Laboratoire 4
Le HDLC et le réseau multipoint

CEG 3585 - Communication de données et réseautage

Université d'Ottawa

Professeur : Mohamed Ali Ibrahim

Noms et numéros des étudiants :
Gbegbe Decaho Jacques 300094197
Sissoko Ramatoullaye Bahio 300144949

Groupe 21

Date de soumission:31-03-2023

Table des matières:

1- Description du problème

L'objectif de laboratoire est dans un premier temps de se familiariser avec le réseau multipoint qui contient une station primaire et plusieurs stations secondaires. Puis dans un deuxième temps de mieux comprendre l'importance et rôle des protocoles. Ensuite nous aider à maîtriser le mécanisme de la fenêtre d'anticipation. Et enfin, d'étudier un protocole concret de la couche liaison de données, le HDLC.

2- Explication du programme de solution

Dans ce rapport, nous avons été amené à compléter l'une des méthodes présente dans la classe SecondaryHDLCDataLink fourni pour le présent laboratoire. La méthode en question est la méthode dlDataReques, cette méthode permet d'envoyer un message à la station primaire avec plusieurs trames-I du HDLC.

Toutes les classes fournies et autres méthodes dans le rapport étant complètes, nous n'avons pas eu à les revisiter à cet effet. Pour la résolution de ce problème, nous avons suivi les différents conseils donnés au courant du lab afin de nous retrouver.

Pour commencer, nous avons implémenté une boucle afin de construire plusieurs frames pour le début de notre travail. Screenshot 1.

Ensuite, nous avons procédé grâce à plusieurs recherches, nous avons initialisé des variables frames, nr, frameBit et un compteur a. tous entiers sauf frameBit qui est un String qui servira à récupérer les frames sous formes de bit et les concaténer en chaînes de caractères.

De plus pour remplir la boucle déjà commencé de la méthode nous avons initialisé une condition à remplir dans celle-ci (screenshot 2 et 3). Ces dernières nous ont permis de vérifier si les tableaux contenant les frames sont inférieurs, supérieurs ou nul par rapport à notre compteur.

De plus, pour le bon fonctionnement de notre classe, nous avons créé une nouvelle méthode creeFrame afin d'initialiser des frames car vide au début (screenshot 4).

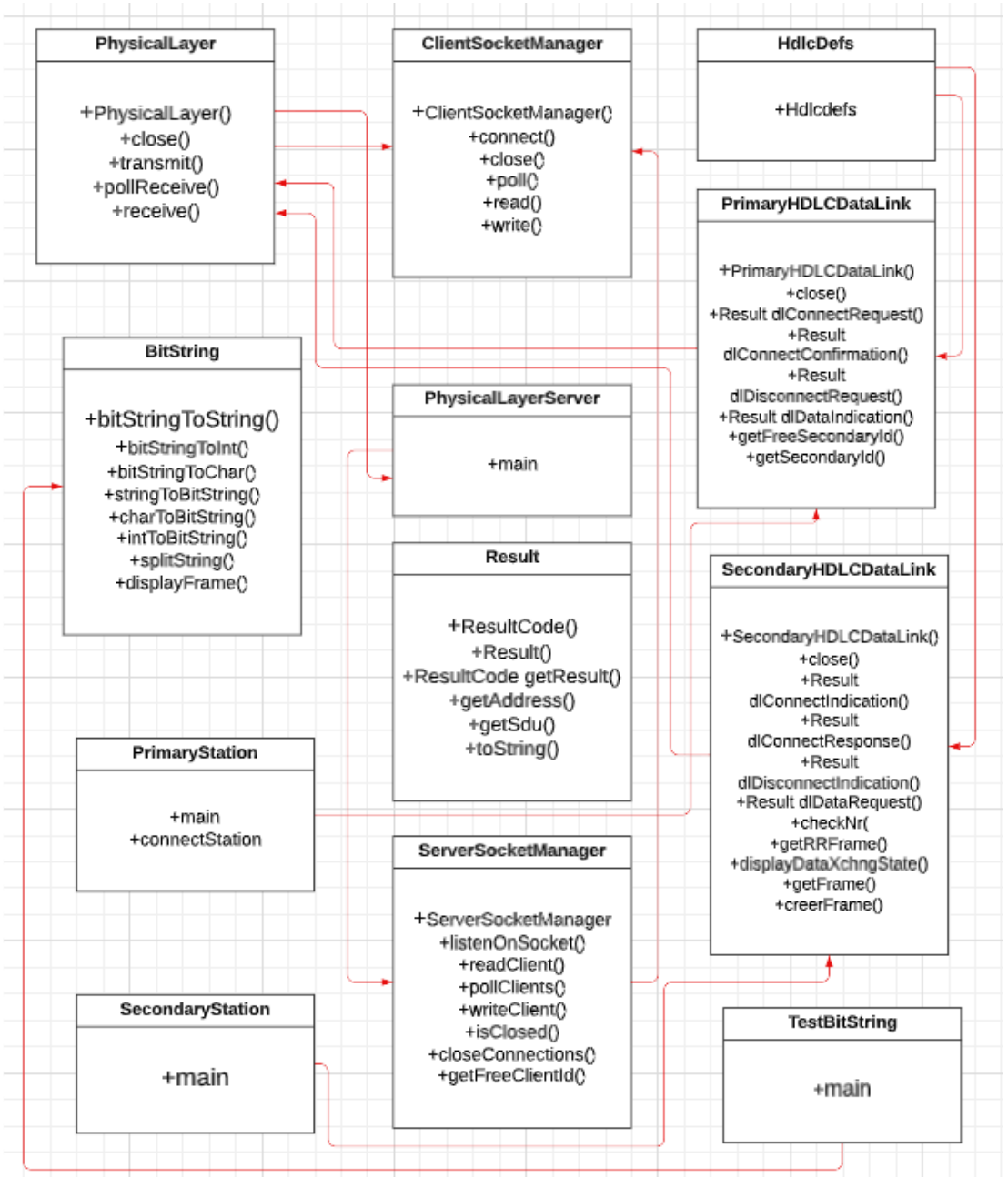
A la suite de ses manipulations, nous avons commencé nos tests en suivant la séquence imposée dans le document du lab afin d'obtenir les différents résultats (tous les screens suivant).

Nous avons utilisé divers outil java afin de faciliter notre travail notamment :

- BitString.displayFrame
- et différents raccourcis et appels de variables et méthodes diverses.

Nous avons été obligé de toucher à certaines autres méthodes afin de régler les divers bug rencontrés.

3- Document Représentatif (UML)



4- Capture d'écran

```
do {
    frame = getRRFrame( wait: true);
}while (frame.charAt(HdlcDefs.PF_IX) == '0');
```

figure 1 : screenshot 1

```java
while( a < dataArr.length || frameBuffer.size() > 0 )
{
    // Send frame if window not closed and data not all transmitted
    if( vs != rhsWindow && a < dataArr.length )
    {
        frameBuffer.add(frameBit = dataArr[a]);
        vs = ++vs % HdlcDefs.SNUM_SIZE_COUNT;

        creerFrame(frameBit,  frameNumber: a % HdlcDefs.SNUM_SIZE_COUNT,  isFinal: a == dataArr.length - 1);
        a++;
        displayDataXchngState( msg: "Data Link Layer: prepared and buffered I frame >"+BitString.displayFrame(frameBit)+"<");
    }
```

figure 2 : screenshot 2

```java
// Check for RR
frame = getRRFrame( wait: false); // just poll
if((frame != null) && (frame.charAt(HdlcDefs.PF_IX) == '0')) // have a frame
{
    nr = BitString.bitStringToInt(frame.substring(HdlcDefs.NR_START, HdlcDefs.NR_END));
    frames = checkNr(nr, rhsWindow, windowSize);
    rhsWindow = (rhsWindow + frames) % HdlcDefs.SNUM_SIZE_COUNT;

    // Remove transmitted frames from buffer
    for (int j = 0; j < frames; j++)
        frameBuffer.remove( index: 0);
    displayDataXchngState( msg: "received an RR frame (ack) >"+BitString.displayFrame(frame)+"<");

}
```

figure 3 : screenshot 3

```java
private void creerFrame(String info, int frameNumber, boolean isFinal)
{
    String finalBit = isFinal? "1" : "0";
    String address = BitString.intToBitString(stationAdr, HdlcDefs.ADR_SIZE_BITS);
    String control = HdlcDefs.I_FRAME + BitString.intToBitString(frameNumber,  numBits: 3) + finalBit + BitString.intToBitString(vr,  numBits: 3);
    String frame = HdlcDefs.FLAG + address + control + info + HdlcDefs.FLAG;

    physicalLayer.transmit(frame);
}
```

figure 4 : screenshot 4

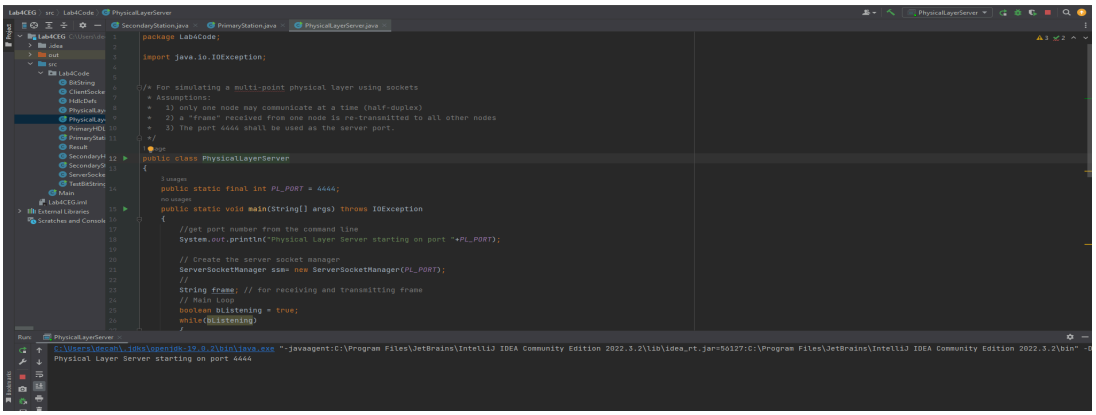


figure 5 : Lancement du thread par la classe PhysicalLayerServer

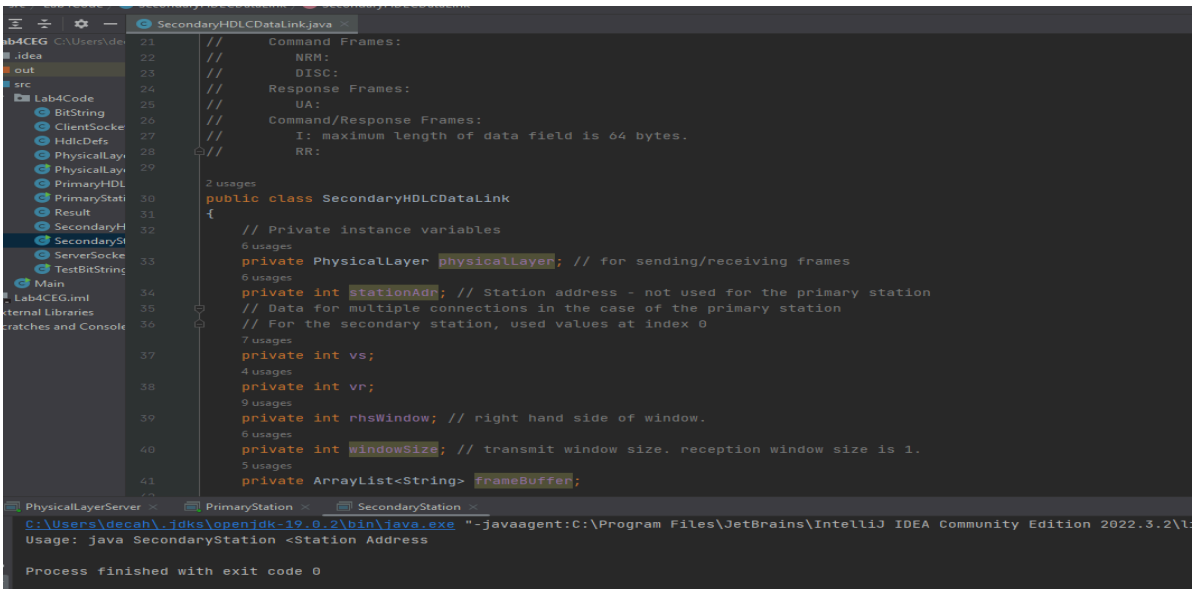


figure 6 : Sortie de la compilation de la classe SecondaryStation

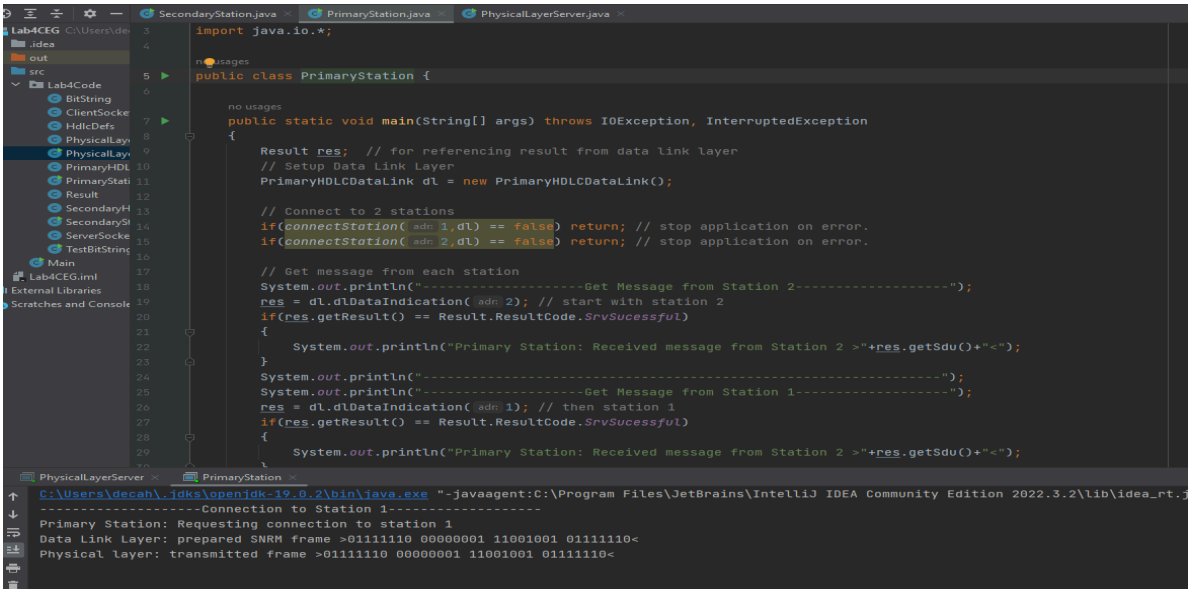


figure 7 : Sortie de la compilation de la classe PrimaryStation

5- Discussion

Dans ce laboratoire on a utilisé toutes les données misent à notre disposition pour mener à bien le travail demandé. Néanmoins on a eu quelques difficultés notamment la simulation de la classe SecondaryHDLCDataLink. Pour régler ce problème nous avons été contraint à modifier certaines méthodes de la classe comme c'est le cas de la méthode checkNr et getRRFrame, ainsi que la simulation du programme qui contre attente ne fonctionnait pas quelque soit la méthode que nous utilisions mais finalement après cela elle a pu se faire sans accroc. Ceci a été la partie qui nous a donné le plus de mal à être réalisé mais toutefois ce fut une belle expérience de nous familiariser un peu plus avec les thread. Finalement, ce fut une expérience différente qui sûrement nous sera d'une utilité dans le futur.

6- Conclusion

L’objectif du laboratoire 4 était de compléter et de faire marcher la simulation du programme fourni lors du laboratoire. C'est-à-dire, notre tâche est de compléter la méthode dlDataReques() de la classe SecondaryHDLCDataLink et de compléter la classe pour son bon fonctionnement. Toutes les autres méthodes des classes sont complètes. Tout cela dans le langage de notre choix, le nôtre a été conçu avec le langage de programmation Java.

En somme, ce laboratoire a été gratifiant pour nous car il nous a permis d’approfondir nos connaissances en matière de thread, d'interactions entre différents programmes sur une mémoire partagée. On a pu affronter et régler les différents problèmes rencontrés lors de l’implémentation des différentes classes et nous nous en servirons pour les prochaines expériences.