Contrôle d'ordonnancement dans un système de stimulation électrique distribué

Samy Lafnoune

Rapport de Stage

Master 2 EEA
Spécialité : "Robotique"

Contrôle d'ordonnancement dans un système de stimulation électrique distribué

Préparé au
Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier
Au sein de l'équipe DEMAR commune entre INRIA et LIRMM UMR (CNRS-UM2) N 5506
161, rue Ada
34095, Montpellier

Encadré par Daniel SIMON et David ANDREU

Soutenu par Samy LAFNOUNE

Août 2014

Table des matières

Table des figures

Chapitre 1

Introduction générale

La stimulation électro fonctionnelle (SEF) est une technique de thérapie physique utilisée en médecine de rééducation, dans de nombreuses pathologies musculaires (renforcement musculaire chez le sportif, prévention et/ou traitement de la fonte musculaire (atrophie) de non-utilisation ou par atteinte neurologique). De nombreux travaux ont été réalisés ces 20 dernières années, tant dans la compréhension du fonctionnement du muscle que dans les applications cliniques de la SEF. En neurologie, la SEF a pour but de pallier la déficience de contraction musculaire, quelle qu'en soit l'origine.
Dans le cas particulier d'une lésion de la moelle épinière, tous les muscles ne peuvent pas bénéficier d'une SEF, car tous les muscles ne sont pas stimulables. Les territoires musculaires qui dépendent des étages médullaires marqués par une lésion, présentent un phénomène de dénervation et voient leur typologie musculaire se modifier avec installation progressive d'une importante atrophie.
Les territoires musculaires dépendants des étages médullaires situés sous la lésion ne subissent pas les effets de la dénervation, mais sont simplement déconnectés du contrôle cérébral volontaire. Ces muscles sont stimulables, même si la typologie de leurs fibres est aussi modifiée. Face aux modifications chimiques et mécaniques dans ces muscles dits sous-lésionnels, la SEF a fait la preuve de sa capacité à inverser les modifications chimiques et mécaniques, à réduire l'atrophie et à provoquer une contraction musculaire puissante et fonctionnelle.
Le problème principal auquel se heurte la SEF est celui de la fatigue musculaire engendrée chez le patient et de la capacité du muscle à se contracter longtemps et efficacement [Tou11].

Le présent travail de master a été mené au sein de l'équipe-projet DÉambulation et Mouvement ARtificiel (`http://www.lirmm.fr/demar`/DEMAR), créée en 2004. Cette équipe est hébergée par le LIRMM et regroupe divers partenaires de recherche qui sont l'INRIA, le CNRS, l'Université Montpellier 2 et l'Université Montpellier 1. De plus, l'équipe DEMAR travaille déjà en étroite collaboration avec des partenaires industriels telle que l'entreprise MXM – Neuromedics et des partenaires cliniques, en particulier le centre de rééducation PROPARA. Les principaux axes de recherche complémentaires de ce projet impliquent :

- La modélisation du système sensorimoteur.
- La synthèse et la commande de mouvement. À partir des modèles développés, le but est d'obtenir le contrôle souhaité suivant la déficience ciblée.
- Le développement technologique de neuroprothèses qui sont les systèmes microélectroniques qui permettent d'activer ou d'observer le système neuro-musculaire.

Les travaux de ce stage de master se situent à l'interface entre ces axes, apportant un outil de simulation permettant l'amélioration du procédé de développement de ces technologies et notamment les systèmes de stimulation électrique distribuée.

Chapitre 2

Contexte et problématique

La stimulation électro-fonctionnelle (SEF) est une voie technologique très intéressante. Elle est utilisée généralement dans les cas de rééducation musculaire, restauration de mouvement et contrôle de mouvement involontaire. Au cours de ces travaux nous allons nous intéresser seulement à la SEF externe qui ne nécessite aucune implantation. Cette stimulation utilise des électrodes collées sur la peau qui envoient des charges électriques aux muscles afin de les actionner. Des signaux électromyogramme (EMG), donnent l'image de l'activité myoélectrique, peuvent aussi être observés grâce à ces électrodes. La Stimulation Electro-Fonctionnelle consiste à appliquer, via des électrodes, une série de stimuli électriques à un nerf (neurostimulation) ou directement aux fibres musculaires (myostimulation) afin de produire artificiellement un potentiel d'action induisant une contraction musculaire [Dur00].
Différentes technologies, commercialisées ou encore au stade de prototypes de recherche, sont disponibles pour appliquer la SEF externe. On peut distinguer deux types de technologie : les solutions filaires basées sur un stimulateur offrant de multiples canaux auxquels sont connectées les électrodes, et les solutions sans-fil basées sur les petits stimulateurs plus ou moins autonomes.
La technologie sur laquelle nous allons travailler relève d'une solution sans-fil : il s'agit de la technologie Phenix Liberty commercialisée par la société Vivaltis. Le système PHENIX LIBERTY(cf. figure 2.1) est actuellement composé d'un PC de commande relié à un contrôleur par liaison série, et de PODs communiquent avec le contrôleur par liaison sans fil.

2.1 Description du système Phenix Liberty

Le système actuel est composé de 3 parties : un PC de commande, un contrôleur et des unités distribuées.

2.1.1 Le PC de commande

Le PC de commande sert d'interface utilisateur avec le praticien et gère aussi toutes les applications SEF. Il est relié au contrôleur qui sert de passerelle de communication avec les unités distribuées, appelées PODs.

2.1.2 Le contrôleur

Le contrôleur est une entité technologique (matérielle et logicielle) qui embarque à ce stade la gestion des communications avec les unités reparties (POD). Ce contrôleur est relié au PC de commande et fonctionne comme une passerelle réseau.

La carte contrôleur (cf. figure 2.2) dispose d'un micro-contrôleur Freescale MC1322X muni d'un émetteur-récepteur radio-fréquence (coupleur physique 2.4Ghz) intégrée.

FIGURE 2.1 – Système PHENIX LIBERTY commercialisé par Vivaltis

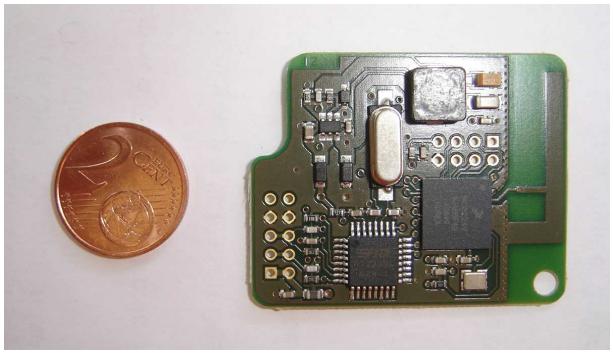


FIGURE 2.2 – Carte contrôleur de l'architecture distribuée

2.1.3 Les unités distribuées

Il existe différents types d'unités distribuées.
Les PODs stim/bio (cf. figure 2.1) permettent de faire l'acquisition de signaux EMG ou de réaliser de la stimulation, ils sont équipés de deux cartes électroniques (cf. figure 2.3) :

— Une carte « générique » qui embarque toute l'électronique nécessaire pour l'alimentation (par batterie), un LED multicouleur comme indicateur lumineux, le bouton marche/arrêt, et le même micro-contrôleur Freescale MC1322X [Fat10] que le contrôleur, intégrant le coupleur physique. Le micro-contrôleur supporte cette fois ci : la pile protocolaire de communication pour les unités déportées (différente de celle du contrôleur), le traitement des requêtes venant du contrôleur et le contrôle général de la carte (surveillance de l'énergie embarquée, contrôle de l'indicateur lumineux, arrêt automatique, etc.).
— Une carte dédiée aux parties métiers (propre à la fonction à réaliser) qui se connecte à la carte générique. Sur cette carte « métier » on retrouve un second micro-contrôleur Renesas R8C27 et l'électronique nécessaire pour réaliser les processus applicatifs de stimulation ou d'acquisition.

Les PODs universels permettent l'ajout d'autres parties métier par exemple de nouveaux capteurs (Centrale inertielle, goniomètre ...)

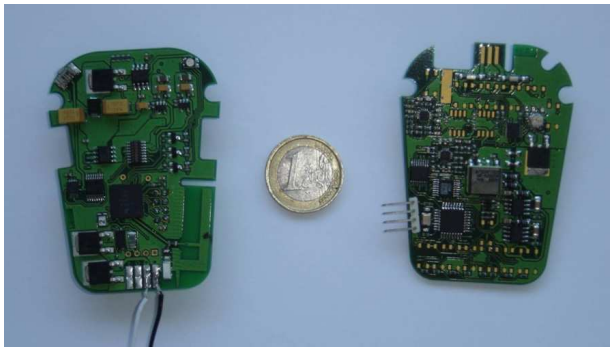


FIGURE 2.3 – Cartes électroniques du POD : à droite la carte générique et à gauche la carte de la partie "stimulation".

2.2 Principe de la SEF

La plus petite charge électrique générant un potentiel d'action est définie comme un stimulus seuil. L'intensité de la stimulation est fonction de la charge électrique totale transférée à la cible, laquelle dépend de la forme, de l'amplitude, de la durée ainsi que de la fréquence du train d'impulsions. La forme d'impulsion la plus simple et la plus typique est le créneau (cf. Figure 2.4). En augmentant l'intensité de la stimulation électrique, il est possible de dépolariser des cellules de diamètre de plus en plus faible et de plus en plus éloigné des électrodes. Ainsi, la modulation de l'intensité de la stimulation délivrée aux muscles permet de contrôler le taux de recrutement des fibres et du coup la force musculaire produite artificiellement. Par conséquent, en jouant sur les paramètres des impulsions de stimulation, les contractions générées peuvent actionner une articulation en relation avec les muscles stimulés. L'angle d'une articulation ou le couple résultant peut être contrôlé au travers la tension mécanique produite dans les muscles fléchisseurs et extenseurs de cette articulation. Généralement on contrôle la largeur d'impulsion ou l'amplitude de la stimulation.

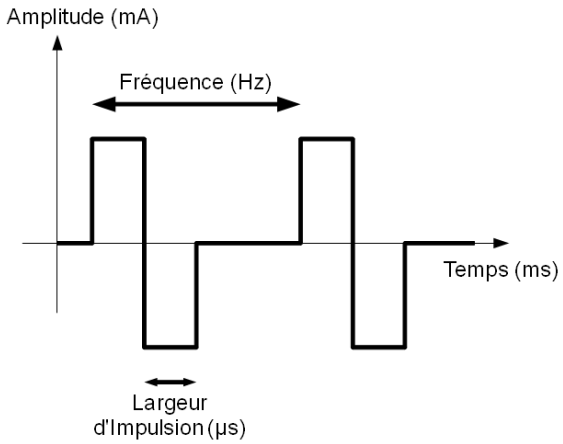


FIGURE 2.4 – Impulsion créneau biphasique.

2.3 La Simulation temps-réel

Pour les systèmes complexes, il est souvent difficile, voire impossible, d'obtenir leurs solutions analytiques. C'est pourquoi la simulation numérique est utilisé à un stade précoce de la conception des

équipements, et d'établissement de systèmes tels que des boucles de commandes et l'analyse des phénomènes dynamiques.
Le but principal de la simulation numérique est de reproduire le comportement du phénomène physique complexe aussi fidèlement que possible et d'obtenir des résultats les plus précis possibles.

Simulation en temps réel implique l'appariement de deux concepts de temps :

— Le temps réel : le temps physique ou la référence de temps du système physique réel que nous voulons modéliser et simuler ;
— Le temps simulé : est le temps écoulé pendant l'exécution de la simulation qui peut être mesuré par l'horloge de l'intégrateur numérique.
— Software in the loop (SIL) : le SIL est une technique de simulation, incluant le code d'exécution qui tournera sur un contrôleur réel avec les mêmes performances que ce dernier, ceci évite d'utiliser des composent réel au sein d'une boucle de commande comme c'est le cas du Hardware in the loop (HIL).
— Hardware in the loop (HIL) : Simulation HIL se compose d'une combinaison de composantes réelles et simulées, ce qui permet d'inclure des paramètres réels dans une simulation par exemple l'intégration de coupleur réseau réel entre différents simulateurs.

2.4 Calcul temps-réel

Les systèmes temps réel sont souvent modélisés par des tâches parallèles en concurrence les unes avec les autres, où chaque tâche est un processus d'une activité de calcul. Le terme en temps réel ne signifie pas vraiment "aussi vite que possible", mais il implique l'existence de contraintes de temps réel qui doivent être respectées. Ces contraintes sont souvent représentées par les délais de tâches.
Le système en temps réel peut être présentée par une architecture simple comme dans la Figure 2.5, où l'interaction entre le système contrôlé et le contrôleur sont déclenchées par des contraintes temps réel.

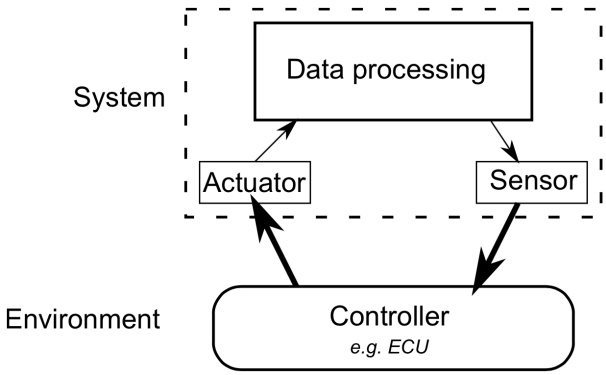


FIGURE 2.5 – Système de contrôle temps-réel

2.4.1 Contrainte temps-réel

Selon le type de contrainte impliquée, à défaut de respecter un délai (dépassements) peut entraîner plusieurs conséquences.

Temps-réel dur : Les contraintes sont considérées comme "dur" quand il est obligatoire que le système satisfasse toutes les échéances. Manquant de telles contraintes implique l'échec du système.

Système strict : Les contraintes critiques se trouvent dans les systèmes critiques embarqués, où l'échec du système conduit à des conséquences dramatiques comme des blessures graves à des personnes, de graves dommages à l'équipement ou des dommages catastrophiques à l'environnement.

Temps-réel souple : Les contraintes sont considérées comme "souple" si le non respect d'une contrainte temps réel dégrade les performances du système, mais sans remettre en cause le comportement souhaité. Par exemple, le streaming vidéo en direct est un système temps réel souple où le temps de chargement peut dépasser le temps réel. Le système ici ne sera pas endommagé en raison de l'excès de temps, mais les attentes de l'utilisateur ne seront pas satisfaites.

Temps-réel ferme (weakly hard) : Un système temps réel ferme peut manquer un certain nombre d'échéance, mais d'une manière prévisible, afin de garantir un niveau de qualité de service. Le manque d'échéance peut se produire, mais il est contrôlé et ses conséquences sont prévisible. La plupart des simulations en temps réel précisent le pourcentage maximal de dépassements que par exemple 1% ou 2% [BKEF14].

2.4.2 Ordonnancement temps-réel

Ordonnancement temps réel définit l'ordre d'exécution de chaque tâche (ou processus) sur le processeur.

Paramètres de tâche temps-réel

Pour effectuer l'ordonnancement, chaque tâche T_i est caractérisée par plusieurs paramètres :

- Le temps de réveil r_i : ou la date d'activation est la date à laquelle une tâche est prête à être lancée.
- Le temps de lancement s_i : c'est l'instant à laquelle une tâche a commencé son exécution.
- Le temps de calcul C_i : est le temps d'exécution d'une tâche sans interruption, il est généralement considéré comme le pire cas de temps d'exécution (WCET).
- Le temps de fin f_i : est la date à laquelle une tâche a terminé son exécution.
- Le temps de réponse R_i : est le temps écoulé entre le moment de réveil et l'heure de fin d'une tâche. Sa valeur est déterminée par la relation suivante $R_i = f_i - r_i$.
- La date limite absolue d_i : C'est l'heure à laquelle une tâche doit être terminée.
- Date limite relative D_i : est la date limite par rapport au temps de réveil. Sa valeur est déterminée par la relation suivante $D_i = d_i - r_i$.

Tous ces paramètres sont illustrés à la Figure 2.6.

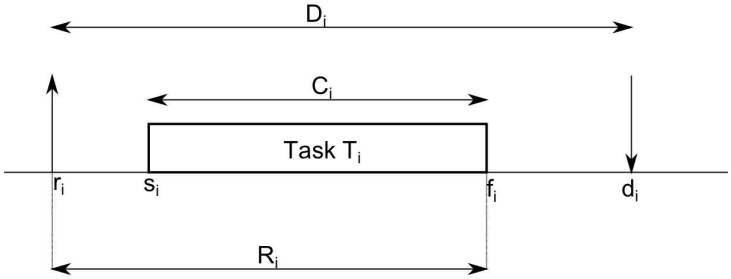


FIGURE 2.6 – Paramètres de tâche temps-réel

Un ordonnancement est dit possible, si toutes les tâches sont exécutées en fonction des contraintes spécifiées. Un ensemble de tâches est ordonnançable si au moins un algorithme de planification est en mesure de produire un ordonnancement possible. Un ordonnanceur peut être soit hors ligne ou en ligne. Cela dépend si les décisions d'ordonnancement sont effectuées avant ou pendant le fonctionnement du système. Un ordonnanceur peut également être soit préemptif ou non préemptif. La préemption consiste à suspendre l'exécution d'une tâche, car une tâche de priorité plus élevée devient prête à être exécutée, puis à reprendre l'exécution sans affecter son comportement. Enfin, un ordonnanceur peut être avec une priorité fixe ou avec une priorité dynamique [BKEF14].

2.5 Problématique

Il est très difficile, voir impossible de réalisé des tests de nouvelles approches dans un système de stimulation électrique, du fait de contraintes matérielles (ré-certification à chaque modification d'un matériel destiné à un usage médical), mais aussi de contraintes éthiques (nécessité d'autorisation au comité de protection de personne (CPP)), c'est pour cela qu'il est impératif d'utiliser des simulateurs dans un stade précoce du développement pour valider les différentes approches avant l'implémentation sur un système réel.
Dans ce Stage nous allons-nous intéresser à la mise en œuvre d'un simulateur temps réel capable de simuler le système de stimulation distribuée décrit précédemment, ce simulateur nous permettra de réaliser et tester de nouvelles fonctions telles que la commande en boucle fermée de membres stimulé et la mise en œuvre d'un ordonnancement régulé pour les tâches exécutées sur le contrôleur.

Chapitre 3

Fonctionnement du système

Ce chapitre décrit le fonctionnement du système courant. Nous allons distinguer deux contextes de fonctionnement, que nous appellerons modes de fonctionnement.

3.1 Cas du mode "connecté"

Le mode "connecté" correspond à l'utilisation actuelle de la technologie pour les clients de Vivaltis, principalement pour la rééducation chez un kinésithérapeute par exemple [Tou11]. Dans ce cas le système est géré par un PC de commande qui est fixe et reliée par liaison série (USB) au contrôleur. Ce dernier sert de passerelle réseau afin de communiquer avec les PODs par liaison sans fil (cf. figure 3.1). Ce mode d'utilisation est dit "connecté", car tous les calculs sont centralisés sur le PC de commande et donc le système n'est pas autonome. Ce mode est celui utilisé actuellement.

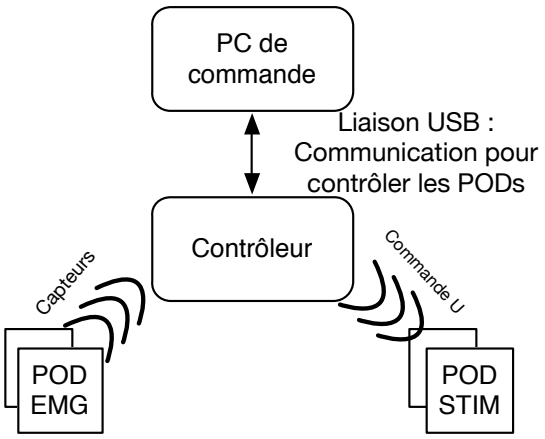


FIGURE 3.1 – Commande mode "connectée" via PC de commande

3.2 Cas du mode "autonome"

Le mode "autonome" est le mode de fonctionnement auquel nous allons nous intéresser, il a été introduit dans la thèse de Toussaint [Tou11], mais jamais implémenté sur le système. Il permet d'utiliser le système en toute autonomie par le patient lui-même (à la maison par exemple), on a plus besoin du PC de commande (sauf pour la configuration). Le contrôleur se chargera alors d'effectuer toutes les tâches de communication et de commande relevant de l'application de SEF considérée. Ce cas ne pourra être utilisé qu'en mode mono patient, le contrôleur sera porté par le patient est sera alimenté par batterie. Il

communiquera avec les différents PODs par liaison sans fil (cf. figure 3.2).

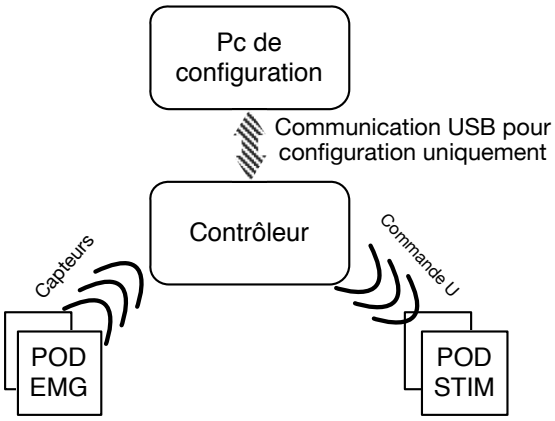


FIGURE 3.2 – Commande mode "autonome" par le contrôleur

3.3 Communication

3.3.1 Le protocole de communication

La communication entre le contrôleur est les PODs est réalisé avec une liaison sans fil IEEE802.15.4 2.4Ghz, le protocole utilisé est le STIMAP (Sliding Time Interval based Medium Access Protocol).
La pile protocolaire choisie (cf, figure 3.3) prend comme référence le modèle OSI réduit à 3 couches (Physique, MAC, Application).

- **La couche physique** est la partie électronique chargée de la conversion entre bits et signaux électriques, électromagnétiques ou optiques, et de la transmission effective de ces signaux entre les interlocuteurs. Son service est généralement limité à l'émission et la réception de données sans en connaître la signification.
- **La couche MAC (medium access control)** doit gérer et arbitrer l'accès au médium, elle ne doit permettre qu'à seule unité de parler à la fois sur le canal de transmission.
- **La couche application** a pour but d'extraire les données d'une requête reçue et de les présenter au processus applicatif visé. De même pour une réponse, voire pour un acquittement, les données sont conditionnées (encapsulées) et transmises à la couche MAC pour être émises sur le médium (par la couche physique).

Le format de trame échangée sur le médium est donné par la figure 3.4. On retrouve en tête du paquet un préambule, un délimiteur de début de trame (Start) et la longueur des données (Long.). Une trame peut contenir un paquet de 125 octets de données et elle se termine par un code de détection d'erreur : Frame Check Sequence (FCS) calculé et ajouté automatiquement par l'interface physique, cette interface offre la possibilité, pour certaines méthodes d'accès au médium, de filtrer automatiquement différents types de trames (donnée, acquittement . . .) en utilisant un champ de contrôle (Ctrl Trame) précisant le type d'une trame. Pour identifier les trames échangées sur un même réseau, on introduit dans les trames transmises un identifiant réseau logique (Id Réseau) commun et unique compris entre « 1 » et « 65535 ».

La communication avec les PODs est basée sur une méthode par élection, les PODs ne peuvent engager une communication avec le contrôleur que si ce dernier leur donne le droit de parole. Il existe deux types de DP, le DPI(Droit de parole individuel) qui permet au contrôleur de communiqué directement avec un POD choisi (ex : lecture d'information capteur . . .) et le DPG (Droit de parole de groupe) qui

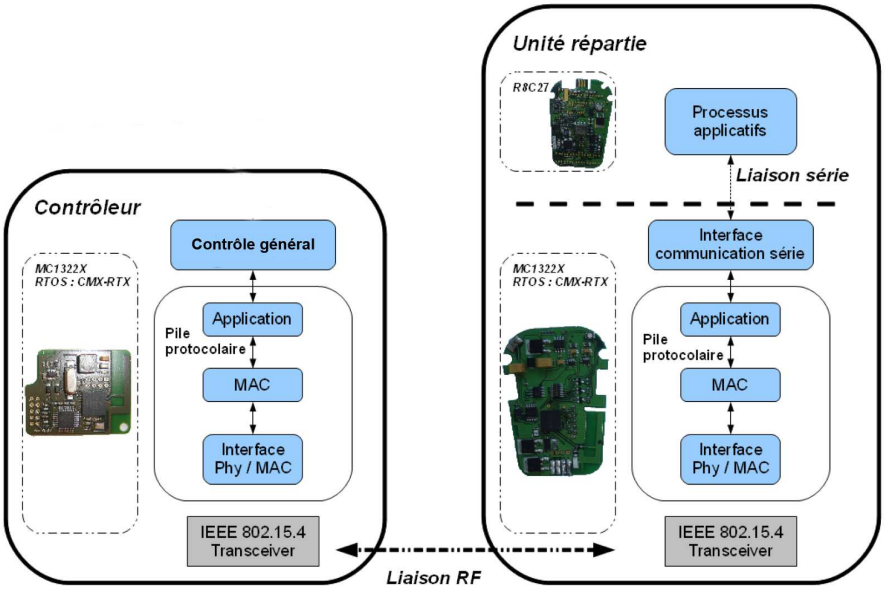


FIGURE 3.3 – Architecture logicielle globale non détaillée et sa projection sur les entités matérielles

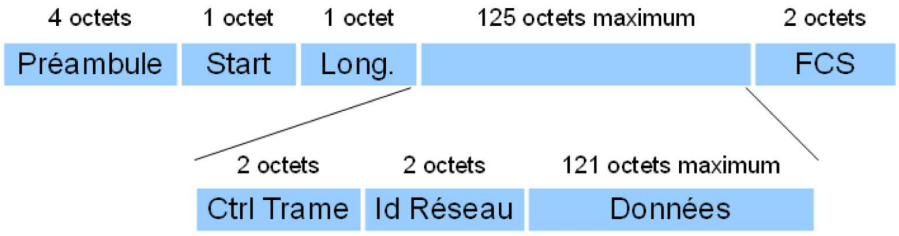


FIGURE 3.4 – Format d'une trame au niveau de la couche physique

permet la communication de plusieurs PODs avec un intervalle de temps glissant basé sur le niveau de priorité.
La réponse d'un POD sur le médium sera donc dépendante de deux choses : l'acquittement, voire le retour d'information, qui est demandé et le droit de parole donné par le contrôleur. Une UR ne pourra émettre une trame que si ces deux éléments du contexte sont réunis.

3.3.2 La gestion du réseau par le contrôleur

Du côté du contrôleur, la gestion de la configuration et de la communication des nœuds sur le réseau est réalisée par la tâche « Gestion du Réseau » (cf. figures 3.5). À partir des requêtes de la couche Application, cette tâche a la charge de créer les trames selon les spécifications du protocole STIMAP et de les transmettre à son interface Physique pour être émise sur le médium. Les requêtes à émettre sont fournies par la couche Application sous la forme de structure de données précisant :

- l'identifiant de l'application de destination (type de la requête)
- l'identifiant de l'application source (émetteur de la requête)
- le type de la requête (configuration de nœud ou envoi de requête)
- l'identifiant du destinataire, soit un nœud ou un groupe de nœuds
- le type d'acquittement
- le type de glissement

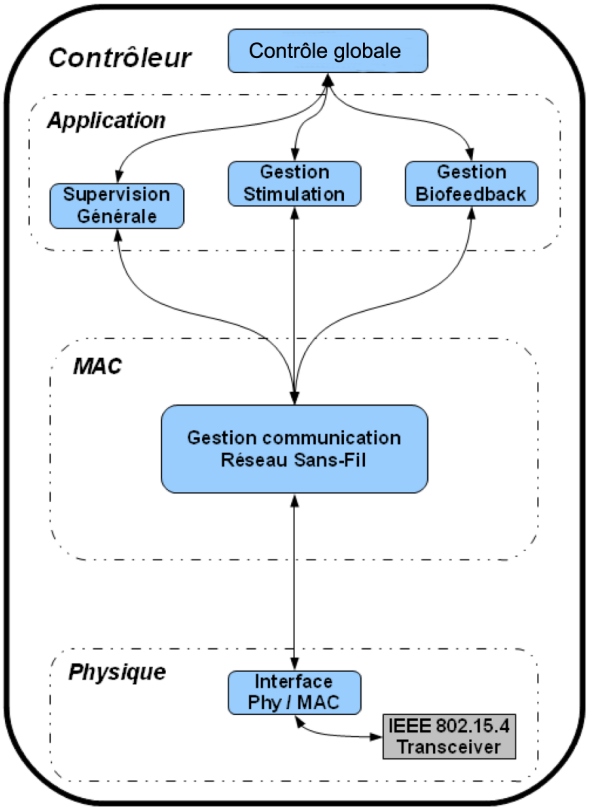


FIGURE 3.5 – Représentation schématique détaillée des modules de la pile protocolaire au niveau du contrôleur

— le timeout, le temps maximal d'attente pour la réponse d'un nœud
— le nombre de réémissions en cas de non-retour d'acquittement
— la taille des données en nombre d'octets
— les données de la requête à émettre

Le traitement d'une requête à émettre commence par le décodage du type de requête. La requête peut concerner une configuration d'un nœud du réseau (association d'une nouvelle unité, abonnement à un groupe) et dans ce cas c'est principalement la tâche Gestion du Réseau qui va devoir traiter la requête et les données à transmettre. Dans l'autre cas, on demande simplement de transmettre une requête et les données sont juste encapsulées dans la trame. Après le traitement du type de la requête, l'entête est créé en fonction du type d'acquittement et éventuellement du type de glissement puis la trame est adressée à une unité ou un groupe d'unité selon l'identifiant du destinataire. Dans notre fonctionnement, afin de sécuriser et d'assurer une fiabilité de nos échanges on choisit de toujours demander un acquittement de réception ou d'exécution de la requête. L'identifiant de l'application de destination sert de nature de la charge. Juste avant d'envoyer la trame, on vérifie qu'aucune autre trame n'a été reçue et non prise en compte, ceci ne devrait pas arriver normalement, mais nous verrons que c'est un cas plausible qu'il faut néanmoins prévoir si le timeout est trop juste. On déclenche donc un « timer » qui va compter le temps maximal à attendre un acquittement (timeout), puis la trame est transmise à l'interface Physique et enfin on passe alors en attente de la réception de l'acquittement. Un message Timeout est posté par un traitant d'interruption lorsque le timer est écoulé.
Pour un DPI si l'acquittement n'est pas reçu avant la fin du timeout, on réémet la requête autant de fois que spécifié par le nombre de réémissions. Si aucune trame n'est reçue, une notification de perte de trame sera retournée à l'application source. Dans le cas particulier où un timeout survient juste avant la réception de l'acquittement attendu (timeout trop court), cet acquittement ne peut pas être pris en compte, mais

il sera supprimé juste avant l'envoi d'une nouvelle requête comme cela a été précédemment mentionné. Pour une réponse de groupe, le timeout correspond à un intervalle de temps de parole pour une unité. Dans notre fonctionnement, nous ne faisons pas de glissement ; à chaque fin d'intervalle de temps, on vérifie s'il y a eu une réponse. Après avoir attendu la réponse de chaque unité dans le groupe, on relance une requête si des acquittements n'ont pas été reçus. Pour optimiser l'exploitation du médium, on pourrait aussi envisager une stratégie qui viserait à relancer par un adressage individuel uniquement, les unités dont l'acquittement n'a pas été reçu ; le but étant de récupérer des acquittements perdus plus rapidement qu'avec un adressage de groupe, il faudrait alors définir un nombre maximum d'unités à adresser individuellement qui garantisse un temps total de retour d'acquittement inférieur au temps normal de réponse de groupe.
Par la suite, il est parfois nécessaire de traiter au niveau de notre tâche gestion du réseau les acquittements reçus ; notamment pour le retour d'une requête d'association ou d'abonnement à un groupe, car il faut enregistrer la nouvelle unité associée au réseau ou abonnée à un groupe s'il n'y a pas eu d'erreur au niveau de l'unité dans l'exécution de la requête. Enfin grâce à l'identifiant de l'application source, l'acquittement est retourné au sous-module de la couche Application qui est à l'origine de la requête. Notre tâche ne peut traiter qu'une requête à la fois.

3.3.3 Droit de parole de groupe

Lorsque plusieurs USRs sont sollicitées, on fait appel au droit de parole de groupe (DPG). Dans ce dernier cas, les différentes USRs sont alors abonnées à un groupe via une opération d'abonnement. Chaque USR possède une priorité dans le groupe qui lui permet de se positionner automatiquement par rapport aux autres USRs.
La Figure 3.6 récapitule les deux types de droit de parole :

— L'attribution d'un DPI à l'USR3 à droite de la figure.
— L'attribution d'un DPG à un ensemble de 5 USRs à gauche de la figure. Chaque USR se positionne selon son intervalle dans la fenêtre de temps. Ce positionnement est défini par la durée de l'intervalle D et la priorité (donc la position) de l'USR :

*Date de début d'intervalle = D * Priorité*

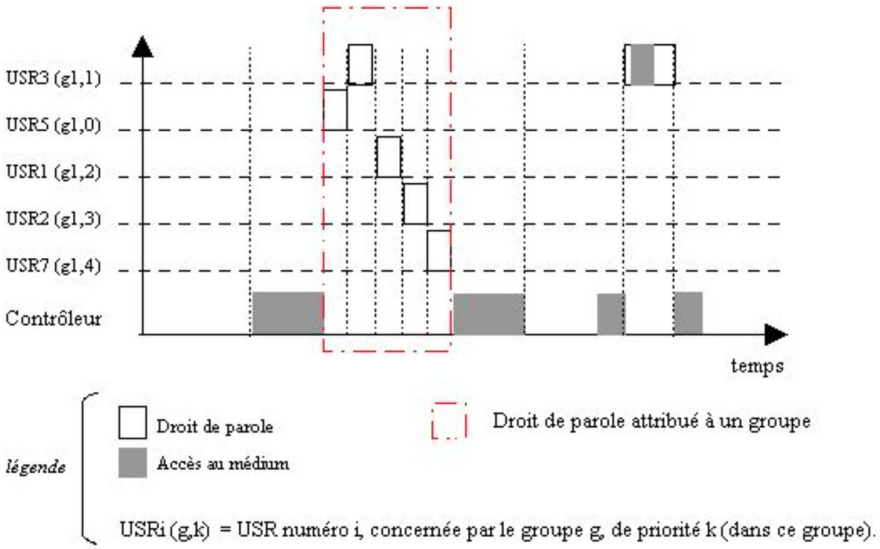


FIGURE 3.6 – Droit de parole individuel et droit de parole de groupe

Ce positionnement est relatif, i.e. chaque USR calcule la date de début de son intervalle par rapport à l'instant de réception du message d'attribution du DPG de groupe envoyé par le contrôleur. Le temps

de propagation n'est pas nécessairement le même pour atteindre chaque USR. Ce retard est supposé constant dans le temps (topologie et distances préservées), mais il constitue un risque de chevauchement des intervalles qui de fait induit un risque de collision (accès non déterministe) ; plusieurs USRs se trouvant potentiellement en possession du DPG Figure 3.6.

Glissement : Le glissement peut être activé lors de l'octroi du droit de parole de groupe à une USR. Ce glissement repose sur l'écoute du premier intervalle. Si l'USR a envoyé une réponse, il y a des chances que le deuxième intervalle soit réservé à une réaction du contrôleur. Au contraire, si le premier intervalle n'a pas été utilisé par l'USR, alors chaque USR fait glisser son intervalle d'une demi-période.
La Figure 3.7 récapitule le glissement d'intervalle en droit de parole de groupe.

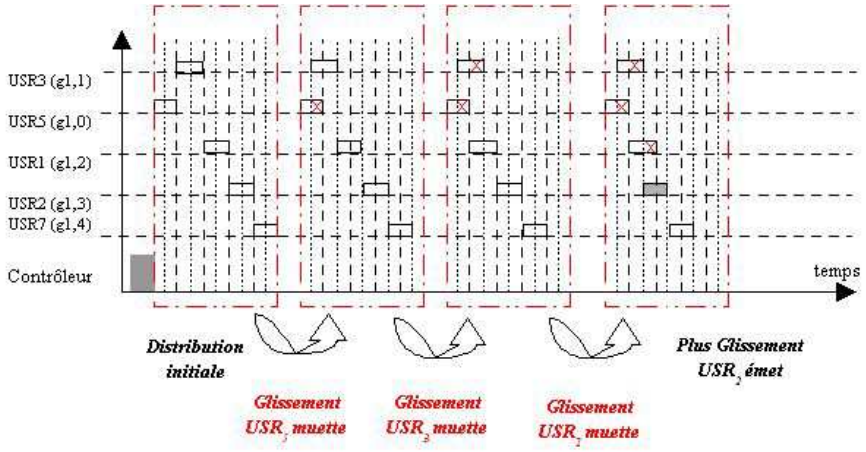


FIGURE 3.7 – Glissements d'intervalle de Droit de Parole au sein d'un groupe

3.3.4 La couche application

La couche application a pour objectif de piloter les différents processus applicatifs de stimulation et d'acquisition ainsi que de configurer et de superviser l'état général de l'unité. Du côté du contrôleur, la couche application traite les demandes des différentes tâches de commandes exécutées sur le contrôleur et gère l'exécution de ces demandes par les unités. Elle a la charge de transmettre ces demandes sous forme de requêtes aux unités visées et de piloter les unités à distance de manière autonome – recueil d'acquisitions ou exécution d'une loi de commande en ou boucle ouverte (BO) en boucle fermée (BF) – tout en retournant régulièrement aux tâches de commandes et d'ordonnancement, des informations sur ces unités (acquisition, état de fonctionnement, autonomie, etc.). En pratique le contrôleur va donc s'adresser régulièrement à ses unités pour s'assurer de leur présence et/ou modifier leurs paramètres fonctionnels suivant une loi de commande et/ou remonter des informations.
Au niveau des unités, la couche Application doit principalement décoder les requêtes du contrôleur et les transmettre aux processus applicatifs correspondants implantés sur la carte métier. Elle a aussi un rôle de supervision locale comprenant : la mise à jour d'informations concernant l'unité, l'observation de l'autonomie et le stockage de mesures venant des processus applicatifs.

La gestion de la stimulation par le contrôleur

Pour la gestion de la stimulation, les requêtes ne concernent pas la partie réseau, elles sont donc simplement transférées par le contrôleur aux unités par la Gestion du Réseau. La tâche de Gestion de la Stimulation doit donc conditionner sous forme de trames de requêtes et transférer aux unités les demandes suivantes :

Configuration : La configuration de la stimulation précise les paramètres par défaut du profil de stimulation (largeur d'impulsion, amplitude et fréquence de l'impulsion électrique) ainsi que le type et les règles de la modulation (amplitude ou largeur d'impulsion) locale- ment réalisée au niveau de l'unité. Lancement : Après la configuration de toutes les unités sollicitées, le lancement de la stimulation- peut être demandé. Cette demande active la stimulation sur toutes les unités configurées et au retour de tous les acquittements de lancement de la stimulation, un timer cyclique (périodique) est activé afin de lancer périodiquement des requêtes de test de présence.

Réglage de l'amplitude : En cours d'exécution d'une génération de stimulation, cette requête permet de mettre à jour la valeur du courant délivré ; ce type de requête peut être utilisé sur demande l'utilisateur ou par une loi de commande embarquée sur le contrôleur .

Réglage de la largeur d'impulsion : De même la largeur de l'impulsion peut aussi être modifiée- fiée sur demande de l'utilisateur ou par une loi de commande.

Arrêt : À tout moment on peut arrêter l'activité de stimulation sur les unités, soit sur demande de l'utilisateur, soit à la fin d'une loi de commande ou encore à la détection d'une erreur conséquente.

On retrouve ainsi du côté des unités les codes opérations suivants relatifs à ces demandes (cf. section A.3).

3.4 Stimulation

Comme on l'a vu dans , l'impulsion créneau biphasique est la forme la plus utilisée sur les stimulateurs du commerce, voire même dans la recherche. Dès lors, la configuration envoyée par le contrôleur se définit par :

— la forme de l'impulsion (créneau biphasique ou monophasique, exponentiel, etc., Figure 2.4 et Figure 3.8),
— la fréquence et la largeur des impulsions électriques,
— l'amplitude par défaut du courant et sa limite maximale (pour la sécurité).

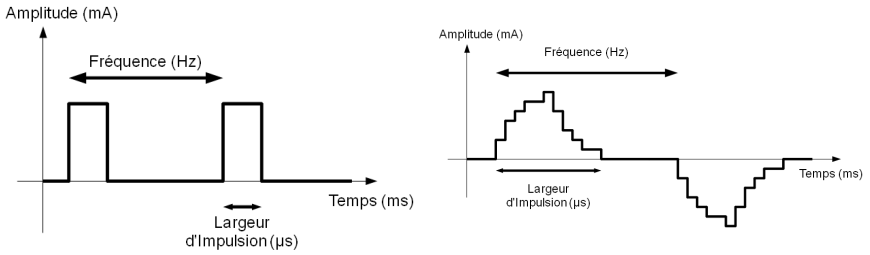


FIGURE 3.8 – Exemple de formes impulsions à générée

Si le profil de stimulation (i.e. la forme de l'impulsion) est figé durant l'exécution, la fréquence, la largeur d'impulsion et l'amplitude du courant restent modifiables dynamiquement à condition que les nouvelles valeurs de paramètres respectent les contraintes définies par défaut ou à la configuration.

Pour la sécurité fonctionnelle du système et celle du patient, on s'assure qu'aucune instruction ni de paramètre incorrect ne soit pris en compte (exemple : instruction inconnue, valeur d'un paramètre de stimulation trop élevé . . .). Pour filtrer les ordres incohérents, comme un lancement de la stimulation alors qu'aucune configuration n'a été reçue, l'interpréteur de requête s'appuie sur une machine à état représentant l'état du système et son évolution normale. La détection du non-respect des contraintes de sécurité, qu'il s'agisse d'erreurs dans le contenu des requêtes du contrôleur ou d'un défaut de la génération d'une impulsion, conduit successivement à :

— une non-prise en compte de la requête ou une mise en état de repli sûr,
— la notification de l'erreur dans l'acquittement au contrôleur,
— l'attente de réarmement (réinitialisation).

Chapitre 4

Objectifs : évolutions du système et solutions proposées

Dans ce chapitre nous proposent les différentes approche et solution proposé pour la réalisations de d'ordonnancement régulé, de commande en boucle fermée et d'ordonnancement et optimisation des communication réseaux.

4.1 Étude d'une commande de SEF en boucle fermée

La commande de la stimulation est réalisée actuellement avec une commande qu'on pourrait qualifier de "pseudo boucle fermée". En effet, dans certaines applications la stimulation est activée ou non selon le retour EMG. Il ne s'agit pas vraiment de BF, car la stimulation n'est pas dépendante (pas modulée) de la mesure EMG ; elle est simplement déclenchée par le fait que le signal EMG recueilli dépasse une certaine valeur.
Le but est de réaliser une commande en boucle fermée qui repose sur le principe général d'une commande qui se déroule en 3 étapes :

— Lecture des informations capteur (EMG, Goniomètre, central inertielle . . .)
— Calcul de la commande.
— Envoi de la commande aux actionneurs (PODs de stimulation).

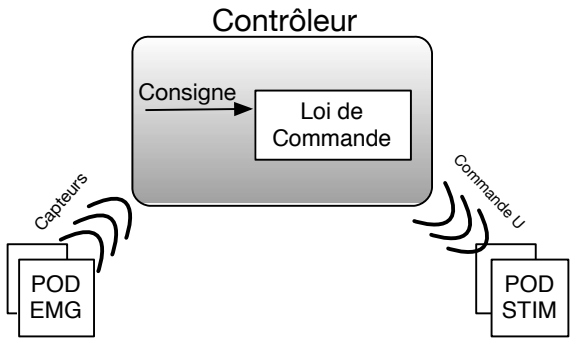


FIGURE 4.1 – Commande en boucle fermée de la stimulation

La commande sera implémentée sur le contrôleur dans le cas du mode "autonome", ou sur le PC de commande dans le cas du mode "connecté".
Pour réaliser la commande, il faut étudier le système, le modéliser et réfléchir à la loi de commande la plus adaptée. Le type de loi de commande dépendra de l'application de SEF à réaliser, ainsi que des ressources de calculs disponibles, mais dans tous les cas toutes les lois de commande auront la structure

citée précédemment.
Un exemple de commande que l'on peut étudier est la commande en position articulaire du coude ou du genou (cf, figure 4.2). Cette commande nécessite comme capteur un goniomètre qui donne l'angle de l'articulation, cette angle est comparé à la consigne ce qui nous donne l'erreur et cette dernière passe par un correcteur PID qui retourne l'amplitude du courant à envoyer au stimulateur afin de converger vers l'angle désiré.
Le profil de stimulation sera réglé avec les paramètres suivants :

— Fréquence de stimulation : $25Hz$
— Large d'impulsion du créneau biphasique : $150\mu s$

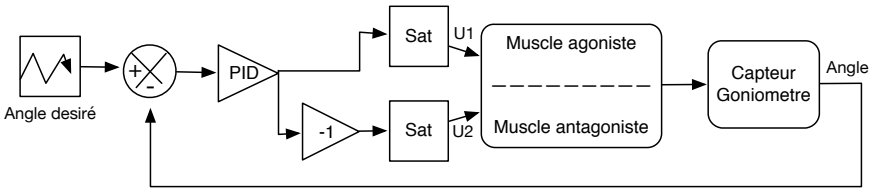


FIGURE 4.2 – Commande avec PID en boucle fermée de l'angle d'une articulation

Les correcteurs PID (pour « proportionnel intégral dérivé ») sont largement utilisés, car ils sont capables de contrôler beaucoup de systèmes entrés uniques / sortie uniques (SISO) à travers une modélisation et un effort de mise en point très simple [RBD+13]. Dans ce cas, l'entrée de commande u est écrite en fonction de l'erreur entre la sortie désirée et la sortie mesurée du système $e(t) = r(t) - y(t)$ en tant que :

$$u(t) = Ke(t) + \frac{K}{T_i}\int e(\tau)d\tau + KT_d\frac{d}{dt}e(t) \tag{4.1}$$

Ici, le terme proportionnel $Ke(t)$ contrôle la bande passante et temps de montée de la boucle de commande, le terme drivé $KT_d\frac{d}{dt}e(t)$ amortit les oscillations et les dépassements et le terme intégral $\frac{K}{T_i}\int e(\tau)d\tau$ annule l'erreur statique.
Le PID continu idéal doit être discrétisé à des fins de mise en œuvre. Par exemple, en utilisant une méthode de différences vers l'arrière (à la période d'échantillonnage de T) on obtient :

$$u(t) = u(t-1) + K[e(t) - e(t-1)] + \frac{KT_s}{T_i}e(t) + \frac{KT_d}{T_s}[e(t) - 2e(t-1) + e(t-2)] \tag{4.2}$$

Ce PID nous permet de ne pas avoir de saturation d'intégrateur et donc pas d'effet *Integral windup*, car avec cet algorithme PID, on ne fait pas la somme des erreurs précédentes pour généré l'action intégral.

4.2 Cas de plusieurs applications SEF

Dans le cas de l'utilisation du système, ce dernier doit pouvoir gérer plusieurs applications SEF à la foi, chaque application équivaut à une commande.
On peut distinguer généralement deux cas :

1. **Cas de plusieurs applications sur un seul patient :** Dans ce cas, compte tenu de la dynamique différente entre différents muscles et articulations, la boucle de commande qui sert à commander les muscles des bras n'est pas la même que celle des jambes, de ce fait le contrôleur doit réaliser plusieurs commandes adaptées à chaque partie commandée (cf. figure 4.3), ce cas peut être utilisé en mode "connecté" ou en mode "autonome".
2. **Cas de plusieurs applications sur plusieurs patient :** Dans ce cas aussi le contrôleur doit gérer différentes SEF avec des dynamiques différentes qui dépendent du patient et des muscles, cette utilisation sera exclusive au mode "connecté".

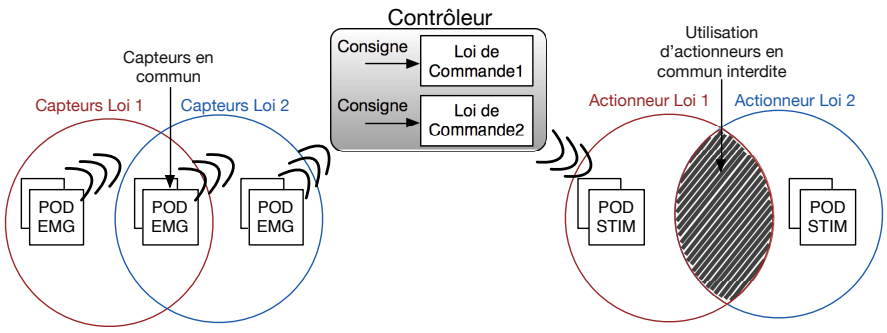


FIGURE 4.3 – Plusieurs commandes de stimulation en boucle fermée dans un même contrôleur

4.3 Travail à réaliser sur la gestion des communications

— Si l'identifiant d'une requête information capteur est le même pour tous les PODs concernés le contrôleur doit alors envoyer une seule trame qui sera reçue par tous les PODs (multicast) puis ces derniers renvoie l'information au contrôleur selon leurs droits de parole.
— Quand les PODs reçoivent un profil de stimulation ils l'appliquent périodiquement jusqu'à que le contrôleur envoie un nouveau profil qui sera lui aussi exécuté périodiquement. Ce procédé permettra la libération de la bande passante en cas d'envoi de profil de stimulation similaire, il faudra néanmoins que le contrôleur envoie une requête de présence à tous les PODs avant la fin de leur timeout même si le profil de stimulation n'a pas changé.
— Lors de l'utilisation de plusieurs commandes sur le même contrôleur, les différentes commandes peuvent avoir des capteurs en commun (cf. figure 4.3), il faut donc utiliser la même information envoyée par les capteurs pour les différentes commandes, ce qui permettra de réduire la charge du réseau.
— Le contrôleur doit pouvoir modifier dynamiquement le droit de parole des PODs selon les besoins en calculs de la commande, il doit aussi faire varier la fréquence d'échantillonnage selon la nature des mouvements et des muscles commandés.
— Lors d'une commande de deux muscles différents agissant sur la même articulation (agoniste/antagoniste), la commande n'est envoyé qu'au muscle qui permet l'activation du mouvement, une trame de présence et envoyé périodiquement à l'autre POD pour le garder actif.

4.3.1 Protocole multi charge/multi-destinataire

La communication peut être optimisée par l'instauration de trame multi-charge multi-destinataire, cette communication permet avec une seule trame d'envoyer des requêtes différentes à différents destinataires (PODs), chaque POD se chargera de ne prendre que la requête qui lui est destinée. Cette modification permettra de décharger le réseau et de libérer le contrôleur afin qu'il puisse exécuter d'autres tâches.

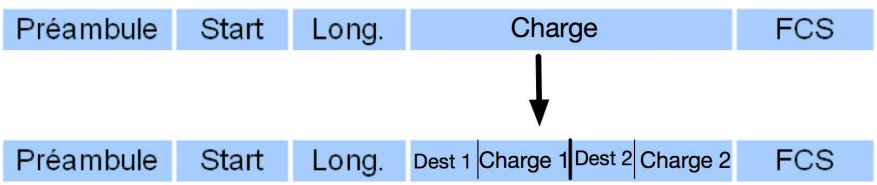


FIGURE 4.4 – Format d'une trame avec l'intégration du multi charge

Pour cela il faudra modifier le format de trame afin de permettre l'identification des charges de la trame par les PODs qui les reçoivent et ainsi concevoir un filtre qui permettra la prise en compte que de la charge voulue.

4.4 Adaptation de la puissance du signal

Actuellement la puissance d'émission du signal est fixe afin de garantir une qualité de transmission optimale, mais cette approche est très gourmande en énergie, car même quand les PODs sont très proches du contrôleur la puissance reste élevée.
Il est souhaitable de réaliser une commande de la puissance du signal en boucle fermée qui utilisera la puissance de réception et la qualité du signal pour ajuster la puissance en temps réel et ainsi minimiser l'impact énergétique.
Les statistiques sur la puissance de réceptions et les erreurs de trame seront envoyées par les PODs au contrôleur, il faudra donc veiller à ne pas surcharger le réseau et le contrôleur.
Ce travail n'a pas pus être implémenté dans le simulateur faute de temps, mais il le sera dans les évolutions futures.

4.5 L'ordonnancement sur le contrôleur

Le contrôleur doit réaliser plusieurs tâches en parallèle tout en respectant les contraintes temps réel, il doit effectuer principalement les tâches de communications et de calcul de commandes, en plus de cela il doit pouvoir gérer aussi plusieurs commandes en parallèle tout en optimisant les temps (ex : utiliser le temps d'attente de réponse d'un POD pour réaliser d'autres opérations). Le contrôleur doit aussi adapter l'intervalle d'échantillonnage par rapport à chaque tâche, donc il faut bien connaître la dynamique du système pour bien adapter les fréquences d'échantillonnage et les faire varier en temps réel.
Différentes méthodes d'ordonnancement peuvent être utilisées, on peut envisager l'utilisation d'un contrôleur d'ordonnancement (feedback scheduler) [DASM13] qui utilise des mesures d'activité du système informatique (par exemple la charge du processeur) pour produire comme variables de commande des paramètres d'ordonnancement des activités contrôlées (par exemple l'intervalle d'échantillonnage, la priorité...) (cf. figure 4.5). Les processus informatiques peuvent ainsi bénéficier des propriétés d'adaptabilité et de robustesse des commandes en boucle fermée.

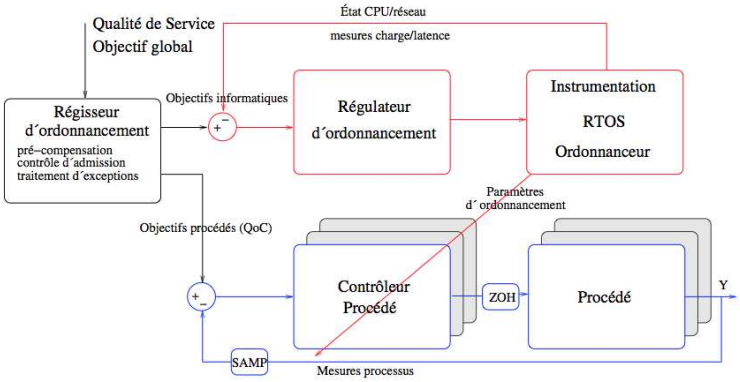


FIGURE 4.5 – Architecture de commande avec ordonnancement régulé [DASM13]

— Les boucles internes (en bleu) sont les commandes de procédés physiques, conçues et paramétrées pour être compatibles avec des paramètres d'ordonnancement variables.
— La boucle externe (en rouge) est la boucle de régulation d'ordonnancement, calculant de façon cyclique de nouveaux paramètres d'ordonnancement en fonction des mesures de l'activité informatique (et éventuellement d'estimations de performance des commandes de procédés).

— Des couches de plus haut niveau (régisseur) concernent la sûreté de fonctionnement du système et le traitement d'exceptions, en gérant les changements de mode de marche et la reconfiguration du système lorsqu'il est nécessaire de commuter entre lois de commandes.

4.5.1 Ordonnancement sous contrainte (m,k)-firm

Le modèle (mk)-firm a été proposé par Hamdaoui et Ramanathan [SSS14] pour caractériser d'une manière précise le niveau de garantie temporelle offerte à des applications temps-réel tolérant le dépassement d'échéances de certaines instances. Le modèle (m,k)-firm est caractérisé par deux paramètres. Plus formellement, une application est dite sous contrainte temporelle (m,k)-firm si au moins *m* instances, parmi *k* instances consécutives, doivent impérativement être exécutées en respectant leur échéance temporelle. Ceci dit que si la contrainte temporelle (m,k)-firm d'une application est respectée, alors dans n'importe quelle fenêtre de *k* instances de suite, il existe au moins *m* instances qui respectent leurs échéances.
Une tâche sous contraintes (m,k)-firm peut se trouver dans l'un des deux états : normal ou échec dynamique. La connaissance de son état à l'instant *t* dépend de l'historique du traitement des *k* dernières instances. Si on associe '1' à une instance respectant son échéance et '0' à une instance ratant son échéance, cet historique est entièrement décrit par une suite des '0' et des '1' de longueur bits appelée un (m,k)-pattern. La figure 4.6 donne un exemple de diagramme état-transition des (m,k)-patterns d'une tâche sous contrainte (2,3)-firm. Par convention, le déplacement des bits dans un (m,k)-pattern se fait de droite vers la gauche.

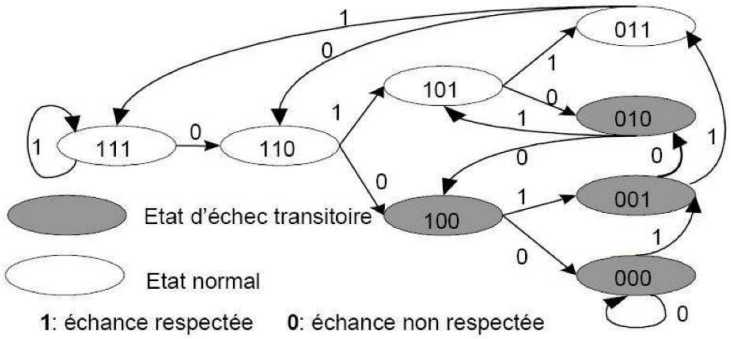


FIGURE 4.6 – Diagramme d'état-transition d'une tâche avec (2,3)-firm

Toute instance qui ne peut pas respecter son échéance ne sera tout simplement pas exécutée (par conséquent sera rejetée par le système et ne consommera pas de ressources). Ce point est particulièrement important pour gérer la situation de surcharge du système.

4.5.2 Approche du contrôleur d'ordonnancement dans notre cas au niveau du CPU

Dans notre cas, le contrôleur doit exécuter différentes tâches en parallèle, les tâches de communication, contrôle de la SEF, supervision et arrêt d'urgence. On va donc réaliser un contrôleur d'ordonnancement qui va gérer toutes ces tâches afin d'obtenir les meilleures performances.
Le temps d'exécution de toutes les tâches ainsi que la charge du système devra être calculé ou fourni par le noyau temps réel de l'OS. Chaque tâche tournera en boucle et intégrera des fonctions qui permettant de récupérer ses performances. Le temps d'exécution, le nombre de tâches, et la charge désirée varieront tout en long de l'exécution. Le rôle du contrôleur d'ordonnancement et d'optimiser les performances globales et réguler la charge du CPU au niveau désiré.
Le contrôleur d'ordonnancement aura comme levier d'action l'intervalle d'échantillonnage de chaque tâche, les objectifs informatiques lui seront fournis par le superviseur, ce dernier aura accès à l'état des

tâches, à la charge du processeur ...(cf. figure 4.8). La charge désirée devra être calculée avec des équations qu'on devra étudier.
Les tâches et le contrôleur d'ordonnancement communiqueront entre eux par des requêtes et des événements. Le contrôleur d'ordonnancement obtient les statistiques d'exécution, telles que les temps d'exécution des tâches et la charge du système à partir du noyau, ces informations permettront de calculer de façon optimale la charge du CPU désiré.
Le contrôleur d'ordonnancement sera une tâche temps réel exécuté par le contrôleur (physique) (cf. figure 4.8) avec un intervalle d'échantillonnage adapté aux performances du système (élevée si le système est très réactif ou faible si système lent). On devra aussi choisir la loi de commande et les équations qui nous permettront de calculer les nouveaux intervalles d'échantillonnage.

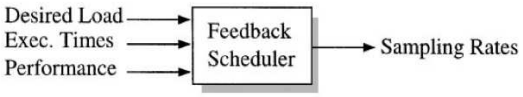


FIGURE 4.7 – Contrôleur d'ordonnancement [EHA00]

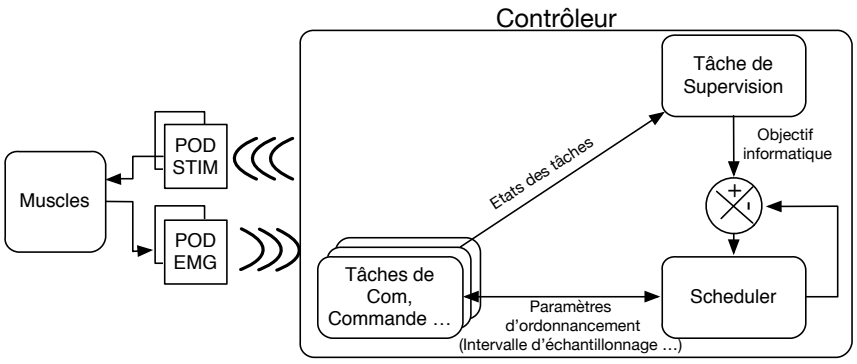


FIGURE 4.8 – Ordonnancement sur le contrôleur

Dans la figure 4.8 on voit que la structure de commande est fixe. Un cas plus avancé qu'on peut étudier et la mise en place d'un ordonnancement adaptatif, par exemple la possibilité d'ajout d'un nouveau membre à stimuler donc une nouvelle loi de commande à chaud sans interrompre la loi de commande qui est déjà en cours d'exécution.// Une étude plus avancée de cette situation sera donnée en temps voulu.

4.5.3 Approche du contrôleur (m,k)-firm pour gérer le réseau dans notre cas

Pour ce qui est de la gestion du réseau, un ordonnancement sous contrainte (m,k)-firm serait plus judicieux, il nous permettra de ne pas envoyer les trames qui sont sûres d'être perdues ou pas délivrées à temps et ainsi limiter la surcharge du réseau. Pour une contrainte (3,4)-firm, pour une suite de 4 envois consécutifs, si 3 trois trames ne respectent pas leurs échéances d'envoi ou un POD ne retourne pas d'acquittement, cette requête sera considérée en échec transitoire et ne sera plus envoyé.
Pour éviter cela on propose une approche "**Distance-base priority**" (DBP) [CA10] qui permet de réduire la probabilité d'échec dynamique. Cette approche définit la notion de distance entre l'état actuel d'une tâche à l'instant t et un état d'échec dynamique. Pour un (m,k)-pattern en état normal à l'instant t,cette distance est définie par le nombre consécutif de bits '0' que l'on doit rajouter au (m,k)-pattern pour être en échec dynamique. De ce fait, la priorité affectée par DBP aux différentes requêtes est inversement

proportionnelle à la distance séparant l'état actuel de l'échec dynamique. Si une requête se trouve déjà en état d'échec dynamique, c'est-à-dire il existe moins d'échéances respectées dans le (m,k)-pattern, la plus haute priorité '0' est affectée à cette requête.
Pour mettre en œuvre cela, une tâche de communication recevra toutes les requêtes des tâches qui souhaite communiquée avec les PODs (cf. figure 4.9), elle attribuera une priorité à toute les requêtes et comptera le nombre d'échecs de chaque requête selon les contraintes (m,k)-firm, elle modifiera les priorités des requêtes selon leur état (normal, proche de l'échec, ou en échec dynamique).

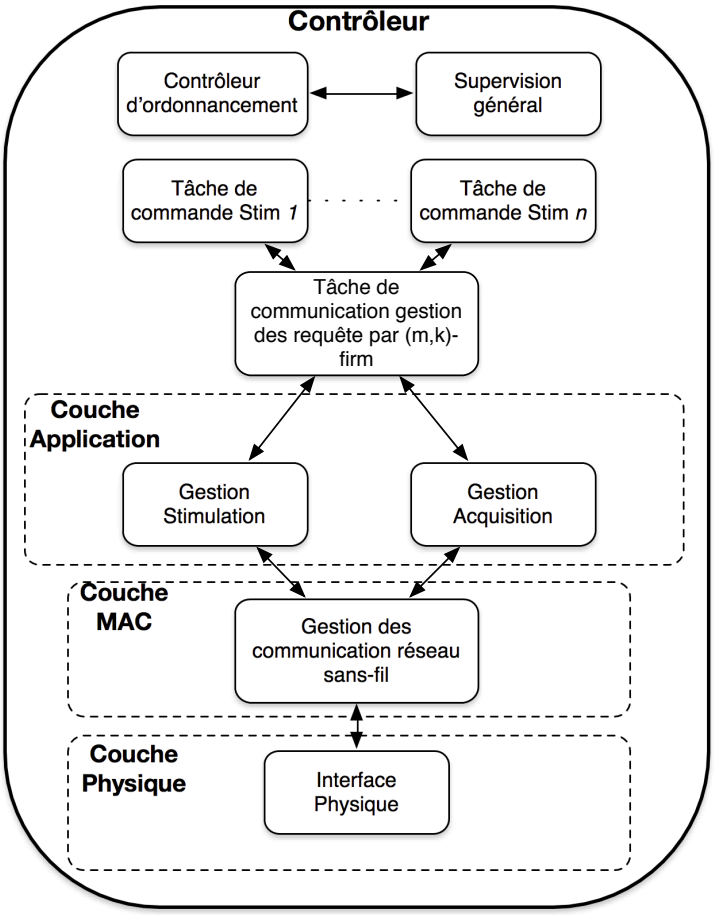


FIGURE 4.9 – Structure des tâches sur le contrôleur

4.6 Sûreté de fonctionnement

Afin d'assurer un fonctionnement optimal en toute sécurité, des tâches de supervision seront réalisées sur les PODs et le contrôleur :

Sur le POD :

- La tâche de supervision devra surveiller les timeouts de réception de trame et les taux de réception de messages non destinés au POD, dans le cas où on atteint un seuil critique, cette tâche devra être en mesure de déclencher un arrêt d'urgence.
- Dans le cas d'une stimulation, une saturation de l'amplitude du signal de stimulation est nécessaire au niveau logiciel et matériel pour éviter toute décharge nocive pour le patient.

— La tâche de supervision fera aussi du "House keeping", cette dernière surveillera les éléments cruciaux au fonctionnement tel que la batterie.
— Prévoir un système d'arrêt d'urgence qui sera toujours accessible par l'utilisateur, quelque soit l'état des commandes en cours.

Sur le Contrôleur : Deux tâches de supervision seront réalisées, une pour la surveillance des communications et une autre pour la supervision des tâches et des ressources système.

— La tâche de supervision système recueillera les informations à propos de toutes les tâches exécutées sur le contrôleur ainsi que les informations de qualité de service, cette tâche communiquera directement avec le contrôleur d'ordonnancement et lui fournira les objectifs informatiques de l'exécution des tâches. Ce superviseur devra aussi avoir une priorité plus élevée que celle des tâches de commande.
— La tâche de supervision des communications devra surveiller les communications avec les PODS et vérifier les retours d'informations et d'acquittements avant la fin des timeouts, et aussi vérifier l'ordre des réceptions dans le cas de communication de groupe. Dans le cas d'un timeout ou mauvais acquittement, une autres requête et envoyé, si aucune réponse n'est reçue, on diffuse une trame sur le réseau afin de stopper toute stimulation en relation avec la commande en cours.

Chapitre 5

Réalisation Simulateur d'architecture de stimulation distribuée

5.1 But du simulateur

Le but de ce simulateur de pouvoir simuler un système de stimulation distribué comportant un contrôleur qui exécute des commandes de fonction SEF en boucle fermée, communiquant avec des PODs via un médium sans-fil (cf. Figure 5.1).

Le code source du simulateur est disponible sur le site de développement collaboratif gforge (en projet privé) `https://gforge.inria.fr/projects/rtstim/` et documenté par Doxygen.

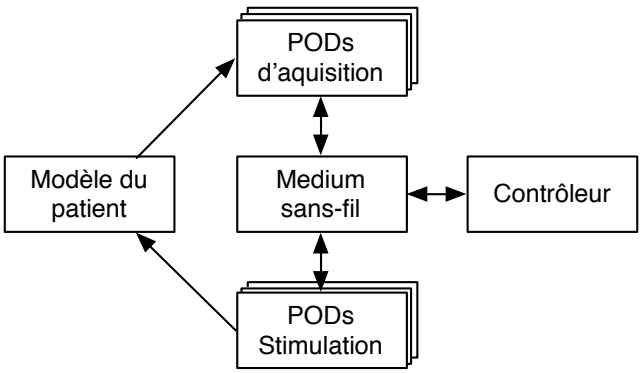


FIGURE 5.1 – Schéma bloc de la structure du simulateur

Notre but est de réaliser un simulateur temps réel de système de stimulation électrique distribué, ce dernier simulera la carte contrôleur, les PODs, le patient, et tout l'aspect communication sans fil. Nous voulons montrer l'effet en terme de performance et de stabilité qu'offrent différents types d'ordonnancement et d'optimisation réseau aux commandes SEF sur un médium simple comportant des délais et des pertes aléatoires. Nous laissons le système ouvert à de futures améliorations du modèle pour qu'il s'approche le plus du modèle STIMAP réel.
Le simulateur sera capable de maintenir la cohérence temporelle, et exécutera les algorithmes de contrôle à la même fréquence que celle utilisée sur le système réel, tout en fournissant les données capteurs au même taux d'échantillonnage que les capteurs réels. Il est important de tenir compte du délai bout en bout dans commande en boucle fermé, c'est pour cela que tous les modèles devront reproduire les délais réels que ce soit au niveau de communications ou traitement de tâches.

L'intérêt de ce simulateur est donc important dans les phases de développement et de test, car il est

nous est impossible d'effectuer des tests de commande SEF en boucle fermée sur des patients réels et il nous permettra de tester des évolutions de l'architecture afin de valider la recherche théorique avant de réaliser une implémentation sur un système réel.

Le simulateur sera donc structuré selon plusieurs modules qui représenteront chacun un élément physique du système réel, en d'autres termes on aura un bloc pour la carte contrôleur, pour les PODs, pour le médium sans-fil, et pour le modèle musculaire et squelettique du patient (cf. figure 5.1). Chaque module contiendra toutes les fonctions qui sont exécutées sur chaque composant du système réel (commande SEF, supervision, gestion des acquisitions).

En ce qui concerne l'ordonnancement, trois hypothèses seront étudiées :

Hypothèse 1. Deux ordonnanceurs non couplés : On envisagerait dans un premier temps un ordonnancement pour les tâches exécutées sur le contrôleur uniquement, puis on pourrait ajouter un ordonnancement au niveau des communications, dans ce dernier cas il faudra tenir compte du caractère non préemptif de l'information sur le réseau, ces deux ordonnanceur comme on les a envisagés ne seront pas couplés.

Hypothèse 2. Deux ordonnanceurs couplés : Cette hypothèse serait le couplage (synchronisations) des différents ordonnanceurs, cela permettra une meilleure interaction entre les ordonnanceur au détriment du coût calcul.

Hypothèse 2. Un seul ordonnanceur global : Cette dernière hypothèse serait de faire un seul ordonnanceur qui puisse gérer l'aspect ordonnancement des tâches ainsi que l'ordonnancement des communications.

Toutes ses hypothèses ne pourraient être justifiées sur le papier, et encore moins être testées sur le système réel, d'où le but d'utiliser notre simulateur qui nous permettra de confirmer le choix de l'une de ces hypothèses.

5.2 Nos besoins pour le simulateur

Pour étudier l'impact qu'auront les différentes hypothèses d'ordonnancement sur le système réel, nous aurons besoin de modèles plus ou moins réalistes :

Modèle du médium : Afin d'assurer une représentation réaliste des échanges sur le médium, le modèle devra pouvoir reproduire les délais de transmission, et aussi de pouvoir générer à la demande ou aléatoirement des pertes de paquet. Une amélioration de ce modèle peu être réalisé par la suite afin de se rapproché au maximum du médium réel.

Modèle du POD acquisition : Comme on ne pourra pas modifier le programme du POD réel (marquage CE), on le représentera dans notre simulateur sous une forme basique, il récupéra simplement informations du modèle musculaire à la demande du contrôleur ou à une fréquence fixe, les stockera et les enverra à la demande sur le médium. Ce modèle inclura aussi des délais pour représenter au maximum les délais de récupérations et encapsulation des données capteur, l'envoi des informations sur le bus entre la partie métier du PODs et la partie générique, puis traitement d'information pour stockage et d'encapsulation dans la trame d'envoi. Ce modèle devra aussi inclure les spécificités réseau du POD réel telles que l'accès au médium (attente de droit de parole individuel, droit de parole de groupe avec intervalle glissant ...) (*cf. section A.1.1*)

Modèle du POD stimulation : Le modèle du POD stim devra généré une stimulation à la demande du contrôleur et avec les paramètres envoyés par ce dernier, il inclura aussi toutes fonctions de sûreté et de gestion de stimulation que comporte le POD réel. On introduira aussi des délais pour représenter les temps de traitement de l'information reçu, encapsulation et l'échange entre la partie stim et la partie générique, il inclura aussi les mêmes spécificités réseau que le POD acquisition.

Modèle du patient : Pour pouvoir simuler notre système de stimulation et exécuter les commandes en boucle fermée, il faudra concevoir un modèle musculaire et squelettique, ce modèle devra pouvoir récupérer les signaux de stimulations et réagir à la façon d'un vrai membre stimulé, et nous fournir des signaux EMG ainsi que la position articulaire d'un membre comme le coude par exemple.

Modèle du contrôleur : Le contrôleur est la pièce maîtresse du système, pour qu'il soit plus proche possible du modèle réel, il devra respecter les contraintes en temps et en puissance de calcul. Il devra représenter les délais induits par la gestion du réseau et par le passage de l'information via la pile protocolaire. Le contrôleur devra gérer la communication sur le médium (Droit de parole, ordonnancement des communications ...), les commandes en boucle fermée ainsi la supervision générale. La gestion du réseau devra pouvoir affecter les PODs à un groupe

Dans le système réel, le temps aller-retour moyen et de 3.02ms, ce qui fait un temps aller-retour moyen cumulé pour un système à un contrôleur et deux PODs de 6.04ms [Tou11]. Nos modèles lors de leur conception devront respecter ces délais afin d'être proches de la réalité.

5.3 Modules du simulateur

Le simulateur sera constitué de modules chaque module représentera un des modèle cité *section* 5.2 (cf. figure 5.2) :

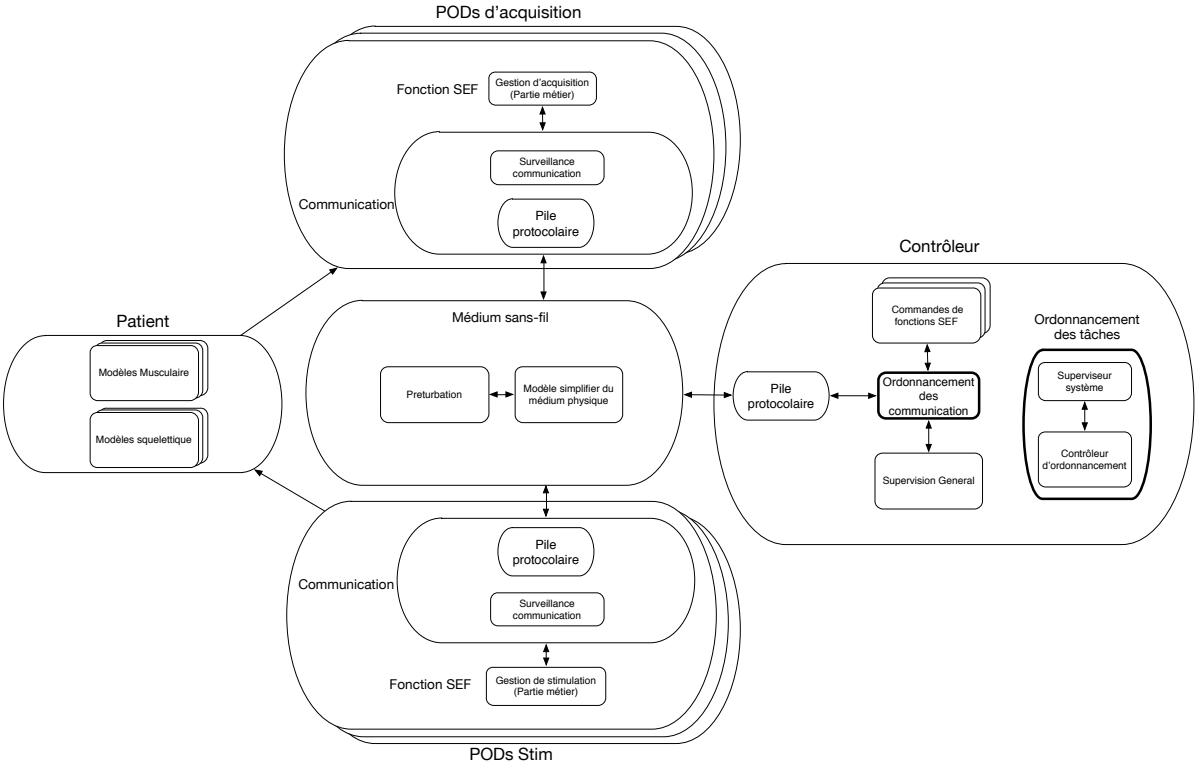


FIGURE 5.2 – Architecture du simulateur de système de stimulation électrique distribué

5.3.1 Module du contrôleur

Ce module simulera la carte contrôleur, il réalisera toutes les tâches effectuées par celle-ci en temps réel. Dans un premier cas on étudiera l'hypothèse de deux ordonnanceur non couplés un pour les tâches

et un autre pour les communications.
Les fonctions qu'inclura ce module sont :

Commandes de fonctions SEF : Cette fonction permettra la commande en boucle fermée de la SEF, elle sera couplée à la fonction ordonnancement des communications, la fonction de commande enverra une requête à cette dernière et se mettra en attente d'un retour d'information capteur, elle calculera ensuite les nouvelles consignes et les enverra au PODs via la fonction d'ordonnancement. Cette tâche aura un intervalle d'échantillonnage variable qui sera commandé par le contrôleur d'ordonnancement toute en gardant la meilleure stabilité possible.

Ordonnancement des tâches : Cette partie sera composée de deux fonctions, la première, superviseur système, permet la récupération des informations système comme la charge du processeur, le temps d'exécutions des tâches (et éventuellement des estimations de performance des commandes de procédés), et d'une 2éme fonction qui sera le contrôleur d'ordonnancement qui calcul de façon cyclique de nouveaux paramètres d'ordonnancement en fonction des mesures envoyées par la fonction de supervision système.

Supervision générale : Cette tâche réunit plusieurs fonctions qui permettent la supervision des parties importantes du système, tel que le test de présence des PODs, la gestion de la puissance du signal, ainsi que la surveillance de la batterie. Toute communication réseau de ses fonctions devra passer par l'ordonnancement des communications.

Pile protocolaire : La pile protocolaire sera composée de fonctions qui simule la couche application ainsi que la couche MAC. Leurs rôles seront d'encapsuler et de transmettre/recevoir l'information sur le médium, ainsi que la gestion du réseau (droit de parole, gestion de trame individuelle ou groupe ...).

5.3.2 Module du POD

Ce module simulera le POD (acquisition et stimulation), il devra simuler les deux cartes du POD, la première s'occupe de la partie réseau, celle-ci exécutera les fonctions de la pile protocolaire, et la deuxième exécutera la fonction d'acquisition des données capteur (EMG, Gonio ...) ou les fonctions permettent de générer des stimulations selon le profil envoyé par le contrôleur.

Pile protocolaire : La pile protocolaire sera composée des différentes fonctions qui simulent les couches de la pile, dans un premier temps leur rôle se limitera aux fonctions basiques, et indispensables aux communications des PODs sur le réseau tel que l'attente du droit de parole, le filtrage des messages ainsi que l'aiguillage et la configuration des PODs (cf. sous-section A.1.1).

Gestion d'acquisition : Cette fonction récupérera les données capteur à la demande de la couche application, elle filtrera ce signal, le formatera et stockera dans un buffer dans l'attente que la couche application de la pile vienne le récupérer. Dans le système réel la carte générique (réseau) et la carte métier sont reliées par un bus série, il y a des délais de traitement d'encapsulation et d'envoi entre les deux cartes, il faudra donc tenir compte de ce délai dans notre simulateur pour gardé des résultats proches du système réel.

Gestion de la stimulation : Cette fonction doit générer un signal de stimulation selon un profil reçu. À l'état initial, cette tâche attend la 1er requête de configuration, puis dès réceptions de cette requête lance la stimulation avec les paramètres choisis en boucle, un timer et aussi lancé pour arrêter la stimulation si aucune requête n'est reçue (présence ou modification de profile de stimulation). Dès qu'un nouveau paramètre de stimulation et calculé par la commande de fonction SEF du contrôleur, il est envoyé au POD et reçu par cette fonction, elle modifie ainsi à la volée le profil de stimulation appliqué au patient.

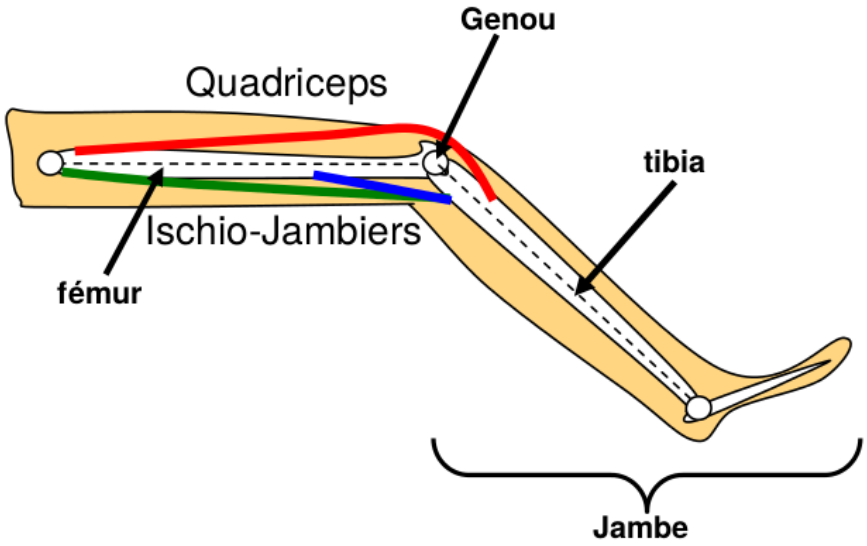
5.3.3 Module du médium sans-fil

Le médium sans-fil sera constitué d'une fonction qui se chargera de propagé les trames envoyées du contrôleur vers les PODs et vice versa, en ajoutant un délai qui correspondra au délai de transmission réel

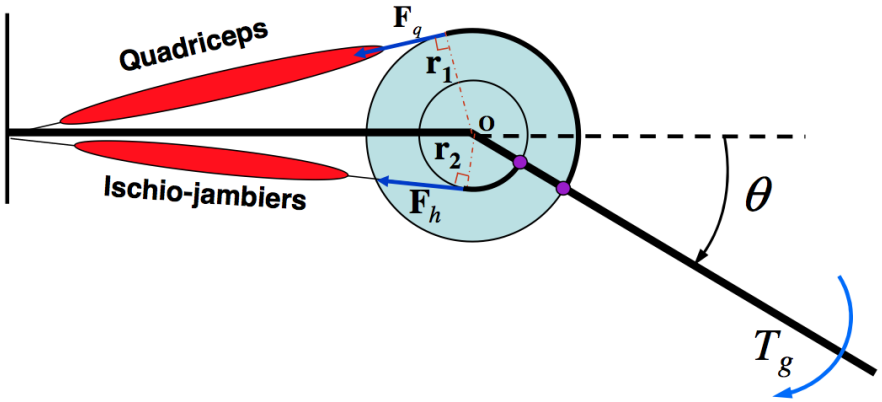
sur le médium, et d'une deuxième fonction qui générera des perturbations en supprimant simplement des trames circulant sur le réseau de façon aléatoire ou contrôlée, cela nous permettra par exemple d'évaluer le niveau de robustesse d'une commande assujettie a des pertes.

5.3.4 Module du patient

Ce module devra contenir un modèle musculaire et squelettique d'un patient. Dans notre cas on utilisera le modèle du genou décrit dans la thèse de BENOUSSAD [Ben09] , ce modèle est constitué du muscle agoniste (Quadriceps) et antagoniste (Ischiojambier), ces deux muscles agissent sur l'articulation pour actionner le mouvement du genou via un système de poulies (cf. Figure 5.3b)



(a) Les actionneurs musculaires de l'articulation du genou.



(b) Système de poulies des actionneurs musculaires.

FIGURE 5.3 – Modèle biomécanique (2D) de l'articulation du genou avec système de poulies [Ben09]

Nous considérerons donc un modèle biomécanique de l'articulation du genou en deux dimensions (2D) à un degré de liberté dans le plan sagittal. Ainsi, le mouvement est contrôlé par deux actionneurs musculaires antagonistes cf Figure 5.3b. La transformation des forces musculaires en couples agissant sur l'articulation du genou est faite selon un système de poulies représentant le passage du muscle par l'articulation. Ce système de poulies est basé sur hypothèse que les bras de levier des deux forces musculaires antagonistes varient très faiblement [Ben09]. Sur la Figure 5.3b, les bras de levier des forces musculaires du quadriceps (Fq) et de l'ischio-jambiers (Fh) correspondent aux rayons des poulies r_1 et $r_2$respectivement. θ est l'angle articulaire du genou autour du centre de rotation o, tel que l'extension

maximale du genou correspond à $\theta = 0$ et sa position verticale à $\theta = 90$. $T_g = M_g \log \sin(\theta - \theta_o)$ est le couple gravité de la jambe autour du genou, où M est la masse de la jambe, Log la distance entre le centre de rotation o et le centre de gravité de la jambe, θ_o l'angle de la position de repos et g accélération de le gravité.
À partir de la Figure 5.3b, les formalisations des longueurs musculaires quadriceps et ischio-jambiers sont données par les équations linéaires Équation 5.1 et Équation 5.1 en fonction des angles articulaires θ :

$$L_1(\theta) = L_1 ext + r_1 \theta \tag{5.1}$$

$$L_2(\theta) = L_2 ext + r_2 \theta \tag{5.2}$$

Avec $L_1 ext$ et $L_2 ext$ les longueurs respectives des muscles quadriceps et ischio-jambiers en extension maximale du genou (c'est à dire à $\theta = 0$). La dynamique du mouvement de l'articulation humaine est souvent donnée par l'équation de second ordre suivante :

$$J\ddot{\theta} = F_2 r_2 - F_1 r_1 + T_g - B\dot{\theta} + T_e \tag{5.3}$$

Où, $\dot{\theta}$ et $\ddot{\theta}$ sont respectivement la vitesse et accélération articulaires du genou et B le coefficient du couple de frottement visqueux. J est le moment d'inertie de la jambe autour du centre de rotation o et T_e le couple élasticité passive, fortement non linéaire, représentant l'étirement passif des muscles et des différentes structures autour du genou (ligaments, tendons). Comme nous n'avons pas pu disposer des paramètres de fortement visqueux et le couple élasticité passive, nous avant remplacé par un modèle linaire valable uniquement en petits mouvements.

5.4 Détail des fonctions des modules

Dans un 1er temps nous allons réalisé la simulation d'un système simple composé d'un modèle de genou, d'un contrôleur qui réalise une commande en boucle fermée, un POD d'acquisition de la position et la vitesse du genou, et un POD de stimulation. Le contrôleur communique avec les PODs par un modèle de médium simple qui représente les délais du système réel.
Chaque module sera représenté par un processus composé de Threads (cf. Figure 5.4), pour cela le langage de programmation utilisé est le C/C++, on utilise l'API ORCCAD et la library POSIX afin de crée des Thread, avoir un système temps réel , récupéré les informations CPU et faire des synchronisations entre threads (Sémaphore, Boite aux lettres ...) [ABPGS10].

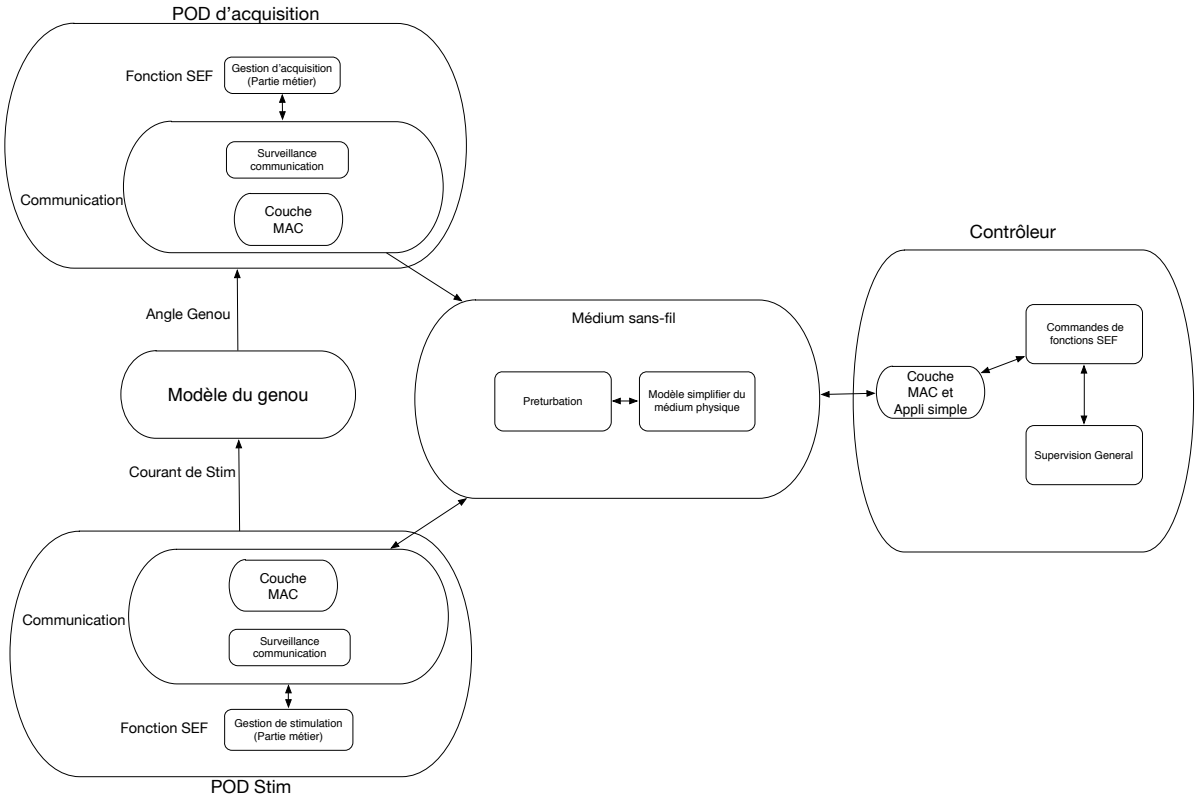


FIGURE 5.4 – Architecture du simulateur de système de stimulation électrique distribué

5.4.1 Processus du Medium

Approche 1 : Dans cette 1erre approche le médium est représenté par un processus composé de deux Threads :

Thread de communication Contro-PODS : Ce premier Thread s'occupe de la communication entre le contrôleur et les PODS, il utilise pour cela un socket UDP pour la communication avec le contrôleur, et autant de Socket UDP que de PODS pour la retransmission de l'information au PODS, un délai de 1.5ms et appliqué entre la réception sur le socket du contrôleur et l'envoi sur les sockets des PODs.

Thread de communication PODS-Contro : Ce deuxième Thread permet la communication retour entre les PODS et le contrôleur, il utilise un Socket UDP pour écouter les PODS (une seule communication à la fois), et retransmet cette information au contrôleur via sont Socket avec l'application d'un délai de 1.5ms.

Cette solution fonctionne, mais nécessite autant de socket que de PODs, une solution utilisant le Multicast (Envoi à plusieurs destinataires en même temps) nous permettrait de nous acquitter de cette contrainte, et ainsi d'éviter les délais d'envoi entre chaque socket.

Approche 2 : Dans cette approche, on améliore la 1erre solution qui est d'utiliser autant de socket que de POD, avec une solution utilisant un seul socket qui permet la diffusion (cf. Figure 5.5), cette solution nécessite des sockets point à point afin de permettre au contrôleur et au POD d'envoyer l'information au médium et ce dernier va la diffusé sur tout le réseau avec la même date de départ. C'est cette approche qui est choisie pour être implémentée sur notre simulateur.
Le médium sera donc structuré comme le montre la Figure 5.5.

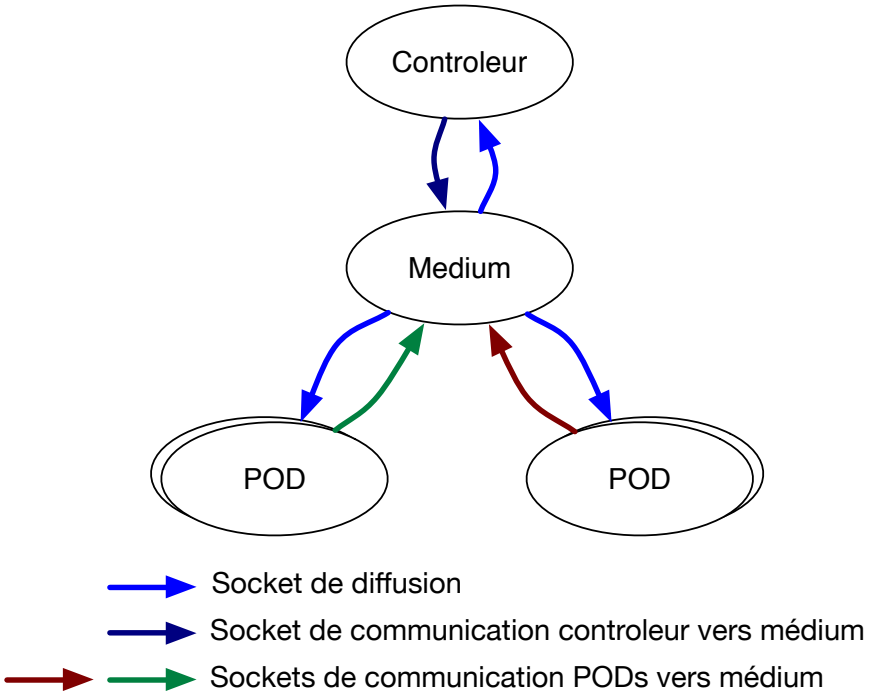


FIGURE 5.5 – Fonctionnement du médium en diffusion

Il y aura donc un thread entre le contrôleur et le médium, ce thread sera chargé de récupéré les trames du contrôleur et de les diffusé sur tout le réseau, et un thread pour chaque POD qui se chargeront de recevoir les trames des PODs et de les diffusé sur le réseau (cf. Figure B.5).

5.4.2 Processus de Stimulation

Ce processus regroupe le POD d'acquisition, le POD de stimulation et le modèle du genou, le regroupement de tous ces modules sur un processus nous permet communication plus simple entre ces différents modules, la Figure B.1 montre les différents interactions entre les threads de ce processus.

Modèle du genou : On utilise le modèle du genou décrit dans la sous-section 5.3.4, ce modèle du genou prend pour chaque muscle les principaux paramètres, la fréquence de stimulation, la largeur d'impulsion et le courant de stimulation. Il nous retourne comme variable de sortie la position articulaire ainsi que la vitesse du genou.
Lors de la stimulation nous utilisant une fréquence de stimulation de $25Hz$ et le courant de stimulation comme commande de chaque muscle. La position de repos du genou et $\theta = 1.57rad$ (jambe à vertical).

Programmation des POD : Au niveau de de la programmation, les différents POD sont programmés dans la même classe C++, le choix du type de POD se fait à l'initialisation de ce dernier.
On dispose de deux types de POD : Le POD d'acquisition d'information capteur, dans notre cas il nous fournit la position du genou, et le POD de stimulation qui applique une commande au modèle du genou. leurs fonctionnements reposent sur les thread suivants :

Thread de communication réseau : Ce Thread se charge de tout l'aspect communication réseau, il effectue le filtre d'adresse, l'aiguillage des requête (MAC, Appli), et traitement des trames reçu (extraction de la requête ...). La réception de la trame se fait via le socket de diffusion, ce socket est couplé à un timeout qui correspond à une période de commande, si le timeout est déclenché, le POD applique la dernière commande reçue (2 fois) puis si aucune commande n'est reçue au bout de la 3e on arrête la stimulation.
Ce thread se charge aussi de faire la gestion des communications de groupe (cf. sous-section 3.3.3) et le traitement de trame multicharge.
Lors de la réception d'une trame destinée à la couche application, ce thread déclenche le sémaphore du thread Application puis attend le renvoi de l'information de ce dernier.

Supervion communication : Ce thread exécute une fonction qui analyse le nombre d'échecs (réception de trame non destinée au POD ou timeOut socket) dans un intervalle donné, si le taux d'échecs consécutif dépasse le seuil limite, cette fonction arrête les tâches de communications et arrête la stimulation si c'est un POD Stim.

Thread Application : Ce thread attend le sémaphore du thread gestion de communication, puis selon le type de POD il effectue les tâches suivantes :
- Si POD d'acquisition, le thread déclenche via sont sémaphore le thread d'intégration du modèle du genou, attend la fin du calcul et retourne la valeur de la position et vitesse à au thread de gestion réseau.
- Si POD de stimulation, le thread déclenche l'intégration du modèle du genou puis met à jour la valeur des commandes pour la prochaine exécution, quand ce dernier c'est bien déroulé un sémaphore est rendu à au thread de gestion réseau et ce dernier renvoie l'acquittement. Les commandes de stimulation sont envoyées selon le sens du mouvement, pour un mouvement dans le sens négatif (cf. Figure 5.3b) la commande est envoyée au muscle quadriceps, et pour un mouvement positif, la commande est envoyée au muscle Ischiojambier.

On note aussi que lors de déclenchement du calcul de l'intégrateur numérique du modelé du genou, un mutex est activé afin de permettre l'avancement de l'intégration numérique en exclusion mutuelle entre les PODs..

Thread d'émission : Ce thread est déclenché par la gestion du réseau lorsqu'on a besoin d'envoyer une trame sur le réseau, avant l'envoi ce thread vérifie si le POD a bien un droit de parole (individuel ou groupe) puis envoie la trame sur le réseau selon le type de DP :
- Si DPI : Le thread envoi directement l'information sur le réseau via le socket de communication POD vers médium, le trame et ensuite diffusée sur le réseau via le médium. Ce thread rend le sémaphore au thread gestion réseau lors du bon déroulement de l'envoi.
- Si DPG : Selon la position du POD dans le groupe (cf. sous-section 3.3.3) un décalage de *D*Priorité* et appliqué avant l'émission de chaque POD, ceci évite les collisions sur le réseau entre les différents PODs.

5.4.3 Processus du contrôleur

Ce process représente le contrôleur, il contient tous les threads qui permettent la gestion du réseau, la commande et la supervision. Pour l'instant, aucun ordonnancement régulé n'a été réalisé, le but et de réaliser une commande de stimulation en boucle fermée simple.

Le processus du contrôleur ce compose de :

Thread de gestion réseau : Ce thread lance l'initialisation de la communication avec le médium, puis attend une requête du thread commande, soit de récupération d'informations capteur, soit de stimulation.

— Si droit de parole individuel : À chaque envoi d'information à un POD, un retour est attendu (soit acquittement ou retour d'information) et un timer équivalent au temps de réponse du POD est enclenché, si aucun retour n'est reçu, on renvoi l'information, au bout de 2 tentatives on fait un arrêt d'urgence, les données reçues sont alors extraites et envoyées à la boucle de commande (cf. Figure B.2).

— Si droit de parole de groupe : On envoi une trame multi charge au PODs, et on attend le retour d'acquittement, l'attente se fait selon la position des PODs dans le groupe, le POD le plus prioritaire (priorité 0) est celui qui doit répondre le 1er, les autres POD répondront après avec un décalage temporel $D * Prio$, on utilise ce temps pour prédire les réponses et ainsi configure le timeout de nos sockets de réception (cf. Figure B.3).

On optimise aussi l'envoi en droit de parole individuel en envoyant la commande seulement au POD qui applique une commande sur un muscle (cf. Figure B.4), en effet lors d'un mouvement négatif du genou (cf. Figure 5.3b) seul le muscle agoniste est actionné, le POD qui gère le muscle antagoniste ne reçoit donc aucune commande. On prévoit aussi une trame de test de présence pour éviter l'arrêt d'urgence du POD.

Thread commande boucle fermée : Ce thread s'occupe de la commande de stimulation en boucle fermée, il envoie une requête de récupération d'information capteur, puis effectue le calcul de la commande (cf. section 4.1), et retourne la valeur de la commande à la couche MAC qui l'envoi aux PODs de stimulation. Dans notre cas on réalise une commande sur le modèle du genou, on utilise pour cela un PID (cf. section 4.1).

Le POD Gonio nous fournit la valeur de la position du genou, cette valeur est ensuite extraite de la position détirée pour calculer l'erreur. La commande est ensuite envoyée au thread de gestion réseau qui l'envoie au PODs concerné, un acquittement est ensuite attendu.
Tous les calculs sont réalisés dans une fonction à boucle infinie qui est activée par un signal produit par un timer, la valeur de ce timer règle l'intervalle d'échantillonnage de cette fonction.

Thread du scheduler : Ce Thread exécute la fonction d'ordonnancement (cf. sous-section 4.5.2) qu'il recalcule l'intervalle d'échantillonnage de la boucle de commande. Cet intervalle et calculé selon des critères comme la charge CPU ou réseau.
Faute de temps aucun ordonnancement régulé n'a été réalisé, mais tout le mécanisme permettant la mise en place de cette régulation a été implémenté. La fonction d'ordonnancement vient à chaque période d'échantillonnage modifier la valeur de la période du Timer qui exécute la fonction de la commande en boucle fermée.

Thread d'émission : Ce thread s'occupe de l'envoi d'information sur le médium.

5.5 Ordonnancement et optimisation sur le réseau

Afin d'effectué une commande en boucle fermée d'un genou, nous avant besoin de 4 PODs, un POD contrôleur, un POD de stimulation du muscle agoniste, un POD de stimulation du muscle antagoniste et un POD d'acquisition. La gestion du réseau du réseau ce fait par le contrôleur et aucun POD ne peut émettre s'il ne possède pas de droit de parole (individuel ou groupe) (cf. sous-section 3.3.3). Comme décrit dans la section 4.1, la commande en boucle s'effectue par les étapes suivant :

— Acquisition de l'information capteur, dans notre cas la position du genou.
— Calcul de la commande.
— Envoi de la commande aux actionneurs (PODs de stimulation).

Nous optant donc pour différents tests afin de choisir la solution la moins gourmande en ressources réseau et aussi la plus rapide. Nous avons donc effectué quatre possibilités.

5.5.1 Communication avec droit de parole individuel sans optimisation

Cette solution consiste à :

— Envoi une requête d'information capteur depuis le contrôleur.
— Traiter l'information reçue sur le POD d'acquisition, récupérer la valeur de la position du genou, puis l'envoyer au contrôleur.
— Le contrôleur reçoit l'information capteur et calcule la commande, puis envoie la commande destinée au POD de stimulation du muscle antagoniste, attend l'acquittement, puis envoie la commande destinée au POD de stimulation du muscle agoniste et attend l'acquittement

Les résultats obtenus montrent un délai de bout en bout moyen de $9.05ms$ et une charge réseau de $150trame/s$, les délais sont calculés sur le contrôleur avec la fonction POSIX clockgettime, l'horloge utilisée est le Clock MONOTONIC RAW pour évité tous ajustement de l'horloge par le système.
On remarque que le délai de bout en bout correspond, au délai du système réel utilisé à actuellement (cf. section 5.2) et qui ne comporte aucune optimisation réseau.
La courbe Figure 5.6 montre le chronogramme de cette communication.

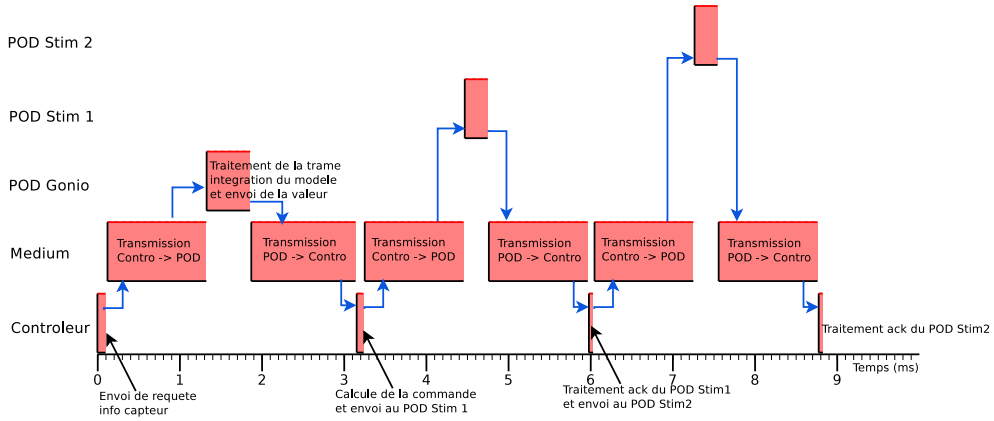


FIGURE 5.6 – Chronogramme d'activation des principaux processus du simulateur dans le cas d'un DPI sans Optimisation

5.5.2 Communication avec droit de parole individuel avec optimisation

Ce cas utilise le même procédé que le cas précédent sauf que la commande n'est envoyée qu'au POD qui réalise le mouvement, en d'autres termes si le mouvement est positif, seul le muscle antagoniste sera utilisé, la commande n'est donc envoyée qu'au POD qui commande ce muscle. Cependant, comme les POD ont un timeout d'attente, il est nécessaire d'envoyer de temps en temps (on a choisi une fois sur quatre) une trame de test de présence pour réveiller le POD et s'assurer qu'il est toujours disponible. Lors d'un changement du sens du mouvement, une mise à jour de la commande (commande nulle) et envoyé au POD dont le muscle n'intervient plus sur le mouvement.
Le résultat obtenu montre une nette amélioration du délai bout en bout moyen de $6.96ms$ et une charge réseau moins un importante de l'ordre de $112trame/s$.
Cette optimisation n'a aucun effet notable sur la qualité de commande, car la période d'échantillonnages et bien plus élevée que le gain en temps obtenu (cf section 5.6).

5.5.3 Communication avec droit de parole de groupe sans glissement et trame multi-charge

La communication de groupe avec trame multicharge, permet l'envoi d'une seule trame à un groupe de PODs, avec une charge destinée à chacun (cf. multicharge). La taille de la trame augmente, mais cette

dernière reste beaucoup moins élevée que la taille de deux trames, ce qui permet de réduire considérablement la charge du réseau.
On envoi donc une trame multicharge au groupe qui contient nos deux POD de stimulation (agoniste et antagoniste), et on attend l'acquittement de la part de ces deux POD. Comme l'explique la sous-section 3.3.3 la réponse d'un POD sur le réseau se fait selon sa position dans le groupe avec un retard de *D*Position* avec D l'intervalle qu'on laisse au POD précédent pour parlé.
Les résultats de cette communication sont un temps bout à bout moyen de $10ms$ et une charge réseau de $125trame/s$.

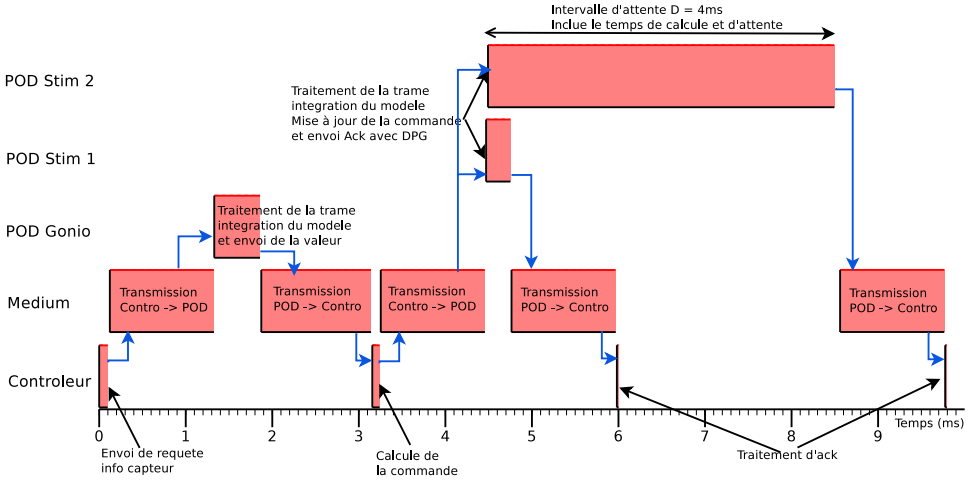


FIGURE 5.7 – Chronogramme d'activation des principaux processus du simulateur dans le cas d'un DPG sans glissement

5.5.4 Communication avec droit de parole de groupe avec glissement et trame multicharge

Comme on le remarque dans la communication de groupe sans glissement, le temps de réponse du dernier POD sera *D*NombredePOD*, ce qui peut être très long si le groupe contient plusieurs POD. Le glissement vient réglé ce problème, en effet son fonctionnement est simple, au lieu d'attendre un intervalle fixe pour chaque POD avant de parler, on écoute le réseau on vérifie si le POD qui nous précède a parlé, si c'est le cas on émet à alors notre réponse sans attendre l'intervalle complet. Cette méthode permet de réduire considérablement le temps bout en bout de la commande (de l'ordre de $7.64ms$) dans le cas d'un droit parole en groupe comme le montre la Figure 5.8.

5.5.5 Conclusion sur cette section

On remarque que l'optimisation apportée dans le cas d'un droit de parole individuel est la plus efficace, mais ce résultat à était obtenu avec un mouvement en sens unique qui ne nécessite l'utilisation que d'un seul POD. Dans le cas d'un suivi de trajectoire ou dans le cas d'utilisation de plus de deux POD, cette méthode s'avère moins efficace que le droit de parole de groupe avec glissement, car ce dernier n'a besoin que d'une seule trame pour envoyer des informations différentes à plusieurs POD.

5.6 Résultat de la commande en boucle fermée

Après la réalisation de notre simulateur, on utilise ce dernier afin d'implémenter la commande en boucle fermée étudiée en section 4.1. On implémente donc la fonction discrétisée du PID (cf. Équa-

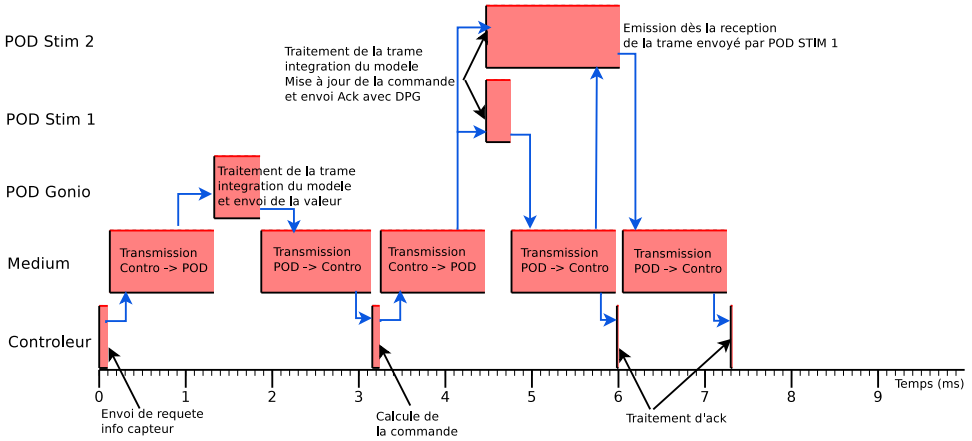
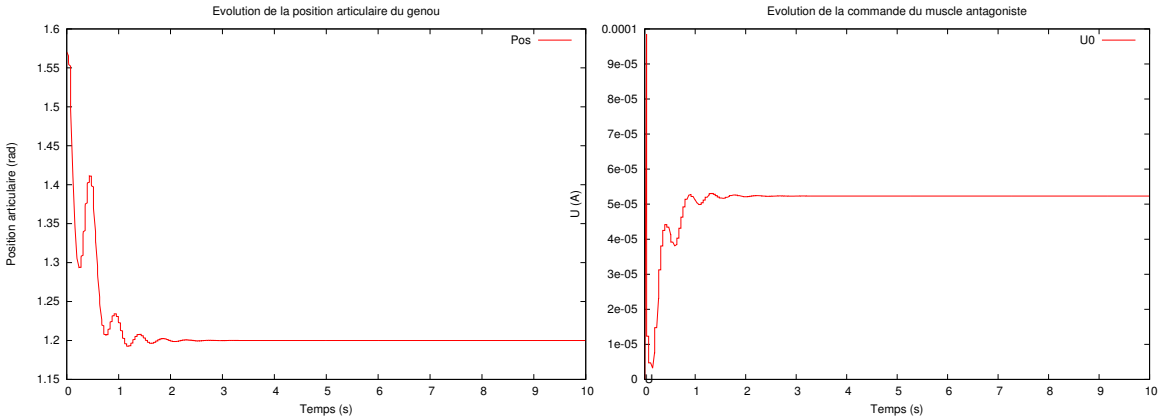


FIGURE 5.8 – Chronogramme d'activation des principaux processus du simulateur dans le cas d'un DPG

tion 4.2) sur notre contrôleur avec une période d'échantillonnage de 40ms, cette fréquence d'échantillonnage peut être variée en temps réel par le contrôleur d'ordonnancement. Notre levier d'actions sur les POD de stimulation est l'intensité du courant de stimulation, notre commande U esquivant alors au courant de stimulation envoyée au POD.
Tous ces résultats sont obtenus en mode communication de groupe, on note aussi que les différences entre types de communication que se soit optimisé ou non, ne modifie pas le profil de la commande, car la période d'échantillonnage de la commande et bien plus supérieur qu'au gain en temps obtenu avec différentes optimisations. Les paramètres de simulation choisis sont décrit dans la section B.4.

5.6.1 Expérimentation avec mouvement négatif

On obtient les résultats suivant pour la commande en positon articulaire du genou :



(a) Evolution de la position articulaire du genou (b) Commande appliquée au muscle antagoniste

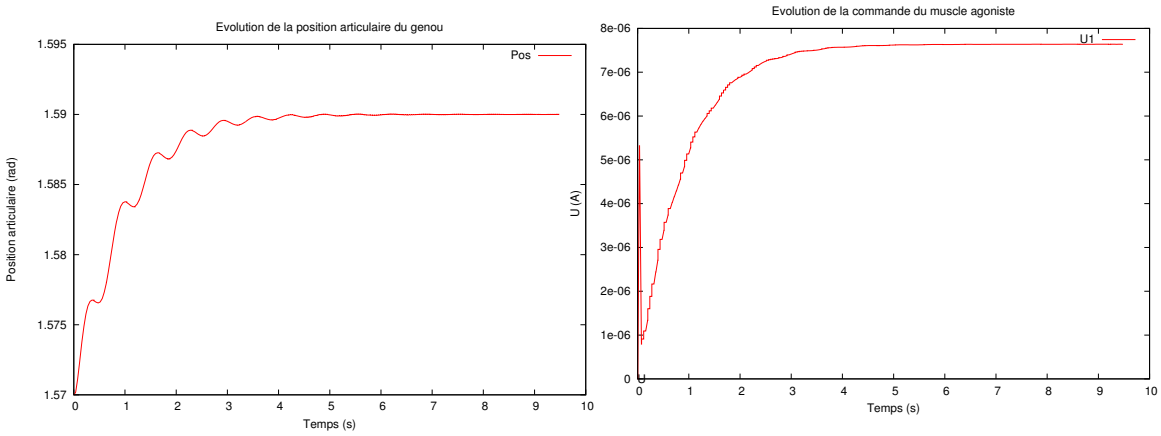
FIGURE 5.9 – Experimentation d'un mouvement négatif du genou

On remarque que la position du genou converge bien vers la position désirée et reste dans cette position, les oscillations persistantes sont dues au faible amortissement du modèle en boucle ouverte, mais aussi à l'application de la stimulation avec une fréquence de 25hz ce qui pour effet de ne pas appliquer le courant en continu sur le muscle. On note aussi que notre modèle est fait pour supporter que de petits mouvements, et dû aux saturations de la commande et à la faible fréquence de stimulation la position maximale que l'on peut atteindre est 0.97rad
La Figure 5.9b montre la commande appliquée au muscle antagoniste, car c'est ce dernier qui actionne

le muscle lors d'un mouvement négatif. La commande appliquée au muscle agoniste dans ce cas est nulle.

5.6.2 Expérimentation avec mouvement positif

Dans cette seconde configuration en garde les mêmes paramètres et l'on modifie la position désirée à 1.59 pour un mouvement positif : On comme dans le cas précédent, le on converge bien la position dé-



(a) Evolution de la position articulaire du genou

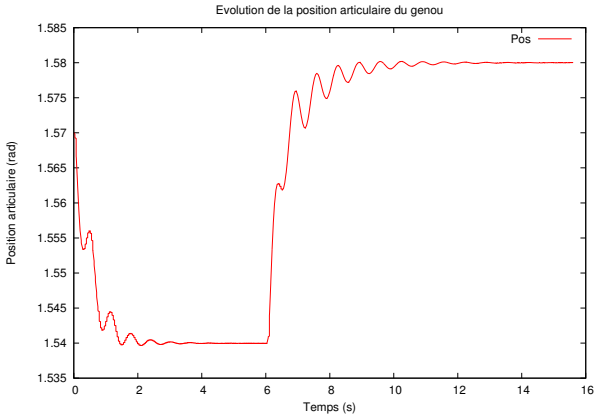
(b) Commande appliquée au muscleCommande appliqué au muscle antagoniste agoniste

FIGURE 5.10 – Experimentation d'un mouvement positif du genou

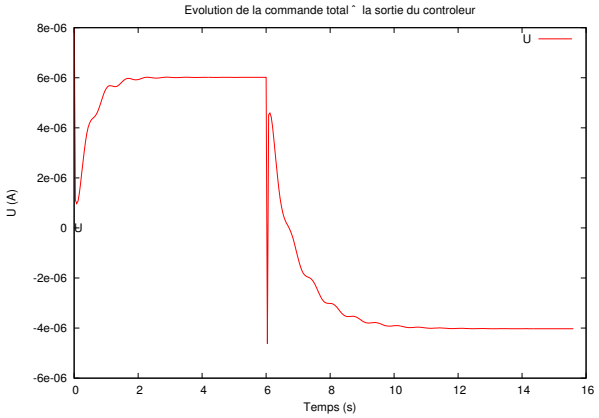
sirée qui est cette fois positif, et on remarque que le muscle utilisé pour ce mouvement et bien le muscle agoniste et la commande envoyée au POD du muscle antagoniste est nulle.

5.6.3 Expérimentation avec mouvement combiné

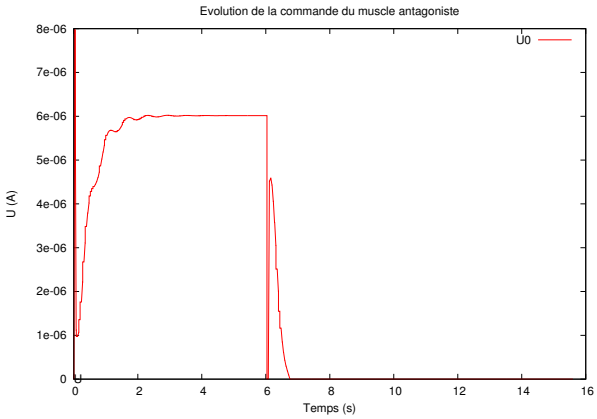
Dans cette 3e expérimentation, nous allons effectué un mouvement négatif vers 1.54*rad* puis un mouvement positif vers 1.58*rad* pour voir la permutation de la commande sur les deux PODs de stimulation.



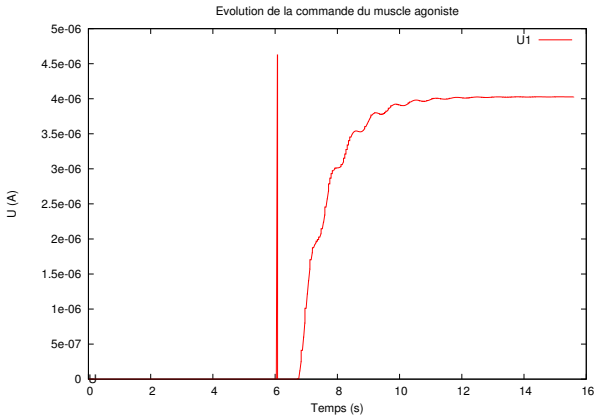
(a) Evolution de la position articulaire du genou.



(b) Commande globale à la sortie du contrôleur



(c) Commande du muscle antagoniste



(d) Commande du muscle agoniste

FIGURE 5.11 – Expérimentation d'un mouvement combiné

Comme le montre la Figure 5.11, on voit bien que le genou atteint bien successivement les positions désirées, on remarque aussi que la commande globale et bien répartie sur les deux PODs de stimulation selon la nature du mouvement.

5.6.4 Conclusion

Les résultats démontrés dans cette section montrent que la commande en boucle fermée sur un membre stimulé (dans notre cas un genou) fonctionne parfaitement, quelque oscillation persiste, mais ces dernières sont dû au modèle utilisé. Pour le moment, la commande qu'on a implémentée permet d'atteindre un point désiré, une commande plus avancée avec une génération de trajectoires dans l'espace cartésien et un modèle plus robuste permettra de concevoir des systèmes qui permettront à un patient par exemple de pouvoir faire des gestes complexes qu'il n'a plus la faculté de faire, ou la possibilité dans des cas de rééducation de permettre la réalisation de mouvement complexe et contrôlé sur un patient.

Chapitre 6

Conclusions et perspectives

6.1 Conclusions générales

Ce stage m'a permis de découvrir des applications médicales de l'informatique et d'engranger un certain nombre de connaissances relatives aux systèmes temps-réel et à la simulation dans le domaine de la stimulation électro-fonctionnelle.
Il m'a aussi permis d'acquérir une méthodologie d'étude d'un sujet scientifique, en commençant par la documentation sur le sujet, l'étude des besoins, et les analyses scientifiques avant d'aborder les étapes de développement et d'implémentation.
Il m'a aussi permis de découvrir de nouveaux outils et méthodes de travail et d'améliorer mes compétences techniques. Il m'a permis notamment d'acquérir la méthodologie de l'étude bibliographique, la rédaction de rapports sur Latex et la maîtrise d'outils de travail collaboration comme le logiciel de gestion de versions(Subversion) et le système de gestion de développement collaboratif de logiciel "gforge".
Il a aussi été l'occasion de découvrir de nouveau programmes et API de développement tels que Orccad et les librairies de développement temps réel de POSIX et d'améliorer mais compétences en programmation C/C++.

Les principaux objectifs fixés en début de stage étaient l'étude du fonctionnement du système Vivaltis existant ainsi que la proposition et l'étude de nouvelles approches de commande, d'ordonnancement et de communication sur le système.
Cette étude préliminaire a conduit à la proposition de développement d'un simulateur et l'utilisation de ce dernier pour simuler un système de stimulation électrique distribuée avec une commande en boucle fermée et valider les approches proposée.

À ce stade, la majorité des objectifs fixés ont été validés. Certaines fonctionnalités restent manquantes telles que la commande de régulation de la fréquence d'échantillonnage des tâches et l'ordonnancement (m,k)-firm sur le réseau, par contre la structure ouverte et les fonctions permettant cela on été réalisées.

6.2 Perspectives

Plusieurs élargissements peuvent être formulés dans la continuité du projet.
Il faudra d'une part, améliorer le modèle du médium afin de se rapprocher d'un médium physique réel. Pour ce faire il faudra ajouter les perturbations sur le réseau et les pertes.
D'une autre part, il faudra réaliser une modélisation plus approfondie du médium afin d'inclure des paramètres physiques tels que la puissance d'un signal émis, la propagation des ondes électromagnétiques dans l'espace...
Ces amélioration nous permettront de réaliser des régulations sur la puissance du signal par exemple, ce

qui aura un grand impact sur la consommation énergétique des POD physiques.

Il faudra aussi inclure les spécifiés du protocole STIMAP. Le modèle du protocole implémenté dans notre simulateur reproduit les principes de base de ce protocole comme la communication à intervalle glissant, et les droits de parole, mais ne réagit pas exactement comme le protocole STIMAP réel.
Pour remédier à cette lacune, plusieurs solutions sont possible :

- Une première solution serait d'implémenter le vrai programme du protocole STIMAP dans notre simulateur et ainsi faire du "Software in the loop", cette solution serait purement logiciel.
- Une seconde solution serait d'ajouter à notre simulateur des coupleurs réseau physique avec le Pile protocolaire STIMAP et ainsi faire du "Hardware in the loop" et permettre l'ajout d'un réalisme à la partie communication.

Enfin, l'implémentation de commandes plus avancées et la possibilité d'ordonnancer différentes commandes sur un même contrôleur permettront l'ouverture à de nouvelles perspectives de commande dans le domaine de la rééducation et du mouvement artificiel chez les patients atteints de déficience du système sensori-moteur.

Bibliographie

[ABPGS10] Soraya Arias, Florine Boudin, Roger Pissard-Gibollet, and Daniel Simon. Orccad, robot controller model and its support using eclipse modeling tools. *5th National Conference on Control Architecture of Robots*, May 2010. http://hal.inria.fr/inria-00482559.

[Ben09] Mourad Benoussaad. *Protocole d'identification sous FES et synthèse des séquences de stimulation chez le blessé médullaire*. PhD thesis, University Montpellier II, 2009. http://tel.archives-ouvertes.fr/tel-00452009.

[BKEF14] Abir Ben Khaled El Feki. *Simulation temps-réel distribuée de modèles numériques : application au groupe motopropulseur*. PhD thesis, Université de Grenoble, 2014.

[CA10] Ye-Qiong Song Christophe Aubrun, Daniel Simon. *Co-design Approaches for Dependable Networked Control Systems*. ISTE Ltd and Wiley, Inc, 2010.

[DASM13] Sylvain Durand, Anne-Marie Alta, Daniel Simon, and Nicolas Marchand. Energy-aware feedback control for a h.264 video decoder. *International Journal of Systems Science*, 2013. DOI :10.1080/00207721.2013.822607.

[Dur00] D.M. Durand. *The Biomedical Engineering Handbook*, chapter Electric Stimulation of Excitable Tissue, pages 17.1–17.20. CRC Press and Springer, second edition, 2000.

[EHA00] Johan Eker, Per Hagander, and Karl-Erik Arzén. A feedback scheduler for real-time controller tasks. *Control Engineering Practice*, 8(12) :1369–1378, Janvier 2000. DOI :10.1016/S0967-0661(00)00086-1.

[Fat10] Thomas Fattal. Etude et métrologie d'un sipositif de stimulation électro-fonctionnelle sans fil. Rapport de stage iut, Université Montpellier 2, 2010.

[Par08] Olivier Parodi. *Simulation hybride pour la coordination de véhicules hétérogènes au sein d'une flottille*. PhD thesis, University Montpellier II, 2008. http://tel.archives-ouvertes.fr/tel-00373347/.

[RBD+13] Eric Rutten, Jérémy Buisson, Gwanael Delaval, Florent de Lamotte, Jean-Philippe Diguet, Nicolas Marchand, and Daniel Simon. Control of autonomic computing systems. *ACM Computing Surveys*, 2013. submitted.

[Sim11] Daniel Simon. Hardware-in-the-loop test-bed of an unmanned aerial vehicle using orccad. *6th National Conference on Control Architecture of Robots*, May 2011. http://hal.inria.fr/inria-00599685.

[SSS14] Daniel Simon, Ye-Qiong Song, and Olivier Sename. Conception conjointe commande-ordonnancement. In Maryline Chetto, editor, *Ordonnancement dans les systèmes temps réel*, pages 293–324. ISTE Editions, June 2014.

[Tou11] Mickael Toussaint. *Architecture sans-fil de mesure et de stimulation electro-fonctionnelle externe : des concepts aux applications*. PhD thesis, University Montpellier II, 2011. confidential.

Annexe A

Communication

A.1 Format de trame en couche MAC

Cette trame se compose de deux parties distinctes (cf. figure A.1). La première partie est composée du Preambule, des champs de source et de destination et de l'entête. La deuxième partie de la trame comporte la charge, qui dépend de l'opération demandée par l'utilisateur de l'ordinateur.

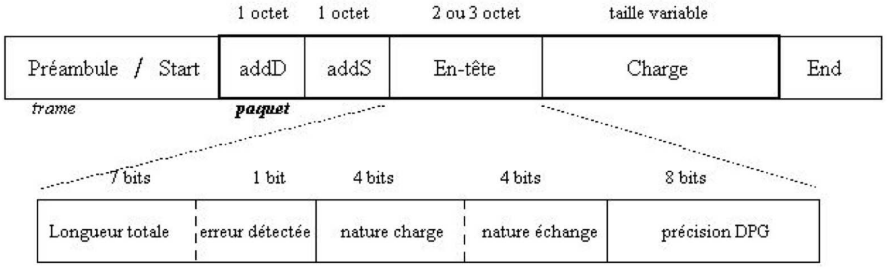


FIGURE A.1 – Format d'une trame au niveau de la couche MAC [Fat10]

Adressage : Dans notre topologie réseau, il est nécessaire de préciser l'adresse logique de destination de la trame à diffuser afin de permettre au(x) nœud(s) récepteur(s) de se reconnaître. Le champ addD comportant l'adresse logique de destination a donc été prévu à cet effet. Quant au champ addS, son rôle est d'indiquer l'adresse logique source de la trame afin que le destinataire puisse identifier l'émetteur et éventuellement lui répondre. Un adressage est donc défini sur 8 bits, soit 256 adresses possibles. L'adresse « 0 » est réservée au contrôleur. Pour la communication en unicast, les adresses « individuelles » utilisées vont de « 1 » à « 126 », l'adresse « 127 » est réservée à une procédure spéciale d'association des unités (cf. section A.2). On utilise le reste des adresses pour la diffusion à un groupe de nœuds (multicast) ; si les adresses de groupe sont comprises entre « 128 » et « 254 », on réserve l'adresse « 255 » pour la diffusion globale d'une trame à tous les nœuds (broadcast). Cet adressage permet donc de désigner de manière exclusive (avec une identification unique), 126 unités différentes et un contrôleur, ainsi que 127 groupes de ces unités. Ainsi, la première fonction de la couche MAC est de filtrer les trames entrantes suivant les adresses individuelles ou de groupe (cf. section A.1.1).

En-tête : La suite de la trame comprend les champs de l'en-tête qui est de longueur variable. Deux ou trois octets suffisent pour exprimer la longueur du paquet, la nature de la charge ainsi que la nature de l'échange spécifiant les acquittements et le type d'accès au médium. Le champ longueur totale permet de définir la longueur du paquet en nombre d'octets et le champ erreur détectée permet lui d'indiquer, lors de la réponse d'une UR au contrôleur, qu'une erreur est survenue sur cette UR.

Nature charge : ce champ définit le type de charge porté par le paquet. Il permet donc de déterminer si la charge doit être décodée par la couche MAC ou la couche Application. En effet, il peut

y avoir des requêtes pour la configuration du nœud dans le réseau (abonnement à un groupe, affectation d'une adresse) et des requêtes propres au processus applicatif. Ainsi pour ajouter des fonctionnalités à la pile protocolaire il suffit de définir un nouveau type de charge. Du coup, les autres charges applicatives ne sont pas modifiées et une unité sera capable de détecter une charge inconnue et de réagir en conséquence. Les données nécessaires à l'exécution de la requête indiquée sont contenues dans le champ charge et dans le cas d'une réponse on y retrouvera les données retournées au contrôleur suivant la requête exécutée.

Nature de l'échange : La nature de l'échange est un champ qui permet :

— Du contrôleur vers l'USR : d'envoyer des informations de droit de parole et de gestion des acquittements.
— De l'USR au contrôleur : de préciser si l'opération envoyée s'est correctement déroulée, et de quelle sorte est la réponse.

bit 1	bit 2	bit 3	bit 4
ACK	Précision ACK	DP	Précision DP

TABLE A.1 – Champ "Nature de l'échange" [Fat10]

Dans le sens contrôleur vers unité :

— Bit 1 : ACK, demande d'acquittement.
— Bit 2 : Précision de l'acquittement (trame ou application).
— Bit 3 : DP (Droit de Parole), par ce drapeau, le contrôleur autorise le nœud à parler.
— Bit 4 : Précision du DP (Individuel ou Groupe).

Dans le sens unité vers contrôleur :
L'unité répond à la demande effectuée par le contrôleur. En supposant que ce dernier ait demandé un acquittement, deux cas sont possibles :
Demande d'acquittement application ou demande d'une réponse de l'unité (i.e. il faut attendre que la requête soit exécutée) :
– Bit 2 = 1 si l'exécution de la requête est positive (exécution correcte).
– Bit 2 = 0 si l'exécution de la requête est négative (dans le cas d'une requête incohérente ou d'une erreur par exemple). Un code d'erreur est retourné si la couche MAC a la place de l'ajouter, sinon elle signale juste qu'une erreur est détectée.
Demande d'acquittement de réception d'une trame :
– Bit2 =1 si la trame a bien été reçue. – Notons que les acquittements de niveau trame sont toujours positifs, les trames erronées étant filtrées par le bloc communication. C'est l'absence d'acquittement par une unité qui permet au contrôleur de déduire une non-réception de trame.
Le premier bit est utilisé pour indiquer si la trame de retour est le résultat de la concaténation d'un acquittement trame et d'une réponse (bit alors mis à « 0 ». Les troisième et quatrième bits ne sont pas utilisés dans ce sens de communication.

Précision du droit de parole de groupe : En présence d'un droit de parole de groupe (DPG), quelques caractéristiques doivent être précisées (cf. Tableau A.2) :

— Priorité de départ : Indique la priorité de la première USR qui parlera au sein d'un groupe.
— Glissement : Indique si le Droit de Parole à Intervalles glissants a été effectué ou non. Le bit Glissement est mis à 0 si le glissement est désactivé. Le bit Glissement est mis à 1 pour que le glissement soit géré.
— Motif du glissement : Indique dans quelle occasion le glissement doit se faire. Le bit Motif du Glissement est mis à 0 lorsqu'on doit glisser si l'USR précédente n'a pas parlé pendant son demi-intervalle. Le bit Motif du Glissement est mis à 1 lorsqu'on doit glisser si l'USR précédente a parlé pendant son demi-intervalle.

4 bits	1 bit	1 bit	2 bits
Priorité de départ	Glissement	Motif glissement	inutilisés

TABLE A.2 – Champ "Précision du DPG" [Fat10]

A.1.1 Couche MAC des unités réparties

Au niveau des unités réparties, les différentes tâches de cette couche réalisent les opérations suivantes :

Filtre d'Adresse logique : elle constitue la partie entrante de notre couche MAC (cf. figure A.2). Elle a pour but de déterminer si une trame qui vient d'être reçue par la couche physique concerne l'unité. Cette tâche extrait l'adresse logique de destination du paquet reçu et doit la comparer à l'adresse individuelle de l'unité, à ses adresses de groupe contenues dans sa « table des groupes » et à l'adresse de broadcast. Si l'unité est concernée par la trame reçue, elle le notifie à la tâche « Aiguilleur MAC ». Si ce n'est pas le cas ou à la fin du traitement de la trame, celle-ci est détruite.

Aiguilleur MAC : Sa fonction est de décoder l'en-tête du paquet et d'aiguiller la charge selon sa nature, vers le sous-module de « configuration des nœuds » ou vers l'interface MAC/Application ». Seules les charges concernant la configuration des nœuds et dont la nature vaut « 7 », sont traitées par la couche MAC, les autres étant relatives à la couche Application.

Interface MAC/Application : Comme pour la tâche Aiguilleur MAC, elle a pour fonction d'aiguiller la charge vers le sous-module concerné de la couche Application.

Configuration des Nœuds : Cette tâche s'occupe de décoder les charges spécifiques à la couche MAC. Les opérations à effectuer comprennent notamment l'affectation de l'unité à un contrôleur, l'abonnement à un groupe d'unités, le retrait d'un groupe (désabonnement), la « rotation de groupe », la définition de l'intervalle de temps D pour la réponse de groupe, ainsi que l'enregistrement et l'identification d'un numéro de fabrication.

Gestion des Interactions : Cette tâche se charge de construire les paquets à retourner, en l'occurrence les acquittements, et d'y introduire le cas échéant la réponse suite à l'exécution d'une requête par la tâche « Configuration des Nœuds » ou par la couche Application. Gestion des Erreurs : Elle garde en mémoire les dernières erreurs (sous forme d'un vecteur d'erreur) qui se sont produites pour les couches MAC et Application.

Accès au Médium : Dès qu'une nouvelle trame est reçue, cette tâche récupère les paramètres du droit de parole qui est attribué à l'unité. Elle met à jour et gère alors son droit de parole en conséquence. C'est à ce niveau que sera effectuée la gestion temporelle et comportementale de l'unité lors d'une réponse de groupe.

Émission : Elle gère l'émission des paquets en interagissant avec l'interface Physique (envoi et attente respectivement des messages DemandeEmission et FinEmission). Pour cela, il lui faut récupérer une trame d'acquittement à émettre venant de la tâche Gestion des Interactions et avoir l'autorisation de la tâche Accès au Médium liée au droit de parole en cours, pour demander à l'interface Physique d'émettre sur le médium. Elle se charge aussi de concaténer les acquittements trame et application le cas échéant.

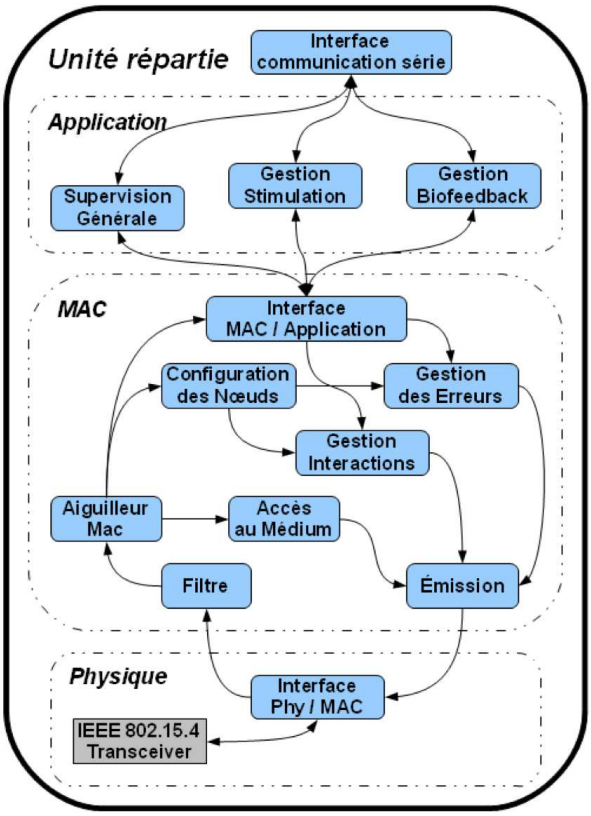


FIGURE A.2 – Représentation schématique détaillée des modules de la pile protocolaire au niveau de l'UR

A.2 La tâche de configuration des nœuds

Ce sous-module de la couche MAC des unités (cf. figure A.2) est responsable des paramètres de configuration de l'unité dans notre réseau. La trame de configuration des nœuds comporte un « code opération » qui permet l'identification des opérations à effectuer et la charge de données nécessaires à l'exécution. Voici les principales opérations de la tâche avec leur code opération :

- $0x00$: Identification des nœuds (test de présence) ;
- $0x01$: Abonnement à un groupe ;
- $0x02$: Retrait d'un groupe (désabonnement) ;
- $0x06$: Association de l'unité ;
- $0x08$: Configuration de D ;
- $0x09$: Configuration de RTT.

On retrouvera généralement dans la charge une adresse de 8 bits dans le cas d'un abonnement à un groupe ou d'un retrait d'un groupe ou encore la valeur de D ou de RTT.

Association de l'unité : En premier lieu nous avons besoin d'associer nos unités à un contrôleur. La procédure d'association consiste à attribuer à une unité répartie une adresse individuelle, une couleur et une priorité principale ainsi qu'à lui indiquer l'identifiant réseau et le canal de transmission par défaut. À sa mise en route, par un appui prolongé (3 secondes) du bouton de marche, l'unité entre temporairement (5 secondes) dans un mode par défaut où elle se configure avec les valeurs de paramètres suivantes :

- Adresse individuelle : $0x7F$
- Couleur : blanc clignotant
- Identifiant réseau : $0xFACE$
- Canal de transmission : $2,480GHz$

Durant ce mode d'association, le contrôleur choisit une adresse individuelle et une priorité principale disponible par rapport à sa table de paramètres réseau, se configure ensuite avec l'identifiant réseau et le canal de transmission par défaut puis envoie à l'unité ses nouveaux paramètres. Après réception, l'unité retourne un acquittement application pour indiquer la validité et la prise en compte des valeurs attribuées ; ce n'est qu'ensuite qu'elle se configure avec ses nouveaux paramètres pour être reconnue et communiquer sur le réseau. Sur le retour de l'acquittement, le contrôleur peut enregistrer les paramètres réseau de la nouvelle unité dans sa table et reprendre sa configuration initiale. Cette étape est essentielle dans la mesure où plusieurs architectures de SEF (ensembles contrôleur et ses unités) peuvent cohabiter dans une même zone, comme un cabinet de kinésithérapie par exemple.

Identification des nœuds : La phase d'identification consiste à vérifier la présence d'un nœud. Le contrôleur interroge simplement une unité qui doit alors répondre. Sans réponse de l'unité au bout d'un certain temps et éventuellement après un certain nombre de tentatives, on considère que le nœud adressé n'est pas présent. Il n'y a pas de charge de données pour cette opération.

Abonnement à un groupe : Permets d'inscrire un nœud dans un groupe. Pour ce faire, on vient noter, dans un des huit registres réservés à cet effet, l'adresse du groupe auquel on veut abonner le nœud, la priorité du nœud au sein du groupe ainsi que la taille du groupe.

Configuration de D : Permets de stocker la durée de l'intervalle D dans l'unité. Cette valeur associée avec celle de RTT sera utilisée pour le calcul du temps à attendre lors de l'une réponse de groupe.

Configuration de RTT : Permets de stocker la mesure de RTT dans l'unité. Contrairement à la valeur de D qui est commune pour toute une architecture pour tout échange au sein d'un groupe, la valeur de RTT calculée peut être propre à chaque unité indépendamment.

Retrait d'un groupe : Permets de supprimer une unité d'un groupe. L'adresse du groupe sur lequel porte le retrait est transmise à l'unité. Lors d'une tentative de suppression, si l'adresse de groupe ne fait pas partie de la table des groupes de l'unité, une erreur est retournée. Pour désabonner de tous les groupes un nœud désigné (« Reset »), on spécifie l'adresse « 0xFF », le nœud concerné doit alors purger sa table des groupes.

A.3 Codes opérations pour la stimulation sur les POD

- 0x01 : Configuration ;
- 0x02 : Lancement ;
- 0x03 : Réglage Amplitude ;
- 0x04 : Réglage Largeur d'Impulsion ;
- 0x06 : Arrêt ;
- 0x07 : État Modulation.

Annexe B

Implémentation

B.1 Organigrammes du Processus de stimulation

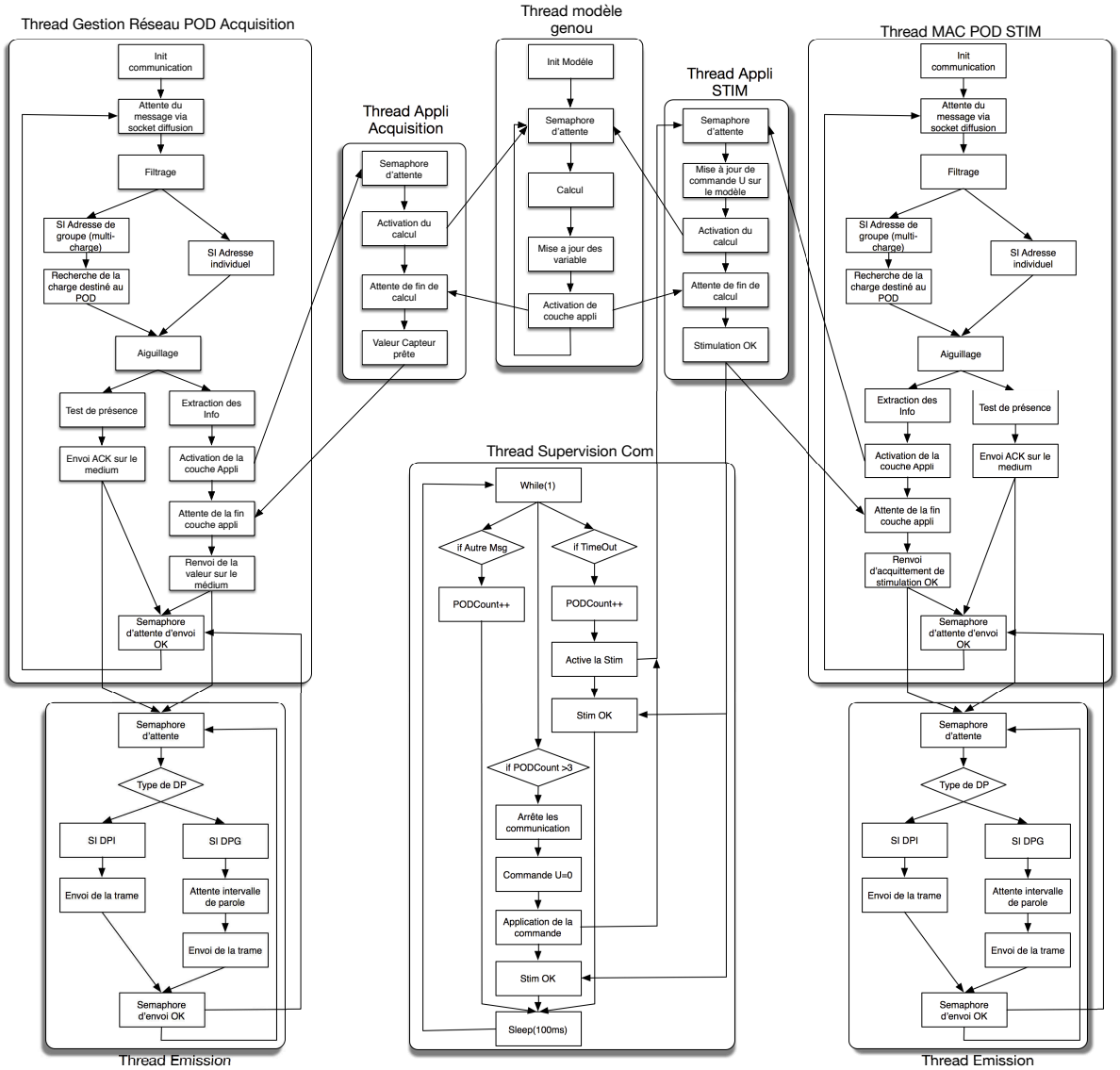


FIGURE B.1 – Fonctionnement simplifié du processus de Stimulation

B.2 Organigrammes de la gestion du réseau par le contrôleur

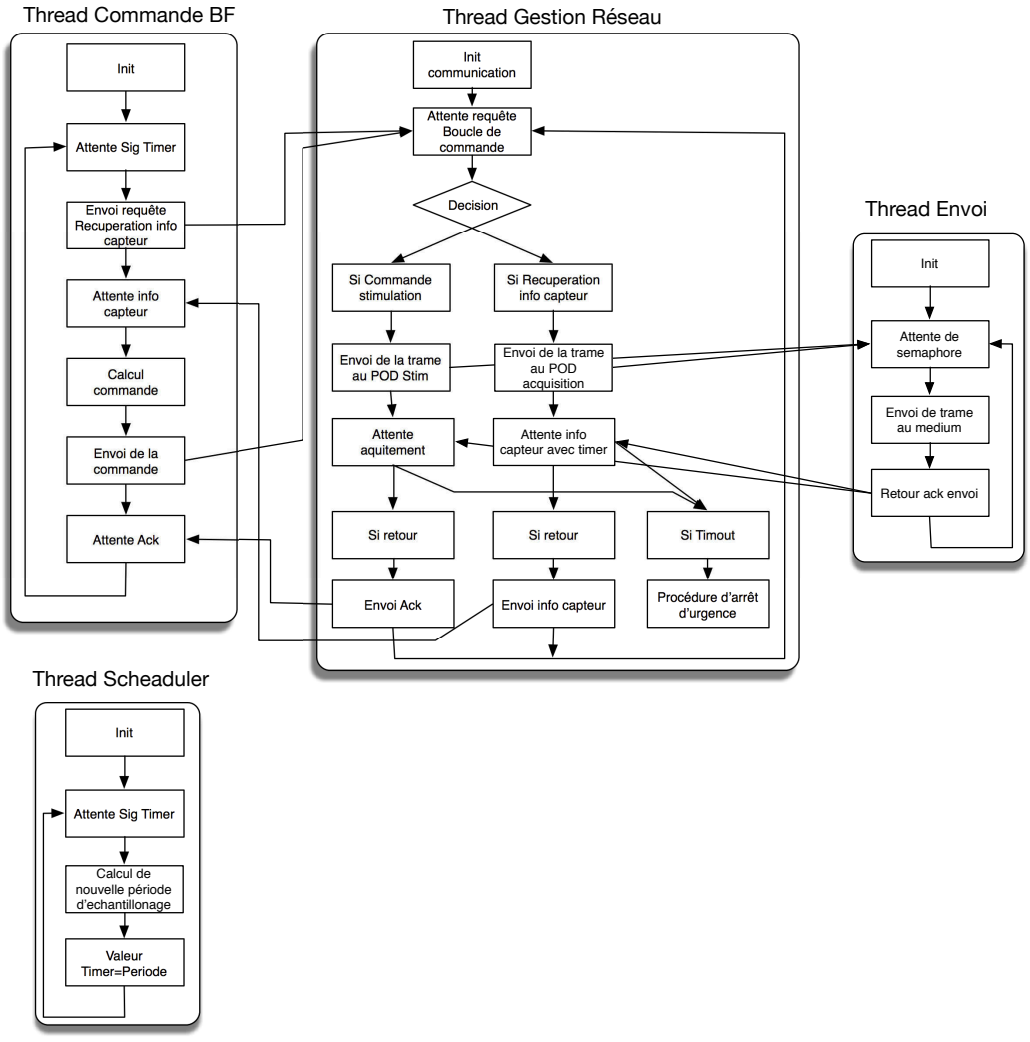


FIGURE B.2 – Fonctionnement simplifié du processus du contrôleur sans gestion du DPG

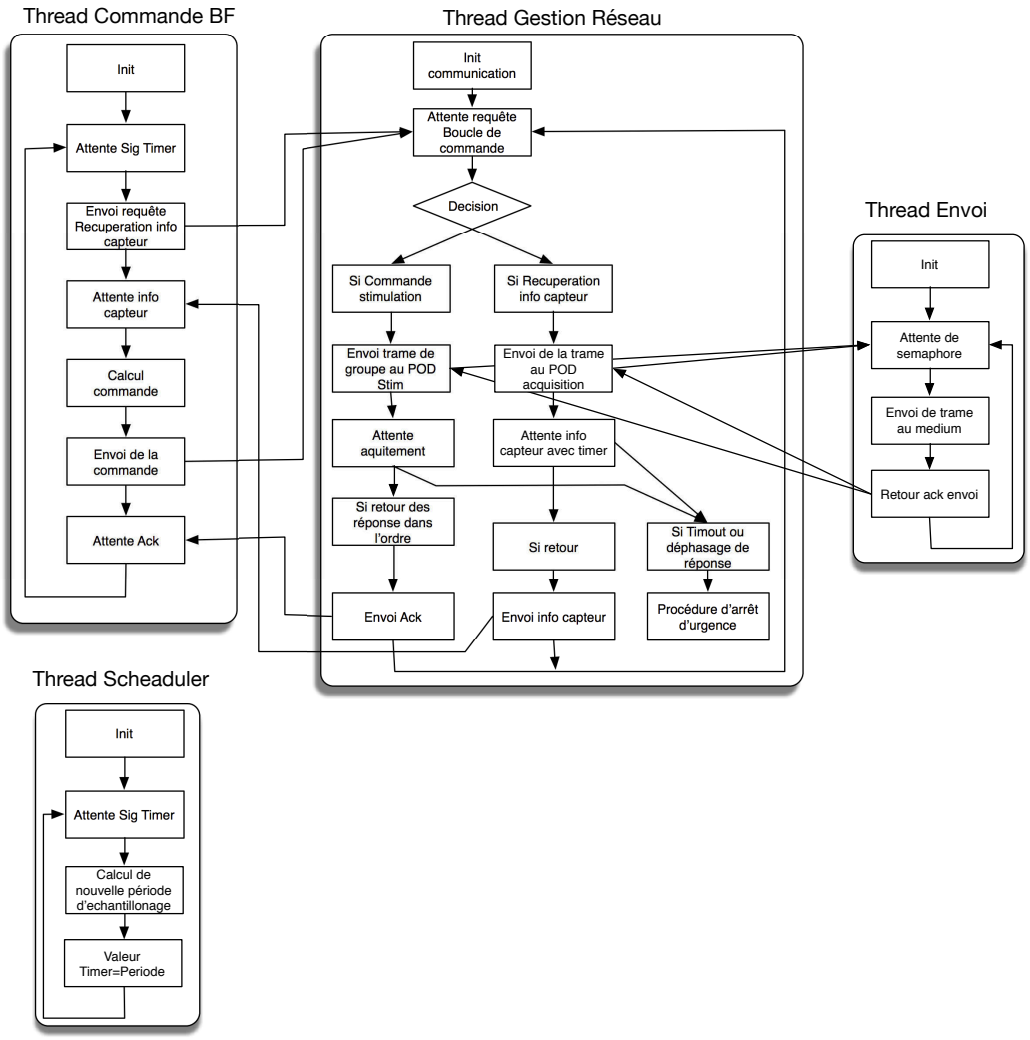


FIGURE B.3 – Fonctionnement simplifié du processus du contrôleur avec gestion du DPG

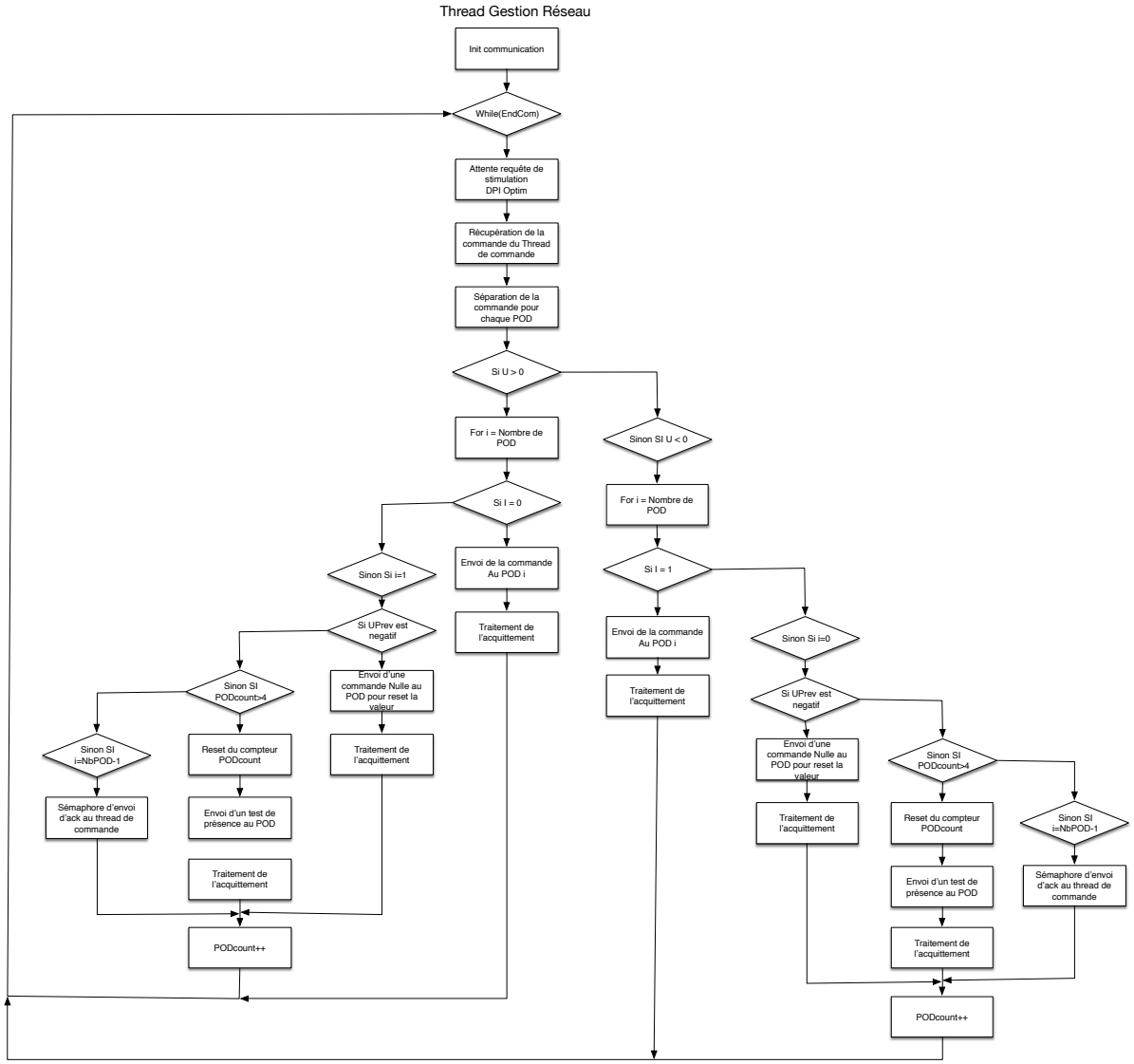


FIGURE B.4 – Fonctionnement du thread de gestion des communications en Mode DPI Optimisé

B.3 Organigramme du Médium

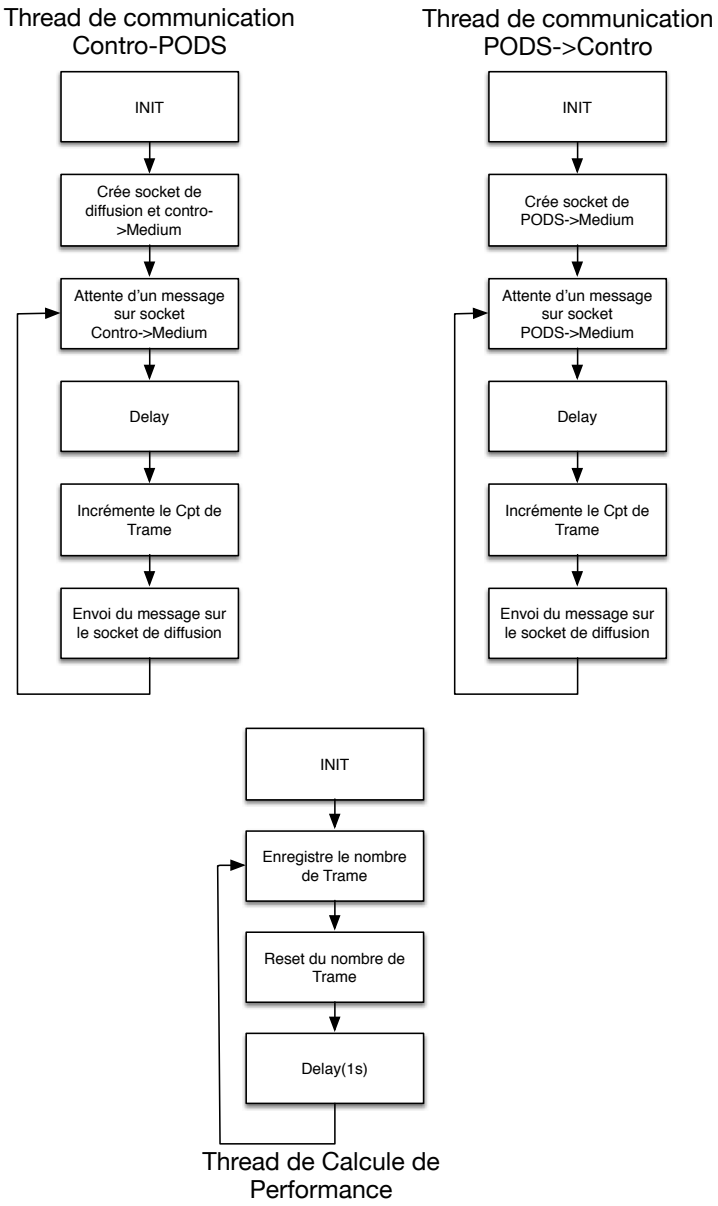


FIGURE B.5 – Fonctionnement simplifié du processus du Médium

B.4 Paramètres de stimulation lors de la simulation

B.4.1 Paramètre de la commande

Période d'échantillonnage T_s : $40ms$

Gain K : -10^{-5}

Période dérivée T_d : 0.12

Période intégrale T_i : 0.2

Position de départ : $1.57rad$

Position désirée : $1.2rad$

Fréquence de stimulation : $25Hz$

Saturation du courant de stimulation : $150uA$

Temps de simulation : $10s$

B.4.2 Paramètres du quadriceps

Fréquence de stimulation F : $25Hz$

Intensité maximale du courant $Imax_q$:

Largeur d'impulsion PW_q : 0.0003

Largeur d'impulsion maximal $PWmax_q$: 0.00042

Raideur maximale en N/m Km_q : 4238

Contrainte maximale en N Fm_q : 780

raideur du tendon en N/m Ks_q : 42382

longueur muscle (élément contractile) en m $Lc0_q$: 0.084

longueur tendon en m (l'élément série) $Ls0_q$: 0.346

paramètre de la relation force-length bl_q : 0.5

B.4.3 Paramètres de l'ischio-jambier

Fréquence de stimulation F : $25Hz$

Intensité maximal du courant $Imax_i$:

Largeur d'impulsion PW_i : 0.0003

Largeur d'impulsion maximal $PWmax_i$: 0.00042

Raideur maximale en N/m Km_i : 10000

Contrainte maximale en N Fm_i : 400

raideur du tendon en N/m Ks_i : 10000

longueur muscle (élément contractile) en m $Lc0_i$: $10.7e-2$

longueur tendon en m (l'élément série) $Ls0_i$: $38.5e-2$

paramètre de la relation force-length bl_i : 0.35

B.4.4 Paramètres du genou

Moment d'inertie rapporté au point de rotation en N.m2 (Système assimilé à une masse ponctuelle) J : 0.3

Rayon de la poulie du quadri en mètre r_q : 0.0497

Rayon de la poulie du ischio en mètre r_i : 0.0497

Masse concentrée en Kg M : 6.1

Position d'équilibre θ_0 : $1.57rad$

B.5 Fonction de recrutement

Ce modèle donne le nombre de fibres musculaires actives en fonction de l'amplitude I ou de la largeur d'impulsion *PW* appliquée. Il correspond au pourcentage d'unités motrices activées sous FES par rapport à la totalité des unités motrices. Le recrutement correspond à une sommation spatiale des forces, plus il y a de fibres activées plus la force sera grande. Le recrutement est fonction linéaire de l'amplitude I ou la largeur d'impulsion *PW*, mais la forme de la non linéarité est la même selon le paramètre modulé, ceci dit la modulation d'un seul des deux paramètres (*I* et *PW*) est suffisante pour contrôler le taux de recrutement et ainsi la force musculaire. La fonction retenue dans la thèse [Ben09] est une fonction sigmoïde modulée par le paramètre *PW* :

$$\begin{cases} \alpha(PW) = 0, \text{pour} PW \leq PW_{th} \\ \alpha(PW) = 0 = \frac{c_1}{1+\exp[c_2(c_3 - PW/PW_{max})\}}, \text{pour} PW > PW_{th} \end{cases}$$

Avec c_1, c_2 et c_3 les paramètres à identifier représentant respectivement l'amplitude du plateau, la pente et le point d'inflexion de la fonction sigmoïde. PW_{max} la largeur d'impulsion maximale et PW_{th} largeur d'impulsion seuil.