|  |  |  |
| --- | --- | --- |
| Université d’Ottawa  Faculté de génie  École d’ingénierie et de  technologies de l’information | UOlogoBW | University of Ottawa  Faculty of Engineering  School of Information  Technology and Engineering |

CEG 3585

Introduction aux réseaux et communications

Laboratoire III

26 mars 2014

Présenté à

Naouar Yaagoubi

*Présenté par*

Mathieu Jobin

5975019

Andréas Kaytar-LeFrançois

6467076

**Table des Matières**

*Chapitre Page*

[1 Introduction 1](#_Toc383615563)

[1.1 Matériel et Composantes 1](#_Toc383615564)

[2 DIAGRAMME DE RÉSEAU 2](#_Toc383615565)

[2.1 Pseudocode 2](#_Toc383615566)

[2.1.1 Serveur 2](#_Toc383615567)

[2.1.2 Clients 3](#_Toc383615568)

[2.2 Diagrammes de fluence 4](#_Toc383615569)

[3 ALGORTHIME DE CONCEPTION 6](#_Toc383615570)

[3.1 Les assomptions 6](#_Toc383615571)

[3.2 Classes 6](#_Toc383615572)

[4 discussion Et conclusion 8](#_Toc383615573)

[4.1 Discussion 8](#_Toc383615574)

[4.2 Conclusion 8](#_Toc383615575)

**Table des illustrations**

*Illustration Page*

[Figure 2‑1 Diagramme de fluence du serveur (A) 4](#_Toc383615576)

[Figure 2‑2 Diagramme de fluence des clients (B et C) 5](#_Toc383615577)

# Introduction

L’objectif de ce laboratoire est de concevoir un réseau composé d'un serveur (primaire) et de deux postes (secondaire). Cette conception nous permettra de nous familiariser avec le contrôle d'accès au médium, comprendre l'importance des protocoles de communication, maitriser le concept de fenêtre coulissante et en concevoir une. Finalement, analyser la couche secondaire soit le HDLC.

Il est important de noter que tous les codes Java sont disponibles dans le fichier joint avec le rapport.

## Matériel et Composantes

* Ordinateur de bureau
* Logiciel de Java
* Manuel de laboratoire

# DIAGRAMME DE RÉSEAU

Dans cette section, nous démontrons à l'aide de diagramme la conception et le déroulement de notre simulation.

## Pseudocode

Note : le pseudocode a été révisé et modifié à partir du power points fournis pour le laboratoire, car il contenait certaines erreurs et incohérences.

### Serveur

**P0**

send “SNRM” to B /\*initialisation\*/

send “SNRM” to C/\*initialisation\*/

wait for “UA” from B

wait for “UA” from C

**P1**

send “RR,\*, P” to B /\*envoie d’un bit « pol » à B pour voir s’il a quelque chose à envoyer\*/

received “RR,\*,F” from B? /\* voir si on reçoit un bit final de B\*/

yes

goto P2

no

received “I, \*, \*” from B

send “RR,\*, P” to C

goto P5

**P2**

send “RR,\*,P” to C /\*envoie d’un bit « pol » à C pour voir s’il a quelque chose à envoyer\*/

received “RR,\*,F” from C? /\* voir si on reçoit un bit final de B\*/

yes

goto P3

no

received “I, \*, \*” from C

goto P6

**P3**

A ask any frame I to send to B? /\*si on a un bit final de C, on a terminé avec C, voir si I frame pour B\*/

yes

send “I,\*,\*” to B

go before P1

no

go before P1

**P4**

A ask any frame I to send to C?\*/ voir si on a une frame I à envoyer à C\*/

yes

send “I,\*,\*” to C

go before P1

no

go before P1

**P5**

Is the frame to A or C? /\* traiter la frame provenant de B\*/

to A

consume

go before P1

to C

buffer the frame to send

goto P4

**P6**

is the frame to A or B? /\* traiter la frame provenant de B\*/

to A

consume

go before P1

to B

buffer the frame to send

goto P3

### Clients

**S0**

wait for “SNRM” from A /\*si on reçoit SNRM on répond à A\*/

send “UA” to A

**S1**

wait for frame from A /\* la fame peut être de typeRR ou I \*/

“RR,\*,P” de A? /\*voir si on reçoit un bit pol de A pour ensuite envoyer si on a quelque chose de type I\*/

yes,

any frame I to send? /\* frame RR \*/

yes

send “I,\*,\*” to A

goto S1

no /\*si on a rien de type I à envoyer on envoie u bit final\*/

send “RR,\*,F” to A

goto S1

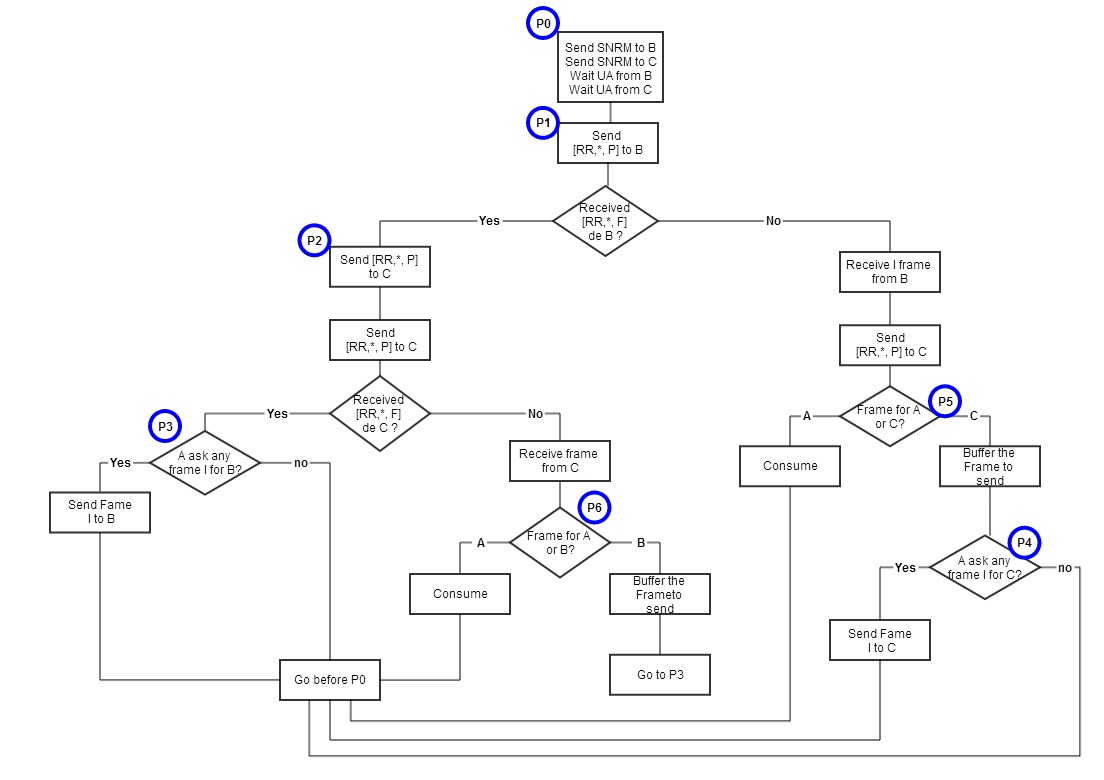
no

Receive I frame from A

Consume

goto S1

## Diagrammes de fluence



Go before P1

Figure 2‑1 Diagramme de fluence du serveur (A)

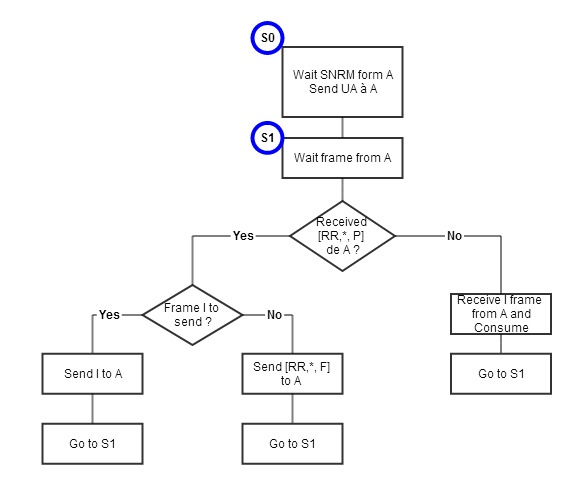


Figure 2‑2 Diagramme de fluence des clients (B et C)

Notes : Ces deux chartes ont été dérivée à partir de la présentation power point fournie pour le laboratoire, mais a été modifiée suite à notre analyse afin de corriger certaines erreurs

# ALGORTHIME DE CONCEPTION

Dans cette section, nous décrirons comment nous avons procédé pour transformer nos diagrammes de réseau en code Java.

## Les assomptions

Afin de réaliser le laboratoire de manière simple, plusieurs assomptions nous ont été proposées.

1. Toutes les stations s’initialiseront et garderont le N(S) et N(R) et N(R) corrects de la transmission et de la réception.
2. Nous n’avons pas besoin de réaliser de timeout, de FCS ni de bourrage de bit.
3. Nous pouvons utiliser des strings pour représenter des flags, adresses et champs de contrôle.
4. Des sockets Java sont utilisées pour simuler la couche physique soit :

* Socket : d`un client a un serveur
* ServerSocket : d`un server a plusieurs clients

## Classes

Voici les classes importantes de notre code avec leurs descriptions :

Connection.java:

* Classe qui gère les sockets pour permettre d'envoyer et de recevoir des données nativement avec java.

MyClientWin.java:

* Le client. Cette classe comprend le GUI du programme chat, ainsi que la logique des stations secondaires du protocole HDLC.

MyServer.java

* Le serveur. Cette classe contient la logique de la station primaire du protocole HDLC.

NetFrame.java

* La classe qu'on utilise pour wrapper les frames. Cette classe est en charge de former des frames de spécification HDLC à partir des informations fournies par le client ou le serveur. Elle transforme toutes les données en un string binaires afin de les transmettre avec une connexion.

Window.java

* fournit une fenêtre coulissante que Connection utilise.

Pour le code utilisé, voir la deuxième pièce jointe au courriel. Sinon, suivre le lien suivant où il est possible de télécharger les fichiers java que l’on peut ensuite exécuter.

<https://github.com/aklef/CEG3585Lab3>

Il suffit de cliquer sur « download ZIP » au côté droit de la page. Les fichiers sont tous inclus il ne suffit que les exécuter dans Eclipse.

# discussion Et conclusion

## Discussion

Globalement, nous pouvons dire que nous avons atteint tous nous objectifs puisque les résultats pratiques concordent parfaitement avec les résultats théoriques, l’échange de frame se fait sans erreurs et selon les spécifications du manuel de laboratoire.

Nous pouvons affirmer aujourd’hui que nous maitrisons l’utilisation de java (éclipse) pour ce qui est des applications en rapport avec ce que nous avons vu au laboratoire.

Ce laboratoire nous a également permis de comprendre l’intérêt et l’utilité de la fenêtre coulissante, du HDLC ainsi que de l’échange de frame.

Nous avons également pu comprendre à quel point l’étape théorique (diagramme de fluence, pseudocode ainsi que compréhension des concepts d’échanges de données utilisés) est importante, car toute erreur dans ces étapes conduit inévitablement à une erreur dans l’implémentation.

## Conclusion

En somme, les résultats théoriques concordent avec les résultats expérimentaux. Cette expérience s’avère donc concluante.