
        }

        private static int BUFFER_SIZE = 1024;
        private final ByteBuffer bufferType =  ByteBuffer.allocate(Byte.
            BYTES);
        private final ByteBuffer bufferHost = ByteBuffer.allocate(Short.
            BYTES);
        private final ByteBuffer bufferAddress = ByteBuffer.allocate(
            BUFFER_SIZE);
        private byte ipType;

        private InetSocketAddress address;
        private  Short host;
        private State state = State.WAITING_TYPE;

        private int IPV4 = 4 * Byte.BYTES;
        private int IPV6 = 16 * Byte.BYTES;

        @Override
        public ProcessStatus process(ByteBuffer bb) {
                if(state == State.DONE || state == State.ERROR) {
                        throw new IllegalStateException();
                }
                bb.flip();
                try {
                        if(state == State.WAITING_TYPE) {

                                if(bb.remaining() <= bufferType.remaining
                                    ()) {
                                        bufferType.put(bb);
                                }
                                else {
                                        var oldLimit = bb.limit();
                                        bb.limit(bufferType.remaining()+
                                            bb.position());
                                        bufferType.put(bb);
                                        bb.limit(oldLimit);
                                }
                                if(bufferType.remaining() != 0) {
                                        return ProcessStatus.REFILL;
                                }
                                ipType = bufferType.flip().get();
                                if(ipType != 4 && ipType != 6) {
                                        System.out.println("ip error");
                                        return ProcessStatus.ERROR;
                                }
                                state = State.WAITING_IP;
                        }
                        if(state == State.WAITING_IP) {
                                if(ipType == 4) {
                                        while(bb.hasRemaining() &&
                                            bufferAddress.position() <
                                            IPV4 && bufferAddress.
                                            hasRemaining()) {
                                                bufferAddress.put(bb.get
                                                    ());
                                        }
                                        if(bufferAddress.position() <
                                            IPV4) {
                                                return ProcessStatus.
                                                    REFILL;
                                        }
                                }
                                else {
                                        while(bb.hasRemaining() &&
                                            bufferAddress.position()  <
                                            IPV6 && bufferAddress.
                                            hasRemaining()) {
                                                bufferAddress.put(bb.get
                                                    ());
                                        }
                                        if(bufferAddress.position() <
                                            IPV6) {
                                                return ProcessStatus.
                                                    REFILL;
```

```
                                        }
                                }
                                state = State.WAITING_HOST;
                        }
                        if(state == State.WAITING_HOST) {
                                if(bb.remaining() <= bufferHost.remaining
                                    () ) {
                                        bufferHost.put(bb);

                                }
                                else {
                                        var oldLimit = bb.limit();
                                        bb.limit(bufferHost.remaining()+
                                            bb.position());
                                        bufferHost.put(bb);
                                        bb.limit(oldLimit);
                                }
                                if(bufferHost.remaining() != 0) {
                                        return ProcessStatus.REFILL;
                                }
                                host = bufferHost.flip().getShort();
                        }
                }finally {
                        bb.compact();
                }
                try {
                        InetAddress inetAddress = null;
                        if(ipType == 4) {
                                byte[] addressBytes = new byte[4];
                                bufferAddress.flip();
                                bufferAddress.get(addressBytes);
                                inetAddress = Inet4Address.getByAddress(
                                    addressBytes);
                                address = new InetSocketAddress(
                                    inetAddress,host);
                        }
                        else {
                                byte[] addressBytes = new byte[16];
                                bufferAddress.flip();
                                bufferAddress.get(addressBytes);
                                inetAddress = InetAddress.getByAddress(
                                    addressBytes);
                                address = new InetSocketAddress(
                                    inetAddress,host);
                        }
                }catch(IOException e){ //UnknownHostException

                }
                state = State.DONE;
                return ProcessStatus.DONE;
        }

        @Override
        public InetSocketAddress get() {
                if(state == State.DONE) {
                        return address;
                }
                throw new IllegalStateException();
        }

        @Override
        public void reset() {
                state = State.WAITING_TYPE;
                bufferAddress.clear();
                bufferType.clear();
                bufferHost.clear();
        }

}
```

Classe lisant plusieurs addresses dans une trame

```
public class LotAddressReader implements Reader<ArrayList<
    InetSocketAddress>>{
        private enum State{
                DONE,WAITING,ERROR
        }
        private State state = State.WAITING;
        private final AddressReader reader = new AddressReader();
        private final IntReader intReader = new IntReader();
        private  int nbAddress;
        private ArrayList<InetSocketAddress> list = new ArrayList<>();
        @Override
        public ProcessStatus process(ByteBuffer bb) {
                if(state == State.DONE || state == State.ERROR) {
                        throw new IllegalStateException();
                }

                bb.flip();
                var readerState = intReader.process(bb);
                if(readerState == ProcessStatus.DONE) { // On recupere le
                     nombre d'addresse que la trame possede
                        nbAddress = intReader.get();
                        intReader.reset();

                        var nb = 0;
                        while(nb < nbAddress) {//On recupere chaque
                            addresse qu'on met dans une liste
                                readerState = reader.process(bb);
                                if(readerState == ProcessStatus.DONE) {
                                        list.add(reader.get());
                                        reader.reset();
                                }
                                else {
                                        return readerState;
                                }
                                nb++;

                        }

                }
                else {
                        return readerState;
                }
                list = list.stream().distinct().collect(Collectors.
                    toCollection(ArrayList::new));
                state = State.DONE;
                return ProcessStatus.DONE;
        }
        @Override
        public ArrayList<InetSocketAddress> get() {
                if(state == State.DONE) {
                        return list;
                }
                throw new IllegalStateException();
        }
        @Override
        public void reset() {
                state = State.WAITING;
                list = new ArrayList<>();
                reader.reset();
                intReader.reset();
        }

}
```

Console

```
private void consoleRun() {
        try {
                try(var scanner = new Scanner(System.in)){
                        while(scanner.hasNextLine()){
                                var msg = scanner.nextLine();
                                sendCommands(msg);
                        }
                }
                logger.info("Console␣thread␣has␣stopped");
        } catch(InterruptedException e){
                logger.info("Console␣thread␣interrupted");
        }
}

private void sendCommands(String msg) throws InterruptedException{
        if(msg == null) {
                return;
        }
        commandQueue.add(msg);
        selector.wakeup();
}

private void processCommands() {
        if(commandQueue.isEmpty()) {
                return;
        }
        var commands = commandQueue.poll();
        if(commands.equals("DISCONNECT")) {
                if(isroot) {
                        logger.info("---------------------\nDisconnecting
                            ␣the␣node␣...");

                        var trameFullDeco = new TrameFullDeco(77);
                        try {
                                broadCast(trameFullDeco);
                                System.out.println("DISCONNECTING␣ALL␣
                                    NODES");
                        }catch (IOException e) {
                                logger.info("Deco␣IOException");
                        }finally {

                        Thread.currentThread().interrupt();
                        logger.info("Disconnected␣Succesfully\n
                            ----------------------");
                        System.exit(0);
                        }
                }
                else {
                        try {
                                var ddl = new DataDoubleAddress(3,
                                    localInet,(InetSocketAddress) scDaron.
                                    getRemoteAddress());
                                var tame = new TrameAnnonceIntentionDeco(
                                    ddl);
                                logger.info("---------------------\
                                    nDisconnecting␣the␣node␣...");
                                if(connexions.size()==1) {
                                        System.out.println("ok");
                                        daronContext.queueTrame(tame);
                                        selector.wakeup();
                                }
                                else {
                                        broadCast(tame);
                                        selector.wakeup();
                                }
                        } catch (IOException e) {
                                logger.info("ProcessCommand␣IOException")
                                    ;
                        }
                }

        }
        else if(commands.startsWith("START")) {
                var lst = Arrays.asList(commands.split("␣")) ;
                if(lst.size()!=6) {
                        launchUsage();
                        return;
                }
```

```
                System.out.println(lst);
                try{
                        String jar = lst.get(1);
                        String qualifiedName = lst.get(2);
                        long start = Long.parseLong(lst.get(3));
                        long end = Long.parseLong(lst.get(4));
                        String fileName = lst.get(5);
                        putInData(jar, qualifiedName, start, end,
                            fileName);
                }catch(NumberFormatException e){
                        logger.info("␣WRONG␣START");
                        launchUsage();
                }
        }
        else {
                allUsage();
        }
}
```

6 Fonctionnalités

6.1 Fonctionnel

- La connexion des applications entre elles
- La table de routage lors d'une connexion
- La table de routage lors d'une déconnexion
- Trame d'Annonce d'intention de déconnexion
- Trame de Confirmation de changement de connexion
- Trame de suppression des tables de routage
- Trame FirstLeaf : Que les feuilles renvoient vers la root pour mettre à jour les tables de routage par le bas
- Trame FullTree : Que la root renvoi à tout le monde pour mettre à jour les tables de routages par le haut
- Trame Ping Envoi : Qui permet de verifier si une application est disponible
- Trame Ping Réponse : En réponse à la trame ping Envoi répond si l'application actuelle est disponible(donc n'as pas de taches)
- Deconnexion d'une feuille

6.2 Fonctionnel Partiellement ou pas entièrement implémenté

- Trame d'envoi des données aux applications en attente (pas implémenté)
- Trame d'envoi des données de déconnexion à traiter aux applications connexes(pas implémenté)
- Traitement des Checkers : cette partie n'a pas encore été intégré au code mais cela fonctionne a part.
- Reconnexion d'une application a une autre lors de la déconnexion de son père.(La route table ne se modifie pas)
- Déconnexion d'une application et re-connexion des application enfant à l'application père de l'application. (la route table ne se modifie pas)

6.3 Non Fonctionnel

- Convertion du projet en JAR

7 Difficulté rencontré

7.1 Difficulté sur la RFC

Avant la soutenance Au début n'ayant pas le sujet assez clair en tête, nous avons commencé à faire une RFC qui n'était pas du tout en rapport avec le sujet du projet, cependant à l'aide de Monsieur Carayol, nous avons mieux appréhendé le sujet et avons pu partir sur de bonnes bases. (On précise que ceci c'est passé bien avant la soutenance)

Après la soutenance Cependant, lors de la soutenance de la RFC, il nous a été montré que notre RFC n'était pas vraiment ce qui était attendu, mais rien de grave, c'étaient quelques petites choses à corriger. Nous avons alors demandé à notre enseignant de plus amples explications pour mieux cerner ce qui était demandé et améliorer notre RFC, aussi pendant ce temps nous avons essayé de commencer à travailler sur la partie code du protocole.

7.2 Difficulté sur le code

Comment commencer Au début du projet, nous ne savions pas exactement comment commencer, mais après avoir pataugé un peu, on a essayé plusieurs choses ce qui nous a permis de nous débloquer. Aussi, nous avons eu un problème de compilation lorsque l'on devait utiliser la classe des Tables de Routages dans la classe principale, la classe de routage n'était pas reconnue lors de la compilation et après avoir fait plusieurs tests et demandé au professeur, il s'avérait que le problème ne venait pas de nous, mais de notre terminal, en effet nous utilisions un terminal Windows pour compiler et lancer le code. Nous avons réglé notre problème en ajoutant l'option -cp lors du démarrage des applications

Les socketChannel Un autre problème que nous avons rencontré mais que nous nous y attendions déjà était le fait de reconnaître de quel socketChannel à été envoyé la trame. Nous avons décidé de changer un peu par rapport a la rfc et d'ajouter l'adresse qui a envoyé la trame et en regardant la table de routage on regardais de quel valeur venait l'adresse.

La table de routage La table de routage aussi a fait partis des difficulté de ce projet, en effet lorsque l'on supprimais les connexions la table de routage ne se mettais pas à jour. elle ne se mettais à jour uniquement si on ajoutais des éléments, on a cependant réussi, au début on pensait utiliser des trames et envoyer pour mettre a jour la table de routage pour les applications à proximité mais nous avons trouvé une méthode pour le faire sans, cependant la table de routage des autres applications doivent être modifier pour supprimer l'application qui s'est déconnecté.

Les trames Après la soutenance Beta nous avons rencontré un problème majeur qui provenait des trames envoyé mais que nous avons essayé de régler, cependant lorsque nous avons soit disant réglé le problème des trames, nous avons eu un problème d'envoi des paquets, en effet les paquet s'envoyaient parfois mais parfois ne s'envoyaient pas. Nous pensons avoir réglé ce problème mais maintenant à par intermittence le terminal qui est sensé recevoir la trame boucle sans rien récupérer et est en mode write.

Le débugage Malgré beaucoup de commits sur le git et de test, nous avons passé beaucoup de temps a débugger le code, une erreur pouvait en entrainer une autre et une erreur pouvait aussi quand elle était résolue en cacher une autre. Pour pouvoir débugger nous regardions a chaque fois les problème qu'affichait le terminal mais quand on ne trouvait vraiment pas, Monsieur Carayol nous a bien prêté main forte.

La deconnexion et la route table Nous avions réussi a faire que la table de routage se modifie lorsque une application se déconnecte et se connecte cependant après plusieurs modifications, cette fonctionnalité n'a plus marché on ne sait pas pourquoi.

La gestion du temps Nous avons d'abord voulus vraiment bien peaufiner la RFC avant de vraiment nous attaquer au code ce qui nous a pris un peu de temps, donc on a commencé à codé avec un peu de retard. Mais aussi lors des examens nous avons préféré mettre le projet de coté pour nous consacrer aux révisions pour les examens. Nous trouvons que avec les autres projet a coté et le début des stages nous n'avions pas assez de temps après la fin des cours pour se

consacrer un maximum sur le projet, surtout que lorsque l'on pouvait on passait une après-midi à essayer de trouver et corriger des bugs.

8 Division du travail

A propos de la division du travail entre binôme, cela à été assez équitable. On a fait le maximum pour éviter de trop charger une personne par rapport a l'autre. On essayait de travailler environ 3h par jours si ce n'est plus sur le projet weekend compris.

8.1 RFC

Pour la RFC, on a beaucoup travaillé de notre coté, mais on faisait des points pour assez vite comprendre la situation et échanger les idées entre nous et nous mettre d'accord dessus ou non.

8.2 Code

À ce niveau, on essayait d'avancer chacun de notre côté lorsque l'on pouvait, mais surtout, on a essayé un maximum de coder ensemble sur Eclipse à l'aide de l'extension CodeTogether comme ça, il était assez simple de savoir assez rapidement d'où venais l'erreur et on pouvait intervenir plus rapidement, lorsque l'on travaillait ensemble sur CodeTogether, on essayait d'envoyer un peu prêt le même nombre de commit chacun sur le git. Cependant lorsque les stages ont commencé il nous à été un peu plus compliqué de nous voir en après-midi, l'un n'était pas encore en stage tandis que l'autre si, on ne pouvait donc se voir que le soir ou le dernier weekend du rendu de projet. ce qui à amener a ce que celui qui n'etait pas en stage de travailler légèrement plus que l'autre.

8.3 Rapport

Le rapport à été mis à jours par une seule personne, cependant l'autre binôme donnait des points qu'il fallait changer ou qu'il fallait ajouter.

9 Annexe

9.1 Commentaires

On a trouvé le projet de réseau plutôt intéressant, un projet qui nous laissait beaucoup de liberté, où il fallait poser beaucoup de questions pour pouvoir se donner une ligne directrice, même si le projet était assez complexe et nous pensions que le projet serait très très long, nous avons pris du plaisir à travailler dessus.

9.2 Remerciement

Nous remercions l'équipe chargé de la matière Réseau pour le projet intéressant qui nous à été fournis mais aussi pour tout le soutient et idées que vous nous avez données tout le long du projet.