République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique .

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Système d'information

Sujet :

Conception et réalisation d'un moniteur EEG en mode Client Serveur

Présenté par : Benelkaid Ismail

Promoteur : M^me^ OUKID
Encadreur : M^r^ ALLAM

Organisme d'accueil : CDTA «Centre de développement de techniques avancées ».

Soutenue le: 03 / 10 / 2005 , devant le jury composé de :

M^R^ Bounouar	**Président**
M^R^ Kabbir	**Examinateur**

-Session Octobre / année 2005.

Remerciements

Nous remercions avant tout le bon dieu qui nous a aidé à réaliser ce modeste travail.

Nous tenons à remercier mes très chers parents pour leurs soutiens durant Toute ma carrière, leurs efforts constants dans mes études, Et Pour leurs encouragements.

Nous tenons à remercier ma promotrice *M^me^ OUKID* et *MON ENCADREUR* M^R^ ALLAME Pour leur aide, leur patience, leur disponibilité, et leur compréhensibilité.

Nous remercions les membres du jury pour nous avoir fait l'honneur de juger notre travail.

Nous remercions M^lle^ massouda farah et M^lle^ Boumahdi fatima et M^r^ Mouhamed Auchaaben pour tout ce qu'ils ont fait pour nous

Nous remercions spécialement et infiniment les agent de notre bibliothèque et particulièrement M^r^ Djamel pour sa modestie.

Nous remercions, de tout coeur, tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Dédicace

Ce mémoire est dédié :

A mes très chers parents pour leur soutiens durant Toute ma carrière, leurs efforts constants dans mes études, Et Pour leur encouragements.

A tous mes frères et mes sœurs, à mon cœur mouhamed et yusra, et à tous les membres de ma famille.

A tous mes amis ben rabeh, rachide, younes, mahfoud, halim, amar, samir. Et particulièrement à moufida,
et à tous me amis qui j'oublie.

Table des matières

Résumé

Anglais

The acquisition and data processing in real time is very important in a big number of domains; one requires of the new systems an instantaneous answer. The described work has for purpose to acquire the electric signals of the human brain, to amplify them via a biologic amplifier of an analog EEG apparatus, and to inject them in a PC thanks to a National Instrument digital acquisition (DAQ) card inserted on PCI bus, then to process these signals by a program worked out from the new version of the National Instrument LABVIEW 7 Express software. The real time answer is primordial in this kind of measurement instrument. A program has also been designed to share signals acquirements toward others computers within a customer-server network architecture by using Labview.

Français

L'acquisition et le traitement de données en temps réel est très important dans grand nombre de domaines; on exige des nouveaux systèmes une réponse quasi instantanée. Le travail décrit a pour but d'acquérir des signaux électriques du cerveau humain, de les amplifier via un amplificateur biologique d'un EEG analogique, et de les injecter dans un PC grâce à une carte d'acquisition de NATIONAL INSTRUMENT montée sur bus PCI, puis de traiter ces signaux par le biais d'un programme élaboré à partir de la nouvelle version du logiciel LABVIEW 7 de NATIONAL INSTRUMENT. La réponse en temps réel est primordiale dans se genre d'instrument de mesure. Un programme a été élaboré également pour partager les signaux acquis vers d'autres microordinateurs type PC dans une architecture en réseau client/serveur en utilisant Labview.

Mots clés : EEG, filtrage, acquisition des signaux, FFT, Labview, architecture réseau client/serveur.

Liste des figures :

Introduction générale

Ils existent plusieurs techniques utilisée en médecine, entre autre les deux techniques : l'électrocardiographie et l'électroencéphalographie. Le préfixe 'électro' vient, non pas d'électronique, mais d'électrique, car ces deux techniques exploitent une activité électrique d'un organe (cœur ou cerveau).

Nous intéressons par le électroencéphalogrammes (EEG) . Les applications de EEG sont uniquement médicales, servant à la détection de maladies et de malformations. L'électroencéphalographie a permis d'avancer considérablement sur notre connaissance du cerveau, avant d'avoir aujourd'hui d'autres méthodes plus efficaces. Cet appareil est néanmoins toujours très utilisé en milieu hospitalier.

L'électroencéphalographie (EEG) est une technique qui permet de mesurer l'activité électrique du cerveau provoquée par le courant électrique généré dans les neurones. Cette technique a été crée en 1924 par Hans Berger, en Allemagne. L'analyse se fait en plaçant un certain nombre d'électrodes, seules ou par paires, à la surface du crâne. Si les électrodes sont placées individuellement, une " électrode de référence " commune est alors utilisée. Chaque électrode doit relever une mesure de tension de surface, puis transmettre ce signal, qui est ensuite amplifié et enregistré. À l'origine, ces enregistrements étaient tracés sur papier à l'aide d'un stylet.

Les inconvénients des EEG analogiques sont essentiellement :

- La perte de calibration des tracés de l'électroencéphalographe.
- Usure et détérioration des plumes qui ne peuvent être remplacées.
- Exploitation manuelle des tracés de l'électroencéphalographe par les médecins.
- Pas de possibilités d'interconnexion de l'appareil avec un PC afin d'effectuer le traitement des signaux.

Par conséquent, les ingénieurs ont trouvé un moyen pour permettre l'amélioration des mesures des signaux électrique du cerveau ainsi que le traitement des données récoltées.

Aujourd'hui, les sorties sont enregistrées par un numériseur, puis sauvegardées sur ordinateur (EEG numérique). Dans certaines applications, les électrodes sont placées sous le cuir chevelu ou en contact direct avec le cerveau.

Notre travail consiste à réaliser et concevoir un logiciel permettant l'acquisition et l'exploitation des signaux EEG et de développer et utiliser une bibliothèque pour partager ces signaux à partir de réseaux.

Afin d'atteindre le principale but de ce travail nous avons organisé notre mémoire comme suit :

Le premier chapitre sera consacré à l'étude du traitement numérique du signal, il comprend dans sa première section des généralités sur les signaux numérique et le transformée de fourier rapide qui sera utilisée pour extraire les fréquences contenus dans les signaux acquis pour aider le médecin de trouver les fréquences qui peuvent l'aider pour savoir l'état du malade, et dans la deuxième section de ce chapitre , nous présenterons les filtres RII avec quatre algorithmes : le filtre de Butterworth , le filtre de Tchebycheff , le filtre de Inverse-Tchebycheff et le filtre de Cauer ou elliptique , tous ces algorithmes sont implémentées par notre logiciel , ceci permet d'aider le médecin de filtrer certaines bande de fréquences , pour concentrer sur d'autres.

Dans le deuxième chapitre, une présentation du protocole TCP/IP et ses caractéristiques générales et des sockets seront présentées, le protocole TCP/IP peut être utilisé pour communiquer au sein d'un ensemble de réseaux interconnecter. Grâce à ce protocole les applications peuvent partager les signaux EEG de façon sûre et fiable, nous présenterons aussi dans les sockets le modèle client serveur que nous intéresse.

Après avoir posé quelques repères théoriques sur ces nouvelles technologies, nous aborderons dans le troisième chapitre, le développement de notre logiciel, suivant le cycle de vie en cascade, et le langage de modélisation UML. Nous verrons dans un premier temps, les fonctionnalités du système à développer. Puis, dans un deuxième temps, une conception de notre logiciel sera faite. Enfin, nous verrons l'implémentation de notre conception.

Nous achevons notre mémoire par une conclusion générale, et des perspectives.

1. Problématique

Le projet de numérisation d'un EEG analogique, permet non seulement de se débarrasser des inconvénients de la partie mécanique ,mais aussi elle permet de numériser les signaux électriques prélevés du cerveau , de les visualiser sur une interface graphique sur PC et d'effectuer des traitements sur les signaux électriques.

Un autre avantage de la numérisation est la création d'une base de données patients contenant les informations relatives au patient, au médecin ainsi que les fichiers d'acquisition des différents patients. Le médecin pourra alors gérer sa base de données ce qui lui facilitera considérablement le travail.

Dans cette section une présentation des différents dispositifs utilisée lors du développement de note logiciel sera établie :

➢ **Description du dispositif du EEG :**

Le synoptique du dispositif est donné par la figure.1, les signaux sont prélevés du cerveau grâce à une têtière, puis amplifiés et filtrés grâce à l'amplificateur biologique du EEG analogique. Nous utilisons deux cartes de la société « NATIONAL INSTRUMENTS », la premier, la BNC2090, sert d'interface entre les connecteurs BNC et la deuxième carte la PCI-M I O-16 E-4, qui permet d'acquérir et de numériser le signal, elle est placée sur bus PCI.

Le signal ainsi numérisé peut être visualisé sur l'interface graphique réalisée à partir du logiciel de la même entreprise, le LABVIEW 7.

La puissance de ce logiciel permet la réalisation de système de mesure capable d'effectuer des mesures en temps réel, ajouté à cela la richesse de sa bibliothèque de traitement du signal qui nous permet un large éventail de traitement sur les signaux prélevés.

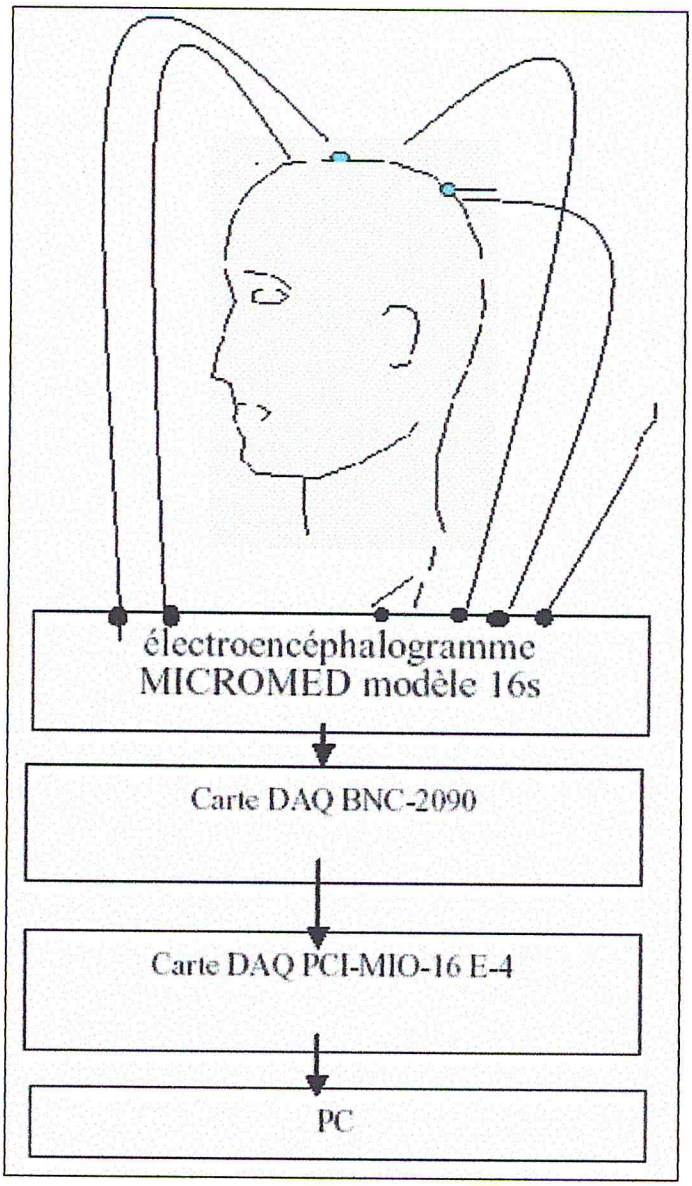


Figure 1 : **Dispositif de numérisation d'un EEG analogique.**

1. Caractéristiques des cartes d'acquisition :

Nous intéressons par deux cartes d'acquisition :

1) la BNC 2090 :

la BNC est un accessoire disposant de (figure.2):

1. 16 entrées analogiques référentiel et 8 différentiels sur connecteurs BNC configurable par switch.
2. D'une entrée connecteur à 68 pins.

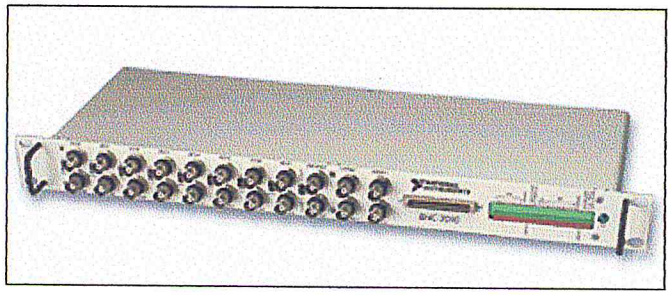


Figure 2 : BNC 2090.

La BNC2090 peut être utilisée de deux manières :

- La première consiste à l'utiliser comme étant un conditionneur de signaux, ce qui veut dire que nous pouvons la configurer de manière harde pour la rendre équivalente à un filtre ou à un diviseur de tension.
- La deuxième, est de l'utiliser comme une interface entre l'élément que l'on veut mesurer et la carte DAQ située sur le PC, les signaux prélevé peuvent être analogiques ou numériques.

Dans notre application nous utilisons les 16 entrées BNC pour connecter les files issues de la têtière.

En sortie la BNC 2090 dispose d'un connecteur à 68 pins qui sera relier à la carte DAQ « PCI-M I O 16 E 4 » par l'intermédiaire d'un câble (SH 6868).

2) La PCI MIO-16 E 4 :

la carte d'acquisition est donnée par la figure.3

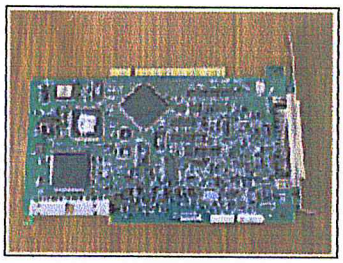


Figure 3 : la PCI MIO-16 E 4 .

➢ **caractéristiques :**

- 16 entrées référentielles et 8 entrées différentielles, configurable par soft, à partir du programme.
- Résolution de 12 bits
- Fréquence d'échantillonnage :
  - 500K échantillons/s par canal.

- 250Kéchantillons/s pour tous les canaux.

Le gain est sélectionné par soft le tableau suivant donne les valeurs du gain pour les deux modes :

Gain par canal (configurable par soft)	Les gammes (configurable par soft)	
	Bipolaire	Unipolaire
0.5	-10V à +10V	-
1	-5V à +5V	0 à 10V
2	-2.5V à +2.5V	0 à 5V
5	-1V à +1V	0 à 2V
10	-500mV à +500mV	0 à 1mV
20	-250mV à +250mV	0 à 500mV
50	-100mV à +100mV	0 à 200mV
100	-50mV à +50mV	0 à 100mV

- tension maximum de travail entre –11V et +11V.
- taille du buffer : 512 échantillons.

2. Objectif

- Concevoir et réaliser un logiciel permettent l'acquisition et l'exploitation des signaux EEG,
- Implémentation des algorithmes : le filtre de Butterworth , le filtre de Tchebycheff , le filtre de Inverse-Tchebycheff et le filtre de Cauer ou elliptique du traitement numérique du signal,
- Développer une bibliothèque (DLL) pour partager les signaux EEG à partir de réseaux.

CHAPITRE I : TRAITEMENT NUMERIQUE DU SIGNAL

Introduction

Dans ce chapitre, nous rappelons certaines notions du traitement numérique du signal qui sont utilisée dans les chapitres suivants de notre mémoire.

Le mot signal désigne le résultat de la mesure d'une grandeur physique. Sans restreindre en aucune façon notre discours, le résultat obtenu sera vu comme une fonction du temps. On dira alors que le signal est analogique ou à temps continu si la mesure est disponible de façon continue à tout instant. Un signal est dit numérique à temps discret si la mesure n'est observée qu'à des instants discrets particuliers en général régulièrement espacés. Le traitement numérique du signal TNS consiste à traiter des signaux à temps discret. L'émergence du TNS est liée, comme pour de nombreux autres domaines, au perfectionnement des composants électroniques. Ce dernier a stimulé l'imagination des scientifiques et des futurologues, engendrant du même coup de nouveaux problèmes et besoins et, avec eux, la nécessité de disposer d'outils encore plus puissants. Les micro processeurs, dédiés à l'origine aux traitements purement informatiques, ou à des fonctions simples de contrôle, sont désormais indispensables en TNS. Ils ont facilité la mise en place d'algorithmes qu'il aurait été bien difficile de mettre en oeuvre autrement. Il suffit pour s'en convaincre de regarder du côté des applications grand public telles que télévision haute définition, radiodiffusion numérique, téléphonie mobile, applications multimédia, etc. Tous ces services nouveaux utilisent largement le traitement numérique du signal en mettant en oeuvre des algorithmes parfois extrêmement complexes et nécessitant des puissances de calcul considérables. Pour se limiter à quelques exemples, on peut citer pêle-mêle:

- la suppression du bruit de fond lors d'une transmission téléphonique à partir de l'habitacle d'une voiture;

- le traitement du signal reçu en téléphonie mobile;
- e codage de l'image et de la parole dans un système de visiophonie;
- la reconnaissance et la synthèse de la parole;
- la détection de défauts dans une pièce mécanique et la maintenance préventive;
- la localisation par sonar des bancs de poissons;
- l'évaluation par radar des position et vitesse d'une cible;
- l'inversion de profil sismique pour la recherche pétrolière;
- l'analyse des vibrations d'une plate-forme pétrolière;
- l'analyse des signaux électro-encéphalographiques pour l'aide au diagnostic médical;
- la réception d'informations délivrées par le système GPS de navigation par satellites;
- la synthèse électronique de sons musicaux;
- le traitement de signaux acoustiques provenant de plusieurs microphones, etc.

Ces problèmes sont loin d'être complètement résolus et font encore l'objet de recherches.

1. Signaux numériques

1.1. Généralités

En TNS « «Traitement numérique de signal », l'information manipulée se présente sous la forme d'une suite de valeurs numériques issues de l'observation des grandeurs physiques. Cette suite porte le nom de signal à temps discret ou signal numérique.

Note: le terme signal numérique est ambigu car il a une signification différente en théorie des communications où il désigne le signal à temps continu associé à la suite de symboles (typiquement des 0 et des 1) du message numérique.

Quand la grandeur observée est une fonction $t \rightarrow X(t)$ du temps, une opération, dite de numérisation, est nécessaire pour effectuer le passage du signal analogique au signal numérique (figure 1). Elle comporte quatre phases.

- La grandeur observée doit tout d'abord être transformée en un signal électrique à l'aide d'un dispositif appelé transducteur comme par exemple un microphone. Par la suite, nous supposerons la présence implicite d'un tel dispositif dans la chaîne de numérisation et ne le représenterons jamais.

- Le signal électrique à numériser est ensuite appliqué à l'entrée d'un filtre passe-bas idéal, dit filtre d'anti-repliement, dont l'utilité sera justifiée par le théorème d'échantillonnage. Le signal ainsi filtré est noté Xa(t).
- Xa(t) est appliqué en entrée d'un CAN « convertisseur analogique/numérique » qui effectue les deux opérations suivantes.

- L'échantillonnage: on prélève toutes les T secondes les valeurs Xe(n) = Xa(n T). T est appelée la période d'échantillonnage et son inverse Fe la fréquence d'échantillonnage.

- La quantification : les échantillons prélevés sont codés sur un nombre fini de bits (généralement en complément à deux). Ainsi, si les informations à numériser proviennent d'une tension comprise entre -5V et + 5V et que l'on effectue une conversion sur 8 bits avec représentation en complément à deux, la valeur 3V sera codée par : $\frac{3V-(-5V)}{10} * 256 = 204.8 \rightarrow 205 - 128 = 01001101_2$, Quant à la valeur maximale 5V, elle est codée par 01111111_2.

Après numérisation, on dispose donc d'une suite de valeurs X(n) qui peuvent être traitées sur calculateur Dans certains cas, de plus en plus rares, on utilise une architecture dédiée (ensemble de registres, additionneurs, multiplieurs, etc.) qui permet d'obtenir des vitesses de traitement supérieures mais qui manque singulièrement de souplesse par rapport au calculateur

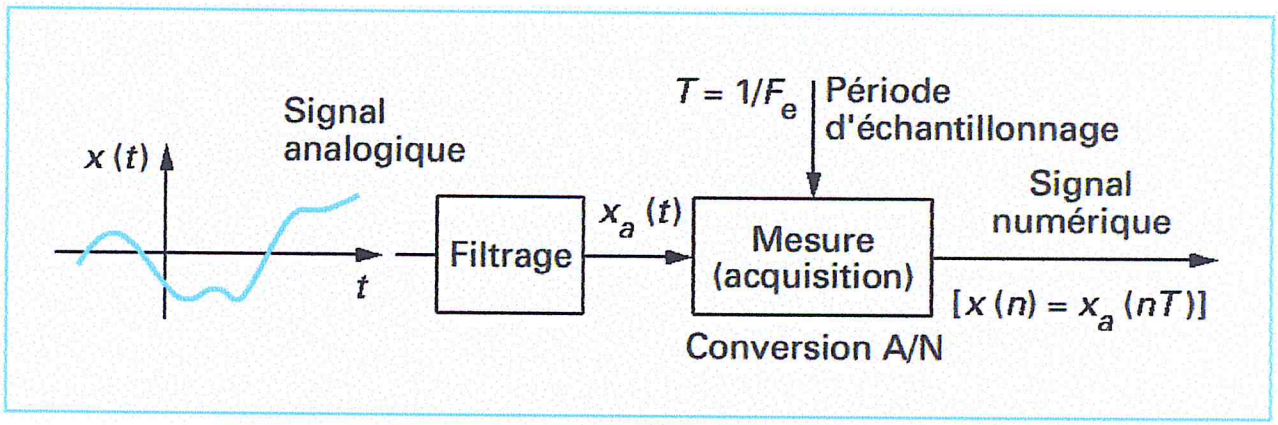


Figure I- 1 : Numérisation d'un signal filtrage anti-repliement conversion analogique numérique

2. Transformée de Fourier

Avant d'aborder les propriétés des signaux numériques, nous rappelons la définition de la transformée de Fourier des signaux temps continu.

- Le signal à temps continu Xa(t) possède (sous des conditions que nous supposerons remplies) une transformée de Fourier et celle-ci s'écrit :

$$Xa(F) = \int X(t) \exp(-2\pi \, j \, F \, t) \, dt$$

où F désigne la fréquence exprimée en hertz.

- Le signal Xa(t) est dit stable ou de module intégrable si :

$$\int_{-\infty}^{+\infty} |Xa(t)| dt < +\infty .$$

- Il est dit d'énergie finie ou de carré intégrable si :

$$\int_{-\infty}^{+\infty} |Xa(t)|^2 \, dt < +\infty .$$

- Il est dit de puissance finie si :

$$\lim_{T \longrightarrow +\infty} \frac{1}{2T} \int_{-T}^{+T} |Xa(t)|^2 \, dt < +\infty$$

- Les opérations de multiplication et de convolution s'échangent par transformation de Fourier, le produit de convolution :

$$(Xa * Ya)(t) = \int_{-\infty}^{+\infty} Xa(u) Ya(t-u) du .$$

a pour transformée de Fourier X(F)xY(F).

- Pour les signaux d'énergie finie on a la formule de Parseval qui s'écrit :

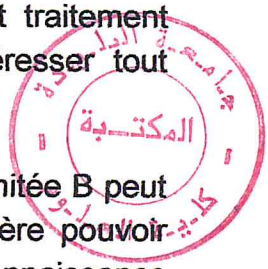
$$\int_{-\infty}^{+\infty} |Xa(t)|^2 \, dt = \int_{-\infty}^{+\infty} |Xa(F)|^2 \, dF$$

La fonction $|Xa(F)|^2$ est appelée spectre ou densité spectrale d'énergie (dse). D'après la formule de Parseval, elle s'interprète en effet comme la répartition de l'énergie le long de l'axe des fréquences. De façon générale, on peut dire qu'un signal qui présente des variations brutales possède de l'énergie dans les fréquences élevées et son spectre s'étale vers les hautes fréquences. le spectre d'un signal réel est une fonction paire.

On dit qu'un signal est à bande limitée si son spectre est nul hors d'une certaine bande de fréquence. Pour les signaux réels cette bande est de la forme (- B, +B) et on dit simplement que le signal est de bande B.

2.1. Théorème d'échantillonnage

Dans la mesure où une opération de numérisation précède tout traitement numérique de signaux à temps continu, il est naturel de s'intéresser tout particulièrement aux conséquences de celle-ci.



Le théorème d'échantillonnage énonce qu'un signal réel à bande limitée B peut être reconstruit de façon parfaite (on entend par là que l'on espère pouvoir reconstituer le signal original à temps continu à partir de la seule connaissance des échantillons du signal) si la fréquence d'échantillonnage est supérieure ou égale à deux fois la bande B.

Le point de départ de la démonstration est la formule de Poisson qui dit que tout signal Xa(t) à temps continu, et possédant une transformée de Fourier notée Xa(F), vérifie :

$$\sigma(F) = \frac{1}{T}\sum_{n=-\infty}^{+\infty} Xa(F - nF_e) = \sum_{-\infty}^{+\infty} Xa(kT)\exp(-2\pi jFkT)$$

où Fe = 1/T Remarquons que la fonction $\sigma_a(F)$ est une fonction périodique de période F_e qui ne dépend que de la suite des échantillons, ce que l'on résume parfois en disant que l'échantillonnage périodise le spectre. en passant aux fréquences réduites, c'est-à-dire en posant $f = F/F_e$, il vient :

$$\sigma(f) = \sigma_a(fF_e) = \frac{1}{T}\sum_{k=-\infty}^{+\infty} Xa((f-n)F_e) = \sum_{-\infty}^{+\infty} x(k)\exp(-2\pi jfk)$$

La fonction $\sigma(f)$ est appelée transformée de Fourier à temps discret TFtd de la suite $x(n)=xa(nT)$. Nous reviendrons en détail sur ses propriétés. Notons toutefois que $\sigma(f)$ est une fonction périodique de période 1.

La figure suivante représente la forme typique de $\sigma(f)$ pour un signal réel de bande B, lorsque $F_e > 2B$, c'est-à-dire pour b < 1/2 où on a posé $b = B/F_e$.

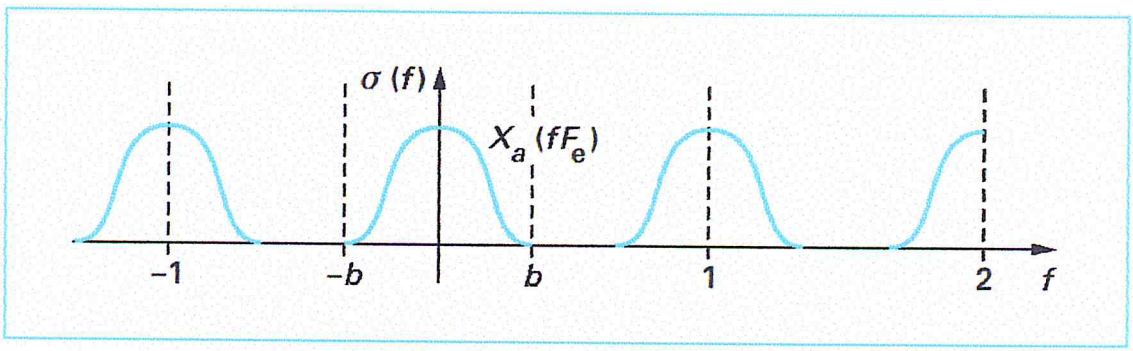


Figure I- 2 : Périodisation du spectre du signal échantillonné

On voit, dans ce cas, que la périodisation ne provoque pas de chevauchements entre les différents ***lobes*** composant $\sigma(f)$. on doit pouvoir reconstruire de façon parfaite Xa(F) à partir de $\sigma(f)$ et par conséquent, en déduire Xa(t) pour tout t. Plus précisément, la formule de reconstruction donnant Xa(t) en fonction de la suite x(n) a pour expression :

$$Xa(t) = \sum_{k=-\infty}^{+\infty} x(k)ha(t-kT)$$

Avec

$$ha(t) = \frac{\sin(2\pi Bt)}{\pi F_e t}$$

La fréquence minimale 2B permettant une reconstruction parfaite porte le nom de **fréquence de Nyquist,** lorsque $F_e < 2B$ l'opération de reconstruction parfaite n'est plus possible. Les « motifs» interfèrent de façon irréversible : on dit qu'il y a repliement de spectre.

Pour les signaux complexes le résultat se déduit de la même façon de la formule de Poisson. Mais, comme le spectre de Xa(t) ne possède plus la symétrie paire, la condition de non repliement s'écrit simplement $F_e \geq W$ où W

désigne la largeur totale du support du spectre. Cette expression donne bien $F_e \geq W$ pour les signaux réels puisque dans ce cas W= 2B.

2.2. Signaux types

Pour caractériser les phénomènes observés, ou encore pour étudier les comportements des systèmes de traitement, on a souvent besoin de faire appel à des suites de référence ou signaux types. Ceci ne surprendra personne puisque c'est aussi ce que l'on fait dans le cas du temps continu. Ainsi, l'analyse harmonique d'un système (linéaire) consiste à caractériser celui-ci en appliquant en entrée des sinusoïdes, et à mesurer en sortie l'amplitude et la phase de la sinusoïde obtenue. De même, l'observation d'un signal pendant un intervalle de temps de durée finie revient à le multiplier par une fonction rectangulaire qui vaut 1 sur cet intervalle et sinon, un filtre linéaire est caractérisé par sa réponse à une impulsion unité ou à l'échelon unité, etc.

1. **L'impulsion unité est définie par :**

$$\delta(0)=1$$
$$\delta(n)=0 \quad pour\ n \neq 0$$

2. **L'échelon unité est défini par :**

$$u(n)=1 \ \text{ pour } n=0$$
$$u(n)=0 \text{ sinon.}$$

3. **La porte rectangulaire de durée N est définie par :**

$$\pi_n(n)=1 \text{ pour } 0=n=N-1$$
$$\pi_n(n)=0 \text{ sinon}$$

4. **L'exponentielle causale et décroissante :**

$$x(n)=a^n u(n) \ \text{ avec } |a|<1$$

5. **L'exponentielle complexe (éternelle) est définie par :**

$$x(n)=\exp(2\pi j f_0 n)$$

Où la fréquence f0 a une valeur comprise entre 0 et 1.

3. Transformée de Fourier à temps discret

3.1. Définition

Trouver une représentation fréquentielle consiste à rechercher des périodicités dans un signal et à en mesurer les contributions. Il s'agit de la transformée de Fourier à temps discret (en abrégé TFtd) qui effectue le produit scalaire entre le signal et l'exponentielle complexe $\exp(2\pi jfn)$ pour tout f Sa définition est :

$$X(f) = \sum_{-\infty}^{+\infty} x(n)\exp(-2\pi jfn)$$

La TFtd joue pour les signaux numériques le rôle de la transformée de Fourier pour les signaux à temps continu. C'est une fonction à fréquence continue $f \in \Re$, périodique de période 1. il est d'usage de la représenter sur un intervalle de longueur 1, à savoir (-1/2, + 1/2) ou (0,1). la fonction $|X(f)|^2$ s'appelle le spectre « de puissance ». Dans la littérature ce terme est parfois aussi associé à la fonction |X(f)| « spectre d'amplitude ».

Inversement, on peut calculer x(n) à partir de X(f) :

$$x(n) = \int_{-\sqrt{2}}^{\sqrt{2}} X(f)\exp(2\pi jfn)df$$

3.2. La TFD : outil de calcul de la TFtd

3.2.1. Calcul pratique de la TFtd

Le calcul pratique de la TFtd, fonction de la variable continue f, ne peut être fait que pour un nombre fini N de points de signal et sur un nombre fini K(supérieur à N) de points de fréquence. En général on prend K points régulièrement espacés entre 0 et 1 de la forme f= k/K avec k={0...K-1}.

Considérons en exemple x (n) = exp ($2\pi jf_0n$) avec ne {0...N-1}, signal de durée finie. L'expression de sa TFtd est :

$$X(f) = \sum_{n=0}^{N-1} \exp(2\pi jf_0n)\exp(2\pi jfn) = \frac{\sin[\pi(f-f_0)N]}{\sin[\pi(f-f_0)]}\exp[-\pi j(f-f_0)(N-1)]$$

Calculée aux points f= k/K, elle est constituée de valeurs différentes de zéro, sauf si f_0 est exactement un multiple de 1/K

On voit sur ces figures qu'on est loin de la forme en sin (π f N)/ sin (πf) de la TFtd. Toutefois, pour s'en approcher, il suffit d'augmenter le nombre K de points de fréquence. Le choix de K joue donc sur la précision du tracé du spectre. Rappelons à ce propos que le nombre N de points d'observation agit pour sa part sur la résolution en fréquence. Ces deux notions ne doivent pas être confondues.

3.2.2. Définition de la TFD

Dans le calcul de la TFtd ,on peut sans perte de généralité, considérer que K= N. en effet si K>N, il suffit pour retomber dans le cas K= N, de compléter la suite x(n) par(K- N) zéros, ce qui ne change rigoureusement rien à la valeur de la TFtd calculée au point k/K (bourrage par des zéros). cela conduit à la définition suivante : on appelle transformée de Fourier discrète (TFD) de la suite finie {x(0) .....x(N-1)} la suite finie :

$$X(k) = \sum_{0}^{N-1} x(nT) W_N^{nk}$$

où k € { 0.....K-1} et W_N = exp (- 2πj/N). On montre facilement la formule inverse :

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}$$

Appelée transformée de Fourier discrète inverse (TFD).la TFD est l'outil de calcul de la TFtd. son succès tient à l'existence d'algorithmes rapides.

3.3. Transformée de Fourier rapide

La transformée de Fourier rapide (TFR), introduite en 1961 ,est une technique de calcul rapide de la TFD. L'algorithme de base, qui connaît de nombreuses variantes, utilise un nombre de points $N = 2^p$.

Pour en utiliser le principe, traitons l'exemple simple p = 3. On décompose la somme en termes de rang pair et de rang impair :

$$X(k) = [x(0) + x(2)W_8^{2k} + x(4)W_8^{4k} + x(6)W_8^{6k}]$$
$$W_8^k [x(1) + x(3) W_8^{2k} + x(5) W_8^{4k} + x(7) W_8^{6k}]$$

Et on réitère le procédé avec les deux termes obtenus :

$$X(k) = [x(0) + x(4)W_8^{4k}] + W_8^{2k}[x(2) + +x(6)W_8^{4k}]$$
$$W_8^k [[x(1) + x(5)\, W_8^{4k}] + W_8^{2k}[x(3) + x(7)\, W_8^{4k}]]$$

Le calcul des termes X(k) peut alors être schématisé par le diagramme classique représenté sur la figure I-3.

La cellule de calcul de base, que l'on voit clairement apparaître sur le schéma, est appelée papillon. Ainsi, partant des échantillons x(0) et x(4) de l'exemple choisi, on calcule deux nouvelles valeurs α et β à travers un « papillon de calcul ».

Les indices des termes x(n) apparaissent dans un ordre donné par le codage binaire renversé de n. Ainsi, le terme de rang 4 = 100_2 est x(001_2) = x(1). On parle d'adressage bit reversa Le tableau de la figure I-4 donne l'ensemble des correspondances.

3.3.1. Évaluation du nombre d'opérations

On peut vérifier que le calcul de N points de TFD nécessite N * log2 (N) additions complexes et (N/2)log2(N- N+ 1) multiplications complexes. On se contente en pratique de l'expression approchée N * log2 (N) multiplications - additions complexes. Un calcul direct aurait nécessité N^2 opérations de ce type, soit un gain de temps de l'ordre de N/log2(N). Ainsi, pour N = 1024, l'algorithme de TFR est environ 100 fois plus rapide que la programmation directe de la formule de TFD. en jouant sur les valeurs particulières des W_N^{Nk} on réduit encore ce nombre d'opérations. Il est en outre possible d'effectuer des décompositions plus complexes que la décomposition examinée ici en termes de rang pair et impair Cela fait apparaître des structures de calcul réduisant le nombre d'opérations au prix d'une programmation plus complexe.

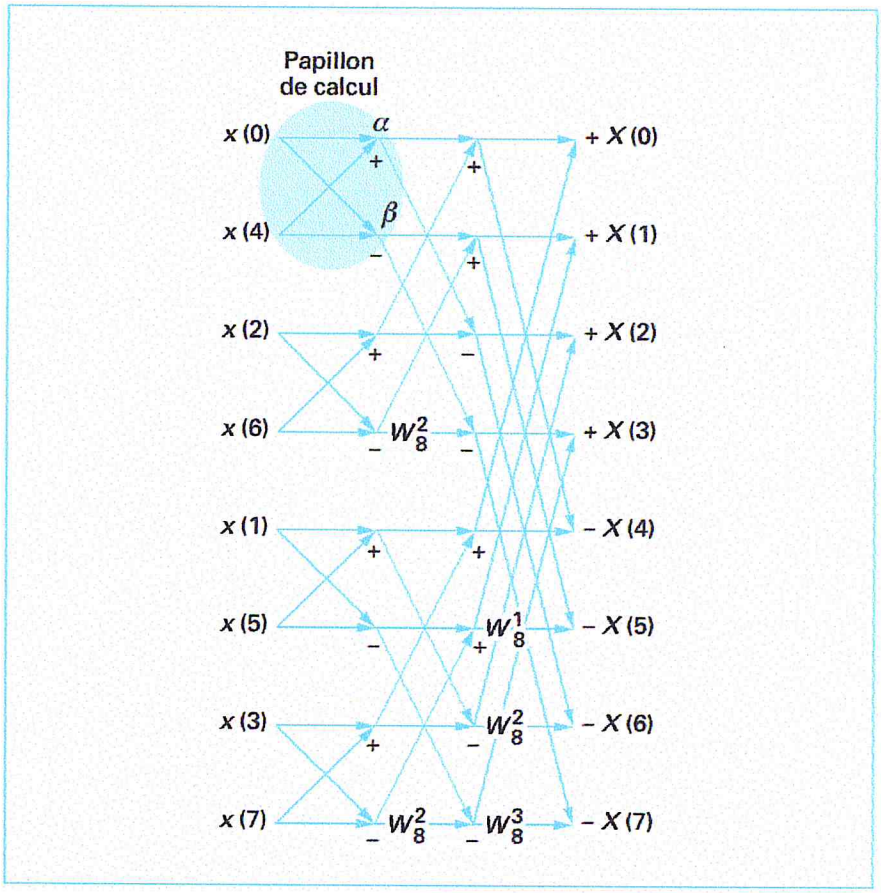


Figure I- 3 : Transformation de Fourier rapide {TFR) calcul des termes X (k)

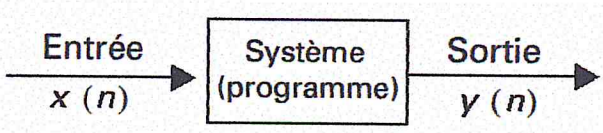
Rang	Codage binaire	Renversement	Élément
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Figure I- 4 :Tableau 1 - Correspondance entre rang, codage binaire et renversement

4. Filtrage linéaire :

Que ce soit pour réaliser un traitement utile (sélection d'une bande de fréquence, extraction d'un signal dans du bruit ...) ou que ce soit pour modéliser des déformations indésirables (transmission à travers un canal), on est conduit à envisager des transformations sur les signaux.

Une façon de décrire ces dernières est de faire appel à la notion de système (boîte noire) ayant une entrée et une sortie. Dans le cas des signaux numériques, il faut généralement entendre par système un programme qui, à partir du signal d'entrée x(n), engendre le signal de sortie y(n).



Dans la majorité des cas rencontrés en pratique, les systèmes peuvent être considérés linéaires et invariants dans le temps (système SLI). On entend par système invariant un système dont les caractéristiques n'évoluent pas avec le temps, une séquence d'entrée produisant la même séquence sortie quel que soit le moment auquel elle est appliquée. On utilise le plus souvent le terme de filtre pour désigner un système SLI.

On montre qu'un filtre numérique a pour équation d'entrée!sortie une relation de convolution discrète. Si x(n) désigne le signal numérique d'entrée et y (n) celui de sortie, on a

$$y(n) = \sum_{k=-\infty}^{+\infty} x(k)h(n-k) = \sum_{k=-\infty}^{+\infty} x(n-k)h(k) = (x * y)(n)$$

La suite h (n) qui caractérise le filtre est appelée sa réponse impulsionnelle (c'est en effet la réponse du système à l'impulsion unité $\sigma(n)$).

Pour un système, une des propriétés principales que l'on peut en attendre est sa stabilité. Le type de stabilité envisagé ici est dit EBSB (entrée bornée/sortie bornée), c'est-à-dire que le signal de sortie reste borné si le signal d'entrée est lui-même borné.

On montre qu'une condition nécessaire et suffisante de stabilité d'un filtre est que la suite h(n) soit de module sommable, ce qui s'écrit $\sum_n |h(n)| < +\infty$ Nous verrons que cette condition prend une forme particulièrement simple sur la transformée en Z de h(n).

Lorsque la réponse impulsionnelle est nulle pour n <0, on dit que le système est causal. Dans ce cas, le calcul de la sortie y(n) ne nécessite pas la connaissance des valeurs d'entrée d'indices supérieurs à n. toutefois, en TNS, la causalité n'a pas de caractère impérieux. il suffit de faire le calcul en temps différé en acceptant un certain retard à la connaissance, comme le montre la relation entrée! sortie

y(n) = ax (n+2)+bx (n+1)+cx(n)+bx(n-1)+ax(n-2)

D'après la relation (38) la réponse impulsionnelle de ce filtre est h(-2)=h(2)=a, h(-1)=h(1)=b et h(0)=c. Ce système n'est donc pas causal.

4.1. Transformée en z, outil d'étude des filtres

4.1.1. Définition

On appelle transformée en z de la suite x(n) la fonction de la variable complexe z définie par :

$$X_z(z) = \sum_{n=-\infty}^{+\infty} x(n) z^{-n}$$

Cette expression n'a de sens que dans une couronne du plan complexe de la forme R1<|z |<R2 que nous supposons non vide. Cette couronne définit le domaine de convergence de cette somme.

Si le signal est causal (x (n) = 0) pour n<0 on vérifie que $X_z(z)$ converge vers z tend vers l'infini et donc le domaine de convergence est de la forme |z| > R1.

Lorsque le cercle unité z = exp (2πjf) appartient au domaine de convergence, $X(f)=X_z(\exp(2\pi jf))$ est la TFtd de la suite x(n). Le cercle unité peut être gradué en valeurs de la fréquence f= θ/2π ou θ représente l'argument de z.

La transformée en z est l'outil privilégié de l'étude des filtres linéaires. Considérons en effet un filtre à temps discret de réponse impulsionnelle h (n), d'entrée x(n) et de sortie y(n). Notons respectivement H (z), X (z) et Y (z) leurs

transformées en z. Si on suppose, en plus, que le cercle unité appartient à leur domaine respectif de convergence, alors les Tftd respectives existent.

H(z) est appelé la fonction de transfert du filtre et H(f) son gain complexe ou réponse en fréquence. La fonction G (f) = |H (f)| s'appelle le gain du filtre et ϕ(f) = arg {H(f)} sa phase.

Le terme de gain en fréquence s'explique simplement: si on applique en entrée le signal $\exp(2\pi jf_0n)$, le signal en sortie a pour expression $H(f0)\exp(2\pi jf_0n)$. Ce résultat est mis à profit pour mesurer la réponse en fréquence d'un filtre (analyse harmonique) en régime sinusoïdal la comparaison des signaux d'entrée et de sortie fournit le nombre complexe $H(f_0)$.

On dit qu'un filtre est sans distorsion d'amplitude si son gain |Hz exp (2πτjf)| est constant et sans distorsion de phase si sa phase arg[Hz exp(2πjf)] est linéaire de la forme c_0f..

On notera que la notion de phase linéaire est liée à celle de retard pur.

4.1.2. Inversion de la transformée en z

Il est important de comprendre que l'inversion de la transformée en z nécessite que l'on se donne le domaine de convergence. Ainsi, à la fonction H (z) = 1 / (1 – az^{-1}), il correspond deux suites numériques différentes, suivant que l'on choisisse |z | > |a | ou |z | < |a|. La première est causale et a pour expression

$$h(n) = a^n u(n), n>=0$$

Et la seconde a pour expression

$$h(n)=-a^n u(-1-n), n>=0.$$

Ce qui les distingue dans le plan en z n'est pas l'expression de H (z) mais le domaine de convergence associé. A titre d'exemple, établissons ce résultat pour |z- < |a|. On a

$$H_z(z)=\frac{1}{1-az^{-1}}=\frac{z}{a}\frac{1}{1-a^{-1}z}=\frac{z}{a}\left(1+\frac{z}{a}+\ldots+\frac{z^n}{a^n}+\ldots\right)=\sum_{n=-\infty}^{-1}a^nz^{-n}$$

Où on a utilisé la condition $|a^{-1} z| < 1$, ainsi que le développement classique pour |u| < 1 :

$$\frac{1}{1-u} = 1 + u + ... + u^k + ...$$

Ce calcul est doublement instructif dans le cas où la fonction H (z) est une fraction rationnelle

on peut s'en inspirer pour l'inversion, après avoir effectué une décomposition préalable en éléments simples;

ce sont les modules des racines du dénominateur qui délimitent les couronnes de convergence. Ainsi, si on a N racines de modules distincts, on a (N + 1) suites différentes. Si aucune des racines n'est de module 1, on peut même trouver une couronne de convergence contenant le cercle unité : dans ce cas la suite associée est de module sommable. Si toutes ces racines sont strictement à l'intérieur du cercle unité, le domaine de convergence s'étend jusqu'à l'infini et la suite associée est causale.

Dans le cas général, la formule d'inversion a pour expression :

$$h(n) = \frac{1}{2\pi j} \int_C H_z(z) / z^n \frac{dz}{z}$$

Où C désigne un contour de Cauchy inclus dans le domaine de convergence. Une façon rapide d'effectuer le calcul est de passer par la méthode des résidus (théorème de Cauchy).

4.2. Filtre défini par une équation récurrente

La définition des filtres linéaires à l'aide d'une équation récurrente permet d'en exhiber très facilement les propriétés. Elle correspond en outre à la définition algorithmique du traitement qu'ils mettent en oeuvre.

On appelle équation récurrente linéaire à coefficients constants une expression de la forme

$$y(n) = a_1 y(n-1) + ... + a_N\ y(n-N) + b_0 x(n) + b_1 x(n-1) + ... + b_M (n-M)$$

Cette forme est analogue, pour le temps discret, aux équations différentielles linéaires à coefficients constants rencontrées dans le cas du temps continu.

Lorsque $a_1 = ... = a_N = 0$, les coefficients $b_0 .....b_M$ apparaissent, d'après l'équation de convolution, comme la réponse impulsionnelle du filtre : on dit que le filtre est à réponse impulsionnelle finie (RIF). Lorsque les a_j ne sont pas tous nuls, on dit que le filtre est récursif Si en plus la fraction rationnelle est irréductible, le filtre est dit à réponse impulsionnelle infinie (RII).

En utilisant la transformée en z et la propriété de retard on déduit de précédente que l'équation récurrente définit un filtre qui a pour fonction de transfert:

$$H_z(z) = \frac{b_0 + b_1 z^{-1} + ... + b_M z^{-M}}{1 + a_1 z^{-1} + ... + a_N z^{-N}}$$

4.3. Introduction aux méthodes de synthèse

4.3.1. Introduction

Le problème de la synthèse, tel que nous le restreignons ici, consiste à calculer la fonction de transfert H(z) d'un filtre à partir de son gabarit en fréquence, ensemble de spécifications portant sur le gain (plus rarement la phase) du filtre (**figure** 5). Par exemple, pour un filtre passe-bas, on donnera

- la plus haute fréquence de la bande passante f_p
- la plus basse fréquence de la bande atténuée f_a
- le taux d'ondulation dans la bande passante
- l'atténuation dans la bande de transition allant de f_p et f_a
- le taux d'ondulation dans la bande atténuée.

Le problème est de synthétiser un filtre dont le gain entre dans le gabarit.

Un premier choix va devoir être fait sur le type du filtre : RIF ou RII. Attention, le fait de ne pouvoir mettre en oeuvre qu'un nombre fini d'opérations ne signifie pas qu'une réalisation RII est impossible.

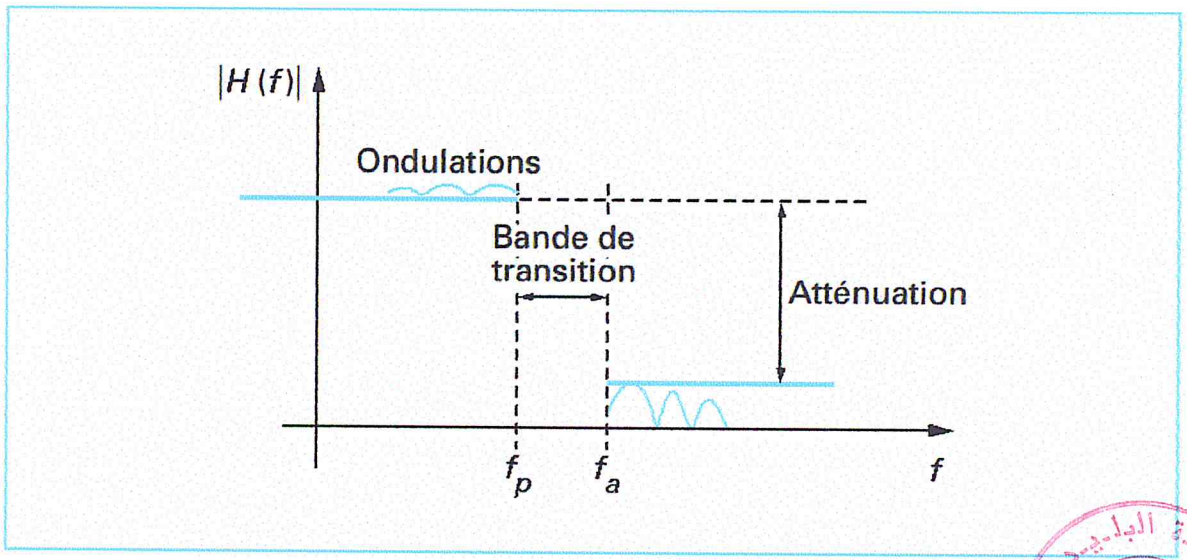


Figure I- 5 : Forme typique du gabarit d'un filtre passe-bas

Ainsi, la mise en oeuvre du programme y (n) = x (n) + a y (n - 1) correspond-elle au filtre de réponse impulsionnelle infinie h (n) =a^n u (n).

4.3.2. Filtres RII

Il existe trois principales techniques de synthèse des filtres RII. La première consiste à partir de filtres analogiques (c'est-à-dire à temps continu) en effectuant une approximation des fréquences continues par des fréquences discrètes. La seconde est basée sur un par:

Calcul direct des coefficients du filtre. La troisième est bâtie sur une procédure itérative d'optimisation lorsque le traitement analytique direct est trop complexe. Nous allons nous contenter ici de présenter la méthode de la transformation bilinéaire.

4.3.3. La méthode de la transformée bilinéaire :

consiste à remplacer l/p $B(z)=\frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}$

Dans l'expression de la fonction de transfert d'un filtre à temps continu (on rappelle que, pour un filtre à temps continu, la fonction de transfert est la transformée de Laplace de sa réponse impulsionnelle).

La transformée bilinéaire est justifiée par le fait que l/p, opérateur d'intégration, peut être approchée en temps discret par B (z). En effet, soit x (t) une fonction à

intégrer, x_n sa valeur à l'instant nT et y_n la valeur de son intégrale entre 0 et nT En utilisant la méthode du trapèze, on a l'équation récurrente $y_n = y_n\ 1 + T(x + x_{n-1}) / 2$ qui a pour fonction de transfert B (z). Si T est suffisamment petit, la distorsion de fréquence introduite lors de cette transformation est négligeable. Dans le cas contraire, il existe des méthodes de compensation.

Pour conclure sur la présentation de cette méthode, indiquons que les trois principaux modèles de filtres analogiques utilisés pour synthétiser un filtre numérique, à partir de la transformée bilinéaire, sont les filtres de Butterworth, de Tchebycheff et de Cauer

4.3.4. Un filtre de Butterworth

a comme fonction de transfert

$$T(p)=\frac{1}{\prod_{k=1}^{n}(p-p_k)} \quad \text{Avec} \quad p_k=\frac{2k-1}{2n}+\frac{1}{2}$$

Et son gain a pour expression $|T(2\pi jf)|^2=\frac{1}{1+(2\pi f)^{2n}}$

Il n'a pas d'ondulation en bande passante ni en bande atténuée.

4.3.5. Un filtre de Tchebycheff

A comme gain

$$|T(2\pi jf)|^2 = \frac{1}{1+\varepsilon^2 T_N^2(f)}$$

Où $T_N(x)=\cos(N \arg\cos x)$ pour $|x|<=1$, ou $T_N(x)=ch(n \arg ch\, x)$ pour $|x|>1$

L'amplitude des ondulations en bande passante est constante et vaut $1/\left(\sqrt{1+\varepsilon^2}\right)$ Ce type de filtre n'a pas d'ondulation en bande atténuée.

4.3.6. Un filtre de Inverse-Tchebycheff :

A comme gain

$$|T(2\pi jf)|^2 = \frac{1}{1+\varepsilon^2 T_N^2(f_a)/T_N^2(f)}$$

Où f désigne la fréquence de début de la bande atténuée. On montre que les N pôles ont pour affixes :

$$p_k = \frac{2\pi f_a A_k}{A_k^2+B_k^2} - j\frac{2\pi f_a B_k}{A_k^2+B_k^2} \text{ avec } A_k = -sh\alpha x \sin\left(\frac{2k-1}{2n}\pi\right) \text{ et}$$

$$B_k = ch\alpha x \cos\left(\frac{2k-1}{2n}\pi\right)$$

Où A_a est l'amplitude imposée en bande atténuée. Les N zéros z_k sont situes sur l'axe imaginaire :

$$z_k = \frac{2\pi jf_n}{\cos\dfrac{2k-1}{2n}\pi}$$

Il y a des ondulations en bande atténuée mais pas d'ondulation en bande passante.

4.3.7. Un filtre de Cauer ou elliptique :

A comme gain

$$|T(\exp(2\pi jf))|^2 = \frac{1}{\varepsilon^2 R_N^2(f,L)}$$

R_n est une fonction rationnelle de Tchebycheff. L caractérise l'atténuation dans la bande de transition. Les filtres de Cauer présentent des ondulations constantes en bande passante. Il y a des ondulations en bande atténuée. Ces filtres sont définis à partir de tables.

Remarque: les méthodes de synthèse ne doivent pas seulement se satisfaire de contraintes de gabarit. Elles doivent aussi prendre en compte l'implantation finale de l'algorithme de filtrage. Ainsi, la présence d'arrondis dans la représentation des coefficients introduit :

- Des phénomènes d'erreurs systématiques que l'on désigne par bruits de calcul;
- Mais aussi de possibles instabilités lorsqu'on a affaire à des filtres récursifs. Ces effets peuvent être particulièrement sensibles lorsque les processeurs utilisés travaillent en virgule fixe.

5. Conclusion

Dans ce chapitre nous avons vu une présentation générale du traitement numérique du signal.

Le but de ce chapitre consiste en deux points : le premier point concerne la transformée de fourier rapide , son rôle est d'extraire les fréquences contenus dans les signaux acquis ceci permettre de trouver les fréquences qui peuvent aider le médecin de savoir l'état du malade.

Le deuxième point est les filtres RII avec les quatre algorithmes que nous avons présentés dans ce chapitre : le filtre de Butterworth , le filtre de Tchebycheff , le filtre de Inverse-Tchebycheff et le filtre de Cauer ou elliptique , ces filtrages sont utilisées pour permettre de filtrer des certaines bandes afin d'aider le médecin de concentrer sur d'autres.

CHAPITRE II : LE PROTOCOLE TCP/IP

1. Introduction

La recherche et le développement en réseaux sont effectués en trois temps. Avant les années 1960, la question fondamentale était : << Comment est-il possible d'acheminer, de façon fiable et efficace, des débits à travers un support de transmission ? >>. Les résultats ont inclus le développement de la théorie du signal.

Aux environ des années 1965, l'intérêt s'est concentré sur la commutation de paquets et la question est devenue : << Comment est-il possible de transmettre des paquets, de façon efficace et fiable, sur un support de communication? >>. Les résultats ont abouti au développement des techniques de communication par paquets et des réseaux locaux.

Des années 1975 jusqu'à aujourd'hui, l'attention s'est portée sur les architectures de réseaux et sur la question : << Comment fournir des services de communication à travers une interconnexion de réseaux >>. Les résultats incluent les techniques d'interconnexion, les modèles de protocoles en couches, les datagrammes, les services de transport en mode connecté et le modèle client/serveur.

Dans ce chapitre nous présentons la programmation d'applications basées sur le modèle client/serveur utilisant les protocoles TCP/IP Internet du D.A.R.P.A. «Défense Advanced Research Projets Agency », car un protocole est un ensemble de règles et de conventions nécessaires pour la communication entre ordinateurs. Comme ces protocoles sont complexes, ils sont découpés en plusieurs couches pour faciliter leurs implémentations, chacune étant construite sur la précédente. Le nombre de couches, leur nom et leur fonction varie selon les réseaux,

Cependant, dans chaque réseau, l'objet de chaque couche est d'offrir certains services aux couches plus hautes. La couche N d'une machine gère la conversation avec la couche N d'une autre machine en utilisant un protocole de niveau N. En réalité, aucune donnée n'est transférée directement de la couche N d'une machine à la couche N de l'autre machine mais chaque couche passe les données et le contrôle à la couche immédiatement inférieure, jusqu'à la plus basse.

En dessous de la couche 1 se trouve le support physique qui véhicule réellement la communication. L'ensemble des couches et protocoles est appelé l'architecture du réseau.

2. Le modèle simplifie du protocole TCP/IP

Les protocoles existants ne suivent pas toujours le modèle OSI, en effet certains protocoles n'utilisent que certaines couches du modèle. Généralement ils se limitent à une représentation en 5 couches. Le plus répandu de ces protocoles est TCP/IP ou UDP/IP.

TCP/IP ou UDP/IP peuvent être utilisés pour communiquer au sein d'un ensemble quelconque de réseaux interconnectés. Les protocoles tels que TCP/IP définissent les règles utilisées pour échanger les unités de données, les détails de leurs structures et indiquent comment gérer le cas d'erreur.

De plus, ils nous permettent de traiter la question des normes de communication indépendamment du matériel réseau de tel ou tel constructeur.

De point de programmes d'application qui utilisent le réseau pour effectuer des de vue de l'utilisateur, une interconnexion TCP/IP se présente comme un ensemble de tâches utiles sans comprendre la technologie TCP/IP, la structure de l'interconnexion sous-jacente, ni même le chemin emprunté par les données pour atteindre leur destination.

Les services d'application les plus populaires et les plus répandus comprennent :

- le courrier électronique,
- le transfert de fichier,
- la connexion à distance.

Le modèle en couche utilisant TCP/IP ou UDP/IP est représenté sur la figure suivante :

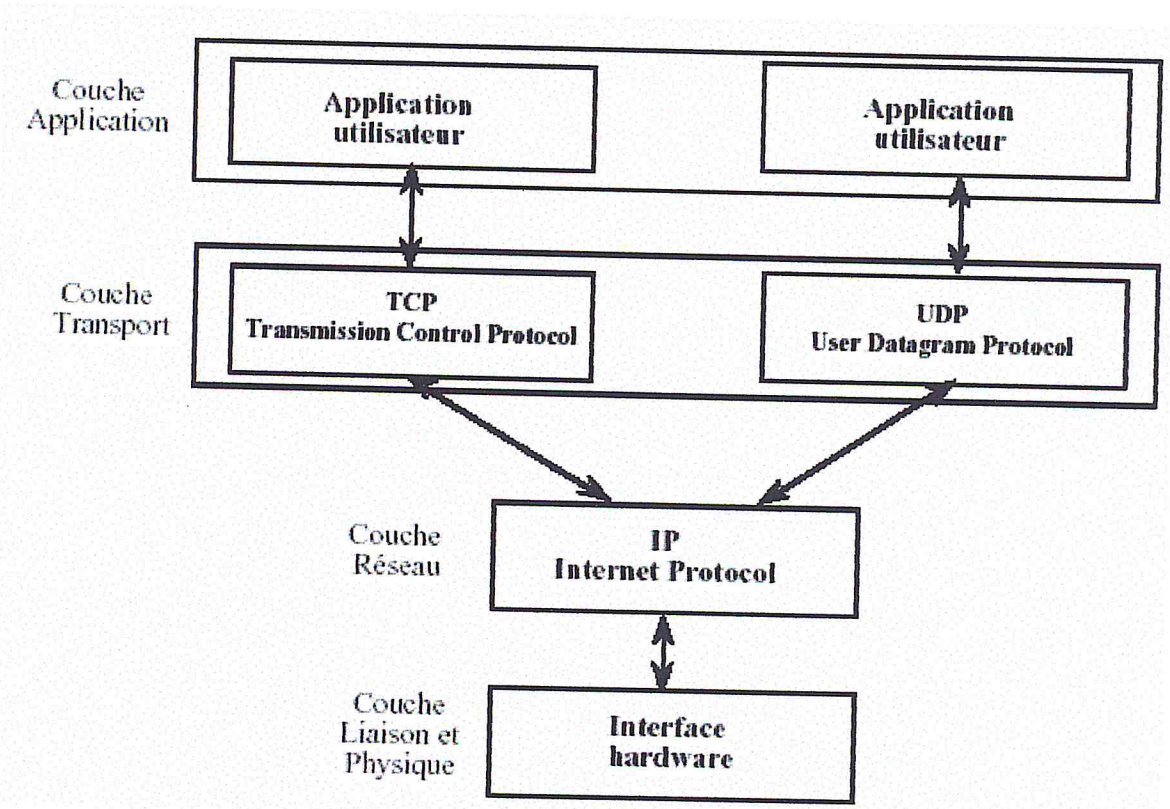


Figure II- 1: réseau en couche utilisant le protocole TCP/IP

L'interface hardware représente les couches 1 et 2 du modèle OSI. Généralement les couches 1 et 2 sont représentées par Ethernet mais on peut aussi avoir"token-ring", X25 .... ; La couche réseau est représentée par IP «Internet Protocol »;

La couche transport est représentée par TCP «Transmission Control Protocol » ou UDP « User Datagram Protocol » ; la dernière couche est la couche application.

2.1. Présentation des unités de données de protocole circulant entre les couches

- **encapsulation des données**

  Les données sont transférées verticalement d'une couche à une autre en y rajoutant un entête, cet entête permet de rajouter des informations identifiant le type de données, le service demandé, le destinataire,

l'adresse source ... : c'est l'encapsulation des données. la figure II-2 illustre le mécanisme d'encapsulation.

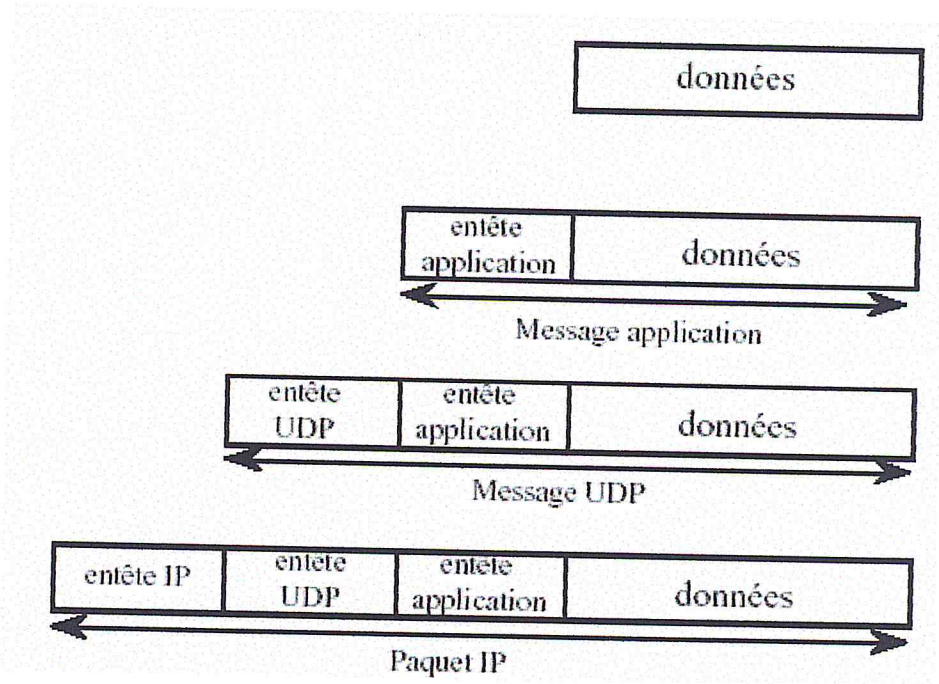


Figure II- 2 : encapsulation de données

TCP/IP est basé sur un modèle qui suppose l'existence de plusieurs réseaux indépendants qui sont interconnectés par des passerelles et que l'utilisateur doit avoir la possibilité d'accéder aux ressources de chacune des machines présentes sur cet ensemble de réseaux interconnectés.

Les datagrammes transiteront souvent par des dizaines de réseaux différents avant d'être délivrées à leur destinataire ; car il existe les datagrammes de type :

le datagramme IP

le paquet UDP

le paquet TCP

2.2. Le protocole TCP

2.2.1. Les caractéristiques du protocole TCP

TCP qui signifie « Transmission Control Protocol », soit en français: Protocole de Contrôle de Transmission, est un des principaux protocoles de la couche transport du modèle TCP/IP ; il permet, au niveau des applications, de gérer les

données en provenance (ou à destination) de la couche inférieure du modèle (c'est-à-dire le protocole IP). Lorsque les données sont fournies au protocole IP, celui-ci les encapsule dans des datagrammes IP, en fixant le champ protocole à 6 (Pour savoir que le protocole en amont est TCP...). TCP est un protocole orienté connexion, c'est-à-dire qu'il permet à deux machines qui communiquent de contrôler l'état de la transmission.

Les caractéristiques principales du protocole TCP sont les suivantes:

- TCP permet de remettre en ordre les datagrammes en provenance du protocole IP
- TCP permet de vérifier le flot de données afin d'éviter une saturation du réseau
- TCP permet de formater les données en segments de longueur variable afin de les "remettre" au protocole IP
- TCP permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne
- TCP permet enfin l'initialisation et la fin d'une communication de manière courtoise

2.2.2. Le but de TCP

Grâce au protocole TCP, les applications peuvent communiquer de façon sûre (grâce au système d'accusés de réception du protocole TCP), indépendamment des couches inférieures. Cela signifie que les routeurs (qui travaillent dans la couche Internet) ont pour seul rôle l'acheminement des données sous forme de datagrammes, sans se préoccuper du contrôle des données, car celui-ci est réalisé par la couche transport (plus particulièrement par le protocole TCP).

Lors d'une communication à travers le protocole TCP, les deux machines doivent établir une connexion. La machine émettrice (celle qui demande la connexion) est appelée client, tandis que la machine réceptrice est appelée serveur. On dit qu'on est alors dans un environnement Client-Serveur

Les machines dans un tel environnement communiquent en full duplex, c'est-à-dire que la communication se fait dans les deux sens.

Pour permettre le bon déroulement de la communication et de tous les contrôles qui l'accompagnent, les données sont encapsulées, c'est-à-dire qu'on

ajoute aux paquets de données un en-tête qui va permettre de synchroniser les transmissions et d'assurer leur réception

Une autre particularité de TCP est de pouvoir réguler le débit des données grâce à sa capacité à émettre des messages de taille variable, ces messages sont appelés segments.

TCP permet d'effectuer une tâche importante: le multiplexage/démultiplexage, c'est-à-dire faire transiter sur une même ligne des données provenant d'applications diverses ou en d'autres mots mettre en série des informations arrivant en parallèle ; ces opérations sont réalisées grâce au concept de ports (ou sockets), c'est-à-dire un numéro associé à un type d'application, qui, combiné à une adresse IP, permet de déterminer de façon unique une application qui tourne sur une machine donnée.

2.2.3. Le format des données sous TCP

Le tableau suivant illustre l'entête de paquet TCP

<table>
<tr><th colspan="16">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</th><th colspan="16">16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</th></tr>
<tr><td colspan="16">Port Source</td><td colspan="16">Port destination</td></tr>
<tr><td colspan="32">Numéro d'ordre</td></tr>
<tr><td colspan="32">Numéro d'accusé de réception</td></tr>
<tr><td colspan="4">Décalage données</td><td colspan="3">réservée</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Fenêtre</td></tr>
<tr><td colspan="16">Somme de contrôle</td><td colspan="16">Pointeur d'urgence</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Remplissage</td></tr>
<tr><td colspan="32">Données</td></tr>
</table>

Figure II- 3 : l'entête de paquet TCP

Signification des différents champs:

- **Port Source** (16 bits): Port relatif à l'application en cours sur la machine source
- **Port Destination** (16 bits): Port relatif à l'application en cours sur la machine de destination
- **Numéro d'ordre** (16 bits): Lorsque le drapeau SYN est à 0, le numéro d'ordre est celui du premier mot du segment en cours Lorsque SYN est à 1, le numéro de séquence est le numéro de séquence initial utilisé pour synchroniser les numéros de séquence (ISN)
- **Numéro d'accusé de réception** (32 bits): Dernier segment reçu par le récepteur
- **Décalage des données** (4 bit): il permet de repérer le début des données dans le paquet. Le décalage est ici essentiel car le champ d'options est de taille variable
- **Réservé** (6 bits): Champ inutilisé actuellement mais prévu pour l'avenir
- **Drapeaux (flags)** (6x1 bit): Les drapeaux représentent des informations supplémentaires:
  - **URG**: si ce drapeau est à 1 le paquet doit être traité de façon urgente
  - **ACK**: si ce drapeau est à 1 le paquet est un accusé de réception

- **PSH** (PUSH): si ce drapeau est à 1, le paquet fonctionne suivant la méthode PUSH
- **RST**: si ce drapeau est à 1, la connexion est réinitialisée
- **SYN**: si ce drapeau est à 1, les numéros d'ordre sont synchronisés
- **FIN**: si ce drapeau est à 1 la connexion s'interrompt

- **Fenêtre** (16 bits): Champ permettant de connaître le nombre d'octets que le récepteur souhaite recevoir sans accusé de réception
- **Somme de contrôle** (Checksum ou CRC): La somme de contrôle est réalisée en faisant la somme des champ de données de l'en-tête, afin de pouvoir vérifier l'intégrité de l'en-tête
- **Pointeur d'urgence** (16 bits): Indique le numéro d'ordre à partir duquel l'information devient urgente
- **Options** (Taille variable): Des options diverses
- **Remplissage**: On remplit l'espace restant après les options avec des zéros pour avoir une longueur de 32 bits

2.2.4. Terminologie

Avant de préciser en profondeur le fonctionnement de TCP, nous devons tout d'abord prendre quelques conventions sur la terminologie. La maintenance d'une connexion TCP nécessite la mémorisation d'un certain nombre de variables ; nous avons prévu que ces données soient enregistrées dans une structure nommée "Transmission Control Block" ou (TCB) ; Parmi les données enregistrées dans ce TCB, on trouvera les sockets local et distant, les informations de sécurité et de priorité de la connexion, les pointeurs vers les tampons de réception et d'émission, les pointeurs vers la pile de retransmission et vers le segment courant. On y trouvera de plus quelques données relatives à la gestion des numéro de séquence :

Variables de séquence d'émission

- SND.UNA - "unacknowledge" - envoi sans accusé de réception
- SND.NXT - "next" - envoi du suivant
- SND.WND - "window" - envoi de la fenêtre
- SND.UP - "urgent pointer" - pointeur de données urgentes
- SND.WL1 - "window last 1" - dernier numéro de séquence de segment envoyé
- SND.WL2 - "window last 2" - dernier accusé de réception
- ISS - "initial send sequence" - premier numéro de séquence du message

Variables de séquence de réception

- RCV.NXT - "next" - réception du suivant
- RCV.WND - "window" - réception de fenêtre
- RCV.UP - "urgent pointer" - pointeur de données urgentes
- IRS - "initial receive sequence" - premier numéro de séquence en réception

Variables du segment courant

- SEG.SEQ - "sequence" - numéro de séquence du segment courant
- SEG.ACK - "acknowledge" - numéro de séquence inscrit dans l'accusé de réception
- SEG.LEN - "length" - longueur du segment courant
- SEG.WND - "window" - fenêtre inscrite dans le segment courant
- SEG.UP - "urgent pointer" - pointeur de données urgentes du segment courant
- SEG.PRC - "precedence" - valeur de priorité courante

Une connexion connaît plusieurs états durant sa durée de vie. Les états définis sont: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, et l'état fictif CLOSED. CLOSED est dit fictif car il correspond à une situation où la connexion elle-même n'existe plus (son TCB non plus). Nous donnons ci-dessous un descriptif rapide des états cités:

LISTEN - La connexion reste en attente d'une requête de connexion externe par un TCP distant. Cet état est atteint après une demande de connexion passive.

SYN-SENT - La connexion se met en attente d'une requête de connexion, après avoir envoyé elle-même une requête à un destinataire.

SYN-RECEIVED - Les deux requêtes de connexion se sont croisées. La connexion attend confirmation de son établissement.

ESTABLISHED - La connexion a été confirmé de part et d'autre et les données peuvent transiter sur la voie de communication. C'est l'état stable actif de la connexion.

FIN-WAIT-1 - Sur requête de déconnexion émise par l'application, la connexion demande la confirmation d'une requête de déconnexion qu'elle a elle-même émise vers le distant.

FIN-WAIT-2 - La connexion se met en attente d'une requête de déconnexion par le distant, une fois reçue la confirmation de sa propre requête.

CLOSE-WAIT - La connexion se met en attente d'une requête de déconnexion émise par l'application.

CLOSING - La connexion attend la confirmation de sa requête de déconnexion par le TCP distant, lequel avait auparavant émis sa propre requête de déconnexion.

LAST-ACK - La connexion attend la confirmation de sa requête de déconnexion, émise suite à une requête similaire à l'initiative du distant.

TIME-WAIT - Un temps de latence avant fermeture complète du canal, pour s'assurer que toutes les confirmations ont bien été reçues.

CLOSED - La connexion n'existe plus. C'est un pseudo-état.

Le changement d'état d'une connexion TCP intervient sur une réception d'un certain nombre de signaux, appelée "événement", les événements peuvent être des commandes émises par l'application, OPEN, SEND, RECEIVE, CLOSE, ABORT, et STATUS; la réception d'un flag dans un segment en provenance du distant SYN, ACK, RST et FIN; la fin d'une temporisation.

2.2.5. Etablissement d'une connexion

Etant donné que ce processus de communication, qui se fait grâce à une émission de données et d'un accusé de réception, est basé sur un numéro d'ordre (appelé généralement numéro de séquence), il faut que les machines émettrices et réceptrices (client et serveur) doivent connaître le numéro de séquence initial de l'autre machine.

L'établissement de la connexion entre deux applications se fait souvent selon le schème suivant:

- Les ports TCP doivent être ouverts
- L'application sur le serveur est passive, c'est-à-dire que l'application est à l'écoute, en attente d'une connexion
- L'application sur le client fait une requête de connexion sur le serveur dont l'application est en ouverture passive. L'application du client est dite "en ouverture passive"

Les deux machines doivent donc synchroniser leurs séquences grâce à un mécanisme communément appelé "three ways handshake "(poignée de main en trois temps), que l'on retrouve aussi lors de la clôture de session.

Ce dialogue permet d'initier la communication, il se déroule en trois temps, comme sa dénomination l'indique:

- Dans un premier temps la machine émettrice (le client) transmet un segment dont le drapeau SYN est à 1 (pour signaler qu'il s'agit d'un segment de synchronisation), avec un numéro d'ordre N, que l'on appelle numéro d'ordre initial du client
- Dans un second temps la machine réceptrice (le serveur) reçoit le segment initial provenant du client, puis lui envoie un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1 et le drapeau SYN est à 1 (car il s'agit là encore d'une synchronisation). Ce segment contient le numéro d'ordre de cette machine (du serveur) qui est le numéro d'ordre initial du client. Le champ le plus important de ce segment est le champ accusé de réception qui contient le numéro d'ordre initial du client, incrémenté de 1

Enfin, le client transmet au serveur un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1, dont le drapeau SYN est à zéro (il ne s'agit plus d'un segment de synchronisation). Son numéro d'ordre est incrémenté et le numéro d'accusé de réception représente le numéro de séquence initial du serveur incrémenté de 1

2.2.6. Fin d'une connexion

Le client peut demander à mettre fin à une connexion au même titre que le serveur.
la fin de la connexion se fait de la manière suivante:

- Une des machines envoie un segment avec le drapeau FIN à 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants
- Après réception de ce segment, l'autre machine envoie un accusé de réception avec le drapeau FIN à 1 et continue d'expédier les segments en cours. Suite à cela la machine informe l'application qu'un segment FIN a été reçu, puis envoie un segment FIN à l'autre machine, ce qui clôture la connexion...

2.2.7. Les Commandes applicatrices vers TCP

Les commandes de base décrites ici sont essentielles pour qu'un TCP puisse supporter une communication inter-processus. Chaque implémentation concrète pourra adopter sa propre syntaxe, et pourra y adjoindre des combinaisons ou fonctions partielles issues de ces fonctions de base., par exemple, certains utilisateurs souhaiteront qu'une premier appel à la fonction SEND puisse automatiquement ouvrir la connexion (OPEN).

Dans son rôle d'intermédiaire de communication, TCP doit non seulement accepter des commandes, mais doit retourner un certain nombre d'informations à l'application, soit en réponse à une commande, soit de façon unilatérale ;ces dernières consistent en:

Une information générale sur la connexion (ex., interruptions, fermeture distante, connexion sur tel socket passif en attente).

Convention : Dans la formulation des commandes qui suivent, nous adoptons la convention de notation suivante pour les paramètres :

- Les paramètres de type bit (sémaphores) sont notés en MAJUSCULE.
- Les paramètres d'un autre type (adresse, entier, long, etc.)sont notés en minuscule

OPEN

OPEN (port_local, socket_distant, ACTIF/PASSIF [, tempo] [, priorité] [, sécurité/compartiment] [, options]) -> nom_local

Supposons ici que le TCP local connaît l'identité du processus qu'il sert, et vérifiera que ce processus est bien autorisé à ouvrir une connexion sur le socket spécifié. Suivant l'implémentation de TCP, les identificateurs TCP et réseau d'adresse source seront soit fournis par TCP soit par le protocole de routage (ex. IP). Ces considérations découlent d'une réflexion sur la sécurité, dont le but est d'interdire à un TCP de se faire passer pour un autre. De même, aucun processus ne peut se faire passer pour un autre sans la complicité des couches inférieures

SEND

SEND (nom_local, adresse_tampon, compteur, PUSH, URGENT [, tempo])

Cet appel permet l'envoi des données contenues dans le tampon de taille compte débutant à l'adresse_tampon sur la connexion définie par nom_local. Dans le cas général, si la connexion n'a pas été ouverte auparavant, cette commande génère une erreur. Certaines implémentations permettront l'usage direct de la commande SEND; celles-ci demanderont les paramètres supplémentaires utiles à l'établissement préalable de la connexion ; Si l'application demanderesse n'est pas autorisée à ouvrir la connexion demandée, une erreur sera retournée.

RECEIVE

RECEIVE (nom_local, adresse_tampon, compte) -> compte, URGENCE, PUSH

Cette commande alloue un tampon de réception à la connexion spécifiée. Si la connexion n'a pas été ouverte au préalable par une commande OPEN, ou l'application n'a pas l'autorisation d'utiliser une telle connexion, une erreur sera renvoyée.

CLOSE

CLOSE (nom_local)

Cette commande demande la fermeture de la connexion spécifiée. Si la connexion n'existe pas, ou l'application n'a pas l'autorisation nécessaire pour utiliser cette connexion, une erreur est renvoyée. La fermeture de connexions est prévue pour se dérouler de façon "propre" dans la mesure ou toutes les émissions en instance seront menées à leur terme (et retransmises si nécessaire), selon les impératifs du contrôle de flux. Ainsi, il est légitime de lancer plusieurs commandes SEND successives suivies d'une commande CLOSE, et de considérer que le message a été correctement et complètement transmis. Il est aussi clair que la communication doit être toujours "écoutée", afin de récupérer tout ce que le distant a à transmettre. De ce fait, la commande CLOSE signifie "je n'ai plus rien à transmettre", et non pas "je ne veux plus rien recevoir". Il peut arriver (notamment si le protocole au niveau application n'est pas suffisamment complet) que le côté fermant la connexion ne puisse se débarrasser de toutes ses données avant intervention de la temporisation. Dans ce cas, la commande CLOSE revient à une commande ABORT, et TCP finit le travail de fermeture.

STATUS

STATUS (nom_local) -> état

Cette commande dépend de l'implémentation et peut être "oubliée" sans conséquence majeure. L'information renvoyée est essentiellement issue des données du TCB associé à la connexion.

ABORT
ABORT (nom_local)

Cette commande abandonne toutes les commandes SEND et RECEIVE en cours. Le TCB est effacé, et un message d'abandon est envoyé au TCP distant. Suivant l'implémentation, l'application recevra des indications en retour sur chacune des émissions et réceptions abandonnées, ou simplement un acquittement global de l'abandon.

3. LES SOCKETS

3.1. Le modèle client serveur

L'architecture client/serveur est devenue la méthode incontournable pour la communication point à point au niveau applicatif, quand le protocole utilisé est de la famille TCP/IP. Les applications peuvent être classées en deux catégories:

- Les **clients**: applications qui prennent l'initiative du lancement de la communication c'est à dire demande l'ouverture de connexion, l'envoi d'une requête, l'attente de la réponse à la requête, reprise de l'exécution du programme
- Les **serveurs**: applications qui attendent la communication, c'est à dire d'ouverture de connexion, la réception d'une requête et l'envoi d'une réponse.

3.1.1. Mode connecté

Deux approches de programmation de serveurs se présentent : l'approche itérative et l'approche parallèle. Rappelons que dans l'approche itérative, le serveur traite lui même la requête, ferme le nouveau socket puis invoque de nouveau accept pour obtenir la demande suivante ; l'enchaînement dans le temps des appels système pour un serveur itératif se résume par la figure suivante:

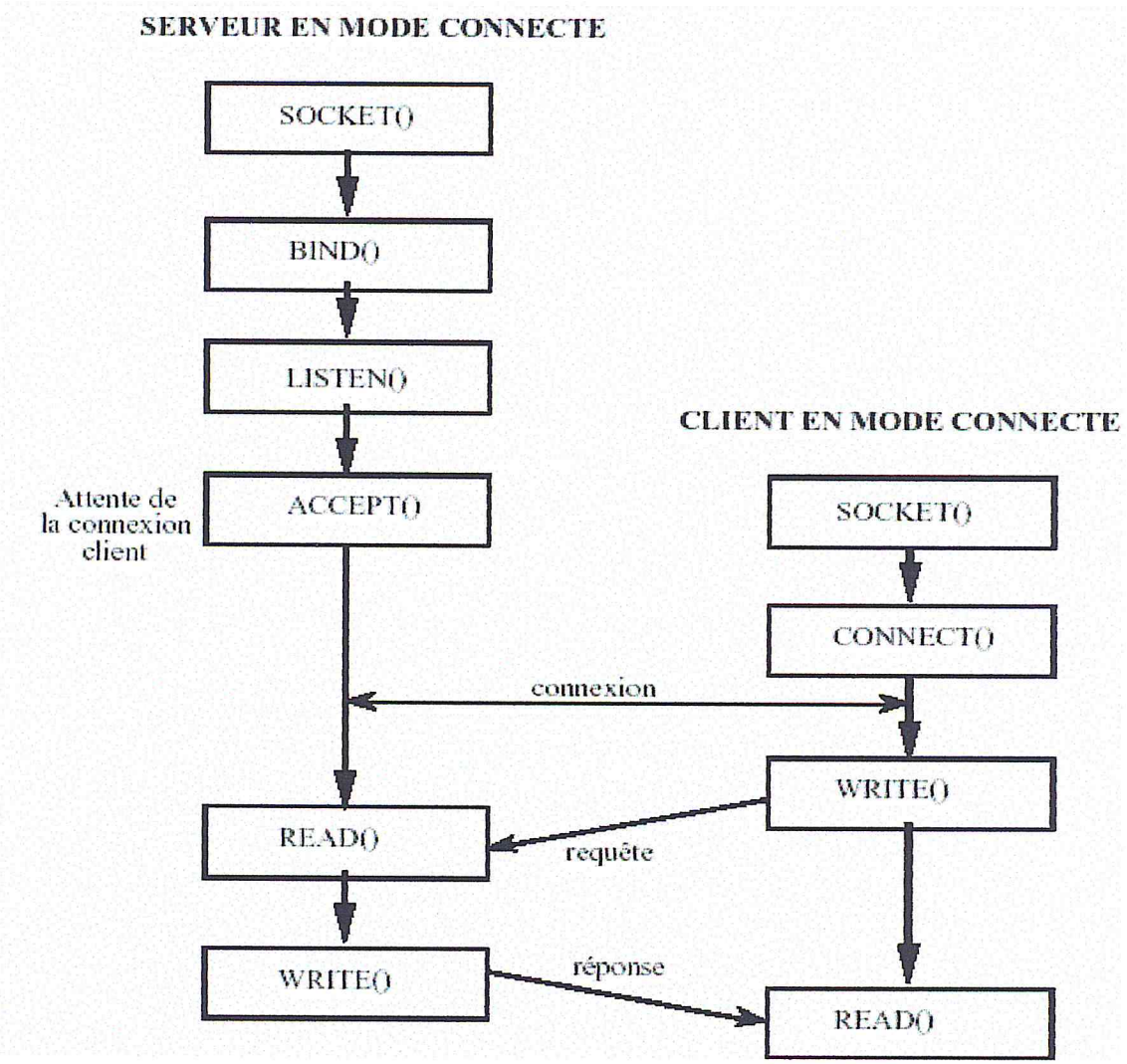


Figure II- 4: Etablissement d'une connexion TCP en mode connecté, approche itérative

Dans l'approche parallèle, lorsque l'appel système accept se termine le serveur crée un serveur fils chargé de traiter la demande (appel de fork et exec). Lorsque le fils a terminé il ferme le socket et meurt. Le serveur maître ferme quand a lui la copie du nouveau socket après avoir exécuté le fork. Il appel ensuite de nouveau accept pour obtenir la demande suivante.

L'enchaînement des appels systèmes pour un serveur parallèle se résume par la figure suivante:

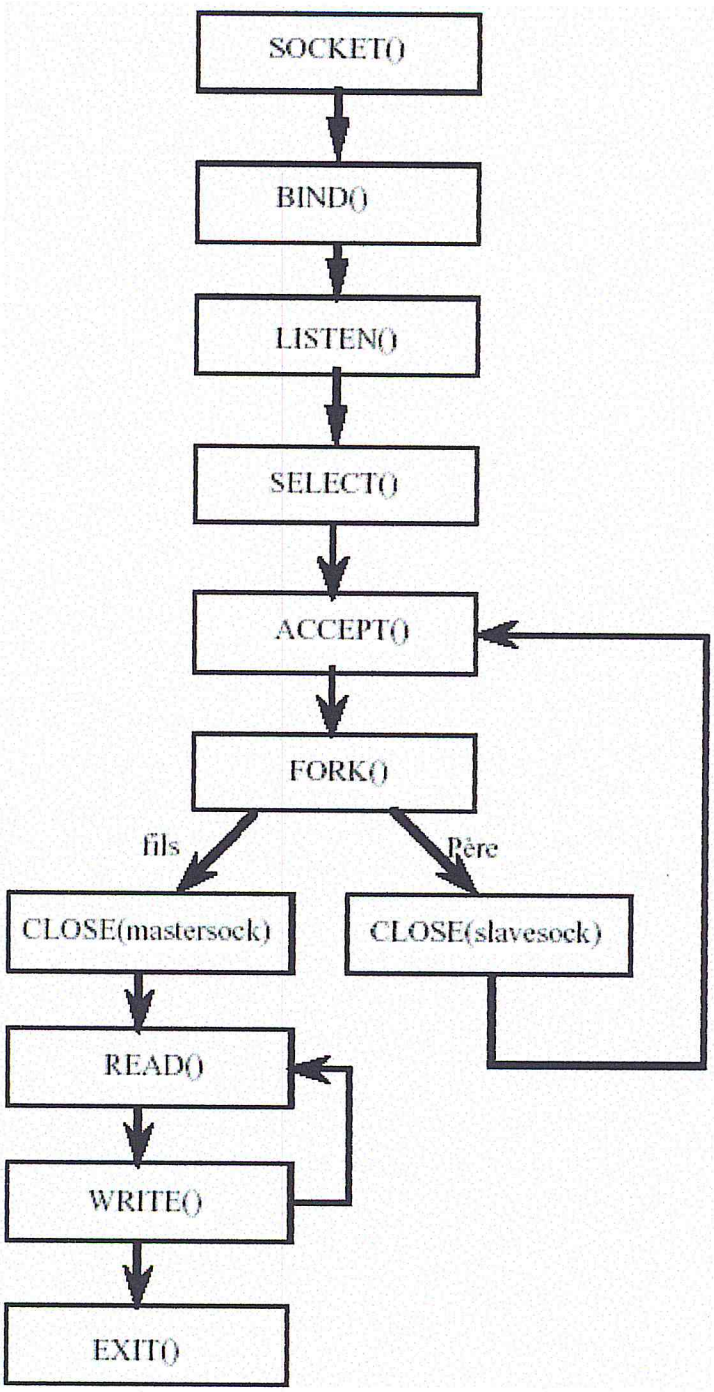


Figure II- 5 : Etablissement d'une connexion TCP en mode connecté, approche parallèle

3.1.2. Mode non connecté

Dans le cas d'un protocole sans connexion, les appels système sont différents. Le client n'établit pas de connexion avec le serveur. Le client envoi un datagramme au serveur en utilisant l'appel système sendto (). Symétriquement le serveur n'accepte pas de connexion d'un client, mais attend un datagramme d'un client avec l'appel système Recvfrom ().

Ce dernier revoie l'adresse réseau du client, avec le datagramme, pour que le serveur puisse répondre à la bonne adresse. L'enchaînement dans le temps des appels systèmes pour une communication en mode non connecté et pour un serveur itératif se résume par la figure suivante:

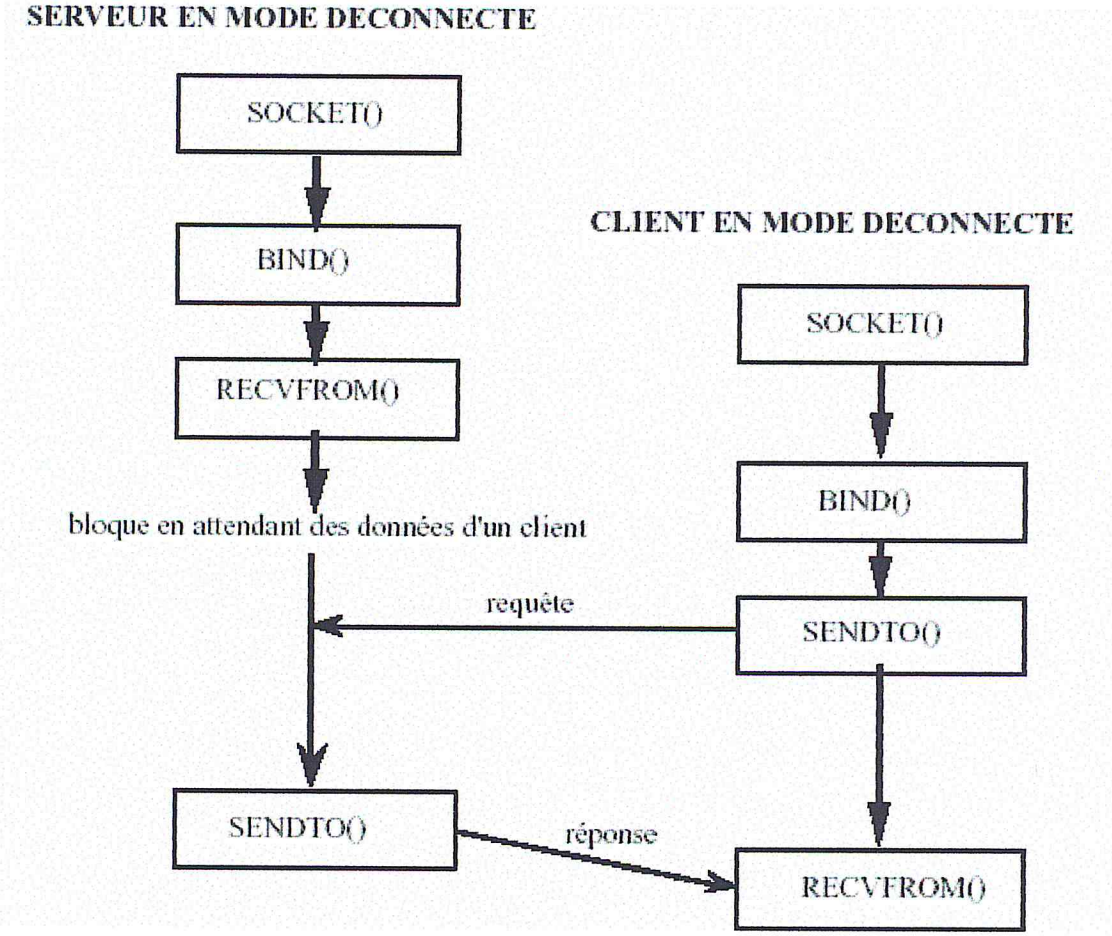


Figure II- 6 : Etablissement d'une connexion TCP en mode connecté, approche itérative

L'enchaînement dans le temps des appels systèmes pour une communication en mode non connecté et pour un serveur parallèle se résume par la figure suivante:

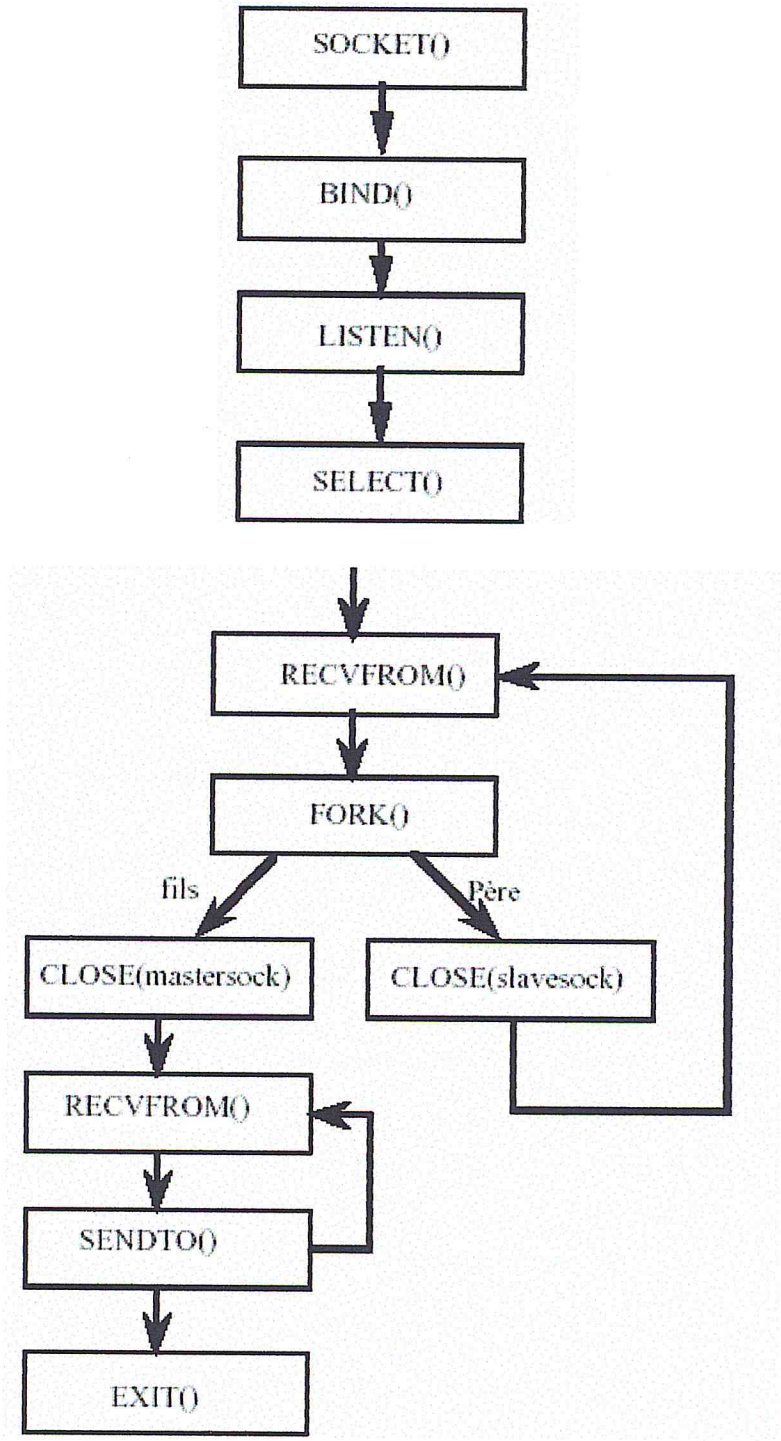


Figure II- 7 : Etablissement d'une connexion TCP en mode connecté, approche parallèle

4. Conclusion

Dans ce chapitre nous avons présentés le protocole TCP/IP. Ce protocole définit des règles utilisées pour échanger les unités de données et les détails de leurs structures.

Le protocole TCP/IP est utilisé pour communiquer au sein d'un ensemble de réseaux il permet un bon déroulement de la communication et de tous les contrôles qui l'accompagnent, grâce à l'encapsulation des données, c'est l'ajout aux paquets de données un en-tête permettant la synchronisation des transmissions et d'assurer leur réception.

Nous avons utilisé ce protocole dans notre travail pour communiquer et partager les signaux EEG dans un réseau.

CHAPITRE III : LA DEMARCHE DE DEVELOPPEMENT

1. Introduction

Dans notre projet nous développons un système "moniteur de EEG " pour effectuer la visualisation des fréquences contenus dans un signal EEG capturé par une carte d'acquisition, la conception de ce système est un ensemble d'étapes par lesquelles passe un logiciel.

Pour augmenter la chance de réussite des logiciels, des méthodes de développement de logiciel ont été définies, ces méthodes permettent de mieux organiser la conception, de réduire la complexité des applications, d'avoir une meilleure compréhension du problème et du fonctionnement du futur système à développer,

➢ Une méthode de développement de logiciel est définie pour représenter le processus, ce processus décrit les étapes à suivre lors du développement du système.

Le processus de développement que nous avons utilisé est le cycle de vie en cascade, ce modèle décrit le cycle de vie d'un logiciel par une suite de phases qui s'enchaînent, l'une après l'autre, depuis l'analyse des besoins jusqu'au test, ces phases sont présentées dans la figure suivante :

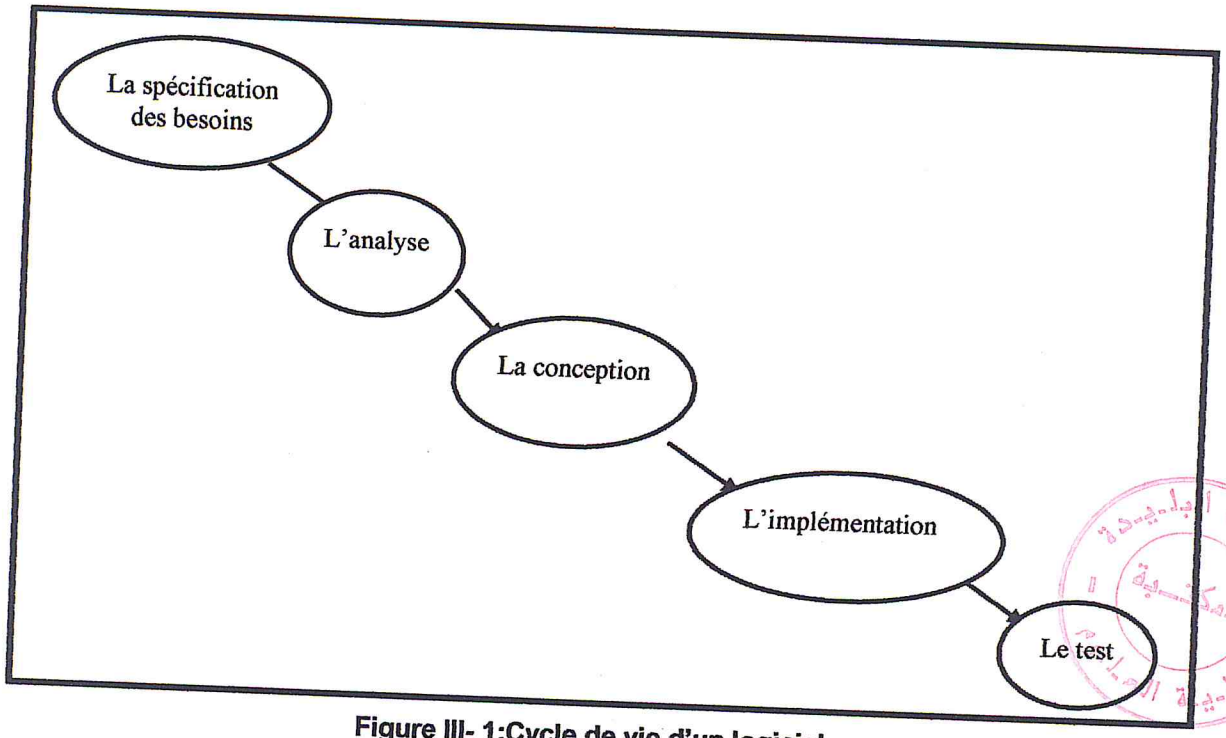


Figure III- 1:Cycle de vie d'un logiciel

La notation unifiée UML se base sur les méthodes d'analyse et de conception suivantes :

- OMT (Object Modeling Technique) de James RUNBAUGH,
- OOD (Object Oriented Design) de Grady BOUCH, et
- OOSE (Objected Oriented Software Engineering) de Ivar JACOBSON.

De ce fait, elle tire les avantages de chacune de ces méthodes, elle permet de couvrir tous le cycle de vie d'un logiciel depuis l'analyse du besoin jusqu'au test. Cette notation est d'une très grande richesse. Elle permet de couvrir à peu près toutes les phases du développement :

– les diagrammes des cas d'utilisation pour définir les besoins des utilisateurs et les fonctions du futur système,

– un ensemble de diagrammes couvrant les parties statiques et dynamiques qui permettent une spécification complète du système,

– la conception, jusqu'à un niveau très proche du code en langage objet de l'implémentation.

– les suites de tests permettant de s'assurer qu'une implémentation candidate est effectivement conforme à la spécification élaborée en phase amont, sous forme de diagrammes de séquences.

Dans ce chapitre nous présentons la démarche de développement du système. Pour réaliser ceci nous procédons comme suit:

Dans une première étape nous commençons par limiter les fonctionnalités du système à développer. A travers les diagrammes des cas d'utilisation offert par UML, passant ensuite à l'étape d'analyse, ou nous présentons les sous système de notre système et les composants de chaque sous système. Puis nous passons à la phase la plus importante du processus de développement d'un logiciel, c'est la phase de conception. Enfin nous avons procède a l'implémentation de l'application en utilisant le Labview et le Visual C#. Net. Et pour mettre en valeur notre travail, nous abordons le test par simulation.

Remarque : les diagrammes d'UML présentées dans ce mémoire sont édités et validés par le logiciel Microsoft Visio 2003, et par conséquence la notation UML de ces diagrammes suit la notation de Microsoft.

2. La spécification des besoins

La spécification des besoins est une étape essentielle au début du processus de développement. Son but est de décrire les fonctionnalités du système à développer et les besoins des utilisateurs.

La finalité de cette étape est la description générale des fonctionnalités du système. Par la réponse à ces questions : "quelles sont les fonctions du système ?", "quels sont les utilisateurs du système?", "et qu'attendent-ils du système ?". Cette étape étudiée le comportement du système exprimé sous la forme des cas d'utilisation, le contexte du système, les acteurs et les scénarios.

2.1. Les Cas d'utilisation

L'analyse détermine le quoi faire, c'est à dire les besoins de l'utilisateur. L'expérience montre que la technique des cas d'utilisation (use cases) se prête bien à la détermination des besoins d'utilisateurs [Muller, 97]. Les cas d'utilisation décrivent sous la forme d'actions et de réactions le comportement du système du point de vue de l'utilisateur.

L'étude des cas d'utilisation débute par la détermination des acteurs (catégories d'utilisateurs) du système.

2.1.1. Définition des acteurs

Notre système est destiné à des utilisateurs, c'est pour eux que le système est fabriqué. Ils sont la réponse à la question : pour qui est fabriqué le système ?

Notre système "Moniteur de EEG" est destiné à un utilisateur qui souhaite effectuer la visualisation des fréquences contenus dans un signal EEG capturé par une carte d'acquisition, c'est un médecin en générale, car notre logiciel est utilisé par un médecin pour savoir des certains maladie suivant les résultats obtenus par ce logiciel.

2.1.2. Descriptions textuelles des cas d'utilisation

Les différents cas d'utilisation recensés pour le système sont :

- La visualisation : ce cas permet à un médecin de visualiser les signaux acquis, après qu'il a choisit le type d'affichage et le mode de graphe. le cas « visualisation » inclut le cas « traitement des données » ceci est exprimé par la relation <<include>>. Et utilise le cas « acquisition » qui est exprimé par la relation <<uses>>.
- La cas « traitement de données» utilise le cas « séparation », c-a-d pour traiter les signaux qui sont regroupés, il faut les sépares, ceci est représenté par la relation <<uses>> dans le diagramme des cas d'utilisations.
- Le cas « L'acquisition des données », est une séquence d'opérations pour lire un échantillon depuis une source, et prendre une durée (dans notre logiciel fixé par une seconde).
- Le cas d'utilisation « séparation » divise le signal acquis en 16 signaux, chacun entre eux représente un canal d'acquisition.

- L'enregistrement : ce cas permet l'enregistrement des fichiers pour permette de les sauvegarder et de les réouvrir en cas de besoins, pour réaliser l'enregistrement, nous devons ouvrir le fichier puis écrire des blocs dans ce fichier, en utilisant des conversions en binaire des échantillons.
- La publication : ce cas permet de transmettre les échantillons acquis vers une ou plusieurs applications à travers des connexion réseaux, qui sont crées suivant le protocole TCP/IP.
- Pour enregistrer ou publier des données il faut d'abord les acquérir ,qui est traduit par la relation d'utilisation<<uses>> entre les deux cas « l'enregistrement » et « publication », et le cas « acquisition ».

- Consultation rapide : ce cas permet à l'utilisateur de visualiser un fichier enregistré directement sans la simulation de temps d'acquisition, le cas ou on visualise un fichier enregistré et on simule en même temps le temps d'acquisition, est inclut dans le cas d'utilisation acquisition (cas d'utilisation acquisition depuis un fichier), le cas « Consultation rapide » utilise le cas « séparation », ceci est représenté par la relation <<uses>> dans le diagramme des cas d'utilisations.

2.2. Diagrammes des cas d'utilisation et diagrammes d'activités

Les différentes fonctionnalités offertes par notre système forment ainsi un ensemble de *cas d'utilisation* (*"Use Case"*), exprimés textuellement dans la section précédente, afin de les formaliser UML propose au travers de diagramme cas d'utilisation de répertorier et de structurer les fonctionnalités que le futur système offrira à ses utilisateurs. Mais les diagrammes des cas d'utilisation ne permettent pas de savoir le déroulement d'un cas d'utilisation, ce ci permet par les diagrammes d'activités.

Un diagramme d'activité permet de décrire le déroulement d'un cas d'utilisations particulier. Il est possible de décrire les acteurs responsables de chaque activités par l'utilisation des «couloirs d'activités» qui permettent de répartir graphiquement les différentes activités entre les acteurs opérationnels [Muller, 01].

Dans la partie de spécification des besoins, nous avons préféré de présenter pour chaque cas d'utilisation le diagramme d'activité puis le diagramme des cas d'utilisation pour permettre une bonne compréhension au lecteur.

2.2.1. Le diagramme d'activité pour les fonctionnalités globale du système

Après l'acquisition d'une échantillon, le système doit réaliser les fonctionnalités du système suivant les demandes de l'utilisateur,

- s'il veut traiter des données, le système doit les traiter,
- le système doit visualiser soit l'échantillon acquis, soit l'échantillon acquis et traité,
- s'il veut enregistrer des données, le système doit l'enregistrer,
- s'il veut partager des données, le système publier ces données,
- une demande de fermeture, si l'utilisateur veut quitter.

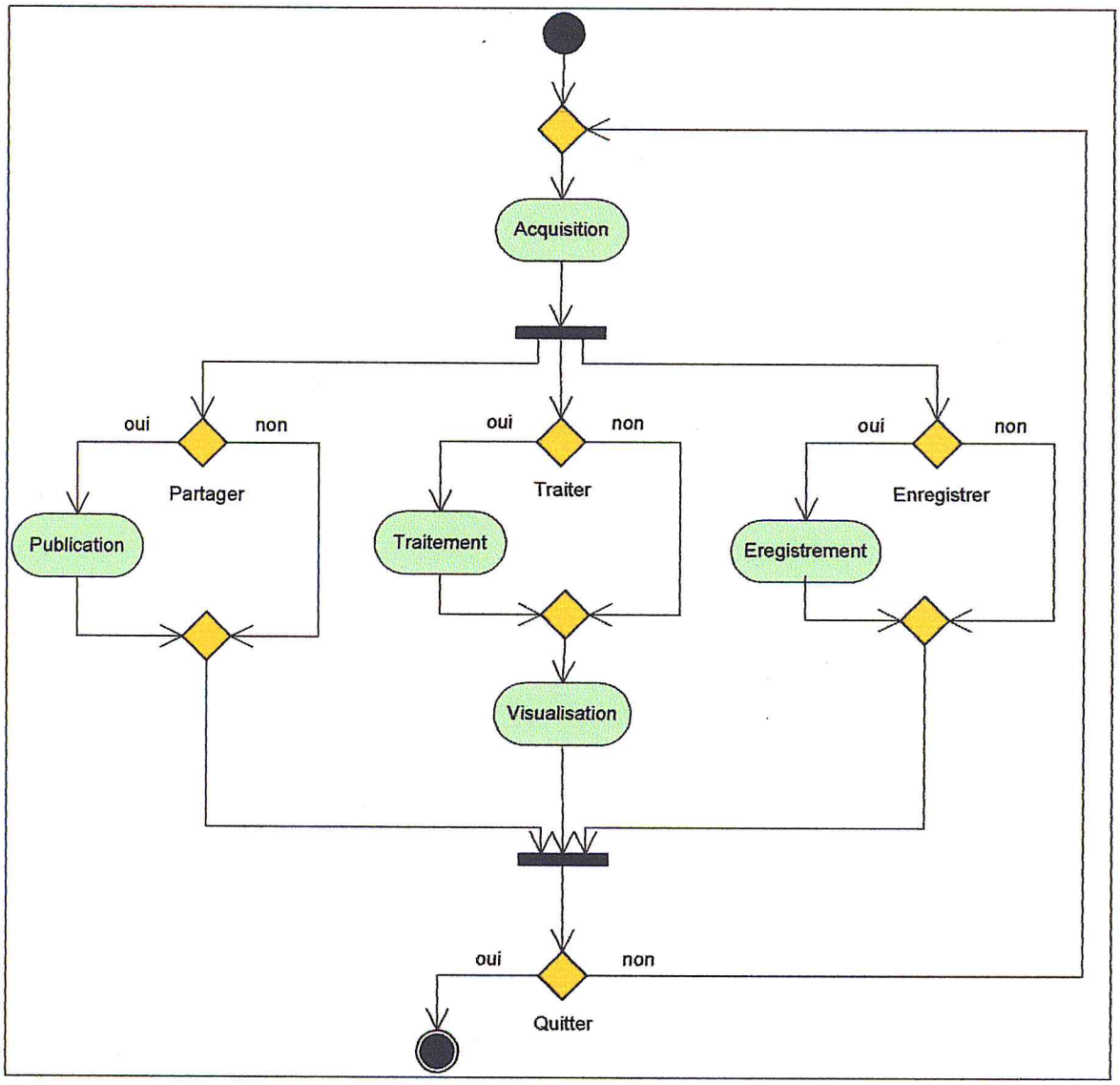


Figure III- 2 : le diagramme d'activité pour les fonctionnalités globales du système.

2.2.2. Le diagramme des cas d'utilisation pour les fonctionnalités globale du système

Dans ce premier diagramme des cas d'utilisation, nous allons citer les fonctionnalités globales du système, il présente le diagramme général des cas d'utilisations, dans la section qui se suit nous présentons pour chaque cas d'utilisation, un diagramme détaillé.

La figure suivante illustre le diagramme global des cas d'utilisation.

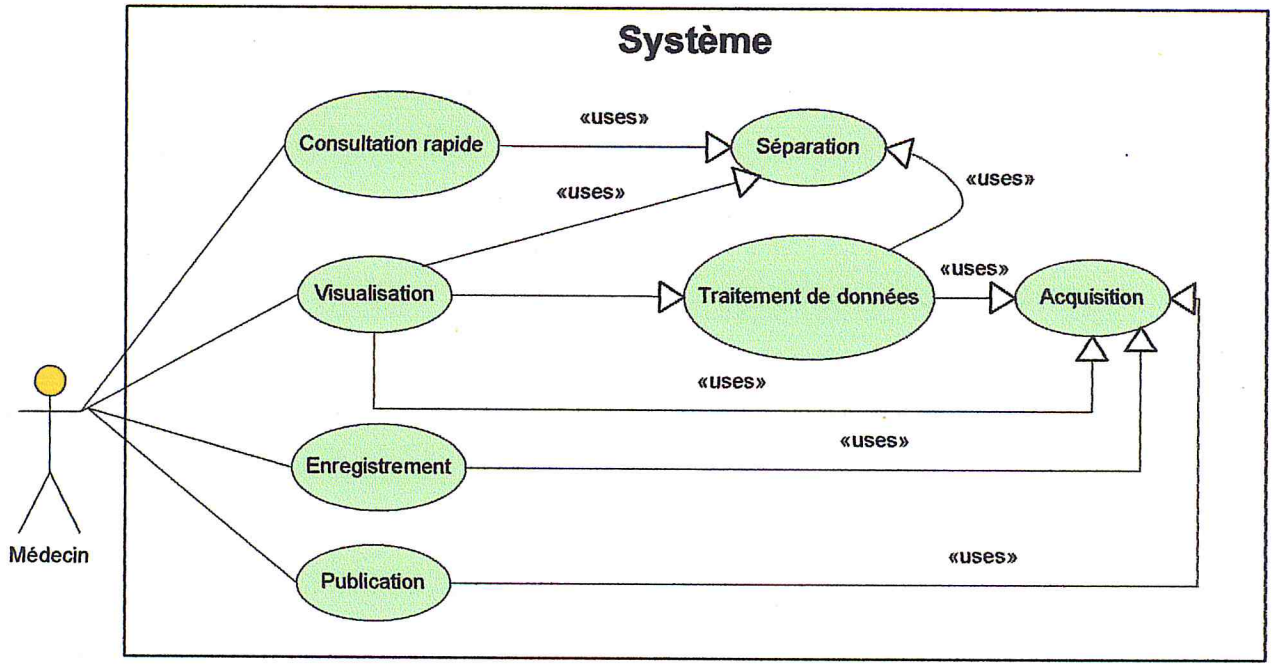


Figure III- 3 : le diagramme global des cas d'utilisations.

Dans cette section nous présentons les diagrammes des cas d'utilisations et d'activités détaillées :

2.2.3. Le diagramme d'activité pour le cas « Acquisition »

- L'utilisateur choisit le mode de lecture, il y a trois modes : lecture depuis un fichier, ou lecture à temps réel ou lecture à distance,

➢ Le cas lecture depuis fichier : il faut récupérer le chemin du fichier à lire puis de verrouiller le fichier s'il existe.

➢ Le cas lecture à distance : il faut configurer le serveur réseau, cette configuration nécessite un certains paramètres comme l'adresse IP de serveur réseau, et le numéro de port selon le protocole TCP/IP, si la connexion n'existe pas il faut alors l'établir puis le la valider.

➢ Le cas lecture à temps réel : ce cas nécessite aussi un certain nombres des paramètres pour configurer la carte d'acquisition : il s'agit des paramètres suivants : la polarité (bipolaire ou unipolaire), le gain (0.5, 1, 2, 5, 10) et les limites d'acquisition, pour avoir une carte bien configurée. Ces limites sont calculées selon le gain et la polarité suivant le tableau si dessous :

Gain par canal (configurable par soft)	Les gammes (configurable par soft)	
	Bipolaire	Unipolaire
0.5	-10V à +10V	-
1	-5V à +5V	0 à 10V
2	-2.5V à +2.5V	0 à 5V
5	-1V à +1V	0 à 2V
10	-500mV à +500mV	0 à 1mV
20	-250mV à +250mV	0 à 500mV
50	-100mV à +100mV	0 à 200mV
100	-50mV à +50mV	0 à 100mV

- Apres les configurations, le système doit lancer l'acquisition et attendre la fin d'une période d'échantillonnage (dans notre cas cette période est exprimée par 1 seconde).
- Le système récupère l'échantillon stocké dans le buffer de la source.

Le diagramme d'activité suivant exprime le cas d'utilisation «Acquisition des données»

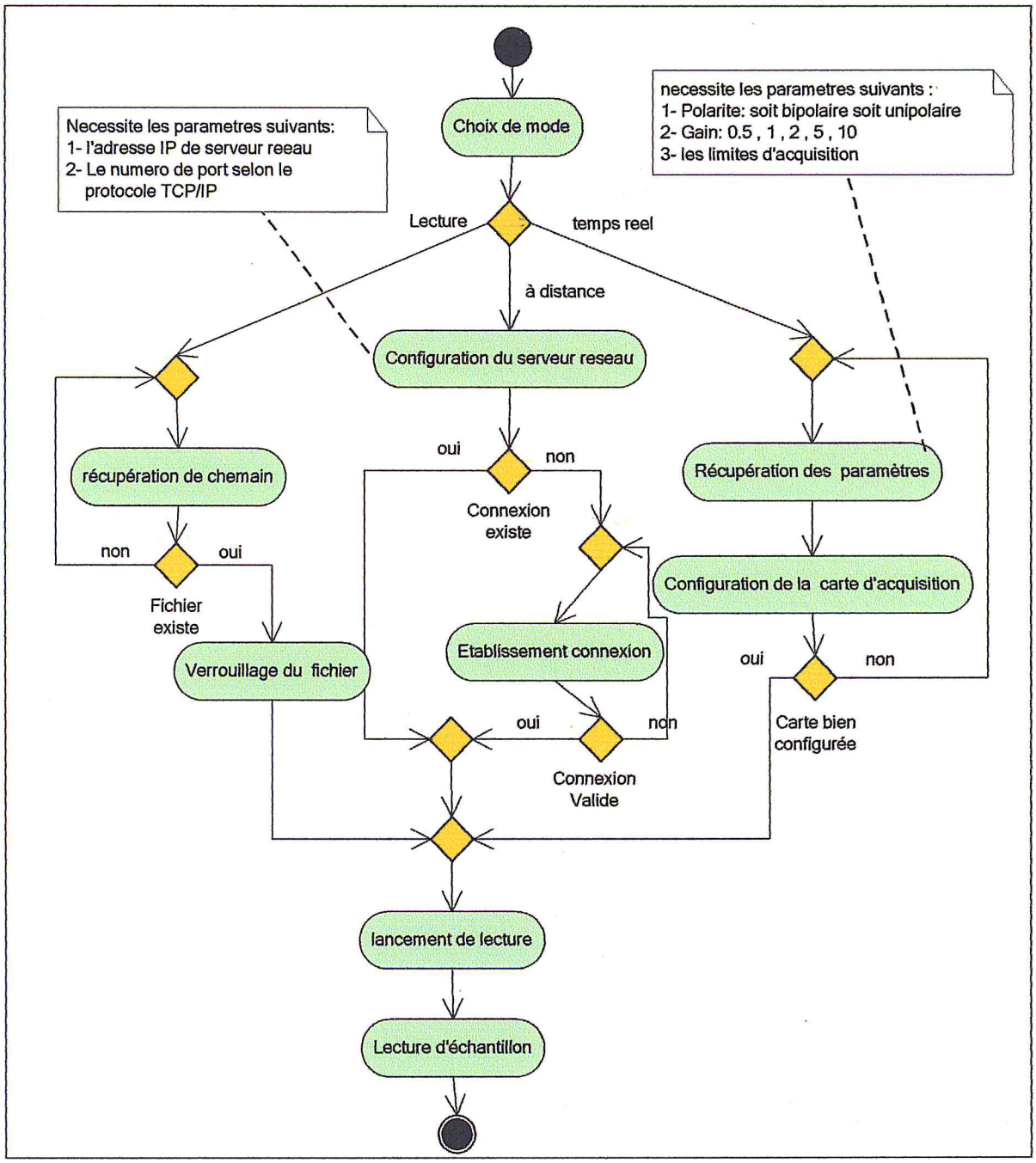


Figure III- 4 : Le diagramme d'activité pour le cas « Acquisition »

2.2.4. Le cas d'utilisation « Acquisition »

L' acquisition n'est pas demandé par l'utilisateur, elle présente un cas utilisé pour réaliser les fonctionnalités du système. L'acquisition des données utilise la sélection du mode : soit en lecture depuis un fichier, ou temps réel depuis la

carte d'acquisition, ou à distance depuis un serveur réseau, ces trois modes configurent les paramètres de la source avant l'acquisition, qui est exprimé par la relation << uses >>entre le cas « la sélection du mode » et le cas « configuration de la source ».

L'acquisition est la lecture des échantillons depuis le buffer de la source, après qu'il a donnée la commande de l'acquisition a la source, ceci est exprimé par le cas d'utilisation « lancement de lecture », la source stocke l'échantillon dans son buffer après la fin d'acquisition.

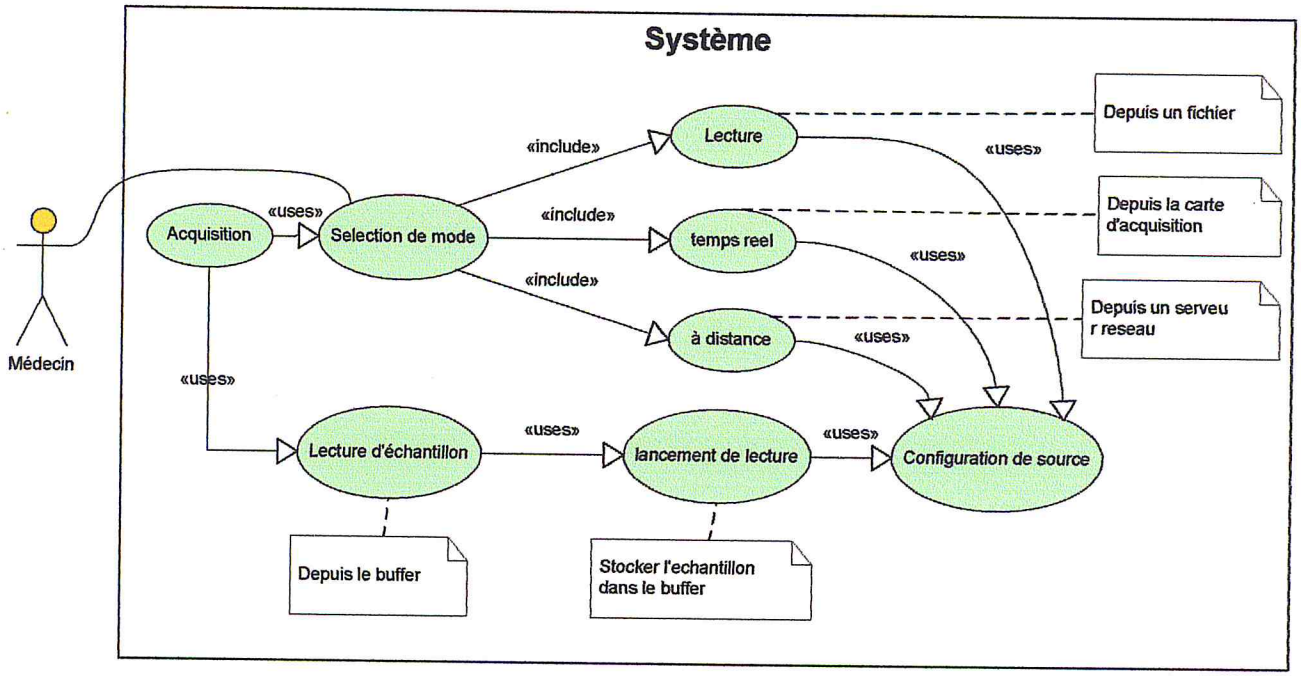


Figure III- 5 : diagramme des cas d'utilisation détaillé « Acquisition »

2.2.5. Le diagramme d'activité pour le cas « Traitement »

Le diagramme d'activité suivant représente le cas d'utilisation « traitement numérique du signal »

- Le contrôleur de traitement récupère l'échantillon,
- Si l'utilisateur demande de faire l'amplification, alors le contrôleur de traitement réalise l'amplification temporelle,
- Si l'utilisateur demande de faire le filtrage, alors le contrôleur de traitement récupère premièrement le genre, le type et l'ordre du filtre puis effectue le filtrage.

- Si l'utilisateur demande de faire la transformation de Fourier, le contrôleur de traitement récupère le type de fenêtre, puis il procède au fenêtrage puis en transformation de Fourier, et suivant le choix du spectre, le contrôleur de traitement convertit le spectre en spectre de puissance si nécessaire.
- Pour limiter l'affichage, il faut récupérer les limites puis éliminer les fréquences inutiles.
- A la fin du traitement une amplification fréquentielle est effectuée suivant la demande de l'utilisateur.

2.2.6. Le cas d'utilisation « Traitement »

Le cas « traitement » inclut quatre autres cas qui sont : le filtrage, transformation de fourrier rapide (FFT), limitation et l'amplification, les trois premier cas utilisent le cas « séparation » et le dernier inclut deux autre cas : amplification fréquentielle ou temporelle.

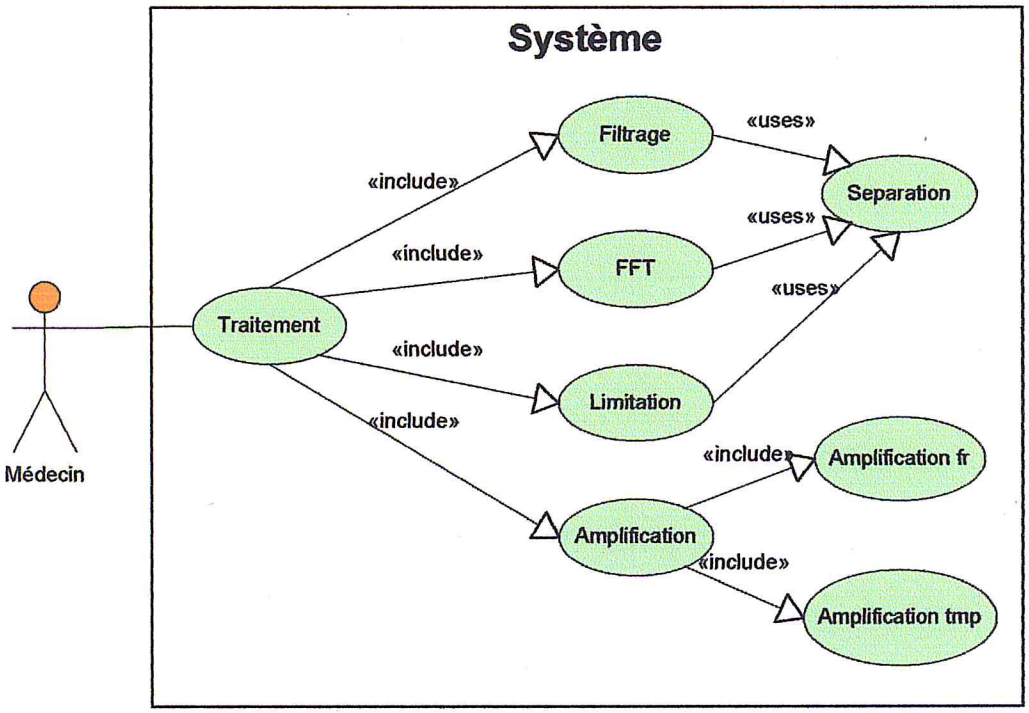


Figure III- 7 : diagramme des cas d'utilisation détaillé « Traitement »

2.2.7. Le cas d'utilisation « Filtrage »

Pour faire le filtrage il faut identifier le type de filtre, le type de filtre sera choisit parmi les valeurs suivantes : passe bande, coupe bande, passe haut et passe bas. Et il faut identifier aussi le genre de filtre qui est introduit par l'utilisateur parmi ses valeurs possibles on trouve : elliptique, Butterworth, Bessel, chebyshev et inverse-chebyshev,

Le filtrage nécessite l'introduction de fréquences de coupures Higher et Lower : c'est le cas limitation.

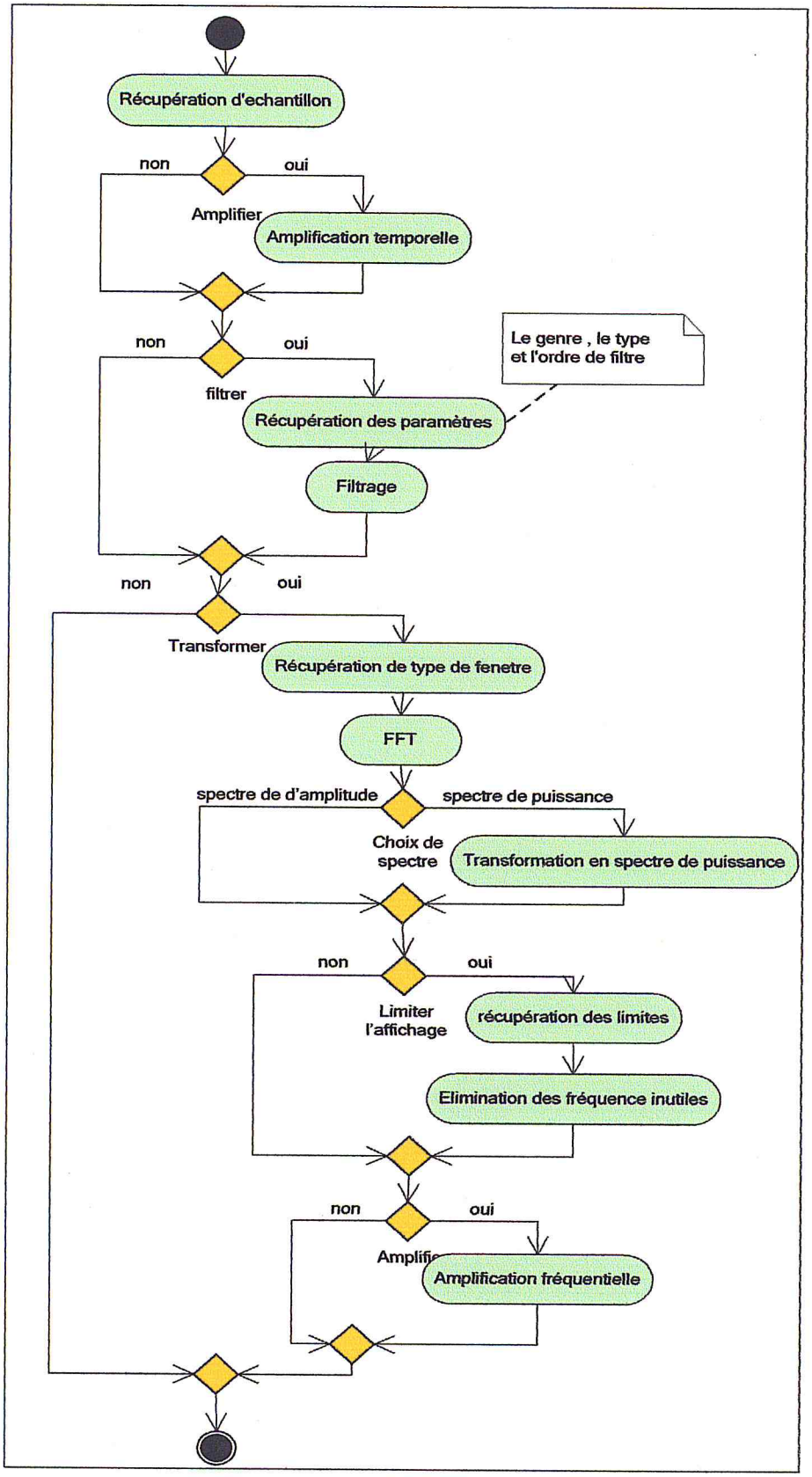


Figure III- 6: Le diagramme d'activité pour le cas « Traitement numérique du signal »

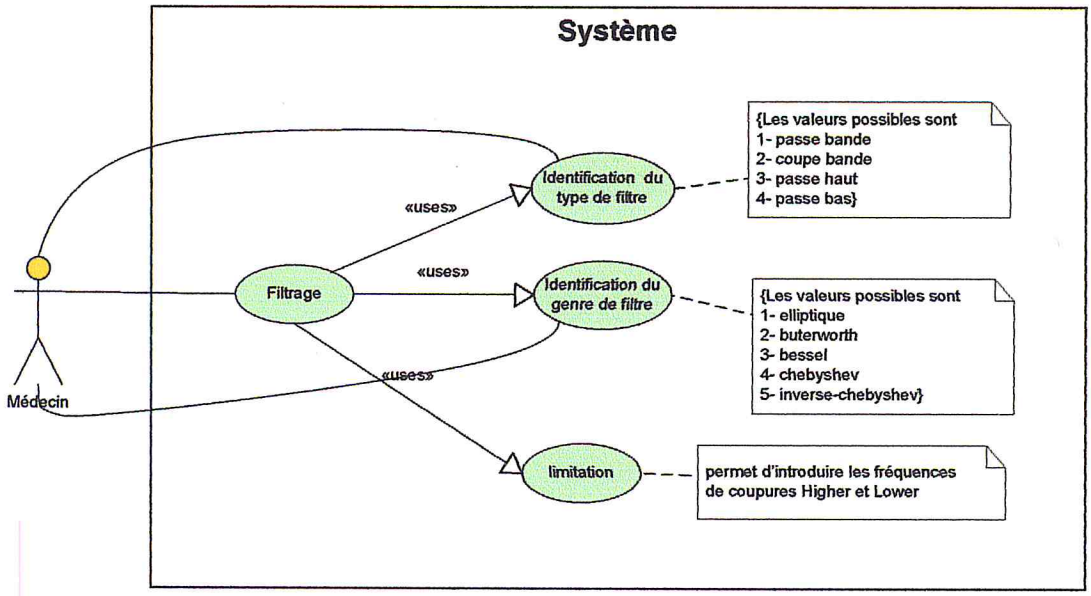


Figure III- 8 :diagramme des cas d'utilisation détaillé « Filtrage »

2.2.8. Le cas d'utilisation « FFT (transformation de Fourier rapide) »

Le cas « FFT » est un cas qui est demandé par l'utilisateur, pour réaliser la transformation de Fourier rapide, il nous faut introduire des certains paramètres qui sont introduit par l'utilisateur :

- le type de spectre soit de puissance ou d'amplitude,
- le type de fenêtre qui peut prendre l'une des valeurs suivantes : Uniform, Hamming, Hanning, Blackman-Harris, Blackman, Exact-Blackman, Low-Sidelobe, Flat-Top, Four-Term Blackman-Harris et Seven-Term Blackman-Harris,

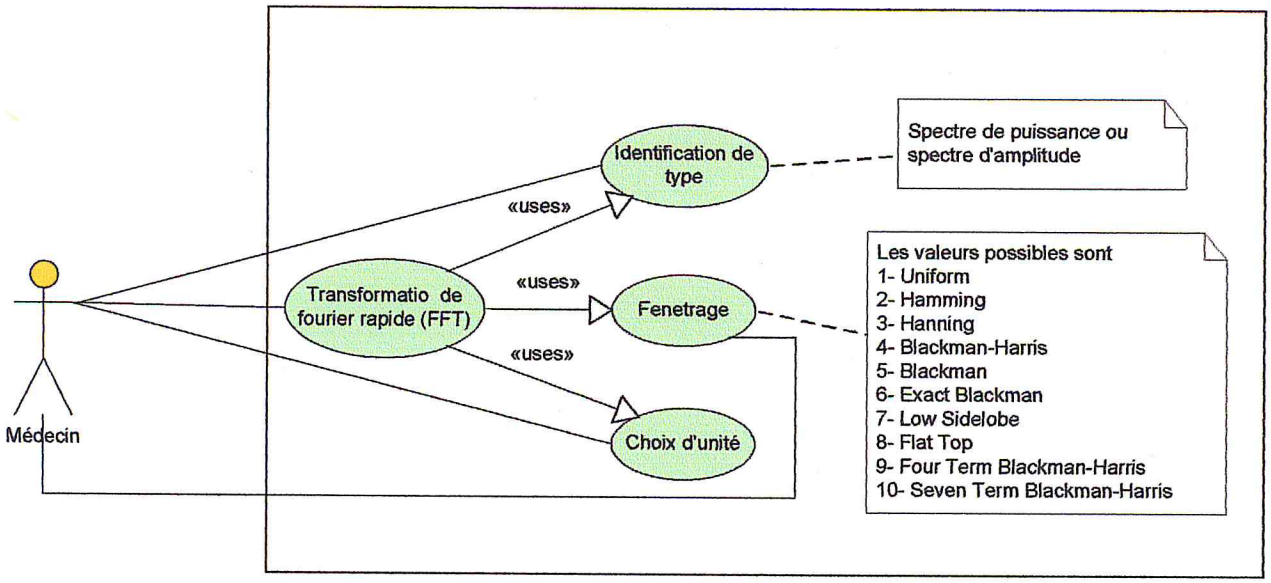


Figure III- 9 : diagramme des cas d'utilisation détaillé « FFT »

2.2.9. Le diagramme d'activité pour le cas « Visualisation »

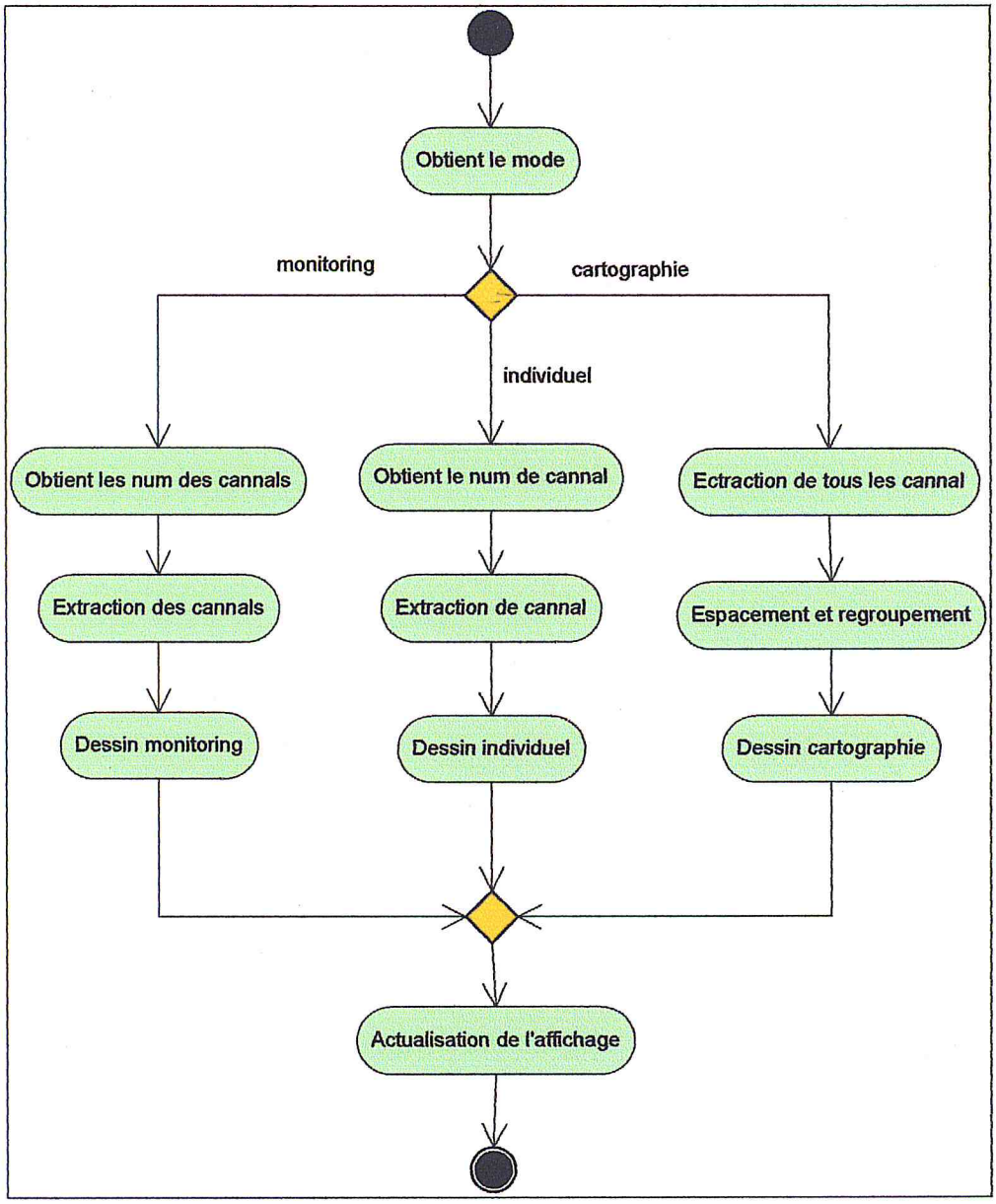


Figure III- 10 : Le diagramme d'activité pour le cas « Visualisation »

2.2.10. Le cas d'utilisation « Visualisation »

Le cas « visualisation » utilise le cas « choix du graphe » et inclut trois autres cas :

- affichage cartographie : il permet d'afficher tous les signaux acquis dans un même graphe pour que le médecin peut avoir une idée générale sur l'état du patient, pour cela il faut savoir l'espacement entre ces graphes pour les regroupés, ça est présenté dans le diagramme des cas d'utilisation par la

relation <<uses >>entre le cas « affichage cartographie » et les deux ces « espacement » et « regroupement »,

- affichage monitoring : il permet d'afficher deux signaux sur l'écran, pour que le médecin puisse les comparer, chacun est supposé comme un affichage individuel.
- Affichage individuel : il permet d'afficher un seul signal sur l'écran ainsi que ses paramètres.

Le cas « affichage monitoring » utilise le cas «Affichage individuel » car on considère que pour chaque affichage d'un des deux signaux en monitoring, comme un affichage individuel.

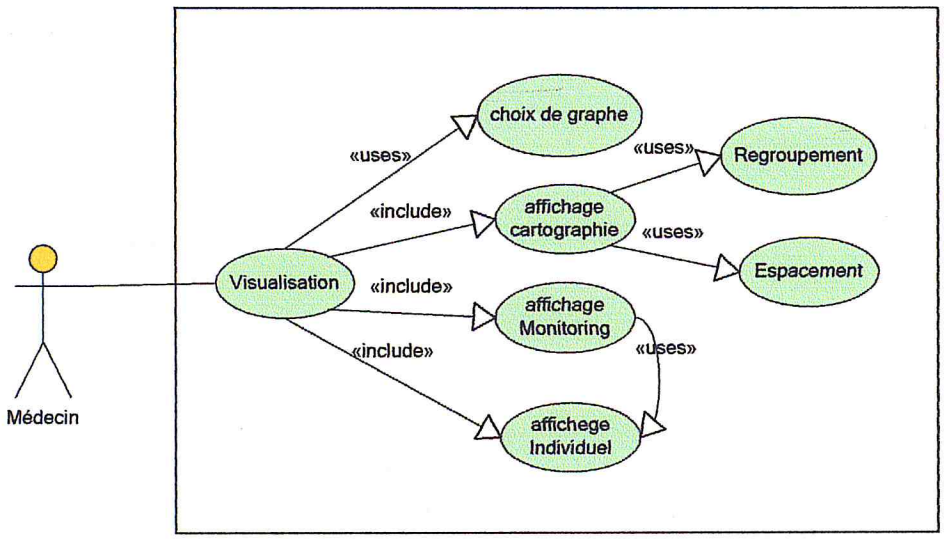


Figure III- 11: diagramme des cas d'utilisation détaillé « Visualisation»

2.2.11. Le diagramme d'activité pour le cas « Enregistrement »

- Le contrôleur d'enregistrement récupère le chemin de fichier à enregistrer,
- Si le fichier est fermé, il doit l'ouvrir,
- Après la vérification du fichier : si le fichier n'existe pas le système va créer un nouveau fichier, et dans le cas ou le fichier existe, soit il l'écrasse ou il l'ouvrir pour l'ajout dans le même fichier, selon le choix de l'utilisateur,
- Une fois le fichier est ouvert, le contrôleur d'enregistrement doit le verrouiller,
- Le contrôleur d'enregistrement récupère les échantillons,
- Les convertir en blocs, puis les ajouter au fichier,

- Si l'utilisateur demande de quitter, une fermeture du fichier est réalisée. Apres qu'il l'enregistre.

Ces descriptions sont illustrées dans le diagramme d'activité de la figure suivante :

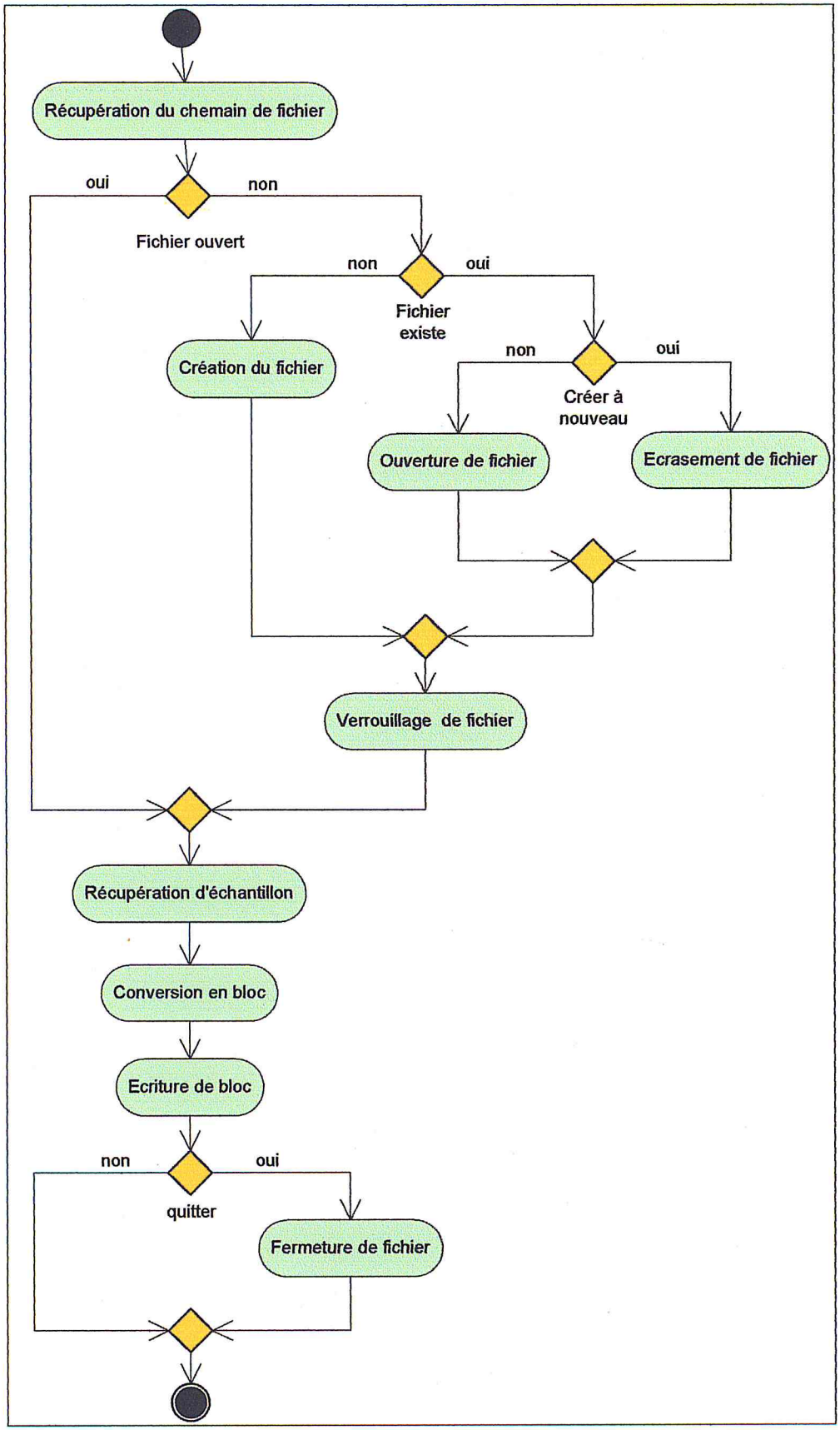


Figure III- 12: Le diagramme d'activité pour le cas « Enregistrement»

2.2.12. Le cas d'utilisation « Enregistrement »

Pour faire l'enregistrement d'un fichier, il faut premièrement ouvrir ce fichier, passer ensuite à l'écriture des blocs qui demande une conversion en binaire des échantillons, puis de fermer ce fichier après qu'il soit sauvegardé.
Pour ouvrir un fichier, il y a trois possibilités :

- soit la création d'un nouveau fichier s'il n'existe pas,
- soit l'écrasement d'un fichier dans le cas où on va créer un nouveau fichier sous un nom qui existe,
- soit l'ouverture d'un fichier existe.

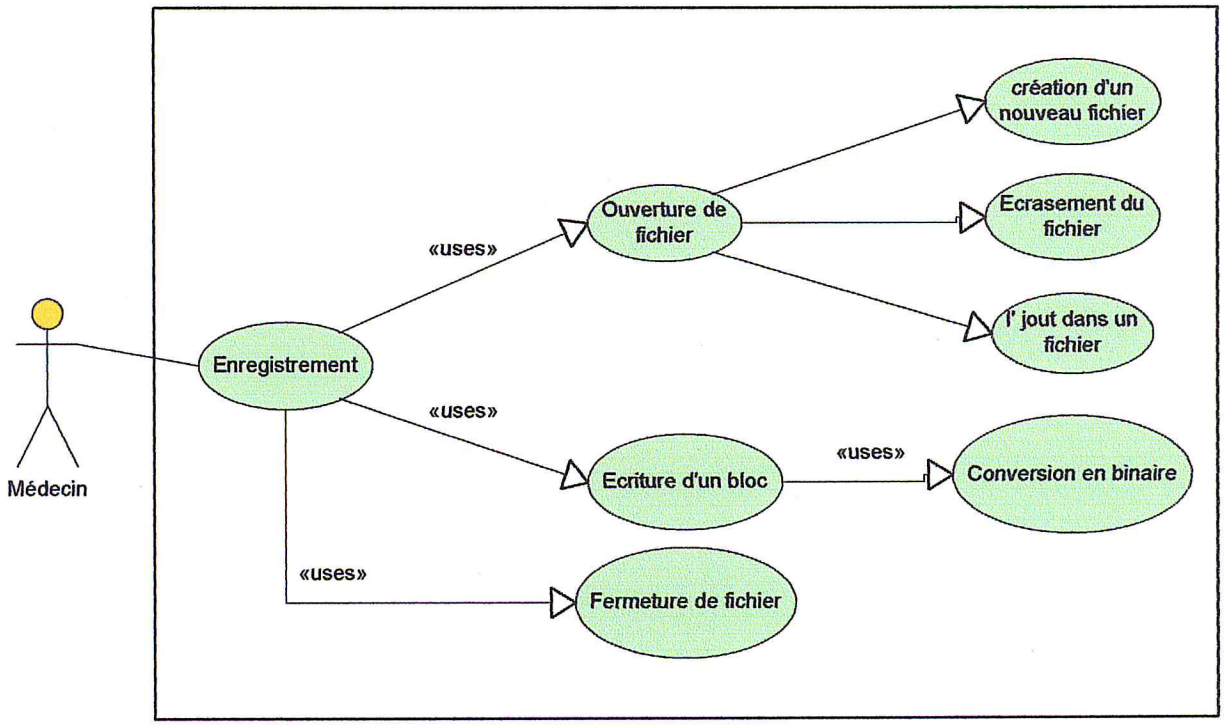


Figure III- 13 : diagramme des cas d'utilisation détaillé « Enregistrement»

2.2.13. Le diagramme d'activité pour le cas « Publication »

- Le système récupère l'échantillon,
- Il convertit l'échantillon en message,
- Il duplique les messages, une copie pour chaque application cliente connectée à cette application,
- Il empile la copie de chaque connexion dans sa file d'attente,

Et de l'autre coté :

- Le système capture les demandes du client,
- Si le client demande une fermeture de la connexion, le système doit la fermer,
- Si le client demande un établissement d'une connexion, le système doit la créer,
- Chaque connexion procède au dépilement des messages, et les transmit vers l'application cliente, qui se charge par le dialogue avec elle.
- Une fermeture du système sera établie suivant la demande du l'utilisateur

La figure suivante illustre le diagramme d'activité pour le cas utilisation « publication »

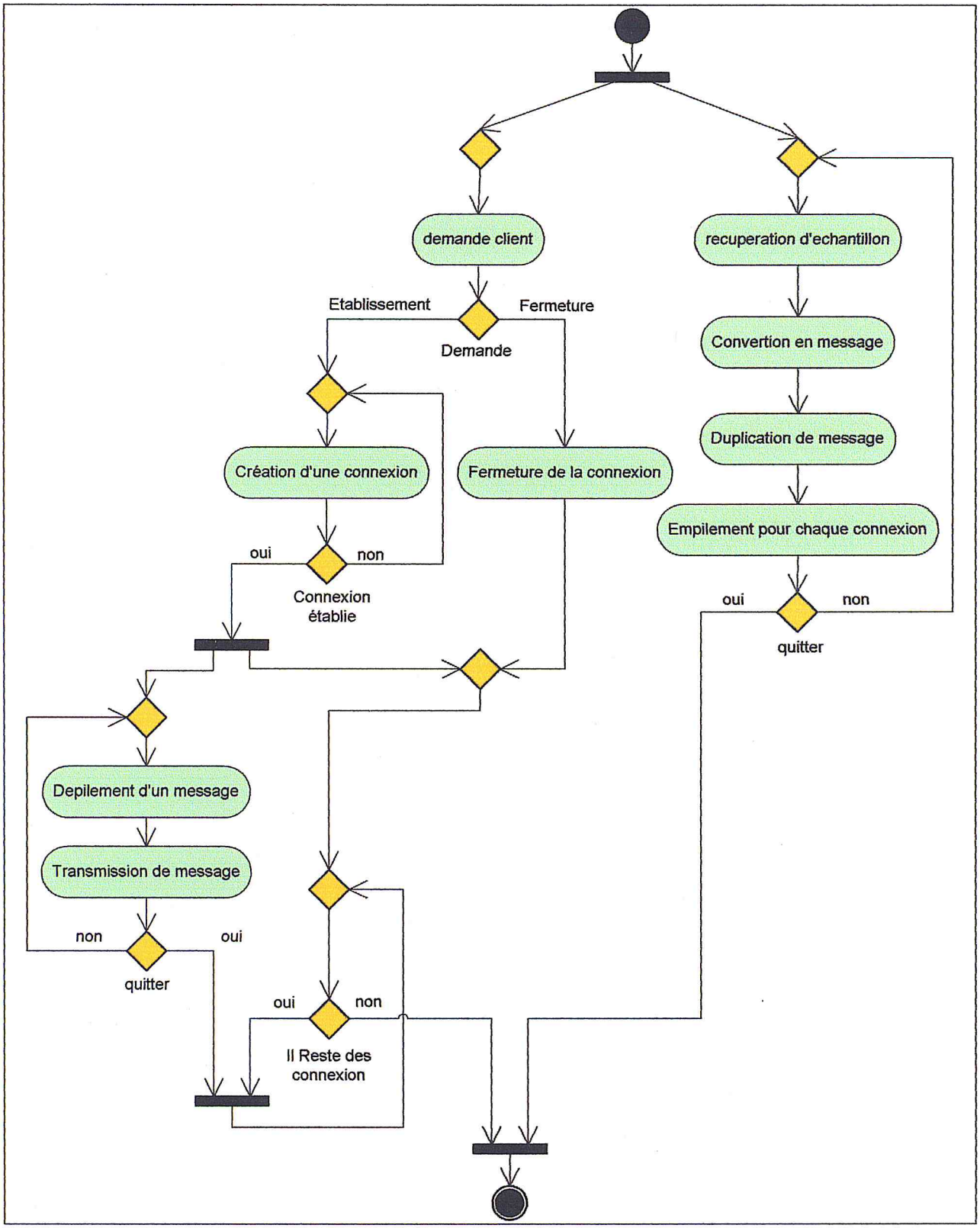


Figure III- 14 : Le diagramme d'activité pour le cas « Publication»

2.2.14. Le cas d'utilisation « Publication »

Pour publier des données dans un réseau il nous faut : la gestion des connexions, la gestion des messages et la transmission des messages, ce qui est exprimé par la relation <<uses>>entre les cas « publication des données » et les trois cas «gestion des connexions », « gestion des messages » et « transmission des messages ».

Une connexion doit être demandé par une autre application situé dans la même machine ou sur une machine dans un même réseau local, pour permettre sa création, elle est contrôler pendant sa durée de vie, si en cas de mal fonctionnement ou une anomalie produite par la connexion pendant la transmission des message, il faut la fermer. Un autre cas où il faut fermer la connexion, c'est la demande du client, ces cas sont inclut dans le cas « gestion des connexions ».
Pour chaque donnée à publier, le système doit la convertir en messages puis duplique ce message en plusieurs copies, chacune pour une connexion c'est le cas « gestion des messages », la connexion se charger par l'envoi de sa copie, c'est le cas d'utilisation « transmission des messages ».

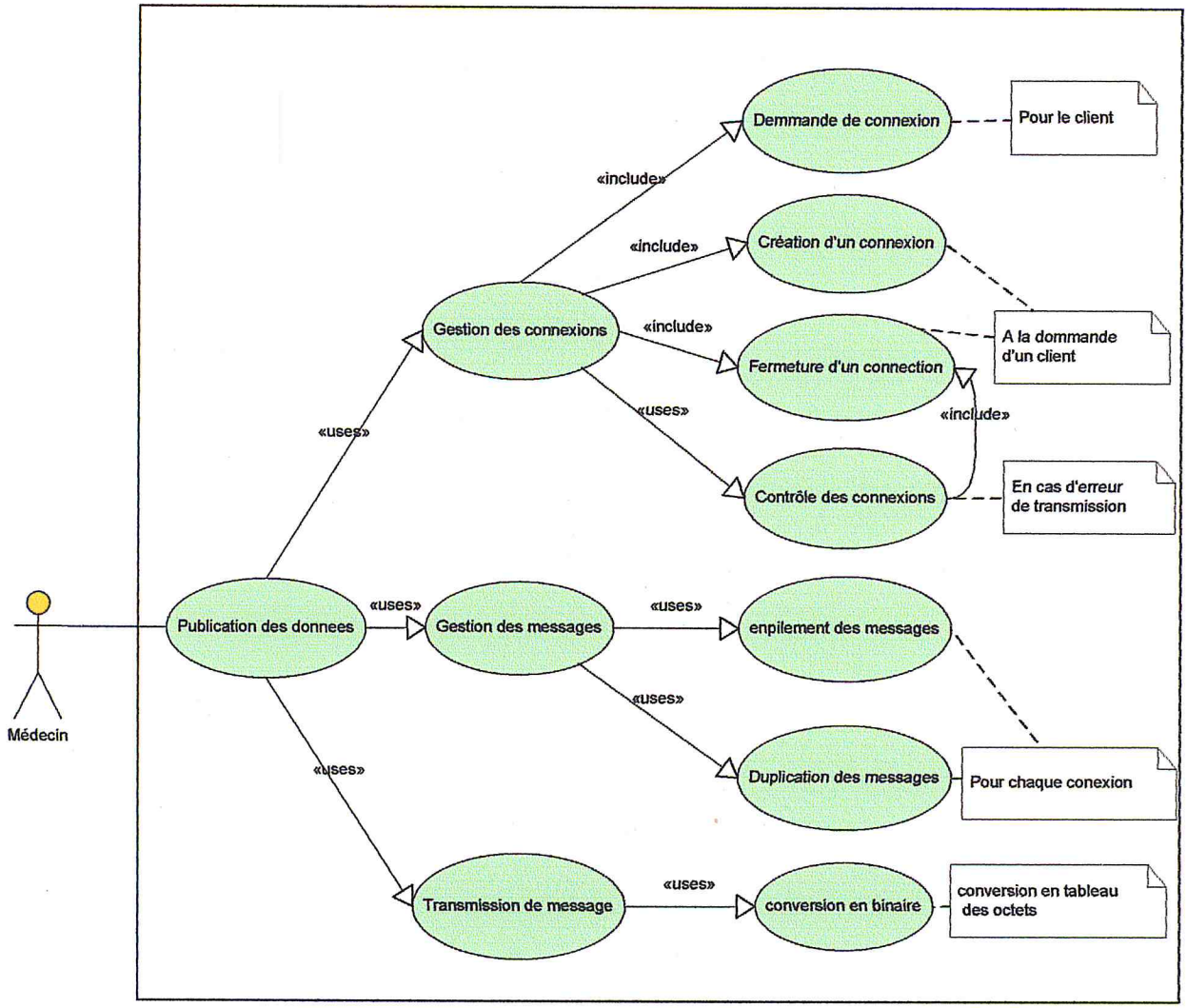


Figure III- 15 : diagramme des cas d'utilisation détaillé « Publication»

2.2.15. Le diagramme d'activité pour le cas « Consultation rapide »

- Le système récupère le chemin de fichier,
- Si le fichier existe, une ouverture de ce fichier est effectuée,
- Tant que ce fichier n'est pas lit complètement, pour chaque bloc il faut :
- Lire ce bloc,
- Convertir ce bloc en échantillon,
- Et à la fin une visualisation est réalisée
- Si la lecture du fichier atteint sa fin, il faut le fermer.

Le diagramme d'activité suivant représente le cas d'utilisation « Consultation rapide »

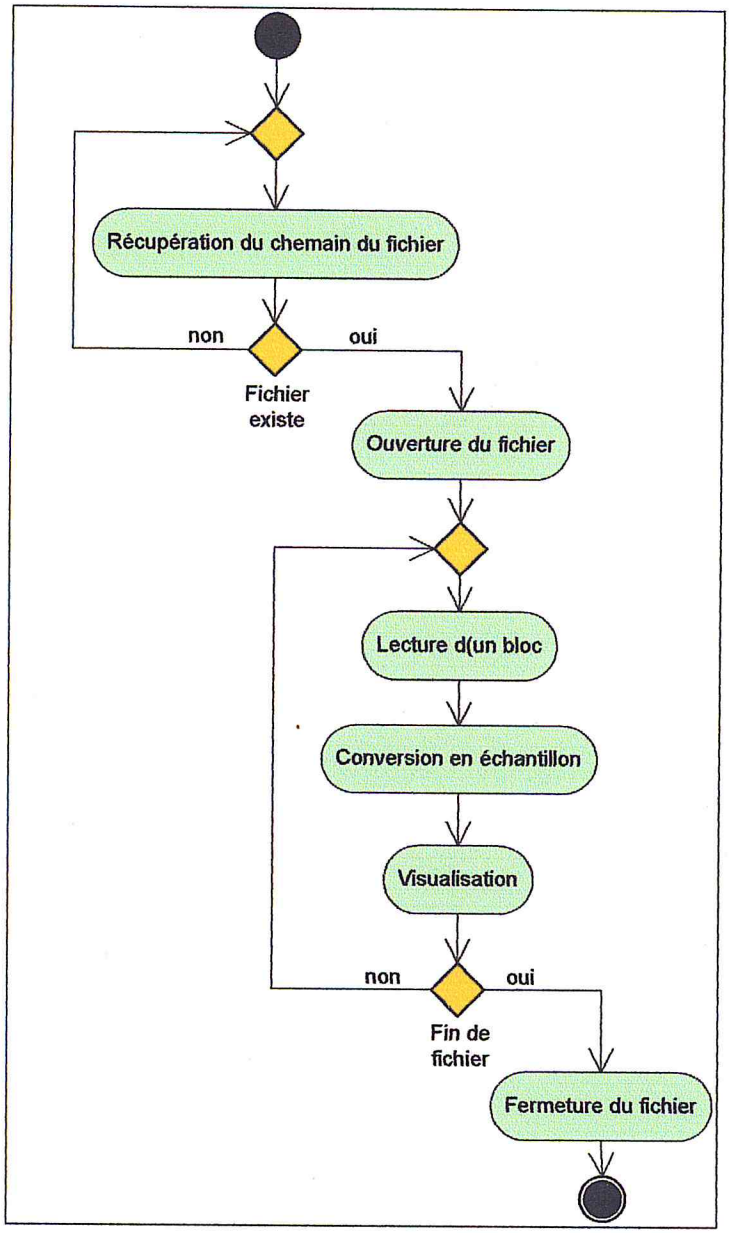


Figure III- 16 : Le diagramme d'activité pour le cas « Consultation rapide »

2.2.16. Le cas d'utilisation « Consultation rapide »

Le cas d'utilisation « consultation rapide » est un cas demandé par l'utilisateur, ce cas utilise trois autres cas :

- L'ouverture des fichiers et par conséquence une allocation des ressources,

➢ Le chargement du fichier : pour chaque bloc, une lecture puis une conversion en échantillon sont établies, ce qui exprimé par la relation <<uses>>,

➢ La fermeture du fichier qui engendre une libération des ressources.

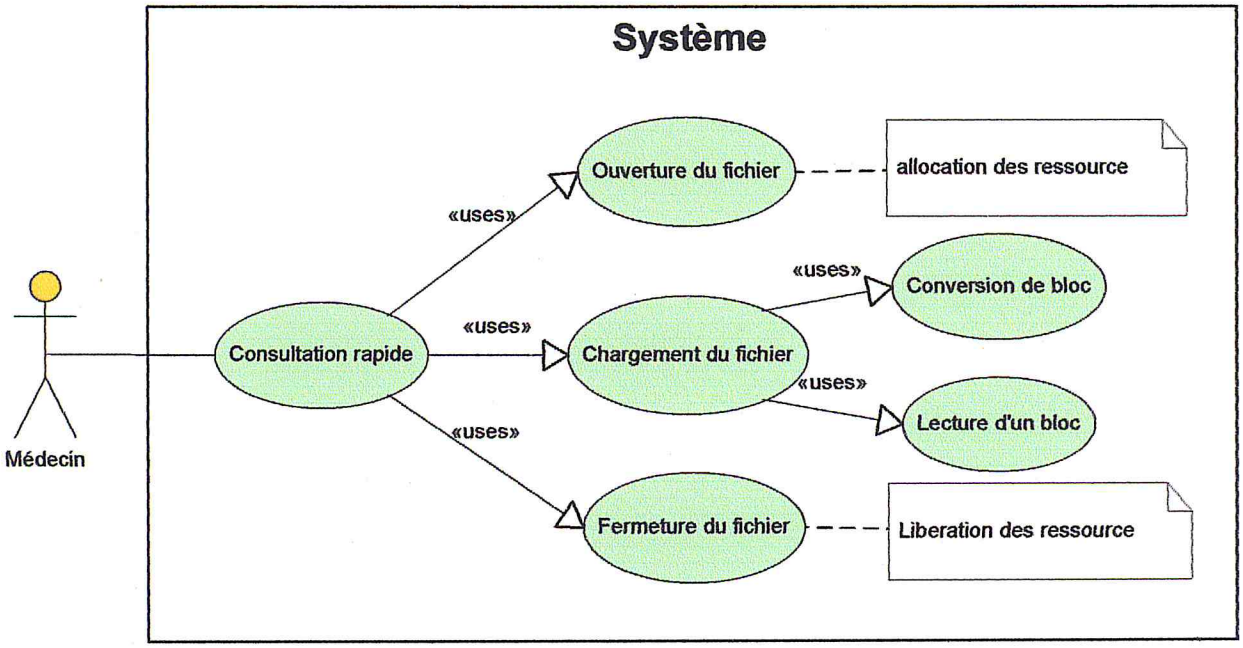


Figure III- 17 : diagramme des cas d'utilisation détaillé « Consultation rapide »

3. Analyse

Le but de l'étape d'analyse est de définir une structure robuste et extensible qui nous servira de base pour la construction du système [Brès, 93].

L'objectif de cette phase est de déterminer les éléments intervenant dans le système à construire, ainsi que leur structure et leurs relations. Nous présentons dans cette partie : les diagrammes de package et les diagrammes de classes.

3.1. Le diagramme de package

Le diagramme de package représente les structure du système en terme de sous système, de package et de classes.... L'analyse de notre système nous a permis de dégager les différents composants du système, que nous classifions dans 3 principales package : Outils, Client_Serveur et application.

→ Le package Outils : ce package regroupe les objets utilisés par le package Client_serveur qui n'ont pas une forte relation avec le terme réseau et peut être utilisés dans une autre application.

→ Le package Client_Serveur : qui contient les classe de gestion de réseau comme serveur, client, connexion.....

→ Le package Application : qui contient les classes de base de notre conception comme contrôleur d'interface, contrôleur de traitement....

Le diagramme suivant présente l'ensemble des packages du système.

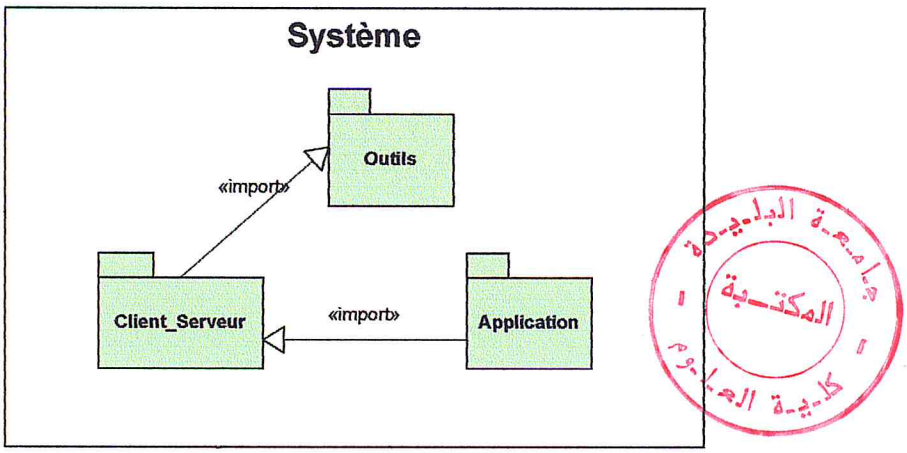


Figure III- 18: Diagramme de package du système

Le package « application » accède au package « Client-Serveur » qui import le package « outils »

3.1.1. Le package Outils

Le package « Outils », contient tous les outils nécessaire pour le fonctionnent du système, il s'agit des composants suivantes :

- Agent : cette classe cache les détaille de création, de gestion et de destruction d'un processus, cette classe implémente les algorithmes de synchronisation et d'accès à la section critique par un moniteur.
- Bloc : cette classe représente une unité des données échangées entre le package Application et Client_serveur, le développeur qui veut importer et utiliser ce dernier dans sa application ,implémente la fonction En _Octets () : byte[] de cette classe.
- File critique : c'est une liste avec l'implémentation de la concept FIFO (first in first out) mais avec la gestion d'accès de plusieurs processus du système en même temps (implémentation des algorithmes d'accès à la section critique par un moniteur.)

➢ Tableau critique : c'est un tableau avec la gestion d'accès de plusieurs processus du système en même temps (implémentation des algorithmes d'accès à la section critique par un moniteur.)

Le diagramme de package suivant illustre les composants de ce package :

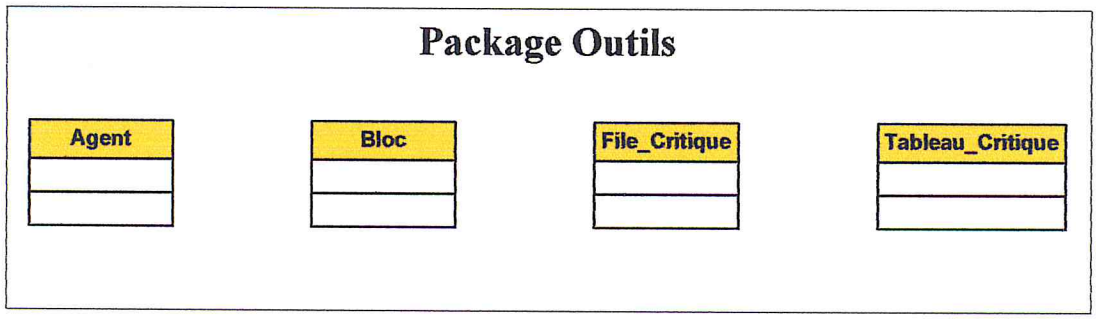


Figure III- 19: Les composants du package « Outils »

3.1.2. Le package Client_Serveur

Ce package permet de gérer les connexions : c'est par les politiques de récepteur et dupliqueur.

- le récepteur gère la réception des demandes de connexion, la création et le contrôle des connexions.
- le dupliqueur gère la réception et la duplication des messages.
- la connexion s'occupe de la transmission et la réception des messages.
- le client.
- le serveur.

Le diagramme de package suivant illustre les composants de ce package:

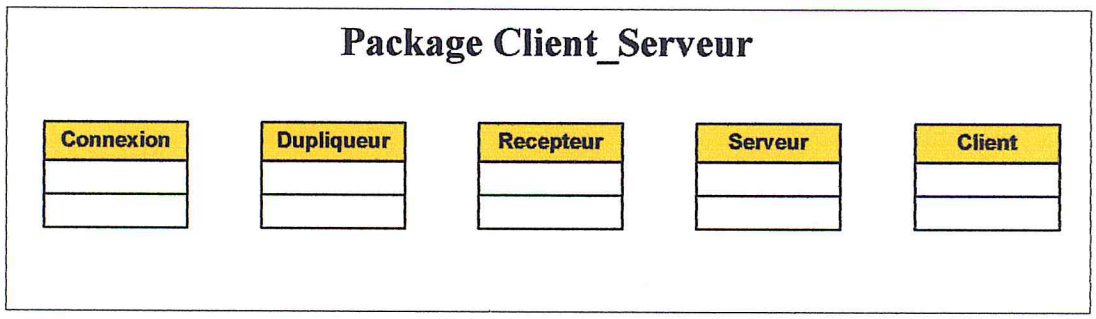


Figure III- 20: Les composants du package Client_Serveur

3.1.3. Le package application

Ce package concerne les différents composants nécessaires pour une application ; il s'agit de :

- contrôleur de visualisation.
- contrôleur de carte d'acquisition.
- contrôleur de Serveur_reseau.
- Contrôleur Interface Graphique.
- Contrôleur Fichier.
- Contrôleur Source.
- Contrôleur de traitement.
- Echantillon.
- Contrôleur d'enregistrement.

Le diagramme de package suivant illustre les composants de ce package:

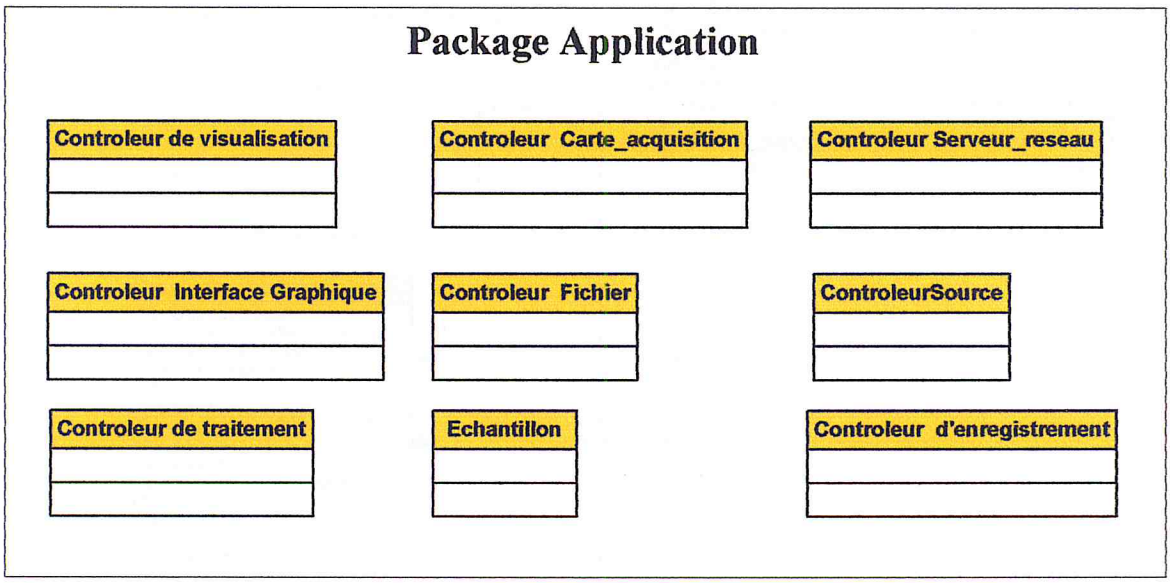


Figure III- 21 : les différents composant du package application

3.2. Les diagrammes de classes

Le diagramme de classe décrit la structure du système, en terme de classes composant le système et les relations entre ces classes. Le diagramme de classe suivant contient toutes les entités de toutes les packages, et les relations entre eux.

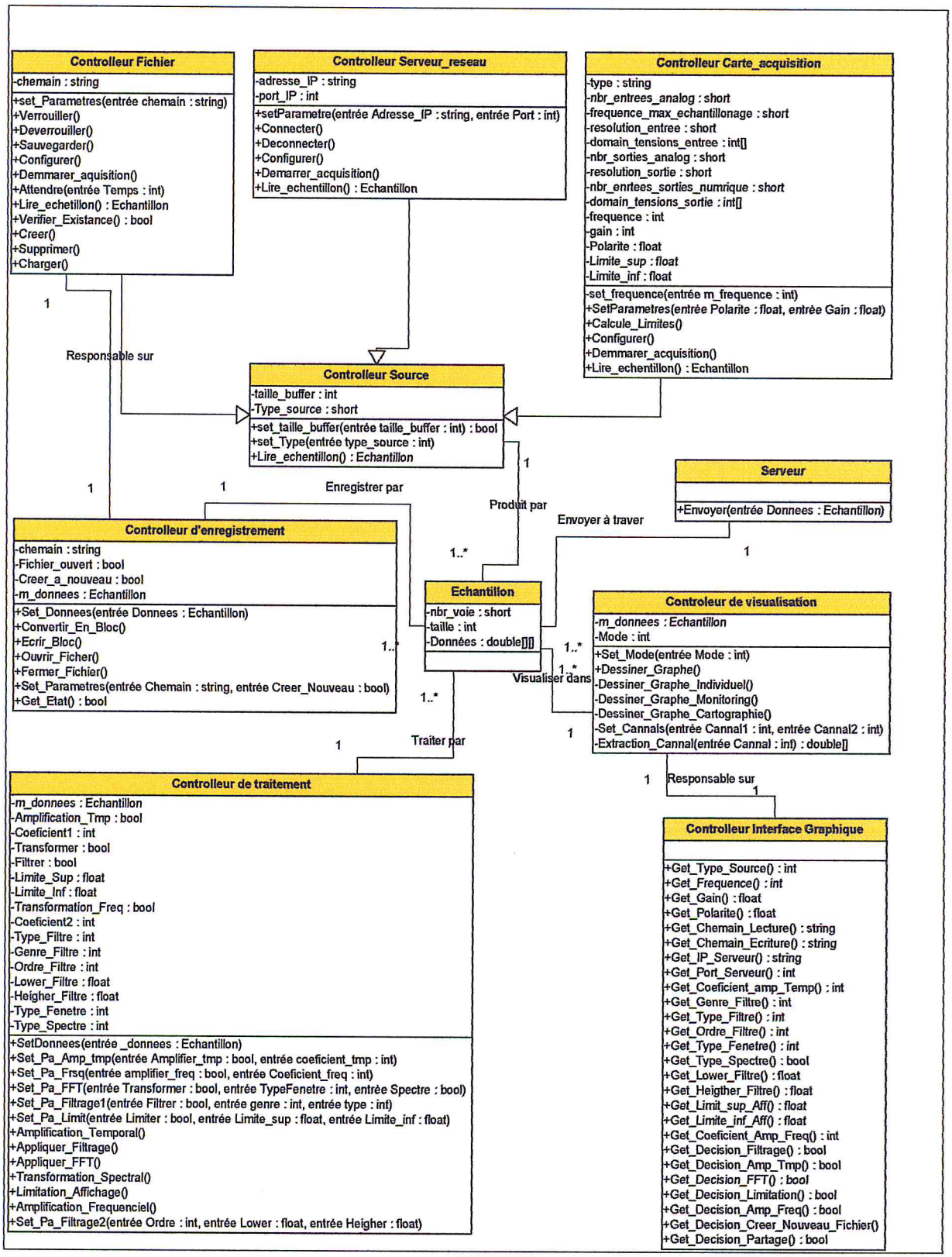
Un agent est une classe générale , elle peut etre spécialisé en : recepteur , dupliqueur et connexion , c'est la relation d'héritage.

La File critique et Tableau critique n'acceptent comme des éléments que des objets instancier de la classe objet ou des objets instancier d'une classe hérité directement (classe Bloc) ou indirectement (classe connexion) de la classe Objet.
La classe serveur est composée d'une liste des connexions (stocké dans un tableau critique) et une file d'attente peut contenir les messages à envoyer, cette file est de type file critique, un objet récepteur et un objet dupliqueur.
La classe client contient un seul objet Connexion, une liste peut contenir les messages reçus, cette liste est de type file critique.

3.2.2. Diagrammes de classes globale de l'application

Un contrôleur de source peut être : soit un contrôleur de fichier, contrôleur serveur réseau ou un contrôleur de carte d'acquisition, ce qui présenté dans le diagrammes de classe par la relation d'héritage entre les trois classes contrôleur de fichier, contrôleur serveur réseau, contrôleur de carte d'acquisition et la classe contrôleur source car elles partagent certains attributs communs et des méthodes communes.

Plusieurs échantillons sont produits par un contrôleur de source, enregistrés par un contrôleur d'enregistrement, traités par un contrôleur de traitement, envoyés à travers le service réseau, et visualisés par un contrôleur de visualisation.
Un contrôleur visualisation gère le dessin des graphes situés sur un contrôleur d'interface graphique.


Controlleur Fichier
-chemain : string
+set_Parametres(entrée chemain : string)
+Verrouiller()
+Deverrouiller()
+Sauvegarder()
+Configurer()
+Demmarer_aquisition()
+Attendre(entrée Temps : int)
+Lire_echetillon() : Echantillon
+Verifier_Existance() : bool
+Creer()
+Supprimer()
+Charger()
Controlleur Serveur_reseau
-adresse_IP : string
-port_IP : int
+setParametre(entrée Adresse_IP : string, entrée Port : int)
+Connecter()
+Deconnecter()
+Configurer()
+Demarrer_acquisition()
+Lire_echentillon() : Echantillon
Controlleur Carte_acquisition
-type : string
-nbr_entrees_analog : short
-frequence_max_echantillonage : short
-resolution_entree : short
-domain_tensions_entree : int[]
-nbr_sorties_analog : short
-resolution_sortie : short
-nbr_enrtees_sorties_numrique : short
-domain_tensions_sortie : int[]
-frequence : int
-gain : int
-Polarite : float
-Limite_sup : float
-Limite_inf : float
-set_frequence(entrée m_frequence : int)
+SetParametres(entrée Polarite : float, entrée Gain : float)
+Calcule_Limites()
+Configurer()
+Demmarer_acquisition()
+Lire_echentillon() : Echantillon
Controlleur Source
-taille_buffer : int
-Type_source : short
+set_taille_buffer(entrée taille_buffer : int) : bool
+set_Type(entrée type_source : int)
+Lire_echentillon() : Echantillon
Serveur
+Envoyer(entrée Donnees : Echantillon)
Controlleur d'enregistrement
-chemain : string
-Fichier_ouvert : bool
-Creer_a_nouveau : bool
-m_donnees : Echantillon
+Set_Donnees(entrée Donnees : Echantillon)
+Convertir_En_Bloc()
+Ecrir_Bloc()
+Ouvrir_Ficher()
+Fermer_Fichier()
+Set_Parametres(entrée Chemain : string, entrée Creer_Nouveau : bool)
+Get_Etat() : bool
Echantillon
-nbr_voie : short
-taille : int
-Données : double[][]
Controleur de visualisation
-m_donnees : Echantillon
-Mode : int
+Set_Mode(entrée Mode : int)
+Dessiner_Graphe()
-Dessiner_Graphe_Individuel()
-Dessiner_Graphe_Monitoring()
-Dessiner_Graphe_Cartographie()
-Set_Cannals(entrée Cannal1 : int, entrée Cannal2 : int)
-Extraction_Cannal(entrée Cannal : int) : double[]
Controlleur de traitement
-m_donnees : Echantillon
-Amplification_Tmp : bool
-Coeficient1 : int
-Transformer : bool
-Filtrer : bool
-Limite_Sup : float
-Limite_Inf : float
-Transformation_Freq : bool
-Coeficient2 : int
-Type_Filtre : int
-Genre_Filtre : int
-Ordre_Filtre : int
-Lower_Filtre : float
-Heigher_Filtre : float
-Type_Fenetre : int
-Type_Spectre : int
+SetDonnees(entrée _donnees : Echantillon)
+Set_Pa_Amp_tmp(entrée Amplifier_tmp : bool, entrée coeficient_tmp : int)
+Set_Pa_Frsq(entrée amplifier_freq : bool, entrée Coeficient_freq : int)
+Set_Pa_FFT(entrée Transformer : bool, entrée TypeFenetre : int, entrée Spectre : bool)
+Set_Pa_Filtrage1(entrée Filtrer : bool, entrée genre : int, entrée type : int)
+Set_Pa_Limit(entrée Limiter : bool, entrée Limite_sup : float, entrée Limite_inf : float)
+Amplification_Temporal()
+Appliquer_Filtrage()
+Appliquer_FFT()
+Transformation_Spectral()
+Limitation_Affichage()
+Amplification_Frequenciel()
+Set_Pa_Filtrage2(entrée Ordre : int, entrée Lower : float, entrée Heigher : float)
Controlleur Interface Graphique
+Get_Type_Source() : int
+Get_Frequence() : int
+Get_Gain() : float
+Get_Polarite() : float
+Get_Chemain_Lecture() : string
+Get_Chemain_Ecriture() : string
+Get_IP_Serveur() : string
+Get_Port_Serveur() : int
+Get_Coeficient_amp_Temp() : int
+Get_Genre_Filtre() : int
+Get_Type_Filtre() : int
+Get_Ordre_Filtre() : int
+Get_Type_Fenetre() : int
+Get_Type_Spectre() : bool
+Get_Lower_Filtre() : float
+Get_Heigther_Filtre() : float
+Get_Limit_sup_Aff() : float
+Get_Limite_inf_Aff() : float
+Get_Coeficient_Amp_Freq() : int
+Get_Decision_Filtrage() : bool
+Get_Decision_Amp_Tmp() : bool
+Get_Decision_FFT() : bool
+Get_Decision_Limitation() : bool
+Get_Decision_Amp_Freq() : bool
+Get_Decision_Creer_Nouveau_Fichier()
+Get_Decision_Partage() : bool
1
Responsable sur
1
1
Enregistrer par
1
1
Produit par
Envoyer à traver
1..*
1
1..*
1..*
Visualiser dans
1..*
1
1
Traiter par
1
Responsable sur
1

4. Conception

La conception globale a pour but de décomposer le logiciel en modules et de préciser les interfaces et les fonctions de chaque module. A l'issue de cette étape, on obtient une description de l'architecture du logiciel et un ensemble de spécifications de ses divers composants [Bres, 93].

4.1. Conception globale :

4.1.1. L'architecteur de système

La figure suivante donne une vision globale de l'architecture de l'outil.

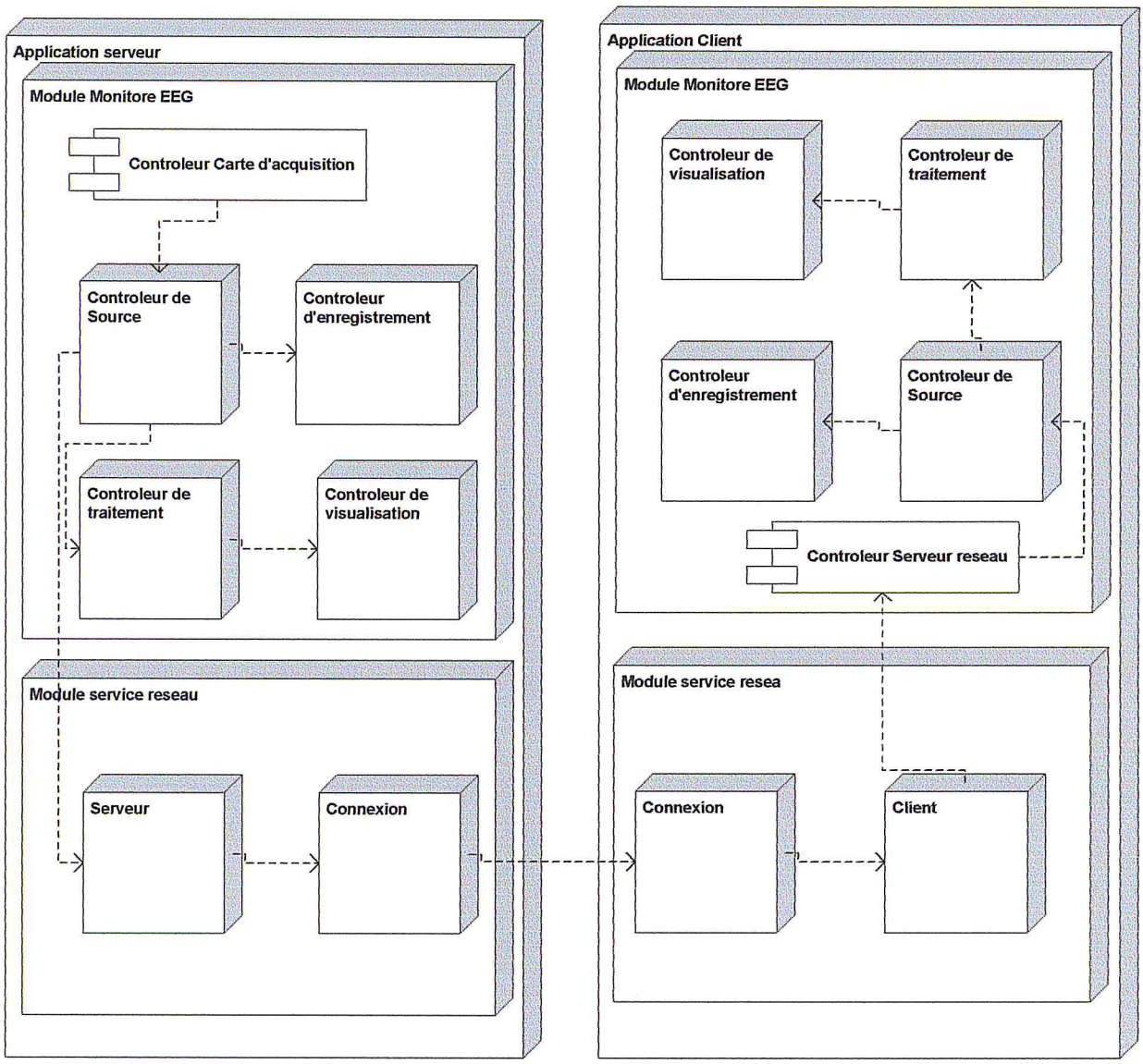


Figure III- 22: L'architecture du système

Afin de mettre en œuvre notre système, nous avons défini une architecture logicielle composée de deux (2) modules:

- Module 1 : moniteur EEG.
- Module 2 : service réseau.

Le module moniteur EEG, c'est une application développée avec Labview et sa sortie de type exécutable, et sa plateforme est la plateforme dont laquelle le Labview est installé pendant le développement de ce module.

Ce module offre des services d'acquisition, de traitement, de visualisation et l'enregistrement d'un signal EEG.

La base de ce module est le package globale de l'application, ce module peut fonctionner autonome sur un poste local mais il besoin d'autre modules pour fonctionner sur l'environnement de réseau qui s'appelle le module service réseau.

Le module service réseau est une librairie développé avec Labview et aussi avec la plateforme .Net de Microsoft à l'aide de le langage de programmation Visual C#.net, et sa sortie de type DLL (dynamic link library).

La plateforme de ce module est la plateforme dont laquelle le Labview est installé pendant le développement de ce module, ou seulement Microsoft Windows avec toutes les versions si le module est développé avec la plateforme .Net .

Ce module offre les fonctionnalités d'envoi des blocs d'octets entre plusieurs applications d'une manière fiable et robuste avec une précision temporelle très élevée.

4.1.2. Diagramme de composants

L'architecture de notre système peut être représenté en UML à l'aide de diagramme de composants suivant :

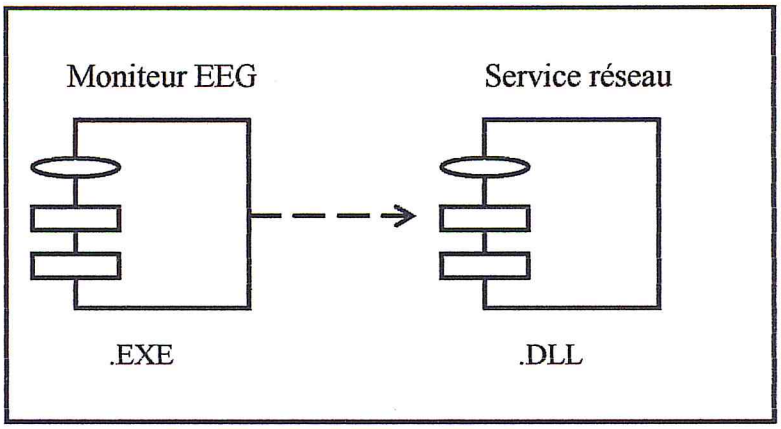


Figure III- 23: Le diagramme de composants pour les différents modules du système

Nous présentons dans le diagramme si dessus les différents modules qui composent le système, comme ils sont présentés dans le schéma de l'architecture du système, mais au point de vue des fichiers exécutables. Chaque composant est un élément physique qui représente une partie implémentée de système [Muller, 97].

4.2. Conception détaillée :

Les modules composent de plusieurs sous modules, chacun s'occupe de la réalisation d'une fonctionnalité particulière, ces fonctionnalités seront détaillées à l'aide des diagrammes de séquences UML présentées dans la section suivante.

4.2.1. Le module moniteur EEG

Nous présentons les diagrammes de séquences pour le module moniteur EEG dans ce qui suit :

4.2.1.1. Le diagrammes de séquence : acquisition

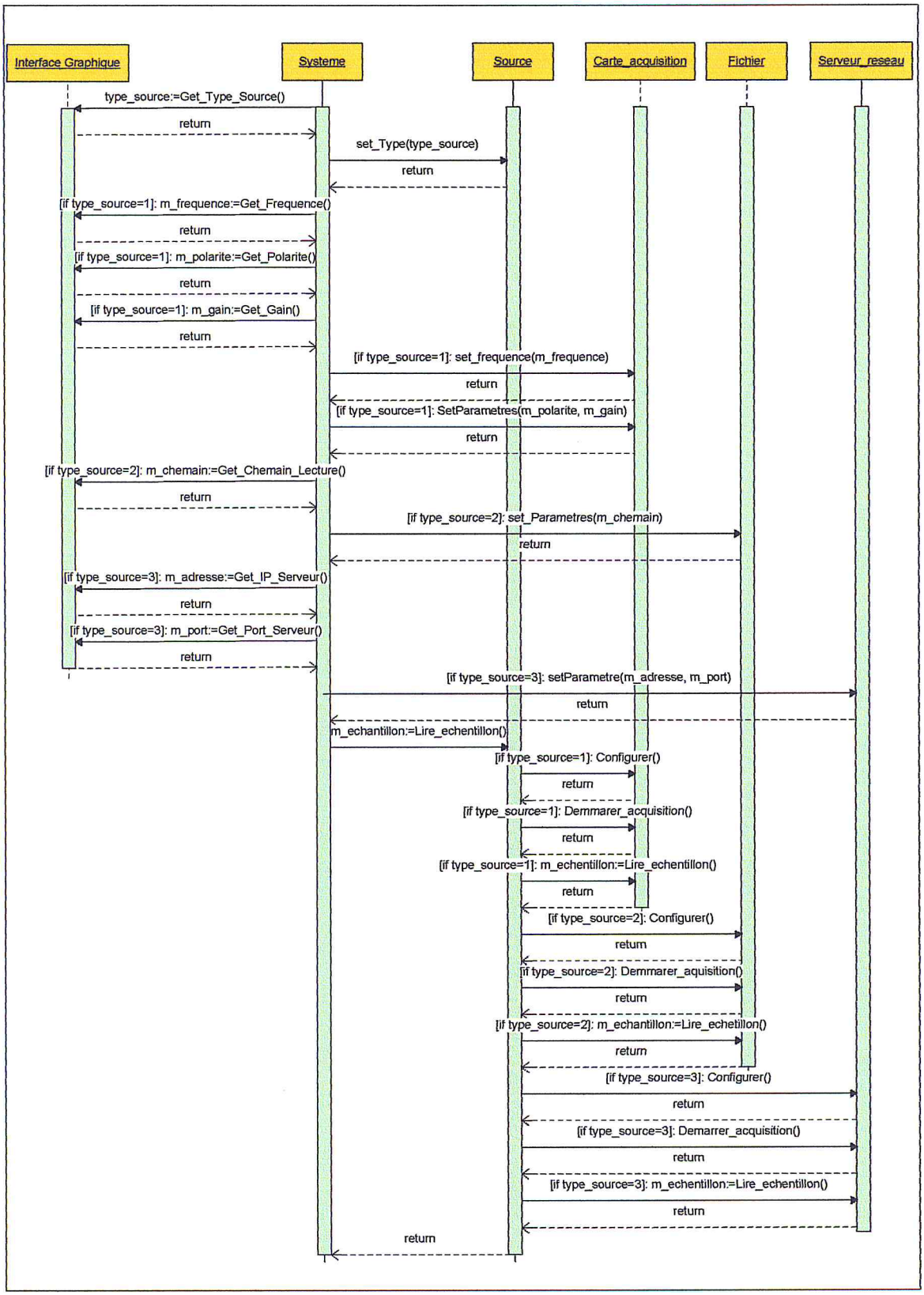


Figure III- 24:Le diagramme de séquence : acquisition

➢ **La description textuelle**

- Le système demande le type de source à l'interface graphique à travers l'invocation de la méthode Get_Type_Source () de l'interface graphique,
- Le système envoie le type de source à travers la méthode set_Type () de la source, vers le contrôleur de la source,
- Si le type de source =1, le système demande la fréquence, la polarité et le gain à l'interface graphique,
- Le système envoie la fréquence de la carte d'acquisition vers le contrôleur de la carte d'acquisition,
- Le système envoie les paramètres : la polarité et le gain vers le contrôleur de la carte d'acquisition,
- Si le type de source =2, le système demande le chemin de lecture à l'interface graphique,
- Le système envoie le chemin de fichier à travers la méthode set_Parametres (m_chemin) du Fichier, vers le contrôleur de fichier,
- Si le type de source =3, le système demande l'adresse IP du serveur et le port serveur,
- Le système envoie l'adresse et le port du serveur réseau, vers le contrôleur serveur réseau,
- Le système demande au contrôleur de la source de lire l'échantillon,
- Si le type de source =1, le contrôleur de source appel la méthode configurer () de la carte d'acquisition, qui envoie une demande de connexion si elle n'existe pas vers le serveur caractérisé par l'adresse et le port obtenue à l'étape précédente, puis appelle la fonction calcul_limite () pour déterminer les limites d'acquisition en utilisant le gain et la polarité.
- Le système appelle la méthode demarrer_Acquisition () de la carte d'acquisition,
- le système appelle la méthode lire_echanillon () de la carte d'acquisition,
- Si le type de source =2, le système appel la méthode configurer () du fichier, qui appelle la fonction Charger ().
- Le système appelle la méthode demarrer_Acquisition () du fichier,
- le système appelle la méthode lire_echanillon () du fichier,
- Si le type de source =3, le système appel la méthode configurer () du serveur réseau, cette fonction appelle la fonction connecté () si aucune connexion existe avec le serveur demandé.
- Le système appelle la méthode demarrer_Acquisition () du serveur réseau,
- le système appelle la méthode lire_echanillon () du serveur réseau,

4.2.1.2. Le diagramme de séquence : Enregistrement

➢ **La description textuelle**

- Le système demande le chemin d'écriture au contrôleur à l'interface graphique,
- Le système demande la décision d'écraser le fichier s'il existe,
- Le système envoie les paramètres vers le contrôleur d'enregistrement (le chemin du fichier et la décision d'écrasement),
- Le système consulte l'état du fichier dans le contrôleur d'enregistrement,
- Si le fichier est non ouvert, le système demande l'ouverture du fichier au contrôleur d'enregistrement,
- Le contrôleur d'enregistrement envoie les paramètres au fichier pour permettre de l'ouvrir, c'est le chemin, vers le contrôleur du fichier,
- Le contrôleur d'enregistrement, appelle la méthode Verifier_Existance () du fichier pour vérifier est ce que le fichier existe ou non,
- Si le fichier existe et la demande et l'utilisateur décident d'écraser le fichier s'il existe, le fichier doit être supprimé,
- A cette étape, si le fichier n'existe pas, il faut créer un a nouveau, c'est l'invocation de la méthode Créer () du fichier par le contrôleur d'enregistrement,
- Le contrôleur d'enregistrement invoque la méthode Charger () du fichier pour permettre son chargement,
- Le contrôleur d'enregistrement invoque la méthode Verrouiller () du fichier pour permettre son verrouillage,
- Pour chaque échantillon à sauvegarder , Le système invoque la méthode Convertir_En_Bloc () du contrôleur d'enregistrement, pour la conversion en tableau des caractères pour qu'il puisse le sauvegarder, la tache du sauvegarde est effectuée grâce à la fonction Ecrir_Bloc () du contrôleur d'enregistrement,
- Si l'utilisateur décide de fermer l'application, le système envoie le message de fermeture du fichier au contrôleur d'enregistrement qui envoie les messages sauvegarder () et Déverrouiller () vers le contrôleur du fichier pour libérer les ressources sur le disque,

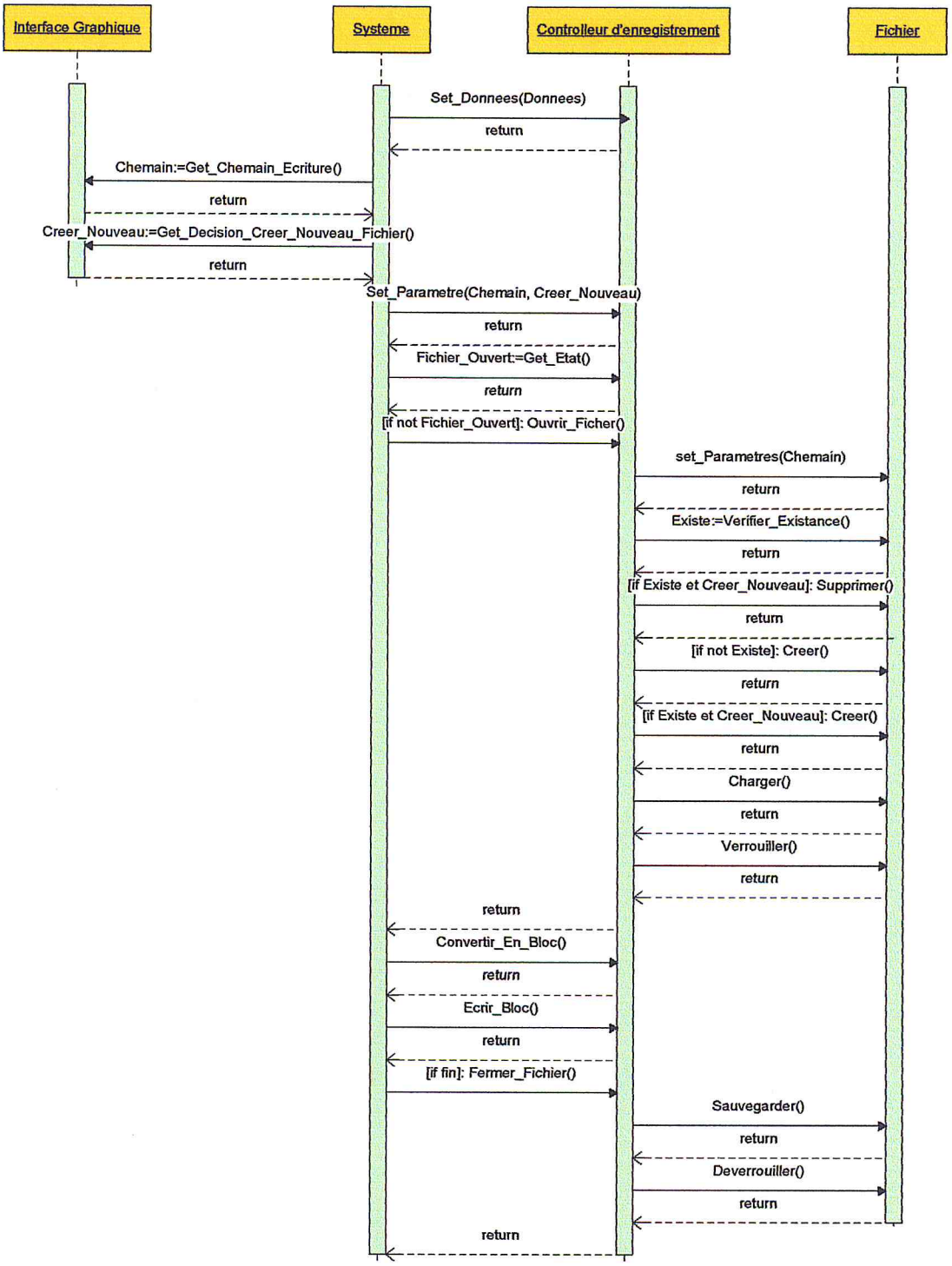
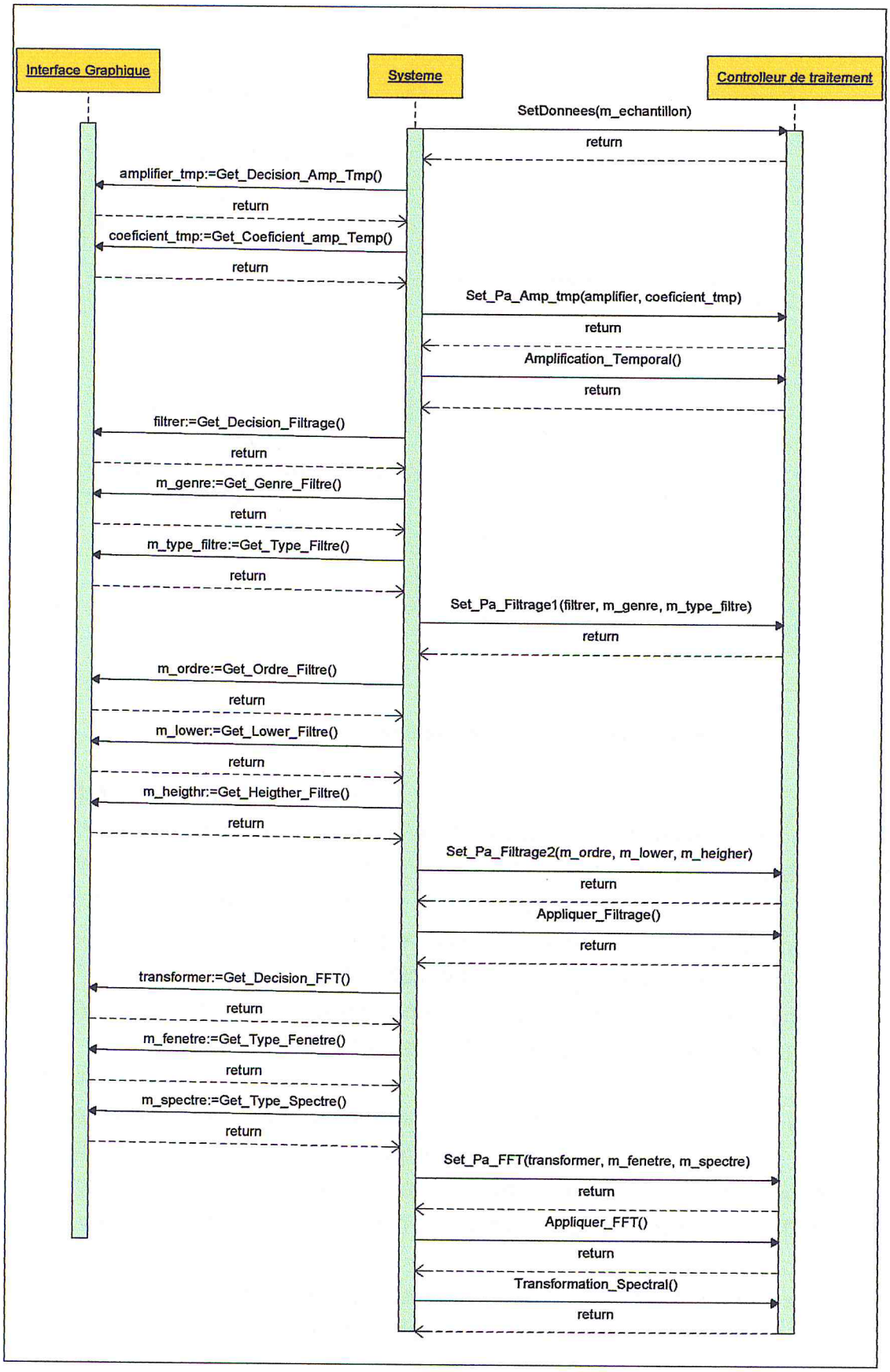


Figure III- 25:Le diagramme de séquence : Enregistrement

4.2.1.3. Le diagramme de séquence : Traitement



4.2.2.1. Les diagrammes de séquences : mise en service réseau

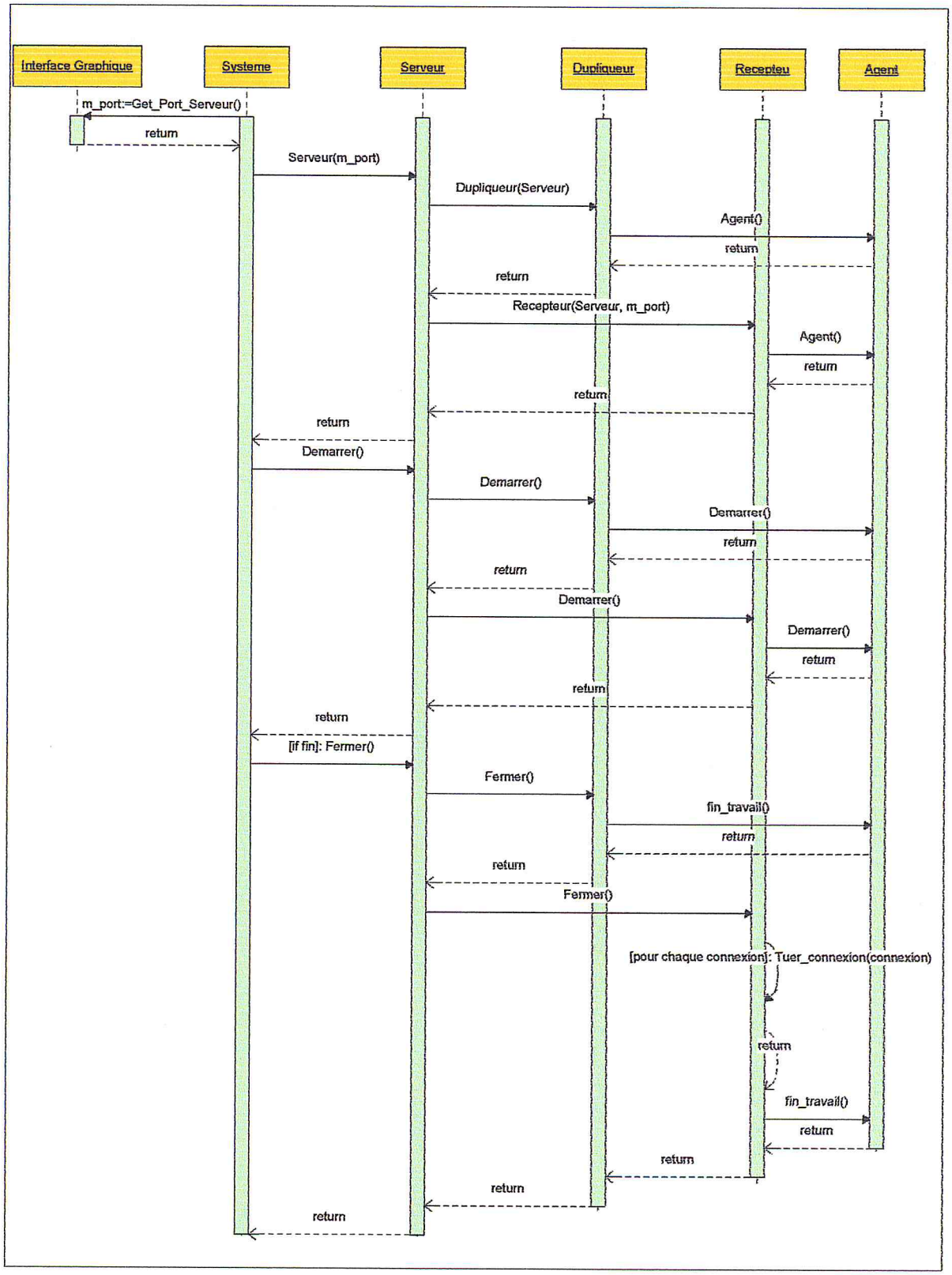


Figure III- 27:Le diagramme de séquences : mise en service

➢ **La description textuelle**

- Le système récupère le port TCP/IP qui est réservé à ce service à partir le contrôleur de l'interface graphique,
- le système crée une instance de la classe serveur qui va créer une instance de la classe dupliqueur et récepteur, ce dernier, chacun pendant sa création instancier la classe agent pour gérer sa tache,
- Le système lance l'exécution de la tache du serveur en parallèle avec le programme principale, ce dernier lance la tâche de dupliquer et de récepteur en parallèle aussi avec sa tache,
- Une fois l'utilisateur décide de fermer l'application, le système détruit l'objet serveur et par conséquence la destruction de récepteur et dupliqueur et ses agents,

4.2.2.2. Le diagramme de séquence négociation

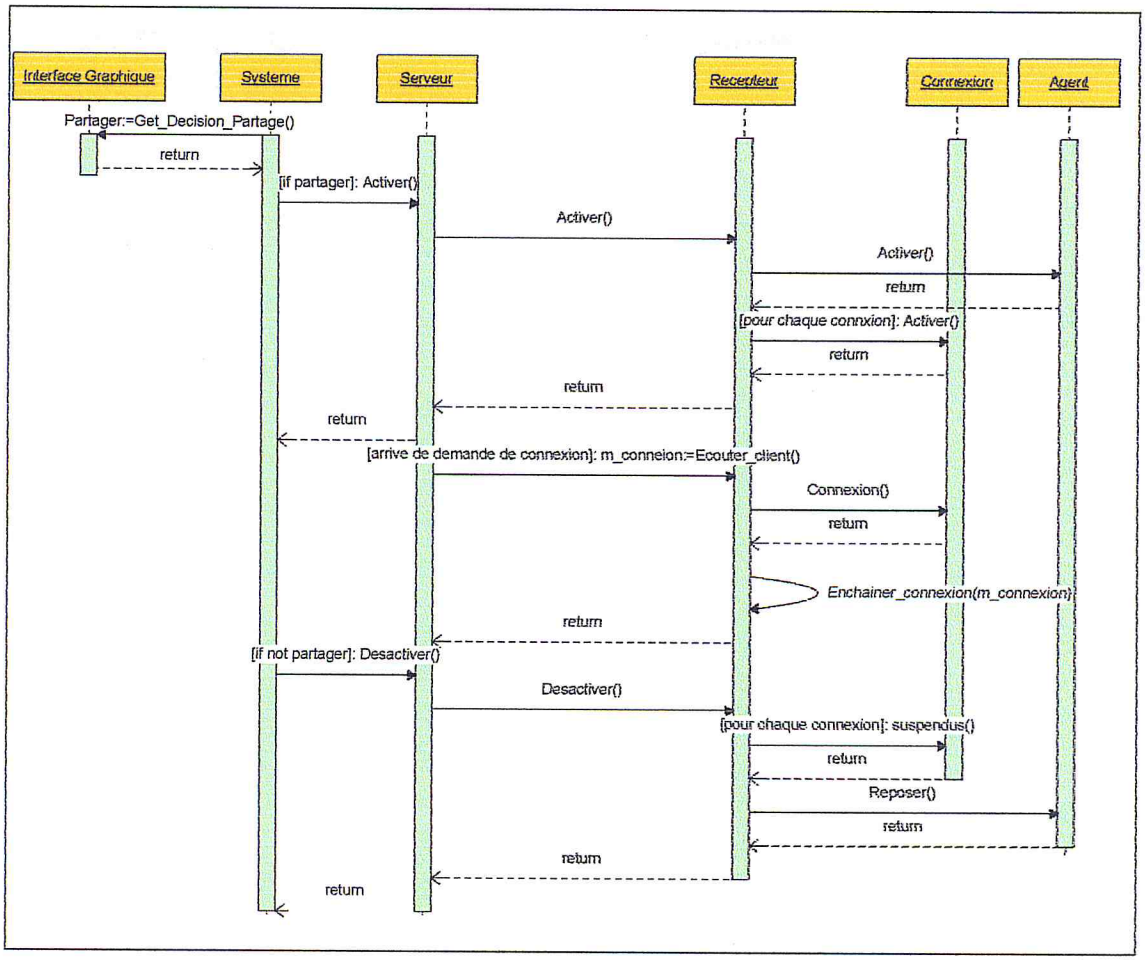


Figure III- 28: Le diagramme de séquence négociation

➢ **La description textuelle**

- Le système consulte la décision de l'utilisateur sur le partage, qui se trouve dans l'interface graphique,
- S'il y a une décision de partage, le système active le serveur, qui active par conséquence son récepteur et toutes les connexions existantes. Et chacun active son agent,
- Le récepteur reste en écoute sur le port du serveur les demande des clients, une fois une demande est arrivé ,le récepteur crée une connexion à travers l'invocation de la méthode connexion() de connexion, et l'enchaîne dans la liste des connexions du serveur
- Si l'utilisateur décide de désactiver le service réseau, le système désactive tout d'abord le serveur qui désactive aussi le récepteur et leurs connexions, qui désactivent de leur part l'ensemble de ses agents,

4.2.2.3. Le diagramme de séquence Envoie des données

➢ **La description textuelle**

- Le système envoie la donnée au serveur,
- Si le service est actif, le serveur empile la donnée,
- Si le dupliqueur est désactive, le serveur active le dupliqueur, qui active son agent,
- Pour chaque connexion, si elle est désactive, le dupliqueur doit l'activer,
- Le rôle principale du dupliqueur est de dépiler les messages arrivant du serveur par l'invocation de la méthode dépiler (), et les dupliquer en plusieurs copie chacune pour une connexion, et l'empiler dans la liste des connexions, une foie le dupliqueur veut dépiler les messages et la liste des messages est vide, le dupliquer lui même.
- Une connexion s'occupe par l'envoie des messages, qui sont empilés dans sa file d'attente, une fois cette file est vide, elle désactive elle même.

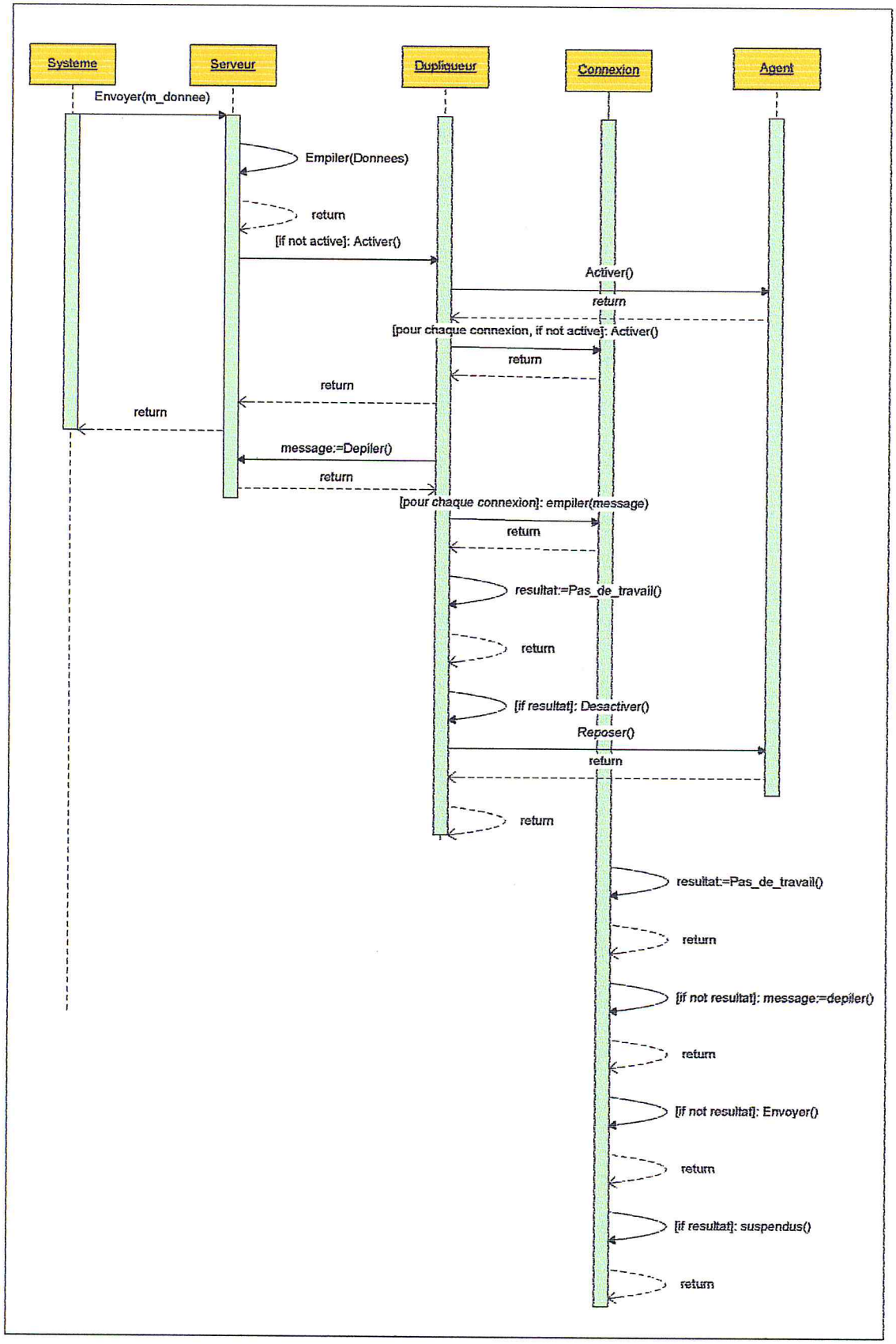


Figure III- 29:Le diagramme de séquence Envoie des données

5. Implémentation et test

A ce niveau, il s'agit d'implémenter la solution retenue au niveau de la phase de conception, c'est la phase au cours de laquelle les structures et les algorithmes définis pendant la conception sont traduits dans un langage de programmation. Le test permet de réaliser des contrôles pour la qualité du système. Dans cette phase nous procédons par présenter les différentes interfaces de notre logiciel pour réaliser les cas d'utilisations qui sont présentés dans l'étape de spécification des besoins et de faire le test par simulation.

5.1. Les langages utilisés

5.1.1. Le logiciel LABVIEW

➢ **Qu'est ce que le LABVIEW:**

LABVIEW est un environnement de développement de programme, tout comme les environnements de développement BASIC ou C modernes, ainsi que LABWindows/CVI de National Instruments. LABVIEW diffère toutefois de ces applications sur un point important. En effet, alors que les autres systèmes de programmation emploient des langages *textuels* pour
Créer des lignes de code, LABVIEW utilise un langage de programmation *graphique*, le *G*, pour créer des programmes sous la forme de diagrammes.
LABVIEW, comme le langage C ou BASIC, est un système de programmation à usage général, avec des bibliothèques de fonctions étendues convenant à toute tâche de programmation. LABVIEW comprend des bibliothèques pour l'acquisition de données, le contrôle d'instrument série et GPIB, ainsi que pour l'analyse, la présentation et le stockage de
Données. LABVIEW comprend également des outils d'élaboration de programme traditionnels, de manière à pouvoir définir des points d'arrêt, animer l'exécution pour visualiser le transfert des données dans le programme, et exécuter pas à pas le programme pour faciliter la mise au point et le développement de programme.

➢ **Comment fonctionne le LABVIEW:**

Bien que LABVIEW soit un système de programmation à usage général, il comporte également des bibliothèques de fonctions et des outils de développement conçus spécifiquement pour l'acquisition de données et le contrôle d'instruments. Les programmes LABVIEW sont appelés *VIs* (pour *Virtual Instruments – Instruments virtuels)* en raison de leur apparence et de

leur fonctionnement pouvant imiter ceux d'instruments réels. Ils sont cependant identiques aux fonctions des langages de programmation conventionnels.

Un VI comprend une interface utilisateur interactive (un diagramme de flux de données équivalent au code source) et des connexions entre icônes permettant au VI d'être appelé par des VIs d'un niveau supérieur. Plus précisément, les VIs sont structurés comme suit :

• L'interface utilisateur interactive d'un VI est appelée *face-avant*, en raison de sa ressemblance avec la face-avant d'un instrument réel. La face-avant peut contenir des boutons rotatifs, des boutons-poussoirs, des graphiques et d'autres commandes et indicateurs. Vous pouvez entrer vos données à l'aide d'une souris et d'un clavier, et voir les résultats sur l'écran de l'ordinateur.

• Le VI reçoit des instructions d'un *diagramme*, que vous construisez en G. Le diagramme est une solution graphique à un problème de programmation. Le diagramme est également le code source du VI.

• Les VIs suivent un format modulaire hiérarchique. Vous pouvez les utiliser comme programmes principaux ou comme sous-programmes au sein d'autres programmes. Un VI utilisé à l'intérieur d'un autre VI est appelé un *sous-VI*. L'*icône* et le *connecteur* d'un VI fonctionnent comme une liste graphique de paramètres permettant aux autres VIs de
Transmettre des données à un sous-VI.

C'est grâce à ces fonctions que LABVIEW tire la meilleure partie du concept de *programmation modulaire*. Vous divisez une application en une série de tâches, que vous divisez à nouveau jusqu'à ce qu'une application compliquée devienne une série de simples sous-tâches. Vous construisez un VI pour accomplir chaque sous-tâche et combinez ces VIs dans un autre diagramme pour accomplir une tâche plus importante. A la fin, votre VI principal contient une série de sous-VIs représentant les fonctions d'une application.

Chaque sous-VI pouvant être exécuté de façon autonome, indépendamment du reste de l'application, la mise au point en est d'autant simplifiée. En outre, comme plusieurs sous-VIs secondaires réalisent souvent des tâches communes à différentes applications, vous pouvez ainsi développer un ensemble spécialisé de sous-VIs correspondant exactement aux applications que vous serez amené à développer.

5.1.2. Le langage Visual C#

C# (prononcé « C sharp ») est un langage de programmation simple, moderne, orienté objet et de type sécurisé. Il sera immédiatement familier aux programmeurs C et C++. C# combine la productivité élevée de langages de

développement rapide d'application (RAD, *Rapid Application Development*) et la puissance brute du C++.

Visual C# .NET est l'outil de développement C# de Microsoft. Il comprend un environnement de développement interactif, des concepteurs visuels pour générer des applications Web et Windows, un compilateur et un débogueur. Visual C# .NET fait partie d'une suite de produits, appelée Visual Studio .NET, qui intègre également Visual Basic .NET, Visual C++ .NET et le langage de script JScript. Tous ces langages permettent d'accéder au Microsoft .NET Framework qui inclut un moteur d'exécution commun et une bibliothèque de classes très complète.

Le C# est un nouveau langage, il dispose d'un accès complet aux mêmes bibliothèques de classes très complètes qui sont utilisées par les outils éprouvés, tels que Visual Basic .NET et Visual C++ .NET.

5.2. L'interface du moniteur EEG

L'interface de moniteur EEG est présentée dans la figure suivante

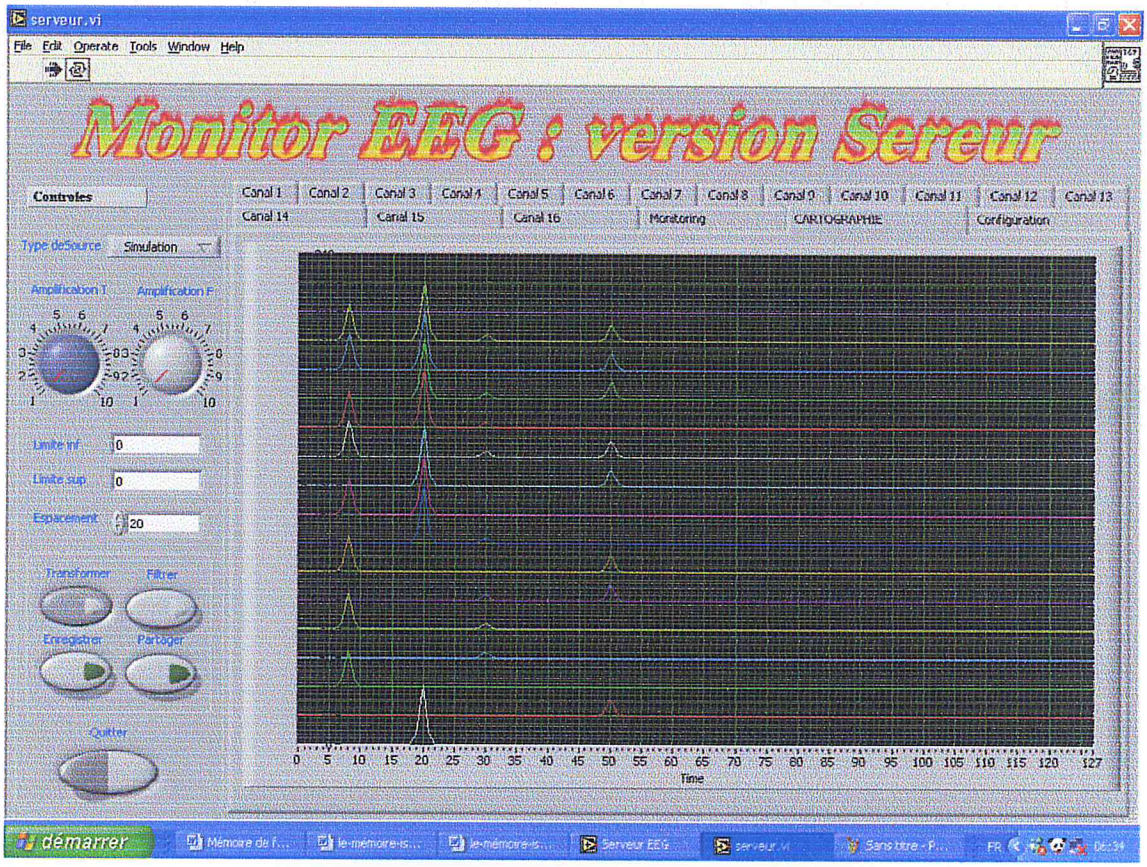


Figure III- 30 : l'intrface du moniteur EEG.

Cette interface comprend :

5.2.1. un menu

Le menu principale du moniteur EEG se trouve en haut de l'interface, elle contient les différentes commandes qu'on peut appliquer : Le menu Fichier , Edit ,Operate ,Tools, Window et Help,dans l'interface suivante nous présentons les deux menus :File et Operate.

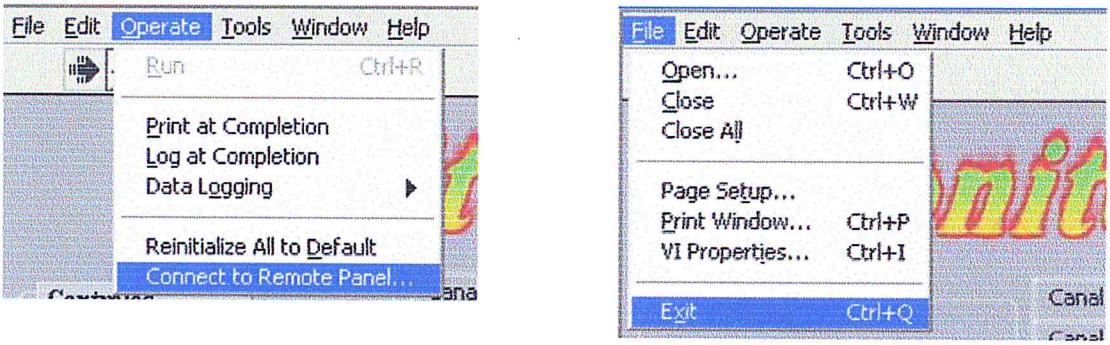


Figure III- 31:les deux menus:File et Operate.

5.2.2. Une palette de contrôle

La palette de contrôle se situe dans la partie gauche de l'interface, elle comprend l'ensemble des fonctionnalités du moniteur tels que : les transformations entre l'amplification temporelle et fréquentielle, le filtrage l'enregistrement et le partage .

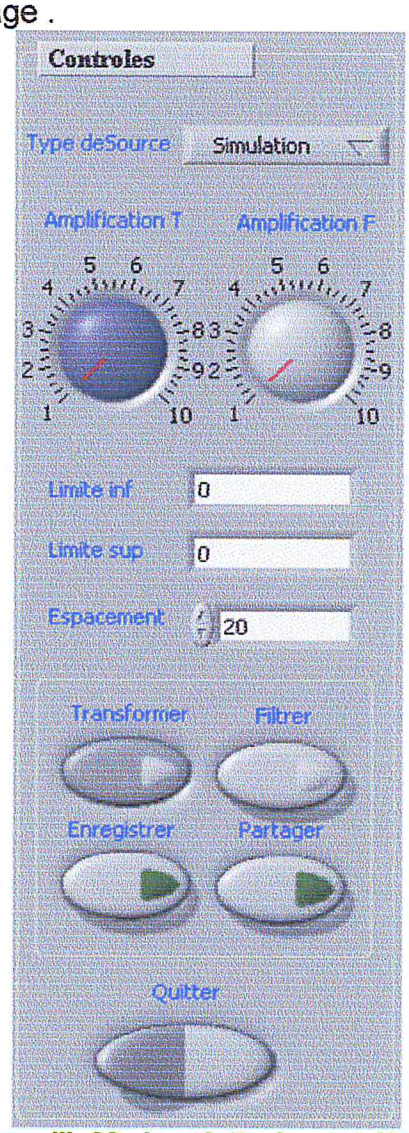


Figure III- 32 : la palette de contrôle

Cette palette comprend aussi :

- Les paramètres : borne inf et borne sup qui permettrent de concentrer seulement sur une partie du signal.
- Le type de sources : qui est choisit parmi : lecture , simulation et carte d'acquisition, qui est présenté dans la figure suivante :

Figure III- 33: le choix du type de source.

- Deux règles de graduations de 1 jusqu'au 10 sous forme circulaire, qui permettent de multiplier le signal (en amplitude ou en fréquence) suivant la valeur choisit par l'utilisateur.
- Les boutons : Transformer , Filtrer ,Enregistrer et partager , et à la fin le bouton Quitter.

5.2.3. Un panneau d'affichage

Est un espace d'affichage pour les signaux qui comprend 19 pages :
Les 16 premières pages sont réservées pour les signaux de 16 canaux, chaque page pour un canal.
Les deux autres pages suivantes sont destiné au type d'affichage :Monitoring et Cartographie, et la dernière page ,est pour les paramètres de configuration. Le panneau d'affichage est illustré dans la figure suivante :

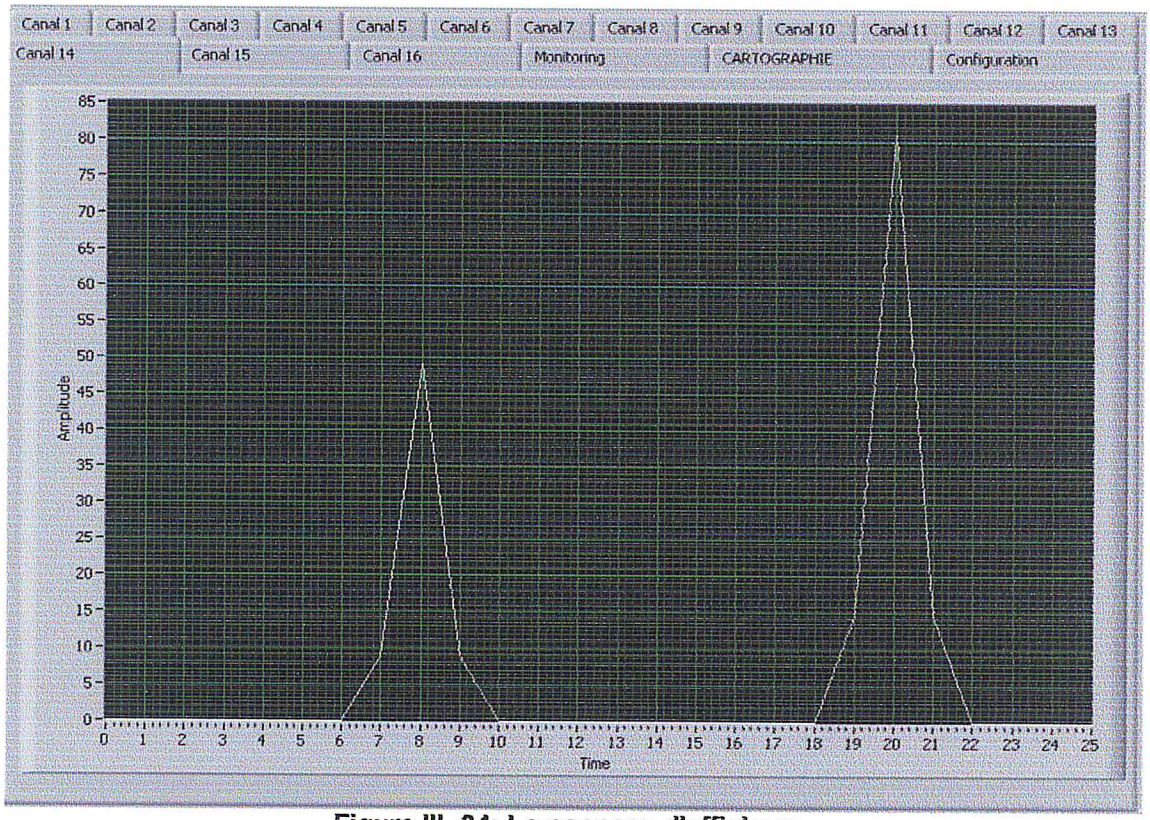


Figure III- 34: Le panneau d'affichage.

5.3. Réalisation des fonctionnalités du moniteur EEG

Premièrement il faut établir les paramètres de configuration qui sont présentées dans l'interface suivante :

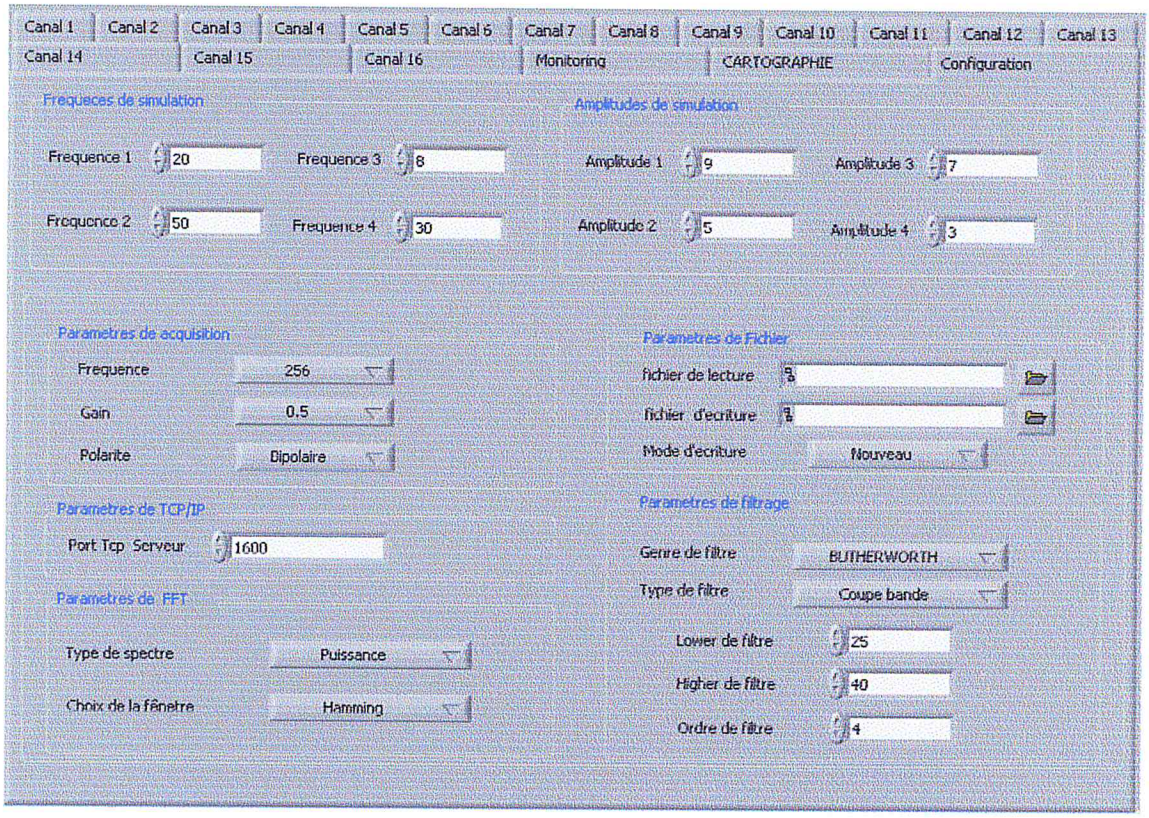


Figure III- 35: Les paramètres de configuration.

L'ensemble des paramètres de configuration est :

- **Les paramètres de fréquences :** il s'agit des quatre premiers fréquences pour les quartes premiers canaux, les 6 canaux suivant sont à deux signaux(canal 5,6,7,8,9 et 10) , les quatre qui se suit sont à 3 signaux(canal 11,12,13 et 14), le signal qui se suit à 4 signaux (canal15).
- **Les paramètres d'amplitude :** il s'agit des quatre premiers Amplitudes pour les quartes premiers canaux,

Pour les paramètres de fréquence et d'amplitude qui sont présentés dans la figure suivante :

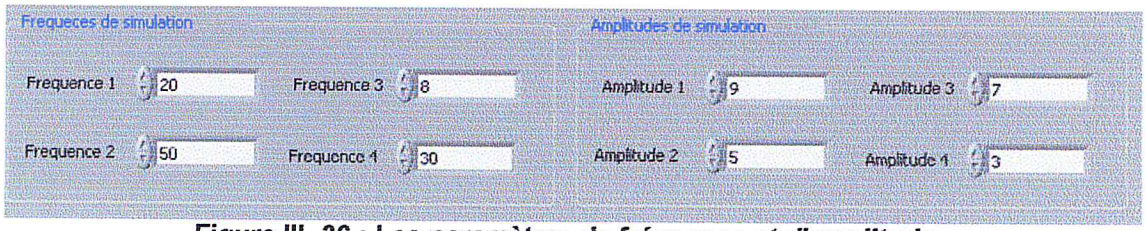


Figure III- 36 : Les paramètres de fréquence et d'amplitude .

On trouve les signaux des canaux 1,2,3,4,5,11,15 (pour les fréquences)

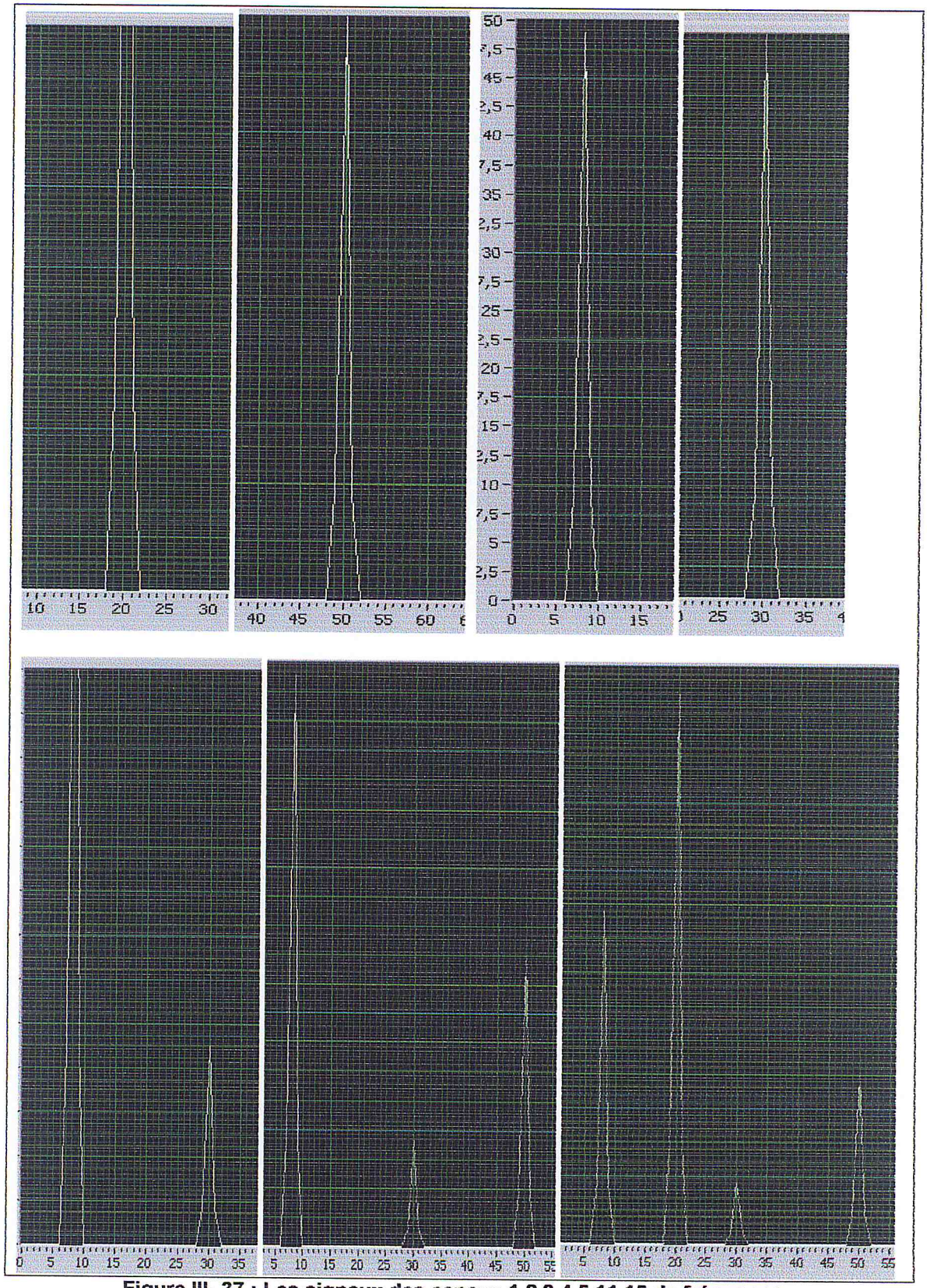


Figure III- 37 : Les signaux des canaux 1,2,3,4,5,11,15 de fréquence .

Et par rapport les amplitudes on trouve les signaux suivant pour les canaux 1,2,3 5 :

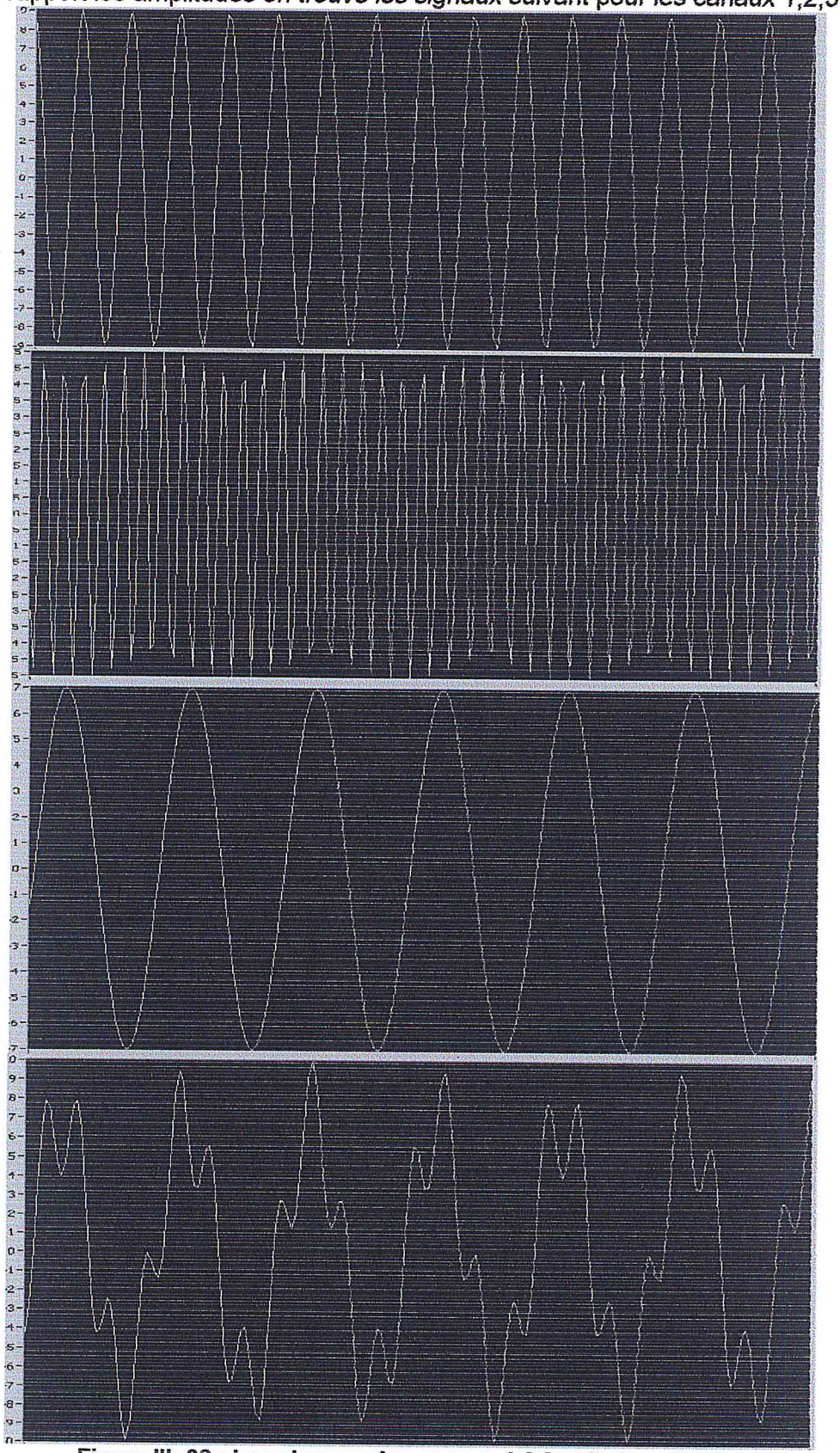


Figure III- 38 : Les signaux des canaux 1,2,3 et 5 d'amplitude.

- **Les paramètres d'acquisition :** la fréquence, le gain et la polarité chaque paramètre sera choisit parmi les valeurs possibles comme il apparut dans la figure suivante

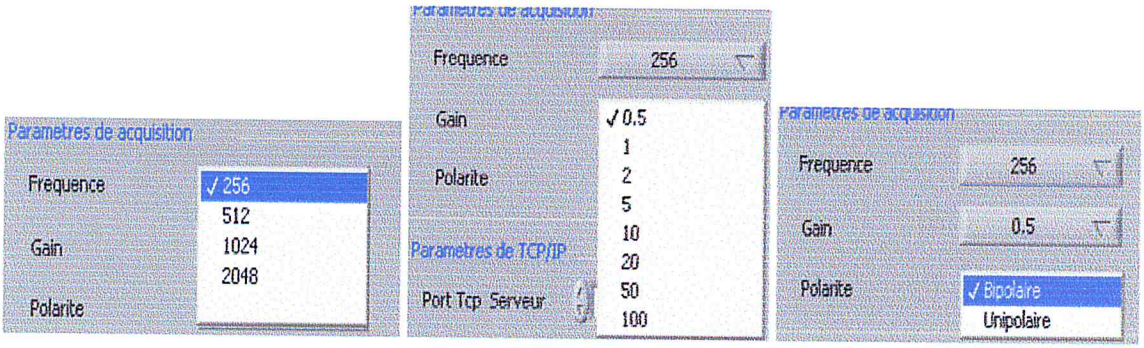


Figure III- 39 : Les paramètres d'acquisition

- **Les paramètres de fichier** : soit un fichier de lecture ou d'écriture et le mode d'écriture
- **Les paramètres TCP/IP** : il concerne le port TCP serveur.
- **Les paramètres filtrage** : il s'agit du genre, le type, le Lower , le higher et l'ordre du filtre.
- **Les paramètres FFT** : type de spectre (amplitude ou puissance) et le choix de fenêtre qui sera parmi les valeurs possibles présentées dans la figure suivante

Figure III- 40 : les valeurs possibles du choix de la fenêtre.

5.3.1. Transformation de Fourier

On peut basculer entre la fréquence et l'amplitude à travers le bouton Transformer qui se situe dans la barre de contrôle, dans la figure suivante un signal avant et après la transformation.

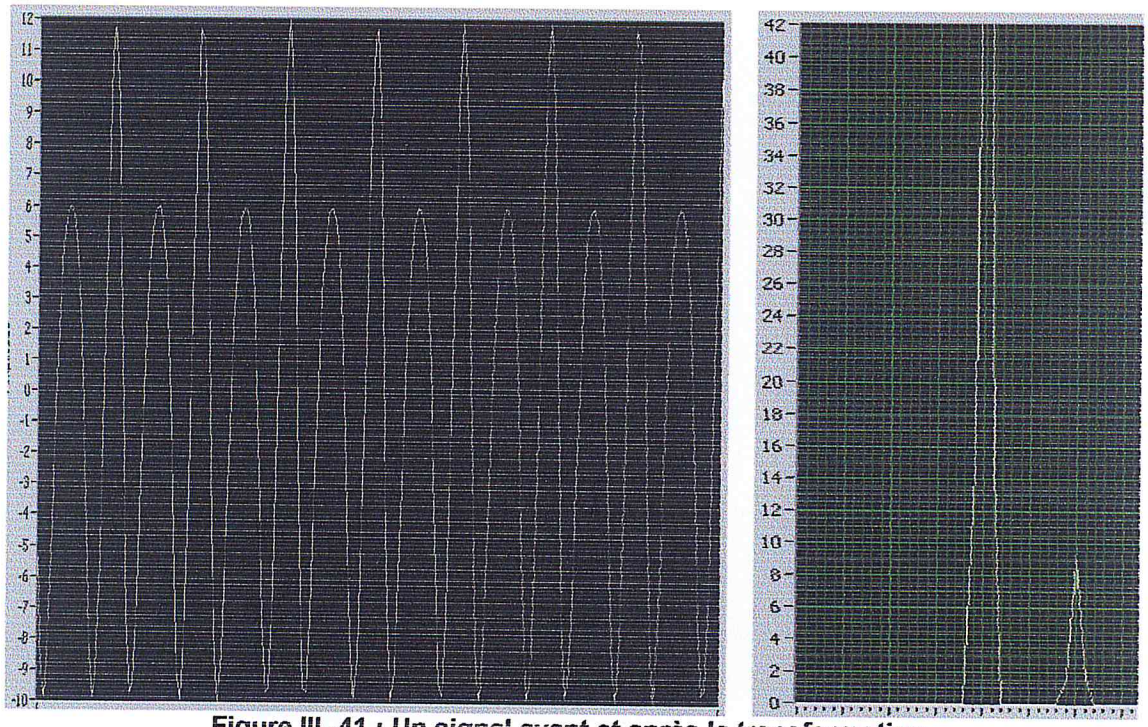


Figure III- 41 : Un signal avant et après la transformation.

5.3.2. Le filtrage

Dans la figure suivante, on représente premièrement le signal puis on lui applique le filtrage, on choisit le type de filtre coupe bande entre 25et40 ceci permet d'éliminer cette partie.

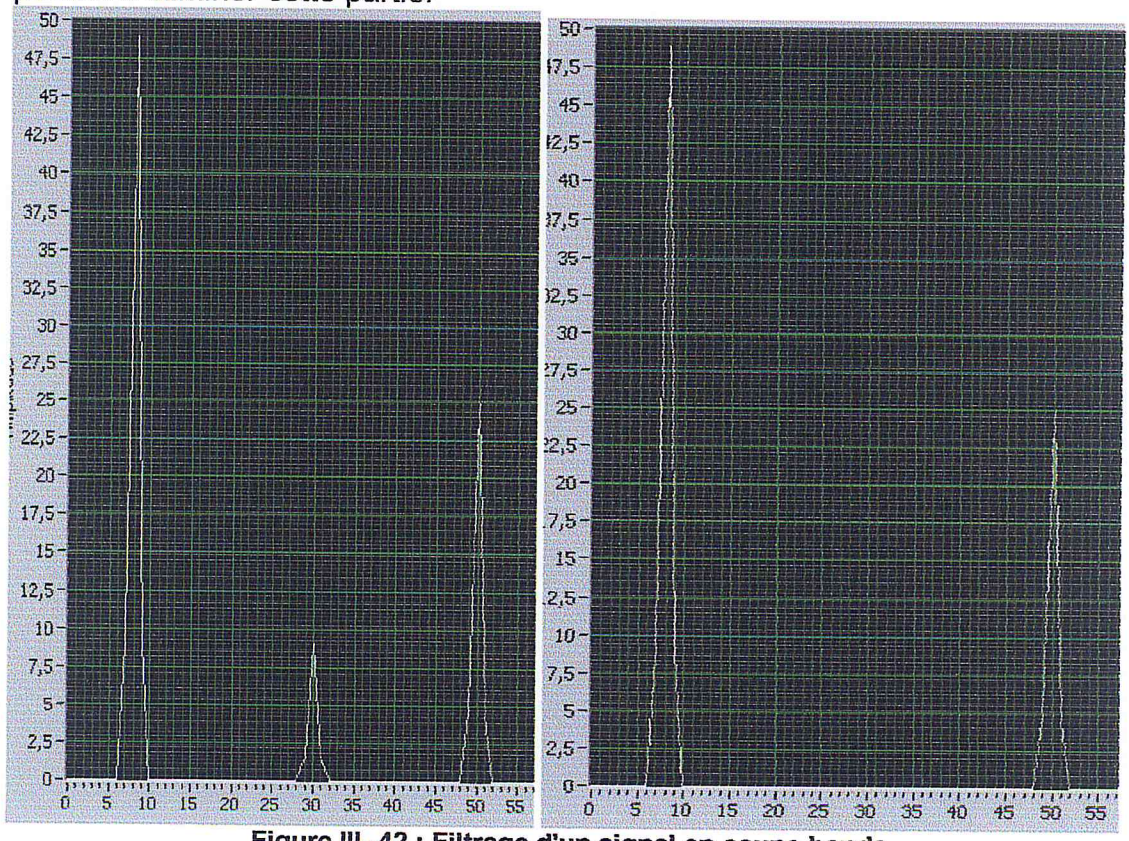


Figure III- 42 : Filtrage d'un signal en coupe bande.

5.3.3. Visualisation

➢ **affichage monitoring** : il permet d'afficher deux signaux sur l'écran, pour permettre leur comparaison, chacun est supposé comme un affichage individuel, c-a-d il a ses propres paramètres, comme il est présenté dans la figure :

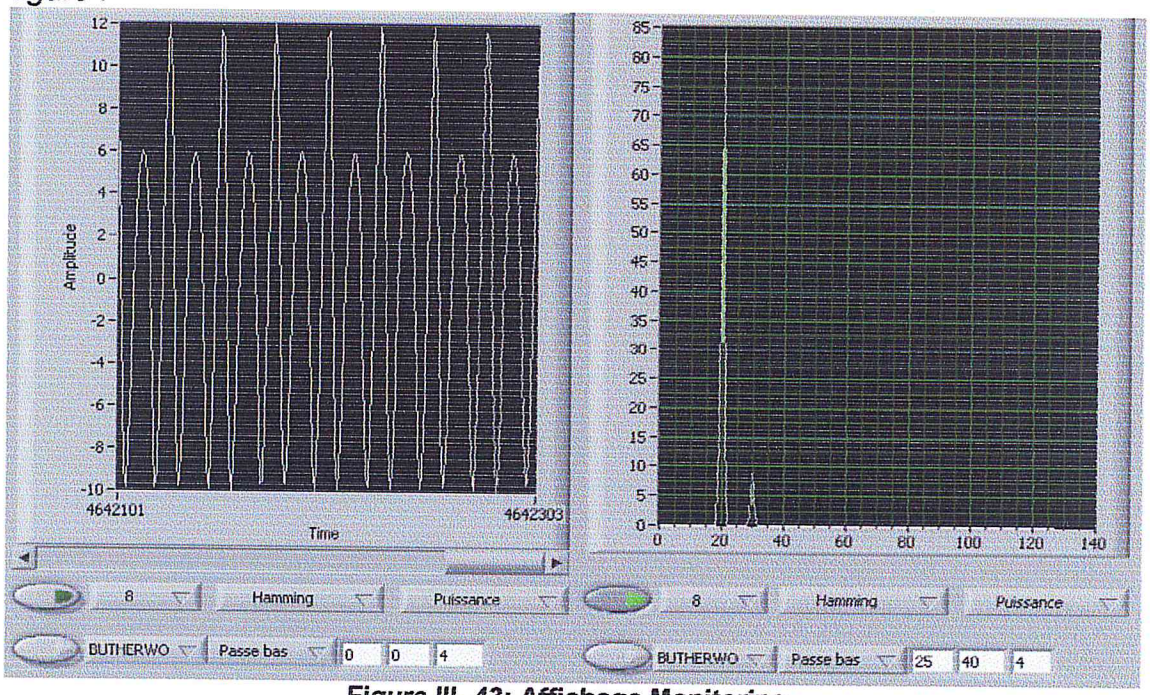


Figure III- 43: Affichage Monitoring.

➢ **affichage cartographie** : il permet d'afficher tous les signaux acquis dans un même graphe, il faut entrer l'espacement entre les signaux.

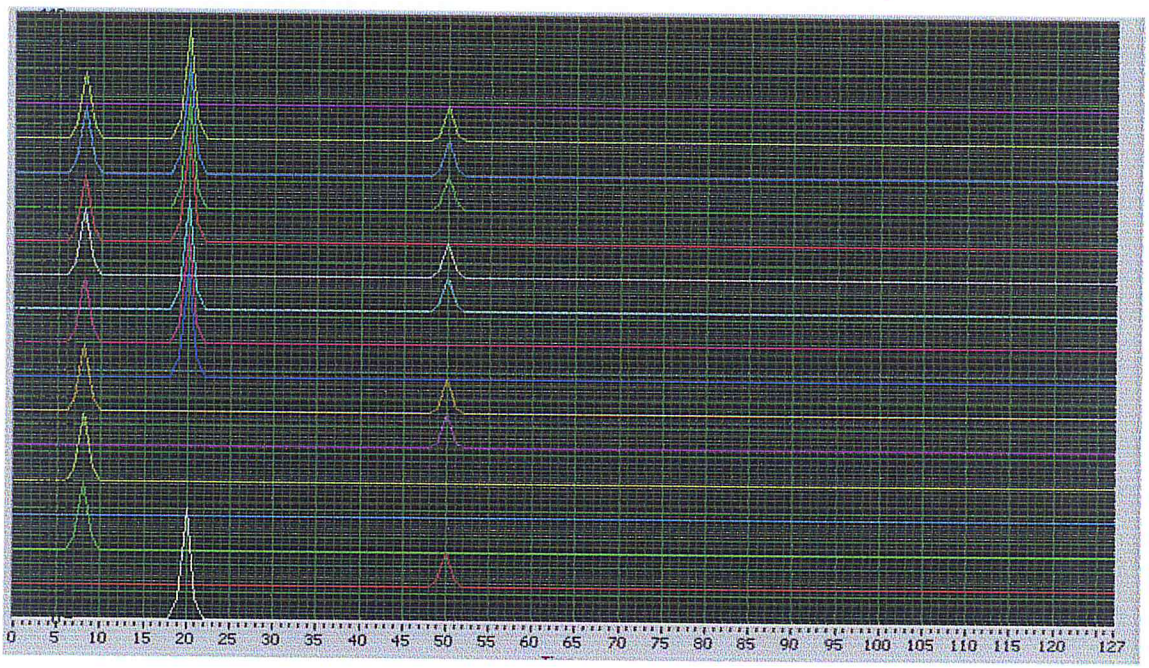


Figure III- 44:Affichage Cartographie.

5.3.4. L'enregistrement

L'enregistrement est effectué par notre moniteur, après une vérification du fichier : si le fichier n'existe pas le système va créer un nouveau fichier, et dans le cas ou le fichier existe, soit il l'écrasse ou il l'ouvrir, selon le choix de l'utilisateur,

5.3.5. Le partage

Un clic sur le bouton Partager de la palette de contrôle permet de publier l'application sur les client à travers le réseau suivant les procédures décrites dans la phase de spécification des besoins.

6. Conclusion

Nous avons présenté, au cours de ce chapitre, les différentes étapes de développement de notre logiciel pour l'acquisition et l'exploitation des signaux EEG. En utilisant une démarche orienté objet qui est présenté par le modèle en cascade et la notation UML , qui représente le langage de modélisation utilisé .

Pour développer notre logiciel nous avons commencé par la spécification des besoins, qui décrit les fonctionnalités du système présentées par les diagrammes des cas d'utilisation et les diagrammes d'activités, puis nous avons entamé l'étape d'analyse pour déterminer les éléments intervenant dans le système .

La troisième étape est la phase la plus importante dans le développement du logiciel, du point qu'elle présente l'aspect dynamique du système c-a-d comment les éléments interagit, ce sont les diagrammes de séquences qui représentent la communication et les comportements des les objets du système. Et à la fin une présentation de l'interface de notre logiciel est établie da,s la phase de l'implémentation et test.

Conclusion générale

L'objectif primordial de ce mémoire consiste à concevoir et implémenter un logiciel pour l'acquisition et l'exploitation des signaux EEG et de développer une bibliothèque pour permettre le partage de ces signaux dans le réseau.

La présence d'un processus de développement formalisé, bien défini et bien gérer est un facteur de réussite d'un projet [Muller 01]. Pour cette raison nous avons suivis le modèle en cascade, utilisant UML comme un langage de modélisation.

Au cours de notre projet nous avons développé un logiciel « moniteur EEG », entièrement dédié aux médecins ou à des utilisateurs qui veulent visualiser des fréquences contenues dans un signal EEG, ce signal est capturé par une carte d'acquisition.

Le système ainsi développé, offre donc de nombreux avantages aux utilisateurs suivant les leurs demandes :

- Le traitement des données s'il veut traiter des données, le système doit les traiter,
- La visualisation de l'échantillon acquis, ou de l'échantillon acquis et traité,
- L'enregistrement des données,
- Le partage des données a travers la publication.

Bibliographie

[Bres, 93] J.bres, ateliers de génie logiciel, masson, 1993.

[Muller, 97] Pierre-Alain Muller, modélisation objet avec UML, EYROLLES, 1997.

[Muller, 01] Pierre-Alain Muller, modélisation objet avec UML, EYROLLES, 2001.

[Craig, 02] Hunt Craig, TCP/IP Administration de réseau O' REILLY, 2002

[Karanjits, 2002] Siyan Karanjits, TCP/IP CAMPUS PRESS, 2002

[Gérard, 2002] Leblanc Gérard, C# et .Net EYROLLES, 2002

[Tom, 2002] Penders Tom, Introduction à UML OEM, 2002

[Bekka, 1998] Rais El'hadi Bekka, Fondements du traitement de signal 1998

[Williams, 2002] Manuel de référence Microsoft visual C# .net Dunod, 2002

[Cottet, 2002] Francis Cottet, Labview Programmation et applications Dunod, 2002

[BLANCHET, 2000] Gérard BLANCHET, Traitement numérique du signal 2000

[Mike, 2000] Busby Mike, Introduction à TCP/IP OEM, 2000