ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

PAR
Kevin LÉVESQUE-LANDRY

ÉVALUATION DE L'INTERGICIEL DE COMMUNICATION DDS POUR SON UTILISATION DANS LE DOMAINE AVIONIQUE

MONTRÉAL, LE 6 JANVIER 2015

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Jean-François Boland, directeur de mémoire
Département de génie électrique, École de technologie supérieure

M. Guy Bois, codirecteur de mémoire
Département de génie informatique et logiciel, École Polytechnique

M. Christian Belleau, président du jury
Département de génie mécanique, École de technologie supérieure

Mme. Gabriela Nicolescu, membre du jury
Département de génie informatique et logiciel, École Polytechnique

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 19 DÉCEMBRE 2014

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier certaines personnes qui m'ont donné leur soutien et leurs encouragements lors de ce projet de recherche.

Tout d'abord, mon directeur de recherche, M. Jean-François Boland, pour ses judicieux conseils ainsi que son soutien tout au long de ce projet.

M. Benoît Gilbert, pour m'avoir aidé à concevoir un plug-in de communication DDS pour le logiciel X-Plane. Son aide m'a sauvé de nombreuses heures de travail.

Tous les membres du projet CRIAQ AVIO509 pour leurs idées et suggestions qui ont grandement aidé au bon déroulement de ce projet de recherche.

Le conseil de recherches en sciences naturelles et en génie (CRSNG), le Mitacs ainsi que le consortium de recherche et d'innovation en aérospatiale au Québec (CRIAQ) pour leur soutien financier.

L'École de technologie supérieure pour me donner l'accès à de l'équipement de laboratoire ainsi qu'à un environnement de travail.

Mon collègue et ami Jean-Sébastien pour ses conseils, son aide ainsi que les bons moments que l'on a passés ensemble.

Ma famille et mes amis pour leurs encouragements et leur soutien dans la poursuite de mes études.

Finalement, j'aimerais remercier ma conjointe Alexandra que j'aime fort et qui m'a grandement encouragé dans ma réussite.

ÉVALUATION DE L'INTERGICIEL DE COMMUNICATION DDS POUR SON UTILISATION DANS LE DOMAINE AVIONIQUE

Kevin LÉVESQUE-LANDRY

RÉSUMÉ

Les aéronefs modernes doivent combler de plus en plus de fonctionnalités afin de satisfaire les besoins de la clientèle. De ce fait, les besoins en communications des systèmes avioniques sont grandissants. De plus, la portabilité et la réutilisabilité des applications sont des défis d'actualité dans le domaine avionique. De ce fait, ce projet de recherche vise à faire une évaluation de la technologie d'intergiciel de service de distribution de données (DDS) pour son utilisation dans le domaine avionique. Cette technologie permettrait de réduire la complexité des communications et faciliter la portabilité et réutilisabilité des applications grâce à son interface standardisée.

Dans ce projet de recherche, la norme DDS est tout d'abord étudiée pour cibler les fonctionnalités qui sont utiles au domaine avionique. Les différentes polices de qualité de services sont ainsi étudiées et dénotent la flexibilité de la technologie DDS.

Un intergiciel DDS est également évalué dans un environnement de laboratoire afin de mesurer l'impact de l'utilisation de cette technologie sur les performances de latence ainsi que sur l'utilisation de la bande passante. Les résultats montrent une faible augmentation de la latence moyenne lorsque l'intergiciel DDS est utilisé.

L'intergiciel DDS est également utilisé dans une étude de cas avec un AFCS (*automatic flight control system*) afin de quantifier les effets de son utilisation sur une application avionique. Les résultats montrent que l'utilisation de l'intergiciel DDS n'empêche pas l'AFCS d'atteindre la stabilité, mais qu'elle ralentit l'atteinte de cette dernière.

Finalement, une étude de cas est effectuée afin de valider que la technologie DDS peut être utilisée pour construire des systèmes redondants. Les résultats montrent que l'intergiciel DDS permet de faire de la redondance de réserve sans avoir un impact visible sur les performances du système redondant.

Mots-clés: DDS, intergiciel de communication, avionique, système automatique de contrôle de vol

ÉVALUATION DE L'INTERGICIEL DE COMMUNICATION DDS POUR SON UTILISATION DANS LE DOMAINE AVIONIQUE

Kevin LÉVESQUE-LANDRY

ABSTRACT

Modern aircrafts must include an increasingly amount of functionalities to satisfy the needs of the customers. Therefore, the communication needs of avionic system are growing. Furthermore, the portability and reusability of applications are current challenges of the avionic domain. Therefore, this research project aims to evaluate the data distribution service (DDS) middleware technology to use it in the avionic domain. This technology would allow to reduce the complexity of communications and ease the portability and reusability of applications with its standardised interface.

First of all, this project studies the DDS standard to find the functionalities that are useful for the avionic domain. Therefore, the different quality of service policies are studied and highlights the flexibility of the DDS technology.

A DDS middleware is also evaluated in a laboratory environment to measure the impact of using this technology on latency and bandwidth utilisation. The results shows a small raise of mean latency when using a DDS middleware.

Furthermore, a case study is also made with a DDS middleware and an avionic flight control system (AFCS) to quantify the effects of using this technology on an avionic application. The results shows that using a DDS middleware doesn't prevent the AFCS to reach stability but raises its stabilisation time.

Finally, one more case study is made to validate that the DDS middleware technology can be used to build redundant systems. The results shows that the DDS middleware allows to make backup redundancy without any visible impact on the redundant system performances.

Keywords: DDS, communication middleware, avionic, automatic flight control system

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AFCS	Automatic Flight Control System
AFDX	Avionics Full-Duplex Switched Ethernet
APEX	Application/Executive Interface
API	Application Programming Interface
ARINC	Aeronautical Radio Incorporated
BSP	Board Support Package
CCS	Common Core System
CDS	Cockpit Display System
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-the-Shelf
CPS	Cyber-Physical Systems
CRIAQ	Consortium de Recherche et d'Innovation en Aérospatiale au Québec
DCPS	Data-Centric Publish-Subscribe
DDS	Data Distribution Service
DLRL	Data Local Reconstruction Layer
EDP	Endpoint Discovery Protocol
FACE	Future Airborne Capability Environment
GPM	General Processing Module
GUID	Globally Unique ID
HIL	Hardware in the loop
HTTP	HyperText Transfert Protocol

IDE	Integrated Development Environment
IMA	Integrated Modular Avionics
IMA2G	Integrated Modular Avionics 2nd Generation
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LRU	Line Replaceable Unit
OMG	Object Management Group
ORB	Object Resquest Broker
PC	Personnal Computer
PDP	Participant Discovery Protocol
PID	Proportionnel Intégral Dérivé
OSI	Open System Interconnection
QoS	Quality of Services
RT-CORBA	Real Time Common Object Request Broker Architecture
RTOS	Real-time operating system
RTPS-DDS	Real-time publish-subscribe Data Distribution Service
SCARLETT	SCAlable & ReconfigurabLe Electronics plaTforms and Tools
SDK	Software Development Kit
SOAP	Single Object Access Protocol
SWaP	Reduced Space, Weight and Power consumption
TFTP	Trivial File Transfert Protocol
TOS	Type of service

UDP	User Datagram Protocol
VoIP	Voice over IP
WCET	Worst-Case Execution Time
WSN	Wireless Sensor Network
XML	Extensible Markup Language

LISTE DES SYMBOLES ET UNITÉS DE MESURE

e_{roulis}	Erreur en régime permanant du roulis (degré)
$t_{s\ altitude}$	Temps de stabilisation de l'altitude (s)
$e_{altitude}$	Erreur en régime permanant de l'altitude (pieds)
$t_{s\ vitesse}$	Temps de stabilisation de la vitesse (s)
$e_{vitesse}$	Erreur en régime permanant de la vitesse (nœuds)

INTRODUCTION

Les aéronefs modernes doivent combler de plus en plus de fonctionnalités afin de satisfaire les besoins de la clientèle. De plus, les normes strictes servant à la certification de ces aéronefs complexifient la conception de ces fonctionnalités. Ces dernières se retrouvent sous diverses formes telles que le divertissement des passagers, la navigation, le guidage, la stabilité, la gestion du carburant, les communications aériennes/terrestres et bien d'autres. De ce fait, les besoins en communications des systèmes avioniques sont grandissants. Bien que l'architecture IMA (*Integrated Modular Avionics*) vise à réduire la quantité de systèmes que l'on retrouve dans les aéronefs, il y a tout de même une augmentation des besoins en communications entre les différentes fonctions.

La conception de nouveaux aéronefs est de plus en plus complexe et coûteuse (St. Clair, 2009). La Figure 0.1 illustre l'augmentation de la complexité des systèmes avionique à travers l'augmentation continue du nombre de ligne de code dans les systèmes avionique depuis les années 1960. Cette figure illustre que si la tendance se maintient, d'ici 2020, la conception des nouveaux aéronefs ne sera plus abordable.

Une portion importante des coûts de conception des aéronefs provient des systèmes avioniques. En 2012, une étude (Bieber, 2012) a démontré que les coûts des systèmes avioniques représentent entre 35 à 40% des coûts totaux pour les aéronefs civils et plus de 50% pour les aéronefs militaires. Une partie de ces coûts peut être diminuée en améliorant la portabilité et la réutilisabilité des fonctions avioniques. Toutefois, le couplage entre les fonctions et les modes de communications qu'elles utilisent est un obstacle important pour ce type d'amélioration. La portabilité et la réutilisabilité sont des défis d'actualité dans le domaine avionique et c'est pourquoi le consortium FACE (*Future Airborne Capability Environment*) aborde justement le problème (FACE Consortium, 2013).

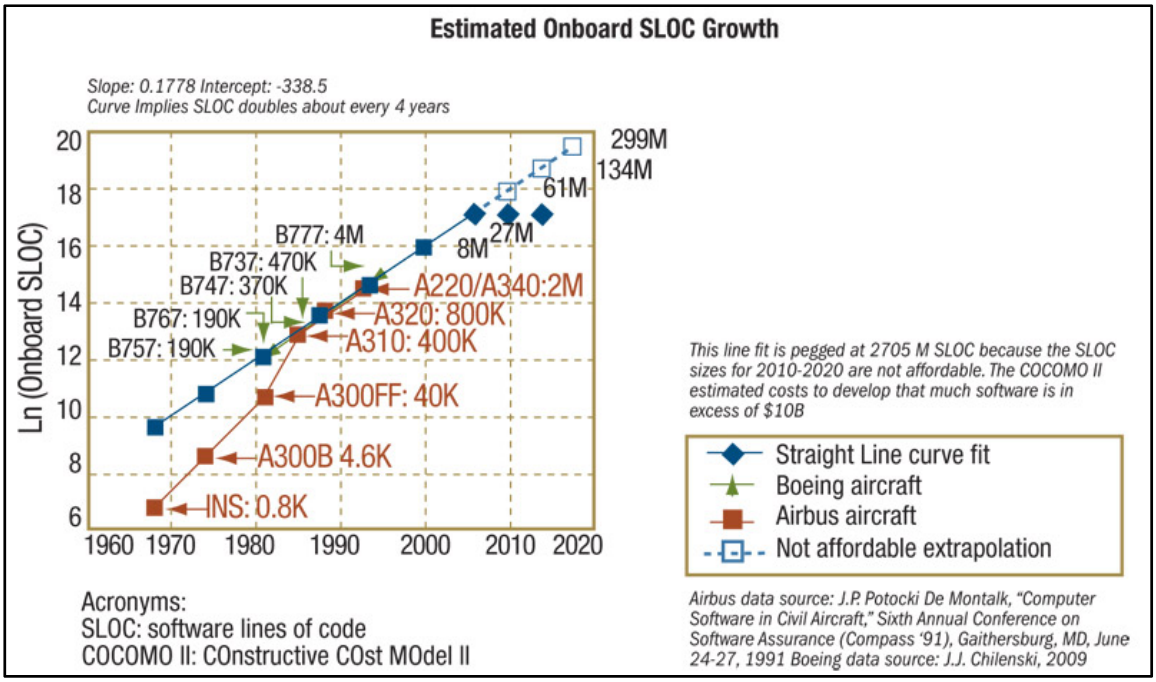


Figure 0.1 Évolution du nombre de lignes de code dans les aéronefs
Tirée de St. Clair (2009)

De ce fait, il serait intéressant d'utiliser un intergiciel de communication afin de simplifier grandement la gestion des communications entre les systèmes dans un aéronef et ajouter ainsi une couche d'abstraction entre les fonctions et les communications qu'elles utilisent. Cependant, les exigences particulières du domaine avionique imposent l'utilisation d'un intergiciel de communication qui aura des caractéristiques temps réel tel que la sûreté, le déterminisme et la fiabilité.

Cette recherche est incluse dans le cadre du projet de recherche AVIO509 qui vise à faire l'exploration architecturale des systèmes avioniques hautement intégrés. Elle vise plus précisément à évaluer la viabilité de l'utilisation d'un intergiciel de communication DDS (*Data Distribution Service*) dans le cadre du domaine avionique. Afin de pouvoir évaluer la viabilité, cette recherche vise à atteindre les trois objectifs suivants:

1. Le premier objectif consiste à analyser les différentes fonctionnalités qui sont offertes par la norme DDS. Pour ce faire, les différentes qualités de service (QoS) offertes devront être évaluées en tenant compte de leur utilité dans le domaine avionique;

2. Le second objectif consiste à caractériser les performances des communications d'un intergiciel DDS. Pour ce faire, la latence et l'utilisation de la bande passante que peut offrir un intergiciel DDS doivent être mesurées et comparées aux performances sans l'utilisation d'un intergiciel DDS. De plus, les performances obtenues devront être comparées aux performances désirées dans le domaine avionique;

3. Le troisième objectif est d'évaluer l'impact de l'utilisation d'un intergiciel DDS sur le fonctionnement d'une application avionique. Pour ce faire, une étude de cas avec une application avionique qui utilise un intergiciel DDS doit être effectué. De plus, l'étude de cas doit être effectuée sur une plateforme embarquée qui a une architecture semblable à ce que l'on peut retrouver dans les aéronefs, donc un système d'exploitation temps réel doit être utilisé.

Ce travail de recherche apporte plusieurs contributions.

En premier lieu, la norme DDS a été méticuleusement étudiée afin de bien comprendre son fonctionnement, ses bénéfices et ses limitations.

En second lieu, une caractérisation des performances de communications de l'intergiciel DDS Connext Micro est effectuée. Différents scénarios expérimentaux sont effectués afin de mesurer la latence, le taux de paquet perdu ainsi que la bande passante utilisée par l'intergiciel DDS. C'est résultats sont comparés aux résultats sans l'utilisation d'un intergiciel DDS ainsi qu'aux performances désirées dans le domaine avionique.

En troisième lieu, une étude de cas avec un AFCS (*automatic flight control system*) pour un Boeing 747-400 a été élaboré. Pour ce faire, un AFCS communique avec le simulateur de vol

X-Plane à l'aide d'un intergiciel DDS. L'AFCS ainsi que l'intergiciel DDS ont été exécutés sur la plateforme de développement FreeScale MPC8572 avec le système d'exploitation VxWorks 6.9 pour ainsi se rapprocher d'une architecture embarquée qui pourrait être utilisée en avionique. Cette étude de cas à permis d'évaluer l'impact de l'utilisation de l'intergiciel DDS sur les performances de l'application d'AFCS. Une implémentation sur le système d'exploitation avionique VxWorks 653 a été envisagée, mais le BSP (*board support package*) n'était pas disponible pour la plateforme FreeScale MPC8572 empêchant ainsi le déploiement facile de ce système d'exploitation.

En quatrième lieu, une méthodologie de portage de modèles Simulink vers une plateforme FreeScale a été élaborée et utilisée afin d'effectuer les simulations HIL (*Hardware in the loop*) de notre étude de cas avec un AFCS.

En dernier lieu, une étude de cas en utilisant les QoS offertes par l'intergiciel DDS pour faire de la redondance de système est effectuée. L'étude de cas mesure l'impact de la panne d'un système AFCS redondant utilisant un intergiciel DDS sur la stabilité de l'aéronef.

Le corps de ce mémoire est structuré comme suit. Le chapitre 1 présente une revue de la littérature sur les systèmes avioniques et sur la conception des AFCS. Le chapitre 2 présente la norme DDS dont son fonctionnement ainsi que les différentes fonctionnalités qu'elle supporte. Le chapitre 3 présente une évaluation des performances de latence et d'utilisation de bande passante d'un intergiciel DDS. Le chapitre 4 présente une étude de cas avec une application d'AFCS pour Boeing 747-400. Le chapitre 5 présente une étude de cas visant à effectuer de la redondance de système à l'aide d'un intergiciel DDS. Finalement, une conclusion des contributions de ce mémoire ainsi que des propositions de travaux futurs qui peuvent découler de ce travail de recherche sont ensuite présentées.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

Le domaine avionique est particulier à cause des exigences élevées en matière d'optimisation de l'utilisation de l'espace, du poids et de la puissance (SWaP) (Wilson et Preyssler, 2009) sur les différents systèmes avioniques. En effet, les systèmes avioniques sont contraints à des normes plus strictes et contraignantes que dans d'autres domaines tel que le domaine automobile. Ce chapitre traite donc des différentes exigences du domaine avionique. Ce chapitre aborde également ce qu'est un AFCS et un contrôleur sera choisi puisqu'une conception d'AFCS sera effectuée dans l'étude de cas du chapitre 4.

Tout d'abord, les caractéristiques et architectures des systèmes avionique sont abordées. Ensuite, le consortium FACE (*Future Airborne Capability Environment*) est présenté. Par la suite, différents travaux portants sur l'utilisation d'un intergiciel DDS sont abordés. Finalement, les particularités de la conception d'un AFCS et le contrôleur PID sont présentés.

1.1 Systèmes avioniques

Étant donné que ce projet de recherche porte sur la viabilité de l'utilisation d'un intergiciel DDS pour les applications avionique, une description de différentes notions fondamentales des systèmes avioniques sera effectuée. Les caractéristiques des systèmes avionique ainsi que les caractéristiques des différentes architectures avioniques seront donc abordées.

1.1.1 Caractéristiques des systèmes avioniques

Les systèmes avioniques contiennent de multiples fonctions qui ont des niveaux de criticités différentes allant d'une criticité faible (ex: divertissement des passagers) jusqu'à une criticité élevée (ex: système de gestion de vol). De ce fait, les systèmes avioniques ont des requis varié en fonction de leur niveau de criticité. Les systèmes avionique dont le niveau de

criticité est élevé doivent être robustes et réagir en temps réel. Les principales caractéristiques que l'on veut retrouver dans ces systèmes sont le déterminisme, la fiabilité, la sureté et la tolérance aux fautes. Afin de bien familiariser le lecteur avec ces métriques, une courte définition de ces dernières sera faite.

Déterminisme

Le déterminisme est une caractéristique d'un système qui garantit des performances prévisibles et reproductibles autant en conditions normales qu'en conditions de failles (Jakovljevic, 2009). Le déterminisme d'une communication peut être défini par une latence maximale bornée et connue.

Fiabilité

La fiabilité représente la probabilité qu'un objet remplisse sa fonction sur un intervalle de temps donné (Wang et al., 2014). La fiabilité est généralement mesurée à l'aide d'un taux de défaillance. Pour les applications de haute criticité, un taux de défaillance inférieur à 10^{-9} défaillances par heure est demandé (Obermaisser et Peti, 2006).

Sureté

La sureté est un état dans lequel il n'y a aucun problème d'équipement, perte de propriété, dégâts environnementaux et aucune perte de vie humaine (Wang et al., 2014).

Tolérance aux fautes

La tolérance aux fautes représente la capacité d'un système à contenir les fautes au sein d'une région de celui-ci (Obermaisser et Peti, 2006). De cette façon, les fautes qui surviennent dans une fonction/système avionique ne se propagent pas aux autres fonctions/systèmes et n'affectent pas le fonctionnement des autres fonctions/systèmes.

Les points individuels de défaillance sont des parties d'un système qui, en cas de failles, causent un arrêt de fonctionnement du système en entier. De ce fait, les points individuels de défaillance doivent être évités puisqu'une seule erreur à un de ces points pourrait causer

l'instabilité des systèmes avioniques. De plus, l'instabilité d'un système de criticité élevé peut engendrer des problèmes menant à la perte de vies.

Afin d'éviter les points individuels de défaillance et d'avoir ainsi une meilleure tolérance aux fautes, les différents systèmes avioniques sont souvent munis de redondance. La redondance consiste à dupliquer les éléments critiques d'un système dans le but d'augmenter la sureté, la fiabilité et la disponibilité en cas de failles ou de dégradation de ce dernier (Gohil, Basavalingarajaiah et Ramachandran, 2011).

1.1.2 Architectures des systèmes avionique

Il existe deux types d'architectures pour les systèmes avionique: l'architecture fédérée et l'architecture IMA (*Integrated Modular Avionics*). Cette section présente une description de ces deux architectures ainsi qu'une introduction à l'architecture IMA de deuxième génération.

Architecture fédérée

L'architecture fédérée est une architecture traditionnelle qui est caractérisée par le fait que chacune des différentes fonctions d'un système possède ses propres ressources dédiées (processeurs, bus de communication, entrées/sorties et mémoires). Les systèmes avioniques ayant leurs propres ressources dédiées sont communément appelés LRU (*Line Replaceable Unit*). Ces ressources dédiées permettent ainsi un découplage entre les systèmes. Ce découplage protège ces derniers contre la propagation des erreurs entre les systèmes. La certification de systèmes avec cette architecture est simple étant donné le découplage évident entre les différentes fonctions. La Figure 1.1 illustre un exemple d'une architecture avionique fédérée composée de deux LRU possédant chacun leurs propres ressources.

Un inconvénient important de cette architecture est que l'ajout de nouvelles fonctions au sein du système existant nécessite également l'ajout de nouvelles ressources matérielles tel que des boîtiers et câblages supplémentaires. Cet ajout de matériel est indésirable puisqu'il ajoute du poids et des coûts additionnels à l'aéronef. De plus, les ressources matérielles sous-

utilisées des LRU ne peuvent pas être utilisées pour d'autres fonctions à cause du découplage matériel entre celles-ci. De ce fait, les ressources ne sont pas nécessairement utilisées efficacement.

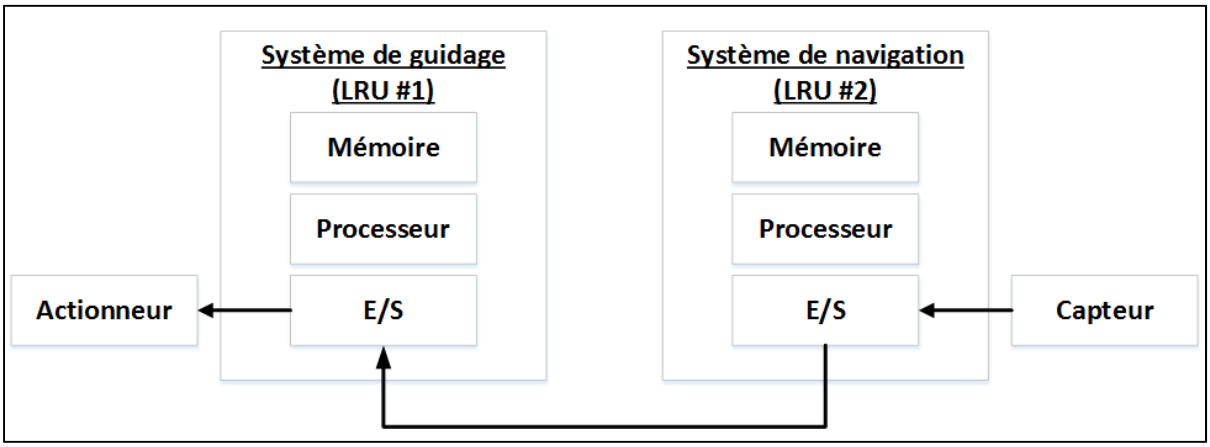


Figure 1.1 Architecture avionique fédérée

Architecture IMA

L'architecture IMA est caractérisée par le fait que cette dernière permet le partage des ressources matérielles telles que les unités de calculs, les bus de communications, les entrées/sorties et la mémoire entre les différentes fonctions logicielles d'un système avionique. La Figure 1.2 représente un exemple simple d'un système IMA qui aurait les mêmes fonctions que dans l'exemple d'architecture fédérée de la Figure 1.1 soit un système de navigation et un système de guidage. Dans cette figure, ce sont les mêmes ressources matérielles qui sont utilisées pour la fonction de guidage et pour la fonction de navigation. Il est toutefois important de savoir que l'exemple de la Figure 1.2 ne représente que la portion matérielle d'un système IMA. L'architecture IMA est également caractérisée par le fait que plusieurs fonctions logicielles de criticités différentes peuvent s'exécuter sur la même plateforme matérielle.

Un avantage de cette architecture est qu'elle permet de réduire la quantité de composants matériels nécessaires pour effectuer les mêmes fonctions que dans une architecture fédérée. De plus, cette architecture répond bien aux besoins actuels des concepteurs d'aéronefs qui

sont de réduire l'espace, le poids ainsi que la puissance consommée (SWaP) des aéronefs (Watkins et Walter, 2007). Dans l'architecture fédérée, il peut arriver qu'une ressource matérielle telle que le processeur ou la mémoire soit sous-utilisée étant donné que la fonction logicielle du système fédérée ne nécessite pas toutes les ressources disponibles. L'avantage de l'architecture IMA est que cette dernière permet à de multiples fonctions logicielles d'utiliser les ressources matérielles sous-utilisées. L'architecture IMA est donc plus flexible et permet une utilisation plus efficace des ressources matérielles. Un autre avantage de l'architecture IMA est que les concepteurs de fonctions logicielles n'ont pas à concevoir de plateforme matérielle ce qui réduit leurs efforts de développements.

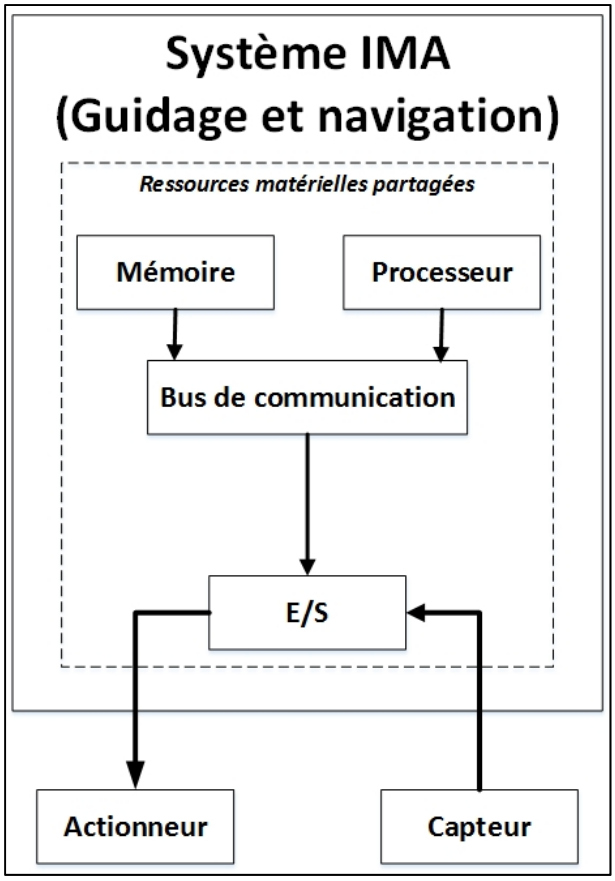


Figure 1.2 Architecture IMA

Cette architecture n'est cependant pas sans défauts. Étant donné que cette d'architecture partage les ressources du système entre plusieurs applications, elle créé ainsi un couplage

indésirable entre les fonctions qui n'était pas présent dans l'architecture fédérée. De ce fait, des mécanismes doivent être instaurés afin de contrer ce couplage entre les différentes fonctions et ainsi empêcher la propagation des fautes. Ces mécanismes sont généralement instaurés au sein d'un système d'exploitation temps réel (Littlefield-Lawwill et Kinnan, 2008) qui se charge de faire un partitionnement robuste de l'espace mémoire (partitionnement spatial) ainsi que du temps d'accès au processeur (partitionnement temporel). Ces mécanismes de partitionnement sont particulièrement importants lorsque des fonctions de criticité différentes veulent utiliser les mêmes ressources matérielles. Il est donc important qu'une fonction de basse criticité ne puisse pas empêcher une fonction de haute criticité d'utiliser les ressources dont elle a besoin.

Le premier avion de ligne qui a eu des systèmes IMA est le Boeing 777 ce qui a mené d'autres aéronefs tels que le Lockheed C130 AMP, le Airbus A380 et le Boeing 787 à contenir également des systèmes IMA (Prisaznuk, 2008). La plateforme Genesis (Randy et Chris, 2006) qui a été développée par Smiths Aerospace est un bon exemple de plateforme IMA. Le CCS (*Common Core System*) dans le Boeing 787 Dreamliner est une implémentation de la plateforme Genesis qui regroupe différents GPM (*General Processing Modules*) ainsi que des interrupteurs réseau (définis dans la norme ARINC 664-P7) dans un cabinet (Randy et Chris, 2006). De plus, le CCS peut contenir plus de 100 fonctions (Watkins et Walter, 2007). Le AIMS (*Aircraft Information Management System*) qui se retrouve dans le Boeing 777 est également un exemple de système IMA (Alena et al., 2007).

Architecture IMA-2G

Cette d'architecture représente l'évolution future de l'architecture IMA actuelle (d'où le nom IMA-2G ou IMA 2e génération). Cette dernière vise à contrer deux défis des plateformes IMA actuelles soit d'avoir de meilleures capacités de reconfiguration ainsi que l'intégration de matériel de traitement de données commercial prêt à l'emploi (COTS) tel que les processeurs multicoeurs (Bieber, 2012).

L'IMA actuel repose sur les normes ARINC 653 et ARINC 664 qui imposent une allocation statique des ressources matérielles, telles que le processeur et la mémoire, pour les différentes fonctions. Cependant, la problématique de l'allocation statique c'est qu'elle ne permet pas la migration de fonctions en cas de failles. De ce fait, le projet *SCAlable & ReconfigurabLe Electronics plaTforms and Tools* (SCARLETT) qui est élaboré par le laboratoire de recherche Onera en collaboration avec Thales et Airbus vise justement a explorer la reconfiguration dynamique d'architectures IMA (Bieber, 2012). Dans l'article (Lopez-Jaquero et al., 2012), les auteurs proposent une couche intermédiaire de redondance et de reconfiguration qui se trouve entre la couche du système d'exploitation temps réel (RTOS) et la couche d'interface d'application/d'exécution (APEX) afin de faire de la reconfiguration dynamique. La Figure 1.3 illustre une couche de redondance et reconfiguration définie dans (Lopez-Jaquero et al., 2012) qui ajoute ainsi une interface supplémentaire entre la couche APEX et le RTOS.

Les processeurs multicoeurs sont de plus en plus performants et présents dans de multiples domaines d'applications. La complexité grandissante des systèmes avionique ainsi que la quantité grandissante de fonctions logicielles à intégrer (Bieber, 2012) va nécessiter l'utilisation de processeurs multicoeurs dans les systèmes avionique. Cependant, ces processeurs sont beaucoup plus complexes à certifier dans un contexte avionique que les processeurs mono-cœur (Fuchsen, 2010). Ceci est dû principalement à cause de l'analyse plus compliquée du pire cas de temps d'exécution (WCET) ainsi que l'analyse complexe des accès à la mémoire cache partagée entre les cœurs du processeur (Bieber, 2012). Le projet SCARLETT vise à implémenter des fonctions de sureté critique sur des processeurs multicoeurs tout en assurant le déterminisme de ces dernières (Fuchsen, 2009). Ceci permettrait ainsi de faciliter la certification des processeurs multicoeurs.

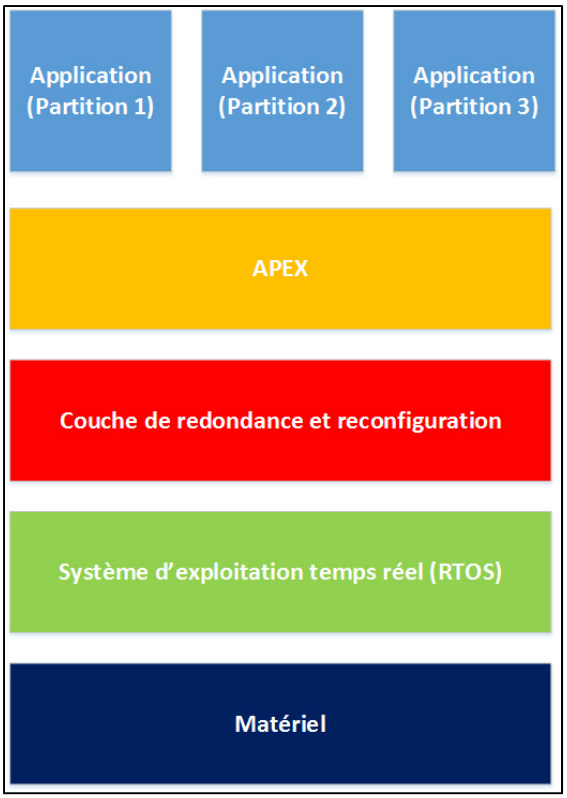


Figure 1.3 Couche de redondance et reconfiguration
Adaptée de Lopez-Jaquero et al. (2012)

Pour ce projet de recherche, l'architecture fédérée est utilisée étant donné que les RTOS qui ont les mécanismes de partitionnement IMA sont dispendieux. De plus, ces RTOS ne fonctionnent qu'avec certaines plateformes de développement qui sont également dispendieuses. Le concept de partitionnement IMA n'est pas indispensable à l'évaluation de l'utilisation d'un intergiciel DDS dans le domaine avionique et il est plus simple d'utiliser une architecture fédérée.

1.2 Consortium FACE

Le consortium FACE (*Future Airborne Capability Environment*) a été formé en 2010 et est un partenariat entre le gouvernement américain et l'industrie de l'aviation afin de définir un environnement avionique pour toutes les plateformes avioniques militaires (The Open Group, 2014).

Le consortium FACE a comme but de définir des architectures logicielles et des interfaces qui permettent le découplage entre les différents segments d'une plateforme avionique. De ce fait, la norme technique FACE (FACE Consortium, 2013) propose une architecture de plateforme qui divise cette dernière en cinq segments:

- le segment système d'exploitation;
- le segment des services d'entrées/sorties;
- le segment des services spécifiques à la plateforme;
- le segment des services de transport;
- le segment des composants portables.

La Figure 1.4 représente l'architecture de plateforme proposée par FACE et les interactions entre les différents segments.

Cette figure présente une interface standardisée entre chaque segment ce qui permet un découplage entre ces derniers. De plus, cette figure propose l'utilisation d'un intergiciel de communication dans le segment des services de transport afin d'effectuer un découplage entre le segment des composants portables et le segment des services spécifiques à la plateforme. Un intergiciel de communication est une entité logicielle qui sert à faire une abstraction des différentes couches de communication. Un intergiciel de communication permet ainsi un découplage entre les applications et le système d'exploitation. De plus, l'intergiciel de communication est normalement simple d'utilisation et permet de configurer facilement les communications. L'intergiciel de communication offre ainsi une grande flexibilité. Le désavantage de ce dernier est que son utilisation est cependant moins efficace que d'utiliser directement la communication au niveau réseau (Garcia-Valls et al., 2010).

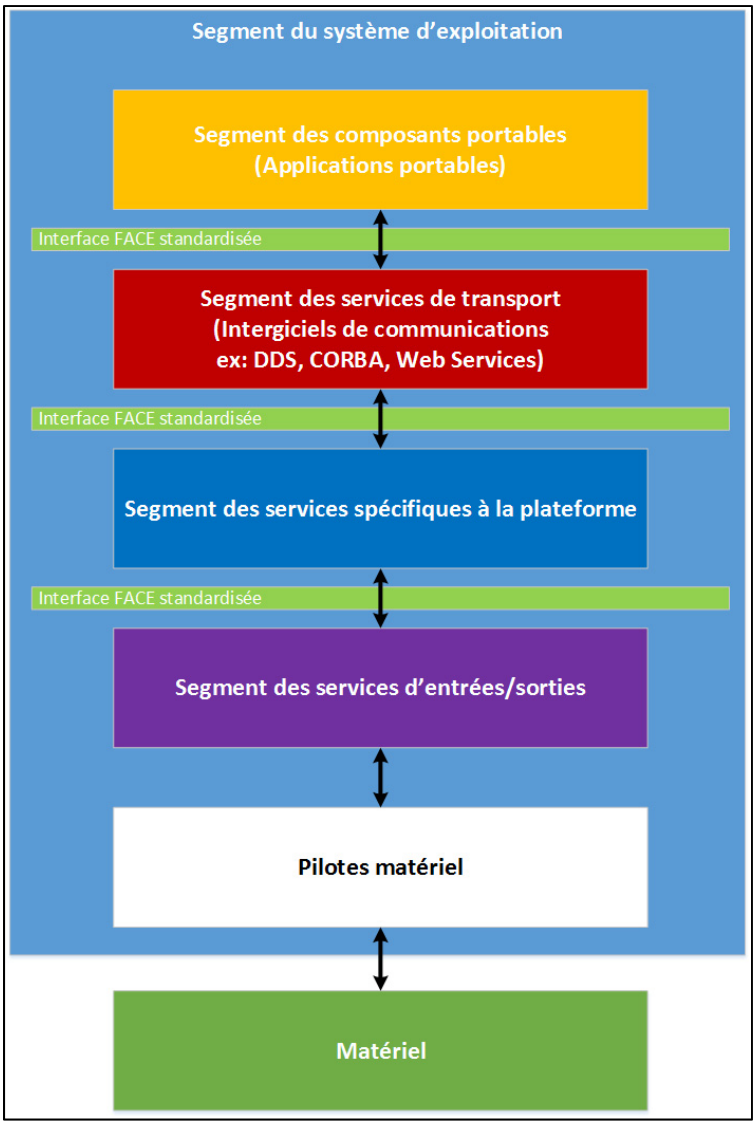


Figure 1.4 Architecture de la plateforme FACE
Adaptée de FACE Consortium (2013)

Étant donné que ce mémoire porte sur l'utilisation d'un intergiciel DDS sur une plateforme avionique, seulement le segment des services de transport est expliqué dans cette section, car les autres segments sortent du cadre des objectifs de ce mémoire.

Selon la norme technique FACE, le but du segment de service de transport est de fournir des interfaces transparentes et standardisées qui facilitent l'intégration d'applications portables au sein de différentes plateformes avioniques qui ont différentes architectures (FACE

Consortium, 2013). Le segment de services de transports peut supporter plusieurs paradigmes de communication tels que le paradigme de publication/souscription et le paradigme de requête/réponse par exemple. De plus, le segment de services de transports doit être configurable et pouvoir distribuer les données. Plusieurs services supplémentaires peuvent se retrouver dans ce segment tels que les QoS, l'association de messages, la translation de paradigme et la conversion de données par exemple. Finalement, ce segment doit au minimum supporter les messages de types suivants:

- longueur fixe;
- longueur variée;
- périodique;
- apériodique;
- émission;
- diffusion multipoint;
- diffusion à un point.

Les normes DDS, CORBA ainsi que les services web (*Web Services*) sont mentionnés comme étant des exemples de services de transports qui pourraient être utilisés dans une plateforme FACE. De ce fait, il est mis en évidence que l'utilisation d'un intergiciel de communication est au cœur des préoccupations actuelles des concepteurs d'aéronefs et que la norme DDS définit justement un intergiciel qui satisfait les besoins du segment des services de transports qui se retrouve dans la norme technique FACE.

1.3 Domaines d'applications des intergiciels DDS

La norme DDS a été créée dans le but de répondre aux besoins en communications des systèmes temps réel. Cette section porte sur l'utilisation des intergiciels DDS dans différents domaines d'utilisation possédant des systèmes temps réel.

1.3.1 Diverses utilisations d'intergiciels DDS

Un intergiciel DDS est utilisé pour faire de la commande distribuée industrielle dans l'article de (Jinsong et al., 2012). Dans cet article, les auteurs comparent l'utilisation d'un intergiciel DDS avec la norme IEC 61499 qui est utilisée pour la commande distribuée et l'automatisation nouvelle génération. De ce fait, les auteurs ont donc fait une mise en correspondance entre le IEC 61499 et le DDS et ont fait plusieurs tests de performance principalement au niveau de la latence. Bien que la communication DDS a obtenu des résultats plus lents qu'avec les *sockets,* les auteurs croient tout de même que l'utilisation d'un intergiciel DDS peut être bénéfique puisqu'elle permet de réduire la complexité des communications réseau puisque les systèmes de commande industriels peuvent avoir beaucoup de contrôleurs qui sont connectés sur le même réseau Ethernet.

Un intergiciel DDS est utilisé pour faire de la vidéo en transit avec l'encodage H.264 SVC (*Scalable Video Codage*) par DDS dans l'article de (Detti et al., 2010). Un des défis de la vidéo en transit dans cet article est de pouvoir adapter dynamiquement le flux de données du vidéo en fonction de la bande passante variante du canal de communication. De ce fait, les auteurs ont utilisé l'encodage H.264 SVC puisqu'il permet justement de modifier le *bit-rate* vidéo en n'envoyant qu'une partie des paquets vidéo. De plus, ils ont utilisé la fonctionnalité de filtrage des données qui est incluse dans les sujets DDS pour ainsi contrôler la quantité de données vidéo qui est transférée par l'intergiciel DDS.

Un domaine dans lequel les intergiciels DDS sont souvent utilisés est dans les CPS (*cyber-physical systems).* Les CPS sont des systèmes distribués qui sont généralement munis d'une multitude de capteurs et actionneurs qui leur permettent d'interagir avec le monde physique (Woochul, Kapitanova et Sang Hyuk, 2012). De ce fait, les réseaux de véhicules, la robotique distribuée (Cruz et al., 2012), les WSN (*wireless sensor network*) (Boonma et Suzuki, 2009), la commande et l'automatisation distribuée (Calvo et al., 2010), les systèmes de gestion de combat, les salles de marché de bourse (*trading-floors*) et le contrôle du trafic aérien (Detti et al., 2010) sont des domaines dans lesquels des intergiciels DDS sont utilisés.

1.3.2 Utilisation d'un intergiciel DDS en avionique

Il y a tout de même quelques références d'utilisation d'intergiciels DDS dans le domaine avionique tel que dans l'article de (Howard, 2013) dans lequel il est mention de l'utilisation d'un intergiciel DDS dans un avion de transport militaire KC-390. Dans cet article, il est mentionné que la compagnie Embraer a utilisé un intergiciel DDS afin d'intégrer des applications logicielles sur différents processeurs et systèmes d'exploitation. Il est également mentionné que l'intergiciel est utilisé afin de faire des communications intranoeuds et internoeuds. Cependant, cet article n'est pas technique et ne discute pas des enjeux et de la méthode d'implémentation d'un tel intergiciel au sein d'un aéronef.

Il y a également l'article de (Tambe, Garcia Aranda et Schlesselman, 2013) dans lequel un intergiciel DDS est utilisé pour effectuer la communication entre un simulateur de vol X-Plane ainsi qu'une carte de développement BeagleBone dans le but de faire une estimation de la santé des capteurs d'un aéronef MQ-9 Reaper UAV. La Figure 1.5 illustre que les auteurs ont utilisé une communication DDS entre un simulateur de vol et une plateforme Beaglebone afin de transmettre les données du moteur de l'aéronef. La santé des capteurs est ensuite estimée à l'aide d'un réseau bayésien sur la plateforme Beaglebone et les résultats sont ensuite envoyés à l'affichage.

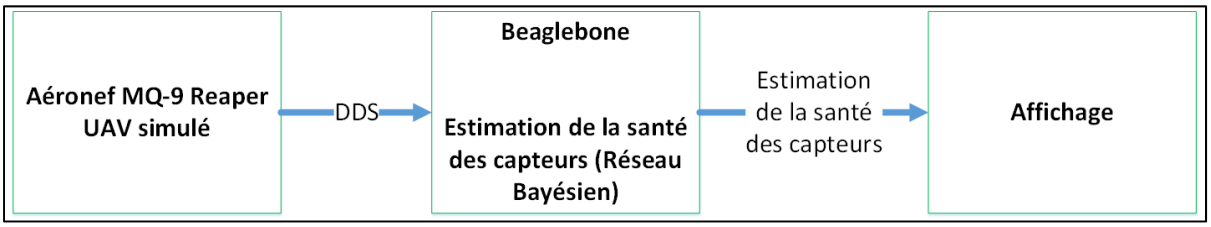


Figure 1.5 Estimation de la santé des capteurs
Adaptée de Tambe, Garcia Aranda et Schlesselman (2013)

De plus, dans l'article de (Jun et al., 2009), les auteurs font la conception d'un simulateur de vol distribué pour un Boeing 737-800 qui utilise un intergiciel de communication DDS tel qu'illustré à la Figure 1.6. L'utilisation de l'intergiciel DDS a permis aux auteurs de concevoir

ce simulateur de vol distribué rapidement et permet également de simplifier l'ajout de nouveaux modules au simulateur.

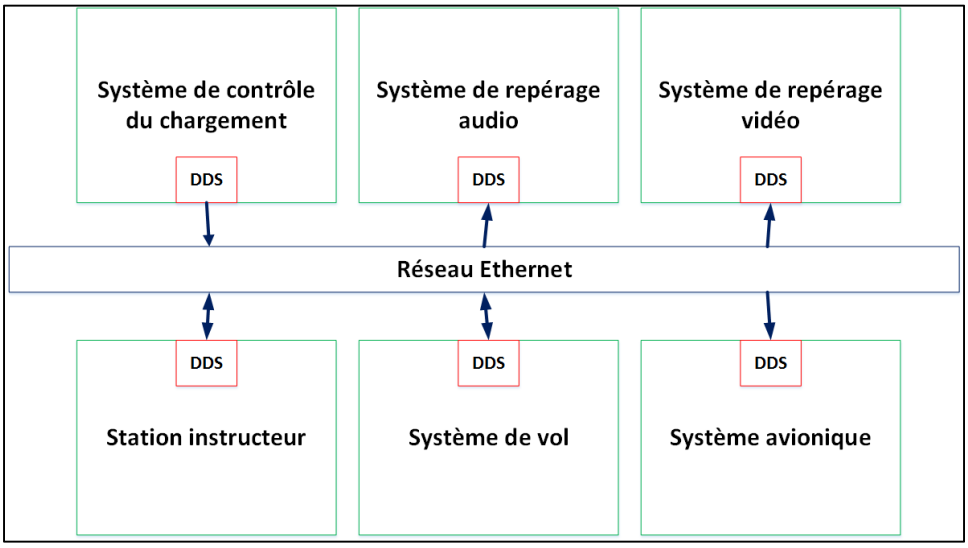


Figure 1.6 Simulateur de vol distribué
Adaptée de Jun et al. (2009)

Les trois articles mentionnés dans cette section sont les seuls qui font mention d'une utilisation d'un intergiciel DDS dans le domaine avionique. L'intergiciel DDS est donc peu utilisé dans le domaine avionique d'où l'intérêt de ce projet de notre projet de recherche. Ce dernier se distingue de ceux présentés dans cette section par le fait que ce n'est pas seulement une utilisation d'un intergiciel DDS, mais également une évaluation de l'impact de l'utilisation de celui-ci sur les performances d'une application avionique. De plus, ce projet de recherche met en évidence les différentes fonctionnalités offertes par la norme DDS qui sont pertinentes au domaine avionique ce qui n'est pas abordé en profondeur dans les articles présentés.

1.3.3 Évaluation des performances d'intergiciels de communication

Dans l'article de (Bellavista et al., 2013), une comparaison des performances de deux intergiciels DDS est faite, et ce, avec différentes grandeurs de paquet et nombre d'abonnés. Les performances telles que la latence, le débit de données, le nombre d'échantillons par

seconde ainsi que l'utilisation du processeur et de la mémoire vive sont évalués dans cet article. Cependant, cet article ne traite que des performances moyennes des intergiciels DDS, or dans le domaine avionique les performances sont mesurées à partir des performances du pire cas.

Dans l'article de (Schmidt, Deshpande et O'Ryan, 2002), les auteurs font une comparaison de la latence moyenne et de la gigue obtenue en utilisant un intergiciel de communication RT-CORBA, et ce, avec différents systèmes d'exploitation. Les résultats de cet article montrent que la latence moyenne et la gigue obtenue avec un système d'exploitation temps réel tel que QNX et VxWorks sont meilleures que ce qui est obtenu avec un système d'exploitation Windows tel qu'illustré au Tableau 1.1. Dans le chapitre 3 de ce projet de recherche, une caractérisation de la latence et de la gigue d'un intergiciel DDS sera effectuée. Cependant, le système d'exploitation Windows 7 sera utilisé puisqu'on a pas de plateforme avionique dans notre laboratoire. De ce fait, les résultats de l'article de (Schmidt, Deshpande et O'Ryan, 2002) serviront de référence afin de faire une analogie entre les résultats obtenus dans le chapitre 3 avec le système d'exploitation Windows 7 et les résultats qui seraient obtenus dans un environnement avionique.

Tableau 1.1 Comparaison des performances des communications de l'intergiciel RT-CORBA dans l'article de Schmidt, Deshpande et O'Ryan (2002)

Système d'exploitation	**Latence moyenne (us)**	**Gigue (us)**
Windows NT	650	500
VxWorks	180	25
QNX	280	175

1.4 Système automatique de contrôle de vol (AFCS)

Cette section présente ce qu'est un AFCS ainsi que le contrôleur qui sera utilisé pour concevoir l'AFCS utilisé dans l'étude de cas du chapitre 4.

1.4.1 Architecture d'un autopilote

Un autopilote est un système qui permet de contrôler un aéronef et de lui faire suivre une trajectoire particulière et naviguer entre plusieurs points de repère. Pour ce faire, l'autopilote est généralement divisé en trois systèmes: le système de guidage, le système de contrôle et le système de navigation tel qu'illustré à la Figure 1.7. Ces différents systèmes ont chacun leur contribution au fonctionnement de l'autopilote.

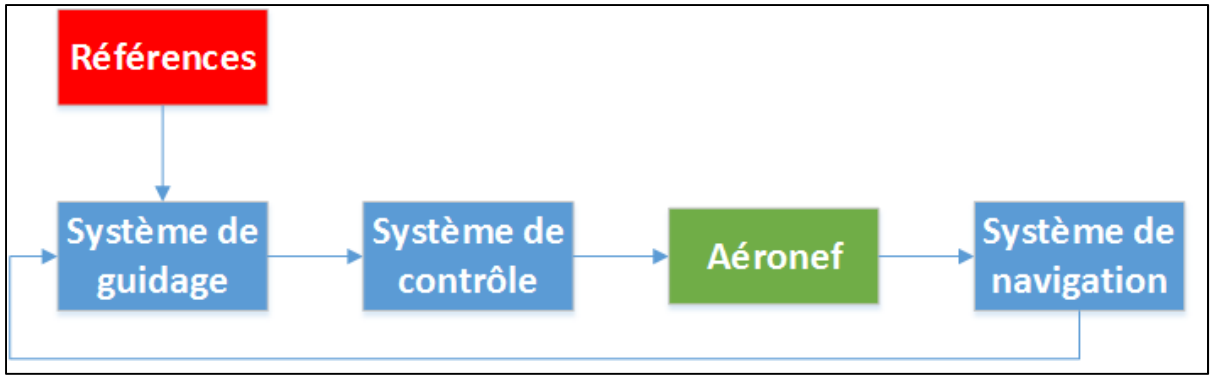


Figure 1.7 Schéma bloc d'un autopilote
Adaptée de Johansen (2012)

Système de guidage

Le système de guidage sert à envoyer les commandes au système de contrôle en fonction de l'emplacement actuel de l'aéronef ainsi que du prochain point de repère que l'aéronef doit atteindre.

Système de contrôle

Le système de contrôle sert à contrôler les différents axes de rotation de l'aéronef (tangage, roulis & lacet), la vitesse de l'aéronef ainsi que son altitude. Pour ce faire, ce système doit être muni de plusieurs boucles de commande de vol qui permettent ainsi de contrôler indirectement les différents axes de rotation par l'entremise des différentes surfaces de contrôle de l'aéronef. Le système de contrôle est communément appelé AFCS.

Système de navigation

Le système de navigation sert à déterminer la position actuelle de l'aéronef et d'envoyer cette dernière au système de guidage.

Étant donné que l'objectif de la conception de l'autopilote n'est que de valider les performances de communications avec un intergiciel DDS, seulement le système de contrôle (AFCS) a été conçu. Le système de navigation est déjà inclus dans le simulateur de vol qui sera utilisé et le système de guidage sera remplacé par une commande fixe de vitesse et d'altitude qui sera fixée par l'utilisateur.

1.4.2 Contrôle d'un aéronef

La méthode qui est utilisée pour contrôler les axes de rotation ainsi que la vitesse d'un aéronef est présentée dans cette sous-section. Les axes de rotation d'un aéronef sont illustrés à la Figure 1.8. Cette figure illustre les trois axes de rotation soit le roulis, le tangage et le lacet. Les axes de rotation des aéronefs sont contrôlés par l'entremise des différentes surfaces de contrôle soit les ailerons, les élévateurs et le gouvernail. Les élévateurs permettent de contrôler le tangage de l'aéronef, donc la rotation de l'aéronef autour de l'axe des y. Les ailerons permettent de contrôler le roulis de l'aéronef donc sa rotation autour de l'axe des x. Finalement, le gouvernail permet de contrôler le lacet, donc la rotation de l'aéronef autour de l'axe des z. Pour contrôler la vitesse de l'aéronef, il suffit de contrôler l'accélérateur de ce dernier.

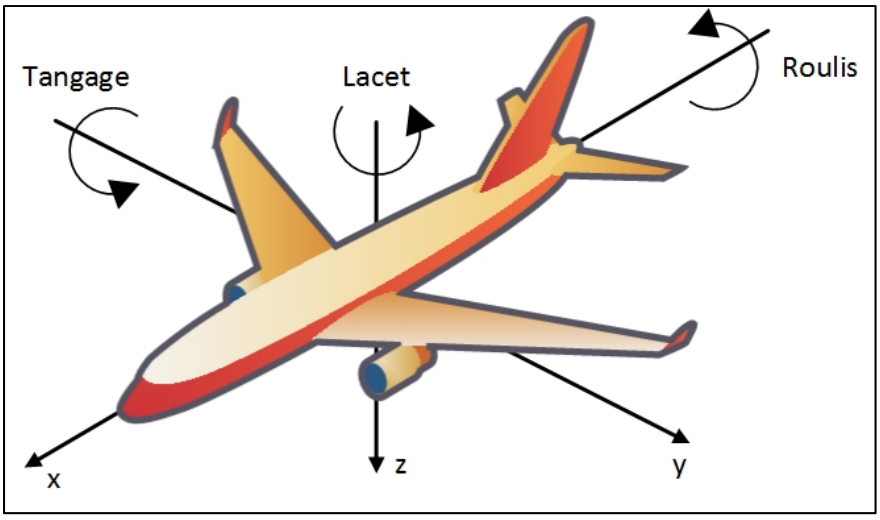


Figure 1.8 Axes de rotation d'un aéronef
Adaptée de Johansen (2012)

1.4.3 Contrôleurs pour boucles de contrôle de vol

Dans le chapitre 4, une application d'AFCS sera conçue. Pour ce faire, des boucles de commande de vol seront conçues. De ce fait, un choix de contrôleur doit être fait. Cette section porte sur les différents contrôleurs qui sont utilisés dans la littérature afin de faire des boucles de commande de vol.

Il existe plusieurs types de contrôleurs qui peuvent être utilisés pour concevoir un AFCS. La plupart des articles dans la littérature utilisent un contrôleur proportionnel intégral dérivé (PID) pour faire le bloc de contrôle de l'autopilote (Al-Jarrah et al., 2011; Barros dos Santos et de Oliveira, 2011; Johansen, 2012; Manocha et Sharma, 2009).

Dans (Johansen, 2012), l'auteur utilise le contrôleur robuste non linéaire *Sliding Mode Control* (SMC) afin de concevoir des boucles de contrôle de vol. Suite à ses expériences, il a conclu que le contrôleur PID offrait de meilleures performances.

Dans le cadre de ce projet de recherche, le contrôleur PID a été choisi à cause de sa simplicité ainsi que la grande quantité d'informations déjà disponibles sur les boucles de commande de vol avec les PID.

1.4.4 Contrôleur PID

Le contrôleur PID est un contrôleur qui est beaucoup utilisé pour le contrôle d'une multitude de procédés dans plusieurs domaines. Ce type de contrôleur consiste à tenter de réduire l'erreur entre la valeur de référence désirée et la valeur actuelle du procédé jusqu'à ce que cette dernière tende vers zéro. Le fonctionnement du contrôleur PID en mode parallèle est représenté par le schéma illustré à la Figure 1.9 .

La formule mathématique pour calculer l'erreur e(t) est la suivante:

$$e(t) = x\_reference(t) - x(t) \tag{1.1}$$

La formule de la sortie du contrôleur PID peut ensuite en être déduite:

$$y(t) = Kp \cdot e(t) + Ki \cdot \int e(t)\,dt + Kd \cdot e(t)\frac{d}{dt} \tag{1.2}$$

Il est cependant important de noter que les contrôleurs PID ne sont pas conçus pour contrôler des procédés non linéaires tels que l'aérodynamique d'un aéronef. De ce fait, les résultats obtenus avec ce type de contrôleur ne seront pas optimaux.

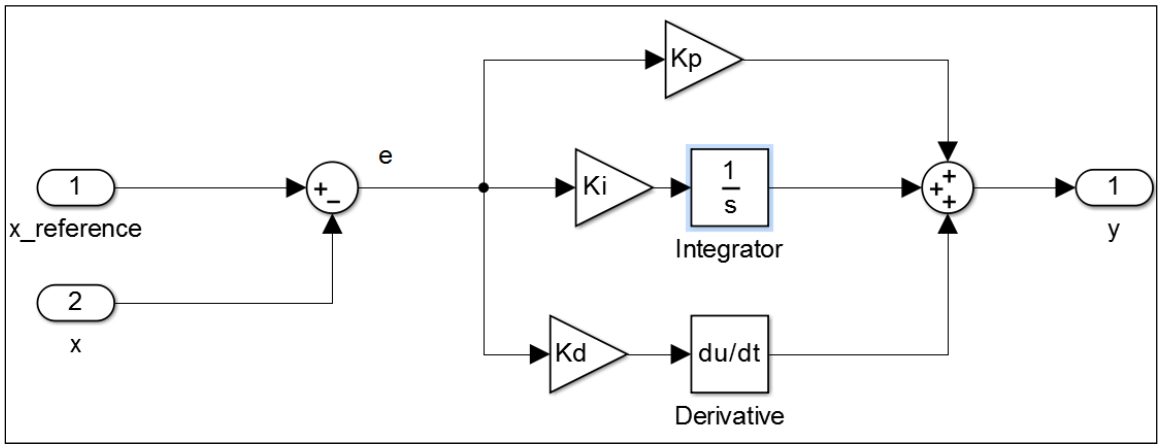


Figure 1.9 Schéma d'un contrôleur PID
Adaptée de Johansen (2012)

La calibration des contrôleurs est une étape importante de la conception de tout type de contrôleur. En effet, la calibration des différents gains des boucles de commande permet d'obtenir des performances qui rencontrent les requis de l'application voulue. Cependant, avant de traiter de la calibration les contrôleurs, une brève description des métriques importantes sera faite.

Temps de stabilisation

Le temps de stabilisation est le temps que prend la sortie d'un contrôleur pour se rendre à plus ou moins 5% de la valeur de référence. Généralement, il est désirable d'avoir le plus petit temps de stabilisation possible.

Dépassement

Le dépassement est la distance entre la valeur désirée et la valeur maximale de la réponse du contrôleur. Il est généralement désirable d'avoir un faible dépassement. Cette valeur est généralement exprimée en pourcentage.

Erreur en régime permanent

L'erreur en régime permanent est l'écart entre la réponse du contrôleur et la valeur de référence lorsque ce dernier est en régime permanent. Cet écart doit être le plus près de zéro possible afin d'avoir de bons résultats.

Stabilité

Un système est considéré comme étant stable lorsque ce dernier converge vers la valeur de référence lorsque le temps tend vers l'infini. Un système peut être considéré comme étant stable lorsque ce dernier oscille faiblement autour de la valeur de référence.

Il existe plusieurs méthodes de calibration de contrôleurs PID qu'on peut retrouver dans la littérature. Une méthode est de calculer mathématiquement les gains du contrôleur PID à partir de la fonction de transfert du procédé à contrôler. Cette méthode peut être effectuée manuellement ou à l'aide d'un logiciel, mais nécessite la fonction de transfert du procédé à

contrôler ce qui n'est pas très pratique dans le cas présent puisque les fonctions de transfert des surfaces de contrôle du Boeing 747-400 ne nous sont pas disponibles.

Il existe également plusieurs méthodes qui permettent de calibrer les gains d'un contrôleur PID lorsque la fonction de transfert du procédé n'est pas disponible. Il y a notamment la méthode de Ziegler-Nichols, la méthode Cohen-Coon et la méthode *process reaction curve* qui permettent d'obtenir les gains d'un contrôleur PID sans avoir accès à la fonction de transfert du procédé (Bennett, 2006). Bien que ces méthodes soient relativement simples, la méthode manuelle d'essai et erreur (*trial and error*) a été utilisée pour déterminer les gains des différents contrôleurs. La méthode manuelle d'essai et erreur a donc été basée sur les valeurs de gains que l'on retrouve dans (Barros dos Santos et de Oliveira, 2011) et la calibration des gains a été effectuée en changeant les gains en fonction des relations décrites dans le Tableau 1.2.

Tableau 1.2 Effets indépendants de la calibration des gains d'un contrôleur PID
Adapté de Ang, Chong et Li (2005)

Réponse en boucle fermée	**Temps de montée**	**Dépassement**	**Temps de stabilisation**	**Erreur en régime permanent**	**Stabilité**
Augmentation K_P	Diminue	Augmente	Augmente	Diminue	Dégrade
Augmentation K_I	Diminue un peu	Augmente	Augmente	Diminue beaucoup	Dégrade
Augmentation K_D	Diminue un peu	Diminue	Diminue	Peu d'effet	Améliore

1.5 Protocole UDP/IP

Le protocole UDP (*User Datagram Prototocol*) est présenté dans cette section puisque l'intergiciel qui sera utilisé dans le cadre de ce projet de recherche utilise ce protocole. Le protocole UDP/IP a été conçu afin de pouvoir envoyer des messages, appelés datagrammes, avec un minimum de mécanismes protocolaires. De plus, ce dernier fait partie de la suite de protocoles IP (*Internet Protocol*). Étant donné que le protocole UDP/IP possède un minimum de mécanismes protocolaires, ce dernier n'assure pas la fiabilité des communications, donc il

est possible qu'il y ait des pertes de paquets ou que l'ordre de réception ne soit pas le même que l'ordre d'envois. Le protocole UDP/IP fait partie de la couche de transport (couche 4) du modèle de représentation OSI (*Open System Interconnection*). La communication UDP/IP utilise des adresses IP et des ports pour effectuer la communication entre deux récipients. Il y a deux types d'adresses IP soit les adresses IPv4 (*Internet Protocol version 4*) et les adresses IPv6 (*Internet Protocol version 6*). Le format de l'entête UDP/IP est illustré à la Figure 1.10.

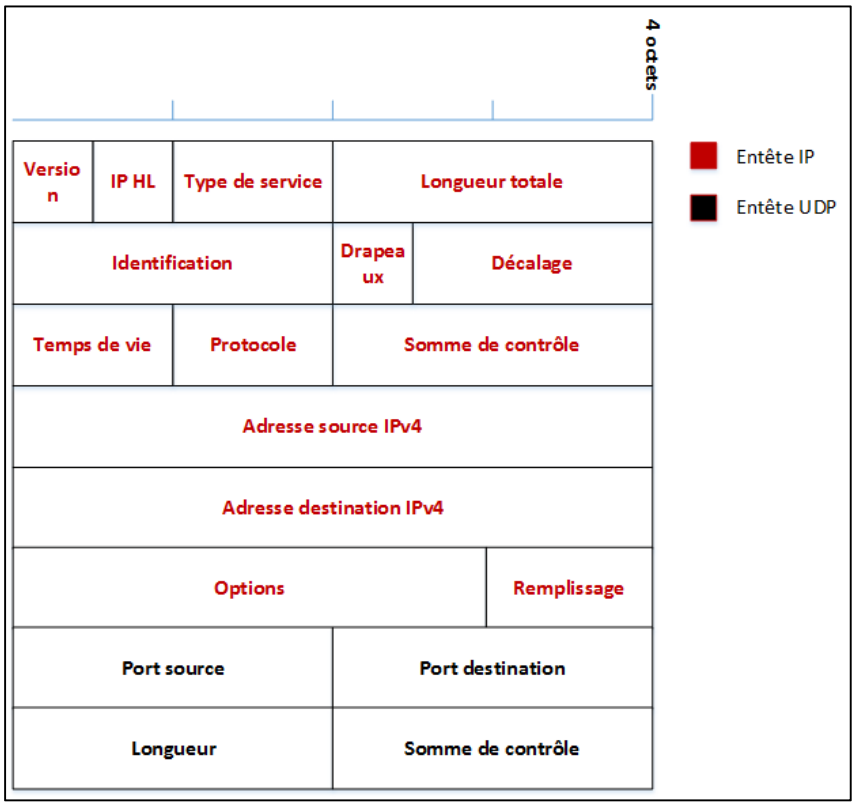


Figure 1.10 Entête d'un paquet UDPv4
Adaptée de Redakca HW Serveru (2003)

Dans l'entête IP, il y a un champ type de service qui peut être utilisé pour donner une priorité différente aux différents paquets. Cette fonctionnalité peut donc être utilisée pour donner une priorité plus élevée aux paquets d'applications de haute criticité face aux paquets d'applications de basse criticité. L'entête UDP permet d'avoir des valeurs de port allant de 0 jusqu'à 65 535. Un paquet UDPv4 (un paquet UDP utilisant le protocole IPv4) possède une

longueur maximale de 65 535 octets. De ce fait, une fragmentation doit être effectuée au préalable si l'envoi de données plus grandes que cette longueur maximale est nécessaire.

Le protocole UDP/IP est couramment utilisé dans les applications qui nécessitent un temps de réponse rapide et qui tolèrent une certaine perte de données telles que la lecture de vidéo en transit et la voix sur IP (VoIP). De plus, le protocole UDP/IP est utilisé en tant que protocole de communication pour le bus de données AFDX (*Avionics Full-Duplex Switched Ethernet*) qui est couramment utilisé dans le domaine avionique (Wang, Wang et Shi, 2012). De ce fait, il est possible d'en conclure que ce protocole peut bien être utilisé dans les applications avionique, s'il est utilisé avec du matériel et logiciel qui permettent de contrer son manque de fiabilité.

1.6 Conclusion

Le but de ce chapitre était de faire une revue de la littérature sur les systèmes avionique ainsi que sur la conception d'un AFCS. Tout d'abord, il a été vu que les systèmes avioniques ont des contraintes de performances élevées telles que le déterminisme, la fiabilité, la sureté et la tolérance aux fautes.

Il a été décidé dans ce chapitre que l'architecture fédérée sera utilisée pour ce projet de recherche. Ce choix a été fait à cause de la simplicité de cette architecture ainsi que le manque d'accès à une architecture IMA. De plus, il a été vu que la notion d'intergiciel de communication permet de créer un découplage entre les fonctions et les communications qu'elles utilisent. Le consortium FACE a été abordé et il a été vu que ce dernier recommande l'utilisation d'un intergiciel de communication DDS afin d'effectuer un découplage entre les applications et les communications.

Pour l'étude de cas avec un AFCS, il a été vu que le contrôleur PID est grandement utilisé pour faire des boucles de commande de vol. De plus, ce type de contrôleur est simple et il y a

une grande quantité d'informations disponibles sur ce type de boucles de commande. De ce fait, le contrôleur PID a été choisi pour concevoir le AFCS dans le chapitre 4.

Le prochain chapitre porte sur une analyse détaillée des fonctionnalités que la norme DDS offre qui sont pertinentes au domaine avionique afin de justifier ainsi l'utilisation d'un intergiciel DDS dans ce domaine.

CHAPITRE 2

ANALYSE DE LA NORME DDS

Afin de pouvoir utiliser adéquatement un intergiciel de communication DDS et de comprendre l'intérêt de son utilisation dans le domaine avionique, ce chapitre présente la norme DDS. De ce fait, ce chapitre porte sur la communication à l'aide d'un intergiciel tel que défini dans la norme DDS. Il présente tout d'abord les caractéristiques de la norme DDS. De plus, il présente également l'intergiciel DDS ainsi que les fonctionnalités qu'il offre qui sont pertinentes au domaine avionique. Finalement, un intergiciel commercial sera choisi afin d'évaluer la viabilité de l'utilisation d'un intergiciel DDS dans le domaine avionique. C'est donc cet intergiciel qui sera utilisé pour les tests effectués dans les prochains chapitres.

2.1 Objectifs de la norme DDS

La norme DDS a été élaborée par le *Object Management Group* (OMG) en 2003 (Object Management Group, 2007). Cette norme définie les spécifications pour une interface de programmation (API) qui se divise en deux couches soit la couche de publication/souscription basée sur les données (DCPS) et la couche de reconstruction locale des données (DLRL). Cette norme a pour objectif de définir une méthode de distribution des données qui est basée sur le paradigme de messagerie de type publication/souscription centré sur les données et qui offre une multitude de qualités de QoS. Les principaux avantages que cette norme procure sont les suivants (Xiong et al., 2006):

- l'indépendance de l'emplacement des interlocuteurs grâce au protocole de publication/souscription anonyme;
- l'évolutivité des communications grâce à la grande quantité de sujets, lecteurs et écrivains possibles;
- la portabilité et l'interopérabilité grâce à une interface et un protocole de communication standardisé.

2.2 Paradigmes de messagerie

Afin de bien comprendre les avantages de l'utilisation du paradigme de messagerie de publication/souscription qui est utilisée dans la norme DDS, ce modèle sera comparé aux différents paradigmes de messagerie couramment utilisés soit la messagerie point à point et la messagerie client-serveur.

2.2.1 Messagerie point à point

La messagerie point à point est la plus simple de toutes puisqu'elle consiste à établir une communication directe et unique entre deux dispositifs. De ce fait, un dispositif qui veut communiquer avec un autre doit absolument connaître l'adresse de ce dernier afin d'établir la communication. Une fois établie, la communication bénéficie d'une grande bande passante entre les deux dispositifs. Cependant, ce modèle n'est pas très adéquat lorsqu'un dispositif doit communiquer avec plusieurs dispositifs. De ce fait, en utilisant ce paradigme de messagerie, la conception et l'évolution d'un système qui doit communiquer avec plusieurs dispositifs peut être très complexe. Le principe de la messagerie point à point est illustré à la Figure 2.1 . Ce type de messagerie est beaucoup utilisé dans le domaine avionique notamment pour faire la communication entre les différents LRU.

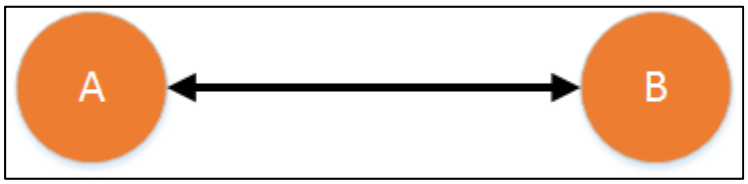


Figure 2.1 Messagerie point à point

2.2.2 Messagerie client-serveur

La messagerie client-serveur, quant à elle, est constituée d'un serveur et de plusieurs clients qui peuvent se connecter à ce dernier. La communication entre un client et le serveur se fait par des requêtes/réponse. L'information doit être centralisée au niveau du serveur avant de

pouvoir être envoyée aux différents clients. De ce fait, il y a une latence supplémentaire qui est engendrée puisque les différents clients ne peuvent pas directement communiquer entre eux, donc ce modèle est peu approprié pour les communications temps réel. L'évolution d'un système qui utilise cette messagerie est beaucoup plus simple qu'avec la messagerie point à point puisqu'il suffit d'ajouter un client au sein du modèle de communication et de connecter ce dernier au serveur pour l'ajouter au système. Le principe de la messagerie client-serveur est illustré à la Figure 2.2 . Ce type de messagerie est notamment utilisé pour faire la communication entre les applications de traitement trois dimensions et les systèmes d'affichage des cabines (CDS) des aéronefs (Lipinski et Ebrecht, 2014).

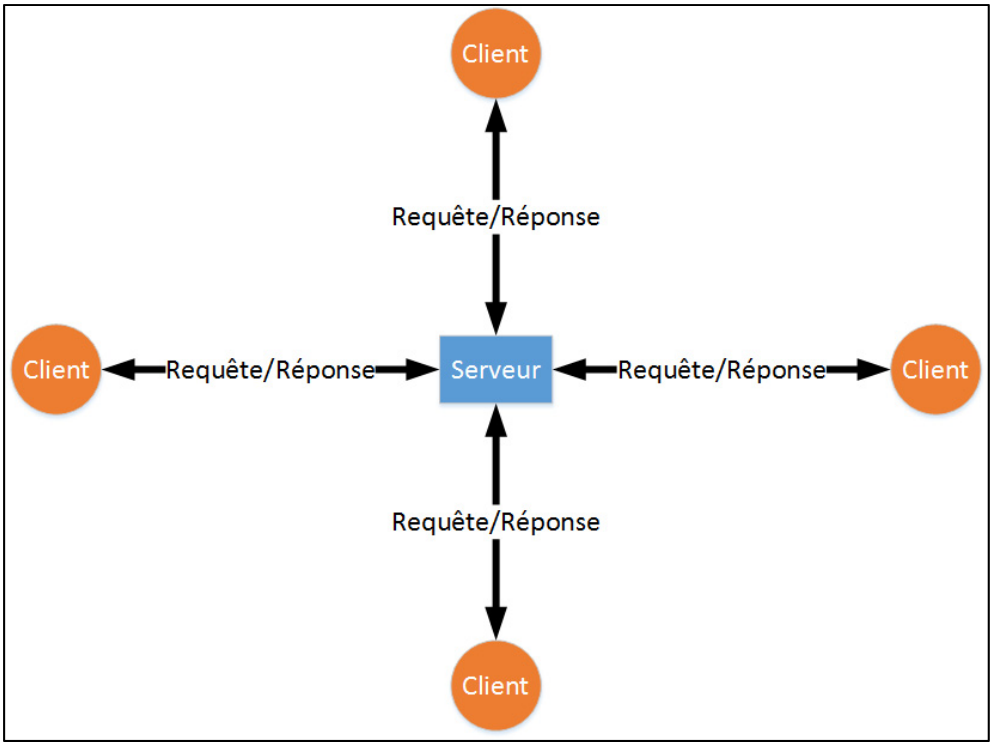


Figure 2.2 Messagerie client-serveur avec quatre clients

2.2.3 Messagerie publication/souscription

La messagerie publication/souscription est celle qui est utilisée dans la norme DDS. Cette messagerie est caractérisée par le fait que chaque point de communication peut être un

publicateur et/ou un abonné de données. Les données sont transmises directement par les différents publicateurs jusqu'aux abonnés et il est possible d'envoyer les données à plusieurs abonnés en même temps avec ce type de messagerie. Un avantage important de ce type de messagerie est qu'elle est flexible et facile à faire évoluer en ajoutant des nouveaux abonnés aux publicateurs existants ou bien des nouveaux publicateurs aux abonnés existants. Un autre avantage est que la communication entre un publicateur et un abonné est anonyme, donc que les deux interlocuteurs ne connaissent pas l'adresse de l'autre (Cam, Sahingoz et Sonmez, 2011). Le principe de la messagerie publication/souscription est illustré à la Figure 2.3 . Ce type de messagerie est peu utilisé dans le domaine avionique, mais est utilisé dans des applications temps réelles par exemple pour effectuer les communications entre les capteurs et les contrôleurs centraux dans les systèmes de commande et d'automatisation distribués (Calvo et al., 2010).

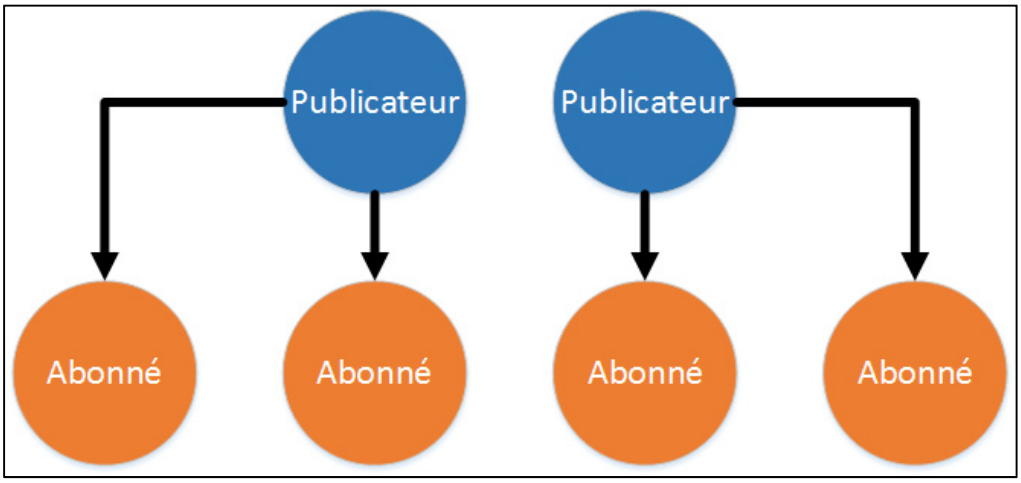


Figure 2.3 Messagerie de publication/souscription avec deux publicateurs et quatre abonnés

2.3 Définition d'un intergiciel DDS

Un intergiciel DDS est une implémentation de la norme DDS sous la forme d'une API. L'intergiciel DDS est une entité logicielle qui sert d'interface entre les applications usagées et le système d'exploitation servant ainsi de couche d'abstraction telle qu'illustrée à la Figure 2.4.

Il est possible de décomposer l'intergiciel DDS en trois parties soit la couche de publication/souscription basée sur les données (DCPS), la couche de reconstruction locale des données (DLRL) et le protocole de publication/souscription temps réel DDS (RTPS-DDS). Le contenu d'un intergiciel DDS est illustré à la Figure 2.5 . La norme DDS ne définit pas le protocole de communication qui doit être utilisé, donc l'utilisation et conformité à la norme sur le protocole RTPS-DDS est optionnelle, mais recommandée.

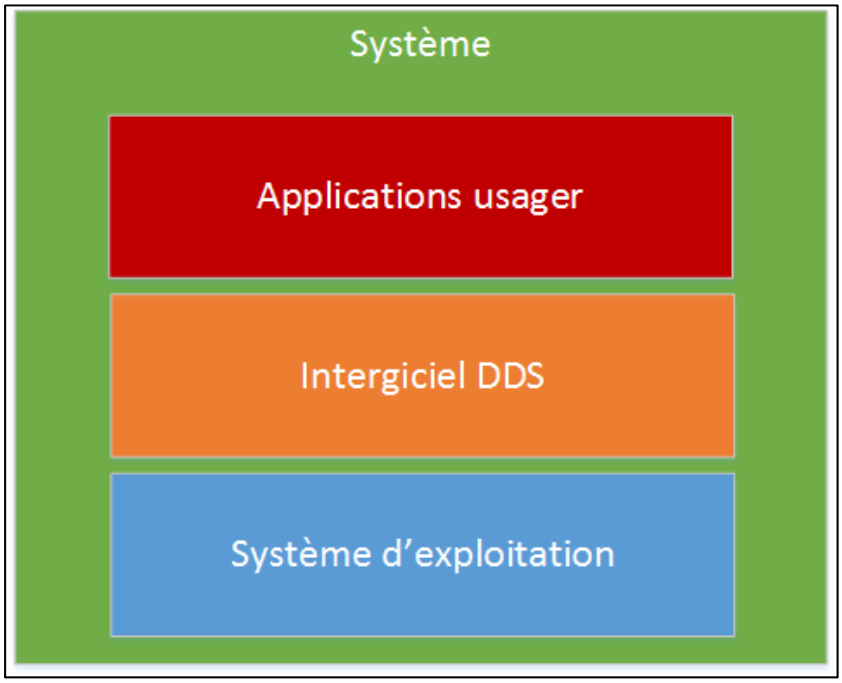


Figure 2.4 Emplacement de l'intergiciel DDS dans un système

La couche DLRL sert à fournir une API aux applications pour avoir accès aux différentes fonctionnalités décrites et implémentées dans la couche DCPS et permet aux applications d'accéder aux données qui transigent par l'intergiciel comme si elles étaient des données locales. La prochaine section traite en détail de la couche DCPS.

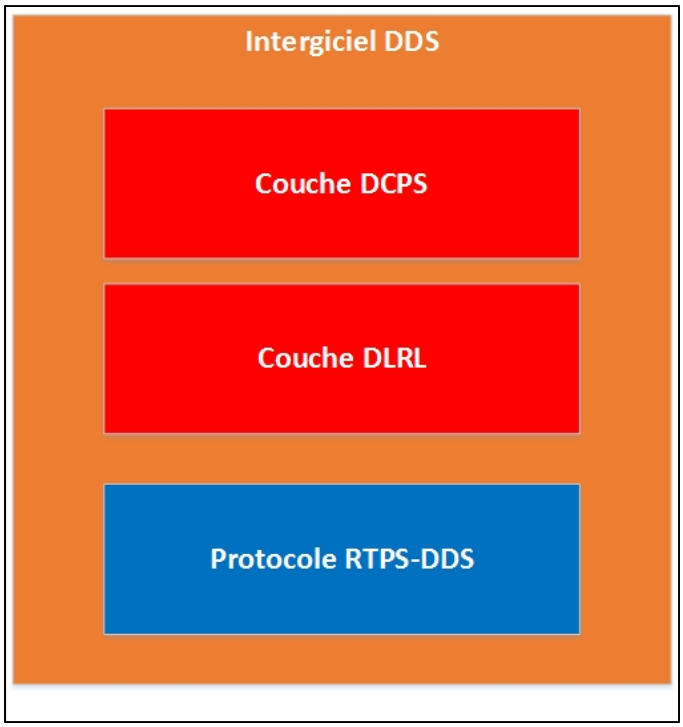


Figure 2.5 Contenu d'un intergiciel DDS

2.4 Couche DCPS

La couche de communication DCPS établit une méthode de communication qui est basée sur le paradigme de messagerie de publication-souscription. De plus, la couche DCPS est centrée sur les données qui doivent être transmises. Cela veut dire que la souscription entre un publicateur et un abonné est basée directement sur la nature des données qui doivent être transmises. Une particularité importante de la communication DCPS est que cette dernière a été conçue pour répondre aux besoins des systèmes temps réel ainsi que les applications où la réception des données est critique. La couche DCPS définit plusieurs entités qui doivent être utilisées afin d'effectuer une communication à l'aide d'un intergiciel DDS. La prochaine sous-section porte donc sur la description de ces entités.

2.4.1 Entités DDS

Tel que mentionné précédemment, la couche DCPS définit plusieurs entités qui doivent être créées afin d'établir les communications définies dans la norme. Chacune des différentes entités ont leurs propres utilités et certaines sont obligatoires, d'autres ne le sont pas. Les principales entités sont les suivantes: participants au domaine, publicateurs, abonnés, écrivains de données, lecteur de données et les sujets. Une description des différentes entités et de leurs interactions sera effectuée dans cette section. Un schéma de la création des différentes entités est illustré à la Figure 2.6.

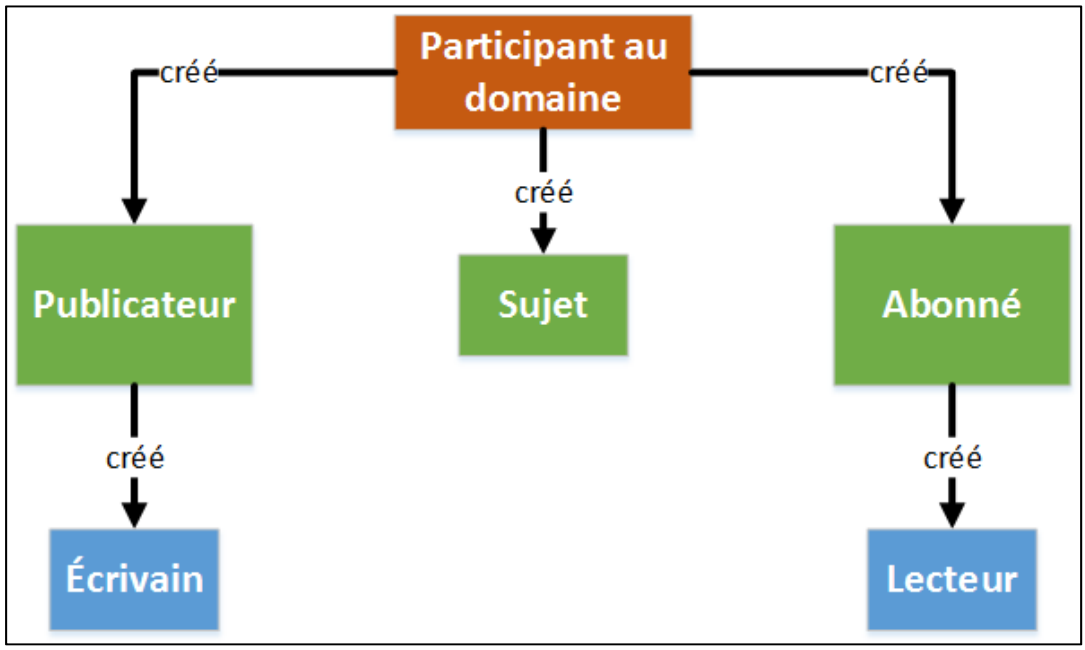


Figure 2.6 Schéma de la création des différentes entités DDS

Participant au domaine

Cette entité est indispensable puisqu'elle permet de créer plusieurs autres entités telles que les publicateurs, les abonnés ainsi que les sujets. Cette entité est liée à un domaine. Un domaine est un réseau logique de communication auquel seulement les participants qui appartiennent au même domaine peuvent communiquer entre eux. C'est donc une méthode permettant l'isolation des données afin qu'elles ne puissent être obtenues que par les bons participants.

Publicateur

Cette entité, qui est créée par un participant au domaine, permet de créer un ou plusieurs écrivains de données. Certaines QoS du publicateur seront attribuées aux écrivains ce qui permet de contrôler les QoS de plusieurs écrivains en même temps.

Écrivain

Cette entité, qui est créée par un publicateur, doit être associée à une entité sujet lors de sa création. Cette entité est indispensable afin de pouvoir envoyer des données avec l'intergiciel DDS.

Abonné

Cette entité, qui est créée par un participant au domaine, permet de créer un ou plusieurs lecteurs de données. Certaines QoS de l'abonné seront attribuées aux lecteurs ce qui permet de contrôler les QoS de plusieurs lecteurs en même temps.

Lecteur

Cette entité, qui est créée par un abonné, doit être associée à une entité sujet lors de sa création. Cette entité est indispensable afin de pouvoir recevoir des données avec l'intergiciel DDS.

Sujet

Le sujet est une entité importante puisque c'est elle qui va déterminer si les lecteurs et écrivains vont pouvoir communiquer ensemble. En effet, seulement les lecteurs et écrivains qui ont été créés avec le même sujet peuvent communiquer ensemble. De plus, un sujet doit être attribué à un type de données lors de sa création. Le type de donnée représente le format des données qui doivent être véhiculées par le canal de communication DDS. Un type de donnée peut être un type primaire tel qu'un *short* par exemple ou bien une structure ou union de plusieurs types primaires. Chaque sujet ne peut être attribué qu'à un seul type de données.

Échantillon

Bien que l'échantillon ne soit pas une entité DDS, ce dernier est tout de même important. En effet pour envoyer une donnée par un canal de communication DDS, cette dernière doit être mise dans un échantillon qui doit être du même type que le sujet avec lequel la donnée doit être publiée.

2.4.2 Configuration de l'intergiciel avec les QoS

Les QoS sont des mécanismes qui permettent d'assurer des performances de haute qualité pour les applications de haut niveau de criticité. Les performances qui peuvent être affectées par les QoS sont diverses telles que la latence maximale, la bande passante utilisée, la gigue et bien d'autres (Microsoft TechNet, 2003).

De ce fait, le comportement des communications effectuées à l'aide de l'intergiciel DDS peut être configuré avec différentes polices de QoS. En effet, chacune des différentes entités présentées dans la section 2.4.1 est munie d'une multitude de polices de QoS qui doivent être configurées en fonction du comportement désiré pour chaque communication. Chaque instance des différentes entités a son propre ensemble de polices de QoS qui peuvent être configurées soit lors de leur création ou en cours d'exécution. Les différentes polices de QoS permettent également de vérifier si les entités sont compatibles ensemble en comparant leurs QoS. En effet, la communication entre un lecteur et un écrivain ne peut s'effectuer que si leurs polices de QoS sont compatibles ce qui peut aider à détecter des erreurs de configuration lors du développement du code. Cette section traite donc des différentes polices de QoS qui sont disponibles et pertinentes aux communications dans un environnement avionique (Object Management Group, 2007). De ce fait, ce ne sont pas toutes les polices de QoS supportées qui seront traitées dans cette section.

Délai de publication/réception

Le délai de publication/réception (*Deadline*) est une QoS intéressante puisqu'elle permet de spécifier un intervalle maximal de publication désiré d'un écrivain ou un intervalle maximal

de réception désiré d'un lecteur. Par exemple, un délai de réception de 1 ms peut être défini à un lecteur. L'intergiciel peut ainsi aviser l'application si jamais il s'écoule plus de 1 ms entre la réception de deux données sur ce lecteur. De la même façon, un délai de publication peut être assigné à un écrivain pour être avisé si ce dernier n'envoie pas une nouvelle donnée dans le délai donné. Il est à noter que l'intergiciel ne peut pas assurer que le délai de publication/réception est respecté puisque cela dépend du code de l'application et du mode de transport utilisé, mais si jamais cet intervalle n'est pas respecté, il peut en aviser l'application.

Durabilité

La police de durabilité (*Durability*) détermine si les échantillons vivent plus longtemps que le temps pris pour les envoyer. Cette police peut posséder trois valeurs différentes soit volatile (*volatile*), transitoire (*transcient*) ou persistante (*persistant*). La valeur volatile permet seulement d'envoyer les échantillons aux abonnés qui existent déjà lors de l'envoi. La valeur transitoire permet de garder plus longtemps les échantillons envoyés pour potentiellement les renvoyer aux nouveaux abonnés qui sont créés après l'envoi original. La valeur persistante permet d'enregistrer les échantillons dans un endroit mémoire de façon permanente.

Historique

La police d'historique (*History*) sert à configurer un service qui permet de déterminer combien d'échantillons les lecteurs et écrivains gardent en mémoire avant de les écraser avec de nouveaux échantillons. Les options sont de garder soit tous les échantillons, seulement un nombre déterminé d'échantillons ou de ne garder que le dernier échantillon.

Budget de latence

La police de budget de latence (*Latency Budget*) permet de déterminer la latence maximale désirée entre l'envoi et la réception d'un échantillon. Si jamais cette latence n'est pas respectée, le lecteur en est informé.

Durée de vie

La police de durée de vie (*Lifespan*) sert à spécifier une durée maximale de validité pour une valeur qui a été écrite par l'écrivain. Cette police est intéressante au domaine avionique, dans lequel la fraîcheur des données est importante, puisqu'elle permet d'ignorer les données qui ne sont plus valides.

Vivacité

La police de vivacité (*Liveliness*) permet de configurer le mécanisme qui est utilisé pour déterminer la présence (ou l'absence) d'une entité dans le système. Il est également possible de configurer le temps à partir duquel une entité est considérée comme étant morte. Cette police est intéressante dans le cadre du domaine avionique dans lequel il est important de savoir lorsqu'un interlocuteur n'est plus présent dans le système et ainsi prendre les actions nécessaires.

Propriétaire et force de propriétaire

La police de propriétaire (*Ownership*) est applicable aux lecteurs et écrivains et permet de déterminer si plusieurs écrivains peuvent envoyer des données au même lecteur. Il y a deux valeurs possibles pour la police de propriétaire soit partagé (*shared*) ou exclusif (*exclusive*). Un propriétaire partagé signifie que plusieurs écrivains peuvent envoyer des données au même lecteur. Un propriétaire exclusif veut dire que seulement un seul écrivain peut envoyer des données à un lecteur. C'est l'écrivain avec la force de propriétaire la plus élevée qui peut ainsi envoyer des données au lecteur.

Le propriétaire exclusif est une option intéressante au domaine avionique puisqu'il permet ainsi de faire de la redondance d'écrivains. Des tests de redondance avec cette police de QoS sont effectués dans le chapitre 5.

Partition

La police de partition (*Partition)* permet d'introduire une notion de partitions logiques parmi les sujets qui sont visibles par le publicateur et l'abonné. Cette police ajoute donc une

condition supplémentaire pour qu'un lecteur et un écrivain puissent communiquer ensemble puisqu'ils doivent également faire partie de la même partition en plus d'être inscrits au même sujet et faire partie du même domaine. Cette police de QoS est particulièrement intéressante au domaine avionique dans lequel les systèmes IMA fonctionnent déjà avec une notion de partitionnement temporel et spatial.

Fiabilité

Cette police de fiabilité (*reliability)* permet de configurer le niveau de fiabilité que l'on veut pour une communication. Les deux options valides sont fiable (*reliable*) et meilleur effort (*best effort*). Une fiabilité meilleur effort permet la perte d'échantillons et ne nécessite pas le renvoi des échantillons perdus. Une fiabilité avec la valeur fiable va forcer l'écrivain à renvoyer les différents échantillons qui n'ont pas été reçus par les écrivains. Cette police est intéressante puisqu'elle permet à un intergiciel DDS d'utiliser des protocoles de transport qui n'ont pas de mécanismes de fiabilité tel que le UDP/IP pour ainsi forcer le renvoi des échantillons qui n'ont pas été reçus.

Priorité de transport

La police de priorité de transport (*Transport Priority*) permet de donner des priorités différentes aux différents écrivains. Cette politique est intéressante dans le domaine avionique étant donné les différents niveaux de criticité des applications qui communiquent parfois sur le même bus de données. Avec cette police, il est ainsi possible de donner une priorité plus élevée aux applications critiques que les applications non critiques pour que ces dernières ne prennent pas une marge trop importante de la bande passante du canal de communication.

Il est important de noter que l'efficacité de cette politique de QoS est dépendante du transport utilisé. En effet, si le transport ne fournit aucun mécanisme de contrôle de priorités, alors cette politique de QoS ne fera rien. Dans le cas du protocole UDP/IP, cette politique va modifier les bits du type de service (TOS) de l'entête UDP afin d'obtenir la priorité désirée.

De plus, tous les composants réseau tels que les interrupteurs réseau et les routeurs doivent supporter les mécanismes de contrôle de priorités afin d'obtenir le comportement désiré.

Limite de ressources

La police de limite de ressources (*Ressource Limits*) permet de définir le maximum de mémoire qui peut être alloué par les différentes entités. De ce fait, il est ainsi possible de contrôler la quantité de mémoire maximale qui est allouée pour le fonctionnement de l'intergiciel DDS. Cette police de QoS est importante pour le domaine avionique dans lequel la quantité de mémoire allouée pour chacune des applications doit être bornée et minimisée.

Sélection de transport

La police de sélection de transport (*TransportSelection*) permet de spécifier quel transport est utilisé pour envoyer ou recevoir des données. Cette police est utile au domaine avionique puisqu'elle permet d'encapsuler plusieurs modes de transports à travers une seule API standardisée. Cela réduirait donc considérablement la complexité de la gestion de tous les modes de transports que l'on peut retrouver sur une plateforme avionique.

Il a donc été mis en évidence, dans cette section, que l'ensemble des QoS offertes par la norme DDS permettent une grande flexibilité et permettent de bien contrôler le comportement des communications afin d'obtenir le comportement désiré.

2.5 Protocole de communication RTPS-DDS

La norme DDS ne spécifie pas le protocole qui doit être utilisé pour implémenter la méthode de communication publication-souscription. De ce fait, le Object Management Group (OMG) a créé une norme qui définit un protocole de communication qui est adapté aux besoins des fonctionnalités décrites dans la norme DDS (Object Management Group, 2009). De plus, ce protocole standardisé permet à plusieurs implémentations différentes d'intergiciels DDS de communiquer facilement ensemble. Ce protocole n'est cependant pas utilisable par lui même et doit reposer sur un protocole de transport tel que le UDP. De ce fait, les performances de

l'intergiciel DDS sont fortement dépendantes du protocole de transport sur lequel repose le RTPS-DDS. L'utilisation de ce protocole standardisé dans l'implémentation d'un intergiciel DDS n'est pas obligatoire, mais recommandée pour augmenter ainsi l'interopérabilité de l'intergiciel avec les autres implémentations.

2.5.1 Processus de découverte

Le protocole RTPS-DDS définit un processus de découverte qui permet ainsi aux différentes entités DDS de se connecter les unes aux autres automatiquement et de façon anonyme. L'avantage de cette fonctionnalité est que la gestion des communications entre les différents nœuds est simplifiée puisque l'ajout de nouveaux nœuds au sein d'un système existant ne nécessite pas une connaissance approfondie des autres nœuds.

La norme RTPS-DDS définit deux protocoles de découverte soit le protocole de découverte des participants (PDP) et le protocole de découverte des points d'arrivée (EDP). La découverte s'effectue en deux étapes soit la découverte des participants avec le PDP et ensuite la découverte des lecteurs/écrivains avec le EDP. Le processus de découverte est décrit en détail dans les prochaines sous-sections.

2.5.2 Protocole de découverte des participants (PDP)

Le PDP est utilisé pour que deux participants sur le réseau se découvrent et échangent leurs informations. La Figure 2.7 illustre le fonctionnement du processus de découverte des participants ainsi que le contenu des messages échangés. Cette figure présente des entités DDS prédéfinies (un lecteur et un écrivain) qui sont créées pour débuter un échange de données entre les deux participants. Ces entités communiquent sur des ports prédéfinis dans la norme RTPS-DDS, donc ne nécessitent aucune information supplémentaire pour pouvoir communiquer ensemble. La Figure 2.7 présente également les données qui sont échangées entre les deux participants soit l'adresse IP, le numéro de port, les polices de QoS et l'identificateur globalement unique (GUID) des participants concernés. De ce fait, les

participants ont ensuite toutes les informations nécessaires afin de pouvoir se connecter ensemble.

Ces informations sont envoyées périodiquement sur le réseau pour ainsi signaler la présence du participant aux nouveaux participants potentiels et pour pouvoir vérifier la vivacité des participants. La période d'envois des informations est contrôlée par la QoS de vivacité du participant et va ainsi déterminer la quantité de bande passante qui sera utilisée par les métadonnées. De ce fait, il doit avoir un compromis entre la vitesse de détection de la terminaison (normale ou anormale) d'un participant et la quantité de bande passante utilisée par les données de vivacité (métadonnées).

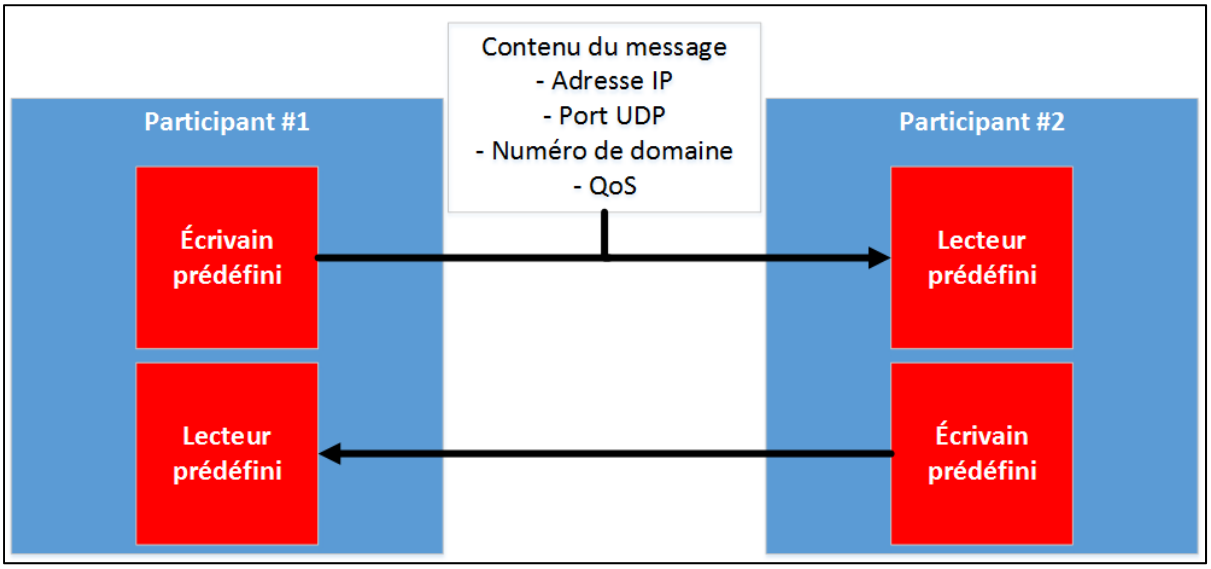


Figure 2.7 Fonctionnement du processus de découverte des participants

2.5.3 Protocole de découverte des points d'arrivée (EDP)

Lorsqu'un lecteur ou écrivain est créé, l'EDP est utilisé pour que les participants puissent échanger les nouvelles informations sur les différents lecteurs et écrivains qu'ils contiennent. Les communications utilisées pour effectuer le processus de découverte des lecteurs/écrivains sont effectuées à partir d'entités DDS qui sont prédéfinies par la norme RTPS-DDS. Il est ainsi possible pour un participant de connecter ses lecteurs et écrivains avec les autres

lecteurs/écrivains qui appartiennent aux autres participants. La connexion entre un lecteur et un écrivain est établie seulement s'ils sont du même sujet et qu'ils ont des polices de QoS compatibles. La Figure 2.8, illustre le fonctionnement du processus de découverte des lecteurs et écrivains. Cette figure présente que les messages qui sont échangés pour effectuer la connexion entre les lecteurs et écrivains incluent les polices de QoS ainsi que l'identificateur globalement unique des différents lecteurs et écrivains qui appartiennent aux participants concernés.

De plus, suite à leur découverte, les écrivains vont envoyer un message périodique de vivacité aux différents lecteurs à une période définie dans les polices de QoS des écrivains. De ce fait, tout comme pour les participants, il doit y avoir un compromis entre la vitesse de détection de la terminaison (normale ou anormale) des écrivains et la quantité de bande passante utilisée par les messages de vivacité des écrivains.

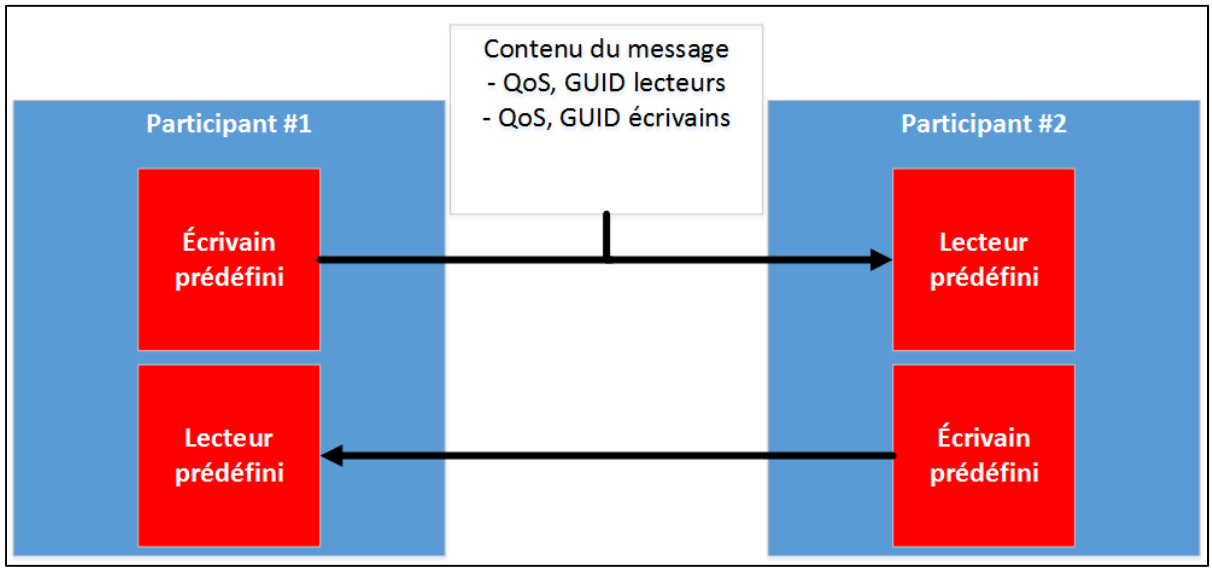


Figure 2.8 Fonctionnement du processus de découverte des points d'arrivée

2.5.4 Configuration du processus de découverte

Le processus de découverte, bien qu'automatique, peut être configuré. En effet, il est possible de configurer le processus de découverte afin que ce dernier n'accepte pas de connecter les

participants qui ne se retrouvent pas dans une liste de descripteurs de pair (*peer descriptors*) préétablie. Un descripteur de pair contient généralement le participant ID, le nom du transport utilisé, son adresse réseau et son adresse transport tel qu'illustré à la Figure 2.9.

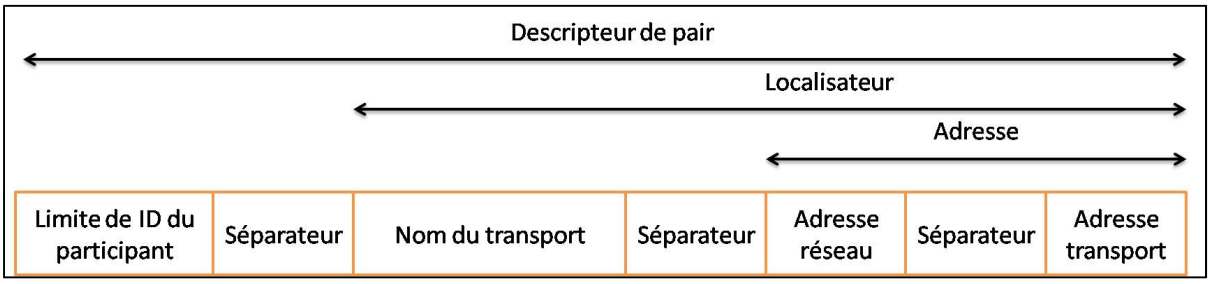


Figure 2.9 Contenu d'un descripteur de pair
Adaptée de Real Time Innovation (2012)

De ce fait, il est ainsi possible d'avoir une configuration statique des connexions qui seront établies entre les différents participants. Cette option est importante, car le domaine avionique préconise souvent une configuration statique au démarrage afin de minimiser les efforts de certification.

2.6 Communication entre les entités DDS

Dans cette section, un simple exemple de communication DDS sera décrit en détail afin de bien expliquer le fonctionnement de la communication DDS. L'exemple est une application de système de contrôle de vol automatique (AFCS) qui nécessite les données de capteurs, et ce, sur le domaine #1 tel qu'illustré à la Figure 2.10. Cette figure présente les différentes entités que les deux applications doivent créer afin de communiquer ensemble.

L'AFCS doit donc créer un participant au domaine et lui indiquer que ce participant pourra communiquer sur le domaine #1. Par la suite, le participant doit ensuite créer un abonné ainsi que le sujet "capteur" auquel seront attribuées des données d'un type "structure_capteur" qui pourrait contenir une multitude de champs tels que la vitesse, l'altitude, le roulis, le taux de roulis, le tangage et le taux de tangage de l'aéronef tel qu'illustré à la Figure 2.11. Ensuite, l'abonné devra créer un lecteur et lui attribuer le sujet "capteur" pour lui indiquer que ce

lecteur est intéressé à recevoir les données de sujet "capteur". Pour finir, il ne suffit que d'appeler les fonctions de lecture et d'utiliser le lecteur créé.

De son côté, l'application capteurs doit également créer un participant au domaine et lui indiquer que ce participant pourra communiquer sur le domaine #1. Par la suite, le participant doit créer un publicateur ainsi que le sujet "capteur" auquel seront attribuées des données de type "structure_capteur" (Figure 2.11). Ensuite, le publicateur devra créer un écrivain et lui attribuer le sujet "capteur" pour lui indiquer que cet écrivain ne peut envoyer que des données de sujet "capteur". Pour finir, il ne suffit que d'appeler les fonctions d'écriture et d'utiliser l'écrivain qu'il a créé.

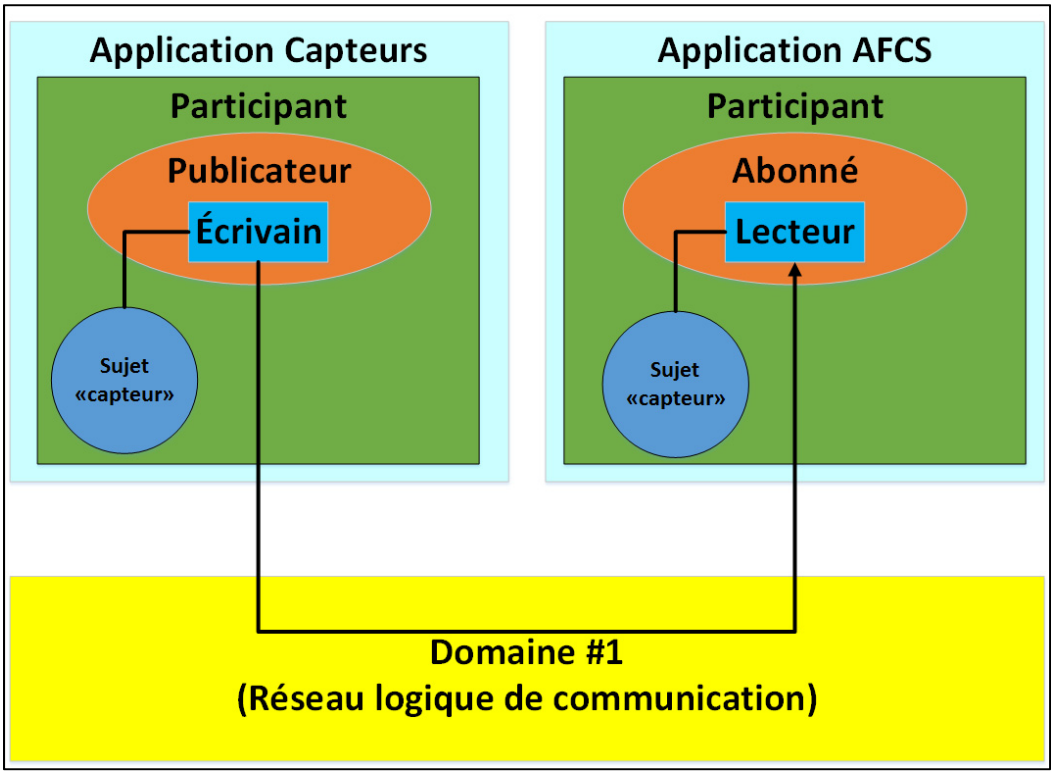


Figure 2.10 Communication DDS entre un AFCS et des capteurs

```
struct structure_capteur
{
    float vitesse;
    float altitude;
    double roulis;
    double taux_roulis;
    double tangage;
    double taux_tangage;
}
```

Figure 2.11 Type de donnée "structure_capteur"

2.7 Autres intergiciels de communication

Bien qu'il existe plusieurs autres normes d'intergiciels de communications, ce ne sont pas tous les normes qui sont appropriées aux besoins de systèmes temps réel tel que les systèmes avioniques. De ce fait, cette section porte sur la comparaison entre la norme DDS et les normes d'intergiciels réseau mentionnés par le consortium FACE dans la section 1.2. La norme RT-CORBA (*Real Time Common Object Request Broker Architecture*) ainsi que les Services Web (*Web Services*) sont ainsi comparés à la norme DDS.

RT-CORBA

Le RT-CORBA est une norme définie par le OMG. L'intergiciel défini dans cette norme permet aux applications de configurer et de contrôler les ressources du processeur, les ressources de communication (à travers les propriétés du protocole) et les ressources mémoires (Schmidt et Kuhns, 2000). Ces ressources sont configurables à travers une API standardisée qui fournit une multitude de QoS. Cette norme ne se limite pas seulement qu'à la configuration des communications contrairement à la norme DDS. Étant donné que cette section porte sur la comparaison de la norme RT-CORBA avec la norme DDS, cette section se limite qu'aux fonctionnalités de configurations des communications offertes dans la norme RT-CORBA.

Tout comme l'intergiciel DDS, un intergiciel RT-CORBA doit être muni de mécanismes et polices qui supportent la garantie de ressources de communication à travers une multitude de

QoS. Sur une plateforme utilisant un intergiciel RT-CORBA, les noeuds sont munis d'un ORB (*Object Request Brokers*) qui sert de couche d'abstraction entre les fonctions bas niveau du système d'exploitation et les applications. La communication entre les ORBs est transparente de la location tout comme dans la norme DDS. L'intergiciel RT-CORBA doit également fournir une API standardisée qui permet ainsi aux applications de transmettre leurs besoins en ressources. Cependant, le paradigme de messagerie client-serveur est utilisé dans la norme RT-CORBA contrairement au paradigme de publication/souscription utilisé dans la norme DDS. L'utilisation de ce paradigme de messagerie rend plus complexe l'évolution de systèmes existants en comparaison avec l'utilisation d'un intergiciel DDS. De plus, il est important de noter que la norme RT-CORBA est une extension de la norme CORBA et cette dernière n'est pas appropriée pour les applications temps réel (Object Management Group, 2005). De ce fait, la norme RT-CORBA n'est qu'une extension à une norme qui n'était pas originalement conçue pour les applications temps réel alors que la norme DDS a été conçue spécifiquement pour ce type d'applications.

La Figure 2.12 illustre le fonctionnement le modèle de communication RT-CORBA. Cette figure illustre que la communication entre une application serveur et une application client doit passer par l'entremise d'ORBs en utilisant le protocole GIOP (*General Inter-ORB Protocol*).

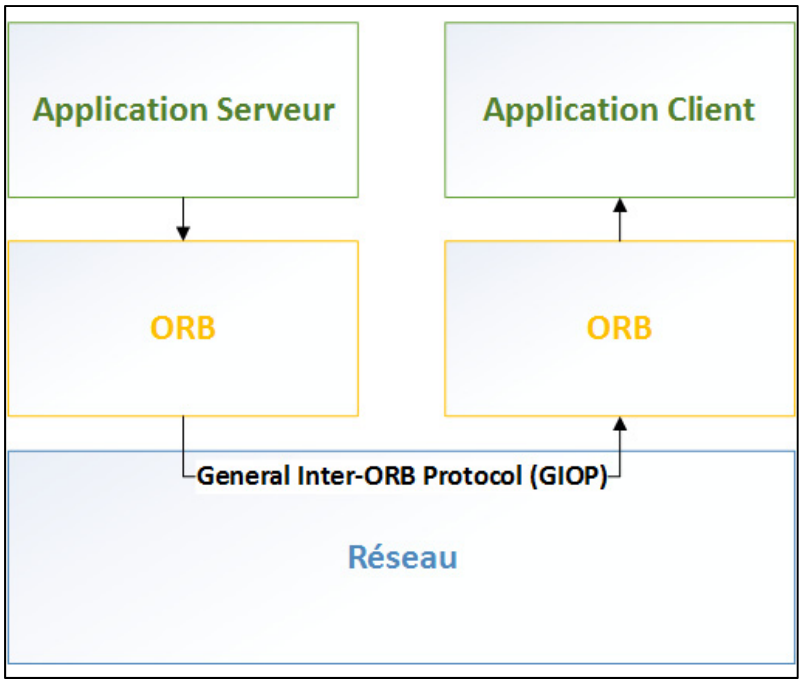


Figure 2.12 Communication RT-CORBA

Le TAO, un exemple d'implémentation de RT-CORBA, a été utilisé par Boeing en tant qu'architecture de calcul de mission avionique (*avionic mission computing architecture*) (Schmidt et Kuhns, 2000).

Services Web

Les services web sont composés de trois services fondamentaux: le consommateur de service, le fournisseur de service et le registre de service tel qu'illustré à la Figure 2.13.

Le fournisseur de service permet de publier des données en indiquant dans le service de registre comment obtenir les données. Le consommateur de service peut ensuite consulter le service de registre pour savoir quelles données sont disponibles et comment les obtenir. Les services web sont basés sur un paradigme de messagerie de client-serveur contrairement à la norme DDS qui utilise un paradigme de messagerie de publication/souscription. De ce fait, les services web sont moins flexibles sur l'évolution d'un système existant que l'intergiciel DDS. Le SOAP (*Simple Object Access Protocol*) est un protocole de messagerie XML

(*Extensible Markup Language*) qui utilise le HTTP (*HyperText Transfert Protocol*) et qui permet le transfert de données entre les fournisseurs et les consommateurs.

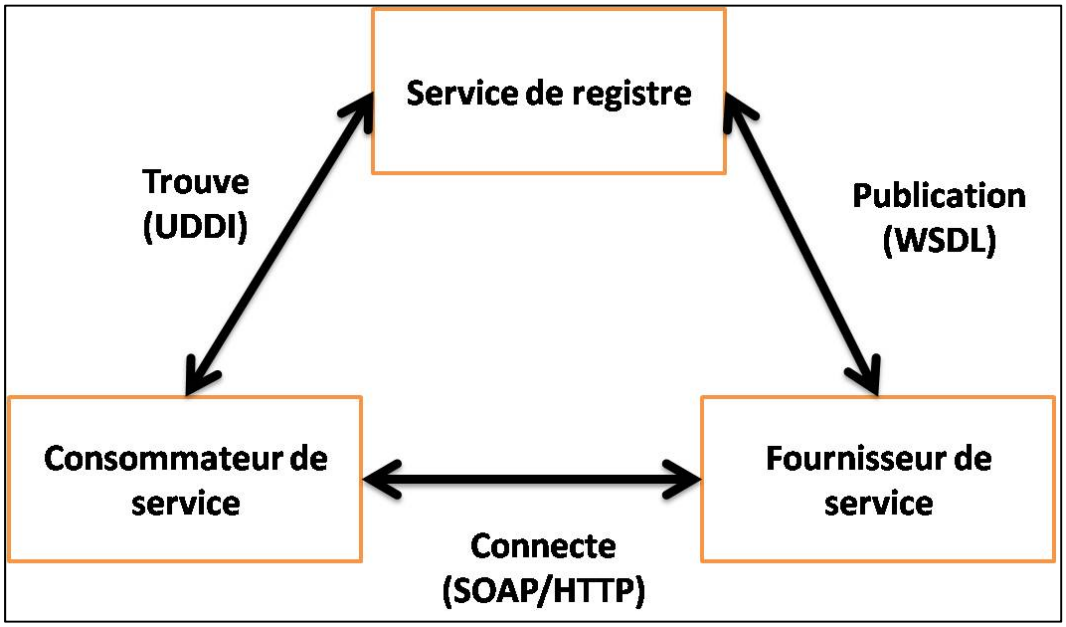


Figure 2.13 Services fondamentaux des services web
Adaptée de Kulkarni, Nayak et Rao (2012)

Les services web représentent la technologie d'intergiciel la plus utilisée dans l'entreprise des systèmes de *Command, Control, Communication, Computer and Intelligence* (C4I). Les systèmes C4I permettent d'obtenir des informations du champ de bataille militaire (Kulkarni, Nayak et Rao, 2012). Suite aux expérimentations effectuées dans (Kulkarni, Nayak et Rao, 2012), les auteurs ont conclu que le DDS offre un meilleur contrôle des QoS, une meilleure latence et une moins grande dégradation de la bande passante lorsque la taille des paquets change. Ce type d'intergiciel introduit également un grand en-tête aux paquets contrairement au DDS.

On constate donc que ce type d'intergiciel est peu approprié pour les applications avioniques étant donné ses faibles performances, son obligation d'utiliser les documents XML ainsi que son utilisation du service de registre qui est un point individuel de défaillance. Son utilisation du paradigme de messagerie de client-serveur engendre également une plus grande utilisation

de bande passante que le paradigme de publication/souscription. Son inclusion au sein de la norme FACE est probablement due a sa popularité au sein des entreprises civiles dans lesquelles les hautes vitesses de transferts et où les requis de latence ne sont pas très contraignants.

2.8 Choix de l'intergiciel DDS

Il existe plusieurs intergiciels DDS commerciaux qui sont disponibles sur le marché, donc une comparaison de ces derniers a dû être effectuée afin de faire un choix. Le choix de l'intergiciel DDS qui a été utilisé dans le cadre de cette recherche s'est donc basé sur plusieurs critères soit les systèmes d'exploitation supportés, les langages de programmation supportés, la disponibilité de l'intergiciel ainsi que les outils de développement disponibles. Le Tableau 2.1 présente les différents intergiciels DDS considérés ainsi que ce qu'ils supportent.

Suite à la comparaison des différents intergiciels DDS offerts, l'intergiciel Connext Micro a été choisi. Plusieurs facteurs ont influencé le choix de cet intergiciel. Le premier facteur est qu'il supporte une multitude de systèmes d'exploitation ainsi que le langage C. Le deuxième facteur est la multitude d'outils de développements qui viennent avec le logiciel et qui permettent de faciliter les efforts de développement. Le troisième facteur est la disponibilité du logiciel à travers une licence académique gratuite de recherche et développement. Finalement, le dernier facteur important de la décision est que Connext Micro est en cours de certification pour la norme avionique RTCA DO-178B/C.

Tableau 2.1 Comparaison des différents intergiciels DDS considérés

Intergiciel DDS	Windows	Linux	VxWorks	Langage C	Code Source ouvert	License académique	Certification RTCA DO-178B/C	Outils de développement
Connext DDS Professional	X	X	X	X		X		Beaucoup
Connext Micro	X	X	X	X		X	X	Beaucoup
Open DDS	X	X			X			Peu
OpenSplice RTE	X	X	X	X				Beaucoup
CoreDX DDS	X	X	X	X		X		Beaucoup
InterCOM DDS	X	X	X					Peu
MilSOFT DDS	X	X	X					Peu
ORTE	X	X			X			Aucun

2.9 Conclusion

Le but de ce chapitre était d'évaluer la norme DDS et de mettre en évidence les fonctionnalités pertinentes au domaine avionique que cette dernière offre.

Tout d'abord, il a été vu que la norme DDS définit une API qui permet d'établir des communications basées sur le paradigme de messagerie de publication/souscription centrée sur les données. Il a été mis en évidence que ce paradigme de messagerie offre une plus grande flexibilité et facilite l'évolution de système existant comparé aux autres paradigmes de messagerie. Ensuite, il a été vu qu'un intergiciel DDS fournit une multitude de polices de QoS. Ces polices de QoS servent à configurer le comportement des communications effectuées à l'aide de l'intergiciel DDS. De plus, ces dernières offrent une grande flexibilité telle que le contrôle de la priorité des communications et de la mémoire utilisée par l'intergiciel DDS par exemple. Cette flexibilité est idéale pour les applications avioniques puisque ces dernières nécessitent une configuration approfondie afin d'obtenir le comportement désiré. Par la suite, il a été vu que le processus de découverte, utilisé dans l'intergiciel DDS, permet aux différents interlocuteurs de se connecter ensemble sans avoir une connaissance approfondie des autres interlocuteurs. De plus, ce processus est

complètement configurable afin de limiter la bande passante utilisée par le processus ainsi que modifier la période de vérification de la vivacité des interlocuteurs. Ensuite, l'intergiciel DDS a été comparé à différents autres intergiciels de communications. Il a été mis en évidence que l'intergiciel DDS est plus approprié pour le domaine avionique que l'intergiciel RT-CORBA et que les services web. Pour finir, l'intergiciel DDS Connext Micro a été choisi parmi plusieurs intergiciels DDS commerciaux pour être utilisé dans les tests des prochains chapitres.

Dans le prochain chapitre, des tests seront effectués afin de caractériser les performances des communications avec l'intergiciel DDS Connext Micro.

CHAPITRE 3

CARACTÉRISATION DES PERFORMANCES DE L'INTERGICIEL

Ce chapitre porte sur la caractérisation des performances de l'intergiciel DDS choisi dans le chapitre précédent. Le but de ce chapitre est d'évaluer l'impact de l'ajout d'un intergiciel DDS sur les performances des communications ainsi que vérifier si les performances obtenues avec l'intergiciel DDS satisfont les requis du domaine avionique.

Idéalement les expériences seraient effectuées dans un environnement avionique afin d'obtenir des résultats qui représentent bien les performances d'un intergiciel DDS dans un tel environnement. Cependant, on n'a pas accès à un environnement avionique dans notre laboratoire de recherche, donc les performances de l'intergiciel DDS sont effectuées sur des ordinateurs personnels avec le système d'exploitation Windows 7. De ce fait, on a émis l'hypothèse que les performances observées en effectuant les même tests dans un environnement avionique seraient meilleures que dans l'environnement Windows 7. Cette hypothèse se base sur l'article de (Schmidt, Deshpande et O'Ryan, 2002) dans lequel les auteurs font des tests de latence avec un intergiciel de communication RT-CORBA, et ce, avec différents systèmes d'exploitation. Les résultats de cet article montrent que la latence moyenne et la gigue obtenue avec un système d'exploitation temps réel tel que QNX et VxWorks sont plus petites que ce qui est obtenu avec un système d'exploitation Windows tel qu'illustré dans le Tableau 1.1.

Dans ce chapitre, les métriques de performances qui seront évaluées sont la latence ainsi que le débit de données. Les performances de latence de la communication sans l'utilisation d'un intergiciel DDS sont tout d'abord mesurées. Ces performances servent de base de comparaison pour mesurer l'impact de l'utilisation d'un intergiciel DDS sur la latence d'une communication. Par la suite, un intergiciel DDS est ajouté au scénario de test et les performances de latence de la communication sont mesurées. Ces dernière sont ensuite comparées aux performances de la communication sans intergiciel ainsi qu'aux performances

généralement désirées dans le domaine avionique. Finalement, des tests d'utilisation de la bande passante de l'intergiciel DDS avec différentes tailles de paquets sont effectués.

3.1 Caractéristiques de Connext Micro

Les implémentations d'intergiciel DDS ne sont pas toutes identiques, donc cette section porte sur les différentes caractéristiques propres à Connext Micro. Cette section présente les caractéristiques qui sont pertinentes à l'utilisation de Connext Micro ainsi qu'à la compréhension des résultats obtenus.

3.1.1 Limitations particulières de Connext Micro

Étant donné que Connext Micro est un produit qui est encore en développement, il ne supporte pas encore toutes les fonctionnalités qui sont définies dans la norme DDS.

Il est important de mentionner que Connext Micro ne supporte que le transport UDPv4. De plus, Connext Micro ne supporte pas toutes les QoS définies dans la norme DDS. Les QoS qui ne sont pas du tout supportée ainsi que celles qui ne sont supportées que partiellement sont identifiées dans le Tableau 3.1.

Tableau 3.1 Liste des QoS pas supportées ou bien partiellement supportées par Connext Micro

Partiellement supportées	Pas du tout supportées
• Durabilité • Historique	• Budget de latence • Durée de vie • Priorité de transport

3.1.2 Opération de Connext Micro dans le modèle OSI

Afin de comprendre les limitations possibles de l'intergiciel Connext Micro, une présentation de son opération sur les différentes couches du modèle OSI est présentée. Connext Micro

opère sur les couches OSI suivantes: la couche application (7), la couche de présentation (6), la couche de session (5) et la couche de transport (4) (Real Time Innovation, 2013) tel qu'illustré à la Figure 3.1.

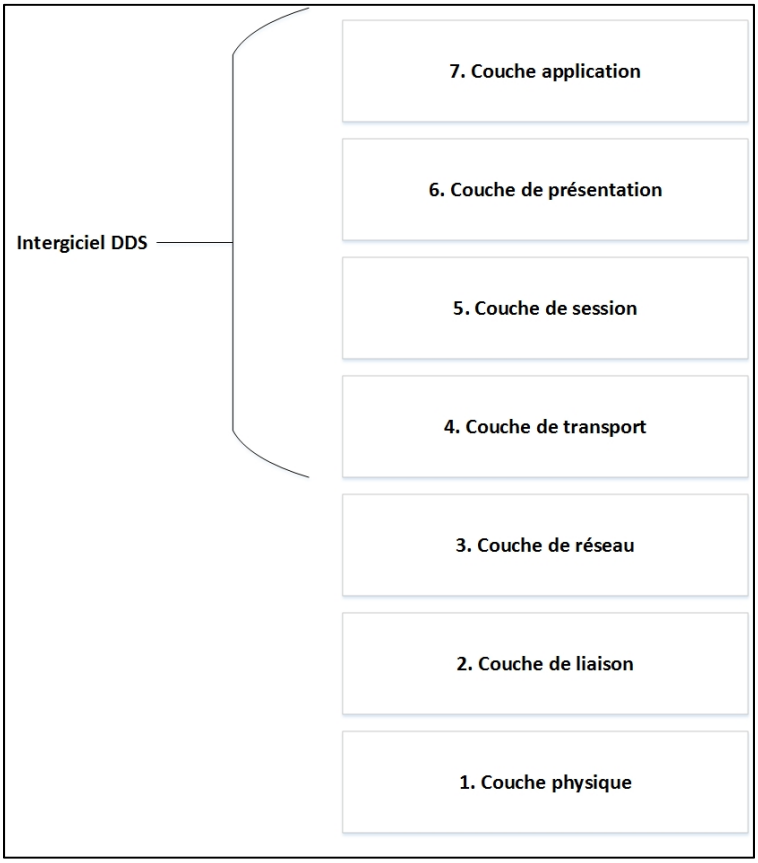


Figure 3.1 Opération de Connext Micro dans le modèle OSI

De ce fait, Connext Micro doit utiliser une couche de réseau (3), une couche de liaison (2) et une couche physique (1) qu'il ne contrôle pas directement afin de fonctionner. Cela implique que les performances obtenues par Connext Micro ne sont pas entièrement sous son contrôle puisqu'elles dépendent également des limitations de ces couches. Les limitations peuvent, par exemple, être une limite de débit de données ainsi qu'une latence minimale qui est causée par les couches 1 à 3. De plus, certaines QoS offertes par Connext Micro sont également dépendantes de ces couches.

3.1.3 Modèle de fils d'exécution de Connext Micro

Afin de comprendre les interactions entre l'intergiciel DDS et les applications usager, une description des différents fils d'exécution (*threads*) utilisés par l'intergiciel est nécessaire. Les participants au domaine utilisent trois types de fils d'exécution soit les fils d'exécution de base de données, les fils d'exécution d'évènement et les fils d'exécution de réception. Il est possible de changer la priorité des différents fils d'exécution à partir des différentes QoS offertes par l'intergiciel DDS. Cette option est particulièrement utile pour les applications temps réel dans lesquelles changer la priorité des différents fils d'exécution au sein d'une application est généralement désirable afin d'obtenir les performances voulues. Cette section présente donc les différents fils d'exécution de Connext Micro.

Fil d'exécution de base de données

Ce fil d'exécution contient la base de données des informations sur les différentes entités connues par le participant au domaine. Ce type de fil d'exécution n'ayant aucun impact sur les performances des communications, il est préférable de lui allouer une faible priorité.

Fil d'exécution d'évènement

L'intergiciel DDS doit périodiquement vérifier l'état de plusieurs événements périodiques ou bien des événements activés par interruptions. Plusieurs polices de QoS utilisent ce fil d'exécution pour faire leurs vérifications tel que la police de délai d'envois/réception ainsi que la police de vivacité. De plus, la réception de données asynchrones à l'aide des récepteurs (*listener*) est effectuée dans le fil d'exécution d'événement. De ce fait, il important d'attribuer une haute priorité à ce fil d'exécution afin d'obtenir de faibles latences.

Fil d'exécution de réception

Ce fil d'exécution sert a désérialiser et enregistrer les paquets dans la queue de réception lors de la réception de ces derniers. De plus, il se charge également de traiter les paquets du processus de découverte. Il est possible d'avoir plusieurs de ces fils d'exécution par participants au domaine en fonction des QoS des participants, lecteurs et écrivains. Par

défaut, il y a deux fils d'exécution de réception créés pour le processus de découverte et deux pour les données utilisateur. Pour faire une analogie avec les *sockets* UDP, il y a un fil d'exécution de réception de créé par *socket* UDP utilisé. Afin d'obtenir de faibles latences, il est préférable de donner une priorité élevée à ces fils d'exécution.

3.1.4 Réception de données avec Connext Micro

L'intergiciel Connext Micro supporte deux méthodes d'accès aux données ainsi que trois méthodes de réception des données. Les méthodes d'accès supportées sont l'accès par lecture et l'accès par retrait. L'accès par lecture ne fait que lire la donnée dans la queue de réception et laisse cette donnée disponible pour les autres lecteurs alors que l'accès par retrait enlève la donnée de la queue de réception ce qui fait qu'aucun autre lecteur ne peut accéder à cette donnée. Seulement l'accès par retrait est abordé dans cette sous-section puisque c'est cette méthode d'accès qui a été utilisée pour l'ensemble de cette recherche.

On présente ci-dessous les trois différentes méthodes de réception des données supportées par Connext Micro soit la réception par sondage, la réception asynchrone et la réception bloquante.

Réception par sondage

La réception par sondage consiste à essayer de lire ou retirer des données sans savoir s'il y a des données de disponibles dans la queue de réception. Cette méthode de réception est illustrée à la Figure 3.2. Cette figure illustre que les données reçues du réseau arrivent tout d'abord dans le fil d'exécution de réception. Ensuite, ce dernier fait la désérialisation des données et les enregistre dans la queue de réception. Finalement, l'application usagée doit faire une lecture non bloquante afin d'obtenir les données contenues dans la queue de réception. Bien que cette méthode est simple, c'est la moins efficace puisqu'il s'il n'y a pas de donnée dans la queue de réception lorsqu'un lecteur effectue une lecture, il y a utilisation inutile du processeur.

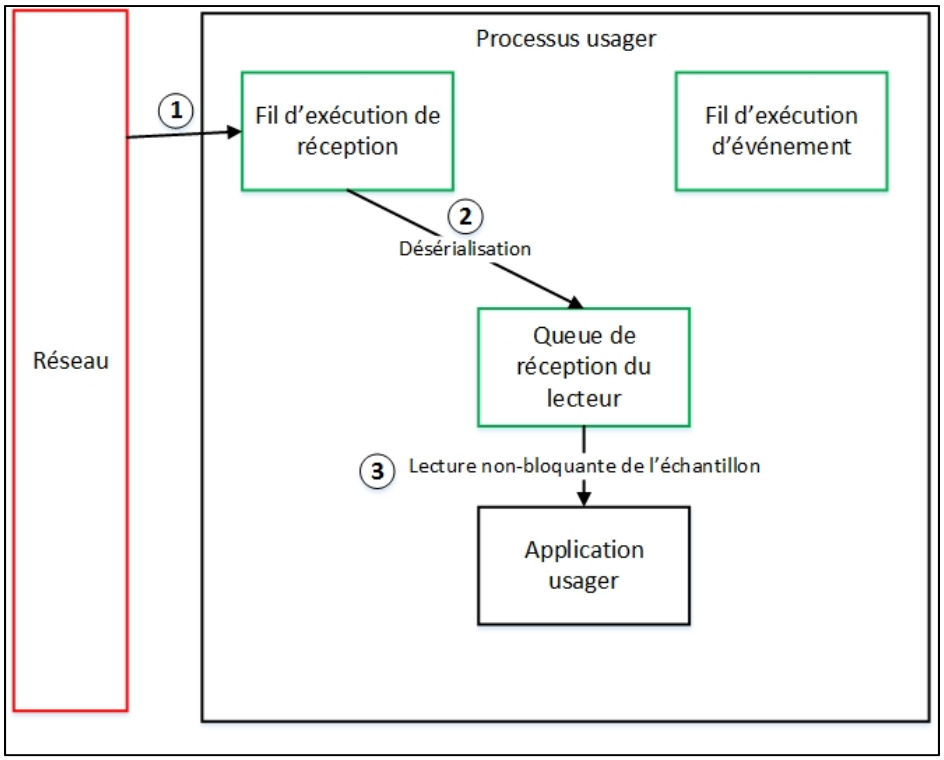


Figure 3.2 Réception par sondage avec Connext Micro

Réception asynchrone

La Figure 3.3 illustre la méthode de réception asynchrone. Tout comme dans la réception par sondage, cette figure illustre que la réception de la donnée arrive tout d'abord dans le fil d'exécution de réception. Par la suite, ce dernier va désérialiser les paquets reçus du réseau et va les enregistrer dans la queue de réception des lecteurs intéressés. Cependant, afin d'effectuer une réception asynchrone, un récepteur (*listener*) doit d'abord avoir été créé et attaché à un lecteur. De cette façon, le fil d'exécution d'événement va ainsi aviser automatiquement le lecteur lorsqu'un nouvel échantillon est arrivé dans la queue de réception ce qui va déclencher ainsi une interruption dans le code de l'application tel qu'illustré à l'étape 3 de la Figure 3.3. Cette interruption va ainsi permettre à l'application de lire le nouvel échantillon dès son arrivée dans la queue de réception ce qui va permettre d'obtenir ainsi la donnée plus rapidement qu'avec la réception par sondage.

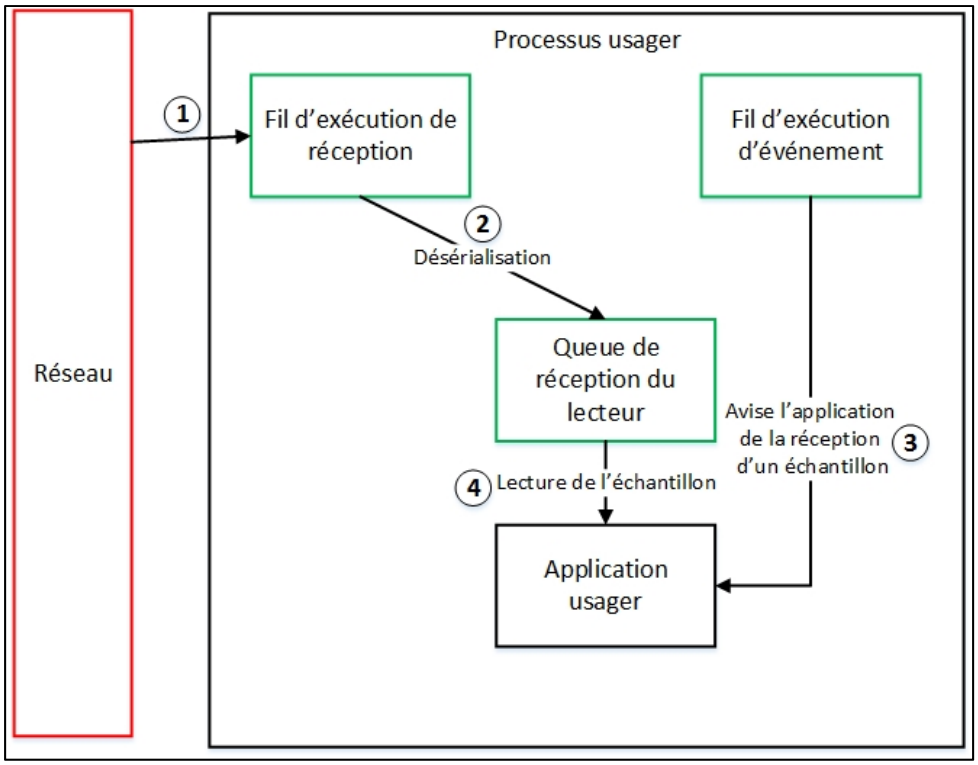


Figure 3.3 Réception asynchrone avec Connext Micro

Réception bloquante

La Figure 3.4 illustre la méthode de réception bloquante. Cette méthode est similaire à la méthode de réception illustrée à la Figure 3.2. La seule différence entre ces deux méthodes est que la méthode de réception bloquante effectue une lecture bloquante de la donnée telle qu'illustrée à l'étape 3 de la Figure 3.4. Cela signifie que si la queue de réception est vide lors de l'opération de lecture, alors l'application va bloquer tant qu'il n'y a pas réception de nouvelles données. Cette méthode ne sera pas traitée davantage pour le reste de ce document puisque les lectures bloquantes ne sont généralement pas désirables pour les systèmes temps réel.

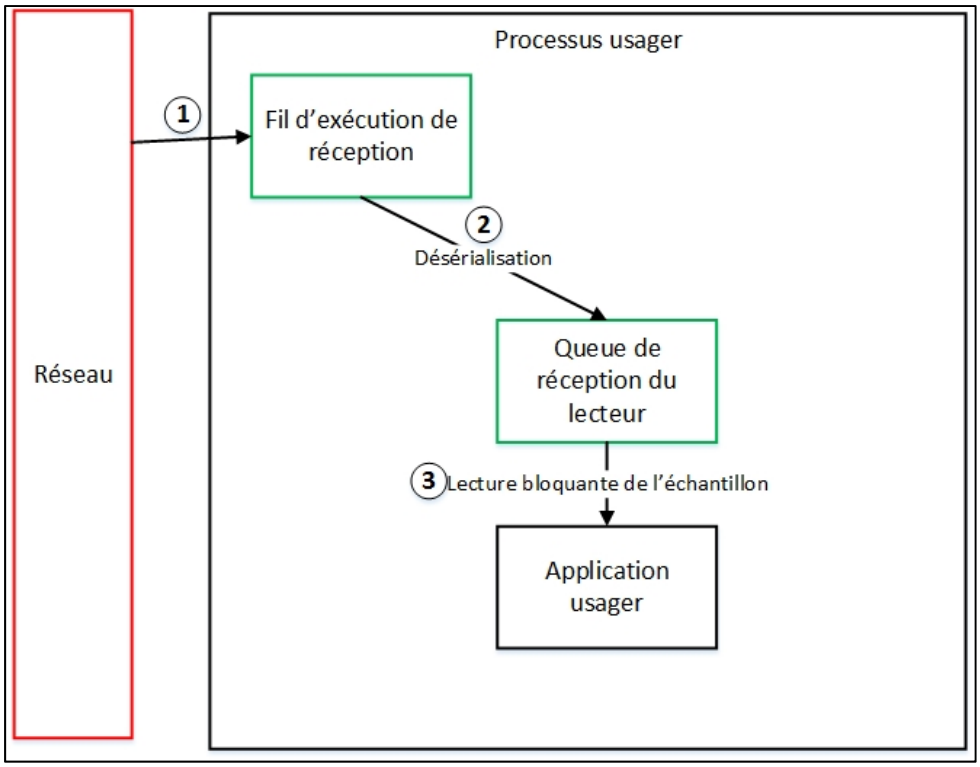


Figure 3.4 Réception bloquante avec Connext Micro

3.2 Métriques de performances utilisées

Afin de bien pouvoir évaluer les performances des communications de l'intergiciel DDS, différentes métriques doivent être mesurées. Les principales métriques qui sont couramment utilisées dans la littérature sont décrites dans cette section. Ces métriques sont notamment utilisées dans l'article de (Bellavista et al., 2013) qui porte sur la comparaison des performances de deux intergiciels DDS.

Latence

On distingue deux types de latence soit la latence de bout en bout et la latence d'aller-retour. La latence de bout en bout est les temps que prend un paquet de données pour se rendre de la source jusqu'à la destination. Cette mesure est exprimée en secondes. La latence d'aller-retour est le temps que prend un paquet de données pour se rendre de la source jusqu'à la destination et de revenir à la source. Cette mesure est exprimée en secondes.

Débit des données

Cette métrique représente la quantité de données qui est transférée à chaque seconde. Cette mesure est exprimée en bits/seconde. L'équation suivante représente le calcul du débit des données:

$$Débit = \frac{Nombre\ de\ bits}{Seconde} \tag{3.1}$$

Bande passante utilisée

Cette métrique représente le pourcentage de la bande passante du canal de communication qui est utilisée par la communication en cours. Cette mesure est exprimée en pourcentage. L'équation suivante représente le calcul de la bande passante utilisée:

$$Bande\ passante\ utilisée = \frac{Débit}{Bande\ passante\ du\ canal} \tag{3.2}$$

Taux d'erreur des messages

Cette métrique représente le pourcentage des messages envoyés qui ont été erronés. Cette mesure est exprimée en pourcentage. L'équation suivante représente le calcul du taux d'erreur des messages:

$$Taux\ d'erreur = \frac{Nombre\ de\ messages\ erronés}{Nombre\ de\ messages\ total} \tag{3.3}$$

Gigue

Cette métrique représente la variabilité de la latence d'une connexion. De ce fait, une connexion avec une latence constante aurait une gigue nulle. Cette mesure est exprimée par l'écart type de la latence qui est calculée par l'équation (3.4).

$$Gigue = \sqrt{\frac{\sum_{i=0}^{n}(Latence_i - Latence\ Moyenne)^2}{n-1}} \tag{3.4}$$

tel que *n* est le nombre d'échantillons de latence utilisés pour calculer la gigue.

3.3 Environnement d'expérimentation

Cette section présente l'environnement d'expérimentation qui est utilisé pour le reste de ce chapitre. De plus, les outils utilisés pour mesurer les métriques de performances présentées précédemment seront également présentés.

Idéalement, afin d'obtenir des résultats qui sont significatifs pour démontrer la viabilité d'intergiciel DDS dans le domaine avionique, il faudrait effectuer les tests sur des plateformes avioniques qui utilisent un RTOS avionique. Cependant, nous n'avons pas accès à deux plateformes avec un RTOS avionique, donc les tests seront effectués dans un environnement d'ordinateurs personnels (PC) avec le système d'exploitation Windows 7. Étant donné que Windows 7 n'est pas un RTOS, ce dernier va donner des valeurs de performances extrémales. De ce fait, ces valeurs extrémales seront ignorées puisque ces dernières ne seraient pas présentes dans un environnement avionique, donc seulement les performances moyennes seront retenues. De plus, on a émis l'hypothèse que si les tests étaient effectués dans un environnement avionique, on observerait des meilleures performances que dans l'environnement Windows 7. Cette hypothèse se base sur l'article de (Schmidt, Deshpande et O'Ryan, 2002) dans lequel les auteurs font des tests de latence en utilisant un intergiciel de communication RT-CORBA, et ce, en utilisant différents systèmes d'exploitation. Les résultats de cet article montrent que la latence moyenne et la gigue obtenue avec un système d'exploitation temps réel tel que QNX et VxWorks sont plus petites que ce qui est obtenu avec un système d'exploitation Windows tel qu'illustré au Tableau 1.1.

Les différents tests sont donc effectués sur des PC dont les spécifications sont décrites dans le Tableau 3.2. Chaque fois que ce mémoire se réfère à un PC, il s'agit des trois PC présentés

dans ce tableau. De plus, tous les tests seront effectués en utilisant le protocole de transport UDPv4. Le choix de ce protocole est dû au fait que l'intergiciel DDS Connext Micro ne supporte que le protocole de transport UDPv4.

Le Tableau 3.3 présente les logiciels qui ont été utilisés pour effectuer les différentes expérimentations qui sont décrites dans les prochaines sections. Le logiciel Microsoft Visual Studio a été utilisé pour faire la compilation du code. De plus, l'intergiciel DDS Connext Micro a été utilisé. Les logiciels WireShark et RTI Analyser ont également été utilisés afin de détecter les anomalies dans les tests effectués.

Tableau 3.2 Spécifications des PC utilisés

	PC #1	**PC #2**	**PC #3**
Processeur	Intel Xeon W3670 3.20 GHz	Intel Core i3-2350M 2.30 GHz	Inter Core i7-450U 2.40 GHz
Mémoire vive	16 Go	6 Go	6 Go
Système d'exploitation	Windows 7 64 bits	Windows 7 64 bits	Windows 7 64 bits
Carte réseau	Broadband NetXtreme Gigabit Ethernet	Realtek PCie GBE Family Controller	Realtek PCIe GBE Family Controller

Tableau 3.3 Logiciels utilisés

Nom du logiciel	**Version**	**Utilité**
Microsoft Visual Studio Professional 2012	11.0.61033.00 Update 4	Compilation du code en C
RTI Connext Micro	2.2.3	Intergiciel de communication DDS
Wireshark	1.2.3	Analyseur de protocoles réseau
RTI Analyser	5.0.0.0	Déverminer les entités DDS

Une horloge haute précision a également été utilisée pour calculer la latence d'aller-retour des communications ainsi que pour calculer la quantité d'échantillons reçus par seconde. Pour avoir accès à cette horloge haute précision, la fonction *QueryPerformanceCounter* de la librairie kernel32.lib a été utilisée.

3.4 Mesure des performances de latence

Dans cette section, plusieurs expérimentations seront effectuées afin de mesurer et caractériser les performances des communications DDS. Tout d'abord, une description des requis de latences des communications en avionique est effectuée. Ensuite, des mesures de latence sans intergiciel DDS sont effectuées. Ces mesures serviront de base de référence afin d'évaluer l'impact de l'utilisation de l'intergiciel DDS sur la latence des communications. Par la suite, des mesures de latence de l'intergiciel DDS avec une réception par sondage et une réception asynchrone sont effectuées. Finalement, une analyse des résultats obtenus ainsi qu'une comparaison entre les résultats des différentes méthodes de communication sont effectuées.

3.4.1 Requis de latence des communications en avionique

Les requis de performances des communications en avionique sont difficiles à spécifier, car ils dépendent des applications impliquées. Le niveau de criticité des applications, ainsi que la fonction des applications vont avoir un grand impact sur les requis des communications. Dans le chapitre 4 de ce document, un AFCS contenant des boucles de commande de vol d'un avion de ligne sera conçu. De ce fait, la littérature a été explorée afin de trouver les requis de latence des boucles de commande de vol. Dans (Schuster et Verma, 2008), il y est stipulé qu'une latence de l'ordre de 10 ms est suffisante pour les contrôles d'un avion de ligne alors que les contrôles d'un avion de chasse nécessitent une latence de l'ordre des microsecondes.

Il est donc important que l'intergiciel DDS puisse envoyer des données avec une latence inférieure à 10 ms afin de satisfaire les requis en latence des boucles de commande de vol d'un avion de ligne. Normalement une analyse de latence maximale (*worst case*) serait effectuée afin de s'assurer que le système obtient toujours une latence inférieure à 10 ms. Cependant, tel que mentionné dans la section 3.3, étant donné que les tests ne sont pas effectués dans un environnement avionique, seulement la latence moyenne sera considérée.

3.4.2 Mesure de la latence sans intergiciel DDS

En premier lieu, un test avec de simples *sockets* UDP sans intergiciel DDS est effectué afin d'avoir une base de comparaison pour pouvoir évaluer l'impact de l'utilisation d'un intergiciel DDS sur les performances de latence de la communication. De ce fait, la librairie <winsock2.h> fournie par Windows a été utilisée pour effectuer une communication entre deux PC à travers un seul câble Ethernet RJ45 avec un communication de 100 Mbits/seconde.

Le test effectué est illustré, sous forme de diagramme de séquence, à la Figure 3.5. Le PC #1 démarre tout d'abord l'horloge afin de calculer la latence d'aller-retour et envoie ensuite la donnée au PC #2. Ensuite, il fait une lecture bloquante sur le *socket* jusqu'à ce que la donnée envoyée soit reçue. Lorsque cette dernière est reçue, le PC #1 arrête l'horloge et effectue le calcul de la latence d'aller-retour. De son côté, le PC #2 ne fait qu'une lecture bloquante sur un *socket* UDP et renvoie les données reçues au PC #1.

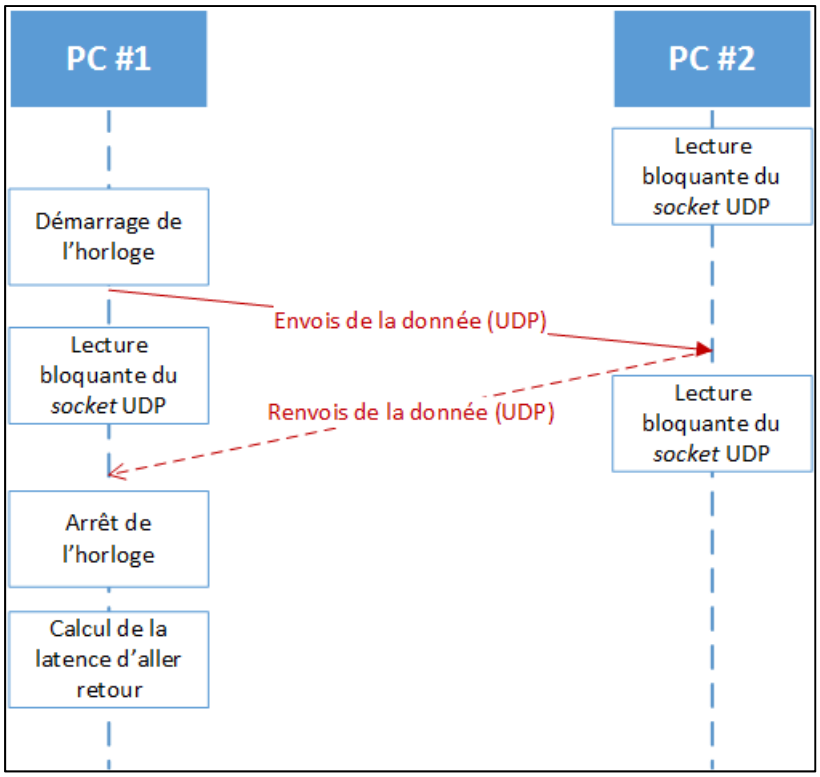


Figure 3.5 Diagramme de séquence du test de latence sans intergiciel DDS

Afin d'obtenir la latence d'aller-retour moyenne, 100 000 échantillons ont été envoyés. Cette quantité d'échantillons a été choisie arbitrairement afin d'avoir assez d'échantillons pour avoir une moyenne significative ainsi que pour observer les valeurs extrémales possibles. La taille des paquets utilisée est de 128 octets puisque cette taille de paquet a été utilisée dans l'article de (Bellavista et al., 2013) qui porte sur l'évaluation des performances des intergiciels DDS. Pour le reste de ce chapitre, il sera estimé que la latence de bout en bout est environ égale à la latence aller-retour divisée par deux afin de pouvoir comparer les résultats obtenus avec les requis de latence tirés de la littérature dans la section 3.4.1.

Les résultats de latence bout en bout obtenus sont illustrés à la Figure 3.6. Afin de bien observer les mesures de latence entre 0 et 10 ms, les valeurs extrémales n'ont pas été tracées dans cette figure. Les résultats de la Figure 3.6 illustrent une latence moyenne est de 0.57 ms avec une faible gigue de 0.29 ms. Cependant, il est important de noter que 0.008% des échantillons ont donné des valeurs extrémales allant jusqu'à 57.17 ms. Comme il a été

mentionné précédemment, ces valeurs extrémales sont attribuables au fait que le système d'exploitation Windows 7 n'est pas approprié pour les applications temps réel et ne permet pas d'assurer tout le temps les ressources nécessaires par l'application de test afin d'obtenir de faibles latences. De ce fait, seulement la latence moyenne sera considérée.

De plus, aucun paquet n'a été perdu dans le test effectué. Les résultats de paquets perdus ne seront pas abordés pour les prochaines expériences puisque le pourcentage de paquets perdus observé est toujours de zéro puisque l'environnement de test n'implique que deux PC connectés ensemble. Afin d'observer des paquets perdus, il faudrait ajouter plusieurs PC sur un interrupteur réseau et augmenter la quantité de trafic sur le réseau.

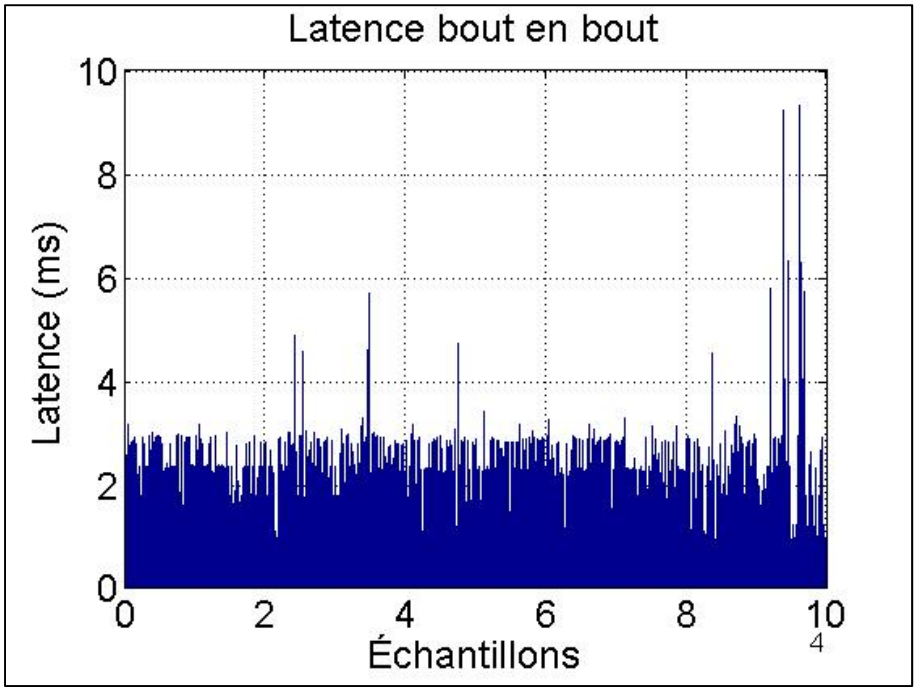


Figure 3.6 Test de latence sans intergiciel DDS

Ces observations sont pertinentes puisque ce sont à partir de ces dernières que les performances de l'intergiciel Connext Micro seront comparées. En effet, étant donné que l'intergiciel DDS va utiliser les *sockets* UDP fournis par Windows 7, les résultats obtenus

avec l'intergiciel DDS ne pourront pas être meilleurs que les résultats qui ont été présentés dans cette sous-section.

3.4.3 Mesure de la latence avec intergiciel DDS (Réception par sondage)

Afin d'évaluer l'impact de l'utilisation d'un intergiciel DDS sur la latence des communications, un test a été effectué avec l'intergiciel Connext Micro avec réception par sondage. Le test consiste à effectuer une communication de 100 Mbits/seconde entre deux PC connectés ensemble par un câble Ethernet RJ45 tout comme dans le test de la section précédente, mais en utilisant l'intergiciel DDS pour effectuer la communication. Un lecteur et un écrivain de sujets différents ont été créés sur les deux PC pour ainsi former deux liens de communication distincts. Les deux sujets créés permettent d'envoyer une structure de taille 128 octet et 100 000 échantillons sont envoyés, tout comme dans le test précédent. Le test effectué est illustré, sous la forme d'un diagramme de séquence, à la Figure 3.7.

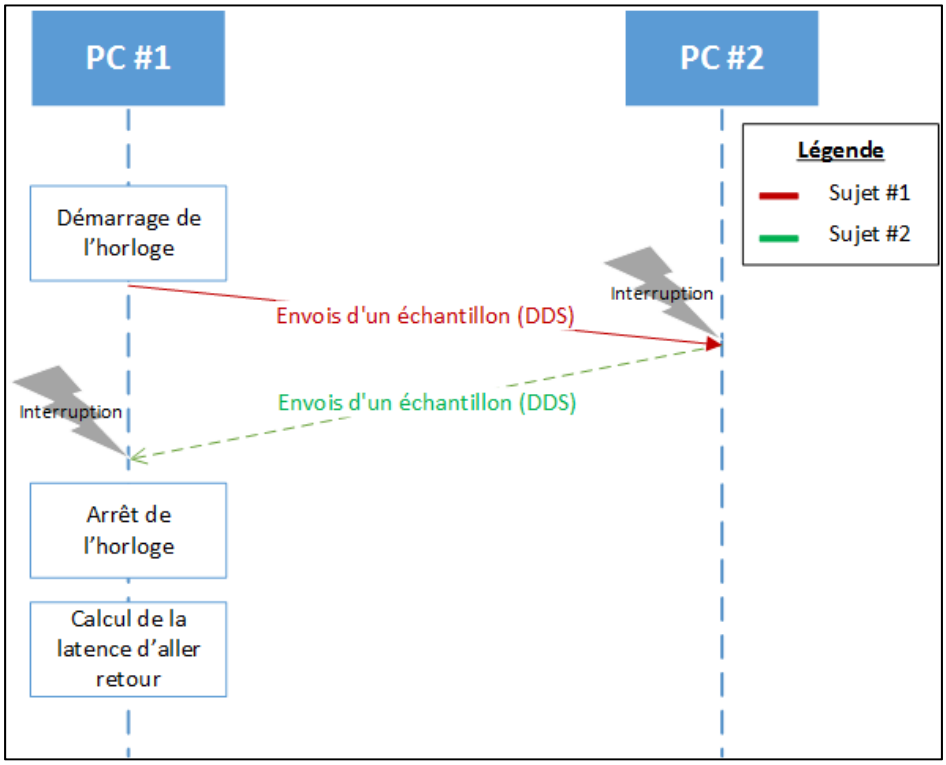


Figure 3.7 Diagramme de séquence du test de latence DDS par sondage

Cette figure illustre que le PC #1 démarre tout d'abord l'horloge qui permet de calculer la latence d'aller-retour et envoie ensuite un échantillon au PC #2 avec son écrivain de sujet #1. Ensuite, il fait des lectures non-bloquantes en continue sur un lecteur du sujet #2 jusqu'à ce que l'échantillon renvoyé soit reçu. Lorsque ce dernier est reçu, le PC #1 arrête l'horloge et effectue le calcul de la latence d'aller-retour.

De son côté, le PC #2 ne fait que faire des lectures non-bloquantes en continue sur son lecteur de sujet #1. Lorsqu'il reçoit un échantillon, il envoie un échantillon au PC #1 avec son écrivain de sujet #2.

Les résultats obtenus sont illustrés à la Figure 3.8. Encore une fois, les valeurs extrémales n'ont pas été tracées sur cette figure afin de bien présenter les mesures de latence entre 0 et 10 ms. Cette figure montre une latence moyenne est de 0.82 ms une gigue de 0.33 ms. Il est important de noter que 0.014% des échantillons ont donné des valeurs extrémales allant jusqu'à 30.43 ms. Ces résultats seront analysés plus tard dans la section 3.4.5.

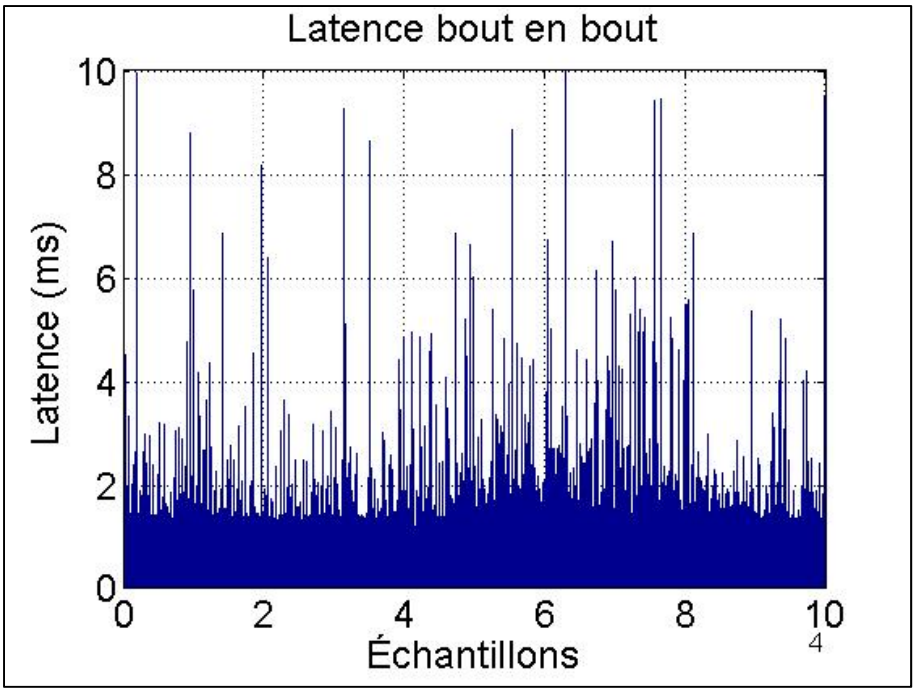


Figure 3.8 Test de latence avec intergiciel DDS avec réception par sondage

3.4.4 Mesure de la latence avec intergiciel DDS (Réception asynchrone)

Dans la section précédente, les performances de l'intergiciel DDS avec une lecture par sondage ont été présentées. Or il est également possible de faire la lecture des données à l'aide d'une routine de rappel (*callback*) asynchrone. De ce fait, les récepteurs ont été utilisés afin de débuter une routine de lecture des échantillons à chaque réception de ces derniers. Un lecteur et un écrivain de sujets différents ont été créés sur les deux PC pour ainsi former deux liens de communication distincts. La mesure de la latence moyenne est encore effectuée avec 100 000 échantillons d'une taille de 128 octets. Le but de ce test est d'évaluer l'impact de l'utilisation d'un intergiciel DDS avec une réception asynchrone sur la latence de la communication entre les deux PC. Le diagramme de séquence du test de latence avec l'intergiciel DDS avec réception asynchrone est présenté à la Figure 3.9.

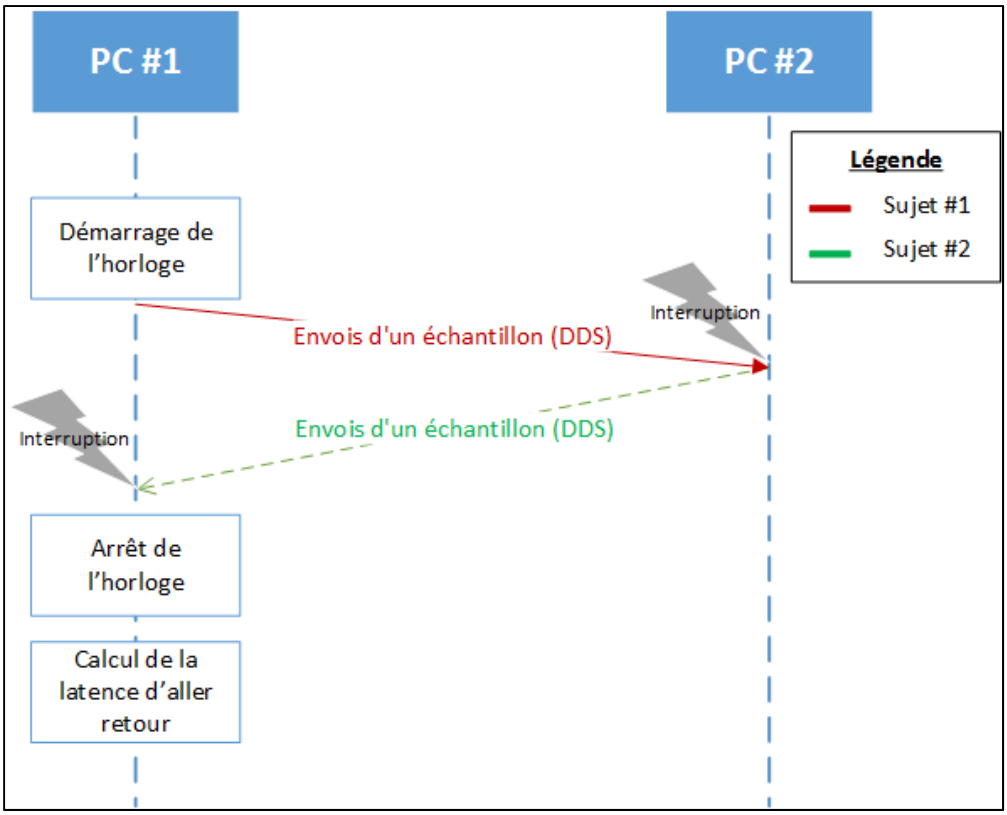


Figure 3.9 Diagramme de séquence du test de latence DDS par réception asynchrone

Pour ce test, le PC #1 démarre tout d'abord l'horloge qui permet de calculer la latence d'aller-retour et envoie ensuite un échantillon au PC #2 avec son lecteur. Ensuite, il ne fait rien jusqu'au moment où l'intergiciel DDS déclenche une interruption pour aviser l'application qu'un échantillon est reçu. Lorsque ce dernier est reçu, le PC #1 arrête l'horloge et effectue le calcul de la latence d'aller-retour.

De son côté, le PC #2 est muni d'un lecteur de sujet #1 et d'un écrivain de sujet #2. Le PC #2 ne fait rien jusqu'au moment où l'intergiciel DDS déclenche une interruption pour aviser l'application qu'un échantillon est reçu. Le PC #2 lit ainsi l'échantillon reçu avec son lecteur et envoie un échantillon au PC #1 avec son écrivain.

Les résultats de latence obtenus sont illustrés à la Figure 3.10. Encore une fois, les valeurs extrémales n'ont pas été tracées sur cette figure afin de bien présenter les mesures de latence

entre 0 et 10 ms. Cette figure montre une latence moyenne de 0.62 ms et une gigue de 0.38 ms. De plus, 0.002% des échantillons ont donné des valeurs extrémales allant jusqu'à 28.31 ms. Ces résultats seront analysés dans la section 3.4.5.

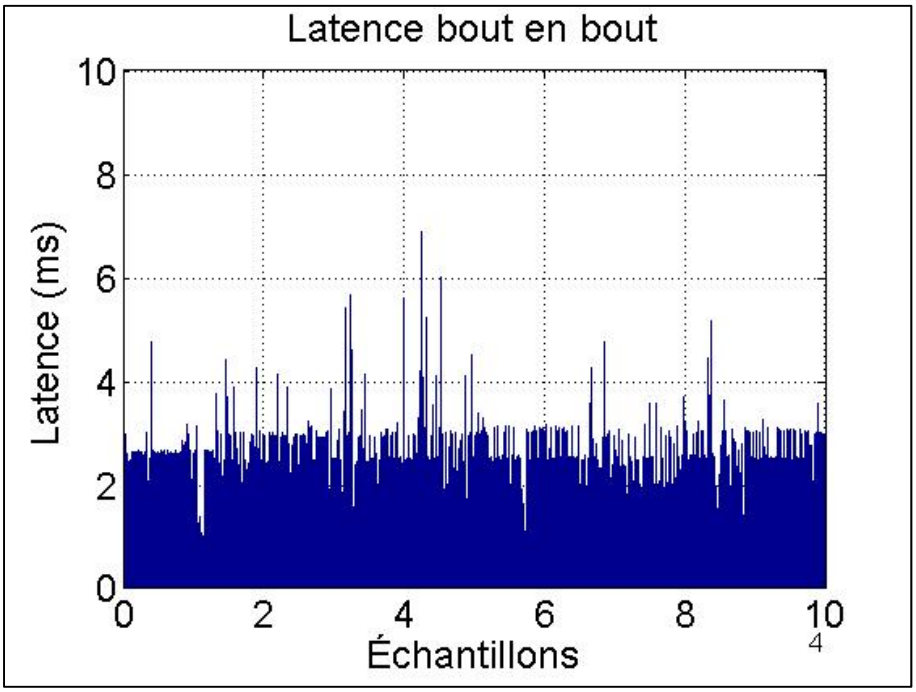


Figure 3.10 Test de latence avec intergiciel DDS par réception asynchrone

3.4.5 Comparaison des performances de latence

En prenant les différents résultats de latence obtenus dans les sections précédentes pour une taille de paquet de 128 octets, le Tableau 3.4 est ainsi obtenu. Ce tableau montre que le scénario sans intergiciel DDS offre une meilleure latence moyenne soit de 0.57 ms alors que les scénarios avec intergiciel DDS ont une latence moyenne de 0.82 ms et de 0.62 ms pour une réception par sondage et pour une réception asynchrone respectivement. La raison pour laquelle le scénario DDS avec une réception par sondage à une latence moyenne plus élevée est qu'il est possible que l'on doive effectuer une deuxième lecture si jamais un nouvel échantillon est reçu presque en même temps que la requête de lecture est effectuée. Il est possible d'en conclure que l'ajout de l'intergiciel DDS cause une faible augmentation la latence moyenne de la communication, mais que cette dernière reste bien en dessous du 10

ms qui est requis pour le contrôle d'un avion de ligne. Selon l'hypothèse posée précédemment, il serait possible d'observer une plus petite augmentation de la latence moyenne si un intergiciel DDS était utilisé dans un environnement avionique.

Tableau 3.4 Comparaison des résultats de latence des différents scénarios

Scénario	**Latence moyenne (ms)**	**Gigue (ms)**	**Latence maximale (ms)**	**Pourcentage des échantillons >10ms**
Sans DDS	0.57	0.29	57.17	0.008%
Avec DDS (Réception par sondage)	0.82	0.33	30.43	0.014%
Avec DDS (Réception asynchrone)	0.62	0.38	28.31	0.002%

Le tableau montre également que la gigue est faible dans les trois scénarios étudiés. Il est donc possible d'en conclure que la distribution de la latence est faible et que la plupart des échantillons ont une latence près de la moyenne. De plus, puisque la gigue est faible, il est possible d'en tirer que la latence moyenne est un bon indicateur des performances du lien de communication.

Cependant, le Tableau 3.4 montre que la latence maximale est très élevée par rapport à la moyenne soit de 57.17 ms sans l'intergiciel DDS, de 30.43 ms pour l'intergiciel DDS avec réception par sondage et de 28.31 ms pour l'intergiciel DDS avec réception asynchrone. La latence maximale est importante dans le domaine avionique puisque les performances d'un système sont souvent évaluées en tenant compte seulement du pire cas (*worst case*). Cependant, comme il a été mentionné précédemment dans la section 3.3, ces valeurs extrémales sont dues à l'utilisation du système d'exploitation Windows 7 et ces valeurs ne seraient pas présentes si un environnement avionique avait été utilisé. De ce fait, seulement la latence moyenne sera considérée comme indicateur de performances.

3.5 Mesure du débit de données

Lors de l'évaluation d'un intergiciel de communication, il est également important de regarder quel est le débit maximal de données que ce dernier peut délivrer. Or le débit maximal qu'il peut délivrer est généralement fonction de la taille des paquets envoyés. De ce fait, des tests ont été faits en changeant la taille des paquets et en comptant le nombre maximal d'échantillons qui peuvent être reçus par un seul lecteur. Pour ce faire, le diagramme de test de la Figure 3.11 a été effectué.

Cette figure présente que le PC #1 envoie 1000 échantillons au PC #2. La valeur de 1000 échantillons a été choisie arbitrairement afin d'avoir assez d'échantillons pour avoir une moyenne significative. De son côté, le PC #2 démarre une horloge, reçoit les 1000 échantillons par réception asynchrone et arrête ensuite l'horloge. Il fait ensuite le calcul du nombre d'échantillons reçus par seconde en divisant 1000 par le nombre de temps sur l'horloge.

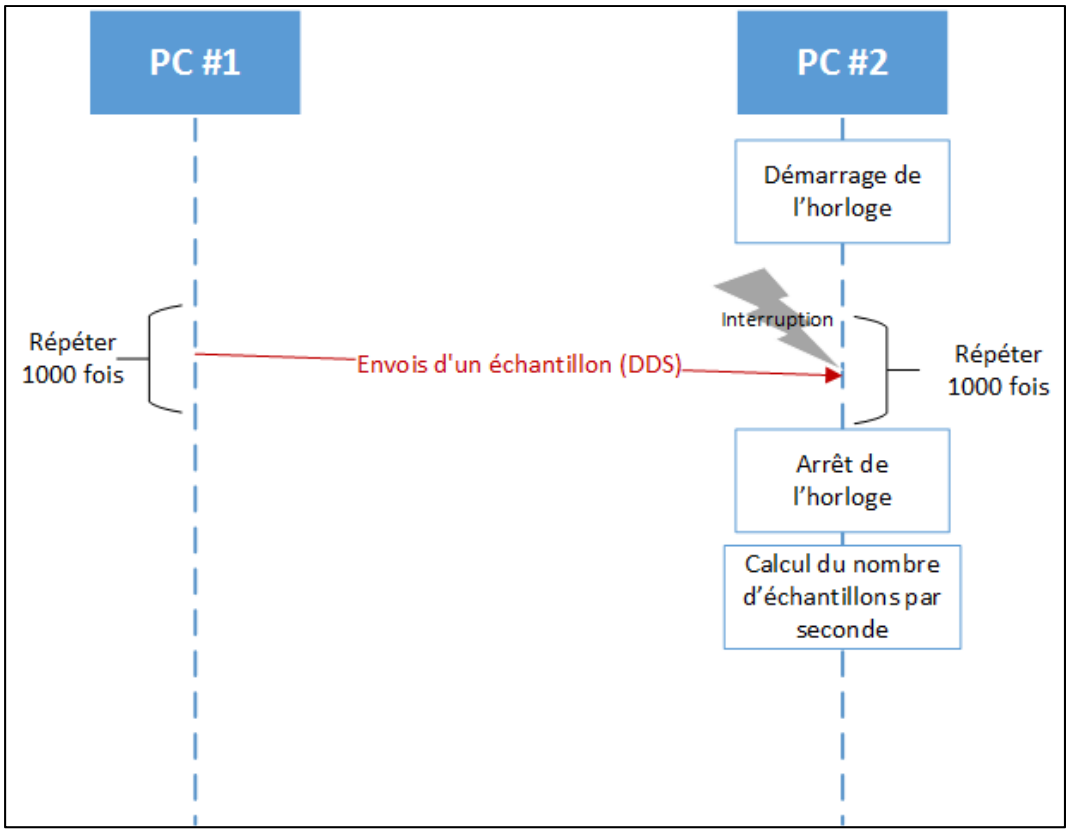


Figure 3.11 Diagramme de séquence des tests de débit de données

Ce test a été répété à plusieurs reprises en changeant la taille des paquets entre 128 et 4096 octets afin d'observer l'effet de cette dernière sur le nombre d'échantillons reçus par seconde. Il est à noter ici que la taille des paquets n'inclue que la charge utile du paquet. Une connexion de 100 Mbit/s a été utilisée entre les deux PC et ces derniers sont connectés par un câble Ethernet RJ45.

Le nombre d'échantillons moyens par secondes obtenues pour différentes tailles de paquets sont présentés au Tableau 3.5. Le débit de données a ensuite été calculé à partir de l'équation (3.5).

$$Débit\ de\ données = T * N * 8\ bit \tag{3.5}$$

Tel que T est la taille du paquet en octets et N est le nombre d'échantillons moyens par seconde.

Tableau 3.5 Résultats du débit de données en fonction de la taille des paquets

Taille du paquet (octets)	Nombre d'échantillons moyen par seconde	Débit de données	Bande passante utilisée
128	5534	5.67 Mbit/s	5.67%
256	4771	9.77 Mbit/s	9.77%
512	4819	19.74 Mbit/s	19.74%
1024	3506	28.72 Mbit/s	28.72%
2048	3093	50.68 Mbit/s	50.68%
4096	2587	84.77 Mbit/s	84.77%

À partir du Tableau 3.5, la relation entre le nombre d'échantillons reçus par secondes et la taille de ces derniers à été tracé à la Figure 3.12.

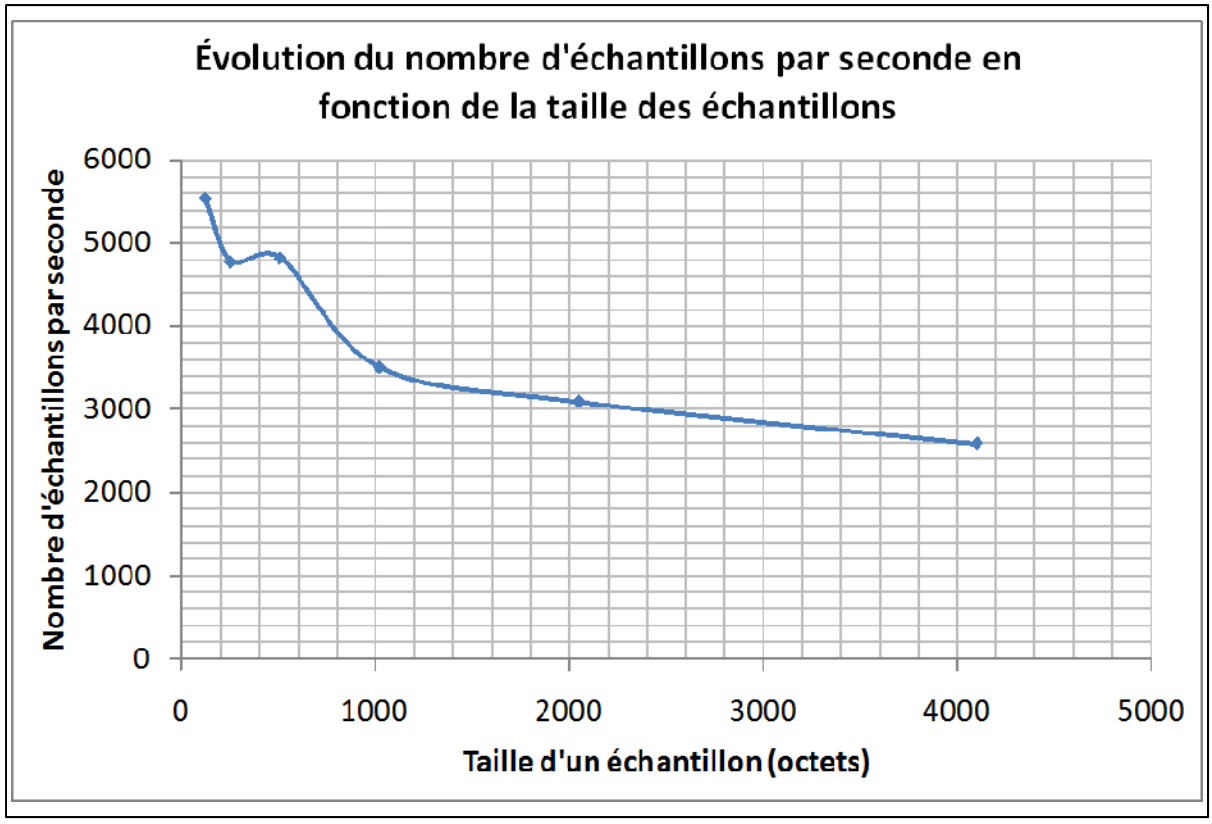


Figure 3.12 Nombre d'échantillons par seconde en fonction de la taille de ces derniers

Cette figure illustre que le nombre d'échantillons reçus par seconde diminue lorsque la taille de ces derniers augmente. La courbe obtenue a la même tendance que les résultats obtenus dans l'article de (Bellavista et al., 2013) qui porte sur la mesure du nombre d'échantillons par seconde pour un intergiciel DDS. Cependant, la courbe obtenue à la Figure 3.12 ne peut pas être directement comparée puisque dans cet article les auteurs utilisent des implémentations d'intergiciel DDS différentes ainsi qu'un système d'exploitation Linux.

La Figure 3.13 a ensuite été tracée, à partir du Tableau 3.5, afin d'illustrer le débit de données en fonction de la taille des paquets.

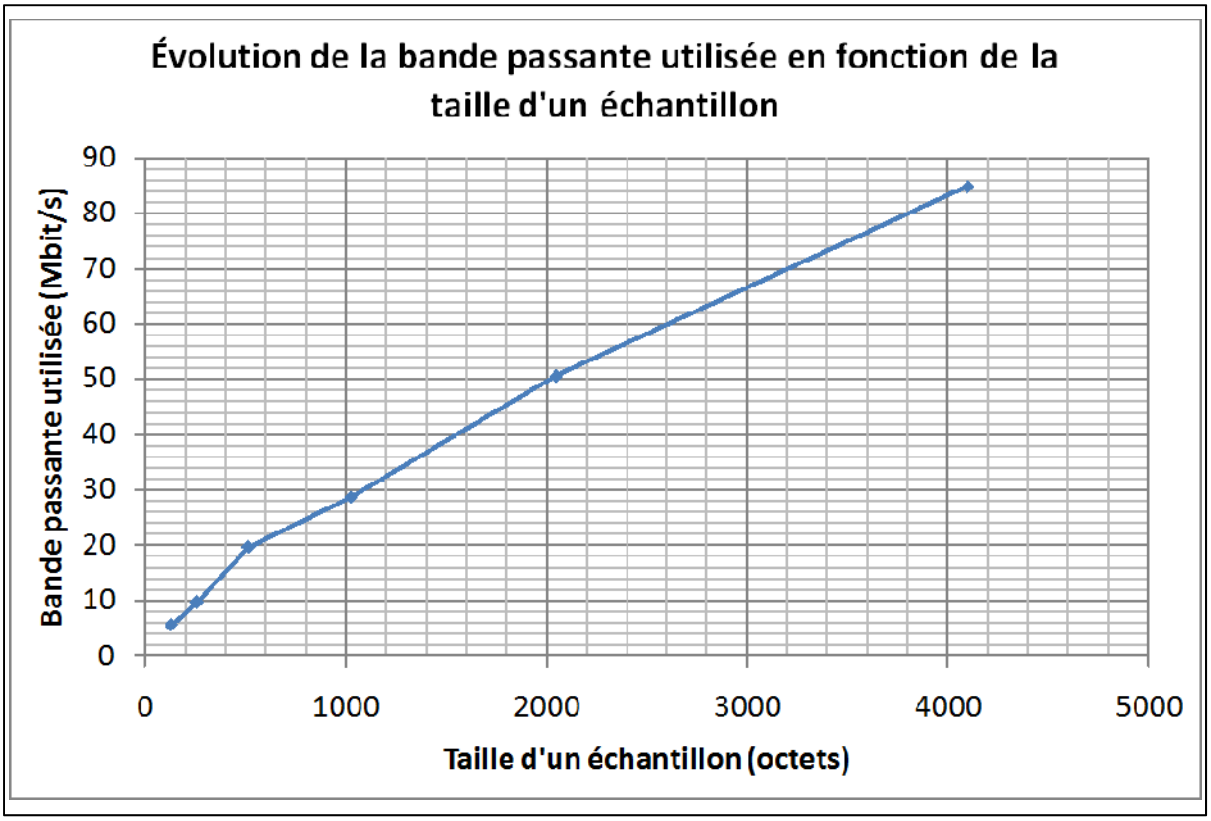


Figure 3.13 Débit de données en fonction de la taille des échantillons

Encore une fois, la courbe obtenue a la même tendance que les résultats obtenus dans l'article de (Bellavista et al., 2013). La Figure 3.13 montre que le débit de données augmente

lorsque la taille des paquets augmente. En effet elle passe de 5.67 Mbit/s pour des paquets de 128 octets à 84.77 Mbit/s pour des paquets de 4096 octets. Considérant le fait que le lien de communication à une bande passante maximale de 100 Mbit/s, il est ainsi possible d'en conclure que l'intergiciel arrive à utiliser une grande partie (jusqu'à 84.77%) de la bande passante disponible sur le lien de communication. L'utilisation de l'intergiciel DDS n'est donc pas un goulot d'étranglement à l'utilisation de toute la bande passante disponible sur le lien de communication. Il est important de noter que cette utilisation de la bande passante est une utilisation utile de la bande passante mesurée au niveau de l'application. De ce fait, cette dernière ne tient pas compte de l'entête des messages, mais seulement de la charge utile des messages.

3.6 Conclusion

Le but de ce chapitre était de caractériser les performances des communications en utilisant un intergiciel DDS. Il a été vu dans ce chapitre que l'utilisation de l'intergiciel DDS Connext Micro cause une légère augmentation de la latence moyenne de la communication entre deux PC. En effet, la latence moyenne dans le test sans l'utilisation d'un intergiciel DDS est de 0.57 ms alors que les tests avec l'intergiciel DDS donnent une latence moyenne de 0.82 ms et de 0.62 ms pour la réception par sondage et pour la réception asynchrone respectivement. Cependant, cette augmentation de la latence moyenne donne tout de même une latence moyenne bien inférieure au 10 ms requis pour les contrôles des avions de ligne. De plus, on a posé l'hypothèse que les performances qui seraient observées dans un environnement avionique seraient meilleures à celles observées dans l'environnement Windows 7.

Cependant, la latence maximale observée est élevée dans tous les cas de tests soit avec et sans intergiciel DDS. Ces latences élevées sont attribuables à l'utilisation du système d'exploitation Windows 7 qui n'est pas approprié pour les applications temps réel. Dans les faits, Windows 7 ne serait jamais utilisé dans le cadre de systèmes de sureté critique tels que les systèmes avioniques. La latence moyenne a donc été retenue comme indicateur de

performance puisque les valeurs extrémales ne seraient pas présentes si un RTOS avionique avait été utilisé.

De plus, il a été vu que l'intergiciel DDS évalué permet d'utiliser presque toute la bande passante disponible sur une connexion de 100 Mbit/s soit 84.77% de la bande passante pour une taille de paquet de 4096. Il est donc possible d'en conclure que l'utilisation de l'intergiciel DDS n'est pas un goulot d'étranglement à l'utilisation de la bande passante du lien de communication.

Bien que les performances mesurées dans ce chapitre soient pertinentes, elles ne sont pas suffisantes pour valider la viabilité de l'utilisation d'un intergiciel DDS dans le domaine avionique. De ce fait, le prochain chapitre portera sur une étude de cas dans laquelle un intergiciel DDS sera utilisé afin d'établir une communication entre un AFCS pour un Boeing 747-400 et un simulateur de vol. Cette étude de cas servira à évaluer l'impact de l'utilisation d'un intergiciel DDS sur les performances d'une application avionique.

CHAPITRE 4

ÉTUDE DE CAS D'UN AFCS POUR UN BOEING 747-400

Les performances observées dans le chapitre précédent ne sont pas suffisantes pour valider la viabilité d'un intergiciel DDS dans le domaine avionique. De ce fait, une étude de cas avec une application avionique est effectuée dans ce chapitre. Pour ce faire, une application d'AFCS pour un Boeing 747-400 a été conçue. Tel que mentionné dans la section 1.4.1, un AFCS est un système qui permet de contrôler les axes de rotation, la vitesse ainsi que l'altitude d'un aéronef. L'AFCS servira à contrôler un Boeing 747-400 dont l'aérodynamique est simulée à l'aide du simulateur de vol X-Plane. Le but de ce chapitre est de mesurer l'impact de l'utilisation d'un intergiciel DDS sur les performances de l'AFCS conçu.

Le chapitre porte tout d'abord sur le simulateur de vol X-Plane et les particularités du Boeing 747-400. Par la suite, la conception de l'AFCS est effectuée. Ensuite, les performances de l'AFCS sont caractérisées à travers une multitude de tests de temps de stabilisation et d'erreur en régime permanent. Les tests incluent les performances de l'AFCS sans l'intergiciel DDS et avec l'intergiciel DDS. De plus, les scénarios de tests sont effectués sur des PC avec le système d'exploitation Windows 7 ainsi que sur une plateforme FreeScale avec le système d'exploitation temps réel VxWorks 6.9. Les différents résultats obtenus seront comparés entre eux afin de mesurer l'impact de l'utilisation de l'intergiciel sur les performances de l'AFCS.

4.1 Simulateur de vol X-Plane

Cette section traite du logiciel X-Plane, le simulateur de vol utilisé dans le cadre de cette recherche afin de simuler un Boeing 747-400. X-Plane est un simulateur de vol qui permet de simuler les capteurs, les actionneurs ainsi que l'aérodynamique d'un aéronef (Barros dos Santos et de Oliveira, 2011). Ce simulateur a été choisi parmi plusieurs logiciels similaires tels que FlightSim et HeliSim à cause de sa disponibilité, sa simplicité d'utilisation ainsi que

sa grande quantité de documentation disponible. La version de X-Plane qui a été utilisée est X-Plane 10.25 (build 102503 32-bit).

4.1.1 Mécanismes d'entrées/sorties

Le logiciel X-Plane est muni d'une interface usager d'entrées/sorties qui permet d'envoyer les valeurs des différents capteurs ainsi que de recevoir des commandes pour les actuateurs avec une communication UDP/IP. Il suffit de sélectionner les valeurs que l'on veut envoyer ou recevoir par X-Plane et déterminer le débit auquel les valeurs doivent être envoyées. Les étiquettes qui ont été sélectionnées dans l'interface d'entrées/sorties de X-Plane afin de concevoir l'autopilote sont décrites dans le Tableau 4.1. Ces étiquettes sont importantes, car elles seront utilisées comme variables d'entrées et de sorties des boucles de commande de l'AFCS.

Tableau 4.1 Entrées/Sorties sélectionnées dans X-Plane

Numéro d'étiquette	Valeurs
3	Vitesses
8	Levier de commande (élévateur, aileron, gouvernail)
16	Vitesses angulaires
17	Tangage, Roulis, Lacet
20	Latitude, Longitude, Altitude
25	Commande de l'accélérateur

4.1.2 Caractéristiques du Boeing 747-400

Le Boeing 747-400 illustré à la Figure 4.1 a été choisi, car il s'agit d'un aéronef dont les performances sont bien connues (Barros dos Santos et de Oliveira, 2011) et c'est également un aéronef dont le modèle est inclus avec la version X-Plane que nous avons dans notre laboratoire. Afin de pouvoir envoyer des commandes de vol à un avion Boeing 747-400, il est d'abord important de connaître certaines caractéristiques de cet aéronef. Il est important de savoir que pour contrôler les différentes surfaces de contrôle de l'aéronef dans le logiciel X-Plane, il suffit d'envoyer des commandes au levier de commande. La valeur de la commande que l'on doit envoyer au levier de commande X-Plane doit être comprise entre -1

et +1 ce qui va représenter l'angle que va prendre la surface de contrôle que l'on veut changer (Bittar, de Oliveira et de Figueiredo, 2013). Le Tableau 4.2 présente les variations maximales des surfaces de contrôle pour un Boeing 747-400. Ces valeurs ont été soutirées du logiciel X-Plane par expérimentations. De plus, la valeur qui doit être envoyée à l'accélérateur dans X-Plane doit se situer entre 0 (aucune accélération) et 1 (accélération maximale).

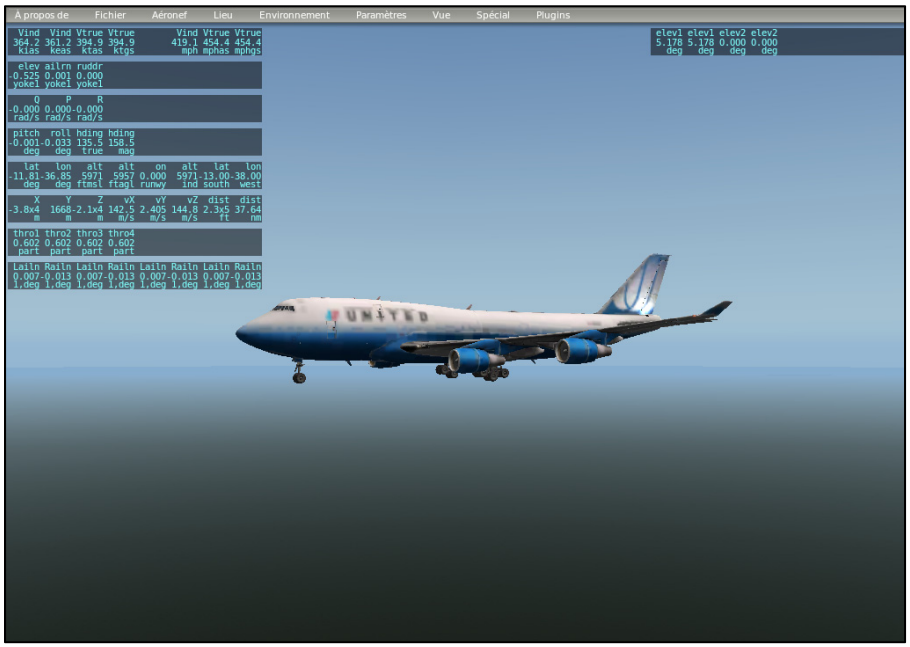


Figure 4.1 Capture d'écran illustrant le Boeing 747-400 dans le logiciel X-Plane

Tableau 4.2 Variations possibles des surfaces de contrôle d'un Boeing 747-400

Surfaces de contrôle	**Valeur minimale (-1 dans X-Plane)**	**Valeur maximale (+1 dans X-Plane)**
Angle des élévateurs	+10 degrés	- 20 degrés
Angle de l'aileron gauche	+10 degrés	-20 degrés
Angle de l'aileron droit	-20 degrés	+10 degrés

Les différentes surfaces de contrôle d'un aéronef sont contrôlées par des servomoteurs. Il est donc nécessaire de connaître les fonctions de transfert des différents servomoteurs afin de pouvoir envoyer correctement des commandes aux différentes surfaces de contrôle du Boeing 747-400. Les fonctions de transfert utilisées dans le cadre de cette recherche sont

présentées dans le Tableau 4.3 et proviennent de l'article de (Barros dos Santos et de Oliveira, 2011).

Tableau 4.3 Fonction de transfert des servomoteurs d'un Boeing 747-400

Servomoteur	Fonction de transfert
Servomoteurs des élévateurs	$\frac{10}{s+10}$
Servomoteurs des ailerons	$\frac{10}{s+10}$
Servomoteur de l'accélérateur	$\frac{20}{s+20}$

4.2 Méthodologie d'expérimentation

Cette section porte sur la méthodologie d'expérimentation composée de sept étapes illustrées à la Figure 4.2. Cette dernière montre que la première étape consiste à concevoir l'AFCS dans Simulink. L'AFCS sera utilisé pour contrôler un Boeing 747-400 dont l'aérodynamique est simulée dans le simulateur de vol X-Plane. Le but de la conception de cet AFCS est d'avoir une application avionique afin de pouvoir évaluer l'impact de l'utilisation d'un intergiciel DDS sur une telle application.

La seconde étape consiste ensuite à générer le code en langage C de l'AFCS pour ainsi l'exécuter en dehors de l'environnement Simulink.

La troisième étape consiste à mesurer les performances de l'AFCS dans un environnement de PC sans utiliser d'intergiciel DDS. Deux scénarios de tests seront effectués à cette étape.

La quatrième étape consiste à ajouter la communication DDS au sein de l'AFCS conçu. Pour ce faire, un *plug-in* de communication DDS a dû être développé afin de permettre au simulateur de vol X-Plane d'effectuer des communications par l'intergiciel DDS.

La cinquième étape consiste à mesurer les performances de l'AFCS toujours dans un environnement de PC, mais en utilisant un intergiciel de communication DDS. Deux scénarios de tests seront effectués à cette étape et les résultats de ces derniers seront comparés aux résultats des deux scénarios de tests qui n'utilisent pas d'intergiciel DDS.

La sixième étape consiste à implémenter l'AFCS conçu au sein d'une plateforme de développement FreeScale afin de mesurer les performances de l'AFCS dans un environnement qui se rapproche plus d'une plateforme avionique.

La septième étape consiste à mesurer les performances de l'AFCS sur une plateforme FreeScale en utilisant un intergiciel DDS pour effectuer la communication entre l'AFCS et le simulateur de vol. Un seul scénario de test sera effectué à cette étape. Les résultats de ce scénario seront comparés à ceux de tous les autres scénarios précédents.

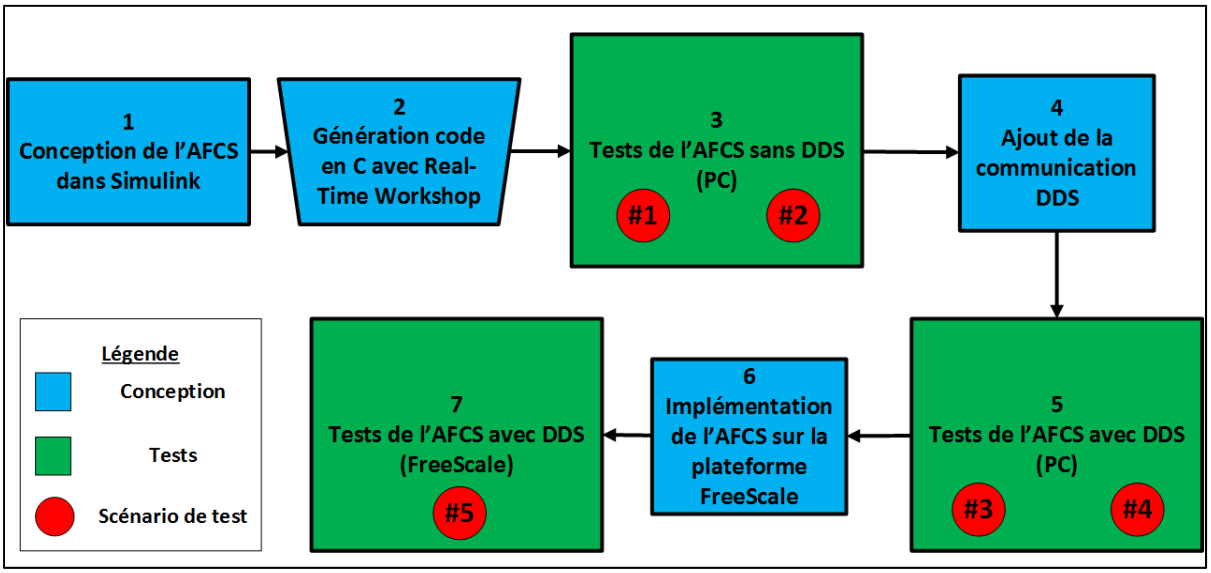


Figure 4.2 Méthodologie d'expérimentation

Les prochaines sections traitent donc des différentes étapes de cette méthodologie d'expérimentation.

4.3 Conception de l'AFCS avec MatLab/Simulink

Cette section représente la première étape de la méthodologie illustrée à la Figure 4.2. De ce fait, cette section porte sur la conception de l'AFCS dans le logiciel Simulink. L'environnement d'expérimentation qui est utilisé ainsi que la conception des quatre contrôleurs qui forment l'AFCS sont présentés dans cette section. Ces contrôleurs sont le contrôleur de roulis, le contrôleur de tangage, le contrôleur de vitesse et le contrôleur d'altitude.

4.3.1 Environnement d'expérimentation

Cette sous-section porte sur l'environnement d'expérimentation qui a été utilisé pour la conception de l'AFCS pour le Boeing 747-400. Afin de concevoir et de valider les boucles de commande de vol de l'AFCS, le PC #2 décrit dans le Tableau 3.2 été utilisé. Ce PC est muni de deux logiciels importants pour le développement de l'AFCS: Simulink et X-Plane. Les boucles de commande de vol qui forment l'AFCS sont conçues dans le logiciel Simulink et l'aérodynamique du Boeing 747-400 est simulée par le logiciel X-Plane. Le schéma expérimental utilisé pour la conception des boucles de commande est donc celui illustré à la Figure 4.3. Cette figure illustre que la communication entre l'AFCS et le simulateur de vol est effectuée par UDP/IP au sein d'un seul PC, donc avec un *udp loopback*. La configuration du solveur (*solver*) Simulink qui a été utilisée est décrite dans le Tableau 4.4.

De plus, un bloc de synchronisation temps réel (*Real-Time Synchronization*) de la boîte à outils *Real-Time Windows Target* a été utilisé pour faire la synchronisation temps réel de l'horloge de Simulink avec une horloge temps réel. Cette partie est importante, car normalement Simulink effectue sa simulation aussi rapidement qu'il le peut, or dans le cas présent, cette situation est indésirable puisque la simulation reçoit des données en provenance de X-Plane. La fréquence d'envois des données de X-Plane qui a été choisi est de 50 Hz, donc un paquet de données est envoyé à chaque 20 ms puisque le pas de calcul de l'AFCS est de 20 ms.

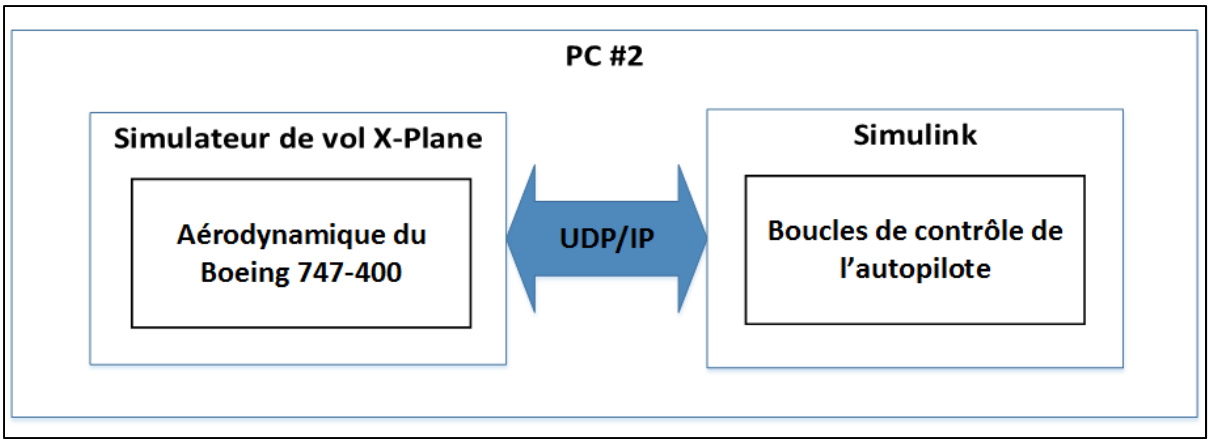


Figure 4.3 Schéma expérimental de la conception des boucles de commande sur Simulink

Tableau 4.4 Configuration utilisée pour le solveur Simulink

Paramètre	Valeur
Temps d'arrêt	Infini
Solveur	Discret (Aucun état continue)
Type de solveur	Pas Fixe
Taille des pas	0.02 secondes

4.3.2 Conception du contrôleur de roulis

Le contrôleur de roulis sert à stabiliser l'aéronef autour de l'axe des x tel qu'illustré à la Figure 1.8. Le contrôleur de roulis conçu est formé d'une boucle interne et d'une boucle externe telles qu'illustrées à la Figure 4.4.

Sur cette figure, la boucle interne sert à compenser pour la vitesse angulaire de l'aéronef et la boucle externe est un contrôleur PID. La fonction de transfert du servomoteur de l'élévateur est ensuite appliquée sur le signal de sortie. Finalement, le signal de sortie est ensuite multiplié par un facteur d'échelle qui permet de rendre le signal entre -1 et 1 tel que demandé par X-Plane. Les facteurs d'échelles utilisés ont été déterminés à partir des variations possibles des surfaces de contrôle que l'on retrouve dans le Tableau 4.2.

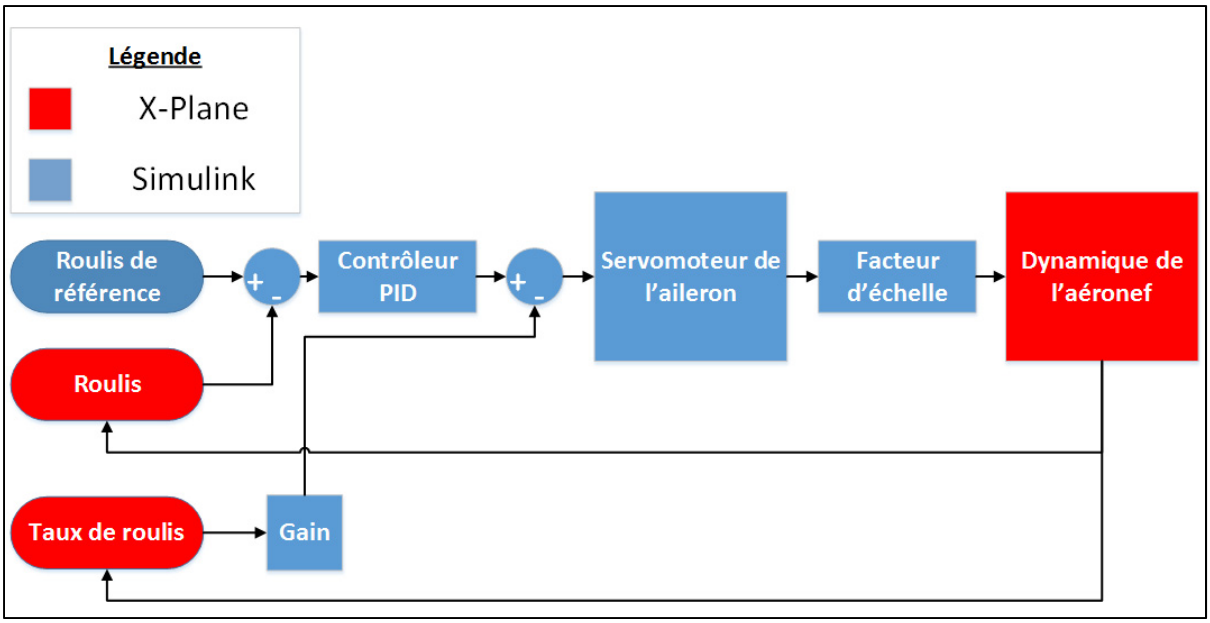


Figure 4.4 Schéma du contrôleur de roulis

Les différents gains de tous les contrôleurs conçus dans la section 4.3 ont été calibrés à l'aide de la méthode manuelle d'essai et erreur décrite dans la section 1.4.4. Pour faire la calibration du contrôleur, une entrée échelon de 5 degrés a été utilisée. Après avoir essayé plusieurs gains différents, les meilleurs résultats obtenus sont ceux illustrés à la Figure 4.5. Cette figure montre que le contrôleur a un temps de stabilisation de 4.18 secondes et un dépassement maximal de 9.52%. Elle montre également que l'erreur en régime permanent de ce contrôleur est de 0.09 degré. Cette erreur en régime permanent est due au fait que le contrôleur PID utilisé a seulement un gain proportionnel et l'utilisation d'un simple gain proportionnel ne permet pas l'élimination complète de l'erreur en régime permanent. Il est tout de même possible d'en conclure que le contrôleur de roulis arrive a stabiliser l'aéronef près d'une valeur de roulis de 0 degré.

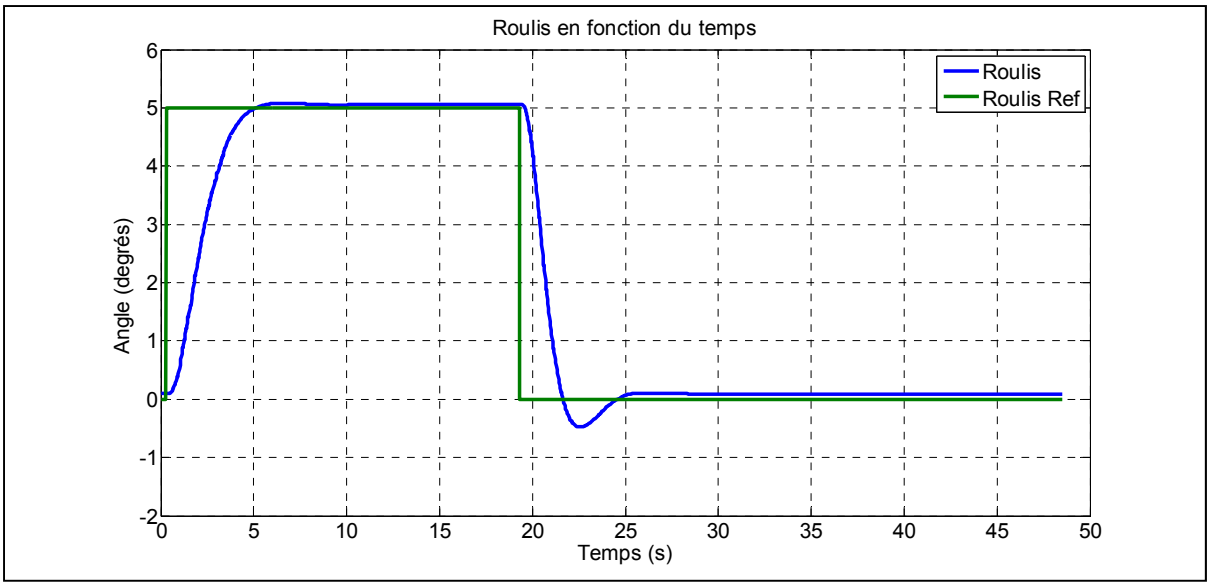


Figure 4.5 Réponse du contrôleur de roulis avec une entrée échelon

4.3.3 Conception du contrôleur de tangage

Le contrôleur de tangage sert à contrôler la rotation de l'aéronef autour de l'axe des y tel qu'illustré à la Figure 1.8. La conception du contrôleur de tangage est particulière puisque ce contrôleur sera utilisé par le contrôleur d'altitude qui sera conçu dans la section 4.3.5. Le contrôleur de tangage qui a été conçu est inspiré de l'article de (Barros dos Santos et de Oliveira, 2011), dans lequel le contrôle du tangage est conçu à l'aide de deux boucles de commande telles qu'illustrées à la Figure 4.6. Le contrôleur illustré sur cette figure est très similaire à celui du roulis illustré à la Figure 4.4.

Sur cette figure, la boucle interne sert à compenser pour la vitesse angulaire de l'aéronef et la boucle externe est un contrôleur PID. La fonction de transfert du servomoteur de l'élévateur est ensuite appliquée sur le signal de sortie. Finalement, le signal de sortie est ensuite multiplié par un facteur d'échelle qui permet de rendre le signal entre -1 et 1 tel que demandé par X-Plane. Les facteurs d'échelles utilisés ont été déterminés à partir des variations possibles des surfaces de contrôle que l'on retrouve dans le Tableau 4.2.

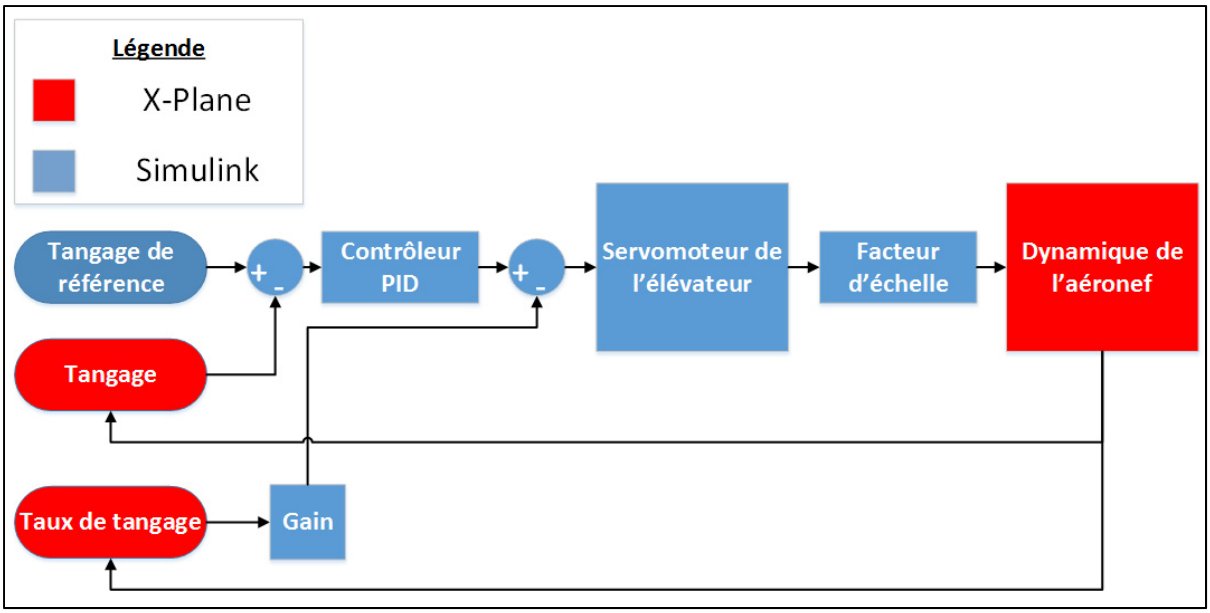


Figure 4.6 Schéma du contrôleur de tangage

Une entrée échelon d'un angle de 5 degrés a été utilisée afin de calibrer le contrôleur de tangage. La calibration a été effectuée avec la méthode manuelle par essai et erreur qui a été décrite dans la section 1.4.4. Les différents gains ont été calibrés afin d'éliminer l'oscillation dans la réponse en régime permanent, de minimiser le temps de montée et de minimiser l'erreur en régime permanent. De cette façon, la réponse du contrôleur de tangage illustrée à la Figure 4.7 a été obtenue. Cette figure montre une erreur en régime permanent de 0.01 degré et un temps de stabilisation de 23.26 secondes. Cette figure montre également un dépassement maximal de 8.19%. Ce dépassement maximal élevé explique le grand temps de stabilisation du contrôleur et est indésirable. Cependant, il est important de savoir qu'une entrée échelon ne sera jamais appliquée à l'entrée de ce contrôleur. En effet, la valeur de référence de ce contrôleur sera donnée par le contrôleur d'altitude et aura une forme plus triangulaire. De ce fait, le contrôleur de tangage a été testé avec une entrée triangulaire. Les résultats du contrôleur de tangage avec une entrée triangulaire sont illustrés à la Figure 4.8. Cette figure illustre que le contrôleur arrive à bien suivre la tendance de l'entrée triangulaire, mais qu'elle la suit avec un délai.

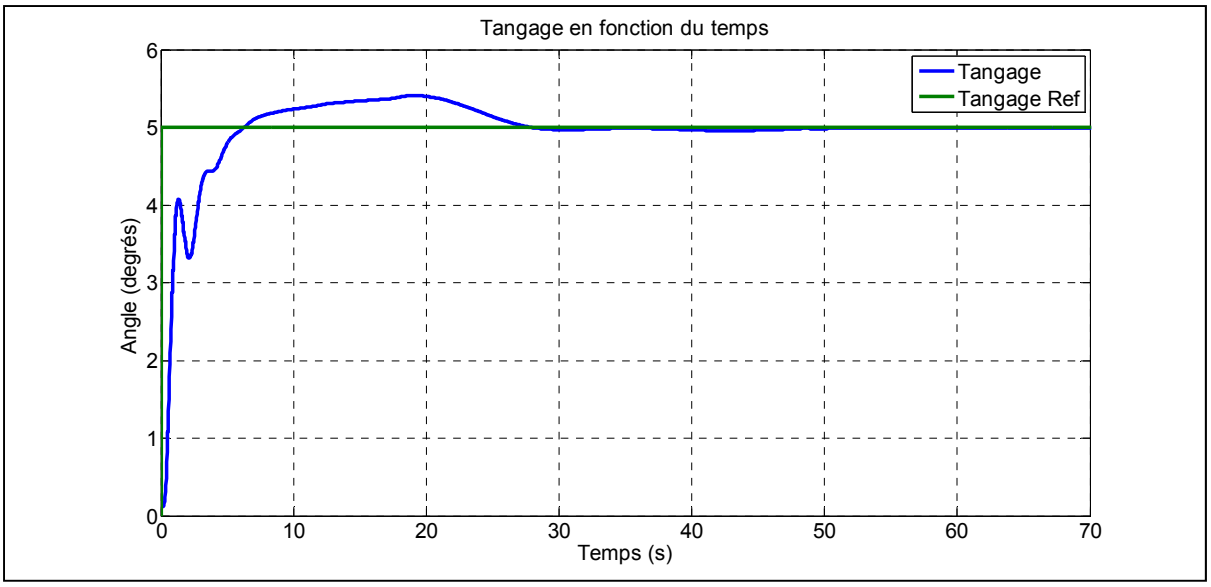


Figure 4.7 Réponse du contrôleur de tangage pour une entrée échelon

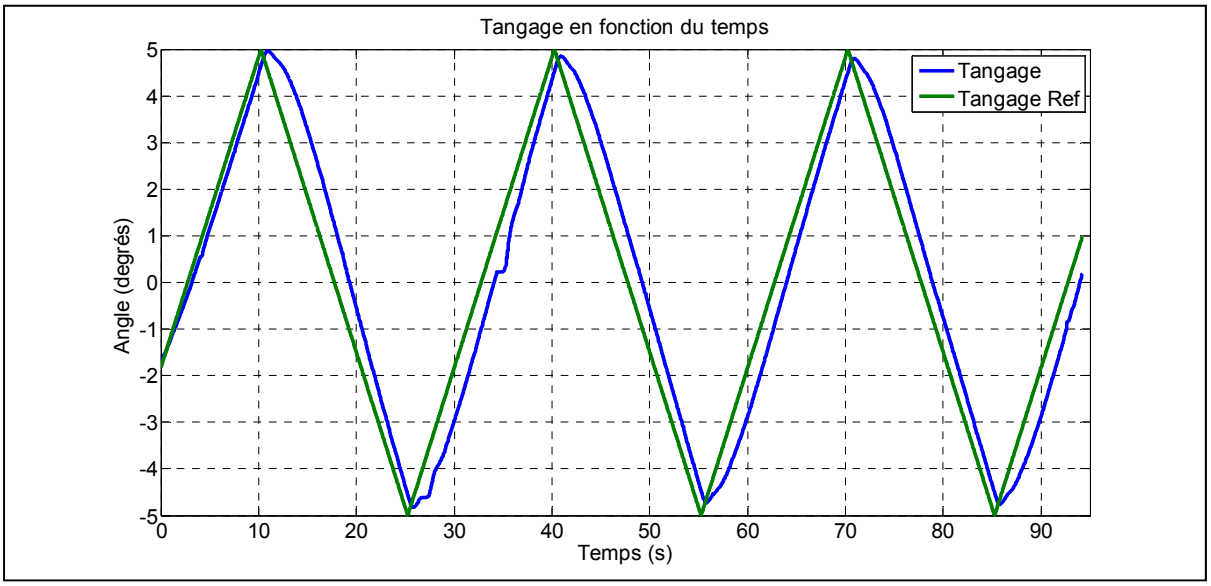


Figure 4.8 Réponse du contrôleur de tangage avec une entrée triangulaire

4.3.4 Conception du contrôleur de vitesse

Le contrôleur de vitesse conçu est utilisé pour contrôler la vitesse de l'aéronef le long de l'axe des x. Le contrôleur de vitesse conçu est similaire aux boucles de commande présentées

précédemment, cependant, elle ne contient pas de boucle externe. De ce fait, le schéma du contrôleur de vitesse est illustré à la Figure 4.9.

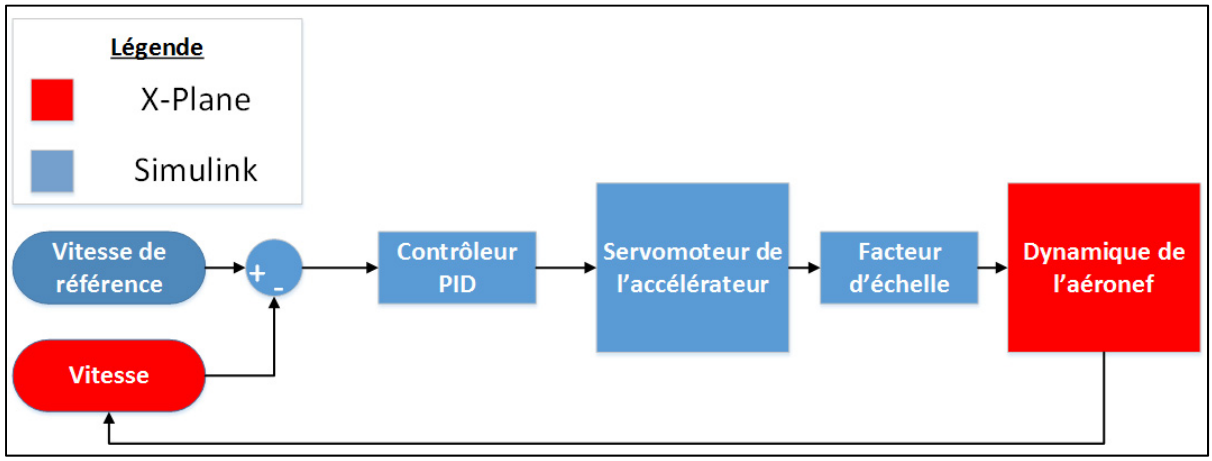


Figure 4.9 Schéma du contrôleur de vitesse

La calibration a été effectuée avec une vitesse de référence de 400 noeuds. Pour des raisons de sécurité, la valeur de l'accélérateur a été limitée entre 20% et 90% de sa valeur maximale. La réponse du contrôleur de vitesse après calibration est illustrée à la Figure 4.10.

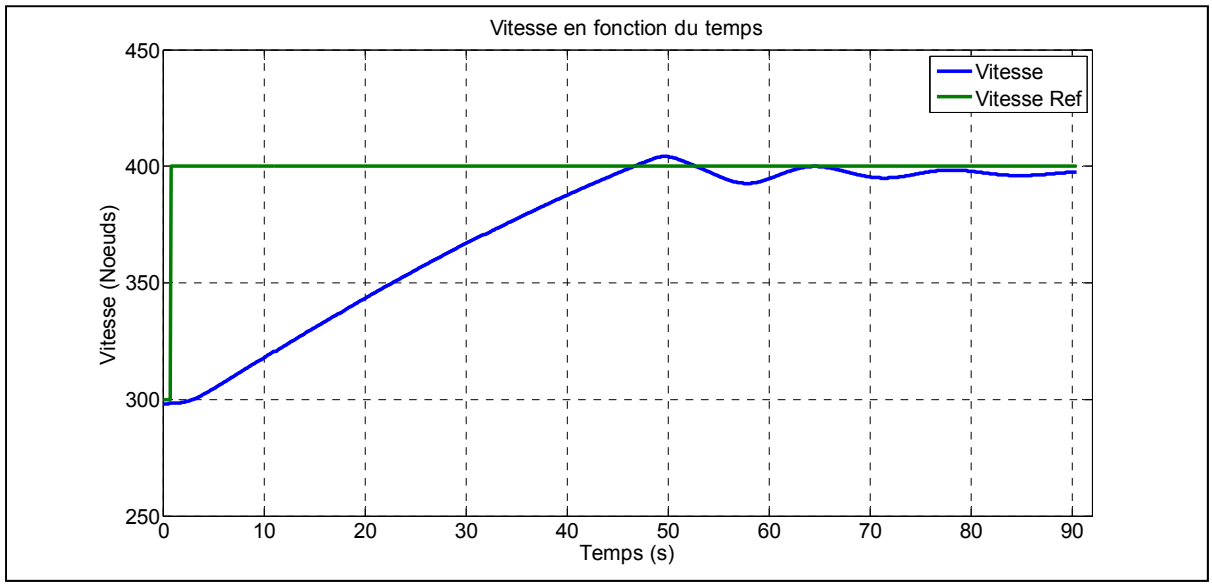


Figure 4.10 Réponse du contrôleur de vitesse

Cette figure montre que le contrôleur de vitesse a un temps de stabilisation de 43.94 secondes et un dépassement maximal de 4.1%. Cette dernière montre également une erreur en régime permanent de 4 nœuds. Encore une fois, cette erreur en régime permanent est attribuable au fait que le contrôleur PID utilisé n'a seulement qu'un gain proportionnel puisque c'est ce qui offrait le moins d'oscillations. En observant cette figure, il est possible d'en conclure que le contrôleur de vitesse arrive à stabiliser l'aéronef près d'une vitesse désirée avec seulement une faible oscillation.

4.3.5 Conception du contrôleur d'altitude

Le contrôleur d'altitude sert à stabiliser l'aéronef à une altitude en particulier. Ce contrôleur diffère des autres conçus précédemment puisque ce dernier n'a pas un contrôle direct sur l'aéronef, mais plutôt sur la valeur de tangage de référence qui sera utilisée par le contrôleur de tangage conçu dans la section 4.3.3. La Figure 4.11 illustre la boucle de commande de l'altitude. Il est important de remarquer sur cette figure que c'est la sortie du contrôleur PID qui va donner la valeur de référence du contrôleur de tangage.

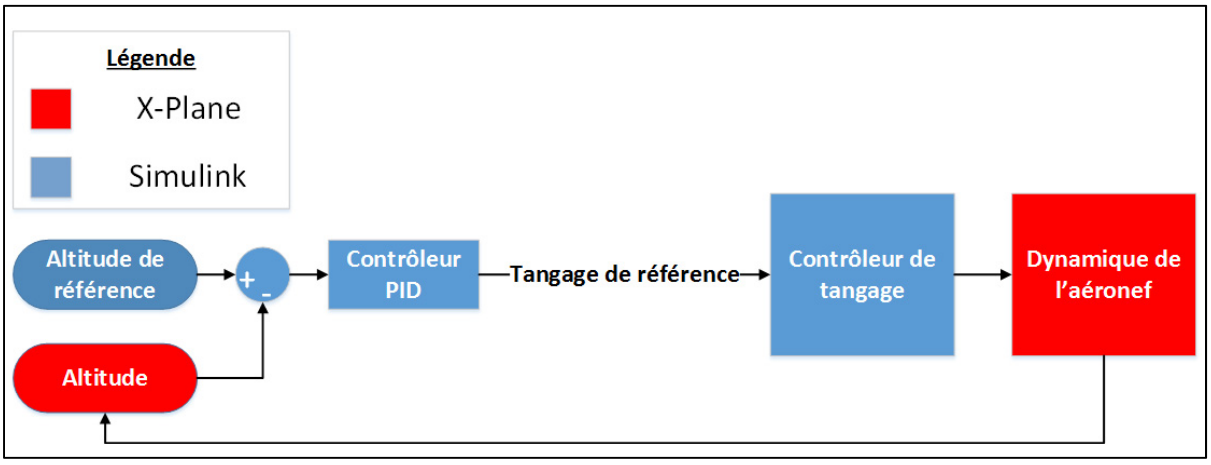


Figure 4.11 Schéma du contrôleur d'altitude

La conception de ce contrôleur est un peu plus particulière que les autres puisque ce contrôleur va donner la valeur de référence du contrôleur de tangage. De ce fait, un simple contrôleur proportionnel a été utilisé puisqu'il faut éviter les changements brusques de valeur

de référence du tangage pour ne pas affecter négativement les performances du contrôleur de tangage. La calibration de ce contrôleur a été effectuée en faisant passer l'aéronef d'une altitude de 3000 pieds à 5000 pieds. La vitesse a été stabilisée à 300 nœuds et le roulis à 0 degré pour la calibration. La Figure 4.12 illustre la réponse du contrôleur d'altitude suite à la calibration de ce dernier. Cette figure montre que le contrôleur d'altitude a un temps de stabilisation de 64.22 secondes ainsi qu'un faible dépassement maximal de 0.65%. Cette dernière montre également une erreur en régime permanent de 16 pieds avec aucune oscillation. En observant cette image, il est possible d'en conclure que le contrôleur d'altitude arrive à bien stabiliser l'aéronef près de l'altitude désirée.

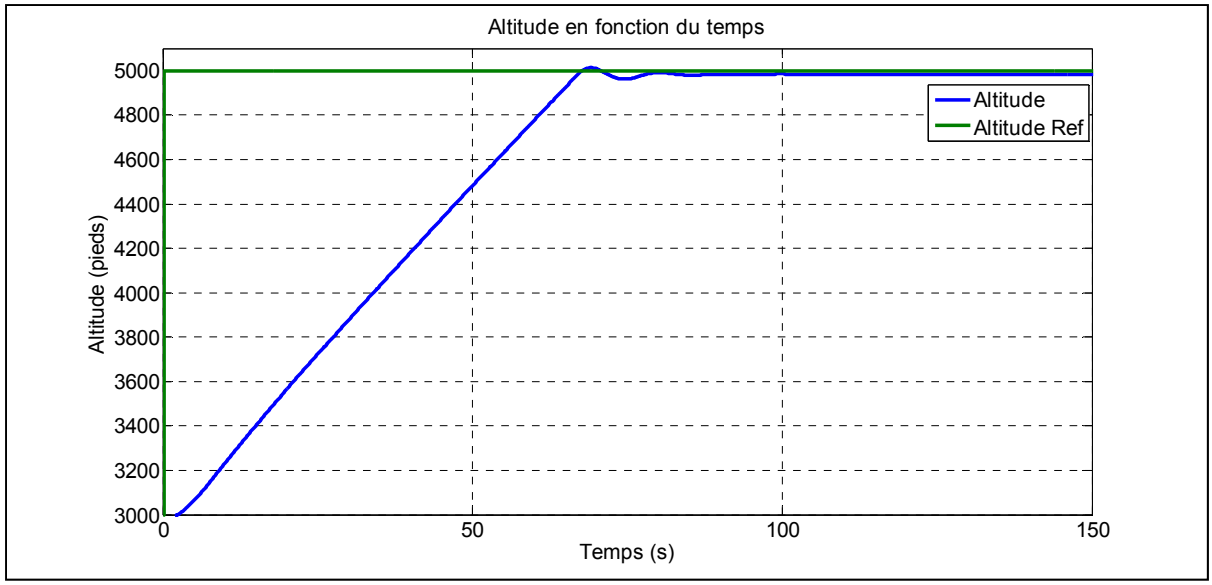


Figure 4.12 Réponse du contrôleur d'altitude

4.4 Test de l'AFCS sur des PC sans intergiciel DDS

Cette section porte sur l'évaluation des performances de l'AFCS sans l'utilisation d'un intergiciel DDS, mais en dehors de l'environnement de Simulink. Cette section traite donc de l'étape 2 et 3 de la méthodologie illustrée à la Figure 4.2. De ce fait, le code en langage C de l'AFCS conçu dans la section précédente a dû être généré. Pour ce faire, l'outil Simulink Coder a été utilisé pour ainsi générer le code en C de l'AFCS à partir de son schéma Simulink.

Afin de vérifier les performances de l'AFCS généré en langage C grâce au Simulink Coder, des expérimentations ont été effectuées. Pour ce faire, du code a été ajouté au code généré par le Simulink Coder afin d'ajouter la réception des données de X-Plane par UDP/IP avec la librairie winsock2. De plus, la routine exécutée à chaque pas de calcul a dû être attachée à une interruption au 20 ms.

4.4.1 Scénarios de tests

Deux scénarios de test ont été effectués afin de mesurer les performances de l'AFCS sans l'utilisation d'un intergiciel DDS. Le premier consiste à exécuter l'AFCS sur le même PC que X-Plane tel qu'illustré à la Figure 4.13. Tel qu'illustré sur cette figure, la communication entre X-Plane et l'AFCS est effectuée à l'aide du protocole UDP/IP. Le second scénario de test consiste a exécuté l'AFCS sur un PC différent que X-Plane et de connecter les deux PC ensemble par un câble Ethernet RJ45 tel qu'illustré à la Figure 4.14.

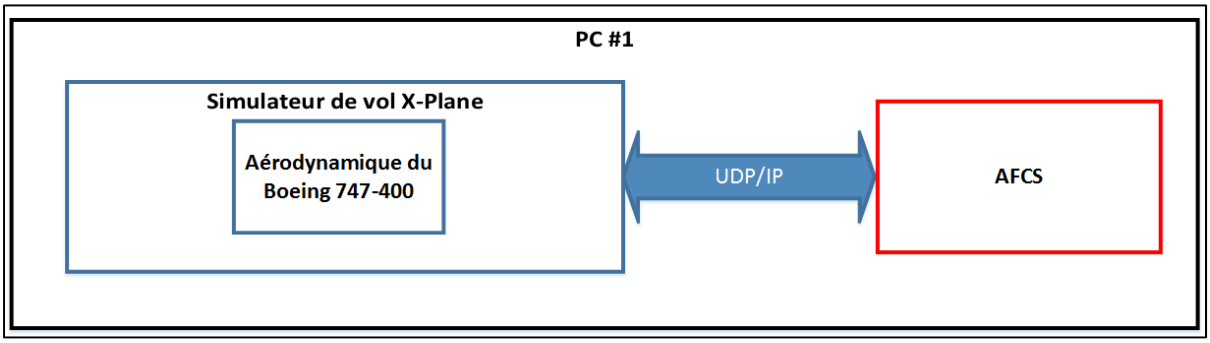


Figure 4.13 Scénario #1: Test de l'AFCS sans intergiciel DDS sur un seul PC

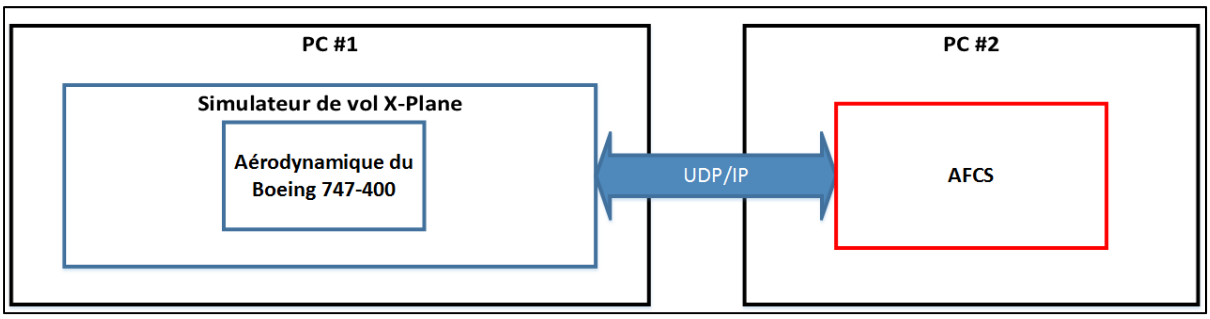


Figure 4.14 Scénario #2: Test de l'AFCS sans intergiciel DDS entre deux PC

Tous les scénarios effectués dans ce chapitre utilisent les paramètres de vol identifiés dans le Tableau 4.5. Ce tableau indique donc que les différents scénarios visent à faire passer le Boeing 747-400 d'une valeur initiale à une valeur désirée d'altitude, de vitesse et de roulis.

Tableau 4.5 Paramètres de vol pour les scénarios de tests

Paramètres de vol	Valeur initiale	Valeur désirée (Consigne)
Altitude	3000 pieds	8000 pieds
Vitesse	300 nœuds	400 nœuds
Roulis	0 degré	0 degré

4.4.2 Analyse des résultats

Les résultats des deux scénarios sans intergiciel DDS sont illustrés par la ligne bleue et la ligne verte dans la Figure 4.15. Cette figure illustre que l'AFCS arrive a stabiliser le roulis, l'altitude ainsi que la vitesse de l'aéronef. Les résultats présentés dans le Tableau 4.6 ont été tirés de la Figure 4.15.

Ce tableau montre une faible erreur en régime permanent du roulis (e_{roulis}) de 0.10 degré pour le scénario sans DDS sur un PC et de 0.09 degré pour le scénario sans DDS avec deux PC.

De plus, ce tableau montre que le temps de stabilisation de l'altitude ($t_{s\ altitude}$) de l'AFCS est légèrement plus élevé pour le scénario sans DDS sur deux PC par rapport au scénario sur un seul PC. Ce tableau présente également que l'erreur en régime permanent de l'altitude ($e_{altitude}$) est de 13 pieds pour les deux scénarios.

Tableau 4.6 Résultats des scénarios de test sans DDS

Scénario	e_{roulis} (degré)	$t_{s\ altitude}$ (s)	$e_{altitude}$ (pieds)	$t_{s\ vitesse}$ (s)	$e_{vitesse}$ (noeuds)
Sans DDS (1 PC)	0.10	81.51	13	117.61	4
Sans DDS (2 PC)	0.09	83.40	13	120.83	4

L'AFCS obtient également un temps de stabilisation de la vitesse ($t_{s\ vitesse}$) légèrement plus élevée pour le scénario sans DDS sur deux PC par rapport au scénario sur un seul PC. Finalement, l'erreur en régime permanent de la vitesse ($e_{vitesse}$) de l'AFCS est de 4 noeuds pour les deux scénarios de test.

Ces résultats seront utilisés en tant que base de comparaison afin d'évaluer l'impact de l'utilisation d'un intergiciel DDS dans les prochains scénarios de tests de ce chapitre.

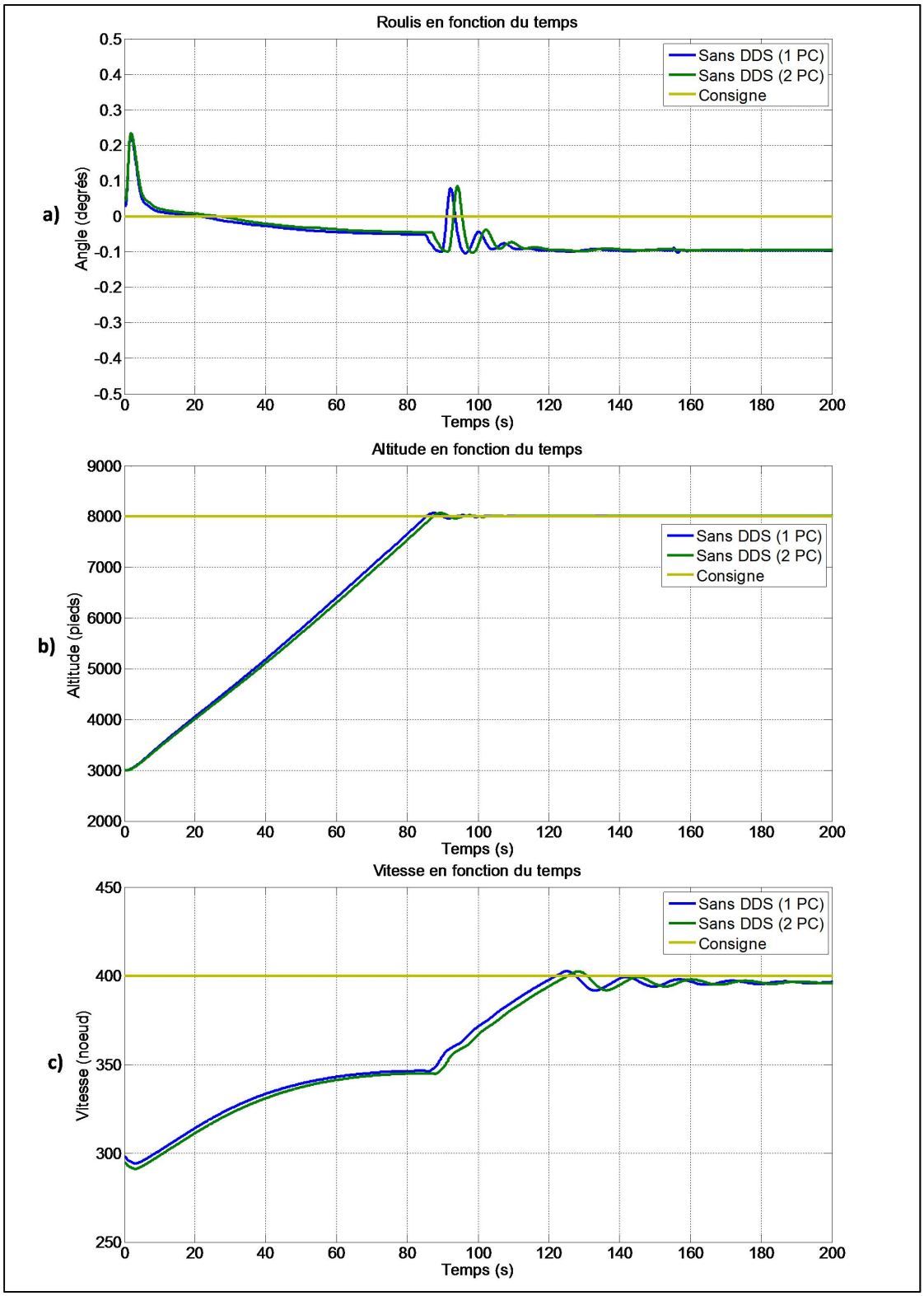


Figure 4.15 Résultats des scénarios de tests sans intergiciel DDS
a) roulis, b) altitude, c) vitesse

4.5 Test de l'AFCS sur des PC avec intergiciel DDS

Cette section porte sur l'ajout d'un intergiciel DDS aux scénarios de tests effectués dans la section précédente. De cette façon, il sera ainsi possible mesurer l'impact de l'utilisation de l'intergiciel DDS sur les performances de l'AFCS dans un environnement PC. Cette section traite de l'étape 4 et 5 de la méthodologie illustrée à la Figure 4.2.

4.5.1 Conception d'un plug-in DDS pour X-Plane

Suite aux expérimentations de l'AFCS avec une communication UDP/IP sans intergiciel DDS, la communication DDS a été ajoutée dans l'environnement d'expérimentation. Pour ce faire, un *plug-in* a dû être conçu afin de permettre à X-Plane de communiquer par DDS avec l'AFCS. Le *plug-in* a été conçu avec la trousse de développement logiciel (SDK) de X-Plane. Ce dernier fait appel aux différentes fonctions de RTI Connext Micro qui permettent ainsi d'établir une communication DDS. Ce *plug-in* est configurable à travers un fichier XML qui est lu au lancement du *plug-in*.

Pour le reste des expériences de ce projet de recherche, une période d'envois des données de 10 ms a été fixée dans le fichier XML afin de s'assurer que le *plug-in* envoie les données à l'AFCS assez rapidement pour assurer le bon fonctionnement de ce dernier.

4.5.2 Scénarios de tests

Afin de pouvoir comparer l'impact de l'utilisation d'un intergiciel DDS sur les performances de l'AFCS, deux scénarios de tests supplémentaires ont été effectués. Le premier consiste a exécuter l'AFCS sur le même PC que X-Plane et d'effectuer la communication entre les deux avec l'intergiciel Connext Micro tel qu'illustré à la Figure 4.16.

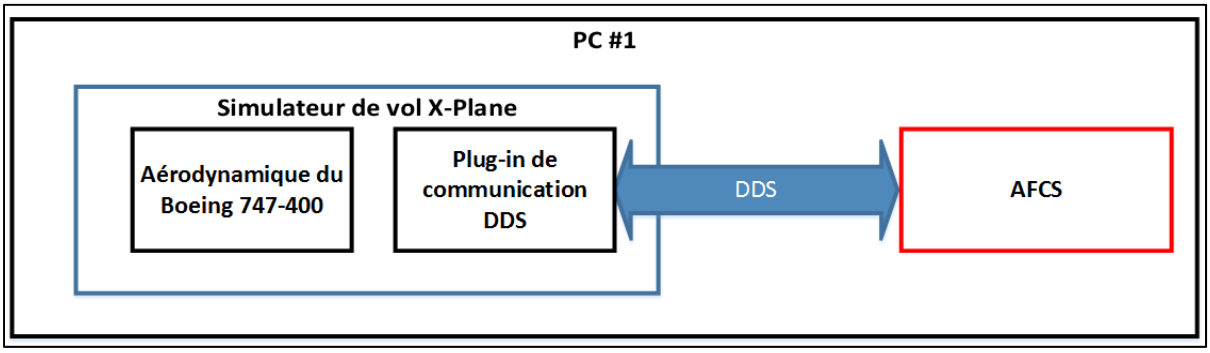


Figure 4.16 Scénario #3: Test de l'AFCS avec intergiciel DDS sur un seul PC

Le second scénario consiste à exécuter l'AFCS sur un PC différent que X-Plane tel qu'illustré à la Figure 4.17. Dans ce scénario, X-Plane et l'AFCS communiquent ensemble à l'aide de l'intergiciel Connext Micro à travers un câble Ethernet RJ45.

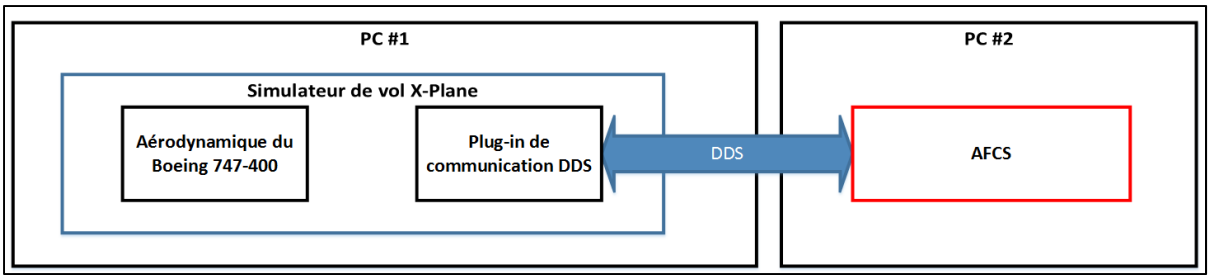


Figure 4.17 Scénario #4: Test de l'AFCS avec intergiciel DDS entre deux PC

Les polices de QoS utilisées pour ces deux scénarios de tests sont présentées dans le Tableau 4.7. Un délai de publication/réception de 20 ms a été imposé afin de respecter le pas de calcul de l'AFCS. La vivacité a été établie à 1 seconde afin que X-Plane permette à l'utilisateur de contrôler manuellement l'aéronef si l'AFCS n'envoie pas son message de vivacité toutes les 1

seconde. La fiabilité a été configurée pour le meilleur effort afin de ne pas ralentir le système avec le renvoi des paquets manqués puisque l'AFCS n'est intéressé que par les valeurs les plus récentes des capteurs de l'aéronef. Toutes les autres polices de QoS ont la valeur par défaut telles que définies dans le guide de l'utilisateur de l'intergiciel Connext Micro (Real Time Innovation, 2012).

Tableau 4.7 Polices de QoS utilisées pour la communication DDS

Police de QoS	**Valeur**
Délai de publication/réception	20 ms
Vivacité	1 seconde
Fiabilité	Meilleur effort

Les deux scénarios de test présentés utilisent les mêmes paramètres de vols pour l'aéronef que pour les expérimentations sans intergiciel DDS soit ceux indiqués dans le Tableau 4.5.

4.5.3 Analyse des résultats

Les résultats des deux scénarios de tests avec intergiciel DDS sur des PC sont illustrés dans la Figure 4.18. Les courbes des scénarios de tests sans intergiciel DDS sont aussi tracées sur cette figure, afin d'observer l'impact de l'ajout de l'intergiciel DDS sur les résultats de l'AFCS. Cette figure montre que l'ajout de l'intergiciel DDS n'empêche pas l'AFCS de stabiliser l'aéronef au roulis, à l'altitude ainsi qu'à la vitesse désirée. Les résultats présentés dans le Tableau 4.8 ont été tirés de la Figure 4.18.

L'ajout de l'intergiciel DDS a eu peu d'impact sur l'erreur en régime permanent (e_{roulis}) du roulis puisqu'elle passe de 0.09 degré pour le scénario sans DDS avec deux PC à 0.07 degré pour le scénario avec DDS avec deux PC.

De plus, le temps de stabilisation de l'altitude ($t_{s\ altitude}$) de l'AFCS augmente lors de l'ajout de l'intergiciel. En effet, l'ajout de l'intergiciel DDS sur un seul PC fait passé le temps de stabilisation de l'altitude de 81.51 secondes à 92.89 secondes. L'ajout de l'intergiciel DDS au

scénario à deux PC fait passer le temps de stabilisation de l'altitude de 83.40 secondes à 98.38 secondes. De ce fait, il est possible d'en conclure que l'ajout de l'intergiciel a participé à l'augmentation du temps de stabilisation du contrôleur d'altitude. Cette augmentation est attribuable au délai de communication causé par l'utilisation de l'intergiciel DDS et au fait que le *plug-in* de communication ne permet pas d'envoyer les commandes à l'aéronef aussi directement que par UDP/IP tel qu'utilisé dans les scénarios sans intergiciel DDS. Il y a également une faible diminution de l'erreur en régime permanent de l'altitude ($e_{altitude}$) pour les deux scénarios de tests.

Tableau 4.8 Résultats des scénarios de tests avec DDS sur PC

Scénario	$\mathbf{e_{roulis}}$ **(degré)**	$\mathbf{t_{s\ altitude}}$ **(s)**	$\mathbf{e_{altitude}}$ **(pieds)**	$\mathbf{t_{s\ vitesse}}$ **(s)**	$\mathbf{e_{vitesse}}$ **(noeuds)**
Sans DDS (1 PC)	0.10	81.51	13	117.61	4
Sans DDS (2 PC)	0.09	83.40	13	120.83	4
Avec DDS (1 PC)	0.08	92.89	10	137.81	4
Avec DDS (2 PC)	0.07	98.38	9	147.78	4

Le Tableau 4.8 montre également une augmentation considérable du temps de stabilisation de la vitesse ($t_{s\ vitesse}$). En effet, le temps de stabilisation de la vitesse passe de 120.83 secondes pour le scénario sans DDS entre deux PC à 147.78 secondes pour le scénario avec DDS entre deux PC. Cependant, l'ajout de l'intergiciel DDS n'a pas eu d'impact sur l'erreur en régime permanent de la vitesse ($e_{vitesse}$) qui est restée à 4 noeuds.

Il est donc possible d'en conclure que l'ajout de l'intergiciel DDS ne mène pas l'AFCS à l'instabilité, mais augmente le temps que prend le système pour atteindre sa stabilité. Cette augmentation du temps de stabilisation est attribuable au délai de communication causé par l'utilisation de l'intergiciel DDS et au fait que le *plug-in* de communication ne permet pas d'envoyer les commandes à l'aéronef aussi directement que par UDP/IP tel qu'utilisé dans les

scénarios sans intergiciel DDS. Il est également possible d'en conclure que l'ajout de l'intergiciel DDS n'a pas un impact significatif sur l'erreur en régime permanent du système.

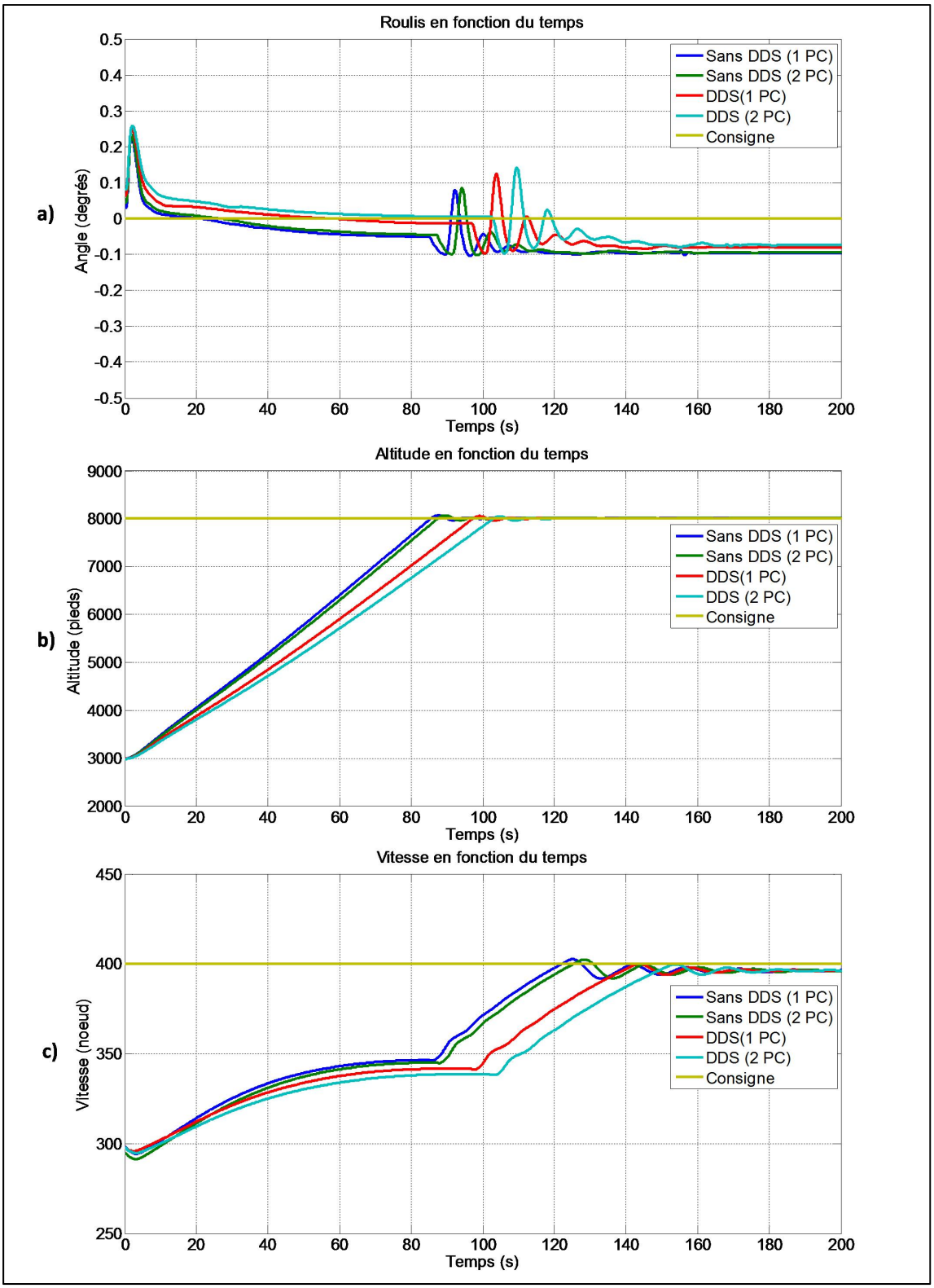


Figure 4.18 Résultats des scénarios de tests avec intergiciel DDS
a) roulis, b) altitude, c) vitesse

4.6 Test de l'AFCS avec intergiciel DDS sur la plateforme FreeScale

Cette section porte sur les tests de l'AFCS avec un intergiciel DDS, mais sur une plateforme FreeScale MPC8572DS. Le but de ces tests est de mesurer les performances de l'AFCS en utilisant un intergiciel DDS sur une plateforme embarquée dont les spécifications se rapprochent d'une plateforme avionique. Cette section traite donc de l'étape 6 et 7 de la méthodologie illustrée à la Figure 4.2.

4.6.1 Plateforme FreeScale MPC8572DS

La plateforme FreeScale MPC8572DS est une carte de développement basée sur une architecture PowerPC. La Figure 4.19 illustre cette plateforme de développement.

Figure 4.19 Plateforme de développement FreeScale MPC8572DS
Tirée de FreeScale (2014)

Cette dernière est munie d'un processeur e500 bi-cœur de 1.5 GHz. De plus, cette carte de développement est également munie de deux ports Ethernet pouvant aller jusqu'à 1 Gbit/s ainsi que d'un port série RS232. La carte développement est également munie d'une banque de mémoire flash de 1 Go. Cette plateforme a été choisie parce qu'elle est déjà disponible dans notre laboratoire ainsi que le BSP (*board support package*) de VxWorks 6.9 pour cette plateforme de développement. De plus, tel que mentionné dans l'article de (Sanchez et al.,

2013), FreeScale a un historique de supporter les requis du domaine aérospatial tels qu'un long cycle de vie avec support, le support d'une grande plage de température ainsi que l'accès aux données critiques de leurs conceptions.

4.6.2 Environnement de test

Le système d'exploitation temps réel VxWorks 6.9 de Wind River est utilisé pour effectuer les tests sur la plateforme de développement. Ce n'est cependant pas un système d'exploitation qui répond aux besoins de la norme avionique ARINC 653. Le système d'exploitation VxWorks 653 de Wind River, qui est conforme à la norme avionique ARINC 653, a été considéré dans le cadre de ce projet. Cependant, le manque de disponibilité d'un BSP à faible coût ainsi que les efforts substantiels qui auraient été nécessaires pour déployer l'intergiciel Connext Micro sur ce système d'exploitation ont justifié le choix de VxWorks 6.9 comme première étape pour la validation.

Le système d'exploitation VxWorks 6.9 vient avec l'environnement intégré de développement (IDE) Wind River Workbench 3.3. Cet IDE est utilisé afin d'exécuter l'AFCS sur la plateforme FreeScale mentionnée précédemment.

En premier lieu, le système d'exploitation VxWorks 6.9 a dû être installé sur la plateforme FreeScale. Le processus de chargement du système d'exploitation VxWorks 6.9 sur la plateforme FreeScale est expliqué dans l'annexe I.

En second lieu, le code de l'AFCS a dû être adapté afin d'être utilisable sur l'OS VxWorks 6.9. Pour ce faire, le code de l'AFCS a été compilé dans un RTP (*Real Time Process*) afin d'être exécutable dans un environnement VxWorks 6.9. Lorsqu'on veut démarrer l'exécution de l'AFCS, il suffit d'envoyer le RTP à la plateforme FreeScale par l'entremise de l'IDE Wind River Workbench. De plus, une version de Connext Micro spécifiquement pour VxWorks 6.9 a été utilisée afin d'effectuer des communications DDS sur la plateforme FreeScale.

4.6.3 Scénario de test

Le scénario de test effectué dans cette section est illustré à la Figure 4.20. L'AFCS s'exécute sur la plateforme FreeScale alors que X-Plane s'exécute sur le PC #1 tel qu'illustrée à la Figure 4.20. De plus, la plateforme FreeScale et le PC #1 sont connectés ensemble par un câble Ethernet RJ45.

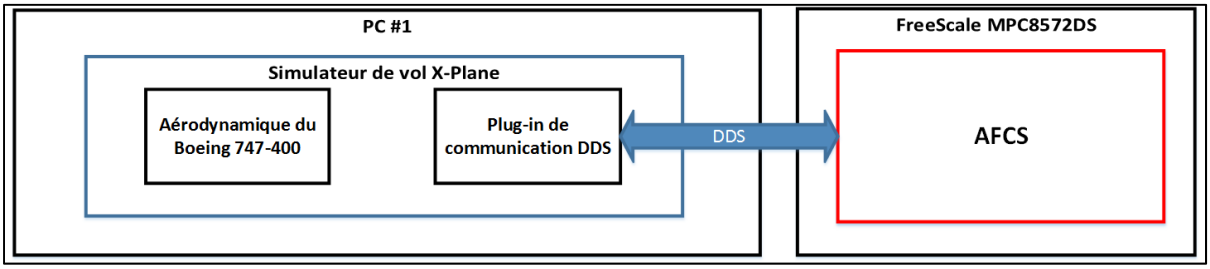


Figure 4.20 Scénario #5: Test de l'AFCS sur la plateforme FreeScale avec intergiciel DDS

4.6.4 Analyse des résultats

Les résultats du scénario de test avec intergiciel DDS sur la plateforme FreeScale sont illustrés à la Figure 4.21. Sur cette figure, tous les résultats précédents ont également été tracés afin de pouvoir faire la comparaison des performances. Cette figure montre que le scénario de test avec DDS sur la plateforme FreeScale arrive à bien stabiliser le roulis, l'altitude ainsi que la vitesse de l'aéronef tout comme dans les scénarios précédents. Cependant, le temps de stabilisation sur la plateforme FreeScale est plus grand que pour les autres scénarios. Les résultats présentés dans le Tableau 4.9 ont ainsi été tirés de la Figure 4.21.

Dans ce tableau, le scénario avec DDS sur la plateforme FreeScale donne une erreur en régime permanent du roulis (e_{roulis}) de 0.07 degré tout comme le scénario avec DDS entre deux PC. Il est donc possible d'en conclure que le passage de l'environnement PC à la plateforme FreeScale n'a pas eu d'effet observable sur l'erreur en régime permanent du roulis.

De plus, le scénario sur la plateforme FreeScale donne un temps de stabilisation de l'altitude de 98.71 secondes ce qui est presque identique à ce qui a été obtenu dans le scénario avec DDS entre deux PC. Il est donc possible d'en conclure que le passage d'un PC sous Windows 7 vers une plateforme FreeScale avec VxWorks 6.9 n'a pas eu d'impact significatif sur le temps de stabilisation de l'altitude. L'erreur en régime permanent de l'altitude ($e_{altitude}$) du scénario sur la plateforme FreeScale est similaire à ce qui a été obtenu dans le scénario avec DDS sur deux PC.

Tableau 4.9 Résultats du scénario avec DDS sur la plateforme FreeScale

Scénario	**e_{roulis} (degré)**	**$t_{s\ altitude}$ (s)**	**$e_{altitude}$ (pieds)**	**$t_{s\ vitesse}$ (s)**	**$e_{vitesse}$ (nœuds)**
Sans DDS (1 PC)	0.10	81.51	13	117.61	4
Sans DDS (2 PC)	0.09	83.40	13	120.83	4
Avec DDS (1 PC)	0.08	92.89	10	137.81	4
Avec DDS (2 PC)	0.07	98.38	9	147.78	4
Avec DDS (FreeScale)	0.07	98.71	8	150.55	4

Finalement, une faible augmentation du temps de stabilisation de la vitesse ($t_{s\ vitesse}$) a été observée par rapport au scénario avec DDS sur deux PC puisque ce dernier passe de 147.78 à 150.55 secondes. Il n'y a cependant aucune variation au niveau de l'erreur en régime permanent de la vitesse ($e_{vitesse}$) qui est de 4 nœuds pour tous les scénarios.

En somme, il est possible d'utiliser un intergiciel DDS avec une application avionique tout en gardant la stabilité de cette dernière et en obtenant des résultats similaires à une utilisation sans intergiciel DDS. Cependant, le temps de stabilisation de l'application est affecté par l'utilisation de l'intergiciel DDS ce qui pourrait causer des problèmes pour les applications avioniques dont le temps de réponse est critique.

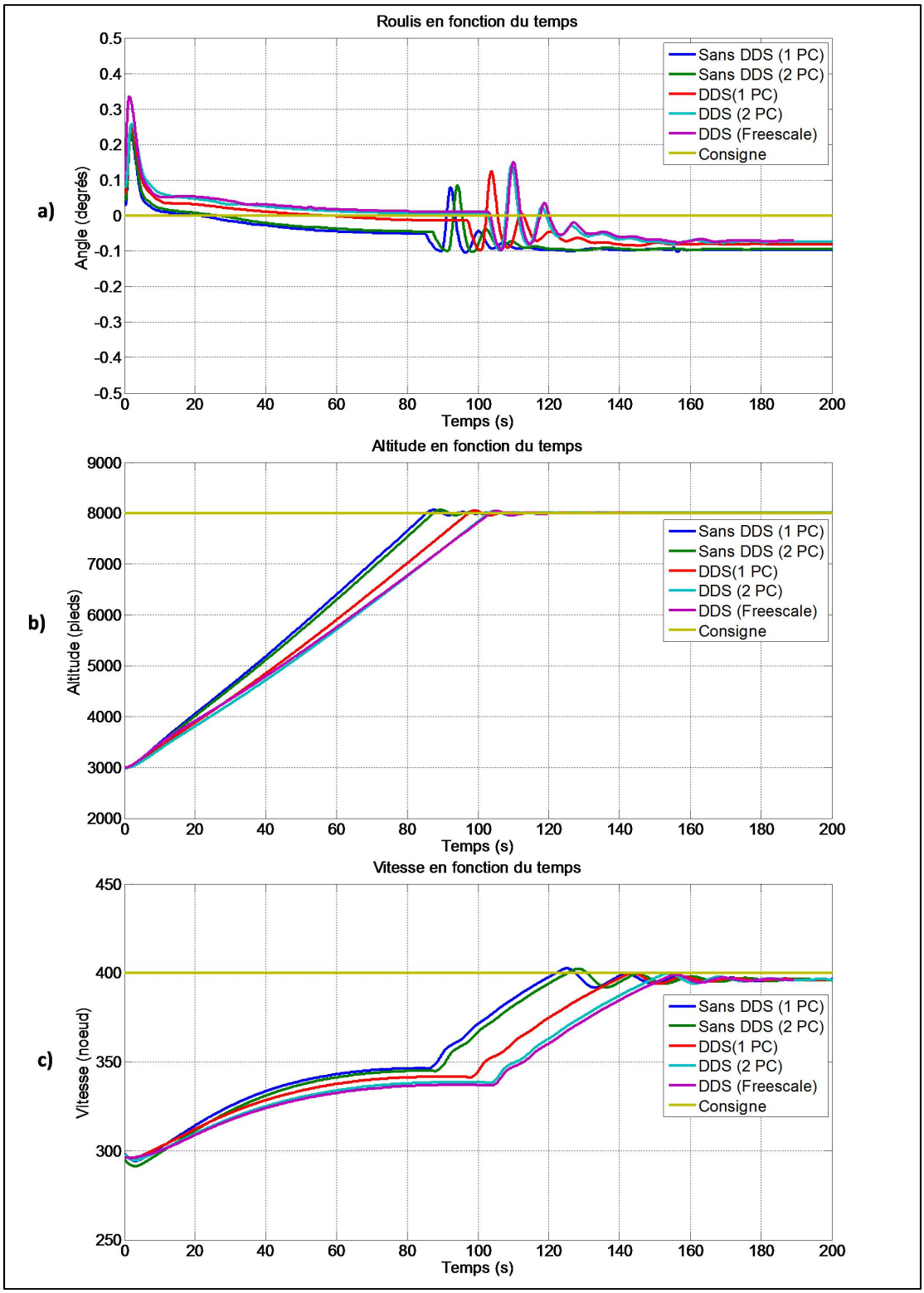


Figure 4.21 Résultats du scénario de test sur la plateforme FreeScale
a) roulis, b) altitude, c) vitesse

4.7 Conclusion

En somme, le but de ce chapitre était d'évaluer l'impact de l'utilisation d'un intergiciel DDS sur les performances d'un AFCS. De ce fait, un AFCS pour un Boeing 747-400 a été conçu dans ce chapitre. Les performances de cet AFCS ont ensuite été évaluées à travers cinq scénarios de tests. Les performances de l'AFCS sont mesurées par rapport au contrôleur de roulis, au contrôleur d'altitude ainsi qu'au contrôleur de vitesse. Tous les scénarios de tests impliquent l'AFCS qui doit communiquer avec le simulateur de vol X-Plane qui sert à simuler l'aérodynamique du Boeing 747-400.

En premier lieu, deux scénarios de tests ont été effectués sans l'utilisation d'un intergiciel DDS sur des PC. Les résultats de ces scénarios de test ont été utilisés comme base de comparaison pour ainsi évaluer l'impact de l'ajout d'un intergiciel DDS sur les performances de l'AFCS.

En deuxième lieu, deux scénarios de tests ont été effectués en utilisant l'intergiciel DDS Connext Micro sur des PC. Les résultats de ces scénarios illustrent une augmentation du temps de stabilisation des différents contrôleurs de l'AFCS. Ceci inclut le contrôleur de roulis, le contrôleur d'altitude et le contrôleur de vitesse. Les performances de l'AFCS sont cependant stables et seulement le temps de stabilisation des contrôleurs de l'AFCS est affecté significativement.

En troisième lieu, un scénario de test sur une plateforme FreeScale a été effectué. Ce scénario a pour but d'évaluer les performances de l'AFCS avec un intergiciel DDS, et ce, dans un environnement temps réel. De ce fait, le système d'exploitation VxWorks 6.9 a été utilisé. Les résultats de ce scénario de test ont démontré des résultats similaires à ce qui a été observé dans les scénarios avec un intergiciel DDS entre deux PC. Une augmentation du temps de stabilisation des différents contrôleurs de l'AFCS par rapport aux scénarios sans intergiciel DDS a été observée, mais l'AFCS est tout de même resté stable.

Finalement, il a été conclu que l'utilisation d'un intergiciel DDS pour une application avionique telle qu'un AFCS est viable puisque les performances de cette dernière sont stables. Il y a cependant une dégradation du temps de stabilisation qui est attribuable à l'ajout de l'intergiciel DDS ainsi qu'au fait que l'utilisation d'un *plug-in* au sein de X-Plane ne permet pas d'envoyer aussi directement les commandes à l'aéronef que par UPD/IP tel qu'utilisé dans les scénarios sans intergiciel DDS. Il est donc possible que l'utilisation d'un intergiciel DDS ne soit pas acceptable pour les applications dont le temps de réponse est critique.

CHAPITRE 5

REDONDANCE EN UTILISANT UN INTERGICIEL DDS

La redondance consiste à dupliquer les éléments critiques d'un système dans le but d'augmenter la sureté, la fiabilité et la disponibilité en cas de failles ou de dégradation de ce dernier (Gohil, Basavalingarajaiah et Ramachandran, 2011). Les systèmes avioniques nécessitent un niveau de sureté, de fiabilité et de disponibilité élevé, de ce fait, les concepteurs d'aéronefs utilisent souvent la redondance. La redondance étant une approche couramment utilisée dans le domaine avionique, il est donc important qu'un intergiciel DDS puisse être utilisé pour effectuer de la redondance. De ce fait, un scénario de test est effectué dans ce chapitre afin de faire de la redondance de systèmes à l'aide des QoS offertes par l'intergiciel DDS. Pour ce faire, la police de propriétaire est utilisée. Le but de ce chapitre est donc d'évaluer les performances d'un système redondant qui a été conçu à l'aide d'un intergiciel DDS.

Ce chapitre aborde tout d'abord le modèle de redondance de réserve et explique comment effectuer ce modèle de redondance à l'aide d'un intergiciel DDS. Par la suite, le scénario de test utilisé est présenté. Finalement, les résultats obtenus sont présentés et analysés.

5.1 Redondance de réserve

Il existe plusieurs modèles de redondance qui ont leurs avantages et inconvénients, donc le modèle utilisé dépend généralement du comportement désiré. Cette section ne fait cependant qu'une description du modèle de redondance de réserve.

La redondance de réserve consiste à avoir un système secondaire identique au système primaire afin de remplacer ce dernier en cas de défaillance. La Figure 5.1 illustre donc le modèle de redondance de réserve. Il existe deux types de redondance de réserve soit la redondance de réserve à froid et la redondance de réserve à chaud.

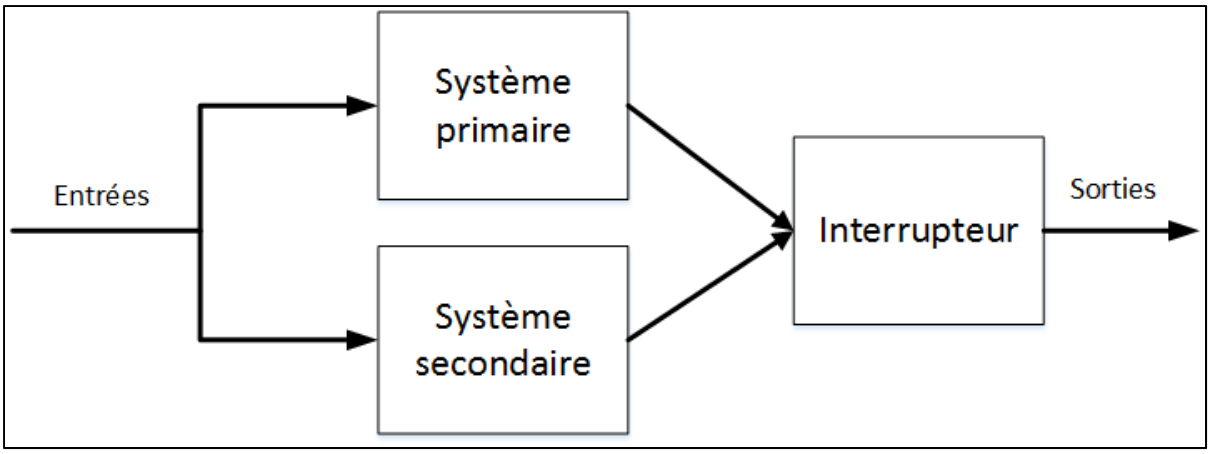


Figure 5.1 Modèle de redondance de réserve
Adaptée de Gohil, Basavalingarajaiah et Ramachandran (2011)

Dans la redondance de réserve à froid, le système secondaire n'est pas activé tant qu'il n'y a pas de faille du système primaire. L'avantage de cette approche est qu'il n'y a pas d'usure de la fiabilité du système secondaire et que ce dernier est protégé contre les dégâts causés par les surtensions. Le désavantage de cette approche est qu'il y a un long temps d'arrêt pour que le système secondaire s'active et remplace le fonctionnement du système primaire.

Dans la redondance de réserve à chaud, le système secondaire s'exécute en même temps que le système primaire et il y a généralement une synchronisation des événement entre les deux systèmes. L'avantage de cette approche est que le temps d'arrêt est très court et qu'il y a peu de perte de données. Le désavantage de cette approche est qu'étant donné que le système secondaire s'exécute en même temps que le système primaire, il va avoir usure de sa fiabilité. C'est ce type de redondance de réserve qui est utilisé dans le cadre de ce chapitre.

La police de propriétaire offerte par l'intergiciel DDS peut être utilisée pour faire de la redondance de réserve à chaud. Pour ce faire, la police de propriétaire des lecteurs doit être configurée à exclusif. De cette façon, les lecteurs ne peuvent recevoir les données que de l'écrivain qui a la plus grande force de propriétaire. La Figure 5.2 illustre un exemple de redondance de réserve à chaud en utilisant la police de propriétaire. Sur cette image, le lecteur à une police de propriétaire exclusif ce qui signifie qu'il va recevoir seulement les

sorties d'un seul écrivain. Le système primaire est muni d'un écrivain avec une force de propriétaire de 1 alors que le système secondaire est muni d'un écrivain avec une propriétaire de 0. De ce fait, si le lecteur est connecté aux deux écrivains, il va recevoir seulement les données qui proviennent du système primaire puisque ce dernier a une force de propriétaire supérieure. Cependant, si jamais il y a une déconnexion entre l'écrivain du système primaire et le lecteur, alors ce dernier va recevoir les données du système secondaire.

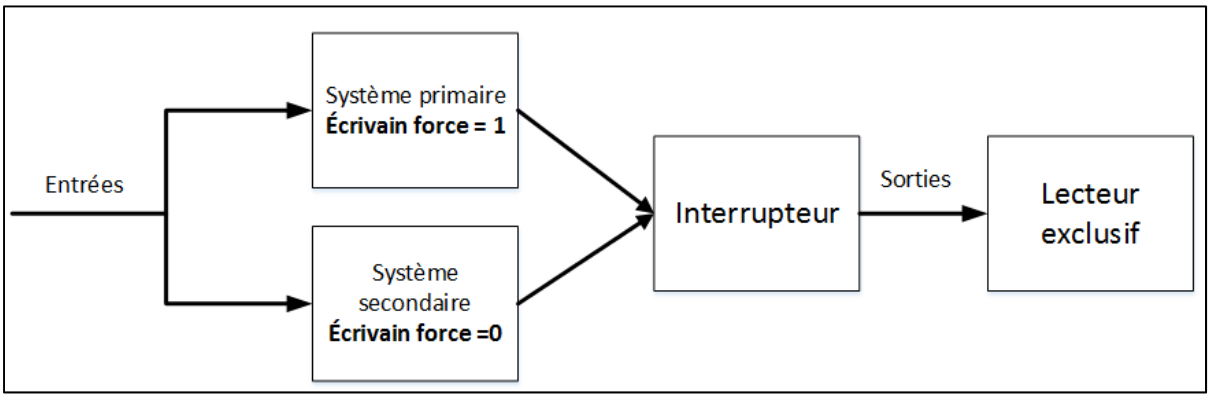


Figure 5.2 Exemple de redondance de réserve à chaud en utilisant un intergiciel DDS

5.2 Scénario de test

Dans cette section, un scénario de test avec une redondance de réserve à chaud à l'aide d'un intergiciel DDS est effectué. Le scénario consiste à avoir deux systèmes qui exécutent le code de l'AFCS conçu précédemment qui sont connecté au simulateur de vol X-Plane a travers un interrupteur réseau tel qu'illustré à la Figure 5.3. Normalement, les systèmes redondants devraient être identiques, mais dans notre cas, avoir deux PC avec des spécifications similaires sera suffisant pour faire une étude de cas. L'interrupteur réseau utilisé est un D-Link DGS-1008D. L'objectif de ce test est de mesurer la stabilité de l'aéronef lorsque le contrôle de ce dernier passe de l'AFCS primaire à l'AFCS secondaire.

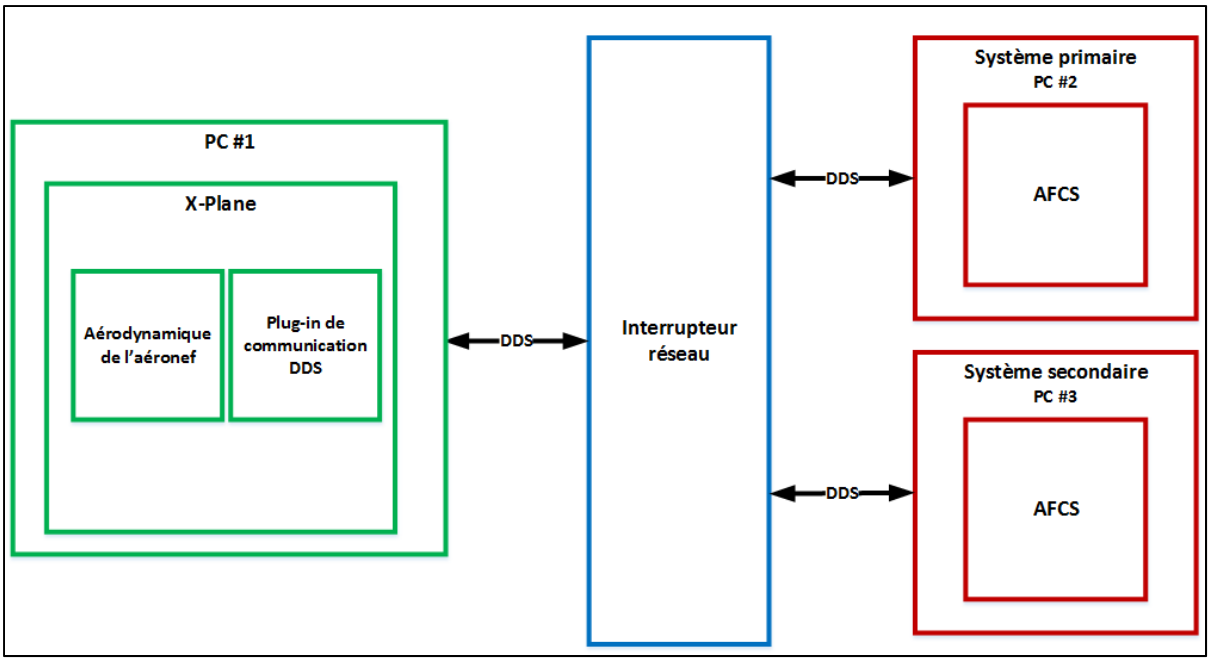


Figure 5.3 Scénario de test avec une redondance de réserve à chaud

L'AFCS qui s'exécute sur le PC #2 a été considéré comme étant le système primaire donc une force de propriétaire plus élevé que l'AFCS du PC #3 lui a été assignée. Les deux systèmes s'exécutent avec le système d'exploitation Windows 7. Le système primaire contrôle ainsi l'aéronef jusqu'au moment où le processus qui exécute l'AFCS est détruit a 6000 pieds afin de simuler un système défaillant. À partir de ce moment, c'est l'AFCS qui s'exécute sur le système secondaire (PC #3) qui contrôle maintenant l'aéronef.

5.3 Analyse des résultats

Les résultats obtenus sont illustrés à la Figure 5.4. Sur cette figure, les résultats du scénario de test du chapitre 4 avec DDS entre deux PC (Figure 4.17) ont également été tracés afin de comparer ces derniers aux résultats du scénario avec redondance.

Cette figure montre que la transition de contrôle de l'aéronef à 6000 pieds survient à 66.22 secondes. Cette transition n'est pas observable sur la courbe du contrôleur de roulis. De plus, cette figure illustre un temps de stabilisation du roulis similaire à ce qui a été obtenus dans le

scénario de tests avec DDS entre deux PC. L'erreur en régime permanant du roulis pour les deux scénarios de tests est de 0.07 degré.

La transition du contrôle de l'aéronef du système primaire au système secondaire n'est pas observable sur l'altitude de l'aéronef. De plus, le temps de stabilisation de l'altitude pour le scénario de redondance est de 99.62 secondes ce qui est très près de ce qui a été obtenus pour le scénario avec DDS entre deux PC soit de 98.38 secondes. L'erreur en régime permanant de l'altitude est la même pour les deux scénarios soit de 8 pieds.

La transition du contrôle de l'aéronef n'est pas observable sur la vitesse de l'aéronef. Le temps de stabilisation de la vitesse est presque identique pour les deux scénarios de test soit de 150.82 secondes pour scénario avec redondance et de 147.78 secondes pour le scénario avec DDS entre deux PC. De plus, l'erreur en régime permanant est la même pour les deux scénarios soit de 4 nœuds.

En somme, la redondance de réserve à chaud est possible à faire à l'aide de la police de propriétaire offerte par l'intergiciel DDS sans avoir un impact significatif sur le fonctionnement de l'application d'AFCS.

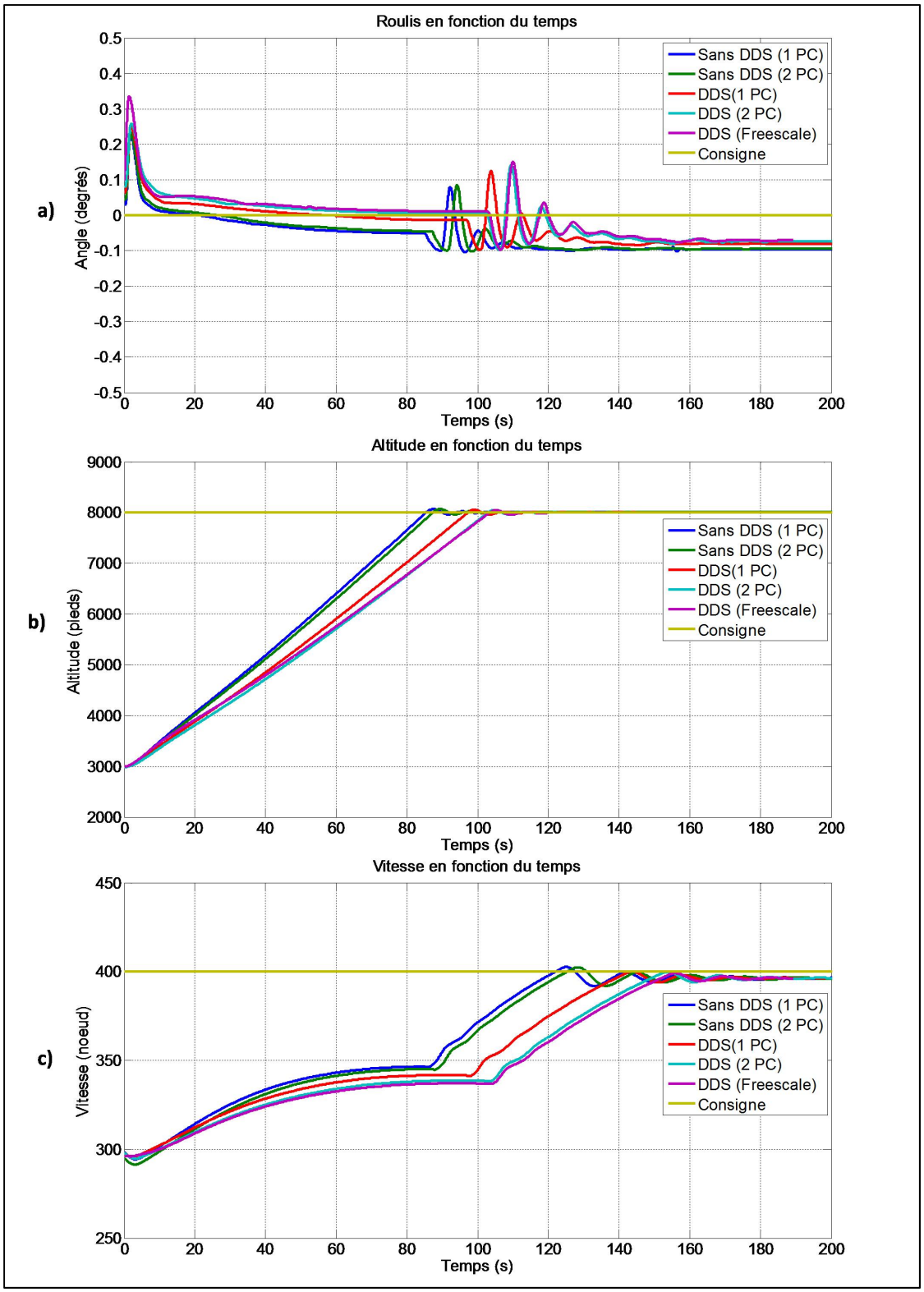


Figure 5.4 Résultats de l'étude de cas avec redondance de réserve à chaud
a) roulis, b) altitude, c) vitesse

5.4 Conclusion

Le but de chapitre était de mesurer les performances d'un système redondant qui a été conçu à l'aide d'un intergiciel DDS. De ce fait, un scénario de test en utilisant la police de propriétaire offerte par l'intergiciel DDS a été effectué afin de faire de la redondance de réserve à chaud. Le scénario comporte deux PC redondants qui exécutent l'AFCS conçu dans le chapitre 4 qui contrôlent un Boeing 747-400 simulé dans le logiciel X-Plane. Dans ce scénario, l'AFCS qui s'exécute sur le système primaire est détruit afin de simuler une faille du système primaire. Les résultats montrent que le transfert de contrôle de l'aéronef du système primaire au système secondaire n'est pas visible sur la stabilité de l'aéronef. En somme, l'utilisation d'un intergiciel DDS pour faire de la redondance de réserve à chaud est viable puisqu'elle permet le transfert de contrôle de l'aéronef du système primaire à un système secondaire sans avoir d'impact significatif sur la stabilité de l'aéronef.

CONCLUSION

Le but de ce travail de recherche était d'évaluer la technologie d'intergiciel de communication DDS pour son utilisation dans le domaine avionique. De ce fait, un retour sur les différentes contributions de ce travail de recherche sera effectué.

La première contribution de ce mémoire est d'identifier les différentes fonctionnalités de la norme DDS qui sont utiles au domaine avionique. Il a été mis en évidence dans le chapitre 2 que le paradigme de messagerie de publication/souscription centré sur les données utilisées par la norme DDS offre une grande flexibilité et facilite l'évolution de systèmes existants. De plus, les différentes polices de QoS offertes par la norme DDS permettent de configurer les communications avec une grande flexibilité afin d'obtenir le comportement désiré. Ces polices de QoS permettent entre autres de contrôler la priorité des communications ainsi que la mémoire utilisée par l'intergiciel DDS. Il a été vu que le processus de découverte utilisé par l'intergiciel DDS permet à différents interlocuteurs de communiquer ensemble sans avoir une connaissance approfondie des autres interlocuteurs ce qui simplifie la gestion des communications entre ces derniers. En somme, la norme DDS offre une grande flexibilité et simplicité d'utilisation qui est désirable dans le domaine avionique.

La deuxième contribution est de caractériser les performances des communications en utilisant un intergiciel DDS. De ce fait, le chapitre 3 présente différents scénarios de tests de latence et de bande passante en utilisant un intergiciel DDS. Les résultats des différents tests ont été comparés entre eux afin de mesurer l'impact de l'utilisation d'un intergiciel DDS sur la latence d'une communication. De plus, les résultats obtenus ont également été comparés à la latence généralement désirée pour les contrôles d'un aéronef de ligne. L'environnement de test utilisé est composé de PC avec le système d'exploitation Windows 7. Étant donné que cet environnement n'est pas temps réel, des valeurs extrémales ont été observées. Cependant, les valeurs extrémales n'ont pas été considérées dans les résultats puisque ces dernières n'auraient pas été présentes dans un environnement temps réel avionique. De plus, on a posé l'hypothèse que les performances observées dans un environnement avionique seraient

meilleures à celle observées dans l'environnement Windows 7. Les résultats présentent une faible augmentation de la latence moyenne causée par l'utilisation d'un intergiciel DDS. Cependant, la latence moyenne en utilisant un intergiciel DDS est bien en dessous de la latence désirée pour les contrôles d'un aéronef de ligne. Il a également été mis en évidence que l'intergiciel DDS arrive à utiliser une grande partie de la bande passante du lien de communication, donc qu'il n'est pas un goulot d'étranglement à l'utilisation de la bande passante disponible.

La troisième contribution est de mesurer l'impact de l'utilisation d'un intergiciel DDS sur les performances d'une application d'AFCS. Le chapitre 4 a donc porté sur la conception d'un AFCS et sur la mesure des performances de ce dernier. Afin de mesurer les performances de l'AFCS, ce dernier a été connecté avec le simulateur de vol X-Plane et les performances du contrôleur de roulis, d'altitude et de vitesse de l'AFCS ont été mesurées. En premier lieu, les performances de l'AFCS ont été mesurées sans utiliser d'interciel DDS, et ce, dans un environnement avec des PC. En deuxième lieu, les performances de l'AFCS ont été mesurées utilisant un intergiciel DDS encore sur des PC. Finalement, les performances de l'AFCS ont été mesurées de nouveau, mais sur une plateforme FreeScale avec le système d'exploitation temps réel VxWorks 6.9. Les résultats ont montré que l'utilisation d'un intergiciel DDS n'empêche pas l'AFCS de stabiliser l'aéronef au roulis, à l'altitude et à la vitesse désirée. Cependant, les résultats montrent également que l'utilisation de l'intergiciel DDS augmente le temps de stabilisation de l'aéronef. En somme, l'intergiciel DDS peut être utilisée pour une application avionique telle que l'AFCS, mais pourrait ne pas convenir pour les applications dont le temps de réponse est critique.

La quatrième contribution est une méthodologie de portage de modèles Simulink vers une plateforme FreeScale dans le but de faire de la simulation HIL. Dans le chapitre 4, les boucles de commandes de l'AFCS ont été conçues dans Simulink et ont été portées sur une plateforme FreeScale avec un système d'exploitation temps réel VxWorks 6.9. Les performances (temps de stabilisation et erreur en régime permanent) obtenues des boucles de commandes sur la plateforme FreeScale sont similaires à celles obtenues sur des PC avec le

système d'exploitation Windows 7. Il est donc possible d'en conclure que la méthodologie de portage utilisée est fonctionnelle.

Dans le chapitre 5, un scénario de test de redondance de réserve à chaud en utilisant la police de propriétaire de l'intergiciel DDS a été effectué. Les deux systèmes étaient des PC qui exécutent l'AFCS conçu dans le chapitre 4 afin de contrôler un Boeing 747-400 simulé dans le logiciel X-Plane. Les résultats montrent que la transition de contrôle de l'aéronef du système primaire au système secondaire n'a pas d'impact visible sur la stabilité de l'aéronef. En somme, l'intergiciel DDS est une technologie viable afin d'effectuer de la redondance de réserve à chaud.

Voici plusieurs recommandations pour des travaux futurs qui pourraient découler de ce travail de recherche.

En premier lieu, les tests de latence des communications avec un intergiciel DDS effectués dans ce travail de recherche utilisent un environnement avec des PC sur le système d'exploitation Windows 7. Or, il faudrait refaire les tests de latence dans un environnement avionique afin d'obtenir des résultats plus significatifs. Pour ce faire, un RTOS avionique tel que VxWorks 653 devrait être utilisé. Cela permettrait de faire une analyse du pire cas des communications DDS à travers la latence maximale. Les tests dans un environnement avionique ont été considérés dans le cadre de ce projet, mais le coût élevé du matériel avionique ainsi que des RTOS avionique ont limité nos tests a un environnement PC.

En deuxième lieu, les tests avec l'AFCS effectués dans ce travail de recherche sont intéressants, mais ne nous permettent pas de généraliser l'impact de l'utilisation d'un intergiciel DDS sur toutes les applications avioniques. De ce fait, il faudrait effectuer une analyse de l'impact de l'utilisation d'un intergiciel DDS sur les performances d'applications avioniques dont la latence est plus contraignante telles que des boucles de commande d'avions de chasse. Il faudrait également effectuer les mêmes analyses avec des applications

qui nécessitent une grande utilisation de la bande passante telle qu'une application de vision augmentée.

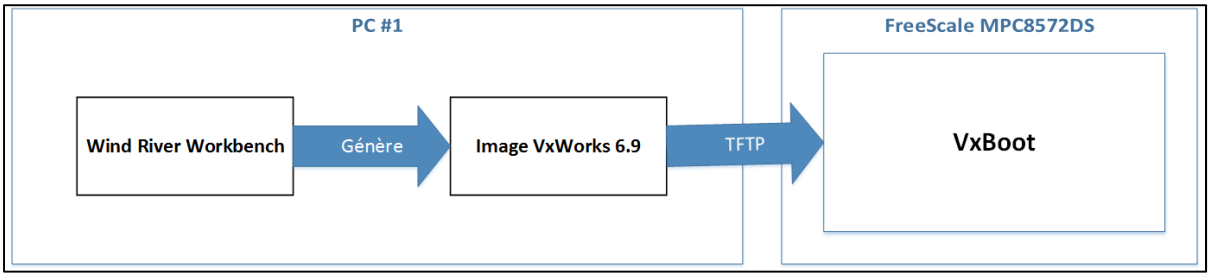
En troisième lieu, les tests dans ce projet de recherche se limitent à seulement deux ou trois nœuds dans un petit réseau avec peu de trafic. Or, il faudrait faire des tests avec un intergiciel DDS dans un réseau avec plusieurs nœuds et beaucoup de trafic afin d'évaluer les limites de l'intergiciel DDS. Il serait ainsi possible d'observer l'effet de la QoS de priorité de transport offerte par l'intergiciel DDS sur les performances d'applications de criticités différentes dans un réseau avec beaucoup de trafic.

Pour finir, les tests ont été effectués à l'aide du protocole UDP/IP sur des câbles Ethernet. Or, le domaine avionique utilise plutôt le bus de données AFDX, ARINC 429 et MIL-STD-1553. De ce fait, il faudrait implémenter ces bus de données au sein d'un intergiciel DDS afin de caractériser les performances de ce dernier avec les bus de données avionique couramment utilisés.

ANNEXE I

PROCESSUS DE CHARGEMENT DE VXWORKS 6.9

La figure ci-dessous illustre le processus de chargement de VxWorks 6.9 sur la plateforme FreeScale. Sur cette figure, on remarque que c'est l'IDE Wind River Workbench qui génère l'image VxWorks 6.9 qui doit être installée sur la plateforme FreeScale. De plus, cette figure illustre que le chargeur d'amorçage VxBoot à dû être installée sur la plateforme FreeScale et que ce dernier va se charger d'effectuer le téléchargement de l'image par le protocole TFTP (*Trivial File Transfert Protocol*) à travers un câble ethernet RJ45.



LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Al-Jarrah, M. A., S. Adiiansyah, Z. K. Marji et M. S. Chowdhury. 2011. « Autonomous aerial vehicles, guidance, control and signal processing platform ». In *Systems, Signals and Devices (SSD), 2011 8th International Multi-Conference on*. (22-25 March 2011), p. 1-17.

Alena, R. L., J. P. Ossenfort, K. I. Laws, A. Goforth et F. Figueroa. 2007. « Communications for Integrated Modular Avionics ». In *Aerospace Conference, 2007 IEEE*. (3-10 March 2007), p. 1-18.

Ang, Kiam Heong, Gregory Chong et Yun Li. 2005. « PID control system analysis, design, and technology ». *IEEE Transactions on Control Systems Technology,* vol. 13, n^{o} 4, p. 559-576.

Barros dos Santos, S. R., et N. M. F. de Oliveira. 2011. « Longitudinal autopilot controllers test platform hardware in the loop ». In *2011 IEEE International Systems Conference, 4-7 April 2011*. (Piscataway, NJ, USA), p. 8 pp. Coll. « 2011 IEEE International Systems Conference »: IEEE. < http://dx.doi.org/10.1109/SYSCON.2011.5929071 >.

Bellavista, P., A. Corradi, L. Foschini et A. Pernafini. 2013. « Data Distribution Service (DDS): a performance comparison of OpenSplice and RTI implementations ». In *2013 IEEE Symposium on Computers and Communications (ISCC), 7-10 July 2013*. (Piscataway, NJ, USA), p. 000377-83. Coll. « 2013 IEEE Symposium on Computers and Communications (ISCC) »: IEEE. < http://dx.doi.org/10.1109/ISCC.2013.6754976 >.

Bennett, James; Bhasin, Ajay; Grant, Jamila; Lim, Wen Chung. 2006. « PID Tuning Via Classical Methods ». < https://controls.engin.umich.edu/wiki/index.php/PIDTuning Classical >. Consulté le 24 mars 2014.

Bieber, P.; Bonial, F.; Boyer, M. ;Noulars, E.;Pagetti, C. 2012. « New Challenges for Future Avionic Architectures ». *AerospaceLab Journal*, n^{o} 4.

Bittar, Adriano, Neusa Maria Franco de Oliveira et Helosman Valente de Figueiredo. 2013. « Hardware-In-the-Loop Simulation with X-Plane of Attitude Control of a SUAV Exploring Atmospheric Conditions ». p. 1-17.

Boonma, Pruet, et Junichi Suzuki. 2009. « Self-configurable publish/subscribe middleware for wireless sensor networks ». In *Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference*. (Las Vegas, NV, USA), p. 1376-1383. 1700854: IEEE Press.

Calvo, I., F. Perez, I. Etxeberria et G. Moran. 2010. « Control communications with DDS using IEC61499 Service Interface Function Blocks ». In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. (13-16 Sept. 2010), p. 1-4.

Cam, H., O. K. Sahingoz et A. C. Sonmez. 2011. « Wireless sensor networks based on publish/subscribe messaging paradigms ». In *Advances in Grid and Pervasive Computing. 6th International Conference (GPC 2011), 11-13 May 2011*. (Berlin, Germany), p. 233-42. Coll. « Advances in Grid and Pervasive Computing. Proceedings of the 6th International Conference (GPC 2011) »: Springer. < http://dx.doi.org/10.1007/978-3-642-20754-9_24 >.

Cruz, Jesus Matinez, Adrian Romero-Garces, Juan Pedro Bandera Rubio, Rebeca Marfil Robles et Antonio Bandera Rubio. 2012. « A DDS-based middleware for quality-of-service and high-performance networked robotics ». *Concurr. Comput. : Pract. Exper.,* vol. 24, n^{o} 16, p. 1940-1952.

Detti, A., P. Loreti, N. Blefari-Melazzi et F. Fedi. 2010. « Streaming H.264 scalable video over data distribution service in a wireless environment ». In *2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 14-17 June 2010*. (Piscataway, NJ, USA), p. 3 pp. Coll. « 2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) »: IEEE. < http://dx.doi.org/10.1109/WOWMOM.2010.5534937 >.

FACE Consortium. 2013. *Technical Standard for Future Airborne Capability Environment (FACE), Edition 2.0*. The Open Group, 348 p. Consulté le 16 juin 2014.

FreeScale. 2014. « MPC8572DS ». < http://www.mouser.com/images/freescale/images/MPC8572DS.jpg >. Consulté le 16 octobre.

Fuchsen, R. 2009. « Preparing the next generation of IMA: A new technology for the scarlett program ». In *Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th*. (23-29 Oct. 2009), p. 7.B.5-1-7.B.5-8.

Fuchsen, R. 2010. « How to address certification for multi-core based IMA platforms: Current status and potential solutions ». In *2010 IEEE/AIAA 29th Digital Avionics Systems Conference (DASC), 3-7 Oct. 2010*. (Piscataway, NJ, USA), p. 11 pp. Coll. « 2010 IEEE/AIAA 29th Digital Avionics Systems Conference (DASC) »: IEEE. < http://dx.doi.org/10.1109/DASC.2010.5655461 >.

Garcia-Valls, M., I. Rodriguez-Lopez, L. Fernandez-Villar, I. Estevez-Ayres et P. Basanta-Val. 2010. « Towards a middleware architecture for deterministic reconfiguration of service-based networked applications ». In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. (13-16 Sept. 2010), p. 1-4.

Gohil, S., A. Basavalingarajaiah et V. Ramachandran. 2011. « Redundancy management and synchronization in avionics communication products ». In *Integrated Communications, Navigation and Surveilance Conference (ICNS), 2011*. (10-12 May 2011), p. C3-1-C3-8.

Howard, Courtney. 2013. « Embraer employs RTI Connext DDS in KC-390 mission-critical avionics software development ». *Avionics Intelligence*. < http://www.avionics-intelligence.com/articles/2013/04/Embraer-RTI.html >. Consulté le 14 mai 2014.

Jakovljevic, M. 2009. « Synchronous/asynchronous Ethernet networking for mixed criticality systems ». In *2009 IEEE/AIAA 28th Digital Avionics Systems Conference. DASC 2009, 23-29 Oct. 2009*. (Piscataway, NJ, USA), p. 1.E.3 (10 pp.). Coll. « 2009 IEEE/AIAA 28th Digital Avionics Systems Conference. DASC 2009 »: IEEE. < http://dx.doi.org/10.1109/DASC.2009.5347559 >.

Jinsong, Yang, K. Sandstrom, T. Nolte et M. Behnam. 2012. « Data Distribution Service for industrial automation ». In *2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA 2012), 17-21 Sept. 2012*. (Piscataway, NJ, USA), p. 8 pp. Coll. « Proceedings of the 2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA 2012) »: IEEE. < http://dx.doi.org /10.1109/ETFA.2012.6489544 >.

Johansen, Ingrid Hagen. 2012. « Autopilot Design for Unmanned Aerial Vehicles ». Norwegian University of Science and Technology, 86 p.

Jun, Jin, Han Junwei, Zheng Shupeng et He Jingfeng. 2009. « DDS based high fidelity flight simulator ». In *2009 WASE International Conference on Information Engineering (ICIE), 10-11 July 2009*. (Piscataway, NJ, USA) Vol. vol.1, p. 548-51. Coll. « 2009 WASE International Conference on Information Engineering (ICIE) »: IEEE. < http:// dx.doi.org/10.1109/ICIE.2009.61 >.

Kulkarni, A. M., V. S. Nayak et P. V. R. R. B. Rao. 2012. « Comparative study of middleware for C4I systems Web Services vis-à-vis Data Distribution Service ». In *Recent Advances in Computing and Software Systems (RACSS), 2012 International Conference on*. (25-27 April 2012), p. 305-310.

Lipinski, Erik, et L. Ebrecht. 2014. « Synthetic vision meets ARINC 661: feasibility study of the integration of terrain visualization in ARINC 661 avionic displays ». In. Vol. 9087, p. 90870K-90870K-11. < http://dx.doi.org/10.1117/12.2052907 >.

Littlefield-Lawwill, Justin, et Larry Kinnan. 2008. « System considerations for robust time and space partitioning in integrated modular avionics ». In *2008 IEEE/AIAA 27th Digital Avionics Systems Conference, DASC 2008, October 26, 2008 - October 30, 2008*. (St. Paul, MN, United states), p. 1B11-1B111. Coll. « AIAA/IEEE Digital

Avionics Systems Conference - Proceedings »: Institute of Electrical and Electronics Engineers Inc. < http://dx.doi.org/10.1109/DASC.2008.4702751 >.

Lopez-Jaquero, V., F. Montero, E. Navarro, A. Esparcia et J. A. Catal'n. 2012. « Supporting ARINC 653-based Dynamic Reconfiguration ». In *2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2012) & European Conference on Software Architecture (ECSA 2012), 20-24 Aug. 2012*. (Los Alamitos, CA, USA), p. 11-20. Coll. « 2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2012) European Conference on Software Architecture (ECSA 2012) »: IEEE Computer Society. < http://dx.doi.org/10.1109/WICSA-ECSA.212.9 >.

Manocha, Amit, et Abhishek Sharma. 2009. « Three axis aircraft autopilot control using genetic algorithms : An experimental study ». In *2009 IEEE International Advance Computing Conference, IACC 2009, March 6, 2009 - March 7, 2009*. (Patiala, India), p. 171-174. Coll. « 2009 IEEE International Advance Computing Conference, IACC 2009 »: Inst. of Elec. and Elec. Eng. Computer Society. < http://dx.doi.org /10.1109/IADCC.2009.4809001 >.

Microsoft TechNet. 2003. « What Is QoS? ». < http://technet.microsoft.com/en-us/library/cc757120(v=ws.10).aspx >. Consulté le 7 octobre 2014.

Obermaisser, R., et P. Peti. 2006. « A Fault Hypothesis for Integrated Architectures ». In *Intelligent Solutions in Embedded Systems, 2006 International Workshop on*. (30-30 June 2006), p. 1-18.

Object Management Group. 2005. *Real-time CORBA Specification*. 107 p. Consulté le 10 octobre 2014.

Object Management Group. 2007. *Data Distribution Service for Real-time Systems Version 1.2*. Object Management Group, 260 p. Consulté le 11-09-13.

Object Management Group. 2009. *The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification v.2.1*. Object Management Group, 216 p. < http://www.omg.org/spec/DDSI/2.1 >. Consulté le 3 avril 2014.

Prisaznuk, Paul J. 2008. « Arinc 653 role in Integrated Modular avionics (IMA) ». In *2008 IEEE/AIAA 27th Digital Avionics Systems Conference, DASC 2008, October 26, 2008 - October 30, 2008*. (St. Paul, MN, United states), p. 1E51-1E510. Coll. « AIAA/ IEEE Digital Avionics Systems Conference - Proceedings »: Institute of Electrical and Electronics Engineers Inc. < http://dx.doi.org/10.1109/DASC.2008.4702770 >.

Randy, Walter, et Watkins Chris. 2006. « Genesis Platform ». In *Avionics*. p. 12-1-12-28. Coll. « Electrical Engineering Handbook »: CRC Press. < http://dx.doi.org /10.1201/9780849384424.ch12 >. Consulté le 2014/06/23.

Real Time Innovation, Inc. 2012. *RTI Connext: Core Libraries and Utilities User's Manual*. U.S.A., 780 p.

Real Time Innovation, Inc. 2013. « At what OSI layers does RTI Connext operate? ». < http://community.rti.com/kb/what-osi-layers-does-rti-connext-operate >. Consulté le 31 mars.

Redakca HW Serveru. 2003. « Description of Basic Transfer Protocols used in Ethernet ». < http://hw-server.com/tcp-udp-protocol >. Consulté le 2 octobre 2014.

Sanchez, C., J. Nowotsch, M. Paulitsch et K. Schertler. 2013. « Image processing in airborne applications using multicore embedded computers ». In *Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd*. (5-10 Oct. 2013), p. 2B3-1-2B3-13.

Schmidt, D. C., M. Deshpande et C. O'Ryan. 2002. « Operating system performance in support of real-time middleware ». In *Object-Oriented Real-Time Dependable Systems, 2002. (WORDS 2002). Proceedings of the Seventh International Workshop on*. (2002), p. 199-206.

Schmidt, Douglas C., et Fred Kuhns. 2000. « Overview of the real-time CORBA specification ». *Computer,* vol. 33, n^{o} 6, p. 56-63.

Schuster, T., et D. Verma. 2008. « Networking concepts comparison for avionics architecture ». In *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*. (26-30 Oct. 2008), p. 1.D.1-1-1.D.1-11.

St. Clair, Bill. 2009. « Growing Complexity Drives Need for Emerging DO-178C Standard ». *COTS Journal*. p. 1. < http://www.cotsjournalonline.com/articles/view/101090 >. Consulté le 11 décembre 2014.

Tambe, Sumant, Fernando Garcia Aranda et Joe Schlesselman. 2013. « An Extensible Architecture for Avionics Sensor Health Assessment Using Data Distribution Service (Draft) ». In *AIAA Infotech@Aerospace 2013 Conference 19 - 22 August 2013*. (Boston, Massachusetts, United States), p. 11.

The Open Group. 2014. « About FACE Consortium ». < http://www.opengroup.org /face/about >. Consulté le 4 juillet.

Wang, Guoqing, Qingfan Gu, Miao Wang et Lihua Zhang. 2014. « Research on integrated avionics system safety ». In *2013 1st Symposium on Aviation Maintenance and Management, November 25, 2013 - November 28, 2013*. (Xi'an, China), VOL. 1 Vol. 296 LNEE, p. 555-568. Coll. « Lecture Notes in Electrical Engineering »: Springer Verlag. < http://dx.doi.org/10.1007/978-3-642-54236-7_61 >.

Wang, Kang, Shaoping Wang et Jian Shi. 2012. « Integrated reliability theory and evaluation methodology of AFDX ». In *IEEE 10th International Conference on Industrial Informatics, INDIN 2012, July 25, 2012 - July 27, 2012*. (Beijing, China), p. 657-662. Coll. « IEEE International Conference on Industrial Informatics (INDIN) »: Institute of Electrical and Electronics Engineers Inc. < http://dx.doi.org/10.1109 /INDIN.2012.6301218 >.

Watkins, C. B., et R. Walter. 2007. « Transitioning from federated avionics architectures to Integrated Modular Avionics ». In *Digital Avionics Systems Conference, 2007. DASC '07. IEEE/AIAA 26th*. (21-25 Oct. 2007), p. 2.A.1-1-2.A.1-10.

Wilson, A., et T. Preyssler. 2009. « Incremental certification and integrated modular avionics ». *IEEE Aerospace and Electronic Systems Magazine,* vol. 24, n^{o} 11, p. 10-15.

Woochul, Kang, K. Kapitanova et Son Sang Hyuk. 2012. « RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems ». *IEEE Transactions on Industrial Informatics,* vol. 8, n^{o} 2, p. 393-405.

Xiong, Ming, Jeff Parsons, James Edmondson, Hieu Nguyen et Douglas C Shmidt. 2006. « Evaluating the Performance of Publish/Subscribe Platforms for Information Management in Distributed Real-time and Embedded Systems ». p. 10. < http:// portals.omg.org/dds/sites/default/files/Evaluating_Performance_Publish_Subscribe_P latforms.pdf >.