# Conception, mise en oeuvre et évaluation d'un routeur embarqué pour l'avionique de nouvelle génération

Antoine Varet

Université de Toulouse

# THÈSE

**En vue de l'obtention du**

## DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :**

Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

---

**Présentée et soutenue par :**
**Antoine VARET**

**le** 01/10/2013

**Titre :**

Conception, Mise en OEuvre et Évaluation
d'un routeur embarqué
pour l'avionique de nouvelle génération

---

**École doctorale et discipline ou spécialité :**
EDSYS : Informatique 4200018

**Unité de recherche :**
Groupe RESCO, Laboratoire TELECOM, Ecole Nationale de l'Aviation Civile (ENAC)

**Directeur(s) de Thèse :**

Nicolas LARRIEU et Christophe MACABIAU
Enseignants-chercheurs à l'ENAC

**Jury :**

| | | |
|---|---|---|
| Francine KRIEF, | LaBRI, | Rapporteur |
| Elie NAJM, | ENST, | Rapporteur |
| Virginie WIELS, | ONERA, | Examinatrice |
| Eric FERON, | Georgia Tech, | Examinateur |
| Nicolas LARRIEU, | ENAC, | Directeur de thèse |
| Christophe MACABIAU, | ENAC, | Directeur de thèse |
| Gilles DESCARGUES, | THALES AVIONICS, | Invité |
| Xavier PLASSIN, | THALES AVIONICS, | Invité |

# Conception, Mise en Œuvre et Évaluation d'un routeur embarqué pour l'avionique de nouvelle génération

**Definition, Design, Implementation, Tests and Evaluation of an Embedded Router for a new generation of Avionic systems**

Antoine VARET

Septembre 2013

Institut National des Sciences Appliquées de Toulouse

# Résumé

Conception, Mise en Œuvre et Évaluation d'un routeur embarqué pour l'avionique de nouvelle génération

Le contexte aéronautique [1] a depuis plusieurs années mis en évidence le besoin croissant de technologies de sécurité permettant d'éviter des utilisations malveillantes des matériels ou services installés à bord des avions par les compagnies pour leurs usagers et leurs besoins propres.

Avec l'apparition prochaine d'un service d'accès à bord à l'Internet cabine pour le plus grand nombre, ce besoin de sécurisation va devenir une priorité. A l'heure actuelle, il n'existe pas de solution de sécurité [2] permettant d'une part de gérer ce nouveau type de trafic air-sol (appartenant à la famille de l'APC pour *Aeronautical Passenger Communication*) et d'autre part de l'intégrer parmi les autres types de trafic échangés entre l'avion et le sol (trafics AOC pour *Aeronautical Operational Control*, AAC pour *Aeronautical Administrative Communication* et ATSC pour *Air Traffic Services Communication par exemple*) tout en maximisant le niveau de robustesse offert.

La plupart des approches de sécurisation 'avion' se concentrent sur des méthodes et techniques permettant de sécuriser les échanges au sein de l'avion (réseau avionique de type AFDX par exemple) ou sur un lien dédié aux seules communications du contrôle aérien (flux ATSC principalement). Cette problématique, bien que nécessaire, ne suffit plus à l'heure où l'interconnexion du réseau avionique [3] avec le reste des réseaux de communication (réseau Internet par exemple) apparaît de jour en jour comme une étape incontournable. En effet, la demande des passagers pour accéder à leurs outils de travail (classiques pour les réseaux terrestres traditionnels) depuis leur siège en cabine devient de plus en plus pressante.

Le problème qui est abordé dans ce travail de thèse vise donc à proposer une architecture de sûreté [4] pour l'ensemble des communications aéronautiques qui viendra

1. L'aéronautique désigne les sciences et les technologies ayant pour but de construire et de faire évoluer un aéronef dans l'atmosphère terrestre.

2. Dans le contexte du transport aérien, la sécurité est la propriété d'innocuité du système, elle vise à protéger le système contre les défaillances et les pannes. Le terme anglais est ***safety***.

3. L'avionique est l'ensemble des équipements électroniques, électriques et informatiques des aéronefs. C'est donc un sous-ensemble du domaine aéronautique : l'avionique concerne uniquement l'intérieur de l'avion, tandis que l'aéronautique englobe le domaine avionique plus son environnement, incluant les installations terrestres de contrôle et de navigation.

4. Dans le contexte du transport aérien, la sûreté est la propriété d'immunité du système, elle

en complément de l'architecture de sécurité utilisée dans le réseau avionique et qui permettra de plus une interconnexion sécurisée entre le monde 'avion' et le monde extérieur (réseau Internet par exemple).

L'enjeu de ce travail est de prendre en compte les problématiques de standardisation adoptées dans le monde avionique, les intégrer comme prérequis et proposer une solution globale de sécurité intégrant le segment avion, le segment sol-bord et le segment sol. Cette solution ne peut avoir de pérennité scientifique et industrielle que si elle prend en compte l'ensemble des critères de sécurité qui caractérisent les divers environnements traversés et considère dès le départ les divers principes de standardisation retenus pour ces derniers.

La solution architecturale proposée dans ce travail de thèse repose principalement sur un composant central de routage, de filtrage et de sécurisation des flux de données aéronautiques. Le travail de conception et de développement de ce composant appelé **Routeur Sécurisé de Nouvelle Génération** (routeur SNG) a permis une validation expérimentale sur un système reproduisant un système embarqué.

Ce travail a permis la publication d'articles dans des conférences internationales aéronautiques et informatiques. La méthodologie de développement logicielle, élaborée initialement pour le routeur SNG, a été présentée à l'issue de la première année. Les résultats de son application au routeur ont fait l'objet d'une seconde publication, axée notamment sur le routage et le filtrage des flux de données.

La deuxième année de thèse a été marquée par un approfondissement de la sécurisation des flux de données, notamment par la modélisation des protocoles de sécurisation *Encapsulating Security Payload* (ESP) et de négociation de la sécurité *Internet Key Exchange* (IKE), ces protocoles étant des protocoles jusqu'à présent inexistants dans le contexte aéronautique, de par la complexité des contraintes imposées de sûreté.

De plus, un protocole de découverte des capacités de sécurité a été élaboré pour compléter l'ensemble des moyens techniques de sécurisation des communications. Ce protocole nommé *Security Capabilities* discovery *protocol Over Unsecured Topologies (SCOUT)* a fait l'objet de deux publications dans des conférences internationales.

La validation du logiciel du routeur sur émulateur puis sur un système embarqué réel et l'évaluation quantitative et qualitative des performances réseau de ce routeur constituent les travaux de la troisième année de thèse, permettant ainsi de compléter et finaliser le cycle de développement industriel auquel est adossée la thèse.

---

qualifie la capacité d'un système à gérer les menaces et dangers externes au système. Le terme anglais est ***security***.

# Abstract

Definition, Design, Implementation, Tests and Evaluation of an Embedded Router for a new generation of Avionic systems

For several years, security technology has become a growing necessity for the aeronautical world. It avoids unexpected and unauthorized access to the on-board services and the systems used by companies for their operational requirements.

With future Internet on-board services for all passengers, this necessity will be a priority. Nowadays, there is no safe and secure solution providing these new ground-board communications (called *Aeronautical Passenger Communicatio*n, APC) and enabling this network data to share the other already existing air-ground communication media with a high level of robustness (for example, the *Aeronautical Operational Control*, AOC, *Aeronautical Administrative Communication*, AAC and the *Air Traffic Services Communication*, ATSC).

Most current approaches are centred on methods and systems to secure data exchanges inside the plane (such as AFDX networks for instance) and to isolate air-ground communications on dedicated links (mostly for ATSC data flows). These approaches are mandatory but they are no longer sufficient to secure avionic data flows. The evolution from isolated to interconnected avionic networks and their opening to other data networks (the global Internet for instance) is more and more viewed as inescapable. Although passengers can easily access their business tools through the Internet in airports, they are now demanding this access when inside the airplane. This thesis explores a way of providing a security architecture for all aeronautical communications, in order to extend the existing safety architecture. This new architecture will enable airliners to interconnect "plane" networks and open them safely and securely with the outside world (for example the Internet).

One of the main requirements in our work is standardisation: the proposed solution must integrate standardisation constraints of avionic systems as a prerequisite, and provide a secure interconnection point between the on-board segment, the ground segment and the air-ground segment. In order to be industrially durable, the solution must consider all safety and security criteria and the associated standardisation requirements of the different environments interconnected together.

The architecture we propose in this thesis is mainly based on a central core component to route, filter and secure aeronautical data flows. We have called this system The **Secure New Generation Router** (SNG Router). The definition, the design and the implementation of this component are here validated on a pseudo-embedded system.

This work has lead us to present articles at international conferences for aeronautical and computer science. At the end of the first year of this thesis, we presented the software development methodology initially elaborated for the SNG router and usable for other classes of software. A second publication presented the results of its application to the SNG router, specifically for data network routing and filtering.

During the second year, we worked on the mechanisms of data security, through the modelling of the protocol *Encapsulating Security Payload* (ESP) to enforce the security of exchanged data, and of the protocol *Internet Key Exchange* (IKE) to negotiate security material. These two protocols are not currently used in the aeronautical world, because of the complexity of safety requirements.

Additionally, we designed, implemented and validated a new protocol to provide discovery of security capability between network nodes, in order to complete the set of technical mechanisms to supply secure communications. This protocol is named *Security Capabilities* discovery *protocol Over Unsecured Topologies (SCOUT)*. It has been presented in two conferences.

The third year of the thesis was dedicated to the validation of the software of the SNG Router on an emulator and then on a real embedded system, and to the quantitative and qualitative evaluation of the network efficiency of our SNG Router. This work completes and concludes the industrial development process involved in this research.

# Remerciements

Les travaux présentés dans ce mémoire ont été effectués à l'Ecole Nationale de l'Aviation de Civile (ENAC) de Toulouse au sain du laboratoire LEOPART (Laboratoire d'Etude et d'Optimisation des Architectures de Réseaux de Télécommunications), devenu ensuite le groupe de recherche RESCO (Réseaux de Communication) du laboratoire TELECOM.

Je tiens tout d'abord à remercier mes directeurs de thèse, Nicolas LARRIEU et Christophe MACABIAU, pour avoir accepté de croire en mes capacités, de m'avoir motivé et donné l'opportunité d'effectuer ma thèse dans d'excellentes conditions, thèse dont l'aboutissement est ce mémoire. Du tout début à l'échéance de ma thèse, leur encadrement, leur disponibilité et leur écoute attentive m'ont permis de m'épanouir durant ces trois années.

Je remercie aussi THALES AVIONICS, notre partenaire industriel, qui a rendu possible cette thèse. Je remercie notamment Xavier PLASSIN et Gilles DESCARGUES, ingénieurs de THALES AVIONICS Toulouse, pour leur soutien régulier ainsi que les nombreux retours d'expérience industrielle qui complètent naturellement les parties académiques de ces travaux.

Je suis également très reconnaissant pour le temps et le travail accordés par l'ensemble des membres du jury de soutenance de ma thèse. Je remercie spécialement Eric FERRON, examinateur, pour avoir présidé ce jury en cette belle matinée du premier octobre 2013, ainsi que Francine KRIEF et Elie NAJM pour avoir accepté d'être les rapporteurs de ma thèse et Virginie WIELS pour avoir accepté le rôle d'examinatrice.

Ce travail de thèse m'a offert l'opportunité d'encadrer des stagiaires, opportunité que j'ai saisie à quatre occasions. Je suis très reconnaissant à Julien MARCHAND, Hassna LOUADAH, Léo SARTRE et Benoît SAINCT pour les travaux qu'ils ont accomplis mais aussi pour les discussions techniques très intéressantes que nous avons eues et les crêpes que nous avons cuisinées puis mangées ensembles.

Je souhaite remercier l'ensemble de mes collègues de mon groupe de recherche à l'ENAC, à commencer par mes collègues doctorants Ons, Mickaël, Slim et Fred. Ces deux derniers ont été diplômés durant le déroulement de ma thèse. J'adresse un remerciement spécial à Slim qui m'a, de plus, co-encadré en PFE avec Nicolas pendant les six mois précédant le début de ma thèse ; leurs conseils avisés à tous les deux m'ont évité bien des écueils... Je souhaite aussi du courage à Ons qui commence actuellement sa troisième année de thèse et sera la suivante après moi à soutenir sa thèse dans notre groupe de recherche.

Je remercie aussi les enseignants-chercheurs permanents : Fabien qui m'a fait découvrir la joie d'une bonne bière au Mulligan's accompagnant des discussions souvent philosophiques et Alain dont la responsabilité de notre groupe de recherche est une tâche prenante. Enfin, je tiens à remercier les stagiaires du Resco/Leopart que je n'ai pas encadré mais avec qui nous avons passé d'excellents moments.

Je tiens aussi à remercier ma famille : ils m'ont en effet toujours accompagné et soutenu durant toutes mes études. J'adresse aussi un remerciement particulier à Dominique ROY, notamment pour m'avoir donné ces disquettes de 1.44 MB avec TP7 (Turbo Pascal 7, un compilateur commercialisé par Borland pour la programmation en langage Pascal).

Mes ami(e)s m'ont également accompagné(e)s durant toutes ces années : Samuel, Ludovic, David, Martial, Maxime, Matthias, Cécile, Julien, Jérémie, Silvia... À eux aussi je suis reconnaissant.

Je tiens enfin à remercier Cathy, Coco, Jean-Marie, Jean-Daniel et Christophe du département SINA de l'ENAC pour les discussions intéressantes que nous avons eues, ainsi qu'à Henri qui m'a transmis sa passion du judo et facilité l'opportunité de le pratiquer, en plus d'avoir contribué de manière significative à ma connaissance de l'application des réseaux dans le petit monde de l'aviation civile.

*« Ne fais jamais ce que tu ne peux défaire avant d'avoir réfléchi à ce que tu ne pourras plus faire une fois que tu l'auras fait. »*

*Subtil Loinvoyant, Roi des Six-Duchés*

L'Assassin royal, Tome 1 : L'apprenti assassin, *Robin Hobb*

# Table des matières

# Table des figures

# Liste des tableaux

# Liste des algorithmes

# 1. Introduction générale

## 1.1. Problématique

Le contexte aéronautique [1] a depuis plusieurs années mis en évidence le besoin croissant de technologies de sécurité permettant d'éviter des utilisations malveillantes des matériels ou services installés à bord des avions par les compagnies pour leurs usagers et leurs besoins propres.

Avec l'apparition prochaine d'un service d'accès à bord à l'Internet cabine pour le plus grand nombre, ce besoin de sécurisation va devenir une priorité. A l'heure actuelle, il n'existe pas de solution de sécurité [2] permettant d'une part de gérer ce nouveau type de trafic air-sol (appartenant à la famille de l'APC pour Aeronautical Passenger Communication) et d'autre part de l'intégrer parmi les autres types de trafic échangés entre l'avion et le sol (trafics AOC pour Aeronautical Operational Control, AAC pour Aeronautical Administrative Communication et ATSC pour Air Traffic Services Communication par exemple) tout en maximisant le niveau de robustesse offert.

La plupart des approches de sécurisation 'avion' se concentrent sur des méthodes et techniques permettant de sécuriser les échanges au sein de l'avion ou sur un lien dédié aux seules communications du contrôle aérien. Ainsi, le réseau avionique [3] embarqué de type AFDX est physiquement inaccessible aux passagers, cette ségrégation garantissant sa sûreté [4]. Concernant les flux ATC air-sol, ceux-ci exploitent des fréquences dédiées aux communications aéronautiques civiles, interdites par la loi à tout autre usage.

Cette problématique de sécurisation, bien que nécessaire, ne suffit plus à l'heure où l'interconnexion du réseau avionique avec le reste des réseaux de communication tels

1. L'aéronautique désigne les sciences et les technologies ayant pour but de construire et de faire évoluer un aéronef dans l'atmosphère terrestre.

2. Dans le contexte du transport aérien, la sécurité est la propriété d'innocuité du système, elle vise à protéger le système contre les défaillances et les pannes. Le terme anglais est ***safety***.

3. L'avionique est l'ensemble des équipements électroniques, électriques et informatiques des aéronefs. C'est donc un sous-ensemble du domaine aéronautique : l'avionique concerne uniquement l'intérieur de l'avion, tandis que l'aéronautique englobe le domaine avionique plus son environnement, incluant les installations terrestres de contrôle et de navigation.

4. Dans le contexte du transport aérien, la sûreté est la propriété d'immunité du système, elle qualifie la capacité d'un système à gérer les menaces et dangers externes au système. Le terme anglais est ***security***.

que le réseau Internet apparaît de jour en jour comme une étape incontournable. En effet, la demande des passagers pour accéder à leurs outils de travail depuis leur siège en cabine devient de plus en plus pressante, ceux-ci étant aujourd'hui habitués à disposer d'accès terrestres à l'Internet omniprésents.

Le problème qui est abordé dans ce travail de thèse vise donc à proposer une architecture de sûreté pour l'ensemble des communications aéronautiques qui viendra en complément de l'architecture de sécurité utilisée dans le réseau avionique et qui permettra de plus une interconnexion sécurisée entre le monde 'avion' et le monde extérieur (réseau Internet par exemple).

L'un des enjeux de ce travail est de prendre en compte les problématiques de standardisation adoptées dans le monde avionique, les intégrer comme prérequis et proposer une solution globale de sécurité intégrant le segment avion, le segment sol-bord et le segment sol.

En effet, ce travail de thèse ne peut avoir de pérennité scientifique et industrielle que s'il prend en compte l'ensemble des critères de sûreté et de sécurité qui caractérisent les divers environnements traversés et considère dès le départ les divers principes de standardisation retenus pour ces derniers.

La solution architecturale proposée dans ce travail de thèse repose principalement sur un composant central de routage, de filtrage et de sécurisation des flux de données aéronautiques. Le travail de conception et de développement de ce composant appelé Routeur Sécurisé de Nouvelle Génération (**routeur SNG**) a permis une validation expérimentale sur un système reproduisant un système embarqué dans un environnement reproduisant un cadre réaliste d'utilisation du routeur SNG.

Ainsi, ces travaux de recherche s'inscrivent non seulement dans la preuve d'exploitabilité de nouveaux concepts mais aussi dans la validation en laboratoire du composant produit. Cela représente donc un travail allant des TRL 2 à 4 (TRL=«Technology Readiness Level», ou en français «niveau de maturité technologique», qui est une mesure d'évaluation de projets, créée par la NASA [Mankins 1995]).

## 1.2. Contributions

### 1.2.1. Contributions et publications scientifiques

Ce travail a permis la publication d'articles dans des conférences internationales aéronautiques et informatiques.

La complexité des développements de systèmes aéronautiques embarqués nous a conduit à un travail préliminaire sur le processus de développement qui a abouti à la création d'une méthodologie. Cette méthodologie de développement logicielle, élaborée initialement pour le routeur SNG, a été présentée à l'issue de la première

année [Varet & Larrieu 2011]. Cette nouvelle méthodologie de développement de logiciel avionique et de prototypage rapide prend en compte dès l'établissement du cahier des charges les contraintes de sûreté et de sécurité associées au système à mettre en œuvre. Elle est l'objet du chapitre 3.

Notre méthodologie de développement du routeur SNG est basée sur la modélisation des fonctionnalités, la transformation automatisée de modèles en code et l'utilisation de systèmes d'exploitation critiques mettant en œuvre la virtualisation. Les modèles présentent en effet des avantages non négligeables en terme de rapidité et de réutilisabilité du développement, ainsi qu'en vérification et en validation des systèmes modélisés. La transformation automatisée contribue à garantir la sécurité du code généré et donc du logiciel final. Les systèmes d'exploitation critiques sont une brique essentielle de la sûreté de l'exécution du logiciel généré avec notre méthodologie.

Les résultats de l'application de la méthodologie au routeur SNG ont fait l'objet d'une seconde publication, axée notamment sur le routage et le filtrage des flux de données [Varet *et al.* 2012]. Cette application met en œuvre les différentes fonctionnalités énumérées au chapitre 2, elle est détaillée et complétée dans le chapitre 4.

La deuxième année de thèse a été marquée par un approfondissement de la sécurisation des flux de données, notamment par la modélisation des protocoles de sécurisation Encapsulating Security Payload (ESP) et de négociation de la sécurité Internet Key Exchange (IKE), ces protocoles étant des protocoles jusqu'à présent inexistants dans le contexte aéronautique, de par la complexité des contraintes imposées de sûreté de fonctionnement.

En plus des fonctionnalités de routage et de filtrage, nous avons aussi modélisé les mécanismes de traduction d'adresse entre les couches liaison et réseau (ce sont respectivement les couches 2 et 3 du modèle Open Systems Interconnection, OSI [ISO7498-1 1994]) tels que les protocoles ARP pour IPv4 et NDP pour IPv6, les fonctionnalités de négociation de la sécurisation des données et les mécanismes de sécurisation de ces données suite à la négociation.

Ces modélisations permettent non seulement de vérifier formellement certaines propriétés sur les protocoles modélisés, mais elles permettent aussi de générer automatiquement un code source mettant en œuvre les protocoles. Ces modélisations sont décrites et commentées dans le chapitre 4. À notre connaissance, nous sommes les premiers à avoir utilisé des modèles, formellement vérifiables, pour mettre en œuvre ces protocoles via une transformation modèle vers code source.

En parallèle du travail de conception des modèles, il est apparu un besoin complémentaire au routeur SNG : le besoin d'identifier et de sélectionner un protocole pour établir les canaux sécurisant les données. Ce protocole de découverte des capacités de sécurité a été élaboré pour compléter l'ensemble des moyens techniques de sécurisation des communications. Ce protocole nommé SCOUT (« Security Capabilities discovery protocol Over Unsecured network Topologies») a fait l'objet de deux publications [Varet & Larrieu 2012a, Varet & Larrieu 2012b] ainsi que d'une

mise en œuvre pour les systèmes basés sur Linux. Ce protocole SCOUT est l'objet du chapitre 5.

La validation du logiciel du routeur sur émulateur puis sur un système embarqué réel et l'évaluation quantitative et qualitative des performances réseaux de ce routeur constituent les travaux principaux de la troisième année de thèse, permettant ainsi de compléter et finaliser le cycle de développement industriel auquel est adossé la thèse. Ces travaux sont l'objet du chapitre 6.

Cette troisième année de thèse a été complétée par la publication de [Varet & Larrieu 2013], relative à l'évaluation quantitative de la qualité d'un code source, aidée par l'outil logiciel «Metrix» dont nous parlerons dans la section 3.4 et appliquée au cas du routeur SNG.

### 1.2.2. Contributions industrielles

Les fonctionnalités attendues par le routeur SNG ont fait l'objet de nombreuses discussions avec notre partenaire industriel, THALES AVIONICS. Deux documents principaux ont été produits, avec pour objectif de faciliter la phase d'industrialisation du routeur SNG.

Le premier est un document décrivant de manière formelle l'ensemble des fonctionnalités techniques du logiciel du routeur SNG et des contraintes fonctionnelles et non fonctionnelles s'y ajoutant. Appelé «Software Requirement Specifications» (SRS), nous avons intégralement rédigé ce document conformément aux standards de qualité internes à l'entreprise THALES AVIONICS. Nous avons rédigé ce document à titre indicatif, pour permettre de faciliter les discussions et d'établir un cadre clair et cohérent pour le routeur SNG.

Notre routeur étant une brique importante d'un système d'information, nous avons parallèlement rédigé un «Profil de Protection» adapté au routeur SNG dans un environnement aéronautique. Ce second document, présent dans l'annexe A, formalise les contraintes spécifiques à la sûreté du produit, conformément à la norme des «Critères Communs» décrite plus en détail dans la section 2.5.

### 1.2.3. Contributions logicielles

Le dernier groupe de contributions concerne les produits logiciels issus de nos travaux.

Le programme principal est le logiciel du routeur SNG. Sa structure et sa mise en œuvre sont amplement détaillés dans les chapitres 3 et 4, ses performances sont mesurées et discutées dans le chapitre 6.

Parallèlement à ce logiciel dont le développement et la mise au point se sont déroulées sur les trois années de thèse, trois autres logiciels ont été développés puis tous ont été

publiés sous licence libre (GPLv3[(FSF) 2007, Smith 2013]). Nous avons choisi de les intégrer aux contributions car bien que développés initialement pour compléter certains de nos travaux de recherche, ces outils sont disponibles librement et gratuitement et ont été conçus pour être réutilisés dans d'autres projets de recherche, d'enseignement ou tout autre usage permis par la licence de diffusion.

Le premier met en œuvre le protocole SCOUT6 qui est l'objet du chapitre 5. Il permet de découvrir les possibilités de sécurisation entre les nœuds d'un réseau (par exemple d'hôtes connectés par l'Internet) et d'établir des canaux sécurisés entre ces nœuds IPv6.

Le second logiciel est appelé «Metrix». Son rôle est d'évaluer quantitativement la qualité de code sources. La section 3.4 détaille les métriques mesurées, les langages employés et est illustrée par l'application de Metrix au code source en C du routeur SNG.

Le troisième logiciel que nous avons développé permet de générer un trafic réaliste de données sur les réseaux, i.e. des données dont l'analyse statistique correspond aux études scientifiques sur les données de réseaux locaux et nationaux. Ce logiciel a été nommé «sourcesonoff» en raison du concept des sources ON/OFF [Leland W. E. & V. 1994, Willinger W. & V. 1997] sur lequel il est basé. Il a été employé pour générer un trafic de données aéronautiques réalistes pour l'évaluation des performances du routeur SNG dans le chapitre 6.

## 1.3. Structure du mémoire

Ce mémoire de thèse est organisé en cinq chapitres principaux. Les trois premiers concernent majoritairement le domaine du génie logiciel tandis que les deux suivants se rapprochent plus du domaine des réseaux informatiques.

Le premier chapitre présente les différents domaines scientifiques et industriels abordés dans le cadre d'études de ma thèse : les réseaux de communication embarqués à bord des avions ainsi qu'une introduction aux moyens de communications sol-bord, c'est-à-dire les systèmes permettant à l'avion et à son personnel de communiquer avec les équipes et équipements au sol.

Dans le second chapitre sont abordées la sûreté et la sécurité ainsi que les méthodes pour les garantir dans le cas des systèmes avioniques. Ce chapitre formalise les besoins énumérés pour le développement de notre routeur SNG.

Le troisième chapitre traite de la méthodologie élaborée et appliquée pour le développement du routeur SNG. En effet, les contraintes imposées pour garantir l'innocuité et l'immunité des systèmes sont nombreuses et restreignent fortement la créativité des développeurs. Elles représentent aujourd'hui la majorité des coûts de développement des logiciels avioniques.

La méthodologie que nous proposons dans ce mémoire de thèse permet de s'affranchir d'une partie de ces contraintes, en garantissant certaines propriétés de sécurité et de sûreté, et ainsi de réduire le temps et les coûts de développement. Le chapitre explique les concepts de systèmes IMA et de systèmes MILS, puis les intègre à la méthodologie pour fabriquer des logiciels avec des niveaux élevés de sûreté et de sécurité.

Les sept étapes abstraites de la méthodologie sont complétées par une énumération non exhaustive d'outils logiciels permettant de réaliser ces étapes, les outils utilisés dans le routeur SNG étant décrits et justifiés. L'énumération ne peut être exhaustive car la méthodologie a été conçue de manière générique, pour s'adapter aux outils existants et futurs tout en garantissant un développement rapide et efficace de logiciel embarqué.

Ce chapitre 3 est complété par un outil logiciel que nous avons développé pour étudier la qualité d'un code source. Cet outil nommé «Métrix» a été appliqué au code autogénéré du routeur SNG.

Par la suite, le chapitre 4 détaille l'architecture mise en œuvre pour le routeur. Contrairement au chapitre précédent assez abstrait où l'accent était mis sur les méthodes, génériques par nature, ce chapitre est appliqué au cas du routeur et détaille les fonctionnalités mises en œuvre ainsi que les solutions algorithmiques utilisées. Il est organisé en quatre sections représentant la décomposition fonctionnelle choisie pour le routeur SNG. Ainsi, nous détaillons les modèles de routage et de filtrage pour IPv4 et IPv6 que nous avons élaboré pour le routeur SNG, puis nous exposons les modélisations des protocoles NDP, IKEv2 et ESP de manière approfondie. Enfin, nous terminons par la partie permettant de lier le code «haut niveau» issu des modèles avec le code «bas niveau» d'accès aux systèmes matériels.

Le cinquième chapitre est dédié à un travail parallèle et complémentaire au routeur. Durant le processus d'élaboration du routeur, la question de la sécurisation des données a été posée, ainsi que celle du niveau protocolaire à laquelle appliquer les mécanismes de sécurisation des données.

En effet, la mise en œuvre de mécanismes de niveau réseau est soumise à des contraintes très différentes de celles qui s'appliquent au niveau applicatif. De plus, il faut tenir compte de l'existant, souvent non sécurisé, parfois impossible à sécuriser à la source lorsque le coût de remplacement des systèmes existants est trop élevé.

Le protocole SCOUT a donc été conçu et validé, avec expérimentations sur systèmes réels non simulés non émulés, afin de garantir qu'au moins une couche protocolaire assure la sécurisation des données transportées et d'activer cette sécurisation le cas échéant.

Le chapitre 6 contient l'étude des performances du routeur SNG : le plan de tests et de validation y est présenté ainsi que les résultats des mesures expérimentales, leur analyse et leur interprétation. Cette évaluation apporte une réponse au problème d'utilisabilité du routeur SNG dans le monde aéronautique du point de vue performance.

Le dernier chapitre conclue ce mémoire ainsi que ma thèse. Il résume le travail que j'ai effectué ainsi que mes contributions. Il est suivi d'un ensemble de perspectives pouvant permettre de compléter certaines problématiques restées ouvertes et d'explorer de nouvelles idées prometteuses.

Les annexes complètent ces chapitres, notamment avec le profil de protection du routeur SNG (discuté dans la section 2.5). La documentation succincte expliquant l'utilisation du dæmon SCOUT6 que nous avons développé est aussi présente en annexe.

L'ensemble de mes contributions est synthétisée dans la figure 1.1. La figure 1.2 montre l'aspect multidisplinaire de nos travaux : ceux-ci s'inscrivent à la fois à l'intersection des domaines de la sécurité des réseaux, du génie logiciel, de l'ingénierie système, des systèmes embarqués et du domaine aéronautique.

### Note sur le déroulement de ma thèse

Ma thèse s'est déroulée à l'École Nationale de l'Aviation Civile (ENAC) de Toulouse, un établissement public administratif dépendant de la Direction Générale de l'Aviation Civile (DGAC), elle-même rattachée au Ministère de l'Écologie, du Développement durable, des Transports et du Logement de l'État Français.

Ma thèse a été menée en partenariat avec l'entreprise THALES AVIONICS Toulouse, grand équipementier fournisseur d'équipements avioniques à l'avionneur Airbus Industries, notamment dans le domaine des calculateurs de bord.

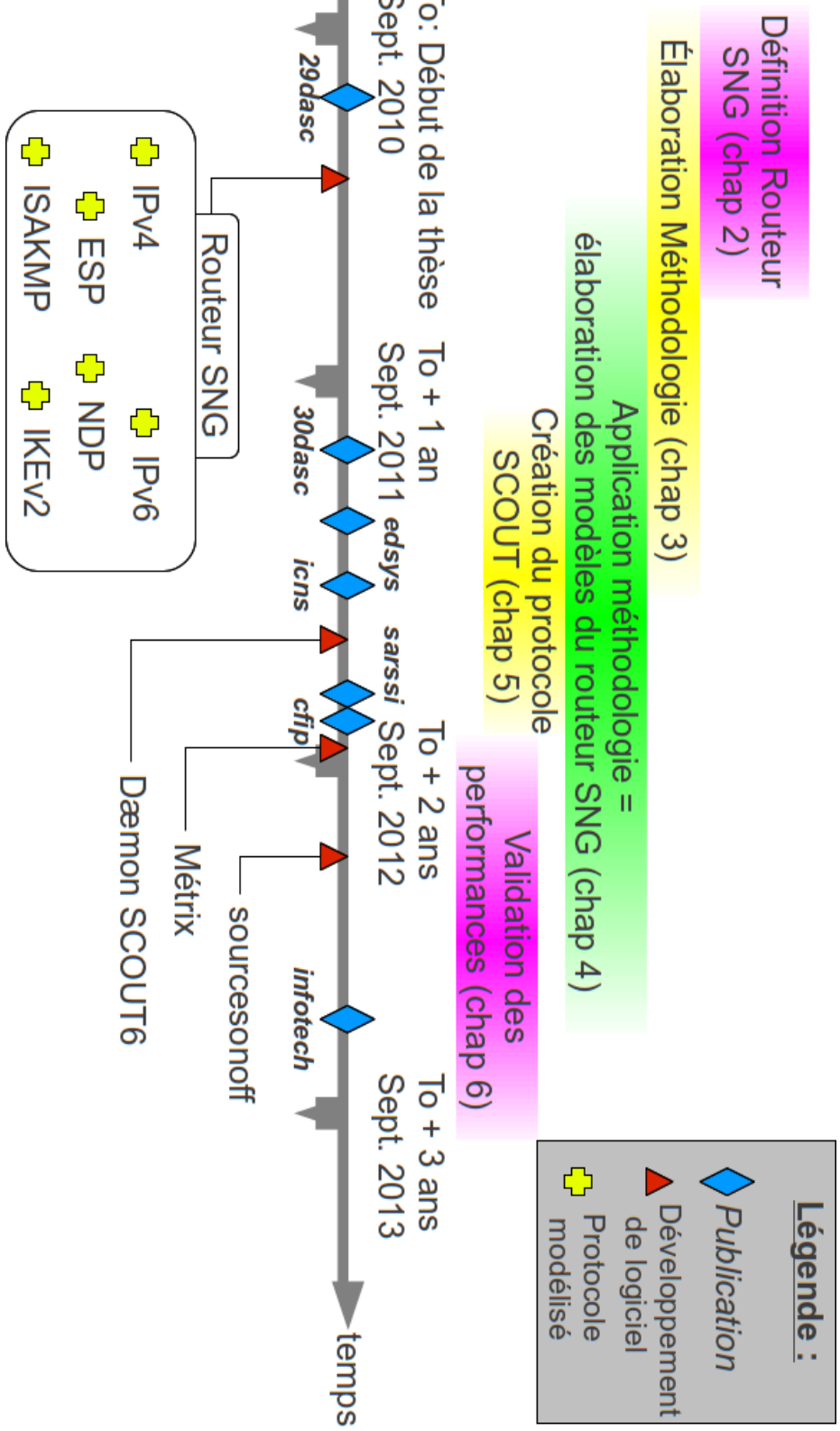



**Figure 1.1.:** Déroulement de la thèse, avec la répartition des contributions

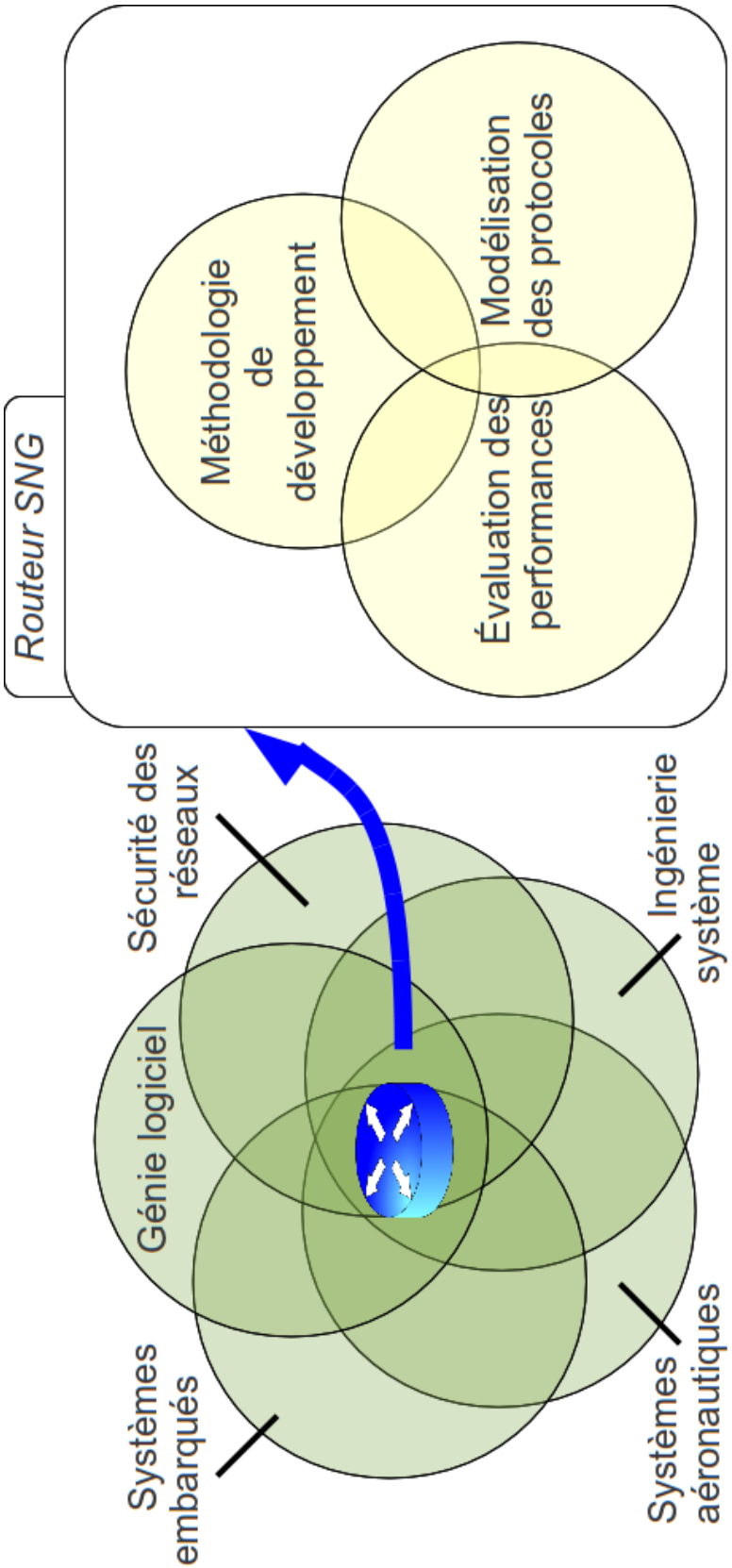



**Figure 1.2.:** Différents domaines scientifiques sont liés au routeur SNG

# 2. Contexte scientifique d'élaboration pour un routeur embarqué de nouvelle génération

Dans ce chapitre, le routeur Sécurisé de Nouvelle Génération (SNG) est présenté : après avoir défini les réseaux avioniques et aéronautiques, notamment les couches liaison et réseau liées à l'élaboration de notre système, nous exposons les besoins ayant conduit au lancement de la thèse puis détaillons les exigences de développement du routeur SNG. Un accent particulier est mis sur les phases de certification et d'évaluation de la sécurité et de la sûreté du routeur SNG.

## 2.1. Introduction aux réseaux avioniques et aéronautiques par la couche liaison

Les réseaux avioniques ont beaucoup évolués depuis les premières interconnexions d'équipements électroniques embarqués formant des réseaux de communication de données en 1968 ([DO-136 1968]). Ils se sont progressivement peuplés avec toujours plus d'équipements utilisateurs de l'infrastructure de communication. Les standards ont évolué pour tenir compte de l'augmentation de l'utilisation des réseaux en apportant simultanément des débits de plus en plus élevés. La sûreté de fonctionnement est cependant toujours restée une contrainte forte, garantissant ainsi une robustesse très élevée des moyens de communication. Dans cette section nous énumérerons les principaux standards mis en œuvre à l'intérieur des avions commerciaux de transport de passagers de masse. Ensuite les moyens de communication sol-bord seront présentés.

### 2.1.1. Les standards et protocoles de communication embarqués

Les premiers équipements embarqués dans les avions étaient connectés en point-à-point, c'est-à-dire qu'une liaison unique reliait exactement deux systèmes. Les premiers réseaux avioniques mettant en œuvre plus de deux équipements sont apparus en 1968 avec le standard DO-136 [DO-136 1968] qui définit le format des premières communications air-sol de données.

En avril 1978 fut publiée la norme[1] ARINC 429 en quatre parties [ARINC429P1-18 2012, ARINC429P2-16 2004, ARINC429P3-19 2009, ARINC429P4 2012]. Cette norme propose une technologie qui permet de relier un émetteur de données à des récepteurs (de 1 à 20) et permet des échanges à une vitesse de transmission de 12.5 kbps ou de 100 kbps. Les données sont encapsulées dans des doubles mots (32 bits, entête comprise). Sa simplicité, sa fiabilité et son déterminisme en ont fait l'un des protocoles les plus employés en avionique. Cependant, il est nécessaire d'installer un bus de données par émetteur, il est monodirectionnel, il est limité à 20 récepteurs et le débit est aujourd'hui jugé insuffisant pour les nouveaux usages qui sont de plus en plus gourmants en terme de débits.

Ainsi d'autres protocoles de communication furent élaborés pour pallier ces limitations. Le standard [MIL1553 1975] et sa dernière version MIL-STD-1553B [MIL1553B 1978] furent développés entre 1968 et septembre 1978 par l'armée américaine pour l'avion de combat F-16 Falcon[MIL1553w 2013]. Ils proposent un bus série asynchrone bidirectionnel géré par un contrôleur et interconnectant jusqu'à 31 terminaux émetteurs de données ainsi que des moniteurs (seulement récepteurs). La vitesse de transmission est de 1 Mbps, donc supérieure à celles de l'ARINC 429, diffusé dans la même période, mais le MIL-STD-1553 est plus complexe électroniquement et plus coûteux à mettre en œuvre. Publié en 1999, l'ARINC 629 [ARINC629-1 1999, ARINC629-2 1999] définit un bus bidirectionnel développé pour le gros porteur civil commercial Boeing 777 [Boeing 2013]. Il permet d'atteindre jusqu'à 2 Mbps partagés par un maximum de 120 terminaux.

La norme ARINC 664 a été publiée en 2006 en plusieurs parties : [ARINC664P1-1 2006, ARINC664P2-2 2009, ARINC664P3-2 2009, ARINC664P4-2 2007, ARINC664P5 2005, ARINC664P7-1 2009, ARINC664P8-1 2010]. Cette norme définit une architecture de communication moderne avec des débits de l'ordre de 100 Mbps. Elle permet d'interconnecter des systèmes avioniques via des topologies réseaux complexes et offre aux applications embarquées la possibilité d'échanger des données via notamment les protocoles UDP et IPv4 aux couches 4 (couche transport du modèle OSI [ISO7498-1 1994]) et 3 (couche réseau), protocoles présentés plus loin dans la section 2.3.1.

Sa partie 7 définit l'"Avionic Full-DupleX switched Ethernet" (AFDX [ARINC664P7-1 2009]), un protocole de niveau 2 (couche liaison), qui fut développé initialement par la communauté aéronautique pour l'avionneur Airbus Industries dans les années 2000. Basé sur le protocole Ethernet (IEEE 802.3), l'AFDX reste compatible avec les matériels Ethernet, courants aujourd'hui dans le domaine "grand public" et non aéronautique. Cela permet notamment de réduire fortement les coûts d'étude et de développement pour les industriels travaillant

1. Un standard est un référentiel public publié par une entité privée tandis qu'une norme est un référentiel public résultant d'un consensus. L'ARINC est à la fois éditeur de standards et de normes, étant donné qu'elle participe et organise les comités de normalisation des solutions qu'elle propose.

| Standard | Nombre maximal d'équipement | Débits* |
|---|---|---|
| ARINC 429 | 1 émetteur, 20 receveurs | 100 kbps |
| MIL-STD-1553(B) | 31 émetteurs/récepteurs, $\infty$ récepteurs | 1 Mbps |
| ARINC 629 | 120 émetteurs/récepteurs | 2 Mbps |
| ARINC 664 / AFDX | 65 536 émetteurs/récepteurs | 100 Mbps |

*Le débit indiqué est un maximum, partagé par tous les équipements

**Table 2.1.:** Protocoles avioniques embarqués de liaisons

avec l'AFDX : par exemple, ils peuvent utiliser des commutateurs Ethernet classiques en laboratoire pour développer et tester les systèmes terminaux destinés à communiquer à terme avec de l'AFDX.

Offrant un débit de 100 Mbps bidirectionnel (Full-Duplex), il contraint temporellement les trames de données à transiter à des créneaux horaires prédéterminés, ce qui permet de garantir la disponibilité du réseau pour les applications et de qualifier de déterministe les réseaux AFDX. Le succès de cette solution est visible par son utilisation par les avions gros porteurs commerciaux : Airbus A380, Boeing B787, Airbus A400M.

Les avions exploitent simultanément plusieurs de ces protocoles de communication, en plus de la redondance pour chacun des protocoles. Ainsi, l'A380 dispose de deux réseaux AFDX similaires (nommés "A" et "B") pour les commandes de vol afin que la perte d'un réseau soit immédiatement secourue par le second. Il dispose en plus de liens de secours ARINC 429 au cas où la défaillance serait inhérente à la technologie AFDX et affecterait donc simultanément les deux réseaux. La table 2.1 résume les technologies présentées dans cette section.

### 2.1.2. Les standards et protocoles de communication entre l'avion et le sol

Les réseaux embarqués permettent principalement de connecter les équipements de bords entre eux, mais aussi d'acheminer les données issues des réseaux terrestres. La liaison entre les réseaux terrestres, dits "au sol" et les réseaux embarqués, dits "à bord", s'effectue via une troisième catégorie de réseaux : les réseaux dit "sol-bord". Ces réseaux reposent sur des ondes radios pour transmettre les données. L'objectif de cette section est de présenter les différentes technologies utilisées aujourd'hui pour assurer le niveau liaison entre l'avion et le sol.

Historiquement, la première technologie exploitait la bande radio Haute Fréquence (HF, [DO-163 1976]) et permettait d'acheminer la voix entre les pilotes et les contrôleurs. D'une portée très grande de l'ordre de 5 000 km, la bande HF 1.5 MHz - 30 MHz allouée à l'aéronautique par l'Union Internationale des Télécommunications (UIT [uit 2013]) a cependant des défauts : mauvaise qualité sonore entraînant une

difficulté des transmissions, faible nombre de canaux simultanément utilisables... Dans les zones terrestres peuplées, la bande VHF (Aeronautical Very High Frequency [DO-169 1979], de 118 à 137 MHz depuis 1994), d'une portée moindre mais d'une qualité meilleure, a vite supplanté la bande HF.

Parallèlement, les technologies par satellite se sont développées et ont permis d'apporter aux communications aéronautiques les mêmes avantages que la bande VHF mais pour toute zone (océanique comme continentale, seuls les pôles étant mal couverts par les satellites). L'organisation Inmarsat [Inmarsat 2013]fut fondée en 1979 et privatisée en 1999 et offre des débits par satellites de 600 bps à 10,5 kbps. La constellation de 66 satellites en orbite basse Iridium [Iridium 2013]est autorisée depuis 2011 par la FAA pour la transmission des données critiques aéronautiques à des débits de 2,4 kbps. Cependant, les prix et les délais des communications par satellite sont généralement bien supérieurs au VHF. Il en résulte qu'en vol, les pilotes exploitent prioritairement la VHF, puis les liaisons SATCOM s'il n'est pas possible d'employer la VHF, et enfin la HF dans le cas où les deux autres moyens seraient inutilisables.

Ces technologies, exploitées actuellement pour les communications vocales du contrôle aérien, sont aussi depuis plus récemment exploitées pour la communication des **données** aéronautiques du contrôle aérien. Ainsi, les réseaux Datalink peuvent exploiter la bande radio HF avec un protocole de niveau liaison appelé HFDL (High Frequency Datalink) et permettant un débit maximal de 1,8 kbps [DO-265 2000]. Le Datalink sur VHF est similaire, il existe quatre versions du protocole VDL (VHF Datalink), mais le protocole le plus utilisé aujourd'hui est le VDL mode 2, spécifié par l'OACI en 1996, il offre un débit limité à 31,5 kbps [DO-281A 2005, DO-281B 2012]. Enfin, les liaisons SATCOM peuvent aussi transmettre des données. A un niveau protocolaire supérieur, le format des messages échangés par ces moyens est appelé ACARS (Aircraft Communication Addressing and Reporting System) et il est standardisé par les normes ARINC 618 à 623 [ARINC618-6 2006, ARINC619-3 2009, ARINC620-7 2012, ARINC622-4 2001, ARINC623-3 2005].

Des technologies nouvelles sont également en cours d'étude. Ainsi, établir un réseau ad hoc où chaque avion est un nœud d'un réseau global permettrait de transmettre des données de proche en proche de l'avion émetteur jusqu'à une station au sol réceptrice, en utilisant des bandes de fréquence ne permettant pas d'interconnecter directement la source et la destination mais permettant de les interconnecter via une chaîne d'avions relayant les données (voir à cet effet l'étude [Besse 2013]).

Le Wifi et le WiMax sont d'autres technologies à l'étude [Matolak *et al.* 2011, Besse 2013]. Leurs faibles portées (de l'ordre de la dizaine de km pour le WiMax, de la centaine de mètres pour le Wifi), inférieures à celle des HF (environ 5000 km), VHF (environ 300 km) et SATCOM (distance non limitée), font qu'ils sont plutôt envisagés pour les communications lorsque l'avion est à l'aéroport. Ces technologies pourraient par exemple permettre le téléchargement durant le stationnement à l'aéroport de mises à jour de cartes aéronautiques et météorologiques.

| Nom | Bande de fréquence | Portée | Débit de données |
|---|---|---|---|
| HFDL | 2 - 20 MHz | ~ 5000 km | < 2 kbps |
| VDL2 | 118 - 138 MHz | ~ 250 km | < 32 kbps |
| SATCOM | 1525-1661 MHz (bande L) | $\infty$ | < 12 kbps |
| WiMax* | 2 - 66 GHz | ~ 50 km | 100 kbps à 22 Mbps |
| réseaux Ad-hoc* | peut utiliser la bande L | $\leq$ 600 km | du kbps à 1 Mbps |

*Prévisions relatives à ces technologies encore en cours de développement

**Table 2.2.:** Protocoles aéronautiques de liaisons sol-bord

Toutefois, quelque soit le moyen de communication utilisé, celui-ci concerne le niveau de liaison (niveau 2 du modèle OSI). Ainsi, notre routeur SNG qui gère le niveau réseau (niveau 3) n'a pas besoin de savoir quel niveau de liaison est employé : il envoie les données à l'équipement adéquat qui traite les données en conséquence. La seule propriété pouvant impacter l'intégration du routeur au sein de l'avion concerne le débit offert par le médium de niveau inférieur : chacun des moyens offre un débit qui lui est propre, comme résumé dans la table 2.2.

### 2.1.3. Les moyens de communication réservés au passager

En plus des communications dédiées au contrôle aérien, d'autres moyens de communication sont apparus dans les années 2000 pour vendre aux passagers des services de communication vocale et d'accès au web. La table 2.3 résume quelques solutions existant actuellement ou ayant existées mais abandonnées pour des raisons de rentabilité (cas de Connexion-by-Boeing).

| Nom | Réf | Mise en œuvre | Débit offert | Liaison |
|---|---|---|---|---|
| Connexion-by-Boeing | [byBoe 2013] | 2004 à 2006 | ~ Mbps | satellite |
| Swift 64 | [Swift64 2013] | depuis 2009 | < 64 kbps | satellite |
| SwiftBroadband | [SwiftBB 2013] | depuis 2009 | < 432 kbps | satellite |
| ARINC Onboard Internet | [Oi 2013] | depuis 2008 | < 432 kbps | satellite |
| Gogo Inflight, by Aircell | [GogoInf 2013] | depuis 2007 | <800 kbps | radio |

**Table 2.3.:** Moyens de communication pour les passagers

## 2.2. Architecture globale d'intégration du SNG

Les protocoles de communication présentés dans la section précédente constituent la couche de liaison entre systèmes «voisins». Ils peuvent utiliser pour cela une connexion filaire (cas des protocoles des communications internes à l'avion) ou une connexion sans fil (cas des communications sol-bord). Dans ce dernier cas, des

systèmes assurent la connexion entre le réseau embarqué et le réseau «extérieur» à l'avion.

Historiquement, chaque besoin de communications de données était pensé individuellement, menant ainsi à la multiplication des liaisons. Cette approche cloisonnée permet de garantir pour chaque communication une indépendance vis-à-vis des autres communications et ainsi apporter des garanties relatives à la sécurité et la sûreté de ces communications. Mais la multiplication des services aéronautiques a conduit les avionneurs à regrouper les systèmes partageant des protocoles de liaison communs en réseaux. Cette mutualisation permet de partager les liaisons et ainsi de réduire d'autant les contraintes d'installation associées : poids et positionnement des câbles, allocation de nouvelles fréquences radio pour les nouveaux services, certification des systèmes et de leurs connexions...

### 2.2.1. Domaines réseaux avioniques et aéronautiques

Deux terminologies sont employées en aéronautique pour classer les réseaux en fonction de leurs usages. La terminologie OACI définie dans l'annexe 10 [OACI ] est celle utilisée par Airbus, elle a une vue centrée sur les communications entre l'avion et les installations au sol et est parfaitement adaptée à la classification des communications sol-bord. L'ARINC 664P5 [ARINC664P5 2005] propose une terminologie alternative, utilisée par THALES AVIONICS, qui est plutôt centrée sur les systèmes embarqués et les services qu'ils fournissent, elle est donc adaptée à la classification des systèmes et réseaux embarqués.

L'annexe 10 de l'OACI [OACI ] définit quatre classes de services et de communications aéronautiques. L'Air Traffic Services Communication (ATSC) regroupe l'Air Traffic Control (ATC) réservée au contrôle, le Position Reporting pour la surveillance et le Flight Information Services (FIS) principalement dédiée à la navigation. L'Aeronautical Operational Control (AOC) regroupe les communications exigées durant les différentes phases du vol pour la sécurité du vol, sa régularité et son efficacité. Ces deux classes sont utilisées par les organismes d'Aviation Civile. L'Aeronautical Administrative Communication (AAC) sert aux aspects commerciaux du vol, cette classe est réservée à l'usage exclusif des compagnies aériennes. Enfin, l'Aeronautical Passenger Communication (APC) est dédiée aux communications passées par les passagers (Internet cabine, téléphonie...). L'ensemble de ces 4 domaines forment une architecture appelée «Aeronautical Telecommunication Network» (ATN) qui englobe donc les communications sol-sol, sol-bord et avioniques (bord-bord).

Le groupe de travail AEEC de l'entreprise ARINC a proposé dans [ARINC811 2005] une terminologie alternative avec quatre domaines avioniques de communication, proche de celle de l'OACI. L'Aircraft Control Domain (ACD) regroupe les systèmes embarqués nécessaires pour la gestion de l'avion et du vol. L'Airline Information Service Domain (AISD) regroupe les systèmes «non essentiels» à la sécurité du vol mais permettant d'optimiser tout processus du vol, ces systèmes sont destinés à la

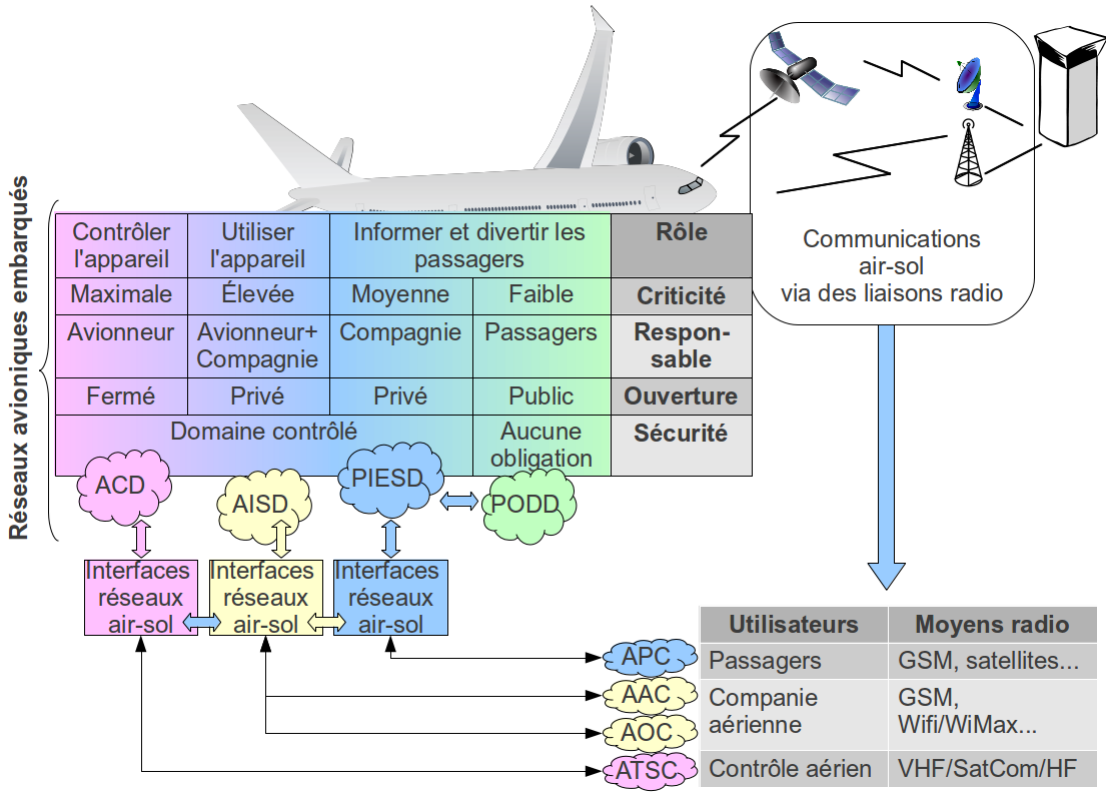



**Figure 2.1.:** Réseaux avioniques et aéronautiques

maintenance et à l'équipage, personnel commercial inclus. Le Passenger Information Entertainment Services Domain (PIESD) est dédié aux réseaux et systèmes embarqués auxquels les passagers ont accès depuis leur siège, il contient notamment l'In-Flight Entertainment (IFE) qui permet la diffusion aux passagers de films et de musiques, de jeux, de journaux en ligne... Enfin le Passenger-Owned Devices domain (PODD) réunit les réseaux et systèmes permettant aux passagers d'utiliser leurs systèmes personnels nécessitant des communications (téléphones portables, smartphones, tablettes tactiles, ordinateurs portables, etc...).

La figure 2.1 présente ces deux terminologies ainsi que les passerelles entre elles. Bien que définies de manière différente, l'ATC et l'ACD sont tous deux associés à des exigences extrêmement élevées en sûreté, en sécurité et en disponibilité. De plus, les systèmes et réseaux associés à l'ACD utilisent et transmettent des informations généralement associées à l'ATSC. Il y a donc un lien fort entre l'ACD et l'ATSC, voir même une certaine confusion... Il en est de même pour les couples AOC/AISD, AAC/AISD, APC/PIESD et APC/PODD.

Basée initialement sur des protocoles réseaux propriétaires fermés, l'architecture ATN proposée par l'OACI est en cours de mutation pour remplacer les protocoles propriétaires par d'autres protocoles courants dans le monde extérieur à l'aéronautique, notamment des protocoles standardisés par l'Internet Engineering

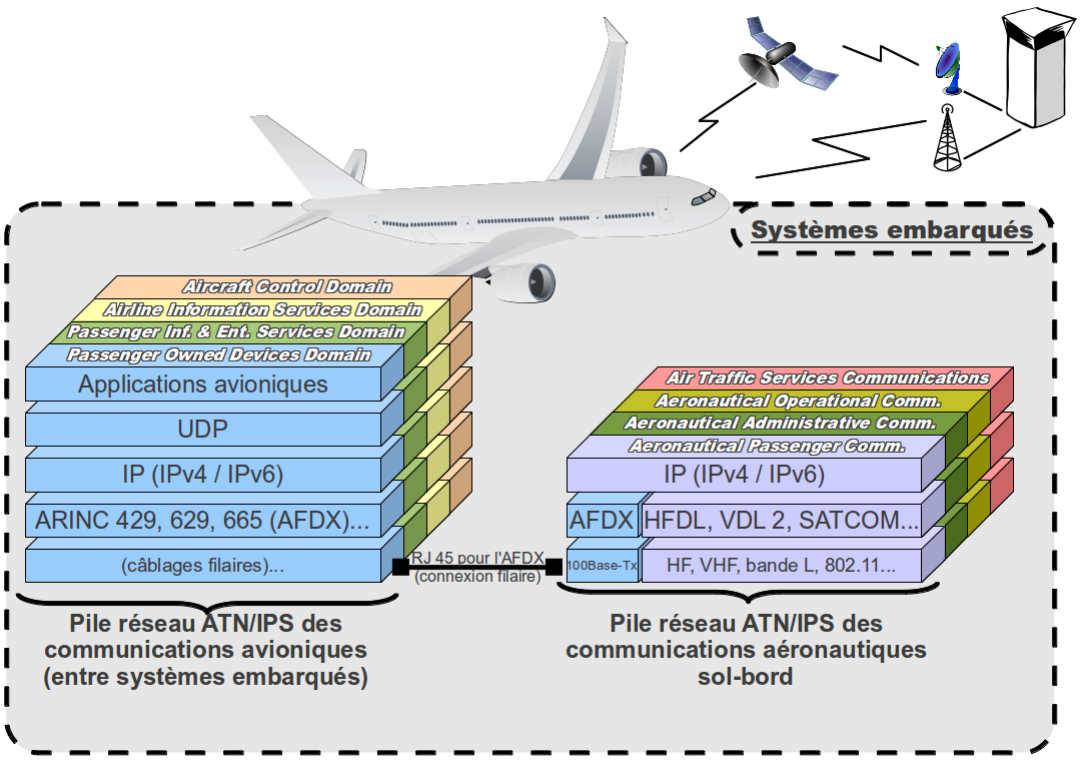



**Figure 2.2.:** Piles réseaux ATN/IPS

Task Force (IETF) qui publie et organise les standards et propositions sous forme de «Request For Comments» (RFC). Cette architecture de nouvelle génération est dénommée «Aeronautical Telecommunication Network with Internet Protocol Suite» (ATN/IPS) et son principal avantage est sa compatibilité avec les équipements réseaux grand public appelés «Composants Sur Étagères» («Commercial Off-The-Self» ou COTS). Notre routeur SNG s'inscrit pleinement dans l'architecture ATN/IPS. Elle est présentée figure 2.2, on y retrouve certains protocoles de liaison présentés dans la section 2.1, les protocoles des niveaux supérieurs sont présentés plus loin dans la section 2.3.1.

### 2.2.2. Interconnexion des domaines avioniques par un routeur de nouvelle génération

Sur les appareils récents tel que l'A380, les exigences de sécurité de l'aviation civile concernant l'ACD, l'AISD, le PIESD et le POD diffèrent. Afin de pouvoir garantir des niveaux élevés d'assurance, la solution actuellement mise en place repose sur le cloisonnement. Ainsi, l'avion embarque plusieurs réseaux avioniques indépendants, chacun dédié à un seul domaine ou à un seul sous-domaine. Historiquement, les informations tels que la température au sol ou l'heure locale, accessibles par le

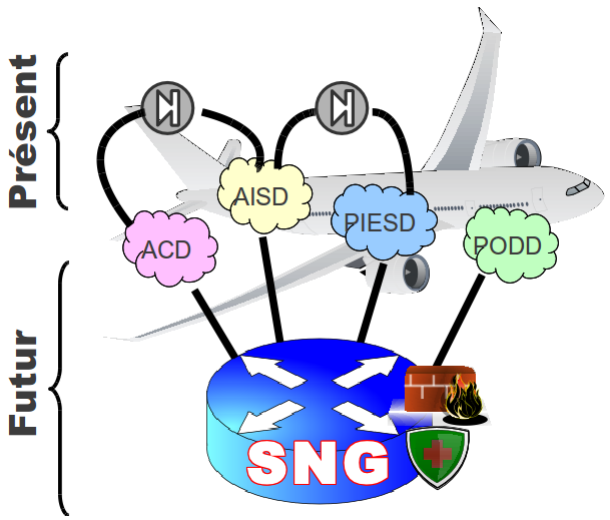



**Figure 2.3.:** Interconnexion de réseaux avioniques par un routeur embarqué

personnel de bord commercial (stewards et hôtesses de l'air), étaient inaccessibles électroniquement par les pilotes et par les passagers, le personnel assurant la diffusion de ces informations par des annonces vocales.

Cependant, de nouveaux besoins sont apparus, certaines informations nécessitent désormais de passer d'un réseau d'un domaine à un réseau d'un autre domaine. Ainsi, il est désormais possible d'afficher l'altitude de l'avion (issue de l'ACD) sur les moniteurs des passagers (connectés au PIESD). Cela est rendu possible par l'utilisation de systèmes interconnectant des réseaux avioniques de différents domaines. Cette ouverture de réseaux jusqu'alors fermés pose des problèmes nouveaux de sûreté et de sécurité. Les exemples de l'A380 et du B787 montrent une ouverture très contrainte et très limitée : l'ACD peut envoyer des données vers l'AISD via un équipement appelé «diode réseau». Comme son nom l'indique, cet équipement permet le passage de certaines données, après un filtrage, uniquement de l'ACD vers l'AISD, bloquant toute donnée vers l'ACD. De même, une diode réseau connecte l'AISD vers le PIESD [FAA 2007].

Ainsi les diodes réseaux sont un premier pas vers l'ouverture des réseaux. Mais leur conception et leur configuration au cas par cas empêchent une mise à l'échelle efficace. C'est pourquoi l'ARINC 811[ARINC811 2005], qui propose une architecture des systèmes d'information embarqués dans les avions futurs, préconise une architecture avionique centrée non plus sur la coexistence de réseaux faiblement couplés mais centrée sur un ensemble de réseaux interconnectés par des composants appelés routeurs, comme illustré sur la figure 2.3.

En fait, l'illustration 2.3 n'est pas parfaitement conforme aux études telles que l'ARINC 811 puisque celles-ci envisagent non pas un seul routeur mais plusieurs équipements pour augmenter la protection contre les pannes ; le concept sous-jacent d'interconnexion des domaines reste cependant invariant.

Le déploiement massif de l'AFDX, aujourd'hui basé sur l'usage du protocole IPv4 comme protocole réseau et prévu pour supporter son successeur le protocole IPv6, permet d'envisager d'utiliser des routeurs IP. Ces derniers existent déjà depuis longtemps en dehors du monde aéronautique, la technologie est donc éprouvée. Mais l'intégration à l'aéronautique pose de nombreuses questions difficiles de certifiabilité et d'évaluation de ces systèmes : les contraintes élevées de sécurité et de sûreté font qu'il n'existe actuellement aucun routeur IP qualifié et pouvant donc être embarqué. L'un des objectifs de ma thèse consiste à étudier les contraintes imposées à la réalisation d'un routeur certifié pour l'aéronautique et à proposer des solutions adaptées pour en réduire les coûts.

### 2.2.3. Mutualisation des liaisons aéronautiques sol-bord par un routeur de nouvelle génération

Un deuxième besoin de l'aéronautique concerne le multiplexage des flux sol-bord : actuellement, les réseaux avioniques sont fermés et ne communiquent pas entre eux. Les moyens de communication sol/bord sont dupliqués pour chaque domaine réseau s'étendant de l'avion au sol, garantissant ainsi la disponibilité du lien et dédiant toute la capacité au domaine utilisant le lien. Cela entraîne une augmentation du nombre de liens et d'équipements (et donc des coûts associés).

Une solution consiste à mutualiser la liaison sol/bord, quand les moyens techniques et la capacité du lien le permettent. La liaison transporte alors simultanément plusieurs flux issus de domaines différents. On parle alors de multiplexage des flux de données sur un lien. Le multiplexage est assuré par le routeur lorsque les données provenant d'interfaces réseaux différentes sont envoyées vers l'équipement de télécommunication embarqué, via une même interface réseau du routeur. Le démultiplexage, opération inverse du multiplexage, est alors effectué par un ou plusieurs routeurs au sol.

La figure 2.4 présente un exemple d'utilisation de la mutualisation air-sol. Un routeur embarqué est connecté physiquement à trois réseaux avioniques, il dispose d'un lien unique avec le sol pour acheminer les données des trois réseaux. Au sol, un premier routeur sépare les données non critiques des passagers et assure la connectivité avec un Fournisseur de services d'Accès à l'Internet (FAI). Les flux avioniques dédiés à l'exploitation commerciale de l'appareil sont transmis à un second routeur au sol qui démultiplexe les flux et les routes vers les équipements adéquats de la compagnie aérienne exploitant l'avion. L'ARINC 811 et son complément l'ARINC 821 [ARINC821 2008] qui approfondit les problématiques de sécurité préconisent l'exploitation de tunnels pour transporter les données en maintenant la ségrégation, comme présenté figure 2.4. Ces tunnels doivent être sécurisés en raison de la sensibilité potentielle des informations transportées : données commerciales, personnelles...

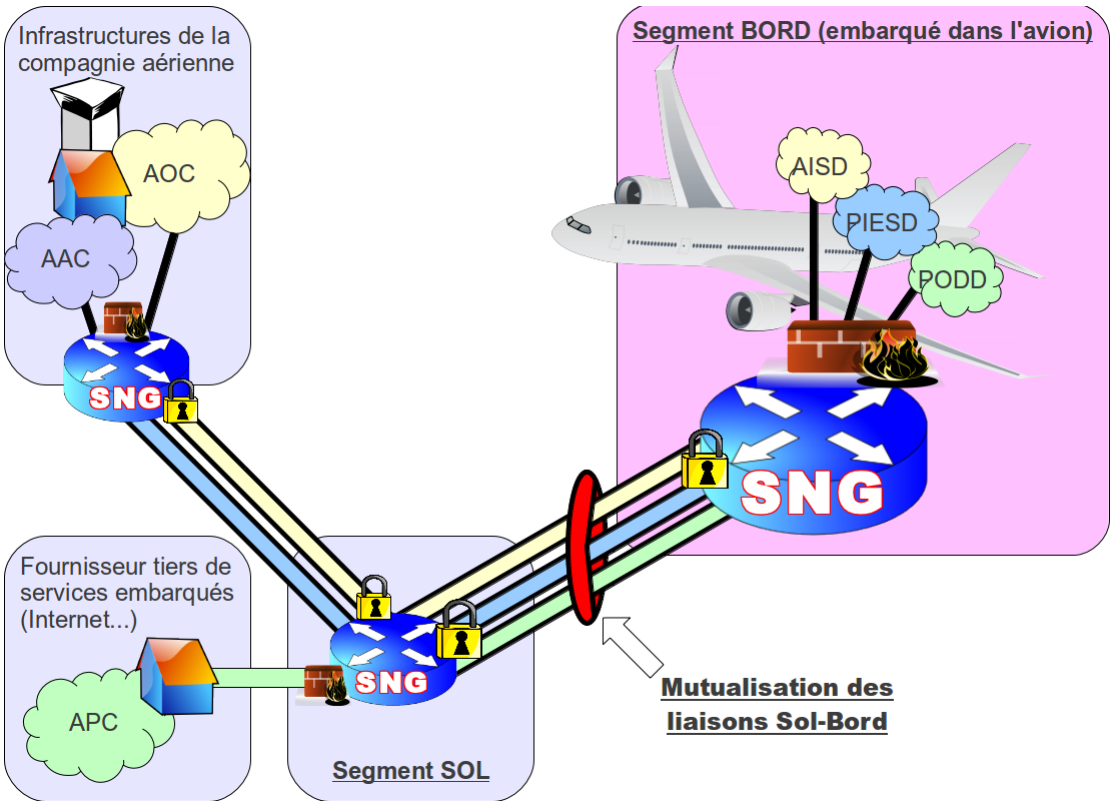



**Figure 2.4.:** Mutualisation des liaisons aéronautiques

## 2.3. Formalisation du cahier des charges industriel

Dans les sections précédentes, l'environnement du routeur SNG a été présenté. L'objectif de cette section est de synthétiser les fonctions du routeur et ses particularités. La fonction principale d'un routeur est de «router» des données, c'est-à-dire diriger les messages reçus des interfaces réseaux du routeur vers la ou les interfaces adéquates. Notre routeur étant destiné à interconnecter des réseaux aéronautiques critiques, nous avons aussi mis en œuvre des mécanismes de filtrage et de sécurisation des données. Les contraintes de développement qui s'appliquent aux systèmes avioniques critiques sûrs et sécurisés tels que notre routeur SNG seront présentées dans les sections ultérieures. Cette section expose les spécifications et non la mise en œuvre : le «quoi» présenté ici est suivi du «comment» dans le chapitre 4.

### 2.3.1. Routage

La fonction première et principale d'un routeur est de router les données entre les différents réseaux que le routeur interconnecte. Dans le cas du routeur SNG, nous avons vu dans le chapitre précédent que le routeur peut être connecté notamment aux domaines réseaux critiques ACD et AISD ainsi qu'au PIESD, présentés section

2.2.1. Ces domaines utilisent tous l'AFDX comme couche de liaison (niveau 2). Ainsi les interfaces réseaux du routeur SNG sont des interfaces AFDX.

Ces domaines réseaux utilisent actuellement le protocole IP version 4 (standard initial publié en 1982 par l'IETF dans la RFC 791 [Postel 1981b]) pour la couche réseau (niveau 3). Cependant l'aéronautique se prépare déjà à utiliser son successeur, le protocole IP version 6 (IPv6, standard initial publié en 1995 dans la RFC 1883 et mis à jour dans la RFC 2460[Deering & Hinden 1995, Deering & Hinden 1998]). Ces deux protocoles IPv4 et IPv6 peuvent coexister sur les mêmes infrastructures matérielles. Ils exploitent tous deux une décomposition des données en sous-ensembles finis de suites d'octets, dénommés «paquets».

Après de nombreuses discussions avec THALES AVIONICS, nous avons choisi de nous orienter vers un routeur «dual-stack» : ce type de routeur possède plusieurs piles réseaux, chacune dédiée à un unique protocole de niveau 3. Dans le cas du routeur SNG, cela signifie mettre en œuvre indépendamment des mécanismes pour IPv4 et d'autres pour IPv6. L'identification des mécanismes à employer s'effectue à l'aide des informations transmises par la couche liaison : pour chaque paquet reçu, cette dernière indique s'il faut employer les mécanismes pour IPv4 ou pour IPv6.

Le routage du paquet s'effectue à l'aide d'une table de routage, indiquant vers quel équipement et donc via quelle interface réseau renvoyer chaque paquet reçu. Pour des raisons de sécurité, nous avons choisi de ne pas utiliser de méthode dynamique de création et de gestion de cette table de routage. Dans le routeur SNG, la table de routage est un élément statique de configuration, défini à l'installation du routeur dans l'avion et non modifiable par la suite.

Les couches des niveaux supérieurs (couche 4 de transport, couche 5 de session, couche 6 de présentation et couche 7 applicative) n'ont pas d'impact sur le routeur : les données et métadonnées sont encapsulées dans les paquets IP et le routeur SNG route les paquets de manière atomique en ne modifiant que ce qui est exigé par les standards réseau. Ainsi la modification du Time To Live (TTL) et le recalcul de la somme de contrôle ont été spécifiés dans les standards pour ne pas impacter les données transportées dans le paquet.

Les réseaux avioniques utilisent actuellement des paquets IPv4 transportés par l'AFDX et encapsulant de l'UDP (protocole de transport défini en 1980 dans la RFC 768 [Postel 1980]). Les besoins futurs s'orientent vers de l'IPv6 pouvant encapsuler soit de l'UDPv6 (protocole de transport UDP modifié par la RFC 2460 [Deering & Hinden 1998] en 1999 pour fonctionner avec le protocole IPv6) soit du TCPv6 (Transmission Control Protocol for IPv6, protocole de transport TCP publié dans la RFC 793 [Postel 1981c] en 1982 et modifié en 1999 par la RFC 2460 pour fonctionner avec le protocole IPv6). Ces considérations permettront d'orienter l'évaluation des performances, présentée chapitre 6.

L'idée initiale de faire du routeur SNG une passerelle IPv4/IPv6, convertissant les paquets d'un protocole réseau à l'autre durant leur routage a été abandonnée : ce concept de passerelles est par nature vulnérable (comme le montre l'ensemble

des RFC relatives à ces solutions : [Aoun & Davies 2007, Thaler *et al.* 2010, Huitema 2006, Bagnulo *et al.* 2011, Despres 2010, Savola & Patel 2004]) et les risques pour la sûreté et la sécurité ont été jugés trop importants par rapport aux gains attendus.

### 2.3.2. Filtrage

Le routeur connecte des réseaux de criticités différentes. Les vecteurs d'attaques sont différents entre l'ATSC, accessible uniquement aux services de surveillance et de navigation aérienne via des équipements certifiés, et l'APC, accessible aux passagers via leurs équipements personnels de communication. L'impact d'une attaque réussie est aussi différent : l'indisponibilité de l'APC est une gêne pour les passagers et impacte l'image de marque de la compagnie aérienne mais n'a aucune incidence sur la sécurité du vol, tandis que l'indisponibilité de l'ATSC peut se traduire par la perte de contrôle de l'appareil et donc l'écrasement de l'avion et la mort d'êtres humains.

Jusqu'alors isolés les uns des autres, l'interconnexion des réseaux augmente leur vulnérabilité et nécessite de mettre en place des systèmes pour y remédier. Cependant, les contraintes élevées de sûreté de fonctionnement rendent difficiles voir impossible la mise en place de solutions logicielles ou matérielles complexes sur un système tel que le routeur.

Les réunions de travail avec THALES AVIONICS ont abouti à la mise en place d'un mécanisme statique de filtrage de l'entête IP de chaque paquet reçu avant d'effectuer tout autre traitement. Sa mise en œuvre est détaillée section 4.2.5.

Il fut envisagé d'inclure des mécanismes plus avancés tel que le «Deep Packet Inspection» ou en français «Inspection de Paquets en Profondeur» (voir à cet effet l'article [Dubrawsky 2003]) qui analyse non seulement le paquet IP mais aussi son contenu, c'est-à-dire les entêtes protocolaires des niveaux supérieurs et le contenu applicatif. Ces mécanismes sont indépendants du filtrage d'entêtes IP présenté ci-dessus ; de ce fait, nos réunions avec THALES AVIONICS se sont orientées vers le développement de deux systèmes complémentaires à déployer dans l'architecture avionique, plutôt que vers le développement d'un système unique complexe, compliqué à mettre en œuvre, à certifier, à évaluer et à maintenir.

### 2.3.3. Multiplexage et sécurisation des communications

La sous-section 2.2.3 a présenté un des scénarios d'utilisation du routeur SNG : la mutualisation des communications sol-bord sur une liaison unique, par l'utilisation de techniques de multiplexage.

Cependant, le multiplexage engendre des problématiques de sécurité, notamment la garantie de maintient de la ségrégation des flux : comment garantir que les ressources sont partagées de la manière prévue par la configuration ? La solution retenue pour

le routeur SNG et présentée section 4.4.3 utilise la «tunnelisation» des paquets IP, c'est-à-dire l'encapsulation des paquets dans d'autres paquets.

Les besoins des communications aéronautiques ont aussi évolués depuis ces dernières années. Historiquement, les besoins exprimés concernaient principalement la disponibilité et l'intégrité des communications et étaient traités uniquement du point de vue sûreté de fonctionnement.

L'**intégrité** est la propriété d'une information d'être correcte, garantir l'intégrité de la communication consiste à empêcher la modification (altération, création ou suppression) indue de l'information, que la modification soit due à des dysfonctionnement ou à des utilisateurs non autorisés.

La **disponibilité** est la propriété d'une information ou d'un service d'être accessible quand l'utilisateur en a besoin, cela suppose de fournir l'accès en lecture et modifications aux utilisateurs autorisés et faire en sorte que les autres utilisateurs ne puissent pas empêcher les premiers d'accéder à l'information.

Le risque terrorisme est apparu plus récemment en aéronautique, il a notamment été mis en avant suite aux attentats du 11 septembre 2001 et a entraîné de profondes évolutions de la sûreté aéronautique. Les besoins aéronautiques ont ainsi été complétés par des besoins de confidentialité de certains flux et d'authentification des communications. La **confidentialité** est la propriété d'une information de ne pas être révélée à des utilisateurs non autorisée. L'**authentification** est le processus de vérification de l'identité des entités communicantes.

Ainsi, les données échangées via le routeur entre deux systèmes terminaux peuvent nécessiter l'application, sur une partie ou l'intégralité du chemin, de services de sécurité tels que la confidentialité, l'intégrité et l'authenticité des messages.

### 2.3.4. Exigences non fonctionnelles : sûreté et sécurité

Le contexte avionique et sécuritaire du routeur est un point complexe traité parallèlement au développement des fonctionnalités. Il impose des exigences particulières durant la conception et l'exploitation des équipements, indépendantes des fonctions qu'ils assurent. Ces exigences sont alors qualifiées d'exigences non fonctionnelles.

Historiquement, la sécurité du transport aérien était garantie par la vérification de l'innocuité des systèmes (aussi appelée sûreté de fonctionnement ou "safety" en anglais), aboutissant à une certification par un organisme habilité. Cependant, les considérations de sûreté aérienne sont de plus en plus présentes dans les cycles de développement et conduisent les équipementiers à vérifier aussi l'immunité des systèmes (aussi appelé "security" en anglais), aboutissant à une évaluation du système.

Dans le domaine des transports, la différence entre sécurité et sûreté peut généralement être rapidement évaluée en se posant la question de la cause du risque traité : le risque est-il la conséquence d'un acte volontaire ou bien involontaire ? Si l'acte a


**Figure 2.5.:** Légende utilisée dans ce mémoire pour la sécurité et la sûreté

pour but d'exploiter volontairement une vulnérabilité, alors le traitement du risque relève de la **sûreté** et est traité en rendant le système immunisé à la vulnérabilité, on parle alors de **sécurité-immunité**. Si la cause est involontaire, alors le traitement relève de la **sécurité** (aussi appelé sûreté de fonctionnement) et il est traité en neutralisant les effets éventuels d'un dysfonctionnement sur l'environnement du système, on parle alors de **sécurité-innocuité**. Cependant, la frontière entre ces deux notions est perméable : une vulnérabilité peut être exploitée par un attaquant pour conduire au dysfonctionnement des systèmes voisins. La protection des systèmes entre alors dans le cadre de l'ingénierie «security-for-safety».

Dans ce mémoire, les exigences de sûreté et de sécurité relatives aux flux traités par le routeur seront dissociées de celles relatives au routeur lui-même. Dans le premier cas, elles sont fonctionnelles : le routeur a pour fonction de sécuriser les communications, donc les flux qui le traversent. Dans le deuxième cas, elles sont non fonctionnelles : la sûreté et la sécurité du routeur sont intrinsèques au système, elles doivent être garanties, qu'il y ait ou non des données à traiter et quelque soient ces données.

La figure 2.5 présente les trois icônes que nous utilisons dans les illustrations de ce mémoire pour mettre en avant le type d'exigence concerné par l'illustration : un bouclier indique la prise en compte de la sécurité intrinsèque du routeur, un mur pare-feu avec une flamme correspond à la sûreté intrinsèque du routeur et enfin un cadenas représente la sécurisation des flux de données traversant le routeur SNG. La figure 2.6 en conclusion de ce chapitre résume ces exigences réunies au sain du routeur SNG.

Les deux sections suivantes traitent respectivement des exigences fonctionnelles de sécurité puis de sûreté.

## 2.4. La certification, garante de l'innocuité

### 2.4.1. Introduction à la certification des logiciels en aéronautique

Quelque soient les solutions technologiques et algorithmiques sélectionnées pour le système, les utiliser dans des systèmes aéronautiques embarqués impose à l'avionneur de certifier leur fonctionnement. L'objectif global de la phase de certification est de prouver l'innocuité du système, ou au minima d'apporter suffisamment de garanties de sûreté de fonctionnement pour autoriser à intégrer le système dans l'avion. Ces garanties sont apportées par la vérification de l'absence de dysfonctionnement et, le cas échéant ou dans le cas de dysfonctionnement imprévisibles aléatoires, par l'ajout de mécanismes de confinement des pannes.

La certification d'un système est délivrée par une autorité de l'aviation civile, nationale ou internationale, pour l'exploitation d'une version d'un équipement donné dans un type d'aéronef donné avec une configuration ou un ensemble de configurations autorisés. En pratique, des accords entre autorités nationales ou multinationales permettent d'éviter de certifier un système auprès des autorités des 180 pays de l'OACI. Par exemple, l'Agence Européenne de la Sécurité Aérienne (EASA [EASA 2013]) délivre un agrément d'exploitation valable dans toute l'Union Européenne lorsque la DGAC française certifie un nouvel équipement pour un Airbus A380. Par contre, utiliser un équipement certifié pour un type d'appareil (par exemple un A380) sur un autre type d'appareil (un A320 par exemple) ou utiliser un équipement non certifié est interdit par la législation, cela entraînerait donc l'interdiction d'exploitation de l'avion sur lequel est installé l'équipement illégal.

La principale norme utilisée pour la certification des équipements avioniques est la norme ARP 4754 [ARP4754A 2010] «The Aerospace Recommended Practice (ARP) ARP4754A (Guidelines For Development Of Civil Aircraft and Systems )». Elle considère l'équipement en tant que système complexe et est complétée par les normes DO-178 «Software Considerations in Airborne Systems and Equipment Certification» [DO-178C 2011] et DO-254 «Design Assurance Guidance For Airborne Electronic Hardware» [DO-254 2000], respectivement pour les aspects logiciels et matériels. Étant donné que cette thèse ne traite que les aspects de certification du logiciel du routeur SNG, nous ne parlerons pas plus en détail des normes ARP 4754 et DO-254.

La norme DO-178 [DO-178 1982] fut initialement écrite en 1982 puis améliorée dans ses versions DO-178A [DO-178A 1986] puis DO-178B[DO-178B 1992], parues respectivement en 1986 et en 1992. Cette norme regroupe un ensemble de recommandations relatives aux différents aspects de la certification des logiciels avioniques : la planification des étapes de certification, de spécification, de développement et de validation du logiciel, les processus de développement employés et règles de codage, la validation des résultats des tests et de leur couverture, la gestion des configu-

| Niveau | Idys(1) | Pdys(2) | #objectifs | #objectifs indep |
|---|---|---|---|---|
| DAL-E | sans effet | - | 0 | 0 |
| DAL-D | mineur | - | 28 | 2 |
| DAL-C | majeur | 1e-5 | 57 | 2 |
| DAL-B | critique | 1e-7 | 65 | 14 |
| DAL-A | catastrophique | 1e-9 | 66 | 25 |

(1) Impact d'un dysfonctionnement
(2) Pdys = probabilité d'admissibilité maximale d'un dysfonctionnement
**Table 2.4.:** Design Assurance Levels du DO-178B

rations et des versions... Un document additionnel, le DO-248B[DO-248B 2001], recueille les questions courantes des industriels relatives à cette norme.

La norme définit cinq niveaux de certification appelés les «DAL» (Design Assurance Level) et classés du DAL-E, non contraignant, au DAL-A, le niveau le plus exigeant. En fonction des fonctionnalités du système, de l'architecture dans laquelle il est intégré, et surtout de l'impact en cas de défaillance, les autorités de certification exigent un niveau minimal d'assurance, tel que défini dans la table 2.4.

La certification est délivrée quand l'équipement valide un certain nombre d'objectifs fixés par la norme. Aucun objectif n'est exigé pour le DAL-E. Les objectifs exigés pour le DAL-C reprennent tous les objectifs du DAL-D avec des objectifs supplémentaires. Ce principe cumulatif est identique pour les DAL-B et DAL-A, établissant ainsi une relation d'ordre total entre les niveaux d'assurance. Ainsi, un équipement certifié au niveau DAL-B implique la certification aux niveaux DAL-C et DAL-D. Ce principe cumulatif se retrouve aussi pour les coûts de certification : chaque objectif est coûteux à contrôler et à valider, il est donc plus onéreux de certifier un code au niveau DAL-A qu'au niveau DAL-D.

De plus, la norme impose de vérifier certains objectifs «avec indépendance». Dans ce cas, l'entité concernée par la validation ne peut être validée par la personne qui l'a créée. Par exemple, au niveau DAL-A, les développeurs du code source ne peuvent valider eux-mêmes l'objectif «le code source est conforme aux exigences de bas niveau» (A.5.1). Pour la même raison d'indépendance, la vérification de la couverture complète des exigences par l'ensemble des tests (A.7.4) ne peut être validée par les ingénieurs ayant participé à l'écriture des jeux de tests.

Afin de réduire ou de supprimer certains objectifs, le processus de certification peut s'appuyer sur des outils logiciels, notamment pour automatiser certaines vérifications. Ces outils doivent être vérifiés conformément à la norme, on parle alors de qualification de ces outils, pour générer ou vérifier un système postulant à la certification.

Il est important de noter que si la norme impose au postulant à la certification d'établir l'assurance de bon fonctionnement du système, elle ne lui impose aucune méthode, processus ni technologie pour arriver à ce résultat. Elle lui im-

pose par contre de vérifier que la méthode permet d'arriver à cette assurance [Baker 2011, DO-178B 1992]. Ainsi, le constructeur avionique et ses équipementiers doivent apporter les éléments prouvant que le système fonctionne correctement ainsi que les éléments prouvant que la vérification et la validation sont valides.

Les groupes de travail WG-71 d'EUROCAE et SC-205 de RTCA ont à nouveau travaillé à sa mise à jour dès 2004. Ils ont publié une nouvelle version en 2012, le DO-178C [DO-178C 2011], ainsi que le recueil actualisé des questions courantes des industriels [DO-248C 2012]. Quatre publications annexes, les DO-330 à DO-333 [DO-330 2011, DO-331 2011, DO-332 2011, DO-333 2011], sont dédiées à la qualification des outils de développement et de tests, au développement basé sur les modèles, au développement orienté objets et aux méthodes formelles.

### 2.4.2. Spécifications officielles du routeur SNG

Le DO-178B est la norme sur laquelle nous nous sommes appuyés pour éviter des écueils de certification à la fin de l'élaboration du routeur SNG. Celle-ci précise qu'il est impératif de définir une spécification claire et explicite du logiciel pour permettre de développer un code source qui aboutira à une certification. C'est pourquoi dans le cadre du routeur SNG, nous avons rédigé un «Software Requirement Specifications» (SRS) sur la base d'un modèle vierge fourni par THALES AVIONICS Toulouse.

Dans notre cas d'un prototype de démonstrateur qui ne sera pas certifié par un organisme extérieur, ce document est facultatif, informel et donc non soumis aux exigences de qualité préconisées par le DO-178B. Cependant, il permet de faciliter la discussion des fonctionnalités du système, de leurs algorithmes et des limites numériques des entrées et des sorties, de préciser ce qui est attendu du système et après le développement de vérifier que toutes les fonctions prévues (et uniquement celles prévues) ont été mis en œuvre dans le routeur SNG.

Ce document synthétise la décomposition du système «Routeur SNG» en sous-systèmes pour chacun desquels un ensemble d'exigences est associé. Comme défini dans [IEEE830 1998] et [IEEE1233 1998], une exigence est une phrase décrivant une contrainte s'appliquant au système, d'une manière claire, correcte, complète, non ambiguë et justifiée. Cette exigence est identifiée de manière unique, traçable et versionnée, c'est-à-dire que toute modification d'un élément de l'exigence conduit à lui attribuer un nouvel identifiant afin d'éviter des conflits ultérieurs de versions différentes.

Maintenant que nous avons présenté le but de la certification qui est de garantir l'innocuité du système, nous allons nous intéresser à l'évaluation qui garantit son immunité.

## 2.5. L'évaluation, garante de l'immunité

Les attentats du 11 septembre 2001 ont entraîné de profondes modifications dans l'aviation civile. L'une d'entre elle concerne la sûreté des équipements, c'est-à-dire leur immunité face aux menaces volontaires extérieures. Cela ne concerne pas les dysfonctionnements et pannes involontaires, tels que la mauvaise utilisation du matériel, la corruption des données et des mémoires par des ions et des radiations (voir à cet effet Single Event Upset [May & Woods 1979])...

En dehors du domaine aéronautique, la sûreté des logiciels est un sujet déjà amplement traité, aussi bien dans la littérature que par les applications industrielles. Cependant, celles-ci sont coûteuses à mettre en place et parfois contraire aux besoins de sécurité de l'aéronautique.

Par exemple, historiquement les communications entre les pilotes et les contrôleurs ne sont pas chiffrées afin qu'un maximum d'aéronefs puissent recevoir et entendre la communication, et éventuellement alerter pilotes et contrôleurs de dangers qu'ils n'auraient pas vu. Cependant, ces communications sont confidentielles et seuls les possesseurs de licences aéronautiques sont qualifiés à communiquer sur les fréquences radiophoniques allouées à l'aviation civile. Cette restriction est assurée actuellement par la délivrance et le contrôle au cas par cas des équipements de communication. Cette solution est vulnérable au vol des équipements, contrairement au chiffrement pour lequel les clefs peuvent être changées régulièrement, mais elle évite la collision entre deux avions pour cause de défaut de communication entre un appareil n'ayant plus de clef valide de chiffrement avec un autre «à jour».

### 2.5.1. La norme ISO CEI 15408 CC ITSEC

Différentes normes sont applicables pour garantir la sûreté des logiciels. À ce jour, les organismes législatifs internationaux aéronautiques n'en ont privilégié aucune ni créé de nouvelle dédiée à l'avionique. Cependant, différentes publications et standards présentent des moyens et des remarques relatives aux considérations de sécurité dans le contexte particulier de l'aéronautique : [Stephens 2004, De Cerchio & Riley 2011, Skaves 2011, Olive *et al.* 2006, iso iec 2008, ARINC811 2005, DO-326 2010].

Dans le cas du routeur SNG, nous en avons choisi une parmi les plus connues : la norme ISO CEI 15408. Nommée «Common Criteria for Information Technology Security Evaluation» et abrégée en CC ITSEC ou encore en CC, souvent appelée les «Critères Communs», cette norme permet d'évaluer un système complet vis-à-vis d'exigences de sécurité. Elle est publiée en quatre parties : [cc1 2009, cc2 2009, cc3 2009, cc4 2009].

Nous avons choisi d'utiliser cette norme des critères communs car elle nous apparaît comme la plus adaptée pour l'élaboration d'un nouveau système basé sur du logiciel embarqué tel que le routeur SNG. De plus, elle est la norme la plus

souvent citée dans les publications relatives à la sûreté dans l'aéronautique : le WG-72 d'EUROCAE étudiant la sécurité des systèmes avioniques et l'ARINC 821 [ARINC821 2008] présentant une architecture avionique sûre basée sur IP se réfèrent tous deux aux Critères Communs. Enfin, certains routeurs commerciaux ont bénéficié de cette norme pour démontrer leur sûreté dans des domaines autres que l'aéronautique [Labs 2009b, Labs 2009a].

Bien que certains auteurs de publications considèrent que cette norme est inadaptée au contexte aéronautique (voir [Hearn 2004, Keblawi & Sullivan 2006]), d'autres auteurs de publications plus récentes la considèrent au contraire adéquate (voir [Haley *et al.* 2008, Huyck 2010]). De plus, la compréhension de cette norme s'améliore au fil du temps, ses mises en application évoluent comme le montrent les publications [Motré & Téri 2000, Mellado *et al.* 2006, Farkhani & Razzazi 2006, Morimoto & Cheng 2005] ce qui expliquerait l'adhésion croissante pour les CC appliquées à l'aéronautique.

De la même manière que la certification de sécurité d'un produit n'est valable que pour la version du produit dans le contexte avionique où elle a été validée, l'évaluation de sûreté est spécifique à un équipement dans une version figée et dans un contexte défini précisément, elle n'est pas généralisable à une gamme de produits.

Les critères communs définissent d'une part des exigences fonctionnelles de sécurité («Security Functional Requirements» ou SFR) qui doivent être mises en œuvre dans le produit. Ils définissent d'autres part des exigences d'assurances de sécurité («Security Assurances Requirements» ou SAR) qui contraignent les évaluateurs durant le processus de validation du produit. La formulation de ces exigences est très contrainte par la norme, afin de garantir sa clarté et de permettre l'automatisation de la validation de l'exigence.

Les exigences peuvent être regroupées dans des «paquetages» d'exigences. Il existe ainsi 7 paquetages constitués uniquement d'exigences d'assurance et appelés «Niveaux d'assurance d'évaluation» («Evaluation Assurance Level» ou EAL). Ces paquetages d'exigences sont numérotés de 1 à 7. L'EAL-2 reprend l'ensemble EAL-1 et y ajoute des SAR supplémentaires. Il en est de même jusqu'à l'EAL-7, le plus haut niveau d'assurance de sécurité que peut recevoir un produit dans la nomenclature CC ITSEC.

Le premier niveau (et le moins contraignant) exige de l'évaluation un rapport sur le produit contenant les spécifications techniques et la vérification d'éléments tels que la cohérence de la conception et la conformité des documentations techniques. Les niveaux supérieurs sont de plus en plus exigeants, des preuves formelles sont exigées à partir de l'EAL-5 et l'EAL-7 nécessite la validation automatique par preuves formelles[cc1 2009].

L'évaluation est basée sur un document de synthèse, la «cible de sécurité» («Security Target» ou ST), qui présente le contexte de l'équipement, l'analyse des risques et l'énumération des éléments vulnérables, les exigences de sécurité qui découlent des

menaces, les exigences pour évaluer le produit et le résultat de l'évaluation, ainsi que les matrices de traçabilité associées à tous ces éléments.

### 2.5.2. Application au routeur SNG : le Profil de Protection

La «Cible de sécurité» étant spécifique à un produit complet figé dans une version, elle est inadaptée à notre routeur logiciel SNG. Par contre, la norme CC ITSEC permet l'écriture de «Profil de Protection» («Protection Profile» ou PP), une variante des ST pouvant être incomplète et générique. Nous avons choisi ce type de document pour le routeur SNG et avons donc créé le «Profil de Protection du Routeur SNG» (PP SNG), joint en annexe A. Ce profil décrit le contexte d'utilisation aéronautique du routeur SNG mais ne précise pas de version. Il contient des exigences fonctionnelles de sécurité applicables aux aspects logiciels d'un tel routeur avionique mais aucune exigence d'assurance.

Ainsi, si deux versions dérivées du routeur SNG devaient être évalués à des niveaux d'assurances différents, alors les deux cibles de sécurité pourraient réutiliser, via une référence «de conformité» ou même une inclusion, la base commune que représente le Profil de Protection du routeur SNG que nous avons rédigé.

Le Profil de Protection du routeur SNG suit le plan imposé par la norme CC ITSEC. Dans une première partie introductive, les données relatives à l'identification du produit sont rappelées et complétées par les informations techniques relatives à la conception et à la mise en œuvre et utiles pour l'évaluation du logiciel de notre routeur.

Ensuite, le problème de sécurité propre au routeur SNG est exposé. Il y est notamment indiqué les parties sensibles du routeur (les données de configuration telles que les tables de routage statiques, les canaux de communication entre partitions...) avec le type de sensibilité (confidentialité des données, intégrité, authenticité...). Cette section est suivie d'une énumération des menaces pouvant altérer la sécurité du routeur, par exemple la corruption de la table de routage. Nous avons conduit pour cela une analyse des vulnérabilités du routeur SNG.

Dans un troisième temps, pour pallier ces menaces, nous avons énoncé les "objectifs de sécurité", spécifiant ce qui doit être fait pour rendre les menaces à l'encontre du routeur SNG inopérantes. Par exemple, il est imposé d'ajouter au routeur un mécanisme garantissant l'intégrité de sa table de routage.

Enfin, les objectifs de sécurité sont complétés par des Security Functional Requirements (SFR). Ces exigences de sécurité sont intégrées au SRS et viennent compléter l'ensemble d'exigences utilisées durant l'application de notre méthodologie au routeur SNG.

Chaque section de notre Profil de Protection est accompagnée d'une sous-section exposant la traçabilité bidirectionnelle avec la section précédente. L'évaluateur peut

ainsi vérifier que pour toute SFR, il existe au moins un objectif de sécurité associé et que tout objectif de sécurité est concrétisé par au moins une SFR.

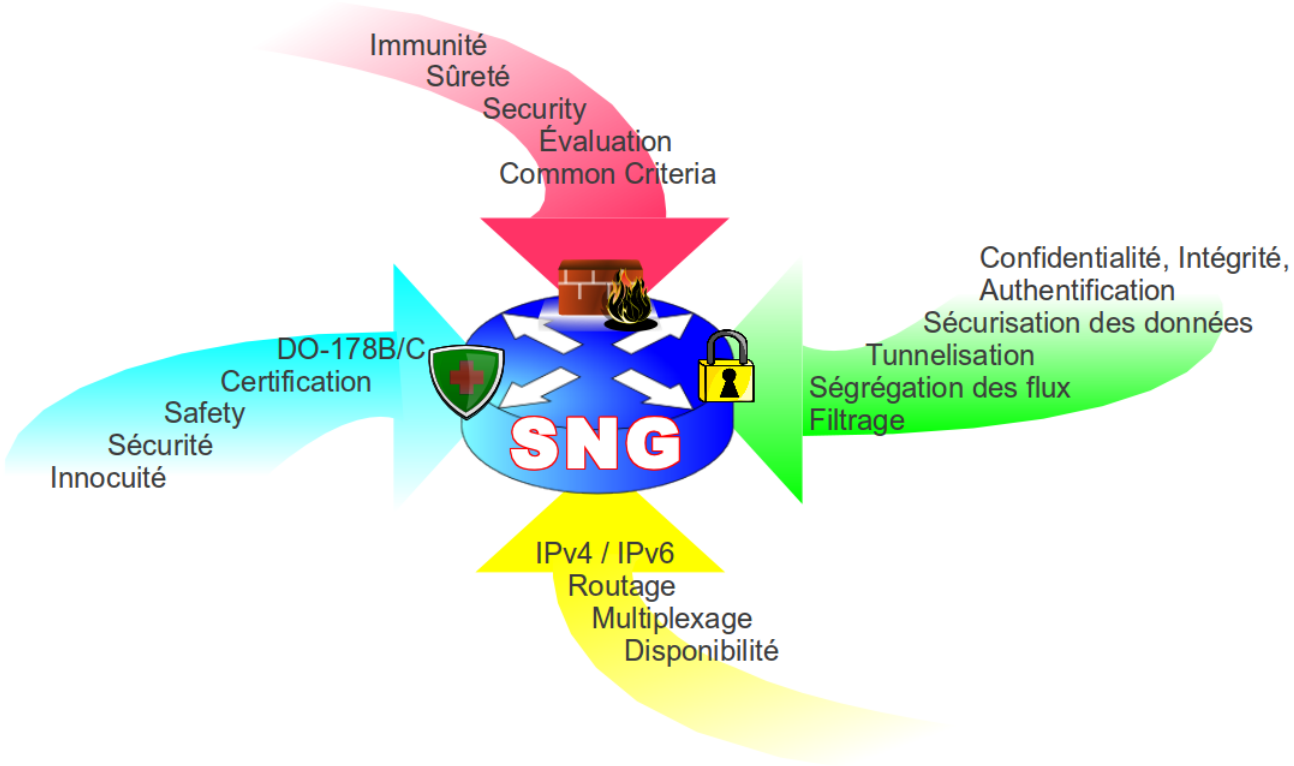
## 2.6. Résumé du chapitre

Dans ce chapitre ont été présentés les objectifs pour l'élaboration d'une nouvelle génération de routeurs embarqués destinés à l'aviation civile. Ces routeurs nécessitent non seulement des fonctionnalités de routage, mais aussi de filtrage et de sécurisation des communications sensibles. Les contextes potentiels d'utilisation de ces routeurs SNG ont été définis, ainsi que l'environnement systémique direct des routeurs.

L'aviation Civile encadre et contrôle le transport de passagers, où des vies humaines sont potentiellement mises en danger. Pour éviter et limiter les risques, différents domaines d'études interviennent dans la définition et la mise en œuvre du routeur : la certification garantit principalement la sécurité, c'est-à-dire l'innocuité du routeur, tandis que l'évaluation fait de même pour la sûreté, c'est-à-dire l'immunité du routeur. Ces domaines ont été présentés avec les taxonomies qui leur sont associés.

La figure 2.6 résume les quatre principaux domaines liés à la définition du routeur : fonctionnalités de routage, certification d'innocuité, évaluation d'immunité et fonctionnalités de sécurisation des flux de données.

Les contraintes de certification et d'évaluation nous ont amené à travailler sur le processus de développement à adopter pour le routeur SNG, ce qui nous a conduit à créer une méthodologie de développement rapide de logiciel avionique. Le prochain chapitre présente cette méthodologie et les gains qu'elle offre pour le développement de logiciel avionique en général, puis le chapitre suivant complète nos travaux en présentant un cas spécifique d'application : la mise en œuvre du routeur SNG avec notre méthodologie de développement.




**(a)** À la croisée des chemins


**(b)** Légende

**Figure 2.6.:** Le routeur SNG : à la croisée des chemins

# 3. La méthodologie de développement rapide en 7 étapes

La thèse dans laquelle s'inscrit le développement du routeur SNG est un cadre idéal pour expérimenter l'application de concepts novateurs. En effet, le développement classique de logiciel aéronautique est très encadré pour garantir des propriétés de sûreté de fonctionnement, comme expliqué section 2.4. Nous développons ici un routeur SNG initial qui, bien que destiné à l'aéronautique, a pour objectif la preuve de faisabilité de nouveaux concepts et non le développement de la version finale à certifier d'un système.

## 3.1. Motivations

C'est pourquoi dans cette première section nous commencerons par présenter le cycle en V, typique du cycle de développement des logiciels destinés à l'aéronautique. Ensuite nous présenterons ce qu'est la virtualisation et son application pour garantir des propriétés de sûreté et de sécurité. La seconde section introduit une nouvelle méthodologie, s'inscrivant dans le cadre du cycle en V et exploitant la virtualisation pour optimiser le cycle de développement, notamment réduire le temps de développement et les coûts de certification. La troisième section est dédiée à l'instanciation de la méthodologie avec les outils logiciels existants qui sont utilisés avec succès pour le développement du routeur SNG.

### 3.1.1. Présentation du cycle en V

Le développement logiciel a été dès l'avènement de l'informatique l'objet de nombreuses études afin de réaliser rapidement et avec le minimum d'effort un logiciel qui satisfasse les besoins pour lesquels il a été créé. Ainsi dans les années 80, un cycle de développement s'est imposé : le cycle en V, présenté figure 3.1. Il est ainsi nommé en raison de la forme qui le caractérise, la forme de la lettre V.

Ce cycle repose sur l'enchaînement des étapes suivantes.

1. Une analyse des besoins est menée avec le client et conduit à l'écriture d'un cahier des charges, informel par nature.

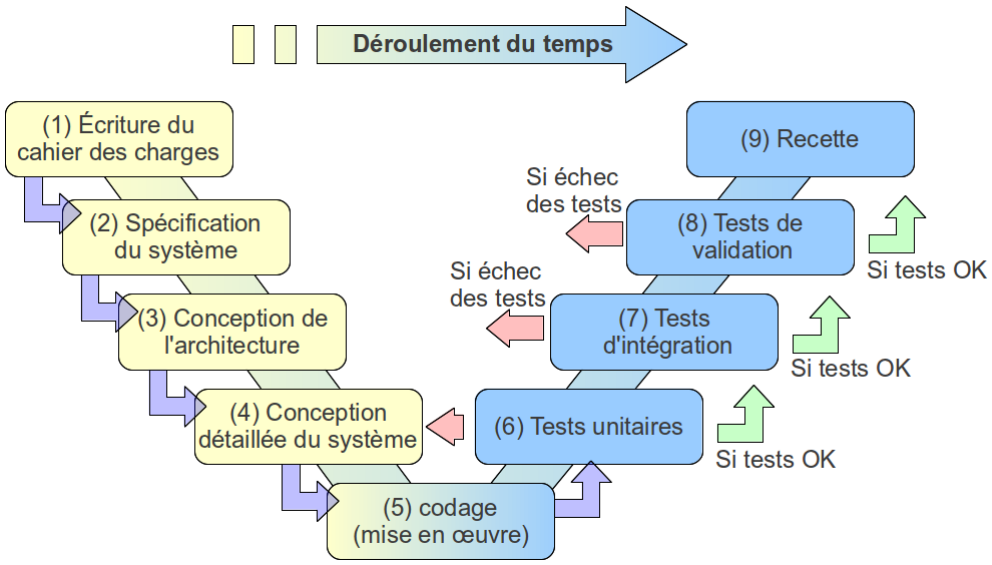



**Figure 3.1.:** Le cycle de développement en V

2. Le cahier des charges est retravaillé pour en extraire des «exigences», c'est-à-dire des paragraphes courts, formels, non ambigus, identifiés de manière unique, vérifiables (voir à cet effet les documents [IEEE830 1998, IEEE1233 1998]). Le recueil des exigences forme la «spécification» du système, document contractuel liant le client avec le créateur du logiciel.
3. La spécification est utilisée pour établir l'architecture du système, c'est-à-dire la décomposition logique en blocs et sous-blocs ainsi que leurs interactions. Chaque bloc est associé à une fonction, pouvant réaliser une ou plusieurs exigences de la spécification.
4. La spécification et l'architecture servent de base à la conception détaillée du système, où à chaque bloc de l'architecture est associé son algorithme et les paramètres associés.
5. La phase de codage consiste à traduire chaque bloc, sa fonction et son algorithme, dans un langage de programmation informatique.
6. La première phase de tests, dits « unitaires », est réalisée bloc par bloc. Le fonctionnement correct de la mise en œuvre de chaque bloc est vérifié indépendamment des autres blocs avec lesquels le bloc est destiné à interagir. Tant que les tests montrent des dysfonctionnement ou des insuffisances, le processus de développement doit être repris dès la phase 4 de conception détaillée et la phase 5 de codage doit être retravaillée. La validation de cette phase de tests unitaires conduit à la phase suivante de tests.
7. La deuxième phase de tests est appelée « tests d'intégration » car les vérifications portent sur le fonctionnement de l'architecture globale où sont intégrés les différents blocs. Y sont notamment vérifiés ici les interconnexions entre les

blocs. L'échec de tests dans cette phase peut conduire le processus à recommencer de cette étape 7 à l'étape 3 d'élaboration de l'architecture du système.

8. La troisième phase de tests, dite «de validation», consiste à vérifier le bon fonctionnement du système conformément aux spécifications. L'échec de cette phase peut entraîner la réécriture d'exigences et ainsi imposer de réitérer toutes les étapes 2 à 8.
9. Enfin, la dernière phase «de recette» s'achève par la validation du système par le client, qui vérifie que le système effectue bien ce pour quoi il a été créé. Cette phase peut montrer des insuffisances dans l'élaboration initial du cahier des charges, peut conduire à modifier ce dernier et donc à refaire l'ensemble du cycle de développement.

D'autres cycles de développement ont été proposés et appliqués dans les domaines non aéronautiques (citons les «méthodes agiles» [Vickoff 2009] telles que l'«extreme programming»[Jean-Louis Bénard 2004] et «Scrum» [Aubry 2010]) mais les fortes contraintes de certification de logiciels aéronautiques conduisent les industriels à leur préférer le cycle en V, bien éprouvé, adapté aux exigences de contrôle et de suivi du DO-178B (présenté dans la section 2.4), ce cycle étant connu des organismes de certification et donc mature en aéronautique.

La méthodologie que nous proposons et appliquons au routeur SNG couvre les étapes 3 d'architecture, 4 de conception détaillée et 5 de codage, pour automatiser certaines tâches associées à ces étapes et ainsi accélérer le processus de développement global. Elle repose sur l'usage de mécanismes de virtualisation, présentés ci-dessous.

### 3.1.2. La virtualisation

La virtualisation est un concept de fonctionnement informatique où l'exécution d'un logiciel est rendue indépendante de son support matériel. Indépendamment des mécanismes mis en œuvre, on qualifie généralement d'«hôte» la partie spécifique au matériel ou aux couches logicielles basses qui varient en fonction du système «réel» sur lequel est exécuté le logiciel. La partie «virtualisée», rendue «indépendante» de l'hôte, est appelée «invitée».

L'exemple typique de la virtualisation concerne les systèmes d'exploitation : la virtualisation permet d'exécuter un système d'exploitation «invité» au sain d'un autre système d'exploitation «hôte». Vu de l'hôte, le système invité virtualisé n'est qu'une application parmi les autres, à laquelle sont assignées des ressources matérielles et éventuellement des couches logicielles intermédiaires pour contrôler ces accès au matériel. Vu du système invité, le système d'exploitation dispose de l'intégralité des ressources de la machine «virtuelle» (VM) et y accède directement. L'illustration 3.2 montre une vue d'ensemble de ces entités.

La virtualisation apporte des avantages au fonctionnement des systèmes. Ainsi, l'hôte peut accueillir simultanément plusieurs invités, ce qui permet de mutualiser

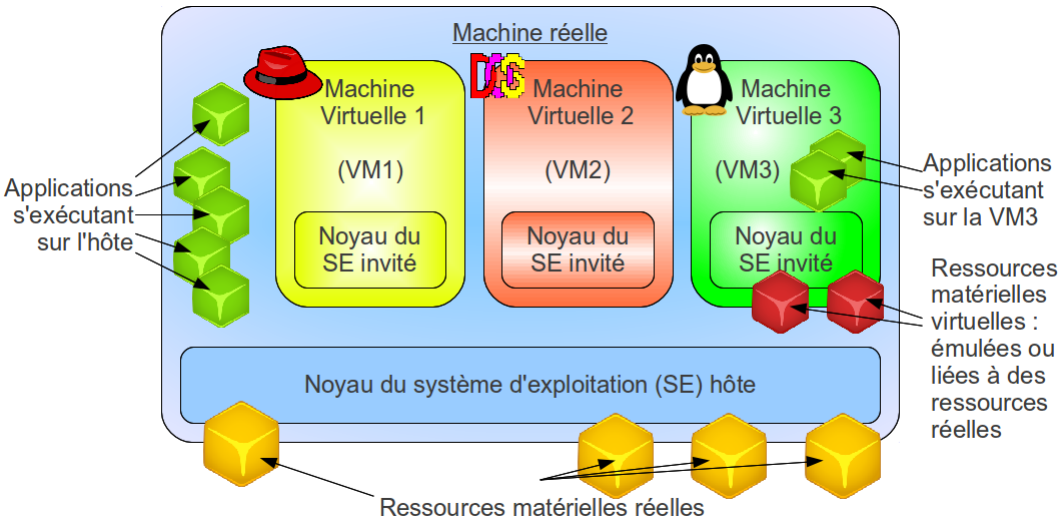



**Figure 3.2.:** Architecture d'un système virtualisé

les ressources pour une meilleure utilisation. De plus, des moniteurs logiciels peuvent contrôler le fonctionnement des systèmes invités et les stopper ou les redémarrer en cas de dysfonctionnement. Aussi, le cloisonnement entre systèmes invités évite la propagation des pannes et la pénétration d'un système par un autre. Enfin, l'indépendance de la machine virtuelle vis-à-vis de la machine réelle permet de déplacer un système invité d'un hôte à un autre (répartition de charge) sans que l'invité soit impacté par son portage.

Cependant, la virtualisation est souvent liée à une perte de performance du système virtualisé ou/et à des exigences de meilleures performances pour les systèmes hôtes. De plus, certaines techniques employées pour la virtualisation nécessitent d'adapter le système invité et ne garantissent pas l'isolation entre systèmes invités et système hôte.

Les techniques de mise en œuvre de la virtualisation sont complexes et sortent du cadre de cette thèse. Nous nous contenterons de renvoyer le lecteur intéressé à [Kenneth Hess 2010, Besson 2011, Bonnet 2008] pour plus d'informations. Dans le cadre du routeur SNG, nous utilisons un système d'exploitation hôte qui virtualise les applications, comme celui présenté section 3.3.6.

### 3.1.3. Le MILS : Diviser pour mieux régner sur un monde sûr

La ségrégation entre les invités, apportée par un virtualiseur, a conduit à l'utiliser pour durcir la sûreté des applications sensibles.

Le concept d'architectures «Multiple [and] Independent Levels of Security[/Safety]» (MILS) a ainsi émergé progressivement. Basée sur les travaux de concept de micronoyau de séparation de John Rushby [Rushby 1981], une architecture MILS ap-

porte des garanties de sûreté, avec un niveau d'assurance élevé, pour l'exécution de programmes sur une même infrastructure.

En effet, un principe fondamental de l'ingénierie consiste à décomposer une tâche complexe en plusieurs tâches plus simples [Descartes 1637]. En informatique, cela revient à **décomposer/diviser un logiciel en modules**. L'évaluation de la sûreté est ainsi simplifiée car l'évaluateur ne doit pas évaluer un système complet monolithique mais un ensemble de plus petits modules distincts et leurs couplages.

La mise en œuvre d'une solution de virtualisation pour réaliser le support d'une architecture MILS est acceptable s'il est prouvé qu'il garantit les quatre propriétés intrinsèques du MILS :

- La solution est incontournable, c'est-à-dire qu'un invité ne peut pas communiquer sans passer par les contrôles de sécurité imposés par le système hôte.
- La solution est évaluable, il est possible de prouver formellement le fonctionnement correct et valide du système de virtualisation (donc de l'hôte).
- La solution est toujours appelée, le contrôle des communications ne s'effectue pas seulement au premier message mais pour tout message échangé.
- La solution est résistante aux altérations, elle empêche toute modification qu'elle n'a pas autorisée explicitement.

Ces propriétés sont garanties par une évaluation de la sûreté de la solution [Jim Alves-Foss & Taylor 2005, USA 2007]. Cette évaluation étant complexe même pour des «petits» systèmes, elle ne peut aboutir que pour des systèmes minimalistes (microsystèmes), dédiés à la virtualisation de systèmes et à leur séparation, appelés «micronoyaux de séparation». Ces noyaux de séparation assurent la mise en œuvre des concepts de séparations temporelle et spatiale entre programmes, ainsi que le contrôle des flux d'information.

Le noyau assure que chaque programme et sa machine virtuelle pourront utiliser les ressources réelles durant les temps qui leur sont assignés. Un programme ne peut empiéter sur un autre et lui «voler» son temps d'exécution ; on parle alors de séparation temporelle entre les deux VM.

Le noyau garantit aussi qu'une ressource réelle ne puisse être attribuée simultanément à deux machines virtuelles. Les espaces d'adressages de la mémoire et des entrées/sorties sont partagés durant la configuration du noyau de séparation, on parle de partitionnement de l'espace d'adressage. Ensuite, le noyau vérifie à l'exécution à chaque accès que l'adresse accédée par la machine réelle lui a bien été assignée et bloque la tentative d'accès le cas échéant. Le noyau assure ainsi la séparation spatiale entre VM.

De la même manière, les VM peuvent disposer de canaux dédiés pour communiquer entre elles, canaux gérés par le noyau de séparation (et non directement par l'électronique sous-jacente). Le noyau assure alors un contrôle de forme des communications : vérification de la longueur maximale des messages envoyés, autorisation des accès, estampillage des messages reçus...

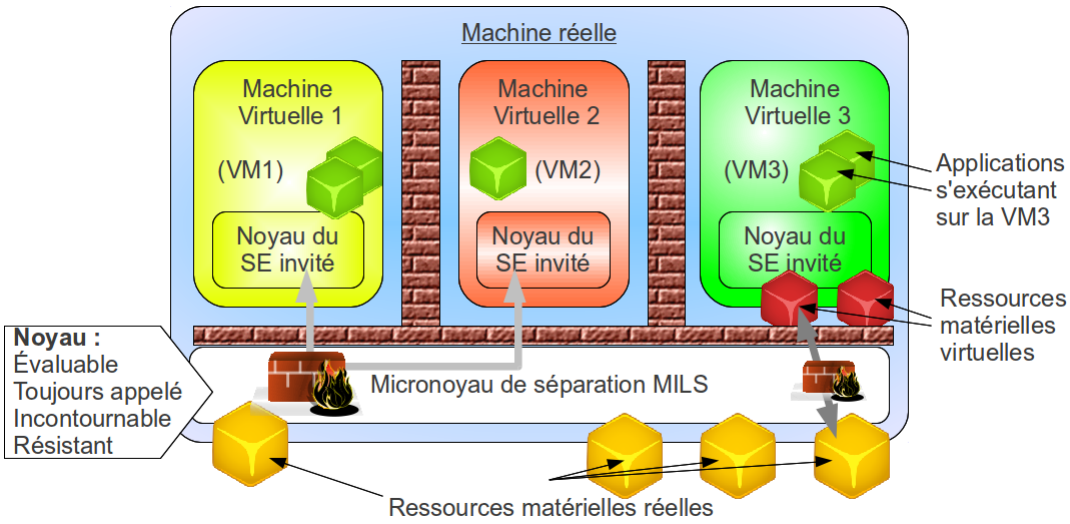



**Figure 3.3.:** Architecture d'un système MILS

Comme illustré par la figure 3.3, dans un système MILS, chaque machine virtuelle s'exécute d'une manière indépendante de ses homologues. Vu depuis le noyau de séparation hôte, chaque VM dispose de son sous-espace d'adressage réservé. L'ensemble de ces sous-espaces est une partition (au sens mathématique) de l'espace global d'adressage de l'hôte. Chaque VM dispose de créneaux temporels dédiés, périodiquement répétés, l'ensemble formant une partition du temps. C'est pourquoi, dans le contexte d'une architecture MILS, le terme «partition» désigne naturellement les ressources temporelles et spatiales associées à une machine virtuelle. Le noyau de séparation est alors, à l'exécution du système, l'ordonnanceur des partitions et le contrôleur incontournable des accès aux ressources.

Un système MILS doit garantir certaines propriétés [Uchenick 2009b, Uchenick 2009a].

- Concernant les flux d'information inter-partitions, seules les sources autorisées peuvent générer les informations et celles-ci ne sont délivrées qu'aux destinataires explicités et autorisés.
- Les données d'une partition ne sont accessibles qu'aux programmes associés à cette partition, elles sont isolées des données d'autres partitions. Les données privées restent privées, sans possibilité d'infiltration (lecture depuis une autre partition) ni d'exfiltration (écriture de données d'une autre partition). Cela impose pour chaque partition de disposer de son espace d'adressage dédié, où une adresse n'a de sens que pour la partition qui l'utilise.
- Le processeur lui-même ne doit pas permettre à des informations de passer d'une partition à une autre, que ce soit par les caches matériels ou même par des mesures de délais de traitement. Par exemple, il existe des attaques basées sur l'analyse des comptages de cycles processeurs pour extraire de l'information sur les clefs cryptographiques, comme exposé dans [Kocher 1996]. Un système MILS doit contrer, dans la mesure du possible, ces classes d'attaques.

– Le dysfonctionnement d'une partition ne doit pas impacter les autres partitions. Il doit être détecté, contenu et corrigé.

Les systèmes MILS, utilisés dans des systèmes d'information militaires et civils, intéressent de plus en plus le monde aéronautique [Huyck 2010]. Celui-ci a cependant déjà abouti à utiliser la virtualisation, avec un objectif subtilement différent : la mutualisation de matériel, avec des garanties élevées de sécurité (sûreté de fonctionnement). C'est l'objet de la sous-section suivante.

### 3.1.4. L'IMA : Diviser pour mieux régner sur un monde aéronautique sécurisé

Le domaine de la recherche et développement en systèmes électroniques et informatiques pour l'aéronautique a toujours eu comme objectifs de réduire les coûts, les consommations électriques et les masses de ses équipements. Dans les années 1990, la complexité croissante des systèmes a entraîné la mutation des systèmes avioniques d'un ensemble de systèmes coexistants et indépendants à un ensemble de systèmes communicants fédérés, insérant ainsi la notion de réseaux de données embarqués.

Dans les années 2000, l'aviation civile a à nouveau franchi un cap conceptuel : les systèmes fédérés sont efficaces mais leur complexité croissante rend leur évaluation et leur certification trop complexes pour aboutir avec des coûts et des temps raisonnables. De plus, les systèmes logiciels mais aussi matériels se sont généralisés et sont devenus beaucoup plus nombreux dans les nouvelles générations d'aéronefs, entraînant ainsi une augmentation des consommations et du poids de matériel embarqué.

La puissance et la maturité des calculateurs embarqués sont devenues dans les années 2000 suffisantes pour envisager de mutualiser sur un même système matériel embarqué plusieurs applications logicielles distinctes et autonomes. Les avionneurs ont ainsi à nouveau muté d'une architecture avionique modulaire **fédérée** à une architecture avionique modulaire **intégrée**. Là où à chaque calculateur était associée une unique fonction dans l'architecture fédérée, l'architecture intégrée permet au calculateur modulaire embarqué de fournir plusieurs fonctions. Cette mutualisation du matériel permet ainsi de réduire le nombre de calculateurs sans limiter le nombre d'applications embarquées. Elle équivaut à la mise en œuvre du concept de virtualisation dans les systèmes avioniques.

Intitulée "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations", la norme RTCA DO-297 [DO-297 2005] du 8 novembre 2005 encadre la conception et la mise en œuvre des systèmes pour les architectures avioniques modulaires intégrées dans l'aéronautique civile. Préparée par le groupe Special Committee 200 (SC-200), cette norme définit l'IMA comme « un ensemble partagé de matériels flexibles, réutilisables et interconnectés et de ressources logicielles qui forment une plateforme fournissant des services, conçus et vérifiés pour un ensemble défini d'exigences de sécurité et de performances, exigences appliquées par des

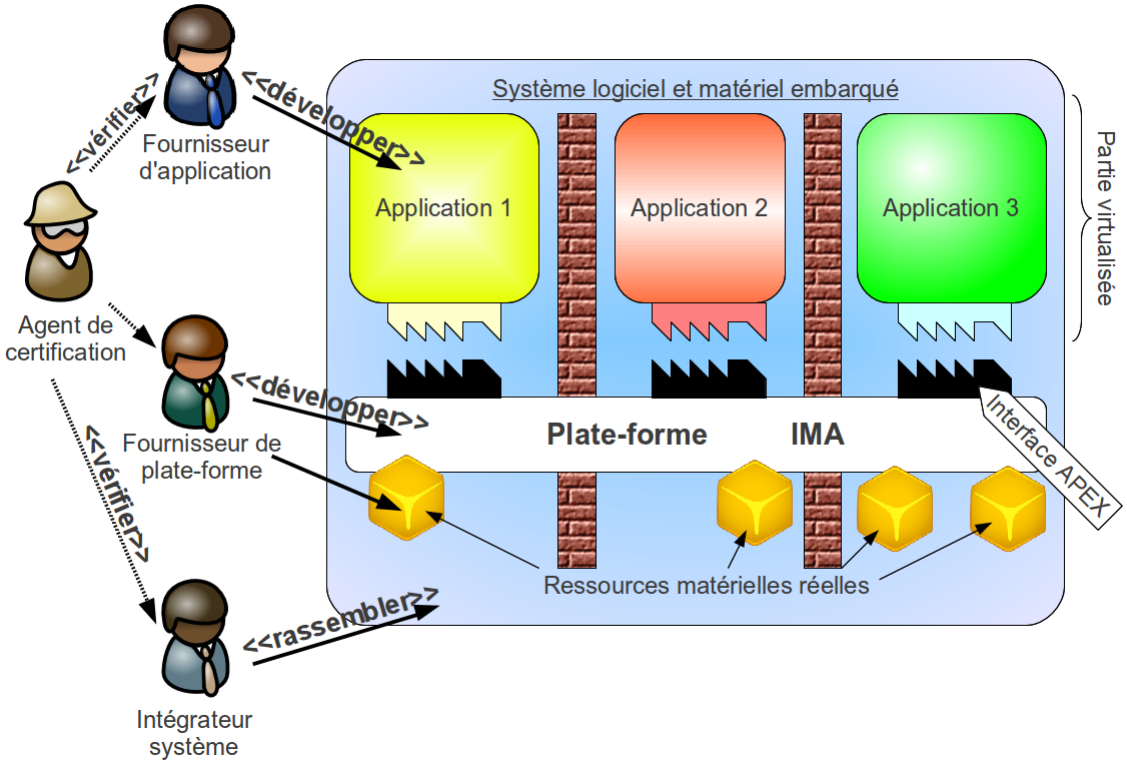



**Figure 3.4.:** Architecture d'un système IMA

applications hébergées [sur le matériel] qui fournissent les fonctions avioniques ». Cette norme explicite et partitionne les rôles des différents fournisseurs de modules IMA : fournisseurs d'applications, fournisseurs de plateformes IMA, intégrateurs des systèmes et agents de certification.

ARINC a publié en 1996 l'ARINC 653, un standard décrivant une interface de programmation et de configuration, afin de permettre d'augmenter l'indépendance des applications avioniques vis-à-vis du matériel. Le standard porte le nom d'ARINC 653 APEX (Application Executive). En 2003 a été publiée une mise à jour introduisant l'IMA. La dernière publication [ARINC653P1-3 2010, ARINC653P2-2 2012, ARINC653P3 2006, ARINC653P4 2012] de 2006 complète la version précédente en clarifiant certains points.

Comparativement aux concepts liés à la virtualisation, la définition d'une application IMA est analogue à une application virtuelle invitée, de même que la plateforme IMA correspond au noyau hôte et au matériel réel. La certification et l'intégration système de l'hôte avec les applications invitées sont cependant traitées spécifiquement au domaine aéronautique.

La figure 3.4 résume la notion d'architecture IMA et ses différents blocs, ainsi que les rôles des différents acteurs de la construction d'un système modulaire intégré.

Ainsi, les architectures fédérées sont constituées d'un grand nombre de calculateurs

ayant chacun une fonction, tandis que les architectures intégrées sont constituées d'un petit nombre de calculateurs ayant chacun un petit nombre de fonctions. Globalement, les deux types d'architectures fournissent le même nombre de fonctions mais la division du nombre de calculateurs permet de réduire les coûts de la certification avionique.

La virtualisation au niveau logiciel sur laquelle se base l'IMA apporte aussi des possibilités de gains en terme de réutilisation, de maintenance et de remplacement des modules logiciels mais aussi des modules matériels (appelés «Line Replacable Unit» ou LRU).

À l'instar du MILS présenté à la section précédente, la norme DO-297 utilise aussi le terme de partition. Dans le contexte IMA, une partition est «une allocation de ressources dont les propriétés sont garanties et protégées par la plateforme des interactions et de l'influence de l'extérieur de la partition».

Le partitionnement est qualifié de «robuste», d'après le DO-248 [DO-248B 2001, DO-248C 2012], si les quatre conditions suivantes sont vérifiées :
– la partition logicielle n'est pas autorisée à contaminer le code, les entrées/sorties et les données des autres partitions,
– elle n'est autorisée à exploiter les ressources processeurs que dans les temps d'exécution qui lui sont allouées,
– elle n'est autorisée à exploiter que les ressources qui lui sont allouées,
– et enfin les défaillances matérielles ne doivent pas impacter les partitions non allouées à ces ressources défectueuses.

Ainsi, le concept IMA diffère très peu du concept MILS développé à la section précédente : les deux imposent des limitations contraignantes en terme de séparations spatiales et temporelles entre partitions. Cependant, l'IMA n'impose pas explicitement des espaces d'adressages indépendants mais se «contente» de mécanismes de contrôles d'accès. Par contre, l'IMA gère la défaillance matérielle partielle en imposant la continuité des partitions non impactées, contrairement à l'approche purement logicielle du MILS qui suppose que le matériel sous-jacent ne peut défaillir (plus exactement, la problématique du matériel n'est pas abordée par les concepts théoriques du MILS, mais doit l'être par le concepteur du système).

En pratique, les systèmes d'exploitation MILS et IMA sont très proches et diffèrent par les processus utilisés pour garantir leurs fonctionnalités : les systèmes IMA sont conçus pour la sécurité et destinés à être utilisés dans des systèmes certifiés, tandis que les systèmes MILS sont reconnus pour la sûreté et destinés à être utilisés dans des systèmes évalués.

Le lecteur intéressé peut consulter [Almeida José 2010, Gaska *et al.* 2010, Inc 2008, Canu *et al.* 2011] pour avoir plus d'informations sur le concept IMA, les solutions apportées et les problématiques restantes. Certains fournisseurs proposent des systèmes à la fois compatibles IMA et MILS. Ce sont ces systèmes que nous exploitons dans la méthodologie présentée dans la section suivante.

## 3.2. Contribution à l'amélioration du processus

Dès le début de ma thèse s'est posée la question du processus de développement à mettre en œuvre dans le cas du routeur SNG, ce processus devant permettre un développement rapide et efficace tout en permettant d'accéder aux plus hauts niveaux d'assurances de sûreté et de sécurité. La méthodologie que nous avons mise au point est présentée dans cette section, elle a été conçue pour être non seulement utilisée pour le routeur SNG mais aussi suffisamment générique afin de permettre le développement de tout autre logiciel avionique, non nécessairement liés aux réseaux de communication informatiques.

### 3.2.1. Les sept étapes démystifiées

La méthodologie que nous proposons est basée sur une adaptation du cycle de développement en V présenté section 3.1.1, notamment les étapes 4 à 7 de l'élaboration de l'architecture aux tests d'intégrations. Nous utilisons simultanément des transformateurs de modèles en code pour accélérer la phase de codage et une infrastructure logicielle de virtualisation compatible MILS et IMA.

L'application de la méthodologie utilise et raffine [1] la spécification du système à développer, les exigences indiquant ce que doit faire le système doivent donc être écrites au préalable.

La première étape de la méthodologie présentée figure 3.5 est le partitionnement : l'architecte divise le futur logiciel en différentes «classes de partitions», chacune étant associée à un sous-ensemble des fonctionnalités du système (et donc à un sous-ensemble des exigences fonctionnelles). Le logiciel final exécutera un ensemble d'instances de ces classes de partition. Les instances de partitions peuvent communiquer entre instances de même classe ou instances de classes différentes. Ainsi, quand la conception d'une fonctionnalité conduit à développer deux classes de partitions, le service rendu par le produit est basé sur l'exécution d'au moins deux instances de partitions (mais il peut y en avoir potentiellement des centaines) et sur les interconnexions de données entre instances.

Dans la seconde étape, le concepteur réalise un modèle par classe de partition. Un modèle peut être mis en œuvre directement en C mais la méthodologie préconise l'utilisation de modèles graphiques, qui augmentent le niveau d'abstraction, facilitent la compréhension générale de la conception du produit, permettent de vérifier plus efficacement certaines propriétés du code (telle que la couverture complète du code) et de s'assurer de propriétés de sûretés et de sécurité. La figure 3.6 présente la complémentarité entre l'architecte et le concepteur.

1. Raffiner : action de réécrire un code d'un niveau d'abstraction vers le niveau de concrétisation suivant

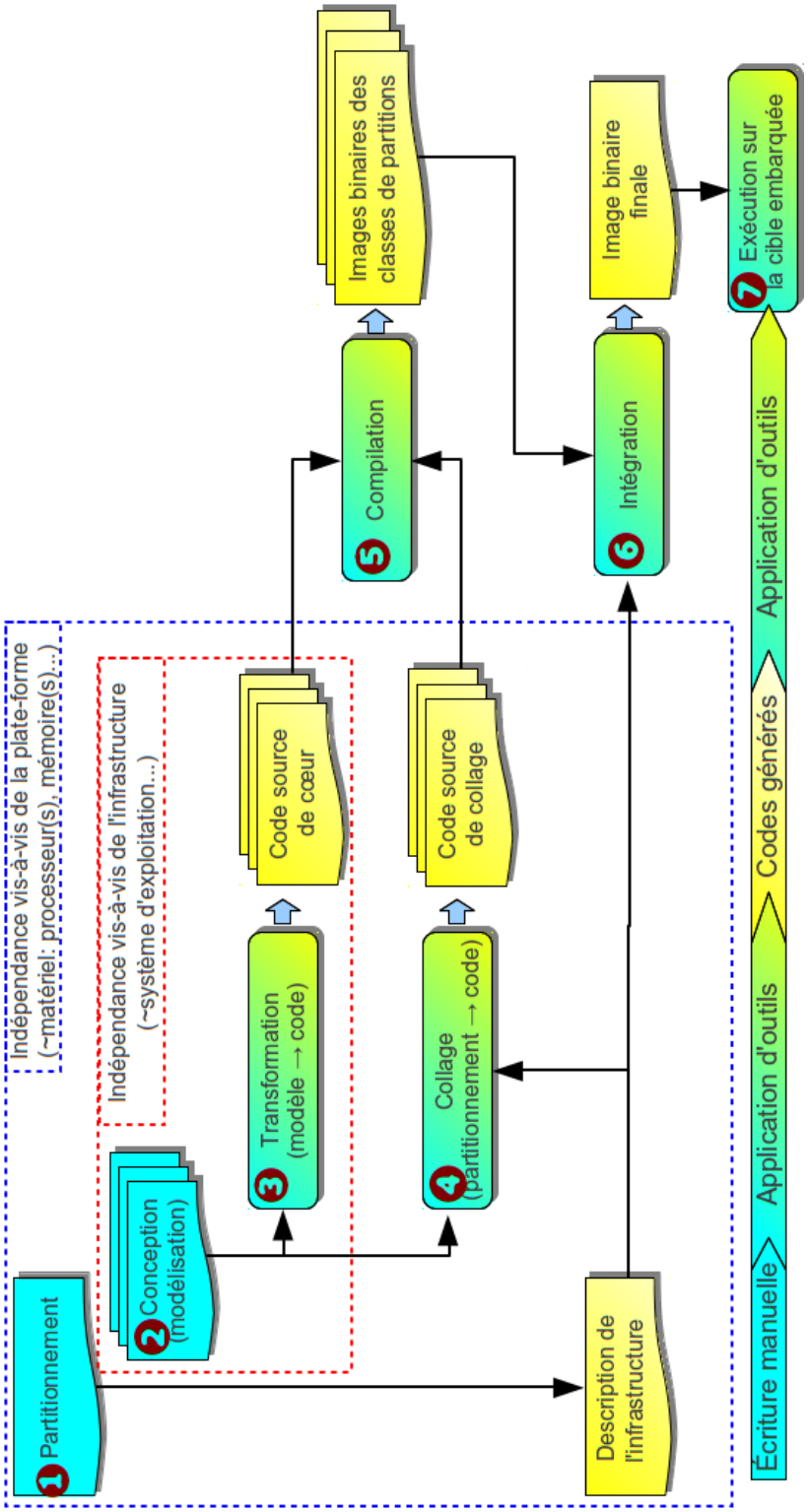



**Figure 3.5.:** La méthodologie de développement

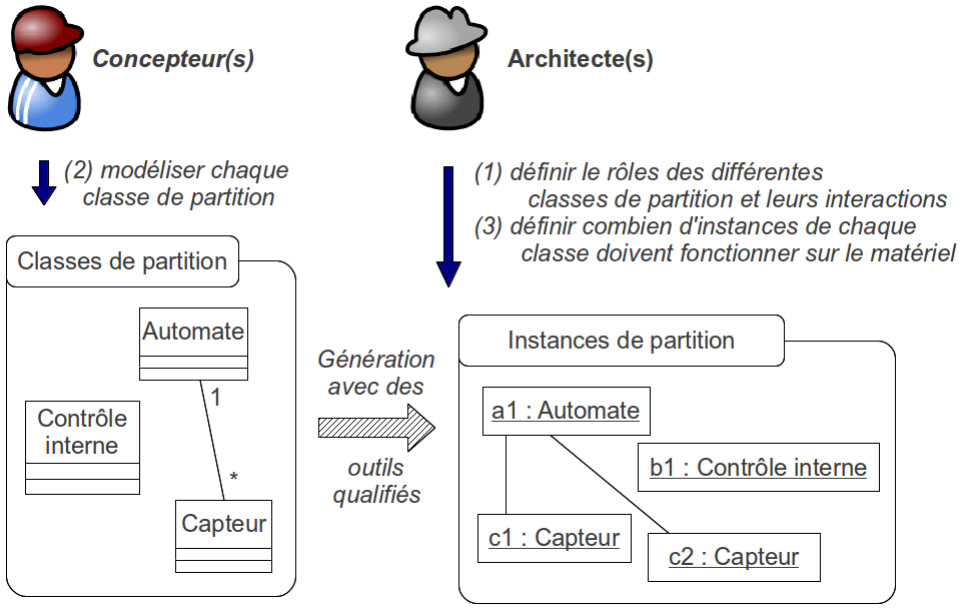



**Figure 3.6.:** Rôles du concepteur et de l'architecte

La troisième étape de la méthodologie consiste à transformer chaque modèle de classe de partition en un code que nous appelons «code source de cœur». Il est important de noter que le code généré ne peut être à cette étape compilé, lié et testé directement sur la cible, car le code de cœur a un point d'entrée par fonction interne et rien n'indique quel est celui à appeler, contrairement à un code source applicatif qui a un point d'entrée unique (par exemple la fonction «main» en langage C).

Durant la quatrième étape, le «code source de collage», liant le code source de cœur à l'infrastructure de virtualisation, est écrit ou généré automatiquement. Ce code source relie les entrées/sorties fournies par l'interface de programmation du micronoyau de séparation avec celles du code source de cœur. Ce code source fournit aussi à l'infrastructure de virtualisation un point d'appel unique qui est mis en œuvre par une routine qui appelle en boucle les fonctions internes adéquates du code de cœur. L'algorithme lié à cette routine est présenté plus bas section 3.3.4.

Les cinquième et sixième étapes consistent à compiler ensemble les codes source de cœur et de collage pour former le code binaire de toutes les classes de partition puis à intégrer ensemble dans une image binaire finale tous les binaires de classes de partition, le code binaire du noyau de séparation et sa configuration. L'étape 1 manuelle d'architecture permet de guider les outils pour la génération automatisée du fichier binaire final.

Les développeurs du logiciel peuvent alors télécharger l'image binaire finale sur la cible embarquée de tests ou sur un émulateur, et effectuer les vérifications et validations pour confirmer le fonctionnement correct et voulu du système et ses performances.

Les sept étapes sont résumées ci-dessous :

1. Partitionnement (manuellement par l'architecte)
2. Modélisation (manuellement par les concepteurs)
3. Transformation (des modèles aux codes source de cœur)
4. Collage (du partitionnement aux codes source de collage)
5. Compilation (des codes sources de cœur et de collage aux codes binaires des partitions)
6. Intégration (des codes binaires des partitions à une image binaire)
7. Exécution (de l'image binaire sur émulateur ou cible réelle)

Il est important de remarquer que les étapes de modélisation et de transformation sont indépendantes du matériel (processeur, mémoires vives et de masses, entrées/-sorties électroniques...) et de l'infrastructure de virtualisation (notamment du système d'exploitation et de son noyau de séparation). Cette généricité du code de cœur est rendue possible grâce à l'étape de collage dont le rôle est justement d'écrire le code adéquat liant le code générique à l'infrastructure concrète.

De la même manière, les considérations spécifiques au matériel tels que l'adressage mémoire et le choix du processeur n'impactent que les étapes 5 à 7 de compilation, d'intégration et de tests. Les étapes d'architecture et de modélisation en sont indépendantes.

### 3.2.2. Avantages de cette méthodologie

Cette méthodologie réduit la charge du processus de développement logiciel de différentes manières.

Les modèles de haut niveau permettent aux concepteurs de vérifier et de corriger le comportement du système «au plus tôt», dès les premières phases de développement, réduisant ainsi les risques de divergences à la recette [2] et les coûts associés de correction ou d'évitement des dysfonctionnements et écarts.

La décomposition fonctionnelle en classes de partitions permet de simplifier l'étape de conception, suivant le principe présenté section 3.1.3 [Descartes 1637]. Elle permet aussi d'isoler les fonctionnalités spécifiées de manières incorrectes (par exemple dans le cas de contradictions entre exigences) ainsi que celles spécifiées de manières trop imprécises.

Cette décomposition facilite aussi les phases d'évaluation et de certification : la garantie de sûreté et de sécurité est plus facile à apporter pour des systèmes modulaires que pour les systèmes monolithiques équivalents [Jim Alves-Foss & Taylor 2005].

Le concept de classes de partitions permet de concevoir des classes de partitions de déverminage, pour instrumenter le code. Par exemple, l'équipe de développement

2. La «recette» désigne la dernière étape où le système est évalué par le client.

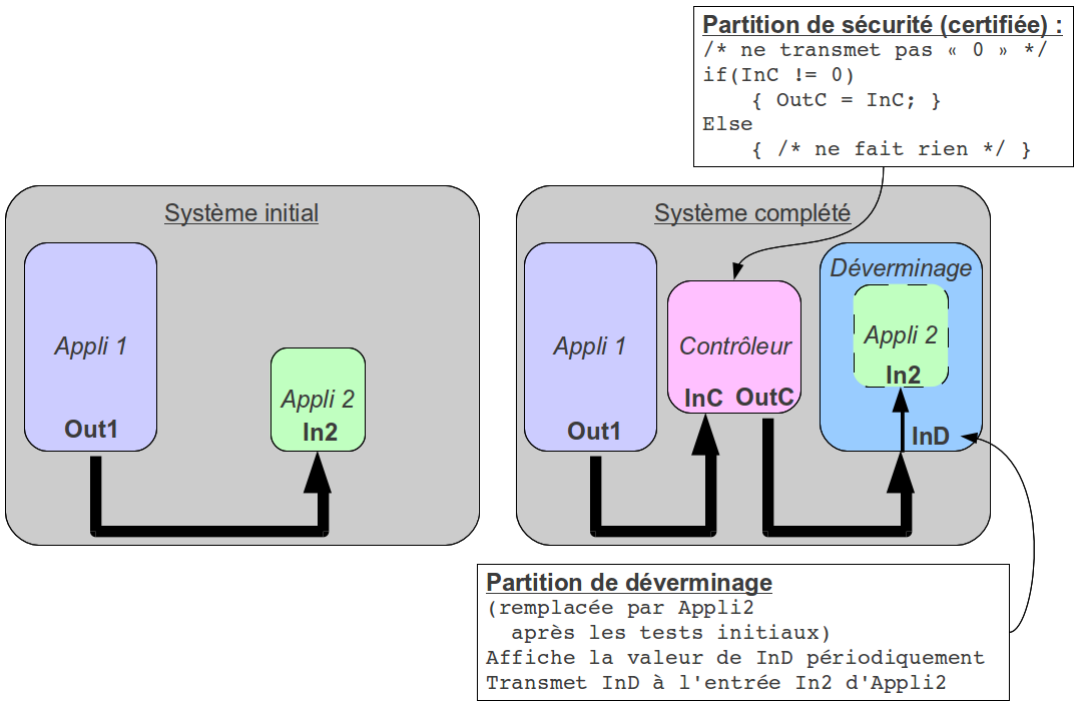



**Figure 3.7.:** Ajout de partitions de sécurisation et de déverminage

peut facilement concevoir une classe de débogage associée à une classe opérationnelle qui sera celle certifiée, avec les mêmes entrées/sorties. En interne, la classe de débogage est reliée à la classe opérationnelle mais ajoute des fonctionnalités telles que l'affichage des appels de fonction, les valeurs des entrées/sorties, etc... Les interfaces d'entrée/sorties étant similaires, la classe de déverminage peut être ainsi utilisée à la place de la classe opérationnelle sans impact sur l'architecture et les autres instances de partitions.

De la même manière, des classes de partitions pour la sûreté et la sécurité peuvent être insérées dans l'architecture pour assurer des contrôles de cohérences des entrées/sorties, de vérification du comportement d'autres classes de partitions, pour corriger des vulnérabilités non détectées durant l'élaboration des versions précédentes du produit... L'adaptabilité apportée par ces classes augmente les assurances de sûreté et de sécurité du produit. Ces deux derniers points sont illustrés par la figure 3.7.

La méthodologie peut être automatisée de l'étape 3 de transformation du code à l'étape 6 d'intégration pour produire l'image binaire finale. Cette automatisation, en plus de permettre une réduction drastique du temps de développement, une reproductibilité parfaite et un déterminisme demandé par les processus de certification, peut s'accompagner de la qualification des outils utilisés aux différentes étapes. Dans ce cas, les tâches exigées pour la certification du système sont réduites voir supprimées en conséquence. Un exemple de cette réduction du coût de la certification est donné dans la table 3.1 avec le choix des outils dans la section 3.3.9.

La génération automatique de la plus grande partie du code source à l'étape 3 de transformation réduit l'occurrence des dysfonctionnements liés à des tâches répétitives et les bogues liés au copier/coller. Par exemple, avec la méthodologie, les programmeurs n'ont plus à gérer manuellement certaines tâches de bas niveau telles que le contrôle des accès aux mémoires partagées ou la validation des calculs car la sémantique des modèles de haut niveau et les différentes étapes de transformation du modèle au binaire assurent ensemble ces tâches répétitives.

Par exemple, une division par zéro conduit le processeur à une faute potentiellement fatale pour l'exécution du logiciel et doit donc impérativement être évitée. Le transformateur peut générer un code source dans lequel chaque division est complétée systématiquement d'une vérification de non nullité du diviseur, cette vérification de type préconditionnelle évitant la panne.

De plus, l'introduction de canaux cachés est rendue plus difficile en raison de l'utilisation d'un système d'exploitation vérifiant systématiquement toutes les entrées et sorties des instances de partition. Ainsi mettre en place un canal caché nécessite de modifier à la fois la configuration du système d'exploitation et les deux applications (ou l'application et le matériel associé) qui communiquent.

Le transformateur étant constitué fondamentalement d'un ensemble de règles de conversion du modèle au code, il est possible de prouver que le code qui est produit automatiquement est conforme aux exigences de qualités et aux standards de développement, rendant inutile la vérification manuelle de la qualité du code généré.

Cependant, une erreur de codage du transformateur peut conduire à partir d'un modèle valide à la production de code source invalide, où l'erreur est dupliquée un grand nombre de fois, invalidant l'intégralité du code source généré. C'est pourquoi les transformateurs doivent passer une phase de qualification de l'outil, apportant des garanties élevées de sécurité du code généré.

Le code de cœur est indépendant de l'infrastructure de virtualisation utilisée et du matériel sous-jacent. Cette abstraction facilite la tâche du concepteur qui n'a ainsi pas besoin de compétences spécifiques dans ces domaines et peut se spécialiser sur la modélisation et les algorithmes liés.

Cette indépendance du code de cœur vis-à-vis de l'infrastructure apporte un avantage non négligeable de portabilité : un même code de cœur peut être réutilisé sans modification avec différents systèmes d'exploitation, seul le code de collage doit être réécrit ou régénéré, or ce dernier représente une part minoritaire du code source compilé. Par exemple, dans le cas de la classe de partition du routeur SNG assurant le routage, le code de collage représente 350 lignes de code (commentaires et lignes vides comprises), soit 1,25% du total de lignes de code pour cette classe de partition. Il est ainsi possible de comparer rapidement et avec peu d'efforts les performances du code sur différentes infrastructures de virtualisation.

Cette indépendance facilite aussi la réutilisation du code : la classe de partition peut être importée d'un projet original à un autre projet, dans le cas du développement

d'une nouvelle version ou même d'un nouveau système assurant d'autres fonctions et ayant en commun avec le système original la sous-fonction assurée par la classe de partition réutilisée.

Les codes source (code de cœur et code de collage) sont indépendants du matériel : ce sont les étapes de compilation et d'intégration qui vont produire des codes binaires dépendant du matériel. Cette indépendance vis-à-vis de la plateforme matérielle apporte encore des avantages en terme de portabilité et de réutilisation. Ainsi, à partir du partitionnement et des modèles conçus pour le routeur SNG sur cible matérielle embarquée, destinée à être certifiée, nous avons reconstruit facilement une version dérivée s'exécutant sur un ordinateur de bureau avec un processeur Intel® 64 bits ; cette version dérivée n'est pas destinée à être embarquée mais à servir en tant que prototype démonstrateur des fonctionnalités du routeur.

La modélisation peut être complétée partiellement par du code source écrit manuellement. Cette étape peut aussi être remplacée complètement par l'écriture manuelle du code source de cœur. Dans ce dernier cas, l'écriture remplace aussi l'étape suivante de transformation. Cette liberté apportée aux développeurs du logiciel annule certains gains pour la certification, mais elle permet à l'entreprise de réutiliser des codes sources programmés antérieurement. Cela peut être aussi indispensable dans le cas où l'infrastructure de virtualisation ne fournit pas d'abstraction adéquate pour accéder à certaines ressources matérielles spécialisées. Dans ce cas, il peut être préférable d'écrire les pilotes de périphériques directement à la main dans des langages de bas et moyen niveaux (au sens « proches du matériel») tels que l'assembleur et le langage C. Nous préconisons toutefois d'architecturer le projet de développement en faisant en sorte d'isoler ces sections manuellement codées dans des classes de partition dédiées. Ainsi, les changements matériels des versions ultérieures n'impacteront que ces classes de partition.

L'utilisation d'une infrastructure de virtualisation sûre et sécurisée, compatible MILS et IMA, garantit la ségrégation entre les différentes instances de partition et contrôle toutes les interconnexions. Cela permet ainsi de réduire les vecteurs de vulnérabilité du système.

De plus, le contrôle fort des communications entre les instances des partitions est associé à un faible nombre de types de communications. Par exemple dans les systèmes IMA, il est possible de disposer de milliers de canaux de communication, mais seules deux catégories de canaux existent : les «message queues» et les «sample queues», la première étant une file finie de messages datés et de taille bornée où les nouveaux messages s'ajoutent à la file, la deuxième étant un tampon mémoire pouvant accueillir un seul message à la fois et où le nouveau message remplace l'ancien. Cela facilite le remplacement d'une instance de partition logicielle entière par une solution matérielle, sans impacter les autres instances.

Enfin, les instances des partitions sont destinées à s'exécuter simultanément. La décomposition fonctionnelle de la première étape oriente l'architecte logiciel vers la conception d'un système modulaire et parallélisé. Cette parallélisation de

l'exécution des modèles associés aux fonctions ouvre des perspectives encore peu explorées en aéronautique. Bien qu'actuellement, les systèmes soient généralement basés sur des plateformes matérielles mono-processeur mono-core (historiquement afin de garantir le déterminisme de l'exécution séquentielle), cette parallélisation pourrait profiter pleinement des processeurs multi-core voir même de processeurs massivement multi-core (voir à cet effet les études d'applicabilité des General-Purpose computing on Graphics Processing Units GPGPU dans l'avionique : [Gallet & Brinton 2011, Agrou *et al.* 2011]) pour permettre un saut dans l'ordre de grandeur des performances des futurs systèmes conçus en suivant cette méthodologie. Ainsi notre méthodologie prépare l'évolution vers une nouvelle génération de plateformes, encore à créer, sur laquelle chaque instance de partition serait associée à un cœur dédié.

Nous venons de voir les principes généraux de la méthodologie présentée dans cette section, ainsi que les gains qu'elle apporte, indépendamment des outils logiciels utilisés pour mettre en pratique cette méthodologie. La section suivante présente quelques outils qu'il est possible d'utiliser pour instancier la méthodologie et rendre ainsi concret le processus de développement du logiciel.

## 3.3. Choix d'outils pour l'application de la méthodologie au routeur SNG

À chacune des sept étapes de la méthodologie peuvent être associés un ou plusieurs outils logiciels. La suite d'outils est alors appelée « chaîne d'outils » car la méthodologie utilise les sorties des premiers outils comme données d'entrée des outils suivants. Cette section commencera par décrire pas à pas la chaîne d'outils qui a été utilisée pour réaliser le routeur SNG. Ensuite une variante constituée entièrement de logiciels libres et Open Sources complétera la section.

### 3.3.1. Organisation architecturale du routeur SNG

La première étape de la méthodologie consiste à partitionner les fonctionnalités du routeur. Pour cette étape manuelle, nous n'avons pas utilisé d'outils autres qu'un traitement de texte (LibreOffice [LibreOffice 2013]) afin de répartir les spécifications en blocs destinés à correspondre à des classes de partition. Le logiciel libre « dia [Dia 2013] » a permis de réaliser rapidement un diagramme de classe résumant l'architecture, c'est-à-dire dans notre cas l'ensemble de classes de partitions et leurs interactions. Ce diagramme est représenté dans la figure 3.8 et commenté dans le chapitre suivant dédié à la mise en œuvre du routeur SNG. Contentons-nous pour l'instant d'énoncer que les exigences fonctionnelles ont été regroupées dans trois classes de partitions : une classe «**Pfr**» («Partition de Filtrage et de Routage») est

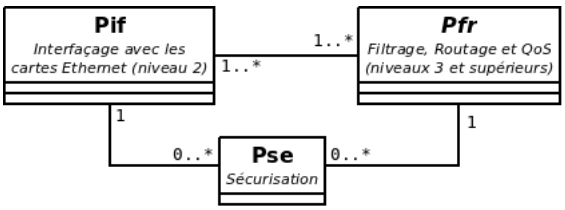



**Figure 3.8.:** Diagramme des classes de partition du routeur SNG avec dia

dédiée au routage et au filtrage des paquets, une classe «**Pse**» («Partition de sécurisation») gère les problématiques de sécurisation des flux (confidentialité, intégrité et authentification) et une classe «**Pif**» («Partition d'interfaçage») est dédiée à la gestion des interactions avec le matériel.

Les trois classes de partition présentées figure 3.8 permettent de réaliser un routeur pour des topologies simples, comme pour la topologie 1 de la figure 3.9 où l'architecture du routeur SNG interconnecte trois domaines.

Sans nécessiter de réitérer l'étape de modélisation des classes de partition, il est possible d'élaborer un second système en modifiant l'architecture précédente. La figure 3.10 montre une seconde topologie réseau à 5 domaines, connectés en deux groupes distincts et indépendants. Le logiciel du routeur SNG est donc architecturé à nouveau en ajoutant de nouvelles instances de partitions des classes de partition existantes (donc déjà certifiées et évaluées) et de nouvelles connections entre instances, la réutilisation réduisant ainsi le travail nécessaire de validation du nouveau système.

Un projet plus complexe pourrait bénéficier d'une solution logicielle plus adaptée à la gestion et au suivi d'exigences dans un environnement où de nombreux corps de métiers interviennent simultanément. Ainsi, Rational DOORS ([IBM 2013]) est une alternative à laquelle nous avions pensé initialement mais que nous n'avons pas sélectionné en raison de la « simplicité» voulue pour le développement de notre routeur.

### 3.3.2. Modélisation avec Simulink et Stateflow

Lorsque les classes de partition ont été définies, avec les fonctionnalités voulues et leurs interactions, la seconde étape de modélisation par les concepteurs peut commencer. Pour réaliser un modèle pour chaque classe de partition, nous avons choisi d'utiliser Simulink [MathWorks 2013b] et Stateflow [MathWorks 2013c], deux extensions à Matlab [MathWorks 2013a]. Ces logiciels permettent de modéliser respectivement des schémas blocs et des diagrammes de machines à états (voir figure 3.11). Ils permettent aussi de simuler le fonctionnement des programmes ainsi que d'effectuer des preuves formelles sur les modèles, pour garantir par exemple l'absence de blocage de l'exécution ou de boucles sans fin.

Le lecteur intéressé par les méthodes formelles peut trouver plus d'informations ici [Clarke *et al.* 1996, Jaeger 2010] et des exemples d'applications dans l'avionique

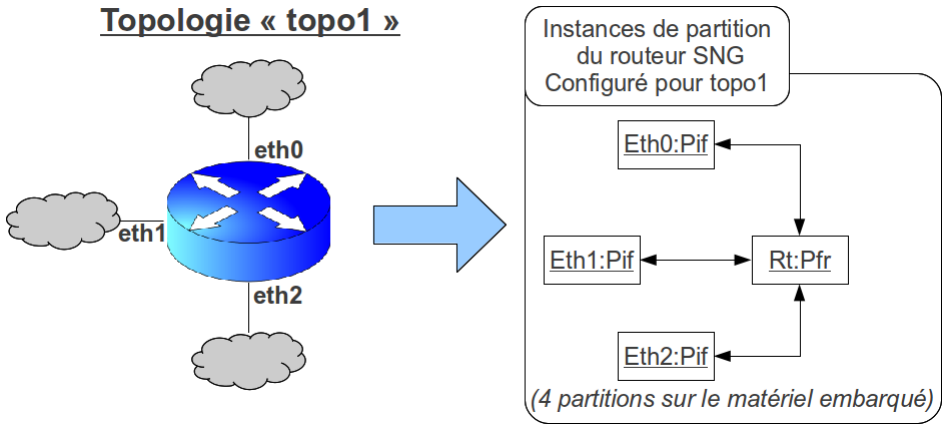



**Figure 3.9.:** Trois domaines interconnectés par le routeur SNG

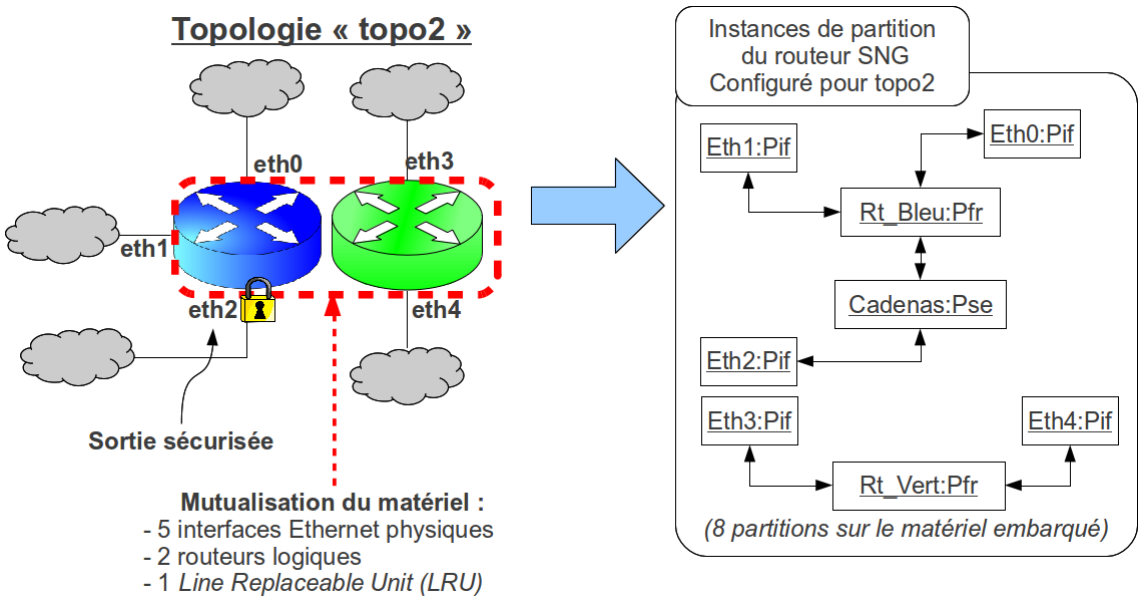



**Figure 3.10.:** Topologie réseau plus complexe du routeur SNG

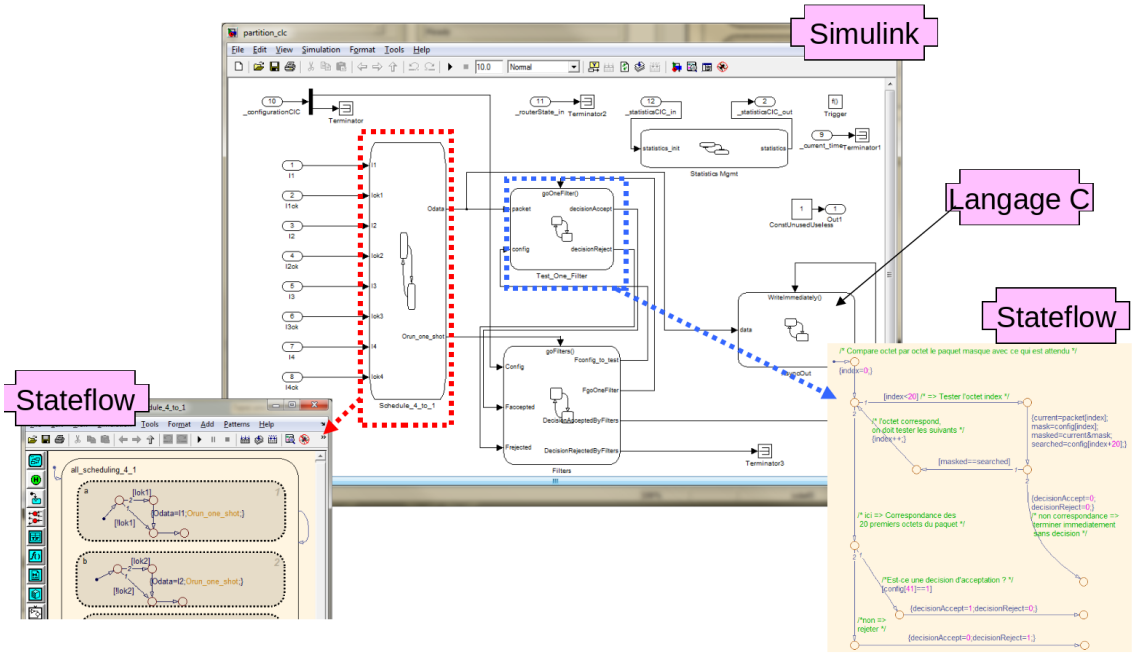



**Figure 3.11.:** Simulink et Stateflow pour modéliser la classe de partition Pfr du routeur SNG

là : [Ghafari 2010, Joyce 2010, Adams *et al.* 2005, Rayadurgam & Heimdahl 2001]. Concernant le développement basé sur les modèles (Model-Based Development, MBD) et ses applications dans l'aéronautique, nous lui conseillons la lecture de [Busser *et al.* 2004, Ákos Horváth 2010, Bhatt *et al.* 2005, Subbiah & Nagaraj 2003, Polgar 2011, da Silva Stanisce Correa *et al.* 2011].

Il est important de noter que les ensembles de fonctionnalités des classes Pfr et Pse sont disjoints, les classes peuvent donc être modélisées en parallèle par des concepteurs différents. Il en est de même pour la classe Pif, toutefois celle-ci a été écrite manuellement directement dans le langage intermédiaire d'entrée de l'étape 5, en raison de son besoin d'accès direct au matériel.

### 3.3.3. Transformation avec GeneAuto vers les langages C et Ada

La troisième étape consiste à transformer les modèles dans des langages de haut niveau en code source dans un langage plus proche du format binaire exécutable par la cible embarquée. Pour réaliser cette transformation, nous avons choisi d'utiliser le logiciel du projet GeneAuto, un projet libre Open Source géré par un consortium européen d'industriels utilisateurs et fournisseurs de services et de partenaires académiques (pour plus d'informations, consulter [Jobredeaux *et al.* 2011, Rugina & Dalbin 2010, Rugina *et al.* 2008, Toom *et al.* 2008, Izerrouken *et al.* 2008, Toom *et al.* 2010,

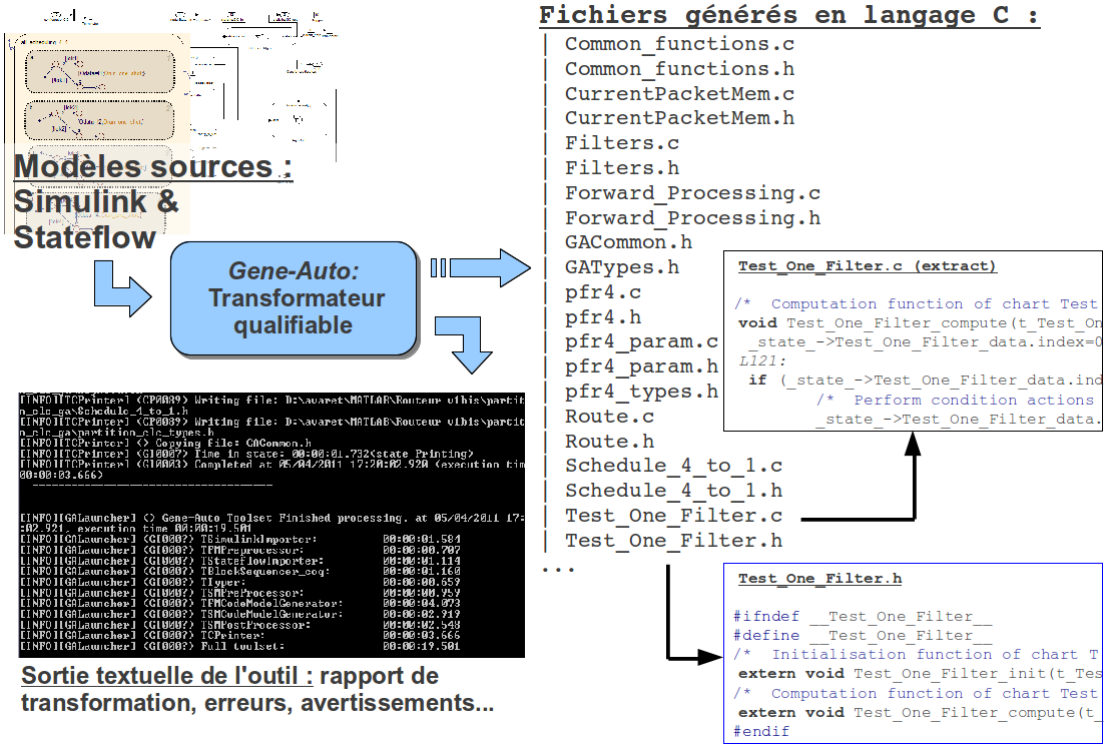



**Figure 3.12.:** GeneAuto pour transformer les modèles en code de cœur

Tonu Naks IB 2010]). Nous aurions pu aussi utiliser son concurrent SCADE [SCADE 2013] commercialisé par Esterel Technologie, mais nous avons préféré privilégier la solution libre et gratuite.

GeneAuto transforme des modèles Simulink, Stateflow et Scicos [INRIA 2013] en code en langage C [Ritchie 1972] ou Ada[Ichbiah 1983]. Il est disponible avec tous les documents exigibles pour la qualification de l'outil conformément à la norme DO-178B [DO-178B 1992].

Dans le cas du routeur SNG, les modèles Simulink (encapsulant des modèles Stateflow) des classes de partition Pfr et Pse ont été convertis en 56 fichiers pour un total de 10 500 lignes de code source en langage C. L'illustration 3.12 montre un extrait de la sortie textuelle de cet outil appliqué à la classe de partition Pfr.

Le code source généré à cette étape est un ensemble de fonctions en langage C destiné à être le cœur du code exécuté dans la partition. Il est indépendant de la plateforme matérielle embarquée ou même du système d'exploitation.

### 3.3.4. Collage manuel mais automatisable

Le code généré à l'étape précédente nécessite d'être lié aux entrées-sorties du système. Bien que certaines entrées-sorties soient très dépendantes du système matériel

**Algorithm 3.1** Code de collage pour les entrées/sorties (extrait de glue-code_for_pfr4.c)

```
1  void read_all_pfr4_partition_inputs(t_pfr6_io * io) {
2    P4_size_t read_size;
3    vm_e_t result;
4
5    /* Lecture de l'entrée 1 :
6       - données dans "I1",
7       - validité dans "I1ok" */
8    result = vm_qport_read(&In1, io->I1, 1516,
9       P4_TIMEOUT_NULL, &read_size);
10   io->I1ok = (result == VM_E_OK) && (read_size == 1516);
11   /* Idem pour l'entrée 2 */
12   result = vm_qport_read(&In1, io->I1, 1516,
13      P4_TIMEOUT_NULL, &read_size);
14   io->I1ok = (result == VM_E_OK) && (read_size == 1516);
15      ...
16 }
17
18 void write_all_pfr4_partition_outputs(t_pfr6_io * io) {
19   /* Si le modèle a indiqué que "O1" contient
20      des données à envoyer, les envoyer. */
21   if(io->O1ok)
22      vm_qport_write(&Out1, io->O1, 1516, P4_TIMEOUT_NULL);
23   /* Idem pour la sortie "O2" */
24   if(io->O2ok)
25      vm_qport_write(&Out2, io->O2, 1516, P4_TIMEOUT_NULL);
26      ...
27 }
```

embarqué, une grande partie d'entre elles sont des communications entre partitions. Les systèmes d'exploitation que nous utilisons sont compatibles avec l'interface de programmation ARINC 653 APEX introduite dans la section 3.1.4. Cette interface étant standard, elle permet de lier les entrées de la partition avec les entrées du modèle d'une manière répétitive, comme illustré par l'extrait de code 3.1, indépendamment de la marque et de la version du système d'exploitation. Il en est de même pour les sorties.

Bien que nous avons écrit manuellement le code C de collage pour les 4 entrées et les 4 sorties des classes de partition du routeur SNG, un petit script qualifié automatisant la génération du code est réalisable et pourrait permettre de gagner du temps pour les partitions avec plus d'entrées-sorties.

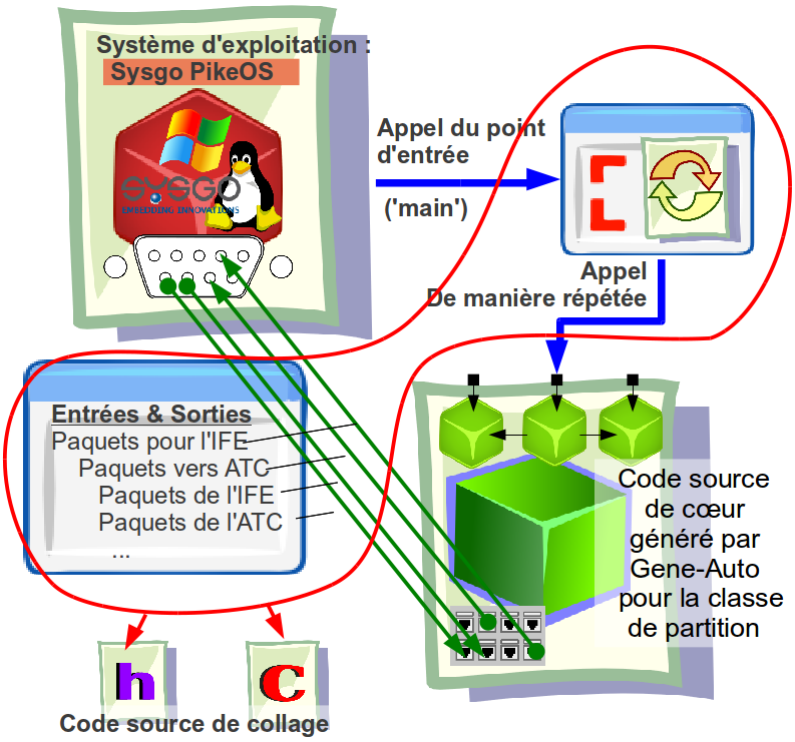



**Figure 3.13.:** Code de collage liant le système d'exploitation au modèle de la classe de partition

L'étape de collage a un deuxième objectif : fournir au système d'exploitation un point d'entrée unique pour la partition à exécuter. En effet, le générateur GeneAuto génère une fonction par bloc Simulink ; de plus, l'exécution de ces fonctions correspond à l'exécution d'un pas de simulation dans l'environnement de Simulink, elle se termine donc à la fin de la première itération. L'exécution désirée est au contraire de recommencer infiniment une nouvelle itération. C'est pourquoi l'algorithme implanté dans l'étape de collage est le suivant :

Cet algorithme 3.2 est indiqué au système d'exploitation comme étant le point d'entrée de l'exécution pour la partition. Ainsi, au démarrage de la partition, l'algorithme est appelé et s'exécute jusqu'à l'arrêt de la partition, consécutive soit à l'extinction du système, soit au déclenchement d'une panne. Les systèmes d'exploitation exigent souvent d'employer un nom spécifique pour identifier le point d'entrée ; cet identificateur dépend du modèle et de la version du système d'exploitation, ce qui en rend le code de collage dépendant.

À la fin de cette étape, le code source de collage est écrit et permet, comme l'illustre la figure 3.13, de lier le modèle de la classe de partition avec le système d'exploitation, la liaison étant pour les entrées et sorties des données et pour l'appel de la routine adéquate du modèle.

**Algorithm 3.2** Algorithme du code de collage (fichier pfr4.app.c intégral)

```
1  /***** Spécifique au système d'exploitation "PikeOS" *****/
2  #include <vm.h>
3  VM_DECLARE_STACK(0x4000)
4
5  /*****          Spécifique au modèle "Pfr4"          *****/
6  #include "pfr4.h"
7  t_pfr4_io io;       /* Entrées + sorties du modèle */
8  t_pfr4_state st;    /* Etat interne persistent du modèle */
9
10 /*****                 Pour le collage                *****/
11 #include "gluecode_for_pfr4.h"
12 #define FOREVER while(1)
13
14 /*** Fonction "main" appelée par le système d'exploitation ***/
15 extern void _p4_entry(void) {
16     vm_init();          /* Notifier le démarrage à PikeOS (OS) */
17     pfr4_init(&st);     /* Initialiser le modèle Pfr4 */
18     gluecode_init();    /* Ouvrir les ports (Message Queues...)*/
19
20     vm_part_set_mode(VM_RESPART_MYSELF, VM_PART_MODE_NORMAL,
21         0, 1, 0);       /* Etat de la partition :
22           'initialisation' --> 'exécution' (cf ARINC 653 APEX) */
23
24     FOREVER {
25         /* 1) lecture de toutes les entrées [OS->modèle] */
26         read_all_pfr4_partition_inputs(&io);
27         /* 2) invocation du modèle (= faire un pas) */
28         pfr4_compute(&io, &st);
29         /* 3) écriture de toutes les sorties [modèle->OS] */
30         write_all_pfr4_partition_outputs(&io);
31     }
32 }
```

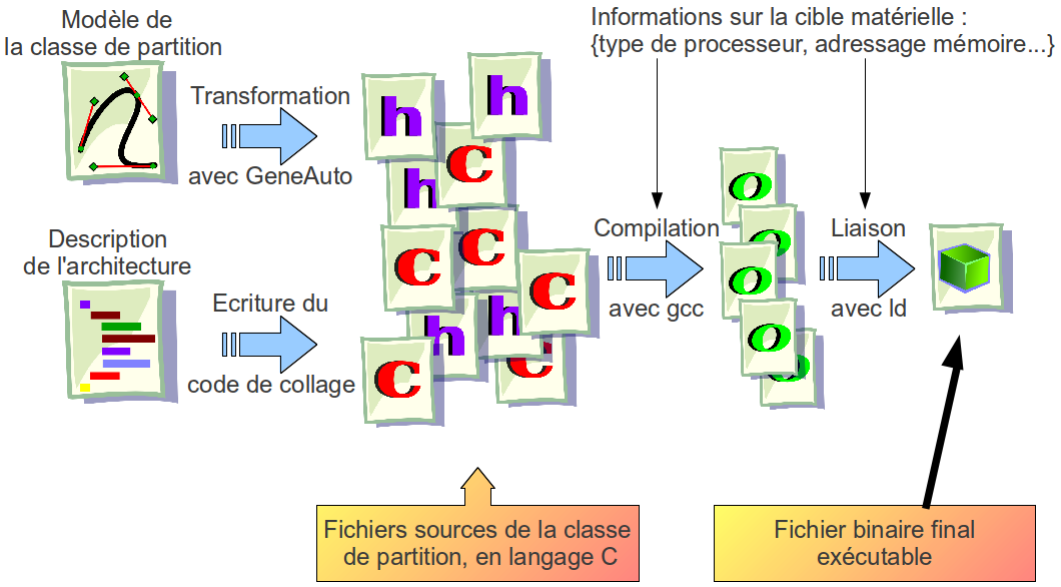



**Figure 3.14.:** Compilation et liaison des fichiers sources de la classe de partition Pfr

### 3.3.5. Compilation avec gcc du code en langage C

Lorsque les codes de cœur et de collage sont écrits, ils sont indépendants du modèle de processeur associé à la plateforme matérielle et de la répartition des données dans l'espace d'adressage. Le compilateur/lieur est le programme informatique chargé d'établir le code binaire adéquat, correspondant au code source, en utilisant les instructions disponibles dans le jeu d'instructions du processeur de la plateforme matérielle et en répartissant les données et le code binaire aux adresses mémoires adéquates.

Pour la cinquième étape de la méthodologie appliquée au routeur SNG, nous avons utilisé une version modifiée de gcc [GCC 2013], figée à une version ancienne éprouvée. Nous aurions pu utiliser une version plus récente et plus optimisée mais avec plus de difficultés pour une éventuelle qualification de l'outil. Nous avons aussi expérimenté le compilateur gnat [AdaCore 2013] de la société AdaCore avec du code en Ada généré par une version expérimentale de GeneAuto qui s'est avérée prometteuse mais incomplète : la transformation de certains types de blocs des modèles vers du code Ada n'étaient pas mis en œuvre dans cette version, cela nous a donc empêché de valider nos expérimentations avec le langage Ada.

À la fin de cette étape, nous disposons d'un fichier binaire par classe de partition, pouvant s'exécuter de manière autonome dans une machine virtuelle. Comme illustré par la figure 3.14, cette étape convertit les fichiers .c et .h des codes de cœur générés par GeneAuto et de collage écrits à la main en un fichier binaire unique exécutable, qui sera utilisé pour toutes les instances de la classe de partition Pfr.

### 3.3.6. Intégration avec Sysgo PikeOS

La sixième étape de la méthodologie consiste à intégrer ces binaires avec un système d'exploitation. Pour cette étape, le choix du système d'exploitation (SE) impacte les propriétés du logiciel final.

- Un SE assurant la virtualisation (voir section 3.1.2) permet d'assimiler chaque instance de partition à une machine virtuelle.
- Un SE compatible MILS (voir section 3.1.3) apporte un avantage pour la sûreté de l'exécution des partitions, mais aussi pour l'évaluation du système final.
- Un SE compatible IMA (voir section 3.1.4) permet de garantir la sécurité de l'exécution des partitions et simplifie la certification du système final.
- Les SE destinés à l'embarqué sont fournis sous forme de modules, pouvant ou non être conservés durant la phase d'élaboration du système, ce qui permet d'embarquer ce qui est nécessaire et que ce qui est nécessaire !
- Les SE commerciaux sont vendus avec des boîtes à outils prêtes à l'emploi (documentation, compilateurs, exemples et tutoriaux, support technique, outils d'aide à la configuration...), ce qui est une aide indéniable à la prise en main.
- Certains SE peuvent fournir des garanties temps réel, nécessaires pour certains systèmes embarqués, mais finalement non utiles dans le contexte d'utilisation de notre routeur SNG.

Après avoir étudié différentes offres, notre choix s'est porté sur le système PikeOS [Sysgo 2010] de la société Sysgo, car il fournit le service de virtualisation, est compatible MILS et IMA, est vendu pour être qualifié suivant le DO-178B (cf. section 2.4) et évalué au niveau EAL-6 (cf section 2.5), son micronoyau a une faible empreinte mémoire (150 ko de RAM + 150 ko de ROM) et est vendu prêt à l'emploi.

Le système VxWorks [WindRiver 2009], vendu par la société concurrente WindRiver, dispose des mêmes avantages et d'une réputation supérieure mais n'a pas été choisi pour des raisons de coûts supérieurs. Les systèmes libres sont discutés plus bas dans la section 3.3.8. L'étude [Leiner & Schlager 2007] compare ces SE candidats avec d'autres candidats potentiels.

Dans la boîte à outils fournie avec Sysgo PikeOS est intégré un environnement de développement rapide (RAD, « Rapid Application Development »), basé sur l'environnement Eclipse [eclipse 2013] étendu avec le *plug-in* Sysgo CODEO [Sysgo 2012]. Visible sur la figure 3.15, cet environnement étendu facilite le développement de systèmes embarqués en fournissant une aide à la conception, des assistants pour configurer l'architecture et des mécanismes de débogage et d'investigation.

Ainsi, après avoir renseigné pour le projet de développement les informations sur la plateforme embarquée (type de processeurs, drivers matériels...) comme illustré par la figure 3.16, nous avons indiqué l'ensemble des instances de partition à exécuter en parallèle, leur ordonnancement, leurs interactions (les communications entre partitions, voir la figure 3.17), les classes de partition associées (plus exactement les

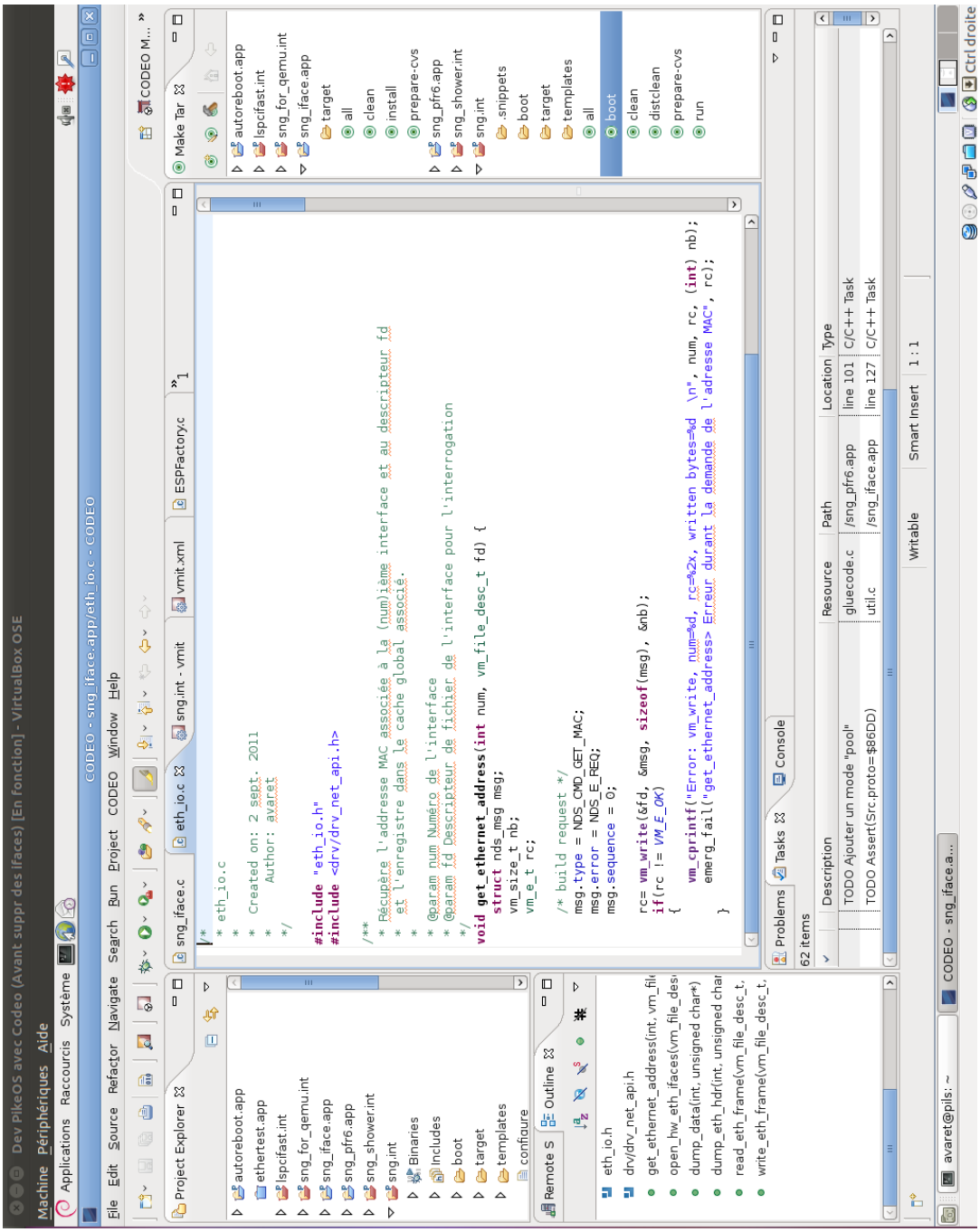


**Figure 3.15.:** L'environnement de développement rapide eclipse étendu avec CODEO

fichiers binaires générés à l'étape précédente), les accès autorisés au matériel, le comportement logiciel à adopter quand une faute est détectée, etc.

Enfin, le processus de génération « pikeos-mkromimage » de la boîte à outils fournie avec Sysgo prend l'ensemble de ces informations, les compile et y ajoute le noyau binaire du système PikeOS pour générer l'image binaire finale du logiciel du routeur SNG.

### 3.3.7. Exécution avec qemu-x86 puis sur système embarqué

La septième et dernière étape de la méthodologie consiste à tester et valider l'image générée à l'étape précédente.

Dans un premier temps, un émulateur de matériel tel que qemu [Bellard 2013] permet d'émuler une plateforme matérielle avec un processeur, de la mémoire vive et/ou de masse et des périphériques PCI. Comme illustré figure 3.18, cela nous a permis de rapidement nous assurer du bon démarrage de l'image, puis de mettre en place un système de démarrage par téléchargement du fichier image via un réseau local.

Dans un deuxième temps, nous avons mis en place une plateforme matérielle de tests, constituée d'un processeur disposant de quatre cœurs Intel® Pentium© 64 bit, dotée de 4 Go de mémoire vive et d'une carte réseau avec 4 ports Ethernet Gigabit. La photographie de cette plateforme est visible dans la figure 3.19.

En fait, à l'exécution, l'image est configurée pour n'utiliser que 1468 ko de mémoire et un seul cœur du processeur en 32 bits, mais le surdimensionnement de la cible matérielle n'est pas ici un problème. Le chargement de l'image s'effectue par le protocole PXE ([Intel & Systemsoft 1999], basé sur les protocoles DHCP [Droms 1997] et TFTP [Sollins 1992]) pour des raisons de simplicité, un système embarqué réel disposerait plus probablement d'une ROM contenant l'image binaire pour assurer le chargement et le démarrage du logiciel du routeur SNG.

Nous avons utilisé cette dernière plateforme pour effectuer l'évaluation des performances du routeur SNG présentée au chapitre 6.

La chaîne d'outils que nous avons utilisé est résumé par la figure 3.20. Elle est à mettre en perspective avec la figure 3.5 au début de cette section présentant les différentes étapes de notre méthodologie.

### 3.3.8. Une autre chaîne d'outils, presque tous libres

Comme nous l'avons vu, la méthodologie est indépendante des outils employés : à chacune des étapes, différents outils sont disponibles pour aider à sa réalisation. L'équipe de développement a donc une grande liberté de choix pour s'adapter au projet de développement et à son contexte.

Il est ainsi possible de réaliser un développement logiciel en utilisant :

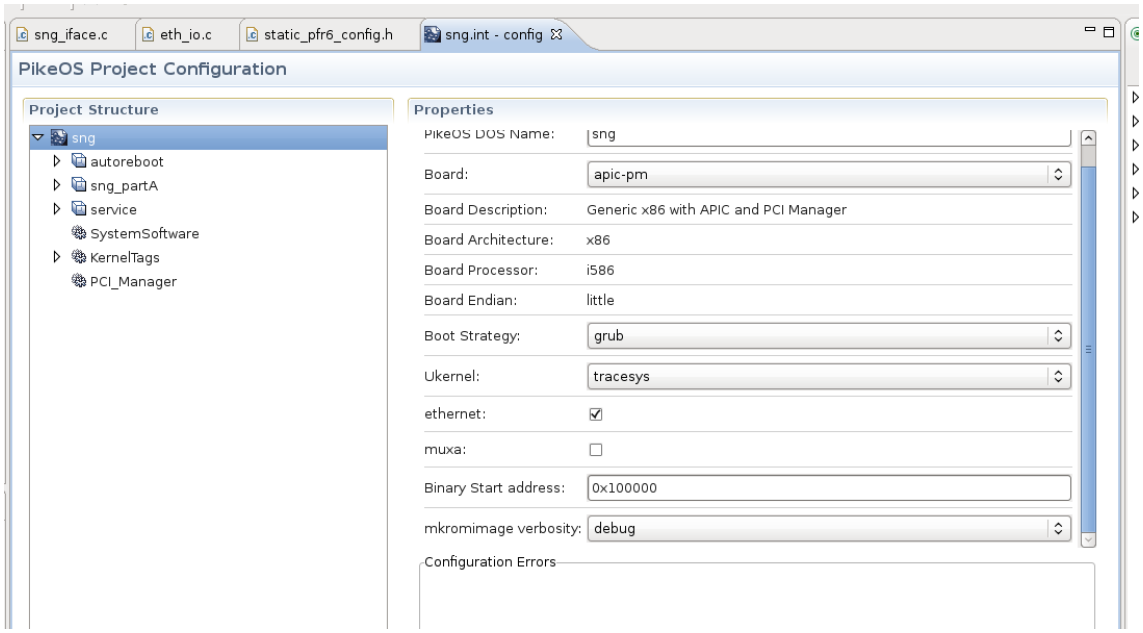



**Figure 3.16.:** Configuration de la plateforme avec CODEO

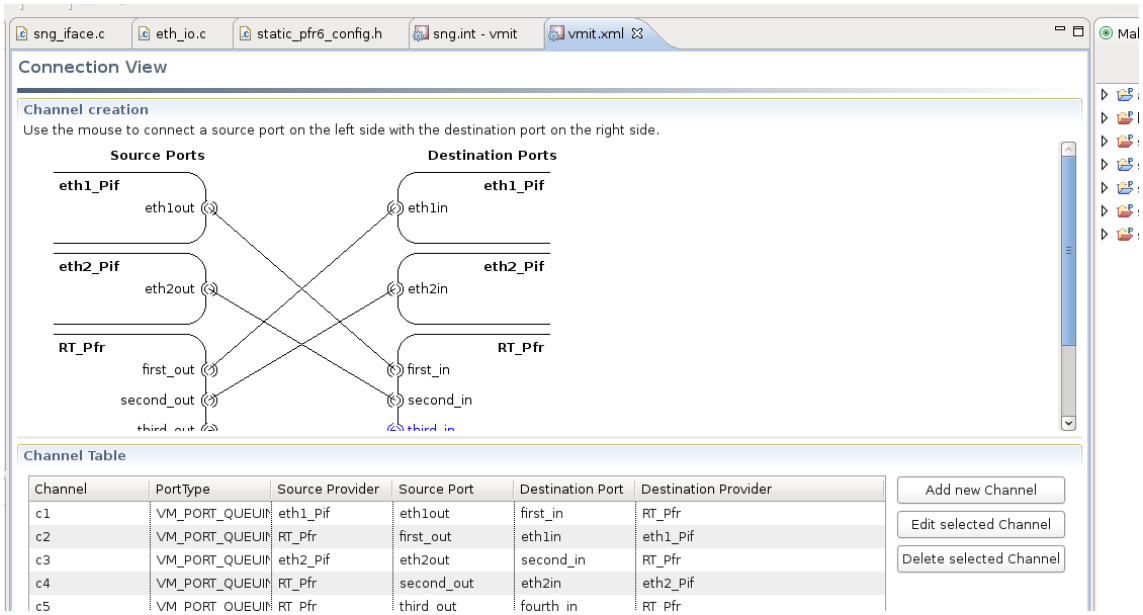



**Figure 3.17.:** Configuration des interactions entre instances de partitions avec CODEO

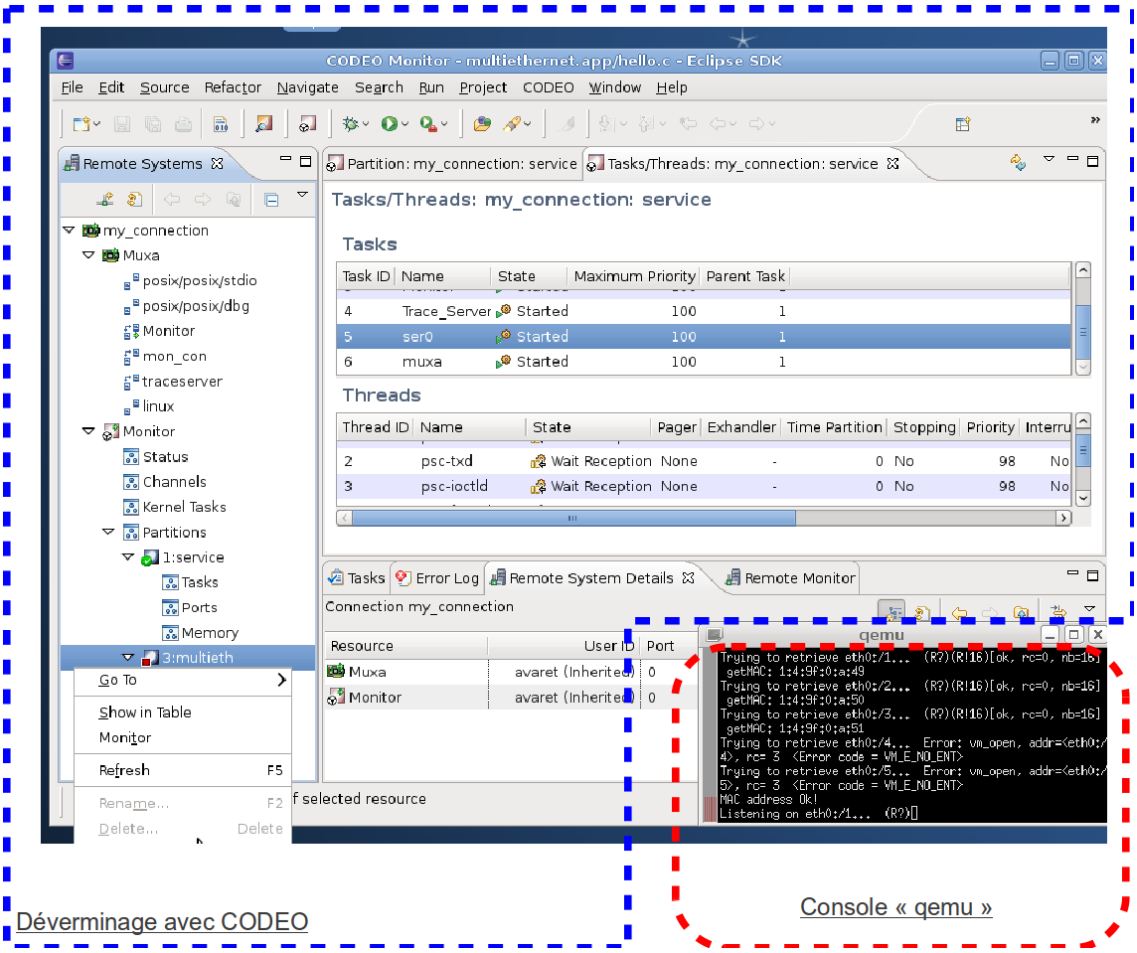



**Figure 3.18.:** L'émulateur de plateforme matérielle, qemu

– Le logiciel libre Scicos pour la modélisation,
– Le logiciel libre et qualifiable GeneAuto pour la transformation en langage Ada,
– Le compilateur libre et qualifiable gnat pour générer le binaire,
– Le système d'exploitation commercial Sysgo PikeOS,
– L'IDE libre eclipse,
– L'émulateur libre qemu pour la plateforme matérielle émulée logiciellement.

Cette chaîne est constituée uniquement d'outils sur étagère (COTS), permettant ainsi de rapidement les déployer.

Tous ces outils sont libres, à l'exception du système d'exploitation (abrégé SE dans la suite de ce mémoire) que nous avons choisi, à défaut de trouver un SE libre compatible MILS et IMA. Nous aurions pu choisir d'utiliser un noyau linux personnalisé pour en faire un hyperviseur, comme Xen ([Keir Fraser 2003]), mais l'effort de certification et d'évaluation nous est dans ce cas inconnu et potentiellement élevé en raison du grand nombre de lignes de code source de ce système [Marcil 2011].

Se limiter à une chaîne intégralement Open Source pour l'instanciation de la

**Figure 3.19.:** Photographie du PC utilisé comme cible matérielle embarquée

méthodologie apporte un certain nombre d'avantages bénéfiques pour le processus de développement.

Un code source ouvert peut être lu par les utilisateurs. Dans notre cas des codes source des outils de développement, les utilisateurs sont des programmeurs, donc capables de lire le code des outils (et aussi souvent curieux). Cela contribue à faciliter et accélérer l'intégration de nouveaux programmeurs aux projets : la lecture du code source peut éviter des incompréhensions, pallier des insuffisances éventuelles de la documentation officielle et aider au déverminage des codes programmés par l'utilisateur des outils.

De la même manière, lors du déploiement du produit chez le client, la lecture du code des logiciels et des scripts peut éviter le recours systématique au support technique, procédure parfois longue, fastidieuse et coûteuse en temps et en argent (surtout lorsque le support technique et le client peinent à se comprendre).

La disponibilité du code source rend possible son adaptation aux cas non gérés par la version officielle, pour par exemple ajouter des fonctionnalités ou des optimisations.

Généralement, l'implication dans le développement des outils est une motivation, pour l'utilisateur qui contribue à un travail collectif visible publiquement mais aussi

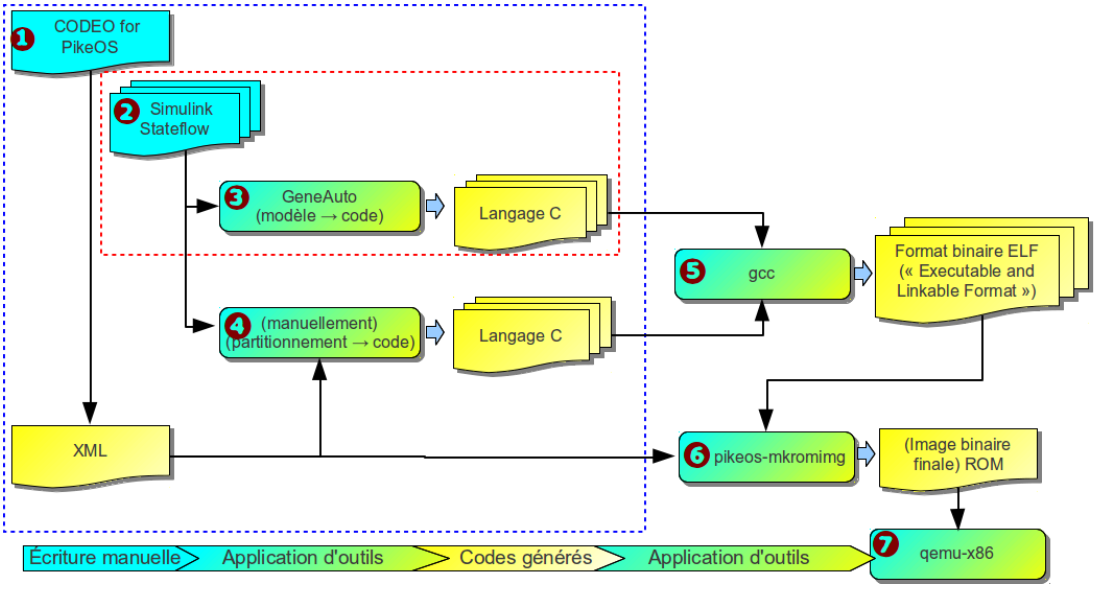



**Figure 3.20.:** Chaîne d'outils que nous avons utilisé pour le routeur SNG

pour l'équipe de développement qui affine sa vision de l'utilisation et des utilisateurs du logiciel.

Le code étant disponible, il est possible de réutiliser certains de ses modules dans d'autres projets.

La disponibilité du code est aussi une garantie de maintenabilité à long terme. Ainsi, si l'entreprise ayant créée les outils disparaît, les utilisateurs ont pu archiver les codes sources et peuvent donc continuer à maintenir les programmes. Cette contrainte est rarement mis en avant par les usagers grand public mais elle est critique dans des domaines spécialisés tels que l'aéronautique où les organismes de certification exigent que le logiciel puisse être reconstruit durant l'intégralité du cycle de vie de l'avion associé (soit actuellement une quarantaine d'années).

La notion de code ouvert (« Open Source ») est souvent (mais non nécessairement) liée à la notion de gratuité (« free software »).

Les logiciels gratuits évitent d'avoir à déployer un serveur de gestion de licence, accélérant le déploiement et évitant les problèmes d'«aucune licence disponible sur le serveur» à déverminer.

De plus, l'absence de gestion de licence évite les limites de nombre maximal d'utilisateurs et/ou de postes, facilitant la ventilation des programmeurs entre les différents projets.

Pour terminer, le code source étant accessible, il est potentiellement lu par plus de lecteurs, entraînant la découverte et la correction de plus d'erreurs et cela plus rapidement, devenant ainsi un code plus sûr et de meilleure qualité que leurs équivalents fermés (voir à ce sujet l'article «Linux: fewer bugs than rivals» sur [Delio 2004]).

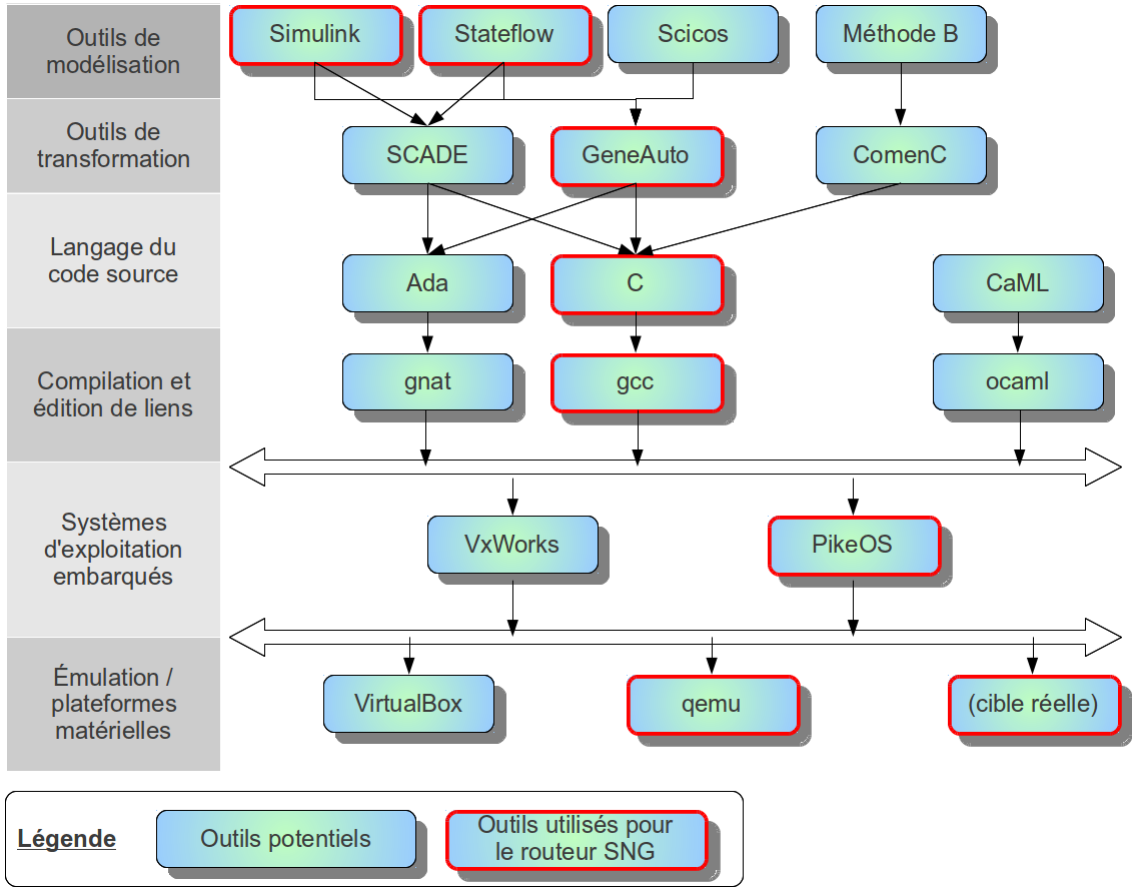



**Figure 3.21.:** Outils applicables dans le cadre de notre méthodologie

J'ai personnellement été confronté à chacun de ces dix points dans le cadre de ma thèse.

### 3.3.9. Synthèse sur les chaînes d'outils

Pour conclure sur les chaînes d'outils, la méthodologie n'est pas limitée aux paradigmes de la programmation impérative. Nous avions envisagé initialement d'utiliser une chaîne d'outils fonctionnels avec du code source écrit en langage CaML [Gérard Huet 1985] et pour le compiler un compilateur ocaml ([INRIA 1996]) prouvé correct à l'aide de méthodes formelles [Leroy 2006]. Nous avions aussi planifié d'utiliser la méthode B ([Abrial 1996]) pour modéliser les classes de partition avec l'outil ComenC ([ClearSy 2013]) pour générer du C. Mais nous n'avons finalement pas exploité ces possibilités, la chaîne d'outils impératifs présentée figure 3.20 étant amplement suffisante et adaptée à notre cas du routeur SNG.

La figure 3.21 résume les différents outils énoncés dans ce chapitre. Cette liste n'est pas limitative, la méthodologie ayant été conçue pour être la plus générique possible.

| Task | Description |
|---|---|
| A.2.6 | The source code is developed |
| A.2.7 | The Executable Object Code is produced |
| A.5.1 | The source code complies with low-level requirements |
| A.5.2 | The source code complies with software architecture |
| A.5.3 | The source code is verifiable |
| A.5.4 | The source code conforms to standards |
| A.5.5 | The source code is traceable to low-level requirements |
| A.5.6 | The source code is accurate and consistent |
| A.6.3 | Executable Object Code complies with low-level requirements |
| A.6.4 | Executable Object Code is robust with low-level req requirements |
| A.7.1 | Test procedures are correct |
| A.7.2 | Test results are correct, and discrepancies are explained |
| A.7.4 | Test coverage of low-level requirements is achieved |
| A.7.5-7 | Test coverage of software structure is achieved |

**Table 3.1.:** Gains de la qualification sur la certification DO-178B

Il est important de remarquer qu'aucune colonne ne contient qu'une seule possibilité, il y a toujours un choix possible. En effet, les autorités de certification exigent des systèmes les plus sensibles qu'ils soient redondés et que les versions redondées soient dissimilaires, c'est-à-dire développées par des équipes différentes utilisant des langages de programmation différents et des outils différents, on parle alors de diversification.

La chaîne d'outils que nous avons appliqués au cas du routeur SNG, visible sur l'illustration 3.20, permet des gains en terme d'évaluation et de certification. En résumé, l'évaluation de la sûreté est simplifiée principalement par l'approche modulaire du partitionnement à l'étape 1 de la méthodologie et par l'emploi d'un système d'exploitation compatible MILS et lui-même déjà évalué conformément aux critères communs. Quant à la certification, la qualification des outils permet de réduire voir d'éliminer 16 des 66 tâches du processus de certification du DO-178B, énumérées dans la table 3.1.

Un autre point exigé par la norme DO-178B est relative à la qualité du code source, la norme exigeant notamment d'établir un recueil justifié de règles de codage et de vérifier le respect de cette exigence qualité (cf. DO-178B [DO-178B 1992] annexe A table 5 entry 4). Un exemple de règle courante est « le code ne doit pas contenir l'instruction **goto** du langage C » (règle issue de la norme automobile MISRA-C 1998 [MISRA-C 1998] relative à la qualité du code source embarqué). Dans ce qui suit, nous avons choisi d'étudier une piste complémentaire et d'évaluer quantitativement non seulement la qualité de la syntaxe mais aussi de la sémantique du code source ; cela nous a conduit à développer l'outil Métrix, présenté ci-dessous.

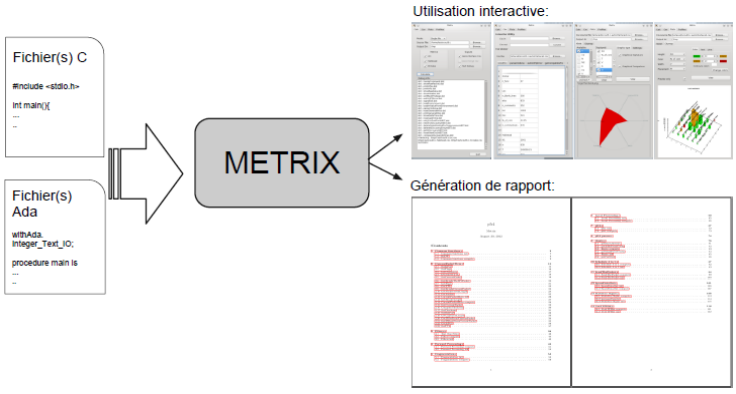



**Figure 3.22.:** Application Métrix

## 3.4. Métrix, logiciel d'évaluation de la qualité d'un code source

La méthodologie développée durant la première année de thèse a soulevé une problématique majeure : comment évaluer, qualitativement et quantitativement, la qualité du code source généré ?

### 3.4.1. Objectifs de Métrix

Afin de répondre à cette question, nous avons développé un logiciel libre appelé « Métrix », capable de mesurer différentes métriques sur un code source. Chacune contribue à apporter des informations sur la qualité du code. Actuellement, seuls les langages Ada et C (qui sont les langages privilégiés pour le routeur SNG) sont évaluables par Métrix. Le logiciel produit dispose d'une interface graphique (présentée en annexe) et génère à partir de code source C ou Ada un rapport sur la qualité du code source analysé. La figure 3.22 résume les entrées/sorties de cette application.

Ce logiciel s'inscrit dans un processus itératif d'amélioration du code source. Ainsi, à partir des codes sources écrits manuellement et générés par les outils de transformation des modèles, utilisés aux étapes de transformation et de collage de notre méthodologie, l'outil Métrix analyse le code et en extrait un rapport synthétisant l'évaluation de différentes métriques mesurées sur le code source et présentées dans la section suivante.

Ce rapport fournit aux développeurs les informations nécessaire pour identifier les points durs du code et des modèles générateurs à retravailler et à améliorer. Il synthétise une analyse automatisée de code source, permettant ainsi à la tâche de revue de code d'être plus productive. La figure 3.23 montre l'intégration de Métrix au processus de développement.

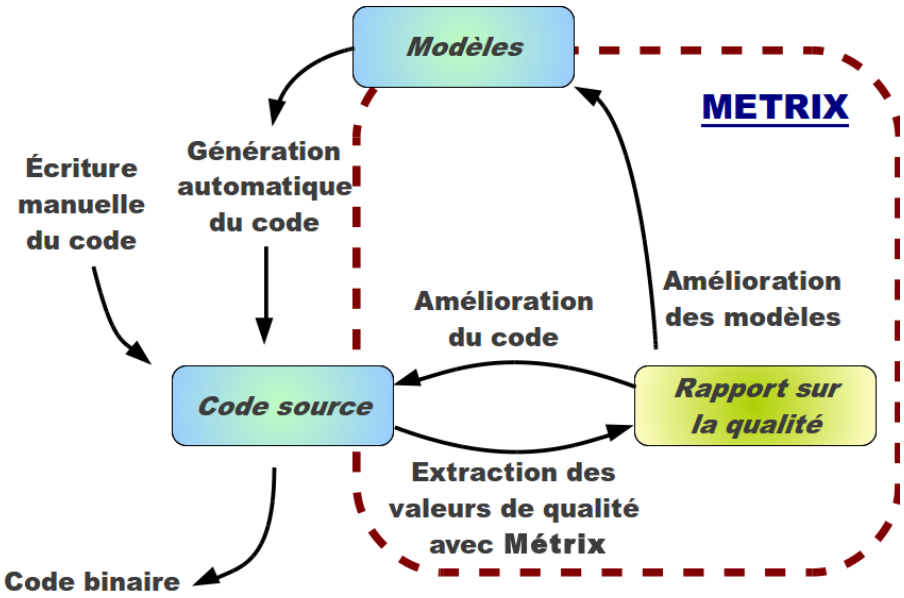



**Figure 3.23.:** Métrix dans le processus de développement

### 3.4.2. Les métriques mesurées sur le code source

Un grand nombre de métriques sont déjà définies actuellement dans la littérature [Zuse 1997, Marcil 2011]. Toutes ne sont pas pertinentes à mettre en place dans le cadre de cette application. Les métriques sélectionnées peuvent être classées dans 3 catégories :

– Les métriques « classiques » telles que le nombre de lignes de codes, le taux de commentaires, etc. Ces métriques, bien que couramment utilisées, sont très dépendantes des langages utilisés par les codes sources. On y trouve :

**Lines Of Codes (LoC)** compte le nombre de lignes de l'ensemble des fichiers composant le code source d'un projet. Elle est appréciée dans la gestion de projet car elle permet d'accéder à une estimation du coût et du temps de développement d'un logiciel. Il est cependant nécessaire de se poser certaines règles quand au mode de comptage des lignes ; en effet, comment doit-on comptabiliser un #include en C ou bien une importation de package en Ada ? Cette métrique se décline généralement en plusieurs variantes, énoncé ci-dessous.

**Effective Lines of Code (eLOC)** est un raffinement de la métrique LoC. Dans ce calcul, les lignes qui ne sont pas compilées en code binaire ne sont pas comptabilisées. Typiquement, les lignes en commentaire ne sont pas comptées, ainsi que les lignes vides, les accolades isolées sur une ligne.

**Logical Lines of Code (lLOC)** ne considère que les lignes constituants des lignes d'instructions. En C et en Ada, cette donnée est obtenue en comptant le nombre de séparateurs d'instructions «;», avec des exceptions pour les mots-

clefs qui utilisent ce caractère comme séparateur (par exemple, le «for (i;c;p) ...» en C).

**Lisibilité du code** dans ce genre de métriques, les lignes du code source qui sont considérées sont principalement le nombre de lignes de commentaires et le nombre d'espaces. Il est communément admis qu'un code aéré et judicieusement commenté est plus lisible qu'un code comportant peu d'espaces et de commentaires. Voir [LLC 2012] pour plus d'informations sur les trois métriques précédentes. Un code plus lisible est plus facile à maintenir et à déverminer. Bien que dans le cas de codes générés par des outils automatiques, le modèle passé au transformateur est probablement plus pertinent que le code généré.

**Taille des fonctions** pour faciliter la lecture, en terme de nombre de lignes, le Linux Kernel Coding Style [Linux 2000] suggère de faire autant de fonctions que possible afin de traiter une seule chose dans chaque fonction. Concernant la taille des fonctions, cela dépend de la complexité de la fonction (mesurable par la complexité cyclomatique, comme développé plus bas dans cette section). Les fonctions peu complexes peuvent être plus longues que celles plus complexes, mais la règle générale de [Linux 2000] est de toujours tenter de ne pas excéder 24 lignes de 80 caractères par fonction.

**Nombre d'appels de fonctions** c'est un point intéressant qui peut permettre notamment de donner des pistes permettant d'optimiser les performances du code. Naïvement, plus une fonction est appelée, plus il faudra porter d'attention à l'optimalité de son code source. En fait, seule l'interprétation ou l'exécution du code permet de savoir concrètement quelles fonctions sont appelées les plus souvent et combien de temps elles prennent, c'est d'ailleurs le rôle d'outils spécialisés tels que valgrind[Seward 2002] d'effectuer ces mesures. C'est pourquoi nous n'avons finalement pas implanté cette métrique.

– Les métriques de Halstead [Verifysoft 2012] (en raison de son auteur, le professeur Maurice Halstead) telles que le volume de code, sa difficulté algorithmique... Ces métriques se basent sur des éléments communs à tout langage impératif (n1, n2, N1, N2) et sont donc peu dépendants du langage utilisé.

**n1** nombre d'opérateurs différents

**n2** nombre d'opérandes différents

**N1** nombre total d'opérateurs

**N2** nombre total d'opérandes

**N** = N1 + N2, la taille du programme

**n** = n1 + n2, la taille du vocabulaire

**V** = N * log_2 (n), le Volume de Halstead permet d'accéder à la taille de l'implémentation de l'algorithme. D'après son auteur, V doit être compris entre 20 (volume d'une fonction d'une ligne sans paramètre) et 1000.

Un volume dépassant 1000 pour une fonction est souvent le signe que cette fonctions traite trop de choses et devrait être divisée en sous-fonctions plus spécifiques.

**D** = (n1 / 2) * (N2 / n2), la Difficulté ou propension d'erreurs

**E** = D * V, l'Effort d'implémentation

**T** = E / 18, le Temps d'implémentation (en seconde). La constante 18 a été déterminée par Maurice Halstead en fonction de la quantité d'opérations élémentaires par seconde que le cerveau humain est capable d'effectuer.

– Les métriques de McCabe, notamment la complexité cyclomatique des algorithmes, ces métriques étant indépendantes du langage source. Thomas J. McCabe a introduit dans [Arthur H. Watson 1996] la « cyclomatic complexity », complexité cyclomatique ou aussi mesure de McCabe en français. Cette complexité est établie en construisant le graphe du programme ou de la fonction puis en comptant le nombre de chemins différents que l'on peut suivre dans ce graphe [Nilsson 2005]. La complexité cyclomatique peut être calculée sur chaque module du programme (iv(G)), ce qui permet de mesurer la complexité de chaque module et ainsi de déterminer si ce module doit ou non être retravaillé afin de le rendre moins complexe. Notons que les métriques de McCabe peuvent s'appliquer aussi aux variables, ce qui permet de calculer la complexité d'un module en fonction de ses variables et peut servir à estimer l'effort à développer lors de la phase de tests du logiciel.

D'autres métriques n'ont pas été sélectionnées car le ratio pertinence/difficulté de mesure nous a semblé inadéquat. C'est notamment le cas pour toutes les métriques exigeant une exécution ou une interprétation de code ou encore celles relatives aux choix de nommage des identificateurs.

Les algorithmes de mesures sur des codes Ada et C des métriques sélectionnées ont été mis en œuvre en langage Perl[Wall 1987], aidé par quelques outils libres (cfg2dot pour le calcul de métriques de McCabe avec l'Ada et graphviz[Graphviz 2000] pour analyser les graphes générés).

### 3.4.3. Représentation des résultats mesurés par Métrix

Les métriques mesurées par Métrix et présentées dans la sous-section précédente fournissent des données numériques. Bien qu'une représentation textuelle de ces données apporte à l'utilisateur les informations, le grand nombre de données mesurées rend la tâche difficile, même pour un petit projet de développement. C'est pourquoi nous avons travaillé sur la représentation graphique des résultats numériques. Cette représentation est plus synthétique et donc plus « lisible » pour l'utilisateur. Nous avons sélectionné différents modes de représentation : les diagrammes de Kiviat et les citymaps, deux diagrammes que nous présentons dans cette section. La navigation entre les différents diagrammes générés et leur paramétrage sont facilités

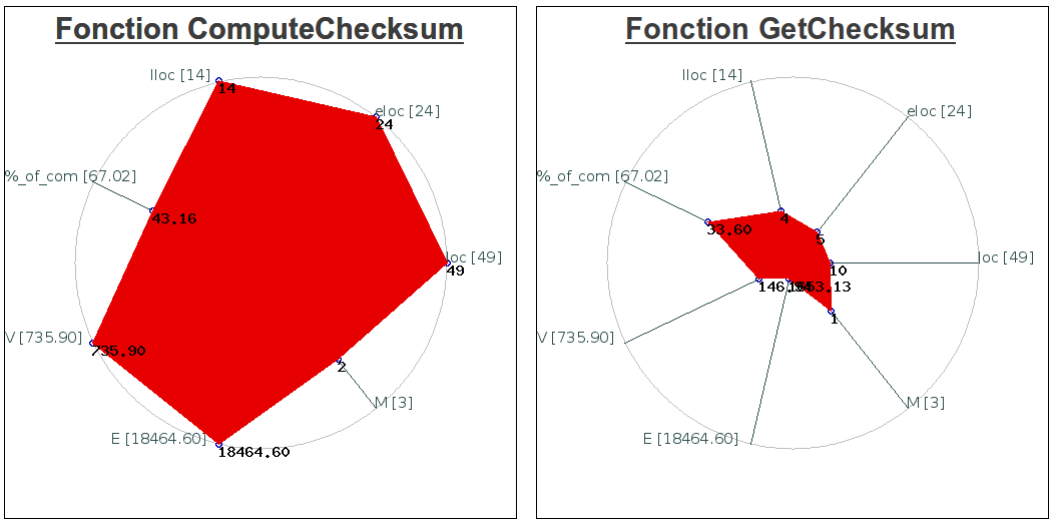



**Figure 3.24.:** Signatures des fonctions ComputeChecksum et GetChecksum

par l'utilisation de l'interface homme-machine que nous avons développée et qui est présentée dans l'annexe C.

Le diagramme de Kiviat est un mode de représentation en coordonnées polaires : la distance au centre est associée à la valeur à représenter tandis que l'angle entre deux données est constant et calculé pour répartir uniformément les données. Les points sont reliés par des segments, l'ensemble formant alors un polygone irrégulier plein. Nous l'avons utilisé de deux manières.

Dans la figure 3.24 nous avons placé les valeurs des différentes métriques relatives à une même fonction. Le polygone en résultant est donc spécifique à la fonction et diffère des polygones associés aux autres fonctions, nous avons donc choisi de parler de signatures des fonctions. Visuellement, la figure 3.24 montre que la fonction ComputeChecksum contient plus de lignes de code et est plus complexe d'un point de vue logique que la fonction GetChecksum.

Il est aussi possible de représenter, sur un diagramme de Kiviat, une métrique calculée pour toutes les fonctions d'un fichier ou d'un projet, comme illustré sur la figure 3.25. Dans ce cas, la métrique permet de comparer les fonctions et de les homogénéiser. Par exemple, sur cette figure 3.25, nous avons représenté la complexité cyclomatique du fichier currentpacket.c du code source du routeur SNG généré par GeneAuto. Il ressort visuellement du diagramme que les fonctions les plus complexes sont SetCurrentPacket puis GetCurrentPacket et ComputeChecksum.

Des variantes de ce type de diagrammes existent mais nous ne les avons pas retenues pour notre logiciel : diagrammes de Kiviat 3D [Kerren & Jusufi 2009, Yuhua 2008] et graphes de Kiviat [Martin Pinzger 2005].

Le deuxième type de représentation introduit par Métrix est le diagramme CityMap. L'idée vient des diagrammes « treemaps », des diagrammes plans cons-

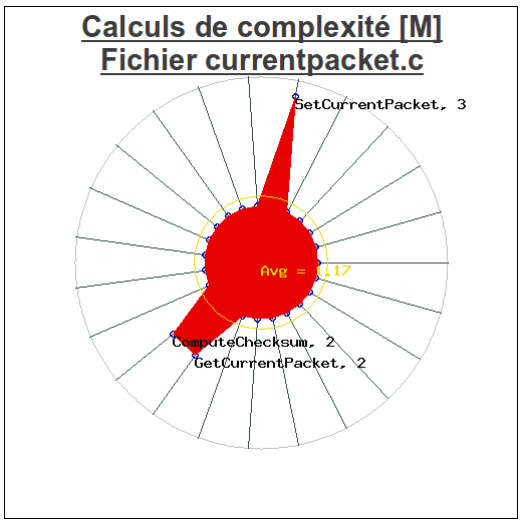



**Figure 3.25.:** Complexités cyclomatiques des fonctions de currentpacket.c

| | | |
|---|---|---|
| la hauteur des immeubles | corresponds | au nombre de lignes de code (LOC) |
| la largeur | “ | au volume du code (V de Halstead) |
| le placement | “ | à la complexité cyclomatique |
| la couleur | “ | au taux de commentaires |

**Table 3.2.:** Apairage (métrique de code source; métrique de représentation) pour la figure 3.26

titués de rectangles dont l'aire est proportionnelle à la valeur à représenter (cf [Toshiyuki Asahi & Schneiderman 2005]).

Dans l'article [Wettel & Lanza 2007], Richard Wettel et Michele Lanza proposent une mise en 3D des diagrammes treemaps. Cela forme donc des colonnes de hauteurs et de largeurs variables, semblables à des immeubles. Selon les auteurs, cela permet une meilleure estimation de la complexité d'un projet. La représentation sous forme de ville permet de mettre en avant des détails qui ne transparaissaient pas avec une simple treemap. Les immeubles sont rassemblés en fonction des fichiers auxquels les portions de code qu'ils représentent appartiennent.

La figure 3.26 illustre un exemple de ce type de diagramme. Celle-ci représente quatre métriques issues des fonctions d'un fichier source du routeur SNG, associées aux variables de représentation comme indiqué dans la table 3.2. Ce diagramme permet donc une comparaison multicritère des fonctions issues d'un même fichier de code source.

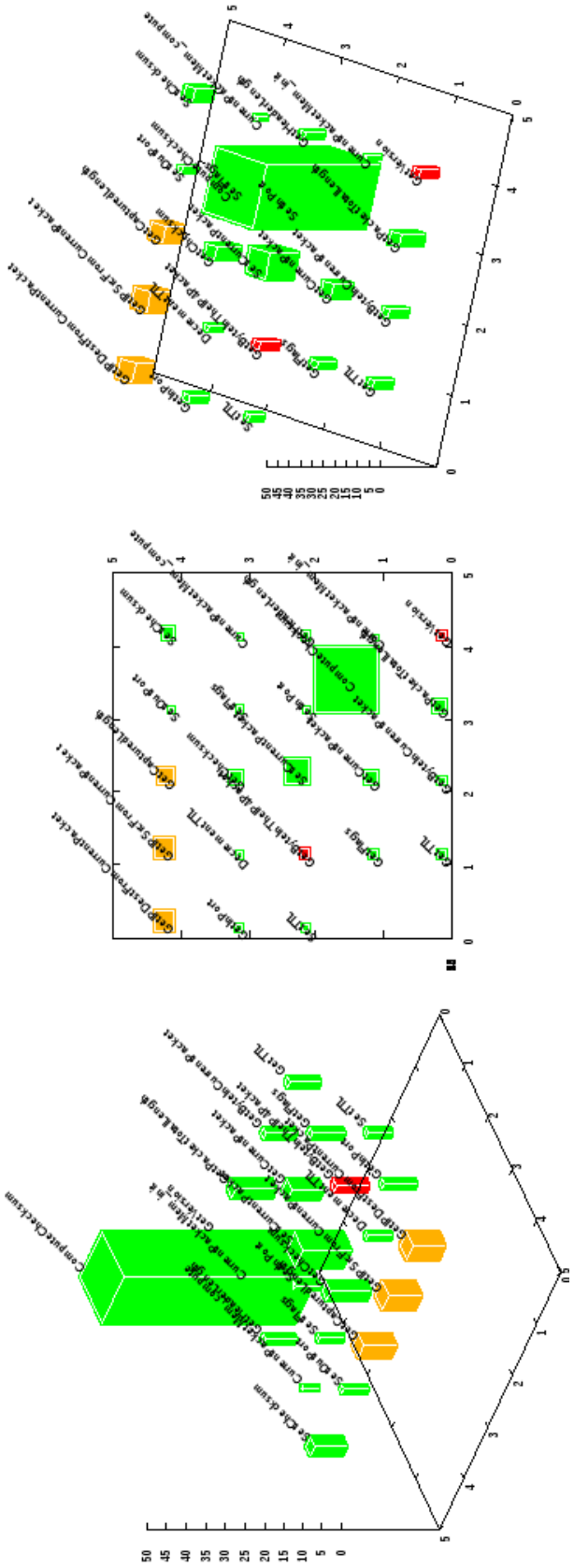


**Figure 3.26.:** Exemple de citymap pour représenter le fichier CurrentPacketMem.c

### 3.4.4. Conclusion sur notre logiciel Métrix

L'application de Métrix au code généré à partir des modèles du routeur SNG a permis d'identifier certaines parties du modèle générateur comme étant trop complexes (la complexité cyclomatique était supérieure à 30) et ainsi d'effectuer à nouveau mais différemment la phase de modélisation pour simplifier le code.

Métrix a été utilisé avec succès pour le code source de la mise en œuvre du protocole SCOUT, présenté au chapitre 5. L'utilisation de Métrix nous a amené à réécrire différemment la majorité des fonctions en C, pour réduire la complexité et la longueur de certaines fonctions critiques et homogénéiser le code source.

Cependant, Métrix ne traite actuellement pas directement les modèles mais le code généré, ainsi le transformateur oriente de manière non négligeable l'analyse du rapport produit par Métrix. Manipuler directement les modèles de haut niveau et non plus le code source intermédiaire est une évolution de Métrix sur laquelle nous travaillons actuellement.

Métrix est publié en Open Source sous la licence libre GPLv3 [(FSF) 2007].

## 3.5. Résumé du chapitre

Dans ce chapitre, nous nous sommes intéressés au processus de développement à appliquer au routeur SNG. Les contraintes de sûreté et de sécurité y sont critiques, d'où le choix d'un système d'exploitation compatible MILS et IMA, deux notions que nous avons présentées conjointement à celle de la virtualisation. Ces contraintes nous ont aussi poussé vers l'utilisation d'outils de modélisation pour pouvoir vérifier le système au plus tôt et vers des générateurs de codes automatisés pour éviter les fautes de traduction d'un langage de haut niveau vers des langages plus proches de notre cible matérielle embarquée. Ces outils nous ont aussi permis d'accélérer la phase de développement et de fabriquer rapidement un prototype fonctionnel pour les tests et la validation du routeur SNG.

Nous avons choisi de présenter séparément la méthodologie, abstraite et générique par nature, de son instanciation avec les outils. En effet, les principes généraux de la méthodologie sont indépendants des outils utilisés, et il existe plusieurs outils candidats pour chaque étape à réaliser. Nous avons présenté ceux utilisés pour le routeur SNG : Simulink et Stateflow pour la modélisation, GeneAuto pour la transformation, gcc pour la compilation, Sysgo PikeOS comme système d'exploitation virtualisant embarqué, eclipse et Sysgo CODEO pour simplifier l'architecture et la configuration du logiciel du routeur SNG et enfin qemu-x86 pour émuler la cible matérielle embarquée.

Nous avons aussi développé un logiciel nommé « Métrix » pour quantifier la qualité du code source puis l'avons appliqué pour améliorer le logiciel du routeur. Les

métriques mesurées sur le code généré par le transformateur GeneAuto orientent le programmeur pour décider des blocs du logiciel à retravailler. Cependant, l'absence de métrique au niveau des modèles limite la portée de l'analyse et est donc une piste d'évolution pour la suite de nos travaux.

Maintenant que le processus méthodologique de développement du routeur SNG a été présenté, nous pouvons approfondir la mise en œuvre des fonctionnalités identifiées au chapitre précédent à la section 2.3. Les modélisations et les choix de conception associés sont l'objet du prochain chapitre, le chapitre 4.

# 4. Mise en œuvre du routeur SNG

Le contexte de l'élaboration du routeur SNG a été présenté au chapitre 2, ainsi que les fonctionnalités attendues de notre routeur. Le chapitre 3 a présenté les aspects méthodologiques de la mise en œuvre du routeur SNG, notamment au travers de la méthodologie novatrice créée pour et appliquée au routeur SNG.

L'objet de ce chapitre est de développer les aspects techniques, les choix de mise en œuvre et les solutions apportées pour construire le logiciel de notre routeur SNG et mettre en œuvre les fonctionnalités énoncées dans la section 2.3. La première section est dédiée à l'architecture du logiciel du routeur SNG et présente donc une vue globale de la décomposition modulaire du logiciel. Elle est suivie de quatre sections détaillant les quatre modules du logiciel.

## 4.1. Architecture du logiciel du routeur SNG

Le routeur doit assurer plusieurs fonctions, développées dans la section 2.3. Le routeur doit filtrer et router les paquets IPv4 et IPv6 entre ses différentes interfaces réseau ; ces fonctions « routage» et «filtrage» forment le premier groupe de fonctionnalités.

Ensuite, le routeur doit pouvoir assurer la sécurité des communications en garantissant la confidentialité et l'intégrité des échanges via des canaux sécurisés, après avoir authentifié les nœuds d'extrémités de ces canaux. Pour limiter une prolifération de canaux sécurisés peu utilisés et donc un gaspillage de ressources réseau, le routeur peut être configuré pour mutualiser ces canaux et ainsi sécuriser plusieurs flots de données dans un même canal. Le deuxième groupe de fonctionnalités est constitué de ces fonctions de multiplexage et de sécurisation des données.

Un troisième groupe de fonctionnalités est relatif au contexte avionique pour lequel est destiné le routeur. En effet, les interfaces réseau du routeur sont des interfaces AFDX, donc des interfaces Ethernet auxquelles certaines exigences fonctionnelles ont été ajoutées pour garantir un meilleur déterminisme.

Enfin, les exigences non fonctionnelles telles que la sûreté et la sécurité du système « routeur SNG » concernent tout le système et toutes les fonctionnalités. Bien que formant un quatrième groupe, elles impactent toutes les autres fonctionnalités et les processus de conception et de mises en œuvre. Elles ne se traduisent pas forcément

par du code ou des modèles et sont donc traitées d'un point de vue méthodologique, comme développé dans le chapitre 3 précédent.

La figure 4.1 présente l'architecture du routeur SNG. Celle-ci formalise et « fige » la décomposition des fonctionnalités en trois classes, détaillées ci-dessous. Cette décomposition fonctionnelle préconisée dans la première étape de notre méthodologie permet de se concentrer pour chaque classe sur le sous-ensemble de fonctions à modéliser en faisant abstraction des sous-ensembles associés aux autres classes, ce qui permet de simplifier la modélisation.

La classe Pfr est associée aux fonctionnalités de routage et de filtrage des paquets, elle est donc fortement liée au protocole de niveau réseau. Étant donné que le routeur est destiné à interagir avec le protocole IPv4 mais aussi avec le protocole IPv6, deux variantes ont été modélisées, l'une pour IPv4 appelée Pfr4 puis son dual Pfr6 pour le protocole IPv6.

Ces deux classes Pfr4 et Pfr6 présentent une interface commune vis-à-vis des autres classes de partition du système. Ainsi, la classe de partition abstraite Pfr définit une interface commune liant les entrées et sorties de la classe de partition de routage et de filtrage Pfr avec celles de la classe de partition d'interfaçage Piface présentée plus loin. Le mécanisme d'héritage est utilisé pour dériver les deux classes Pfr4 et Pfr6 de la classe-mère abstraite Pfr ; ces deux classes héritent donc d'une interface d'entrées/sorties unifiée.

Cependant, bien qu'IPv6 soit spécifié comme le remplaçant d'IPv4, ces protocoles sont trop différents pour mutualiser les mises en œuvre des classes Pfr4 et de Pfr6. L'héritage n'a donc pas été utilisé ici pour mutualiser plus encore la conception de ces deux classes de partition.

La section 2.3.1 indique que le routeur n'a pas besoin d'agir en tant que passerelle protocolaire, rendant inutile d'éventuelles interactions entre ces deux classes de partition Pfr4 et Pfr6.

La classe Piface permet de lier les classes gérant les données (Pfr4 et Pfr6) aux entrées/sorties matérielles. Cette liaison consiste en pratique à réceptionner les trames Ethernet reçues par les cartes réseau, à désencapsuler les données de ces trames et à les transmettre à la classe adaptée (une instance de Pfr4 pour les paquets IPv4, une instance Pfr6 pour les paquets IPv6 ou encore rejeter les trames invalides et de protocoles inconnus). Cette liaison est bidirectionnelle, c'est-à-dire que Piface assure aussi l'émission des paquets IP sur le réseau physique.

Enfin, les fonctionnalités spécifiques à la sécurisation des données par le routeur sont mises en œuvre par une classe de partition dédiée, la classe Pse (« Partition de SÉcurisation »).

Les classes de partition communiquent via des associations représentées sur le diagramme de classe 4.1. Ces associations se traduisent à la mise en œuvre par des files de messages («*Message Queues*»), gérées par le système d'exploitation compatible

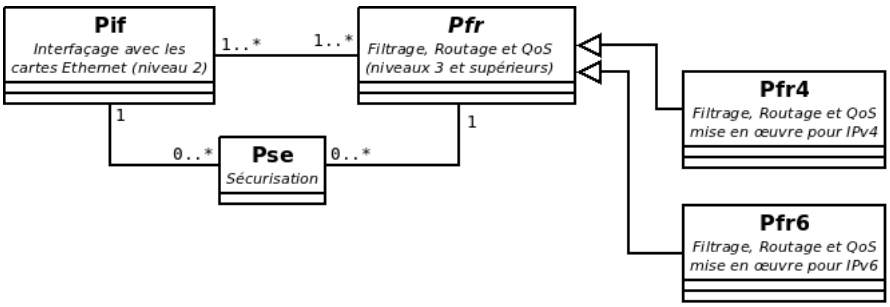



**Figure 4.1.:** Diagramme de classes du routeur SNG

IMA. Nous avons choisi arbitrairement d'utiliser des files dimensionnées à 1 message maximum par file avec un ordonnancement de processus par priorité statique préemptif pour que les instances de la classe Pfr soient prioritaires sur les instances de la classe Pif. Ainsi, les paquets sont traités dès leur arrivée par la partie logicielle du routeur SNG et mis éventuellement en attente dans la file d'attente matérielle de la carte Ethernet (cette dernière étant imposée à 256 trames Ethernet par le constructeur). Ces choix permettent de valider les exigences du routeur SNG, comme démontré dans le chapitre 6.

Le chapitre précédent a présenté la chaîne d'outils mis en place pour l'élaboration de notre routeur SNG. Les deux outils principalement utilisés dans ce chapitre sont les outils Simulink et Stateflow, qui servent ici à modéliser et mettre en œuvre les différentes fonctionnalités du routeur SNG, à l'aide de diagrammes de blocs et de diagrammes à états-transitions.

Nous avons ainsi choisi d'associer à chacune des classes du diagramme de la figure 4.1 un modèle Simulink encapsulant des sous-modèles Stateflow. Ces classes et leurs modélisations sont les objets des sections suivantes.

## 4.2. Pfr4, la partition de routage et de filtrage pour IPv4

La première classe de partition que nous présenterons est la classe Pfr4. Cette classe est dédiée à la gestion logicielle des paquets IPv4 transitant par le routeur SNG.

### 4.2.1. Contexte avionique et AFDX

Le contexte d'un routeur avionique connecté à d'autres systèmes avioniques critiques entraîne certaines contraintes spécifiques. Ainsi, le protocole réseau IPv4 utilise le protocole de liaison AFDX [ARINC664P7-1 2009, Ian Land 2009] qui est un dérivé du protocole Ethernet. Bien que compatibles, les systèmes AFDX n'exploitent qu'un sous-ensemble des possibilités offertes par le protocole Ethernet.

**Spécificité des mécanismes de niveau liaison** La principale différence qui impacte la conception du logiciel du routeur SNG concerne l'adressage de niveau liaison (les adresses MAC). En effet, le protocole Ethernet utilise 6 octets pour adresser la source et 6 autres pour la destination. Deux bits de ces 6 octets ont une signification particulière.

D'après le protocole Ethernet, le 7ème bit signifie une adresse universelle et unique s'il est à zéro et signifie une adresse locale ou encore privée, sans garantie d'unicité, dans le cas contraire. Le protocole AFDX impose aux adresses source et destination des trames AFDX d'avoir systématiquement ce bit à un : l'adressage avionique actuel est en effet statique, défini et configuré par le constructeur de l'avion.

De plus, un module de système avionique a toujours la même adresse MAC, quelque soit l'appareil sur lequel est monté physiquement le module. Cela permet de remplacer un module matériellement défaillant par un autre de même modèle, de manière transparente pour les autres systèmes avioniques.

Le 8ème bit indique que l'adresse est assignée à un système unique s'il est à zéro, à un groupe de systèmes s'il est à un. On qualifie d'adresses **«unicast»** les adresses de systèmes et de **«multicast»** les adresses de groupes. Ce bit permet aux systèmes intermédiaires (les commutateurs) de décider entre l'émission de la trame sur un seul lien physique ou sa diffusion sur l'ensemble des liens.

Dans le cas du protocole AFDX, toutes les adresses sources AFDX sont unicast et toutes les adresses de destinations sont multicast. Le multicast étant traduit au niveau liaison par un mécanisme de **broadcast**, les trames AFDX sont systématiquement diffusées à tous les systèmes présents sur le réseau de l'émetteur. Pour limiter la quantité de données dupliquées sur la couche physique, les commutateurs AFDX sont configurés en usine pour regrouper les systèmes en réseaux virtuels (appelés «Virtual Link»), physiquement présents sur le même commutateur mais logiquement indépendants entre eux.

L'utilisation du broadcast pour assurer le multicast au niveau 2 rend inutile l'emploi d'un protocole de découverte et de gestion des adresses MAC, tel que le protocole ARP (spécifié dans [Plummer 1982] puis complété par [Cheshire 2008, Arkko & Pignataro 2009]).

**Spécificité des mécanismes de niveau réseau** Au niveau IP se retrouve la même tendance à n'exploiter en aéronautique qu'un sous-ensemble des possibilités offertes par les protocoles.

Par exemple, la fragmentation n'est gérée que par les systèmes d'extrémité ; le réseau étant fermé et totalement connu par l'intégrateur de systèmes avioniques, ce dernier définit à la configuration le Maximum Transmit Unit (MTU) de manière à garantir le fonctionnement des systèmes dans tous les cas possibles, en évitant une éventuelle fragmentation sur le chemin par un équipement intermédiaire. Il a donc été choisi de ne pas concevoir ce genre de mécanisme dans le routeur SNG.

Pour des raisons similaires, les systèmes avioniques ne sont jamais confrontés aux cas de paquets IP avec des adresses IP de destination multicast et aux cas d'entêtes IP avec des options et extensions facultatives et donc des entêtes d'une taille supérieure à 20 octets [ARINC664P4-2 2007].

### 4.2.2. Cheminement des paquets

Lorsque des trames AFDX contenant des paquets IPv4 sont reçues par le routeur SNG, ces paquets IPv4 sont envoyés à une instance de la classe de partition Pfr4. Cette classe effectue les actions nécessaires avec les paquets pour les router conformément aux exigences décrites dans le SRS.

Pour cela, la classe a été modélisée avec Simulink. Le modèle global, présenté dans la figure 4.2, contient différents blocs logiques assurant chacun des fonctions nécessaires au routage des paquets.

Les paquets sont transmis par le système au premier bloc (nommé Schedule_4_to_1, à gauche sur la figure 4.2), qui les envoie à son tour à son successeur et cela se répète jusqu'à la sortie du paquet, marquant la fin des traitements. Ainsi, le paquet « chemine » d'un bloc logique au suivant au sein du diagramme de blocs Simulink.

Bien que la position des blocs sur le diagramme n'ait pas d'importance pour la transformation du diagramme en code source, nous avons choisi d'ordonner les blocs de gauche à droite dans l'ordre de traitement du paquet, afin de faciliter la lecture du diagramme.

Les blocs logiques sont tous modélisés par des machines à états avec Stateflow. Nous détaillerons trois blocs essentiels dans ce qui suit : le bloc d'ordonnancement chargé de transmettre les paquets un par un, le bloc de routage chargé de définir l'interface réseau de sortie à associer au paquet et le bloc de filtrage dédié au rejet des paquets correspondants aux règles de filtrage configurées par l'administrateur.

### 4.2.3. Ordonnancement

Le premier bloc logique appelé à la réception de paquets par le routeur est le bloc d'ordonnancement. Son rôle consiste à séquencer les paquets reçus pour les transmettre aux blocs suivants. Il assure un étiquetage interne des paquets pour permettre une ségrégation entre les flots de paquets reçus par les différentes interfaces réseaux du routeur.

Sa mise en œuvre est réalisée par la machine à états de la figure 4.3. Celle-ci est connectée à 4 interfaces réseau de réception des paquets et à une seule sortie connectée au traitement suivant pour les paquets. L'algorithme mis en œuvre ici est le round-robin [Silberschatz 2010], illustré dans la figure 4.4. Nous avons choisi cet

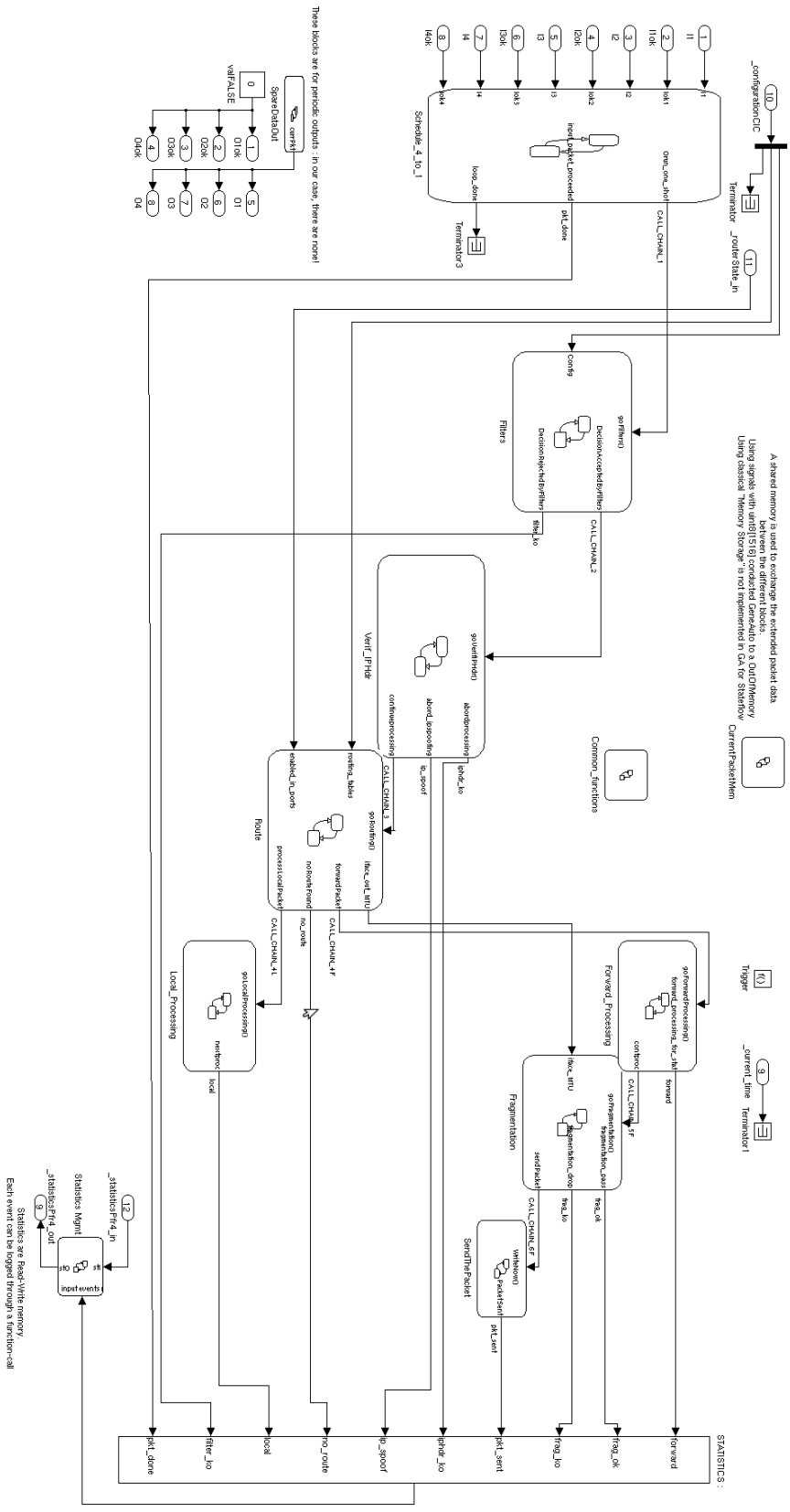



**Figure 4.2.:** Diagramme de blocs Simulink pour Pfr4

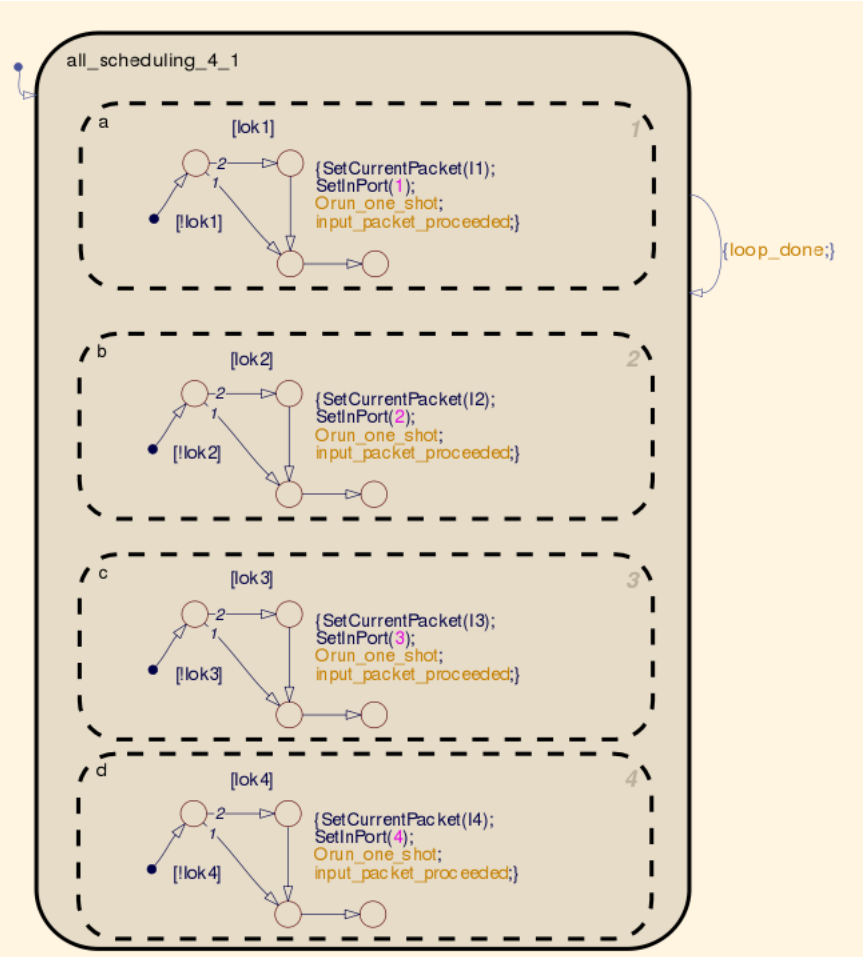



**Figure 4.3.:** Diagramme à états-transitions d'ordonnancement des paquets

algorithme afin d'éviter qu'un flot réseau sature le système et donc pour éviter les problèmes de famine.

Bien entendu, d'autres algorithmes plus complexes peuvent être implantés pour assurer une meilleure qualité de service (QdS), mais les besoins exprimés pour cette version du routeur SNG sont entièrement validés par cet algorithme.

### 4.2.4. Routage

Le domaine avionique actuel étant basé sur des réseaux statiques entièrement connus, nous avons mis en œuvre un mécanisme de routage exploitant une table de routage statique, définie durant la phase de configuration et figée à l'exécution du code du routeur. Cette table dont la structure est présentée dans la table 4.1 est utilisée par la machine à états de routage des paquets IPv4 illustrée figure 4.5. Ce diagramme à états-transitions a été modélisé à l'aide de Stateflow.

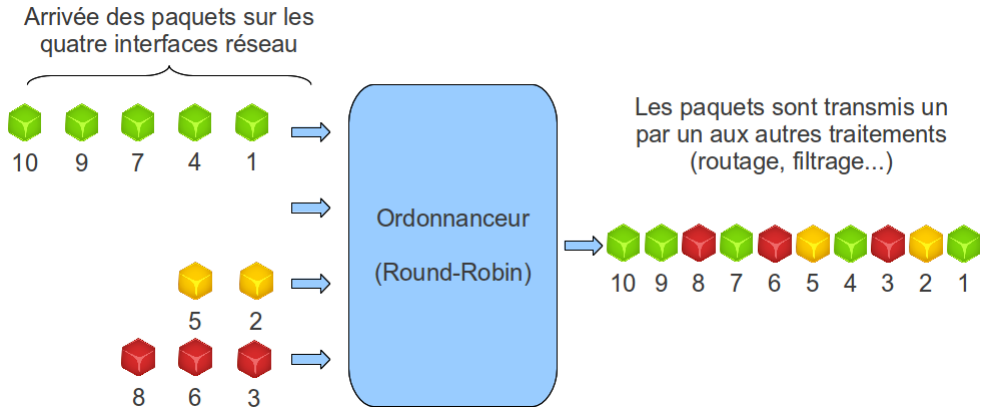



**Figure 4.4.:** Illustration du Round-Robin mis en œuvre par l'ordonnanceur

| Nom du champ | Position | Taille | Description |
|---|---|---|---|
| Adresse | 0 | 4 | Adresse du routeur sur le réseau de destination |
| Masque | 4 | 4 | Masque de sous-réseau associé à l'adresse |
| Port_Dest | 8 | 1 | Numéro de l'interface par laquelle émettre le paquet |
| MTU | 9 | 2 | Taille admissible du paquet |
| (réservé) | 11 | 1 | (octet de bourrage) |

**Table 4.1.:** Structure de la table de routage statique

#### 4.2.4.1. Description de la structure de la table de routage

La table de routage est un tableau de N entrées de routage, où N est une valeur numérique constante choisie durant la phase de conception en fonction des besoins. Une valeur faible limite le nombre maximal de routes traitées par le routeur mais économise de la mémoire. La table de routage étant parcourue pour router le paquet, une valeur élevée peut nuire aux performances et consommer de la mémoire.

Nous avons choisi N=32 pour nos expérimentations, cette valeur étant suffisamment élevée pour contenir toutes nos routes. La taille occupée en mémoire par la taille de routage est de 12*N, donc 384 octets dans notre cas.

Chaque entrée de routage est une structure de base, décrite par la table 4.1, comprenant l'adresse assignée au routeur sur le réseau, le masque du réseau, le port de sortie de la partition vers lequel orienter les paquets correspondant à cette entrée et le MTU associé (Maximum Transmission Unit = Taille maximale des fragments autorisés en sortie sur cette route).

Si un paquet de taille supérieure au MTU est reçu par le routeur, il considère le paquet comme invalide, donc issu d'un système défectueux et rejette alors le paquet silencieusement.

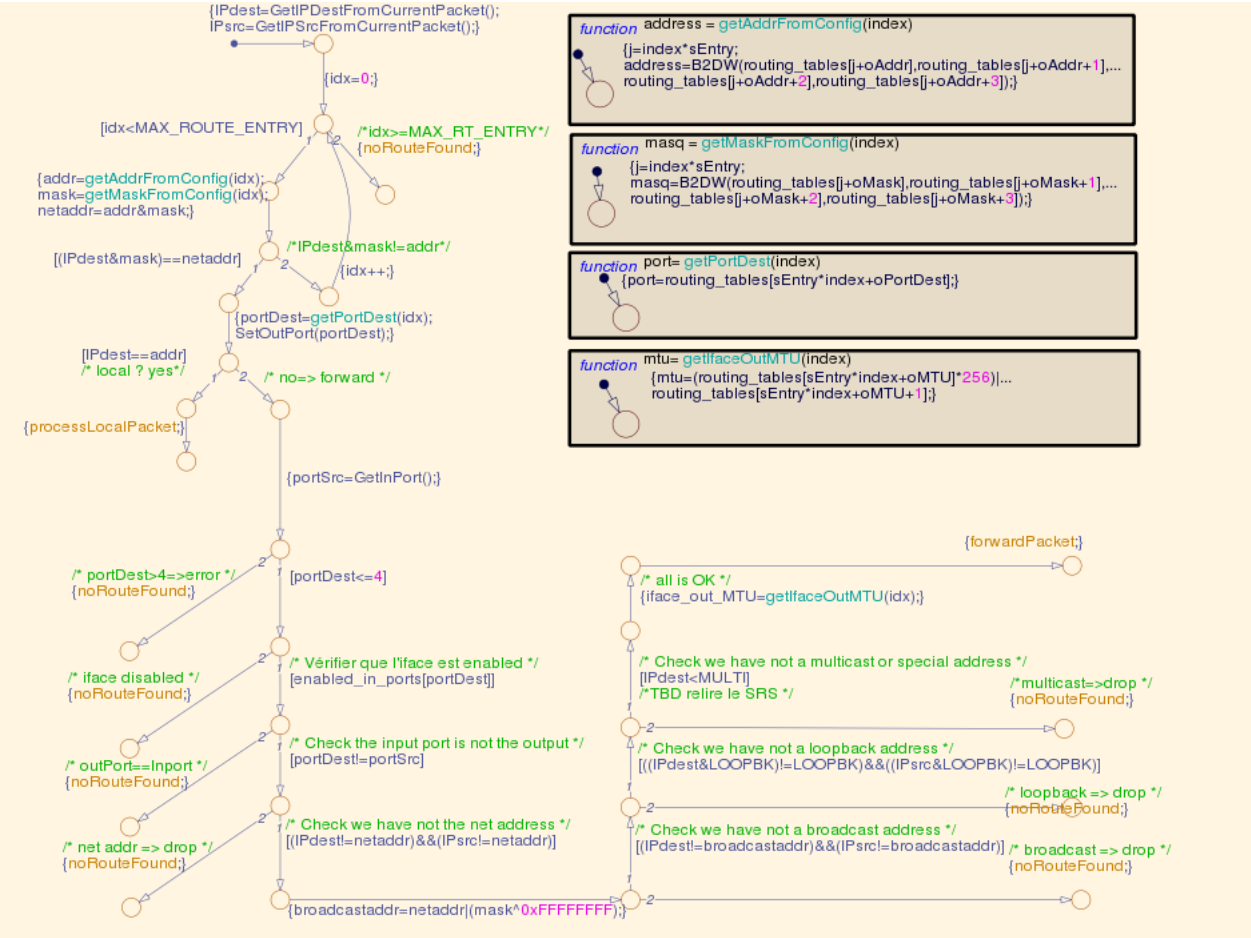



**Figure 4.5.:** Diagramme à états-transitions de routage des paquets IPv4

Un octet de bourrage a été ajouté afin d'aligner les entrées de la table de routage sur des multiples de 12, donc d'optimiser les accès en mémoire sur nos architectures 32 ou 64 bits actuelles.

#### 4.2.4.2. Explication de la mise en œuvre du routage

La figure 4.5 représente la machine à états routant les paquets IPv4 reçus par le routeur SNG. Le routeur teste itérativement chaque route, la première correspondance entraînant le comportement associé, «**Forwarding**» ou «**Local_Processing**», tous deux définis plus bas.

Si aucune route ne correspond, alors le paquet est rejeté. Le signal «No_Route_Found» visible sur le diagramme est alors déclenché et sert à incrémenter une variable interne destinée au déverminage du logiciel.

Dans le cas où l'adresse de destination du paquet correspond à l'adresse du routeur dans l'entrée de routage, la machine à états de routage redirige le paquet vers le sous-

| Destination | Passerelle | Genmask | Indic | Metric | Ref | Use | Iface |
|---|---|---|---|---|---|---|---|
| 169.254.0.0 | 0.0.0.0 | 255.255.0.0 | U | 1000 | 0 | 0 | eth1 |
| 10.31.4.0 | 0.0.0.0 | 255.255.255.0 | U | 1 | 0 | 0 | **eth0** |
| 0.0.0.0 | 10.31.4.1 | 0.0.0.0 | UG | 0 | 0 | 0 | - |

**Table 4.2.:** Table de routage de PC bureautique, issue de la commande «route -n»

système **Local_Processing**. Ce sous-système permet de répondre à une éventuelle requête ICMP ECHO [Postel 1981a, Conta *et al.* 2006], seul service local mis en œuvre par notre routeur pour faciliter les vérifications et validations du routeur SNG à sa configuration et à son installation dans l'appareil.

En masquant l'adresse du routeur avec le masque, on obtient l'adresse du réseau associée à la route. En masquant l'adresse de destination du paquet IP avec le masque de l'entrée de routage, on obtient l'adresse du réseau de destination du paquet. Dans le cas où ces deux adresses de réseau correspondent alors le paquet est routé vers le port de sortie via le sous-système **Forwarding**. Le MTU est renseigné pour rejeter le paquet le cas échéant. Un système de fragmentation éventuellement présent pourrait utiliser cette donnée pour déterminer s'il faut fragmenter ou non le paquet, mais le contexte avionique a rendu cette opportunité inutile.

Les paquets multicast (en fait, ceux dont l'adresse IP de destination est supérieure ou égale à 224.0.0.0) ou de boucle locale (l'adresse de destination est de la forme 127.x.y.z) sont systématiquement rejetés, de même que les paquets de diffusion générale (l'adresse de destination est formée de l'adresse du réseau suivie de bits tous à 1, par exemple 10.255.255.255 pour le réseau 10.0.0.0/8) et les paquets destinés au réseau (l'adresse de destination du paquet est une adresse de réseau).

Les paquets ne doivent pas être dirigés vers un port de sortie étant le même que le port d'entrée, afin d'éviter les attaques d'usurpation appelées « IP-spoofing ». Ils ne doivent pas être dirigés vers un port associé à une interface réseau désactivée (par exemple si le câble est débranché physiquement, ce qui est indiqué par l'entrée enabled_in_ports).

Les conditions de refus de routage précisées dans les deux paragraphes précédents entraînent le rejet du paquet. En effet, un paquet valide vis-à-vis des contraintes avioniques ne peut correspondre à l'une de ces conditions et le routeur rejette donc ces paquets qu'il détecte comme invalides.

L'approche graphique des modèles utilisés avec notre méthodologie est synthétique et complète (tout l'algorithme de routage tient sur une page), elle permet une relecture de modèle rapide. Le modèle de la figure 4.5 que nous avons créé résume visuellement l'ensemble des cas de routage de paquets et des traitements effectués.

| [index] | Adresse | Masque | Port_Dest | MTU | (réservé) |
|---|---|---|---|---|---|
| [0] | 10.31.4.1 | 255.255.255.0 | port0 | 1500 | 0 |
| [1] | 169.254.0.1 | 255.255.0.0 | port1 | 1500 | 0 |
| [2] | 0.0.0.0 | 0.0.0.0 | port0 | 1500 | 0 |

**Table 4.3.:** Exemple de table de routage adaptée à la structure de la table 4.1

#### 4.2.4.3. Exemple d'administration de ce mécanisme de routage

Les tables de routage classiques peuvent contenir une adresse de routeur du prochain saut, servant à identifier à quelle interface réseau envoyer le paquet, nécessitant éventuellement de parcourir plusieurs fois la table pour suivre l'enchaînement des entrées de routage et identifier l'interface de sortie.

Ainsi dans l'exemple de la table 4.2, pour identifier l'interface d'un paquet envoyé à 1.2.3.4, le système de routage va d'abord identifier la "direction" (ici, seule la route par défaut 0.0.0.0 correspond, renvoyant à 10.31.4.1), puis parcourir à nouveau la table pour trouver l'interface (ici, 10.31.4.1 a pour destination le réseau 10.31.4.0 et passe donc par l'interface eth0).

De plus, les tables de routage doivent être triées pour traiter les entrées du réseau de la plus spécifique à la moins spécifique, puis éventuellement prendre en compte d'autres métriques de routage basées sur la QdS.

Toutefois, dans la forme présentée dans ce document, le routeur SNG nécessite de "mettre à plat" cette table de routage : la table doit être triée et à toute destination possible doit être associée une interface de sortie et une adresse IP à associer au routeur. Il est à noter qu'écrire l'adresse du sous-réseau dans le champs Adresse au lieu d'une adresse d'hôte n'entraîne pas de problème, cela signifie alors que le routeur n'est pas directement connecté au réseau mais sait par quelle interface transférer les paquets pour ce sous-réseau.

Ainsi, la table d'exemple 4.2 présentée précédemment doit être convertie sous la forme de la table 4.3 pour être utilisée par le routeur SNG. Nous avons choisi d'utiliser un MTU de 1500 octets, comme celui configuré par défaut sur les deux interfaces réseau d'un PC bureautique Linux standard qui servira de cible de test pour l'évaluation des performances du routeur SNG dans le chapitre 6.

Avec cette table 4.3, le routeur répond aux pings sur 10.31.4.1 et redirige les paquets vers 10.31.4.0/24 via port0, il répond aux pings sur 169.254.0.1 et redirige les paquets vers 169.254.0.0/16 via port1, il ne répond pas aux autres pings et il dirige tous les autres paquets via port0.

Nous verrons au chapitre 6 l'évaluation des performances du routeur, dont les performances, acceptables pour le contexte d'utilisation de notre routeur SNG, résultant des choix que nous avons effectué dans cette section sur la mise en œuvre de la table de routage. La section 6.3.3.1 étudie spécifiquement l'impact de cette mise en œuvre sur les délais de routage des paquets IP.

| Nom du champ | Position | Taille | Description |
|---|---|---|---|
| Masque | 0 | 20 | Masque binaire à appliquer à l'entête IPv4 |
| Motif | 20 | 20 | Valeur binaire à comparer à l'entête après masquage |
| Décision_si_trouvé | 40 | 1 | Si correspondance du motif avec l'entête testé, faut-il accepter le paquet ? Le rejeter ? Tester le filtre suivant ? |
| Décision_si_pas_trouvé | 41 | 1 | Mêmes questions s'il n'y a pas de correspondance |
| (réservé) | 42 | 6 | (octets de bourrage) |

**Table 4.4.:** Structure d'une entrée de filtrage

### 4.2.5. Filtrage

La classe Pfr4 permet non seulement de router les paquets IPv4 qui transitent par le routeur mais propose aussi des fonctionnalités basiques de filtrage sans mémoire de ces paquets, plus couramment appelé « filtrage stateless »[Ingham 2002]. Ce filtrage vérifie l'entête IP de chaque paquet IPv4 entrant dans le routeur.

Le filtrage est réalisé par le diagramme à états-transitions Filters, modélisé à l'aide de Stateflow et représenté dans la figure 4.6. Cette machine à états prend 192 octets de configuration et retourne soit une décision d'acceptation du paquet (le traitement du paquet peut continuer), soit de rejet (le traitement s'arrête, le paquet est rejeté).

À l'instar de la table de routage, définie à la configuration et essentielle au routage IP, le filtrage IP est basé sur une table de filtrage constituée de M entrées structurées de 48 octets chacune. La structure utilisée est représentée dans la table 4.4.

Nous avons choisi M=4 pour nos expérimentations, car quatre filtres suffisaient à nos besoins. Toutefois, une version avec plus de 4 filtres peut facilement être dérivée en modifiant juste la constante NUMBER_OF_FILTERS du modèle de la classe Pfr4 puis en recompilant le projet de développement du routeur SNG.

Le filtrage consiste à appliquer de manière itérative chaque filtre disponible sur l'entête IP ; la décision à appliquer peut être :
– soit d'accepter le paquet, auquel cas le paquet est immédiatement transmis au sous-système suivant de la chaîne de traitement,
– soit de refuser le paquet, auquel cas le traitement passe immédiatement au paquet suivant, rejetant silencieusement le paquet,
– soit de continuer le traitement, c'est-à-dire d'appliquer les filtres suivants jusqu'à la prise d'une décision d'acceptation ou de rejet.

La décision appliquée dépend de la reconnaissance : si le motif est reconnu, c'est

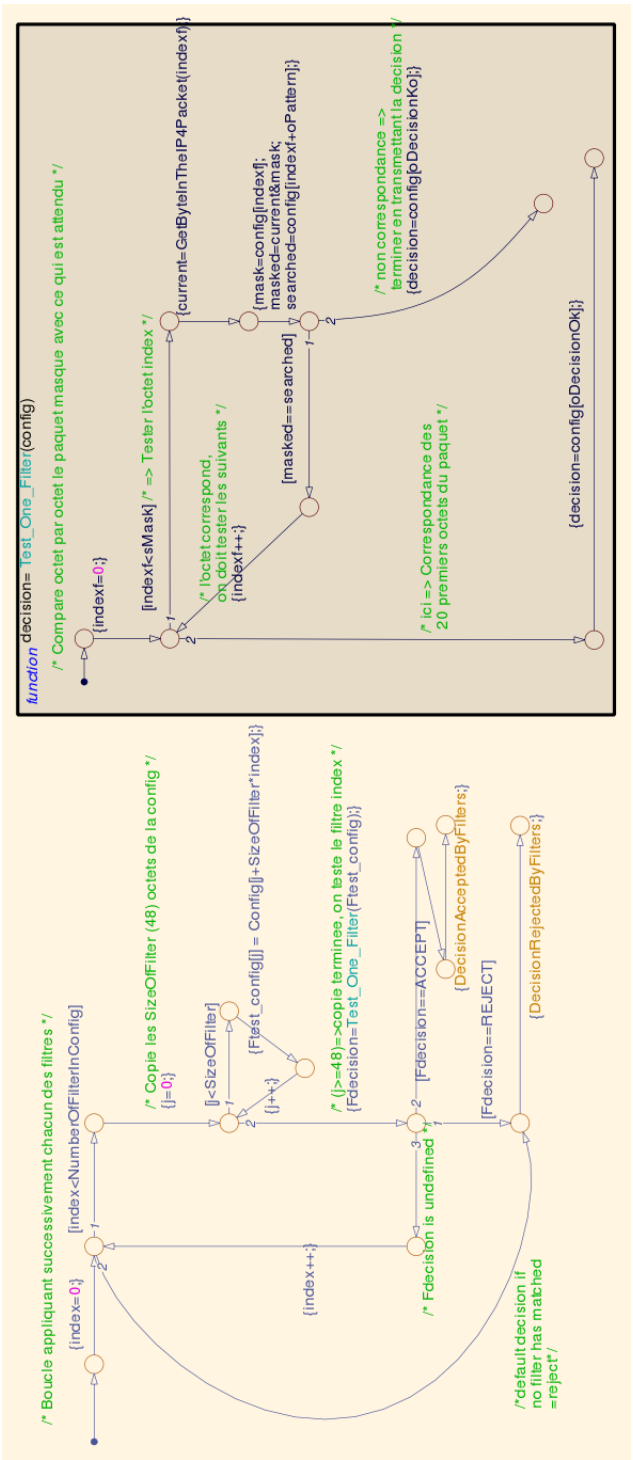



**Figure 4.6.:** Diagramme à états-transitions Filters

| [index] | Contenu de l'entrée de filtrage |
|---|---|
| [0] accepter si Ipdest==10.1.2.3/32 | Masque = {0..0, (ipdest) 255,255,255,255 ,0..0}<br>Motif = {0..0, 10,1,2,3, 0..0}<br>Décision_si_trouvé = Accepter<br>Décision_si_non_trouvé = Continuer |
| [1] rejeter si Ipdest=10.0.0.0/8 | Masque = {0..0, (ipdest) 255,0,0,0 ,0..0}<br>Motif = {0..0, 10,0,0,0, 0..0}<br>Décision_si_trouvé = Rejeter<br>Décision_si_non_trouvé = Continuer |
| [2] rejeter si TTL > 16 | Masque = {0..0, (ttl) 240 ,0..0}<br>Motif = {0..0, 0, 0..0}<br>Décision_si_trouvé = Continuer<br>Décision_si_non_trouvé = Rejeter |
| [3] accepter tout autre paquet | Masque = { 0..0 }<br>Motif = { 0..0 }<br>Décision_si_trouvé = Accepter<br>Décision_si_non_trouvé = - |

**Table 4.5.:** Exemple de table de filtrage à 4 entrées avec deux conditions

la décision indiquée dans Décision_si_trouvé, sinon c'est celle qui est dans Décision_si_pas_trouvé.

Appliquer un filtre sur l'entête Ipv4 du paquet consiste à masquer les bits du paquet IP que l'on ne veut pas tester puis à comparer les bits restants avec le motif recherché. Les champs Masque et Motif de l'entrée sont utilisés à cet effet et permettent de traiter les 20 octets de l'entête Ip, quelque soit le champ à filtrer.

Le filtrage se contente de la comparaison de systématiquement 20 octets, car les systèmes avioniques n'exploitent pas les options IPv4 donc les entêtes IP ont toujours une taille d'exactement 20 octets.

La table 4.5 est un exemple à but didactique à 4 filtres, acceptant inconditionnellement les paquets de 10.1.2.3, rejetant les paquets des autres nœuds du sous-réseau 10.0.0.0/8 et rejetant les paquets avec un TTL supérieur à 16.

Le comportement par défaut du routeur est le rejet des paquets, ce comportement s'applique quand aucun filtre n'a permis de prendre de décision. Ce choix a été décidé pour maximiser la sûreté.

Six octets de bourrage sont présents à la fin de chaque entrée de la table de filtrage, afin d'aligner les filtres sur des multiples de 16 dans un but d'optimisation du temps d'exécution du filtrage.

La machine à états-transitions de la figure 4.6 a une complexité cyclomatique supérieure à un (plusieurs chemins sont possibles pour aller de l'état initial à un état final) et la mise en œuvre pourrait conduire à boucler indéfiniment entre certains

états. Toutefois, notre méthodologie repose sur l'emploi d'un modèle formellement vérifiable ; ainsi, nous avons vérifié avec l'outil de modélisation qu'il n'y a pas de blocage possible à l'exécution du modèle : quelque soit l'état initial du système, celui-ci arrive toujours à un état final en un temps fini. De même, les outils de preuve formelle nous ont permis de vérifier la cohérence des types de données, évitant et corrigeant les cas de débordement d'entiers.

## 4.3. Pfr6, la partition de routage et de filtrage pour IPv6

Après avoir développé et mis au point la classe de partition Pfr4 de routage et de filtrage pour les paquets avioniques au format IPv4, nous avons dérivé cette classe pour gérer les paquets IPv6. À l'heure de la rédaction de ce manuscrit de thèse, seuls des paquets IPv4 transitent sur les réseaux embarqués, mais l'industrie aéronautique a prévu dès la conception du protocole AFDX l'évolution vers la version ultérieure et les usages nouveaux pourraient donc exploiter le protocole IPv6.

### 4.3.1. Ordonnancement, routage et filtrage

Concernant l'ordonnancement, le routage et le filtrage, nous nous sommes basés sur le modèle existant pour IPv4 que nous avons adapté pour supporter IPv6. Notre méthodologie de développement exploite en effet des modèles, qui sont facilement et rapidement réutilisables et adaptables, pour générer plusieurs variantes d'un logiciel ou réutiliser des modèles d'un logiciel dans un autre projet de développement logiciel.

Ainsi, la mise en œuvre de l'ordonnancement de paquets pour IPv4 présentée section 4.2.3 est identique en IPv6 ; aucune modification n'était nécessaire, étant donné que dans les deux cas, la taille maximale admissible des paquets est constante et fixée à 1500 octets. De même, les deux classes de partition ont été élaborées pour disposer de quatre interfaces réseau.

Le filtrage de paquets IPv6 a été légèrement modifié. Les entêtes IPv4 ont une taille fixe de 20 octets minimum, à laquelle peuvent s'ajouter des extensions optionnelles de tailles variables (mais toutes de tailles multiples de 4). L'absence d'utilisation de ces extensions dans l'avionique nous a conduit à mettre en œuvre un filtrage sans état sur les 20 premiers octets du paquet IPv4.

Les entêtes IPv6 ont une taille constante de 40 octets, les extensions éventuelles étant perçues comme des protocoles de niveaux supérieurs, encapsulées dans les données du paquet et donc non considérées comme partie intégrante de l'entête IPv6.

Ainsi nous avons modifié la constante de taille de données à filtrer, en remplaçant la valeur 20 par la valeur 40. L'algorithme de filtrage sous Stateflow n'a besoin

d'aucune autre modification pour mettre en œuvre un filtrage sans état des entêtes IPv6.

Le cas du routage IPv6 est plus compliqué. L'évolution entre IPv4 et IPv6 de la taille des adresses de 4 à 16 octets n'est pas en soi un problème ; il est géré en remplaçant la taille des variables manipulant ces adresses.

Par contre, il est nécessaire de détecter et traiter correctement les adresses IPv6 de groupe pour permettre l'utilisation du protocole NDP, présenté dans la section suivante. Un hôte IPv6 est en effet systématiquement membre de droit à des groupes de diffusion des messages transportant les informations sur le voisinage immédiat de l'hôte, sans nécessiter d'inscription ni d'initialisation préalable.

### 4.3.2. Le protocole Neighbor Discovery Protocol (NDP)

La principale nouveauté de la classe Pfr6 est le support du protocole Neighbor Discovery Protocol [Narten *et al.* 2007, Singh *et al.* 2010], abrégé NDP. Ce protocole remplace et complète le protocole ARP nécessaire à IPv4 pour la résolution d'adresses des protocoles de liaison.

Le protocole NDP permet en effet au protocole IPv6 de connaître les adresses de lien IPv6 et les adresses MAC associées lorsqu'IPv6 doit router un paquet ayant une adresse globale IPv6 en destination, via des paquets NDP, comme illustré sur la figure 4.8. De plus, le protocole NDP offre plus de possibilités aux utilisateurs que le protocole ARP.

Les objectifs de NDP sont les suivants :
– Découvrir mutuellement la présence des hôtes IPv6.
– Déterminer leurs adresses de niveau liaison (cf. ARP).
– Découvrir les routeurs au voisinage.
– Maintenir l'information sur l'accessibilité des noeuds.
– Configurer automatiquement les adresses.

NDP utilise des messages ICMPv6[Conta *et al.* 2006] encapsulés dans des paquets IPv6 pour fonctionner et assurer concrètement :
– La sollicitation (1) et l'annonce d'un voisin (2),
– La sollicitation (3) et l'annonce d'un routeur (4),
– L'indication de redirection (5), lorsqu'un paquet « rebondit » inutilement sur un nœud et que ce nœud indique à l'expéditeur du paquet comment accéder directement à la destination sans rebond.

À l'aide de diagrammes à états-transitions dont certains sont représentés sur la figure 4.7, nous avons mis en œuvre le traitement des cinq messages ci-dessus dans la classe Pfr6. Ensuite nous avons validé son fonctionnement et sa compatibilité avec les systèmes Linux à l'aide d'un programme utilisant le code source en C issu des modèles Simulink et Stateflow, ce qui est rendu possible par la souplesse de la méthodologie présentée dans le chapitre 3.


**Figure 4.7.:** Extrait des diagrammes à états-transitions pour le Neighbor Discovery Protocol

La figure 4.8 illustre le fonctionnement des sollicitations et des annonces des hôtes voisins et routeurs. Lorsqu'un nœud IPv6 souhaite communiquer avec un autre nœud dont il connaît l'adresse IPv6 globale mais pas l'adresse sur le lien physique, il émet une sollicitation sous forme d'un paquet multicast destiné au groupe des voisins. Le nœud ciblé répond par une annonce (« advertisement »), permettant ainsi aux deux nœuds de connaître les adresses de liaison et donc de communiquer. Un mécanisme similaire est disponible pour la découverte des nœuds IPv6 assurant le routage.

La spécification du protocole NDP prévoit certaines fonctionnalités additionnelles, telles que la vérification périodique de l'accessibilité d'un nœud via un échange sollicitation/annonce d'un voisin. Nous avons choisi de ne pas mettre en œuvre ces fonctionnalités facultatives pour simplifier les modèles, le protocole pouvant fonctionner correctement sans celles-ci.

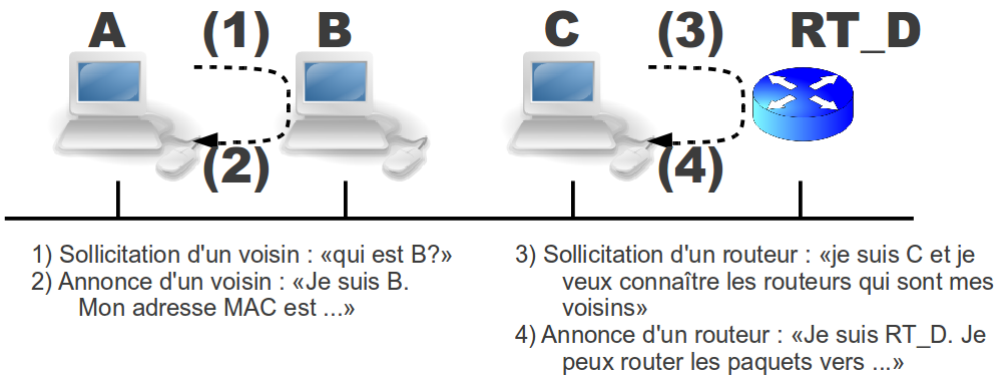



**Figure 4.8.:** Fonctionnement des sollicitations et annonces des hôtes voisins et routeurs avec NDP

La question du déploiement du protocole NDP dans le domaine avionique IPv6 reste posée, étant donné que le protocole ARP n'a jamais été utilisé en IPv4 dans ce domaine. Cependant, le protocole IPv6 n'ayant jamais été déployé dans un contexte avionique embarqué, la question de son utilisation ou de son abandon n'a pas été tranché, c'est pourquoi nous avons choisi de l'intégrer, au moins partiellement, dans notre routeur SNG.

## 4.4. Pse et les mécanismes de sécurisation des flux

Une des pierres angulaires du routeur SNG est la sécurisation des flux de données qui le traverse.

Dans le domaine avionique, il n'existe pour l'instant aucun standard imposé pour la sécurisation des flux. En pratique, les besoins de sécurisation sont résolus actuellement au cas par cas à l'aide du protocole XMPP [Saint-Andre 2011], sécurisant les données des applications critiques au niveau session avec le protocole SSL [Freier *et al.* 2011] ou son successeur TLS [Dierks & Rescorla 2008]. Cette approche client/serveur est adaptée pour des nouvelles applications de communication élaborées avec la contrainte et la possibilité de sécurisation « à la source », mais n'offre qu'une sécurisation application par application, ne permet donc pas la mutualisation des liens sécurisés et n'est pas facilement applicable pour sécuriser les systèmes déjà existants conçus initialement sans sécurité.

Notre routeur SNG travaille au niveau réseau (couche 3), il est globalement indépendant des applications et il est un nœud central de multiplexage des liens. Présentée plus loin dans la section 5.1, l'étude des solutions de sécurité a donc orienté le développement de la sécurisation des flux par le routeur vers la suite IPsec (voir à cet effet l'article [Alshamsi 2005] comparant IPsec avec SSL).

Cette suite protocolaire repose notamment sur le protocole Internet Key Exchange

version 2 (IKEv2 [Kaufman *et al.* 2010]) pour établir les canaux sécurisés et sur le protocole Encapsulating Security Payload (ESP [Kent 2005b]) pour sécuriser ce canal. Le choix de ces deux protocoles est détaillé dans les sous-sections suivantes.

Les canaux sécurisés utilisés avec ESP et établis avec IKEv2 doivent cependant être déclarés et configurés préalablement à leur utilisation : l'administrateur configure le système en précisant les canaux possibles et les identités des hôtes entre lesquels peuvent être établis les canaux, puis le canal est établi automatiquement lors du démarrage du système ou lors du premier échange de paquets à sécuriser. Le routeur SNG repose sur ce schéma (configuration statique préalable manuellement par un administrateur, établissement et utilisation des canaux automatisés) mais nous avons aussi travaillé sur l'automatisation de la configuration préalable, qui est plus adapté à un environnement réseau tel que l'Internet que celui de l'aéronautique. Nous avons abouti à la création du protocole SCOUT, dont la spécification, la mise en œuvre et la validation sont détaillés dans le chapitre 5.

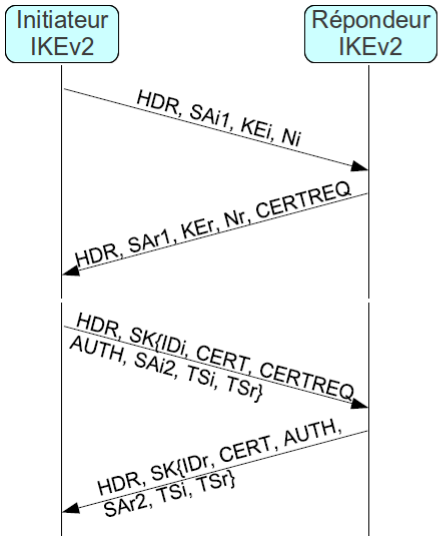
Nous avons choisi de privilégier la sécurisation à travers le protocole IPv6 pour traiter la faisabilité de la sécurisation des flux, notamment leur confidentialité, leur intégrité et leur authenticité : contrairement au protocole IPv4, le protocole IPv6 a été élaboré pour gérer nativement la sécurité d'après les documents de référence (comme écrit dans la spécification [Jankiewicz *et al.* 2011]).

L'exécution des fonctionnalités de sécurité est cloisonnée par notre méthodologie à une partition (cf. 3.3.6), réduisant les interactions avec le reste du système et donc les vecteurs d'attaque. Ce point est essentiel pour les garanties de sûreté et de sécurité de notre routeur SNG. Le cloisonnement permet non seulement de réduire les vecteurs d'attaque contre la partition Pse, mais elle garantit aussi qu'une vulnérabilité éventuelle exploitée dans cette partition ne devrait pas affecter les autres partitions du système.

### 4.4.1. Établissement du canal avec IKEv2 : création d'un tunnel sécurisé

La première étape pour pouvoir sécuriser des données consiste à établir un canal entre au moins deux nœuds du réseau. Cette phase de négociation permet aux entités communicantes de s'identifier et de s'authentifier mutuellement, d'échanger des informations telles que des clefs de chiffrement ou des nombres aléatoires pour fabriquer ces clefs, de définir quels mécanismes et quels algorithmes exploiter pour sécuriser les flux de données utilisateurs.

Différents protocoles d'établissement de canaux sécurisés existent, comme ceux énumérés dans la section 5.1. Nous avons choisi le protocole IKE car c'est un protocole de la suite IPsec couramment utilisé ([OpenSWAN 2012, StrongSWAN 2012, Weber 2001, Microsoft 2005]), les spécifications sont matures, éprouvées et faciles à trouver. Nous avons mis en œuvre la version 2 d'IKE [Kaufman *et al.* 2010] prioritairement à la version 1 [Harkins & Carrel 1998] car elle est non seulement plus




| Notation | Description |
|---|---|
| AUTH | Données d'authentification |
| CERT | Certificat d'authenticité |
| CERTREQ | Demande d'un certificat |
| HDR | Entête IKEv2 |
| IDi | Identification de l'initiateur |
| IDr | Identification du répondeur |
| KEi, KEr | Clefs, données transmises par l'initiateur, le répondeur |
| Ni, Nr | « Nonce », nombres aléatoires à usage unique |
| SA{i/r}{1/2} | Security Association |
| SK{ § } | § est chiffré et signé |
| TSi, TSr | « Traffic Selector », indications sur les flux à sécuriser |

**Figure 4.9.:** Séquence d'établissement de canal par le protocole IKEv2

souvent compatible entre les équipements de différents fournisseurs mais aussi plus simple à implémenter. Il est d'ailleurs indiqué en début de la première version officielle de la spécification d'IKEv2 [Kaufman 2005] que cette seconde version rend obsolète la première version d'IKE.

Le protocole IKE utilise le terme d'«Association de Sécurité» (SA) pour désigner un canal monodirectionnel sécurisé. À l'image de TCP, un canal bidirectionnel est alors formé de deux SA, respectivement appelées SA initiateur (SAi) et SA répondeur (SAr).

Comme illustré par la figure 4.9, le protocole IKE utilise typiquement une séquence de quatre messages pour établir le canal sécurisé.

1. L'initiateur du canal sécurisé envoie d'abord un message IKE_SA_INITi pour établir un « Security Parameter Index initiator » (SPIi), identificateur unique du futur canal, indiquer quels algorithmes et quels mécanismes de sécurisation sont supportés localement (SAi), un nombre aléatoire (Ni) pour générer des clefs, le matériel pour l'échange de Diffie-Hellman (KEi) et d'autres informations diverses (HDR).
2. Le récepteur répond alors avec ses propres informations (SPIr, SAr, KEr, Nr). Il peut exiger à ce moment un certificat de l'initiateur pour la phase suivante (CERTREQ). Toutes ces informations sont envoyées dans un message IKE_SA_INITr.
3. L'initiateur envoie ensuite un message IKE_SA_AUTHi, chiffré avec une clef SK calculée à partir des informations de l'échange IKE_SA_INIT (notamment KEi, KEr, Ni et Nr). Ce troisième message contient entre autres l'identité du nœud (NI), son certificat (CERT) si demandé, des informations sur l'adressage des flux à sécuriser (TSi, TSr) et un entête d'authentification (AUTH).

4. Le répondeur renvoie un quatrième message IKE_SA_AUTHr sur le même modèle avec ses propres informations.

À partir de ce quatrième message, un canal sécurisé existe et a été correctement établi, il est utilisable pour l'échange sécurisé de donnés, présenté plus loin dans la section 4.4.3. Après cette initiation, la spécification permet des échanges supplémentaires optionnels, notamment pour créer d'autres canaux (CREATE_CHILD_SA), changer le matériel de chiffrement (les clefs), reporter des erreurs (IKE_INFORMATIONAL) ou encore détruire un canal sécurisé. Nous invitons le lecteur plus intéressé par les détails de fonctionnement du protocole à consulter la spécification [Kaufman *et al.* 2010].

Le protocole IKEv2 a été mis en œuvre par des modèles Simulink et Stateflow pour le routeur SNG. Bien que des modèles formels aient déjà été utilisé pour valider la spécification ou le code source du protocole, à notre connaissance, c'est la première fois que ce protocole est mis en œuvre par l'intermédiaire d'une approche orientée modèle.

Ces modèles permettent de contrôler la mise en œuvre dès le stade de conception, pour s'assurer notamment de l'absence de blocage et prouver ainsi la terminaison des algorithmes. L'automatisation de la chaîne de transformation et de compilation, du modèle au code source puis au code binaire, apporte des garanties de conformité du code binaire vis-à-vis des spécifications du protocole. De plus, cela limite les phases d'évaluation et de certification à la validation des modèles et des transformateurs, évitant d'avoir à valider le code source intermédiaire complexe.

Les figures 4.10 et 4.11 sont extraites de la modélisation d'IKEv2 pour le routeur SNG. La première montre le modèle qui fabrique les messages IKE_SA_AUTHi et IKE_SA_AUTHr pour les envoyer, tandis que la deuxième présente le modèle associé qui vérifie et valide ces messages à la réception.

La figure 4.12 montre un message IKE_SA_AUTH intercepté et décodé par le logiciel Wireshark, entre son émission par un routeur SNG (ayant donc exécuté le code de la figure 4.10) et sa réception par un autre routeur SNG (qui le validera en exécutant le code de la figure 4.11).

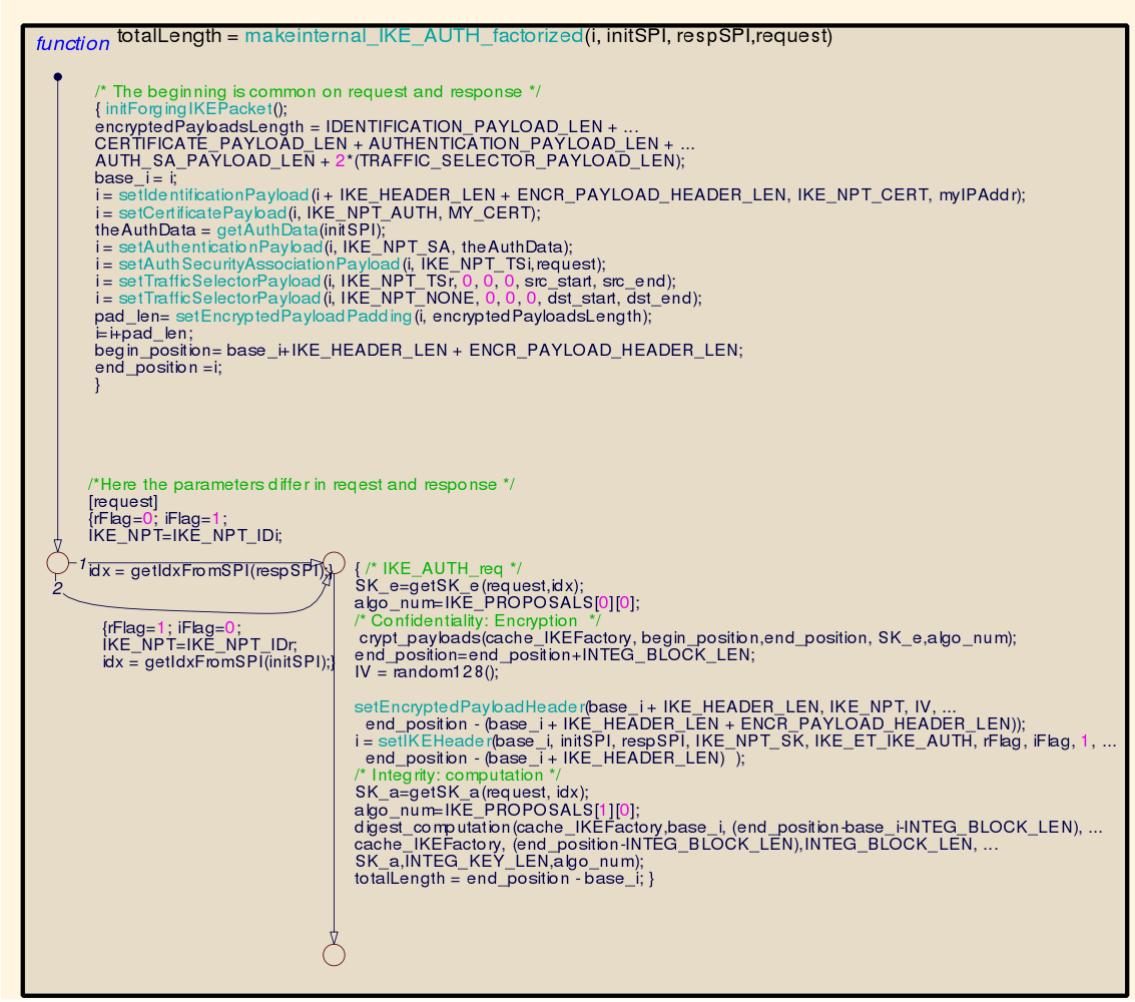



**Figure 4.10.:** Diagramme à états-transitions de fabrication des paquets IKE_SA_AUTH

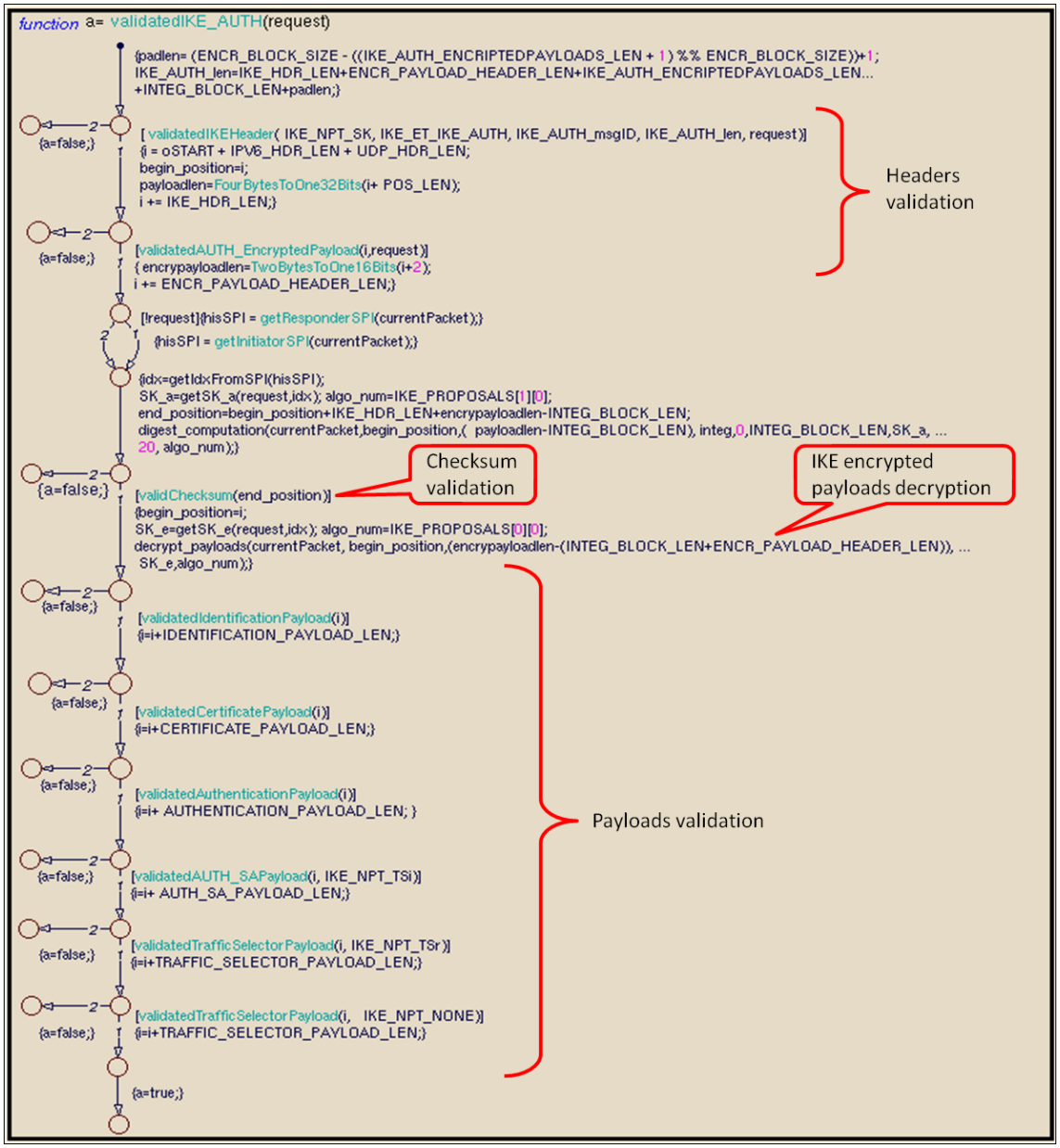



**Figure 4.11.:** Diagramme à états-transitions de validation des paquets IKE_SA_AUTH

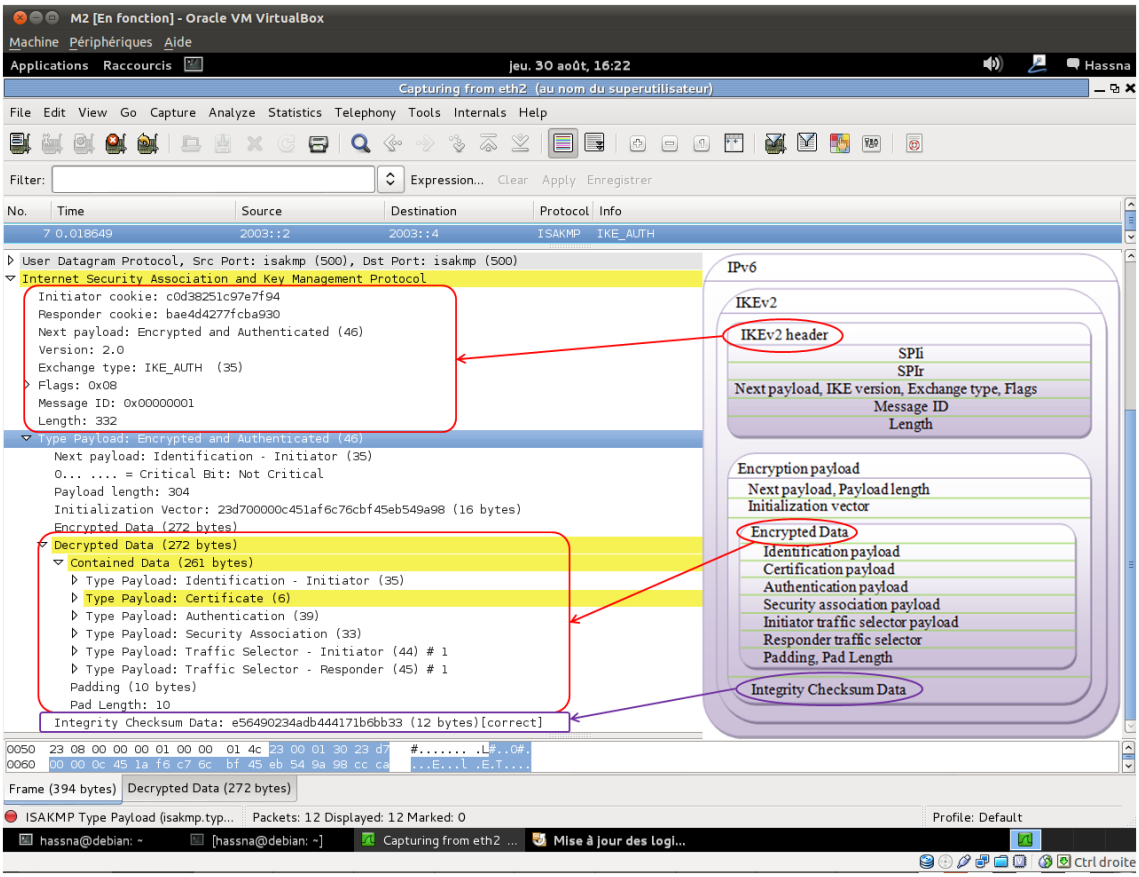



**Figure 4.12.:** Décodage avec Wireshark d'un paquet IKE_SA_AUTH échangé entre deux routeurs SNG

| Variable | Taille | Position | Description |
|---|---|---|---|
| gw* | 16 | 0 | Adresse IPv6 associée à l'autre extrémité du canal |
| IKE_SPI_local | 8 | 16 | Index de la SA sur cette extrémité du canal |
| IKE_SPI_distant | 8 | 24 | Index de la SA sur l'autre extrémité du canal |
| IKE_keys | 128 | 32 | Clefs et autres matériels pour IKE |
| ESP_SPI_local | 4 | 160 | Index de la SA pour le protocole ESP |
| ESP_SPI_distant | 4 | 164 | Index distant de la SA pour le protocole ESP |
| ESP_keys | 128 | 168 | Matériel de chiffrement et de signature pour ESP |
| seq_num | 4 | 296 | Entier «numéro de séquence» pour éviter les attaques par rejeu |
| num_prop_ike | 1 | 300 | Entier pour le routeur SNG indiquant quel ensemble d'algorithmes utiliser |
| num_prop_esp | 1 | 301 | Idem pour le protocole ESP |
| (réservé) | 210 | 302 | Réservé à un usage futur, permet d'aligner la prochaine entrée à un multiple de 512 octets |
| [ TOTAL ] | 512 | | |

*Nécessaire uniquement pour la phase d'initialisation

**Table 4.6.:** Structure d'une entrée SA dans la SADB

### 4.4.2. Stockage du matériel de sécurisation

Le protocole IKEv2 que nous avons mis en œuvre avec notre routeur SNG définit les séquences et les formats des échanges, fournissant ainsi un ensemble d'informations pour chaque canal sécurisé. Ces informations sont ensuite utilisées pour sécuriser des données, seuls les identifiants SPIi et SPIr sont présents dans les échanges postérieurs à l'initialisation.

Il a donc été nécessaire de mettre en place un mécanisme de stockage de ces informations, ainsi que des routines pour manipuler cette mémoire que nous avons appelée « Security Associations Data Base » (SADB). Ce stockage peut être vu comme un tableau bidimensionnel en mémoire, où chacune des N lignes est associée à un SA et contient 512 octets structurés comme expliqué par la table 4.6. La valeur N évolue entre 0 et une constante définie par le concepteur, que nous avons fixé à 16 pour nos expérimentations.

On peut voir une analogie de traitement de cette SADB avec les tables de routage

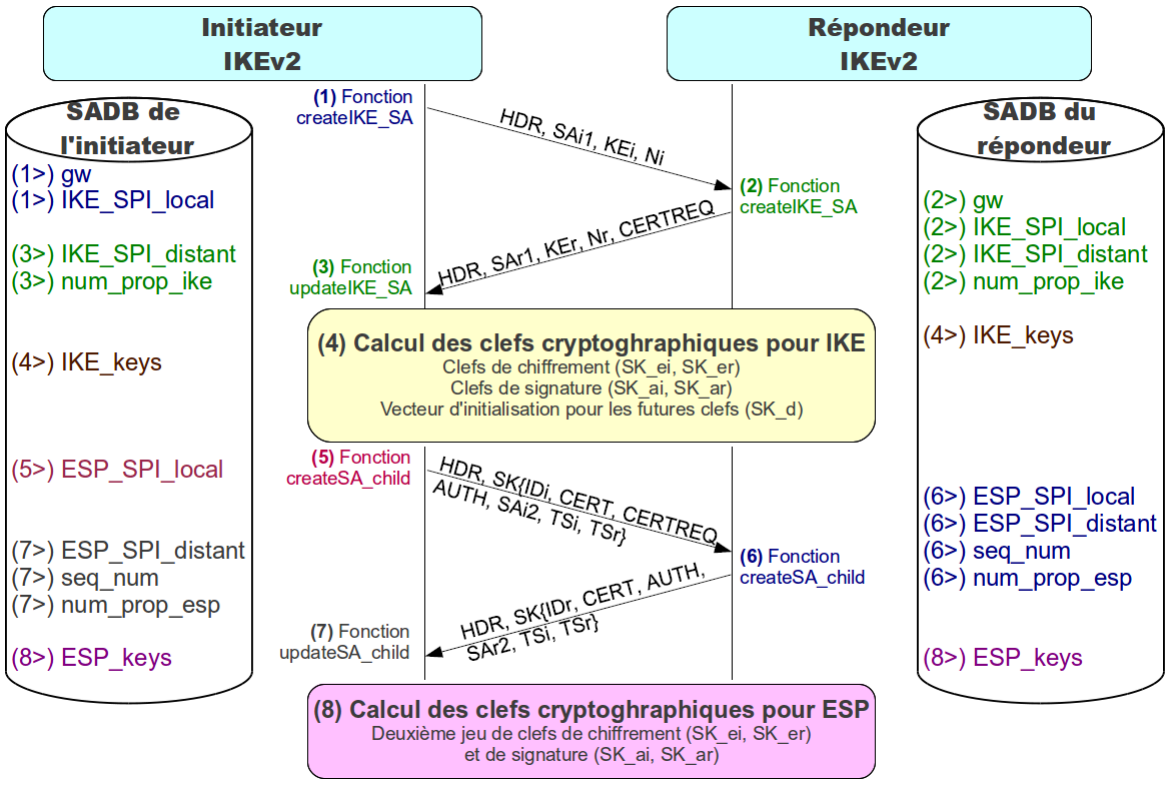



**Figure 4.13.:** Remplissage de la SADB par les messages IKEv2

et de filtrage. Lors de l'établissement d'un canal sécurisé, la SADB est complétée par une nouvelle entrée/ligne, au fur et à mesure des échanges, comme illustré dans la figure 4.13.

### 4.4.3. Sécurisation des données avec ESP : exploitation d'un tunnel sécurisé

Lorsqu'un canal a été établi à l'aide du protocole IKEv2, les flux de données peuvent transiter par ce canal de manière sécurisée. Pour cela, deux protocoles existent, les protocoles Authentication Header (AH, [Kent 2005a]) et Encapsulating Security Payload (ESP, [Kent 2005b]).

Le protocole AH permet de garantir l'intégrité des données échangées ainsi que leur authenticité (vérification que l'identité de l'expéditeur du message est légitime), mais il ne permet pas de garantir la confidentialité des données échangées. C'est pourquoi nous avons choisi d'utiliser à la place le protocole ESP qui offre les trois services dont nous avons besoin : confidentialité, intégrité et authenticité des données échangées.

Deux modes de fonctionnement existent pour ce protocole : le mode transport et le mode tunnel. Le mode tunnel chiffre, signe et encapsule intégralement le paquet IP à sécuriser dans un autre paquet IP créé à cet effet, d'où le nom de « tunnel ». La

taille du paquet résultant est donc la taille du paquet original plus une entête IP et une entête ESP.

Le mode transport modifie l'entête du paquet IP à sécuriser, chiffre uniquement le contenu du paquet original, sans l'entête et ajoute une signature calculée sur l'ensemble du paquet. Bien que le mode transport soit plus économe en taille de données échangées sur le réseau, il pose des problèmes de compatibilité avec les mécanismes de translations d'adresses (NAT [Srisuresh & Egevang 2001]).

C'est pourquoi nous n'avons mis en œuvre que le mode tunnel, présentant le défaut d'une moins bonne efficacité en terme de tailles de paquets mais insensible aux spécificités des réseaux donc «passe-partout». Dans la suite de ce manuscrit, nous emploierons indifféremment les termes de tunnels et de canaux sécurisés.

La figure 4.14 illustre l'encapsulation et la sécurisation de données par un tunnel ESP géré par un routeur SNG. Le logiciel Wireshark a capturé un échange de «ICMPv6 ping ECHO», ce mécanisme de requêtes/réponses permet notamment de s'assurer de la connectivité entre deux nœuds du réseau.

La capture réseau montre à la fois la requête avant sécurisation, puis un paquet «ESP» correspondant à un paquet sécurisé qui encapsule le paquet de requête, mais ici Wireshark n'a pas été configuré avec les clefs secrètes de chiffrement et est donc incapable de déterminer le contenu chiffré du paquet ESP.

Ensuite un paquet ESP est reçu et décodé par le routeur SNG qui désencapsule et déchiffre les données, puis émet le paquet résultant : la réponse ICMPv6 correspondante à la requête initiale.

### 4.4.4. Mécanismes et algorithmes de sécurisation

Nous avons évoqué les mécanismes et algorithmes de chiffrement et de signature dans les sections précédentes sans préciser lesquels ni comment ils sont mis en œuvre. Le choix de ces mécanismes est déterminé par les possibilités énumérées dans les spécifications des protocoles (IKEv2 et ESP dans notre cas du routeur SNG) ainsi que par les choix de développement des logiciels.

Dans le cas où plusieurs algorithmes sont supportés par les deux extrémités du canal, le choix est arbitraire ; si aucun algorithme n'est supporté simultanément par les deux extrémités, l'établissement du canal sécurisé échoue.

Après avoir étudié les différents algorithmes, nous avons choisi de mettre en œuvre les algorithmes AES-CBC-128 [Frankel *et al.* 2003, NIST 2001] pour la confidentialité des échanges de données et HMAC-SHA1 [Madson & Glenn 1998, NIST 2008, NIST 2012] pour signer les messages (authentification + intégrité). Ceux-ci semblent, à l'heure de l'écriture de ce mémoire, résistants aux attaques par cryptanalyses (d'autres algorithmes étaient déclarés comme «cassés» et donc déconseillés).

L'algorithme AES-CBC-128 utilise une clef de 128 bits, soit 16 octets, pour chiffrer les données. Il est facilement extensible à des clefs de plus grande longueur, comme

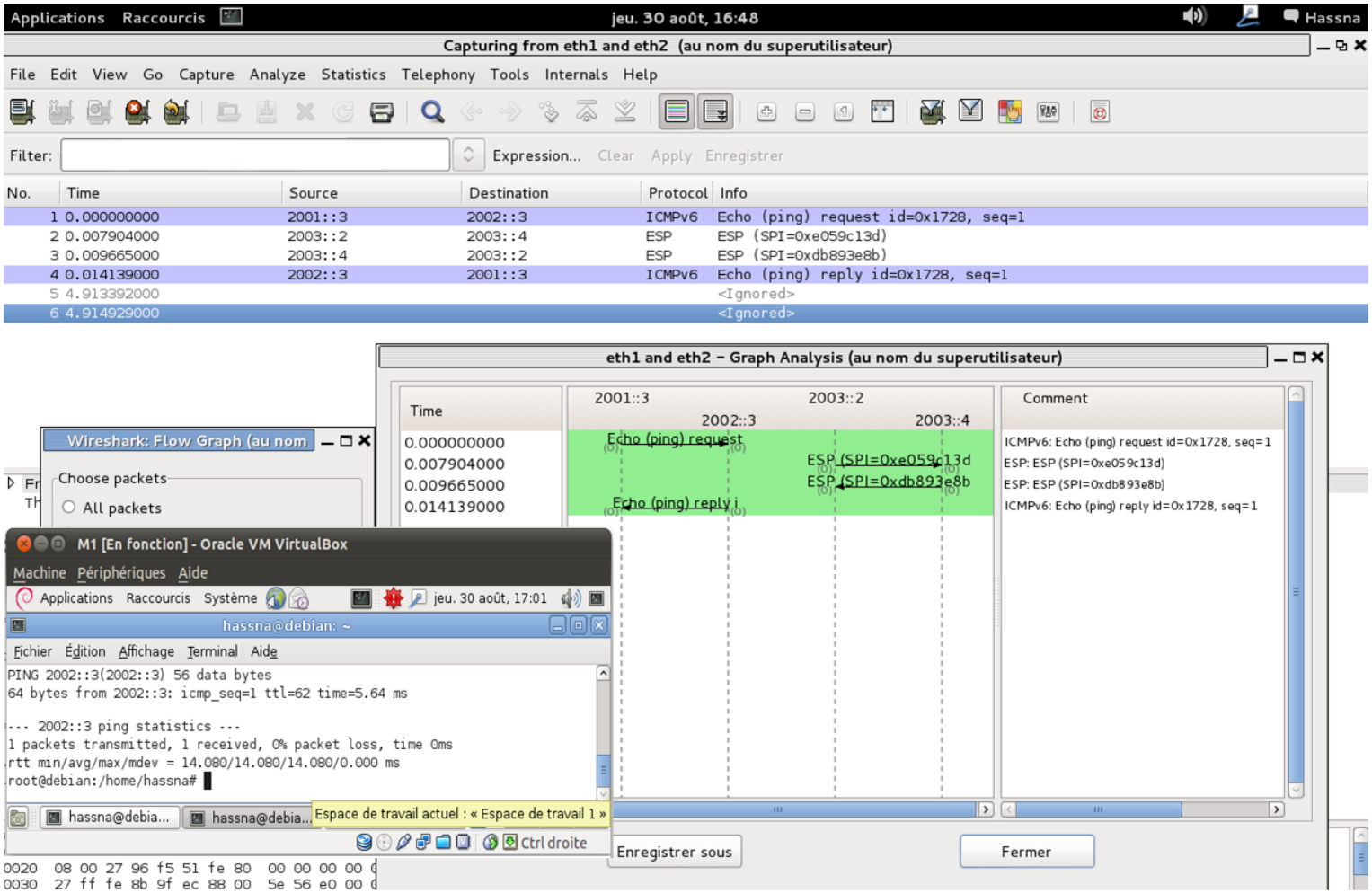



**Figure 4.14.:** Encapsulation et sécurisation de données par un tunnel ESP

par exemple avec une clef de 256 bits (AES-CBC-256). Pour plus de détails sur le fonctionnement de cet algorithme, le lecteur peut consulter [NIST 2001].

L'algorithme HMAC-SHA1 utilise une fonction de hachage, la fonction SHA1, qui condense un message de longueur quelconque en un condensât de 20 octets (160 bits). La fonction HMAC, de l'anglais keyed-hash message authentication code, ajoute des informations d'authentification sur l'émetteur du message avant d'appliquer la fonction de hachage, le condensât garantit ainsi simultanément l'intégrité du message et son authenticité. Pour plus de détails sur le fonctionnement de cet algorithme, le lecteur peut consulter [NIST 2008] relatif à la fonction HMAC et [NIST 2012] expliquant la fonction de hachage SHA1.

La mise en œuvre des algorithmes AES-CBC-128 et HMAC-SHA1 a été facilitée par l'existence de codes sources libres réutilisables avec nos modèles. En effet, notre idée initiale de modéliser les opérations de chiffrement et de signature s'est rapidement opposée au problème de performance de notre routeur ; nous avons jugé que le besoin d'une vitesse élevée pour les opérations cryptographiques était essentiel et avons donc privilégié la réutilisation de code source en langage C.

Ce mélange de modèles et de codes sources dans différents langages (modèles Simulink et Stateflow et langage C) est rendu possible par la généricité de notre méthodologie, la mise en œuvre du « mélange » a été facilitée par l'utilisation du langage C comme langage intermédiaire, qui était à la fois le langage vers lequel sont transformés les modèles et le langage d'écriture des codes source libres que nous avons utilisés ici.

## 4.5. Piface, la partition d'interfaçage du matériel avec Pfr4 et Pfr6

La dernière classe de partition que nous présenterons est la classe Piface. Cette classe permet de lier les entrées/sorties du matériel aux entrées/sorties des applications : elle réceptionne les trames Ethernet reçues par les cartes réseau et les transmet dans un format approprié aux instances de partitions déjà présentées Pfr et Pse gérant le routage, le filtrage et la sécurisation des paquets IP.

Dans cette section, nous nous intéresserons d'abord aux relations particulières entre cette classe Piface et le reste du système, puis nous verrons le fonctionnement interne et la mise en œuvre de cette classe de partition.

### 4.5.1. Intégration de Piface dans son environnement

La classe de partition Piface est unique, car contrairement aux autres partitions qui assurent des fonctionnalités du système, cette partition a pour seule fonction de

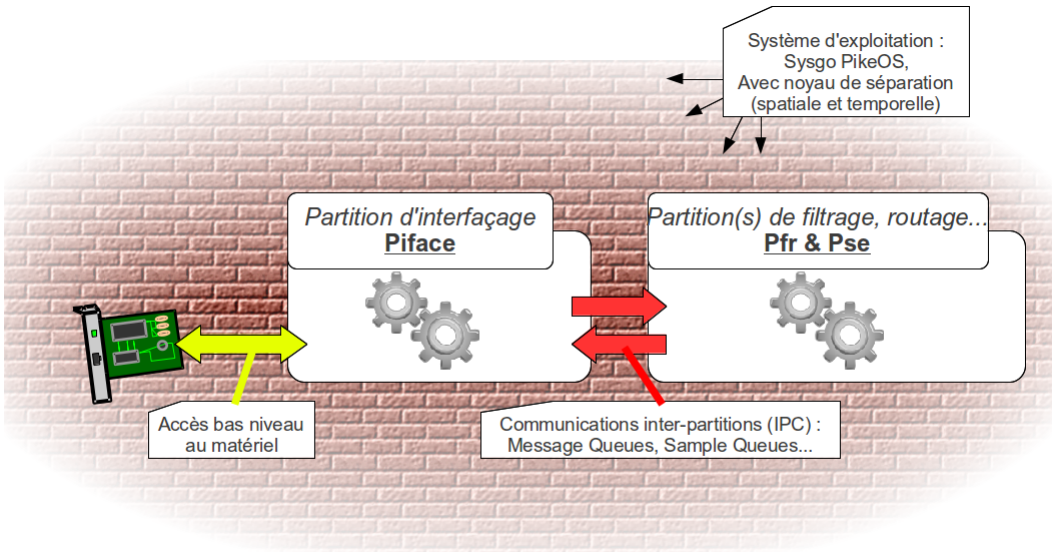



**Figure 4.15.:** Piface et son environnement

rendre les autres partitions opérationnelles. Une analogie peut être faite avec les services et dæmons d'un système d'exploitation, indispensables au bon fonctionnement des applications mais « transparents » pour les utilisateurs de ces applications.

Cette classe a deux objectifs complémentaires. Le premier est de recevoir les trames Ethernet, extraire les paquets contenus et les transmettre à l'application adéquate. Le deuxième objectif est le miroir du premier : recevoir les paquets des applications, fabriquer des trames Ethernet qui contiennent ces paquets et transmettre ces trames à la carte réseau pour émission sur le lien physique.

Cette classe de partition est en elle-même un logiciel autonome, un service, dont la ou les instances sont associées à des partitions PikeOS, Sysgo PikeOS étant le système d'exploitation pour notre routeur SNG embarqué choisi dans la section 3.3.6.

La figure 4.15 présente les différents types d'interactions de cette partition avec son environnement. Du côté « couches basses » matériel, la classe Piface communique avec la/les carte(s) réseau(x) pour recevoir et envoyer les trames Ethernet. Elle utilise pour cela des routines bas niveau d'accès aux pilotes de périphériques PCI.

De l'autre côté « couches hautes » logiciel, la classe Piface communique avec les applications, qui sont ici soit des instances de classes Pfr4, soit Pfr6, soit Pse. Elle utilise pour cela les primitives de communications inter partitions (IPC) fournies par le système d'exploitation.

### 4.5.2. La partition Piface vue de l'intérieur

Pour effectuer la liaison bidirectionnelle des données et l'adaptation de format entre le matériel et le logiciel applicatif, la partition Piface a été développée directement en

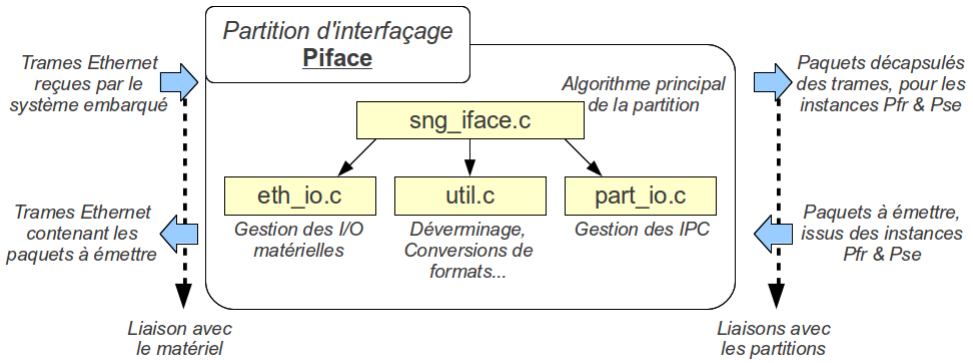



**Figure 4.16.:** Organisation du développement de Piface

langage C, contrairement aux autres classes modélisées avec Simulink et Stateflow. Ce choix est justifié par le besoin d'accéder au matériel, plus complexe à réaliser avec des modèles de haut niveau.

Ce choix est compatible avec notre méthodologie de développement : chaque partition étant développée de manière indépendante des autres partitions, développer entièrement une partition en écrivant manuellement le code source n'impacte pas le développement des autres partitions utilisant des transformations automatisées de modèles en code, telles que Pfr4, Pfr6 et Pse. Cependant, la vérification de cette partition Piface ne peut bénéficier des mêmes outils de vérification formelle que nous avons utilisé pour les autres partitions.

La figure 4.16 présente la structure du développement de cette partition. Les 753 lignes de code source en C sont réparties dans 4 fichiers (util.c, part_io.c, eth_io.c et sng_iface.c), chacun dédié à un sous-ensemble de fonctions logiques (respectivement le déverminage encapsulation désencapsulation, la gestion des IPC, la gestion des cartes Ethernet et enfin l'algorithme général liant l'ensemble).

Sur les 753 lignes de code constituant la partition, seules 282 (soit 37 %) sont des instructions, le reste étant des commentaires ou des lignes blanches pour aérer le code écrit manuellement.

L'algorithme de Piface est décrit dans l'algorithme 4.1 ; après avoir effectué les initialisations des différents modules, le programme boucle indéfiniment sur la lecture des trames puis la lecture des paquets, convertissant et transmettant les données quand il y en a.

## 4.6. Résumé du chapitre

Ce chapitre a présenté la mise en œuvre des fonctionnalités du routeur SNG : le routage, le filtrage et l'ordonnancement des paquets IPv4 et IPv6, la sécurisation des flux assurées par des tunnels sécurisés établis avec IKEv2 et exploités avec ESP

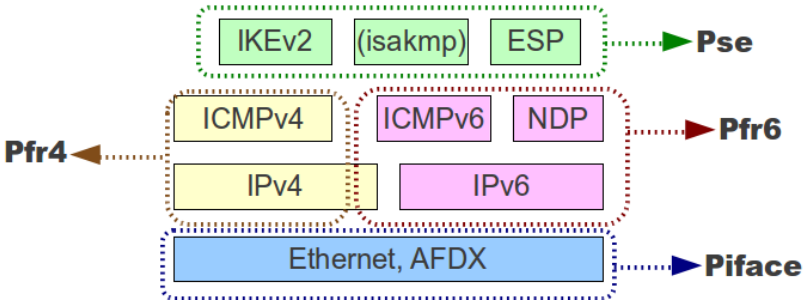



**Figure 4.17.:** Protocoles modélisés / mis en œuvre dans le routeur SNG

et enfin les liens de ces fonctionnalités de la couche réseau avec la couche inférieure de liaison et les spécificités de l'AFDX par rapport à l'Ethernet. La figure 4.17 synthétise les protocoles mis en œuvre par le routeur SNG ainsi que les classes de partition qui leur sont associées.

L'originalité introduite par le routeur SNG vient de l'emploi de modèles de haut niveau dans la mise en œuvre de protocoles existants, dont la majorité ne sont pas encore déployés dans le monde aéronautique mais pourraient l'être prochainement.

La puissance d'expressivité des modèles prônés par notre méthodologie apporte en plus de la rapidité de conception et de relecture des avantages en terme de réutilisation et d'adaptation des modèles, ainsi que des garanties de sûreté et de sécurité conséquemment à la validation des modèles à l'aide de preuves formelles. L'application de notre méthodologie et des outils associés maintient ces garanties jusqu'aux étapes finales de validation du système sur émulateur et sur cible embarquée réelle. Notre méthodologie est de plus compatible avec des approches plus classiques où le code source est écrit manuellement et non généré à partir de modèles, nous verrons au chapitre 6 sur l'évaluation des performances que cette approche peut être bénéfique lorsque les performances de certaines fonctionnalités sont critiques.

À notre connaissance, le routeur SNG est le premier système embarqué critique à mettre en œuvre un processus automatisé de développement depuis la modélisation de fonctionnalités et de protocoles de sécurité largement employés jusqu'à la validation sur cible réelle. La modélisation des protocoles de sécurité a ainsi été vérifiée formellement, en plus de servir à la génération du code source d'un logiciel opérationnel.

Le chapitre suivant présente et détaille le protocole SCOUT, un nouveau protocole dont le besoin est apparu dans la phase de définition du routeur SNG : découvrir dynamiquement les capacités de sécurisation des hôtes au sein du réseau.

**Algorithm 4.1** Programme principal de Piface (main() de sng_iface.c)

```
1  void piface_main(void)
2  {
3    /**** INITIALISATIONS ****/
4    /* Sysgo PikeOS: */   initialiser_pssw();
5    /* Matériel    : */   ouvrir_accès_vers_cartes_réseaux();
6    /* Applications: */   ouvrir_canaux_IPC_vers_partitions();
7
8    /* Changement du mode de la partition : INIT |---> RUN */
9    vm_part_set_mode(VM_RESPART_MYSELF,VM_PART_MODE_NORMAL,0,1,0);
10
11   /**** BOUCLE SANS FIN ****/
12   FOREVER {
13     /* Lecture trames */
14     FOREACH interface_réseau {
15       lire_trame_ethernet();
16       if(trame_lue_sans_erreur()) {
17          decapsuler_paquet_de_trame();
18          envoyer_paquet_vers_partition();
19       } /* if */
20     } /* foreach */
21
22     /* Lecture des paquets en retour */
23     FOREACH partition_d_application {
24       lire_paquet_de_partition();
25       if(paquet_lu_sans_erreur()) {
26         encapsuler_paquet_dans_trame();
27         envoyer_trame_vers_carte_reseau();
28       } /* if */
29     } /* foreach */
30   } /* forever */
31 }
```

# 5. Le protocole SCOUT

Le travail d'élaboration du routeur SNG nous a conduit à étudier et nous interroger sur les différents mécanismes permettant de sécuriser les communications et les protocoles associés à ces mécanismes. La sécurisation des réseaux de communication engendre aujourd'hui une charge importante de configuration.

C'est pourquoi nous proposons un protocole novateur de découverte automatique des capacités de sécurisation, afin de réduire cette charge. Notre protocole « **S**ecurity **C**apabilities discovery protocol **O**ver **U**nsecured **T**opologies » ou SCOUT a été conçu pour rechercher les mécanismes de sécurité supportés par les nœuds distants du réseau et déclencher le mécanisme d'établissement de canal sécurisé adéquat. Si l'hôte contacté ne peut prendre en charge la sécurisation, le protocole SCOUT tente alors d'établir un canal sécurisé avec le routeur voisin de l'hôte.

Ce protocole permet à l'administrateur de ne pas devoir configurer à l'avance un tunnel par couple de nœuds communiquant de manière sécurisée, ce qui réduit sa charge de travail et améliore la capacité de passage à l'échelle et de déploiement des solutions de sécurisation des réseaux.

Ce protocole a été élaboré initialement pour notre contexte aéronautique. En effet, dans le cas du routeur SNG, la première contrainte concerne la configuration : le routeur embarqué dans l'avion peut ne pas connaître à l'avance les destinataires avec lesquels un canal sécurisé sera nécessaire durant les différentes phases du vol, et l'ensemble des possibilités peut être trop vaste pour être préconfiguré de manière statique dans le routeur. Cela nous a obligé à chercher et inventer une solution dynamique de recherche des possibilités de sécurisation.

La possibilité d'établissement d'un canal sécurisé avec le routeur voisin de l'hôte répond à une deuxième contrainte de notre environnement aéronautique. Les matériels embarqués sont généralement associés à un processus très long de développement, de validation et de déploiement. Les systèmes actuels communiquent de manière non sécurisée et les développements de mises à jour pour intégrer des mécanismes de sécurisation s'avèrent complexes et coûteux. Déléguer la sécurisation à un équipement auxiliaire dédié à cet usage est une des solutions alternatives envisagées. Dans notre cas, « l'équipement auxiliaire » désigne bien évidemment notre routeur SNG.

Bien qu'envisagé pour répondre à nos contraintes aéronautiques, le protocole SCOUT est générique et a été mis en œuvre et validé sur l'Internet.

Après avoir introduit le protocole SCOUT et son fonctionnement, nous présenterons une évaluation des performances de sa mise en œuvre sur un réseau IPv6 natif, suivie d'une étude qualitative du gain apporté par SCOUT en terme de configuration pour les administrateurs réseaux. Enfin, nous présenterons une analyse des vulnérabilités potentielles de ce protocole qui conclura ce chapitre.

## 5.1. Introduction

Les mécanismes de sécurité sont de plus en plus utilisés dans les réseaux informatiques. La première et principale raison vient des évolutions techniques : les nouveaux systèmes disposent de ressources de calcul et de capacités mémoires en constante augmentation et ainsi les réseaux permettent des échanges à des taux de transfert de plus en plus élevés. La seconde raison vient de l'impact croissant de la sécurité informatique dans les politiques des systèmes d'information : les médias donnent aujourd'hui aux exploits des pirates informatiques une visibilité mondiale alors que la connaissance des attaques des réseaux restait dans des cercles très privés il y a encore quelques années.

Simultanément, les protocoles de sécurisation ont crû en sûreté, en sécurité et en efficacité. Ainsi, le protocole SSH [Ylonen & Lonvick 2006], créé en 1995, a évolué jusqu'à sa version actuelle et supporte désormais de nombreux nouveaux algorithmes de sécurité. Un autre exemple concerne la suite Internet Protocol Security (IPsec) [Kent & Seo 2005]. D'après la spécification du protocole IPv6 [Deering & Hinden 1998], les systèmes d'exploitation compatibles avec IPv6 doivent supporter la suite IPsec. Cette suite protocolaire définit deux modes de fonctionnement, les modes transport et tunnel.

Les canaux sécurisés peuvent être établis à l'aide de différents protocoles d'établissement de canal sécurisé, tels que les protocoles Internet Key Exchange (IKE et IKEv2 [Harkins & Carrel 1998, Kaufman 2005] ) et le protocole Kerberized Internet Negotiation of Keys (KINK [Sakane *et al.* 2006]). La suite IPsec repose alors sur les protocoles Authentication Header (AH [Kent 2005a]) et Encapsulating Security Payload (ESP [Kent 2005b]) pour sécuriser les données transitant par les canaux négociés à l'aide des protocoles IKE, IKEv2 et KINK.

Toutefois, ces protocoles nécessitent une charge importante de configuration afin de pouvoir établir et utiliser les canaux sécurisés. Il est en effet nécessaire de préciser à l'avance sur chaque extrémité de chaque canal les informations nécessaires à la négociation d'ouverture des canaux. Cette charge est un frein au déploiement des solutions de sécurité de niveau réseau, d'autant plus problématique que les topologies réseaux sont complexes. Cela peut consommer une part importante de temps et d'argent pour les administrateurs réseaux et les sociétés et administrations.

C'est pourquoi nous proposons un nouveau protocole afin de réduire cette charge. Notre protocole de découverte automatique des capacités de sécurité sur des topolo-

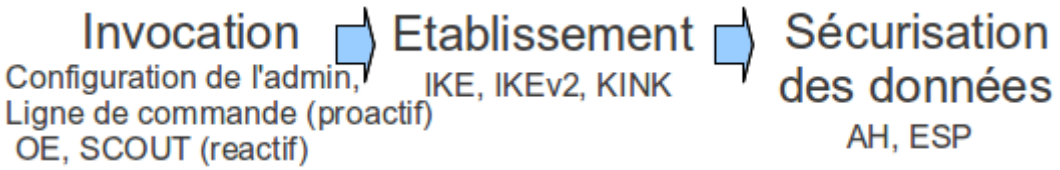



**Figure 5.1.:** Déclenchement, établissement et sécurisation

gies réseau non sécurisées a été nommé SCOUT : « Security Capabilities discovery protocol Over Unsecured Topologies ». Il a été conçu afin de rechercher les possibilités d'établissement de canaux sécurisés sur les nœuds du réseau et d'appeler les mécanismes adéquats pour établir ces canaux et ainsi sécuriser les échanges de données.

Le protocole SCOUT permet aux routeurs d'extrémité de sécuriser les données à la place des hôtes d'extrémité lorsque ces hôtes ne peuvent sécuriser eux-mêmes les données (par exemple en cas d'incompatibilités protocolaires ou de puissance de calcul trop faible). Le protocole SCOUT réduit la charge de l'administration de la sécurité du réseau en évitant à l'administrateur d'avoir à configurer à l'avance l'ensemble des canaux potentiels, c'est-à-dire ceux qui pourraient être utilisés pour sécuriser les données.

Les protocoles de sécurisation des données nécessitent, entre les entités communicantes, une négociation préalable des éléments de sécurité tels que les clefs de chiffrement ou l'ensemble des algorithmes à appliquer. Ces éléments peuvent être intrinsèquement liés aux protocoles employés. Ainsi, le Point-To-Point Tunneling Protocol (PPTP [Hamzeh *et al.* 1999]) utilise une connexion TCP pour gérer et contrôler le tunnel Generic Routing Encapsulation (GRE [Hanks *et al.* 1994]) qui encapsule les données dans des paquets PPP. Le protocole Extensible Authentication Protocol (EAP [Aboba *et al.* 2004]) est généralement utilisé dans les réseaux sans fils (Wireless Network).

Le protocole EAP permet de fournir des méthodes de négociation des canaux sécurisés. L'extension EAP-IKEv2 [Tschofenig *et al.* 2008] est une extension de ce protocole EAP avec la négociation faite par le protocole IKEv2 présenté plus bas. La suite IPsec permet de sécuriser les données au niveau de la couche 3 réseau. Elle repose sur le protocole Internet Security Association and Key Management (ISAKMP [Piper 1998, Kaufman 2005]) pour gérer localement l'ensemble des canaux sécurisés. La négociation des paramètres de sécurité peut être réalisée à l'aide d'un protocole dédié à l'établissement de canal sécurisé.

Différents protocoles permettent de négocier ces paramètres. Les canaux peuvent être négociés à l'aide du protocole Internet Key Exchange (IKE [Deering & Hinden 1998]) ou de son successeur IKE version 2 (IKEv2 [Kaufman 2005, Kaufman *et al.* 2010]). Le protocole Kerberized Internet Negotiation of Keys (KINK [Sakane *et al.* 2006]) est une alternative basée sur des tickets gérés d'une manière centralisée autour des serveurs Kerberos [Kerberos 2011].

Tous ces protocoles d'établissement nécessitent d'être invoqués par un élément externe, comme présenté figure 5.1.

Les protocoles d'établissement IKE, IKEv2 et KINK supposent que l'administrateur réseau a préalablement configuré les canaux sécurisés potentiels. Le protocole SCOUT que nous proposons permet d'automatiser ces étapes de configuration, réduisant ainsi la charge de travail de l'administrateur. Le protocole Opportunistic Encryption (OE [Richardson & Redelmeier 2005]) étend les implémentations d'IPsec OpenSwan [OpenSWAN 2012] et StrongSwan [StrongSWAN 2012] en proposant d'établir à la volée les canaux de communication, comme le protocole SCOUT se propose de le faire.

Cependant, le protocole OE n'a été conçu que pour les réseaux IPv4, il dépend d'un tiers de confiance devant sécuriser un serveur DNS avec le protocole DNSSEC [Arends *et al.* 2005], il ne permet pas aux routeurs de se substituer aux nœuds terminaux pour fournir la sécurité des flux de données et il n'est compatible qu'avec les protocoles IKE et IKEv2. SCOUT contribue ainsi à pallier ces limitations comme nous le détaillons dans la section suivante.

## 5.2. Fonctionnement du protocole SCOUT

### 5.2.1. Le protocole générique SCOUT

Lorsqu'un paquet sort par une interface réseau préalablement marquée par l'administrateur comme étant à sécuriser avec le protocole SCOUT, le protocole négocie un canal sécurisé. Il peut fonctionner de 4 manières différentes, en fonction du support ou non de SCOUT par les nœuds du réseau. Ainsi, la figure 5.2 présente une topologie basique avec deux hôtes d'extrémité H1 et H2, liés respectivement aux deux routeurs R1 et R2. Si les deux hôtes H1 et H2 supportent SCOUT, alors le canal sécurisé peut être établi entre H1 et H2. C'est le mode privilégié **SCOUT Host-Host** : quand H1 envoie son premier paquet vers H2, le dæmon SCOUT sur H1 intercepte ce paquet et négocie le choix d'un protocole d'établissement avec H2.

Un canal sécurisé est alors établi entre H1 et H2 et des règles de routage indiquent d'utiliser ce canal pour les flux H1-H2. Ainsi, après la phase de découverte puis celle d'établissement, tout paquet allant de H1 vers H2 ou en sens inverse est sécurisé au travers de ce canal.

Dans le cas où le routeur R2 supporte SCOUT mais l'hôte H2 ne le supporte pas, le protocole SCOUT permet l'établissement d'un canal sécurisé entre H1 et R2, c'est le mode **SCOUT Host-Router**. Quand H1 envoie son premier paquet à H2, le dæmon SCOUT de H1 l'intercepte et envoie une requête à H2 pour connaître les possibilités d'établissement d'un canal sécurisé. H2 n'exécutant aucun dæmon SCOUT, le système ignore les requêtes SCOUT qu'il reçoit et refuse la communication. H1 contacte alors le routeur R2 à proximité de H2. R2 répond positivement

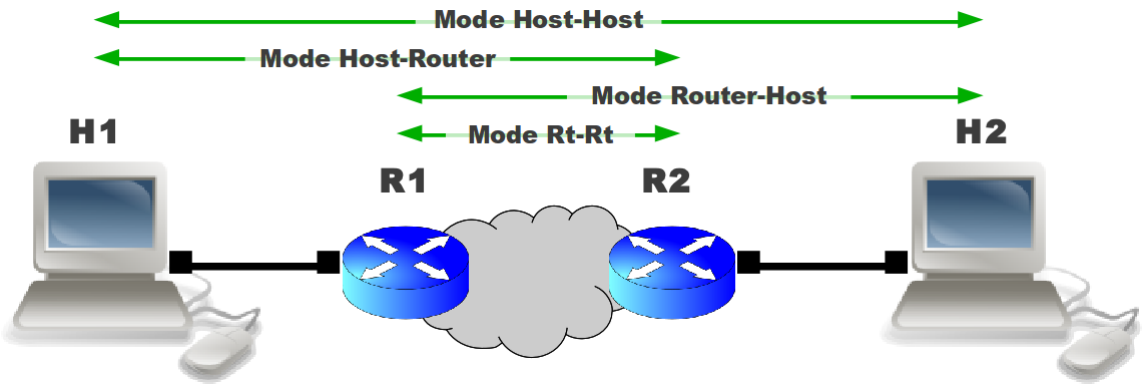



**Figure 5.2.:** Modes de SCOUT sur une topologie réseau basique

à la requête SCOUT de H1 et le canal sécurisé s'établit entre H1 et R2, afin de sécuriser les paquets transitant entre H1 et H2.

Si au contraire, c'est l'hôte d'origine H1 qui ne supporte pas SCOUT mais que le routeur R1 à proximité de H1 et la destination H2 gèrent SCOUT, alors dès que H1 transmet son premier paquet à H2 via R1, le dæmon SCOUT de R1 intercepte ce paquet et négocie l'établissement d'un canal sécurisé entre R1 et H2 ; c'est le mode **SCOUT Router-Host**. Enfin, dans le cas où aucun des deux hôtes H1 et H2 ne gèrent SCOUT mais où les deux routeurs R1 et R2 gèrent SCOUT, le protocole permet l'établissement d'un canal sécurisé entre R1 et R2 pour sécuriser les données de/vers H1 vers/de H2 ; c'est le mode **SCOUT Router-Router**.

Dans les 4 modes de fonctionnement représentés figure 5.2, les données sont sécurisées sur le tronçon « longue distance » entre les routeurs R1 et R2. Le protocole SCOUT a été élaboré afin de s'adapter de manière optimale aux topologies réseaux complexes, tout en réduisant la charge de son déploiement. Il nécessite en effet d'installer et de configurer le dæmon SCOUT sur certains nœuds du réseau : limiter le déploiement aux routeurs uniquement fournit un niveau de sécurité suffisant pour certains réseaux locaux et ne nécessite pas de modification des hôtes d'extrémité.

Ainsi, le protocole SCOUT rend possible la sécurisation des communications entre des hôtes ne pouvant sécuriser eux-mêmes leurs échanges. Ce cas survient lorsque les hôtes ont des ressources matérielles faibles (mémoire, processeur), fonctionnent avec des systèmes d'exploitation pour lesquels il n'existe pas d'implémentation de protocole de sécurité compatibles avec ceux des autres équipements du réseau...

Lorsqu'un seul nœud du réseau supporte SCOUT, la découverte ne peut aboutir positivement et les données ne peuvent donc pas être sécurisées automatiquement. Dans ce cas, le comportement par défaut du nœud consiste à rejeter les paquets ne pouvant être sécurisés.

La table 5.1 présente les différents cas de figure pouvant se présenter avec la topologie de la figure 5.2, selon que chaque nœud prend en charge ou non le protocole SCOUT.

<table>
<tr><td>H1 supporte SCOUT ?</td><td>oui</td><td colspan="2">non</td></tr>
<tr><td>R1 supporte SCOUT ?</td><td>(aucune importance)</td><td>oui</td><td>non</td></tr>
<tr><td>H2 supporte SCOUT</td><td>Mode Host-Host</td><td>Mode Router-Host</td><td rowspan="2">Rejet des paquets à sécuriser</td></tr>
<tr><td>R2 supporte SCOUT, pas H2</td><td>Mode Host-Router</td><td>Mode Router-Router</td></tr>
<tr><td>Ni H2 ni R2 ne supportent SCOUT</td><td colspan="3">Rejet des paquets à sécuriser</td></tr>
</table>

**Table 5.1.:** Modes résultant de la découverte selon la prise en charge de SCOUT

Pour résumer ce tableau, les données sont toujours sécurisées sur le réseau longue distance entre les routeurs d'extrémités ou alors elles sont rejetées si la découverte n'a pu aboutir à une compatibilité des solutions de sécurisation. Elles ne transitent jamais de manière non sécurisée sur le réseau.

Si les hôtes d'extrémité supportent le protocole SCOUT, alors ils prennent en charge la sécurisation des données. Dans le cas contraire, les routeurs sécurisent les données s'ils en ont la capacité, sinon les données sont rejetées.

### 5.2.2. Prérequis de fonctionnement de SCOUT

Certaines hypothèses doivent être réunies pour un fonctionnement optimal et sûr avec le protocole SCOUT. Premièrement, la liaison entre les hôtes d'extrémité et leur routeur voisin est supposée sûre : le mode SCOUT Router-Router ne sécurise les données qu'entre les routeurs d'extrémité et non entre les routeurs et leurs hôtes. Ce mode n'est donc acceptable que si les réseaux locaux sont sûrs, c'est-à-dire que la sécurité périmétrique est assurée (ou que SCOUT est limité par l'administrateur au mode Host-Host exclusivement). L'étude de la sécurité du protocole SCOUT, présentée section 5.4.6 dans ce document, contient un exemple d'attaque où les données sont découvertes sur le tronçon non sécurisé entre le routeur et l'hôte.

La deuxième hypothèse concerne les protocoles d'établissement des canaux sécurisés. Pour fonctionner, le protocole SCOUT requiert au moins un protocole d'établissement de canal sécurisé. Dans le cas où plusieurs protocoles d'établissements sont candidats, nous appellerons « protocole PE » celui qui est sélectionné pour établir le tunnel. PE peut alors désigner indifféremment IKE, IKEv2 ou KINK. Le protocole SCOUT prend en charge la découverte des protocoles susceptibles d'assurer l'établissement d'un canal, choisit le meilleur « PE »

| Mode SCOUT | H1 | R1 | R2 | H2 |
|---|---|---|---|---|
| **Host-Host** | Inspirateur et Initiateur | *(aucun rôle)* | *(aucun rôle)* | Répondeur et Destinataire |
| **Host-Router** | Inspirateur et Initiateur | *(aucun rôle)* | Répondeur | Destinataire |
| **Router-Host** | Inspirateur | Initiateur | *(aucun rôle)* | Répondeur et Destinataire |
| **Router-Router** | Inspirateur | Initiateur | Répondeur | Destinataire |

**Table 5.2.:** Inspirateur, Initiateur, Répondeur et Destinataire, selon les modes

suivant le contexte et ensuite invoque ce protocole PE pour créer le canal sécurisé et reconfigurer en conséquence les tables de routage des systèmes d'exploitation concernés.

La troisième hypothèse concerne aussi le protocole d'établissement « PE ». Le protocole SCOUT fait confiance dans la sécurité de ce protocole PE pour l'authentification des extrémités du tunnel ; SCOUT identifie les extrémités entre lesquelles établir un canal sécurisé et transmet cette information au protocole PE qui authentifie les extrémités. Cela permet de réduire la complexité du protocole SCOUT, la taille des messages échangés et le temps de traitement tout en évitant d'effectuer plusieurs fois la phase d'authentification.

### 5.2.3. Algorithme du protocole SCOUT

Le protocole SCOUT utilise 4 termes différents pour désigner les nœuds du réseau en fonction de leur rôle, un nœud pouvant avoir plusieurs rôles.
- Le nœud « **inspirateur** » désigne le nœud générant les paquets de données à sécuriser.
- Le nœud « **destinataire** » désigne le nœud de destination à qui sont adressées les données émises par l'inspirateur.
- Le nœud « **initiateur** » désigne le nœud initiant la découverte SCOUT.
- Le nœud « **répondeur** » désigne le nœud répondant positivement à cette découverte.

Ainsi, l'inspirateur transmet des données non sécurisées au destinataire, et l'initiateur les sécurise jusqu'au répondeur. La table 5.2 précise quelle fonction est associée à chaque nœud dans les 4 modes de fonctionnement de SCOUT. La table 5.3 associe à chaque rôle les nœuds du réseau pouvant assurer le rôle.

Dans ce qui suit, nous parlerons de conversation pour désigner tous les paquets échangés entre deux nœuds du réseau. Une conversation est caractérisée par le couple (adresse nœud 1, adresse nœud 2) et est indépendant du contenu transporté ou même du sens des échanges (nœud 1=source et nœud 2=destination, ou le contraire en sens inverse).

| Rôle | Nœud de la topologie figure 5.2 |
|---|---|
| **Inspirateur** | H1 (inconditionnellement) |
| **Initiateur** | H1 s'il supporte SCOUT,<br>R1 si R1 supporte SCOUT et H1 ne supporte pas SCOUT |
| **Répondeur** | H2 s'il supporte SCOUT,<br>R2 si R2 supporte SCOUT et H2 ne supporte pas SCOUT |
| **Destinataire** | H2 (inconditionnellement) |

**Table 5.3.:** Nœuds pouvant assurer les rôles d'Inspirateur, Initiateur, Répondeur et Destinataire

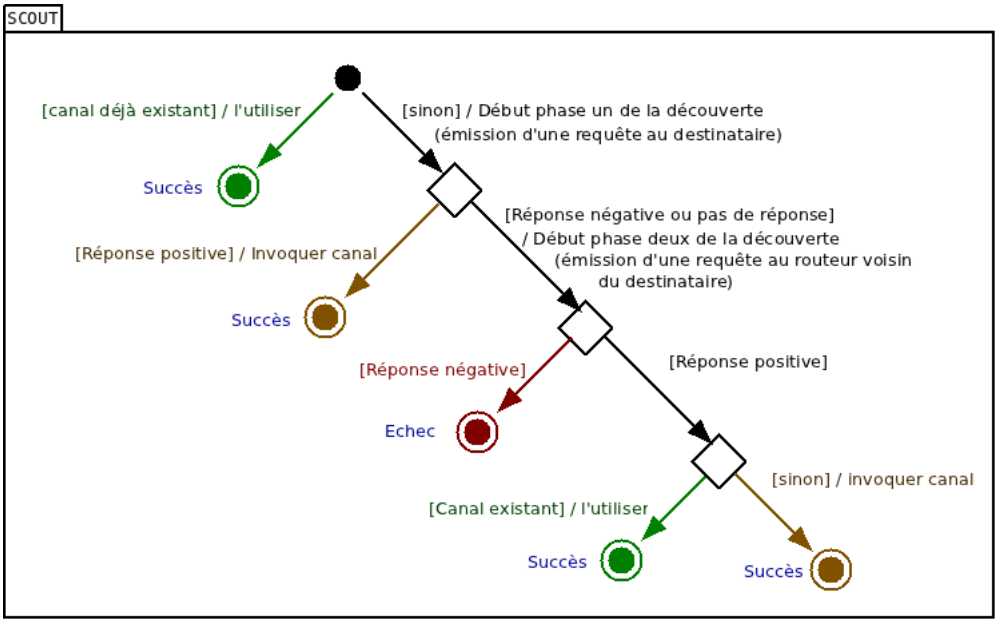



**Figure 5.3.:** Algorithme de SCOUT

#### 5.2.3.1. Première phase de découverte de SCOUT : hôte d'extrémité

L'algorithme de SCOUT peut être résumé de la manière suivante, présentée dans la figure 5.3. Le premier paquet d'une conversation est émis par l'inspirateur, ce qui déclenche la découverte SCOUT. A l'issue de la découverte, si un canal sécurisé a été établi, les autres paquets de cette conversation sont sécurisés au travers de ce canal, sinon ils sont rejetés (afin de ne pas transmettre de données non sécurisées).

Lorsque le premier paquet est émis par l'inspirateur vers le destinataire, le dæmon SCOUT de l'initiateur l'intercepte, le met en attente et émet une requête au destinataire. C'est la première phase de la découverte.

Si le destinataire supporte le protocole SCOUT, alors il répond positivement à la requête et a le double rôle de répondeur/destinataire (c'est un des deux modes Host-Host ou Router-Host). La réponse contient la liste des protocoles d'établissement

supportés, ce qui permet l'établissement d'un canal sécurisé entre l'initiateur et le répondeur/destinataire.

Si le destinataire ne prend pas en charge le protocole SCOUT, alors il répond de manière négative avec un paquet d'erreur, ou ne répond pas. Dès la réception d'un paquet d'erreur, l'initiateur passe directement à la seconde phase de découverte. Dans le cas où l'initiateur ne reçoit aucune réponse, il émet à nouveau plusieurs fois la requête (arbitrairement 3 fois) avant de considérer que c'est un échec et de passer à la seconde phase de découverte.

#### 5.2.3.2. Seconde phase de découverte de SCOUT : routeur au voisinage de l'hôte d'extrémité

Dans la seconde phase de découverte, l'initiateur contacte le routeur au voisinage du destinataire. Comme précédemment, si le routeur supporte SCOUT, alors il prend le rôle de répondeur, répond positivement à la requête et un canal sécurisé est établi entre l'initiateur et le répondeur. Dans le cas où le routeur ne supporte pas SCOUT il ignore la requête ou alors il retourne une erreur. Après plusieurs retransmissions, l'initiateur considère que la seconde phase de découverte est un échec, il ne peut donc pas sécuriser les données et ajoute alors une règle pour rejeter les données ultérieures de la conversation concernée.

Le protocole SCOUT utilise les adresses source et destination des paquets échangés pour déterminer les données nécessitant de la sécurité. L'administrateur peut ainsi configurer différentes interfaces réseaux avec différentes politiques de sécurité. En utilisant deux adresses IP sur une seule interface, il est possible de faire transiter, via la même interface, des données à sécuriser et des données sans besoin de sécurité, en liant par la configuration l'adresse à sécuriser au dæmon SCOUT. Le détail des requêtes des premières et secondes phases et du format des réponses est donné dans la prochaine section décrivant comment le protocole générique SCOUT est instancié avec IPv6 pour former le protocole SCOUT6.

### 5.2.4. SCOUT avec IPv6 : Le protocole SCOUT6

Le protocole SCOUT désigne les grandes lignes du fonctionnement permettant à des nœuds communicant de sécuriser les données : il définit comment un nœud initiateur contacte un nœud répondeur et découvre ses capacités de sécurisation pour ensuite déclencher l'établissement d'un canal sécurisé. Cette section expose la manière dont les spécifications du protocole SCOUT sont utilisées avec les particularités du protocole réseau Internet Protocol version 6 (IPv6). Cette réunion forme un nouveau protocole appelé SCOUT6. La partie générique de découverte est désignée par le nom « SCOUT », la partie d'adaptation à IPv6 par « SCOUT6 ».

Conformément à la RFC 6434 [Jankiewicz *et al.* 2011], les nœuds IPv6 devraient supporter IPsec. Le protocole d'établissement actuellement le plus employé pour

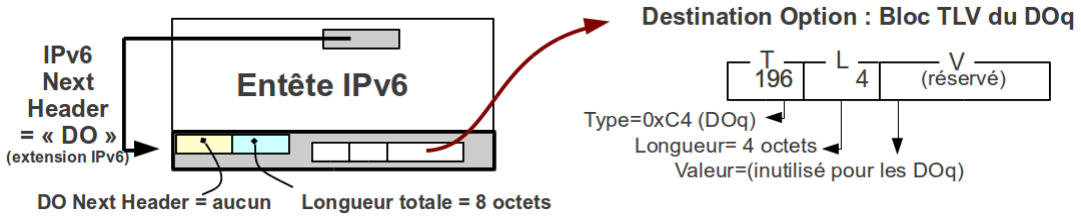



**Figure 5.4.:** Format d'un paquet IPv6 complété de l'entête Destination Option Query (DOq)

établir les canaux sécurisés avec IPsec est le protocole IKEv2, d'où le choix de notre implémentation de privilégier l'usage de ce protocole devant les autres protocoles potentiels tels que KINK et IKE. L'administrateur peut cependant configurer le dæmon SCOUT6 afin de sélectionner préférentiellement d'autres protocoles.

Le protocole IPv6 permet d'adjoindre des données supplémentaires aux paquets dans les « extensions headers ». Ces entêtes optionnels ne sont pas destinés aux applications utilisateurs et sont gérés directement par les services du système d'exploitation, afin d'adapter le comportement du système et notamment de sa pile réseau, en fonction des besoins exprimés dans les entêtes. La conception de SCOUT6 utilise deux entêtes d'extensions détaillés plus loin : le hop-by-hop option et le Destination Option. Ces extensions permettent d'étendre IPv6 avec SCOUT6, tout en maintenant une cohabitation avec les nœuds réseaux non compatibles avec SCOUT6.

Avec SCOUT6, nous avons besoin d'échanger 3 informations différentes : une pour la requête de la phase initiale (où l'initiateur demande au destinataire les protocoles d'établissement supportés), une pour la requête de la seconde phase (où l'initiateur demande ces informations au routeur voisin du destinataire) et une pour la réponse à ces deux requêtes (où le répondeur renvoie les protocoles supportés à l'initiateur).

#### 5.2.4.1. Destination Option Query (DOq)

Le paquet DOq permet à l'initiateur de demander au destinataire les protocoles d'établissement pris en charge. Il est formé d'un entête IPv6 suivi des données d'extension de la Destination Option, comme présenté figure 5.4. La spécification du protocole IPv6 [Jankiewicz *et al.* 2011] précise qu'un paquet de ce type doit être examiné par la destination, les équipements intermédiaires n'ayant pas besoin d'effectuer de traitement particulier pour ce paquet.

Les données d'extension sont au format Type-Length-Value où un octet code le type de paquet, un autre la taille du champ Valeur, suivi éventuellement de données dans le champ Valeur. L'octet de type oriente le traitement par défaut du système d'exploitation, comme expliqué dans la table 5.4, dans le cas où le paquet n'a pas été reconnu et intercepté par l'un des services du système.

Pour le DOq de SCOUT6, nous utilisons la valeur de type expérimentale 196 qui

| Intervalle pour les types du DO | Comportement par défaut du système d'exploitation |
|---|---|
| 0..63 | Ignorer l'extension du paquet |
| 64..127 | Détruire le paquet silencieusement |
| 128..191 | Renvoyer un paquet ICMPv6 Parameter Problem à l'expéditeur |
| 192..255 | Renvoyer un paquet ICMPv6 Parameter Problem à l'expéditeur si et seulement si le destinataire n'est pas un groupe multicast |

**Table 5.4.:** Traitement par défaut des « Destination Option » selon le champ Type

n'est actuellement assignée à aucun service par l'IANA [IANA 2012], l'organisme de normalisation d'Internet gérant notamment les assignations de constantes pour IPv6. Cette valeur influe sur le traitement du paquet par le système l'ayant reçu : si un dæmon SCOUT6 s'exécute sur la machine, il intercepte le paquet et envoie un DOr comme expliqué plus bas. Si aucun dæmon SCOUT6 n'intercepte le paquet, alors le système d'exploitation envoie un paquet ICMPv6 Parameter Problem à l'initiateur qui sait ainsi que la requête a aboutie négativement : la première phase de découverte est un échec car le destinataire ne supporte pas SCOUT6.

#### 5.2.4.2. Router Alert Query (RAq)

Dans le cas où l'initiateur n'a aucune réponse aux requêtes DOq envoyées ou si la réponse est un ICMP Parameter Problem, alors la première phase est un échec et l'initiateur commence la seconde phase de découverte : il tente de contacter le routeur voisin du destinataire.

Pour cela, il émet un paquet de type Router Alert [Partridge & Jackson 1999] contenant le sous-type « SCOUT6 ». Ce paquet Router Alert est un paquet IPv6 avec l'extension hop-by-hop, ce qui indique à chaque routeur intermédiaire d'analyser les données contenues dans l'extension. Nous utilisons le numéro de sous-type 42 qui n'est actuellement assigné à aucun autre usage par l'IANA. L'initiateur envoie le paquet RAq au destinataire. Les routeurs sur le chemin entre l'initiateur et le destinataire analysent le paquet ; celui-ci ne leur étant pas destiné, le paquet est transmis sans modification par chaque routeur au routeur suivant, qu'il y ait ou non un dæmon SCOUT6 actif sur ces routeurs.

Le cas du dernier routeur (le voisin du destinataire) est différent. Si ce routeur exécute un dæmon SCOUT6, alors il intercepte la requête RAq et y répond avec un DOr comme détaillé plus loin. S'il n'exécute aucun dæmon SCOUT6, le paquet est transmis par le routeur non compatible à l'hôte destinataire qui l'ignore : le paquet RAq ne contient aucune donnée autre que l'extension Router Alert.

L'emploi d'une extension nécessitant d'être analysée par chaque routeur intermédiaire peut sembler lourde. Mais en pratique, la RFC 6434 [Jankiewicz *et al.* 2011]

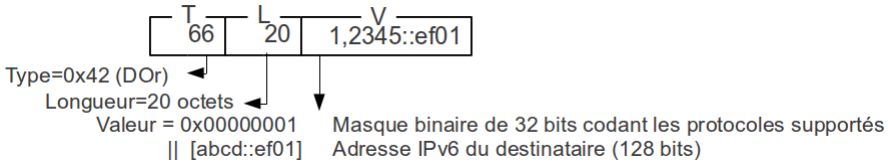



**Figure 5.5.:** Format de l'entête Destination Option Response (DOr)

précise qu'un routeur ne gérant pas spécifiquement les protocoles concernés par les RA n'a pas besoin d'analyser cette extension, les paquets sont alors transmis sans traitement ni délai supplémentaire. De plus, certains fournisseurs d'accès à l'Internet éliminent dès l'entrée dans leurs réseaux les paquets avec l'extension RA. Pour pallier ce problème, nous proposons une solution dans la section 5.5.

#### 5.2.4.3. Destination Option Response (DOr)

Quand le dæmon SCOUT6 reçoit d'un initiateur une requête DOq ou RAq, il répond avec un paquet appelé « Destination Option Response ». Ce paquet IPv6 destiné à l'initiateur contient l'adresse du répondeur et une extension Destination Option avec le type 66. Cette valeur, non assignée par l'IANA, permet à un système n'exécutant pas SCOUT de rejeter silencieusement ce paquet DOr, contrairement au DOq où un ICMP Parameter Problem est renvoyé.

Le DOr, dont le format est représenté figure 5.5, contient dans son champ Valeur deux informations. La première est un masque binaire de 32 bits où chaque bit code le support d'un protocole d'établissement. Un bit de poids faible (Least Significant Bit, LSB) à 1 signifie que IKE est supporté, le bit suivant sert pour IKEv2, le suivant pour KINK et ainsi de suite. Le bit ayant le poids le plus fort (Most Significant Bit, MSB) est un bit utilisé pour des usages expérimentaux. Tous les autres bits sont actuellement inutilisés et réservés pour le support futur d'autres protocoles d'établissement.

La deuxième information contenue dans le champ Valeur du DOr émis par le répondeur est constituée des 16 octets de l'adresse IPv6 du destinataire, c'est-à-dire l'adresse de destination du paquet DOq ou RAq. Cette information permet à l'initiateur d'identifier quelle conversation est associée au DOr. L'adresse IPv6 source du paquet DOr est l'adresse du répondeur (H2 ou R2 dans la topologie figure 5.2).

L'adresse IPv6 contenue dans le champ DOr est l'adresse du destinataire (H2 dans la topologie). Cette adresse est soit la même adresse quand le répondeur est aussi le destinataire (H2 en modes Host-Host et Router-Host), soit une adresse différente quand le répondeur est le routeur voisin du destinataire et non le destinataire lui-même (H2 sécurisé par R2 en modes Host-Router et Router-Router). Cela permet ainsi à l'initiateur de savoir si il communique avec le destinataire ou le routeur voisin du destinataire.

A la réception du DOr, si l'initiateur dispose déjà d'un canal sécurisé vers le répondeur, alors l'initiateur peut réutiliser ce canal existant pour économiser des ressources : il n'invoque pas la création d'un nouveau canal mais ajoute uniquement une nouvelle règle de routage.

## 5.2.5. Mise en œuvre du dæmon SCOUT6

Le protocole SCOUT6 présenté ici est complété d'une implémentation gratuite et Open-Source (licence double LGPL 2 et CeCill-B). La documentation associée est présente dans l'annexe B.

Cette implémentation a permis de valider la mise en œuvre du protocole dans une configuration réseau réaliste ; les résultats de l'évaluation de performance sont disponibles plus loin dans cet article. Elle est composée de deux parties, l'une étant le cœur de l'implémentation du protocole SCOUT6 écrit en langage C, l'autre étant le code, constitué de scripts bash, nécessaire à l'intégration du cœur avec le système d'exploitation Debian que nous utilisons pour la validation.

### 5.2.5.1. Accrochage du dæmon SCOUT6 au système d'exploitation

La figure 5.6 présente la manière dont le dæmon SCOUT6 interagit avec la pile réseau du système : il capture les paquets entrants et sortants le concernant et les traite en conséquence. Ainsi, en reprenant l'exemple de la topologie de la figure 5.2 présentant un réseau basique H1 – R1 – R2 – H2, dans le cas de H1 supportant SCOUT, où le nœud est inspirateur et initiateur, le dæmon sur H1 doit intercepter les paquets sortants issus des applications s'exécutant sur H1 afin de déclencher le processus de découverte SCOUT ; l'interception s'effectue au point de capture n°1 de la figure 5.6.

Lorsque H1 ne supporte pas SCOUT contrairement à R1, le dæmon SCOUT sur R1 a le rôle d'initiateur non-inspirateur et doit intercepter les paquets de H1 en transit sur le routeur R1. Dans tous les cas, l'interception n'est effectuée que s'il n'existe pas déjà de canal sécurisé pour le paquet.

Lorsqu'H2 supporte SCOUT, le dæmon sur H2 a les rôles de répondeur et de destinataire, il doit donc réceptionner les requêtes entrantes et y répondre de manière adéquate. Lorsqu'H2 ne supporte pas SCOUT mais que R2 supporte SCOUT, le dæmon sur R2 a le rôle de répondeur non-destinataire, il doit alors intercepter et répondre aux requêtes émises durant la seconde phase de découverte vers les destinataires à sécuriser.

La figure 5.7 résume les points d'accrochage en fonction du type de système capturant le premier paquet à sécuriser, hôte d'extrémité ou routeur au voisinage.

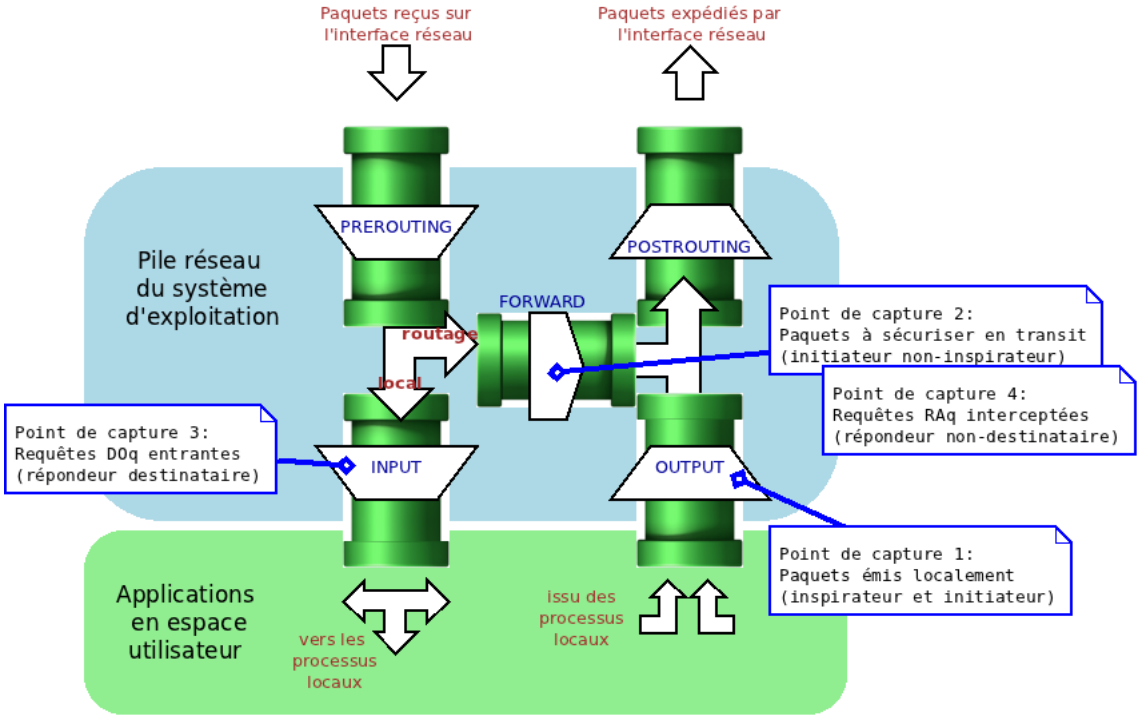



**Figure 5.6.:** Points d'accrochage du dæmon SCOUT6 sur le système d'exploitation

#### 5.2.5.2. Algorithme central du dæmon SCOUT6

La figure 5.8 présente l'algorithme que met en œuvre le dæmon SCOUT6. Elle est une conséquence de l'algorithme de SCOUT présenté figure 5.3. Après avoir extrait la classe du paquet capturé, elle le traite en conséquence : émission d'un DOr en réponse aux DOq et RAq, émission d'un DOq à la capture de données non sécurisées ou encore traitement du DOr et invocation éventuelle du canal sécurisé.

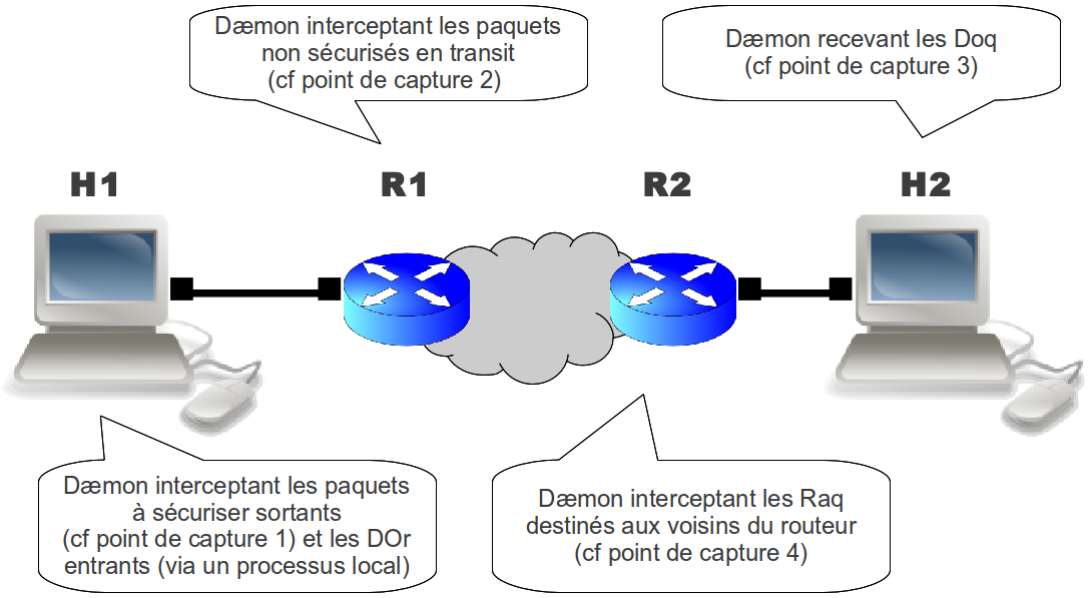



**Figure 5.7.:** Liaison des points d'accrochage et des systèmes du réseau

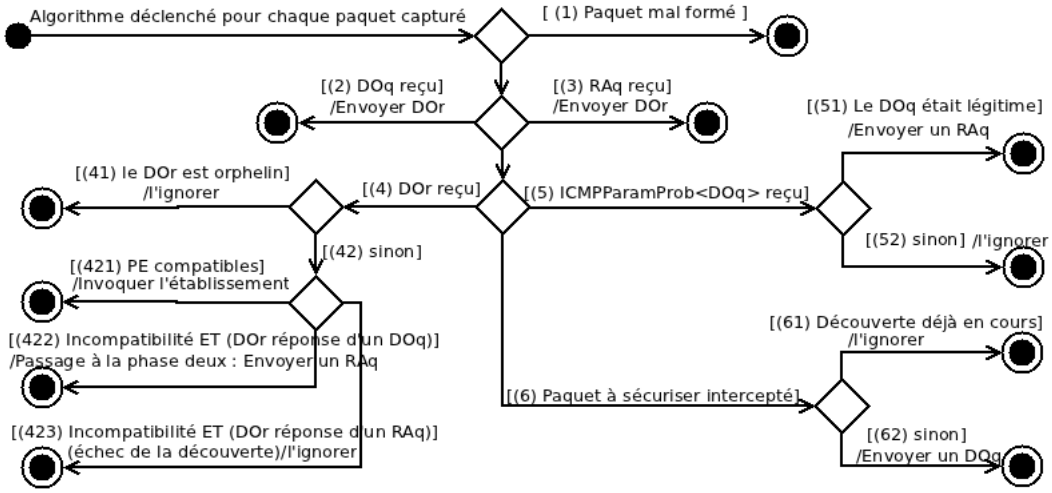



**Figure 5.8.:** Algorigramme du dæmon SCOUT6

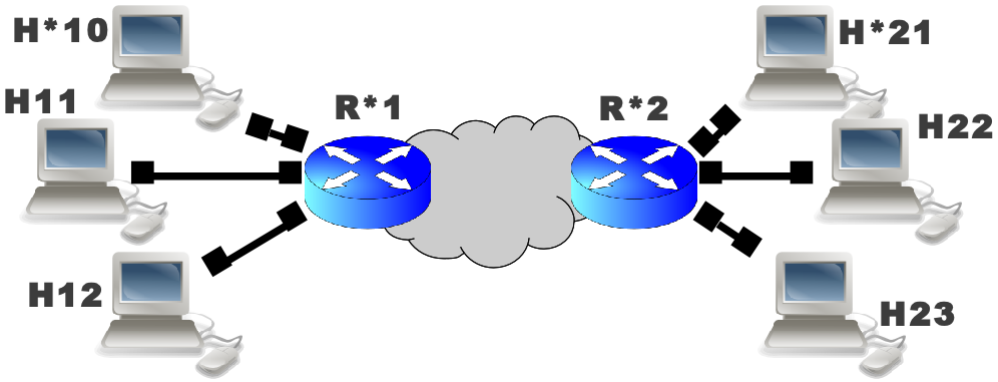



**Figure 5.9.:** Topologie de démonstration des modes de SCOUT

## 5.2.6. Exemples d'application de SCOUT6

Dans les sections suivantes, nous présentons les séquences des messages de découverte pour deux modes de SCOUT : le **mode Host-Host** puis le **mode Router-Router**. Afin d'illustrer ces deux exemples, nous utilisons la topologie présentée figure 5.9. Dans cette topologie, les systèmes exécutant un dæmon SCOUT6 sont indiqués à l'aide d'une étoile « * » dans leur nom : H*10, R*1, R*2, H*21. Les autres systèmes sont compatibles avec IPv6 mais non avec le protocole SCOUT6.

### 5.2.6.1. Le mode SCOUT Host-Host

La figure 5.10 illustre un scénario typique de découverte des capacités de sécurisation avec le protocole SCOUT6. Dans ce scénario, les deux hôtes d'extrémité H10 et H21 supportent tous deux SCOUT6 et un protocole d'établissement « PE ».

Quand le premier hôte inspirateur/initiateur H10 émet son premier paquet vers le destinataire/répondeur H21, le dæmon intercepte ce paquet et envoie une requête DOq à H21. En réponse, H21 renvoie à H10 un DOr contenant son adresse IPv6 et la liste des protocoles supportés. Le protocole « PE » est alors appelé par H10 pour établir un canal sécurisé de communication entre H10 et H21 et ajouter les règles de routage. Sur l'hôte H21, l'établissement du canal sécurisé entraîne aussi l'ajout de règles de routage duales. Ensuite, tout paquet transitant entre H10 et H21 est sécurisé au travers du canal.

### 5.2.6.2. Le mode SCOUT Router-Router

Dans le scénario Host-Host précédent, les routeurs intermédiaires pouvaient ou non supporter le protocole SCOUT6, cela n'influait pas sur la découverte. Dans le scénario suivant présenté figure 5.11, les hôtes d'extrémité H11 et H22 ne supportent pas SCOUT6 contrairement à leurs routeurs voisins R1 et R2.

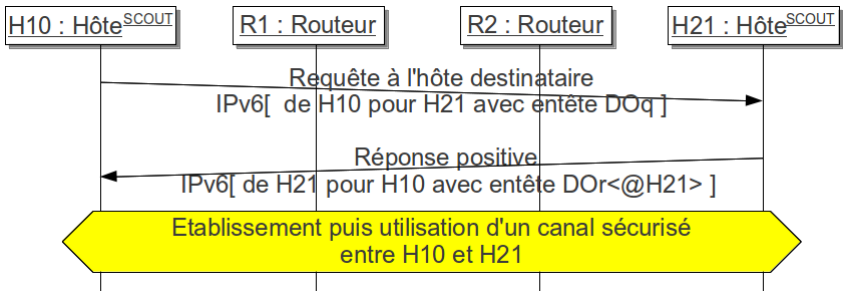



**Figure 5.10.:** Séquence de découverte aboutissant au mode Host-Host

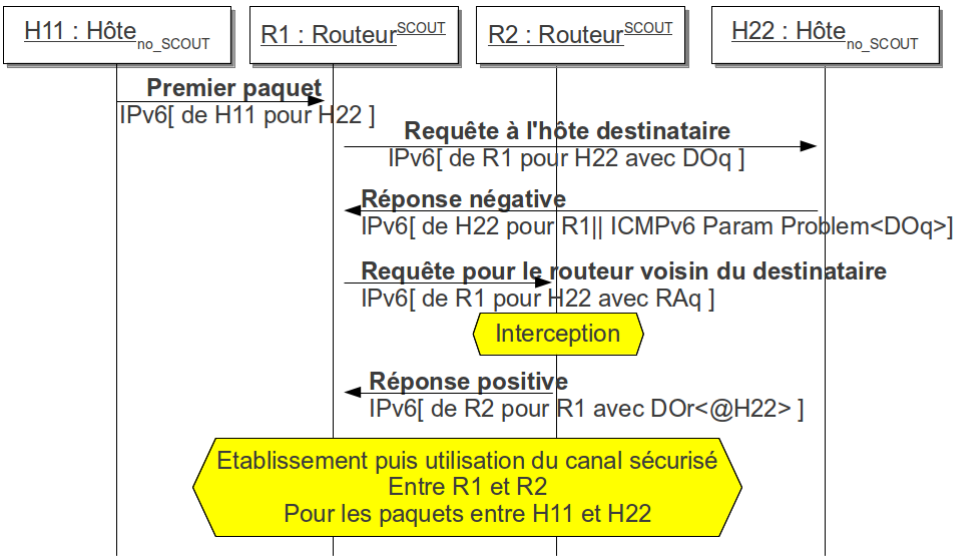



**Figure 5.11.:** Séquence de découverte aboutissant au mode Router-Router

Lors de l'émission par l'inspirateur H11 du premier paquet non sécurisé vers le destinataire H22, le paquet est intercepté par l'initiateur R1 qui lance la première phase de découverte : une requête DOq est envoyée à H22. Ne supportant pas SCOUT6, H22 retourne à R1 un paquet ICMP Parameter Problem, ainsi l'initiateur R1 sait que la première phase de découverte est un échec.

Il passe alors à la deuxième phase de découverte et envoie un RAq à H22. Les routeurs intermédiaires transmettent le RAq sans modification ni interception : ils ne savent pas gérer les paquets SCOUT6 ou ne sont pas concernés (car pas voisins de H22). R2 intercepte ce RAq et accepte de prendre en charge la sécurisation en renvoyant une réponse DOr à R1. L'initiateur R1 établit alors un canal sécurisé avec le répondeur R2. Les paquets échangés ensuite entre H11 et H22 transitent de manière sécurisée sur ce canal entre R1 et R2.

Ce canal sécurisé peut servir pour d'autres communications transitant entre R1 et R2 : le multiplexage de nombreuses communications au sein du même canal évite l'augmentation incontrôlée du nombre de canaux dans le réseau.

Les modes SCOUT Router-Host et SCOUT Host-Router sont des modes intermédiaires des modes SCOUT Host-Host et SCOUT Router-Router, en conséquence nous ne détaillerons pas les diagrammes de séquence de ces deux modes.

## 5.3. Évaluation des performances de SCOUT6

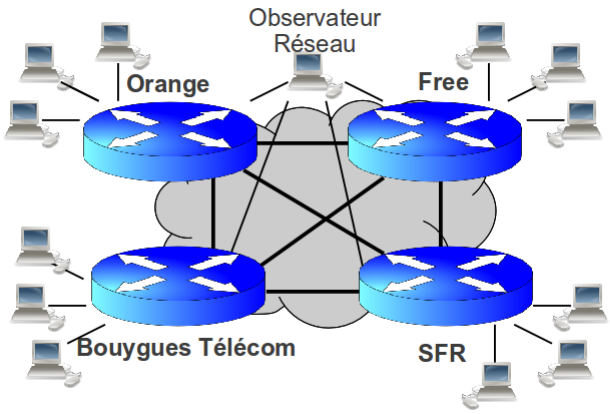
### 5.3.1. Evaluation de l'impact de SCOUT6 en terme de capacité et de délai réseau

Afin de découvrir les différentes possibilités de sécurisation des nœuds, le protocole SCOUT a besoin d'échanger les messages sur le réseau non sécurisé. L'implémentation SCOUT6 est basée sur les DOq, RAq et DOr présentés dans la section précédente. Ces paquets IPv6 ont une taille respectivement de 48, 48 et 64 octets. Dans le meilleur cas, pour une découverte, il y a un paquet DOq et un paquet DOr, soit 112 octets, tandis que dans le pire cas, il y a 3 paquets DOq perdus, 3 RAq et 3 DOr perdus, ce qui représente un total de 480 octets pour les 9 paquets.

Afin d'évaluer les performances de notre implémentation de SCOUT6, nous avons élaboré une topologie d'un réseau IPv6 avec 4 groupes de nœuds représentant 4 fournisseurs d'accès Internet (FAI) français : Orange, Bouygues Télécom, SFR et Free. Chaque communication entre ces fournisseurs a été pénalisée d'un délai de transmission des messages, mesuré le 6 décembre 2011 à 11h lors d'une étape préalable à nos expérimentations à l'aide des commandes traceroute et ping. Pour cela nous sommons les délais entre l'observateur réseau et les domaines d'émission et de réception. Cette méthode de mesure active permet d'obtenir une estimation réaliste du délai présent entre les différents FAI au moment de la mesure. La figure 5.12 contient le tableau récapitulant les résultats de nos mesures de délai et de gigue.

La topologie expérimentale que nous avons utilisée pour valider l'implémentation est aussi représentée dans cette figure 5.12. Chaque groupe a été affecté de 256 clients IPv6. Le réseau expérimental a été émulé à l'aide de VirtualBox [Innotek 2011] et de 4 machines virtuelles mono-core dotées de 256 Mo de RAM, d'un processeur cadencé à 2GHz et de cartes Ethernet virtuelles liées entre elles. Le système d'exploitation Linux Debian a été complété du dæmon SCOUT6, lié au système à l'aide du module noyau netfilter et de commandes iptables ([Andreasson & Blanc 2008]). L'annexe B contient la documentation de notre dæmon, avec des explications pour faciliter sa mise en œuvre.

De plus, le module Traffic Control Network Emulator (netem [Foundation 2011, Thomas Graf 2011]) a été utilisé pour ajouter les délais et la gigue de transmission. Enfin, afin de simuler du trafic réseau et de déclencher la découverte SCOUT, chaque machine virtuelle a exécuté un script générant toutes les 20 secondes un ping d'un




**Délais moyens et gigue (en ms)**

| | *Orange* | *Free* | *SFR* |
|---|---|---|---|
| *BouygT* | 41,1 / 1,7 | 37,6 / 0,5 | 39 / 0,4 |
| *Orange* | | 41,3 /1,6 | 42,7 / 1,8 |
| *Free* | | | 39,2 / 0,6 |

**Figure 5.12.:** Réseau avec 4 FAI

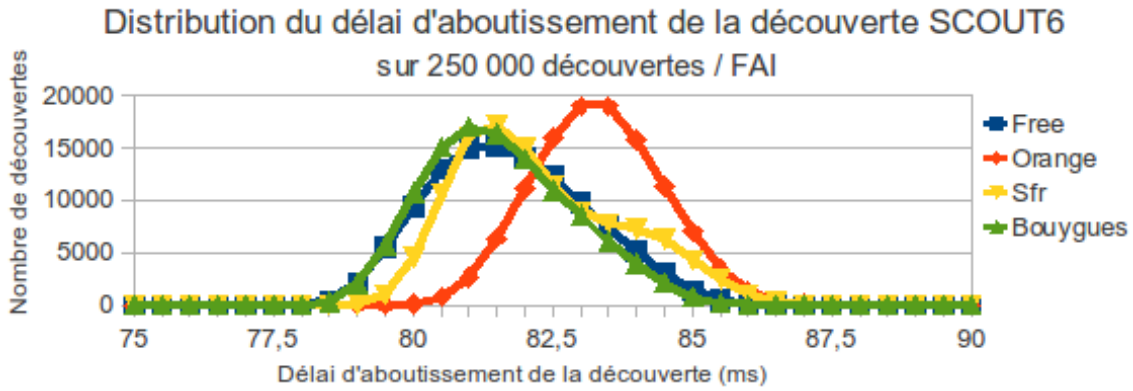



**Figure 5.13.:** Distribution des délais mesurés pour la découverte SCOUT6

client d'un quelconque FAI vers un client d'un autre FAI. La table 5.5 résume les délais attendus par la commande ping sur le réseau de la figure 5.12.

Chaque ping émet un paquet ICMP Echo Request qui est intercepté par le dæmon SCOUT6 local, ce qui déclenche la phase de découverte et aboutit à la création d'un canal sécurisé entre les nœuds. C'est le mode **SCOUT6 Host-Host** qui est utilisé dans cette phase d'évaluation de performance. Le dæmon a été instrumenté afin de mesurer la durée nécessaire pour chaque découverte.

Le graphique de la figure 5.13 montre la distribution obtenue avec les résultats de nos mesures de délais sur le dæmon SCOUT6. Le temps de traitement des paquets SCOUT6 est inférieur à quelques millisecondes et donc négligeable lorsqu'il est comparé aux délais d'acheminement des paquets entre les différents FAI, ces délais étant en moyenne autour de 40 ms par transmission de paquet. Nous pouvons donc en conclure que l'implémentation SCOUT6 et le service de découverte automatique qu'il offre ne pénalise pas les performances du réseau de test.

Enfin, au cours de tests de résistance (stress-tests) additionnels, nous avons mesuré le temps de traitement du dæmon pour 1 000 000 de paquets SCOUT6 consécutifs,

| Mode | Délai attendu sur notre réseau |
|---|---|
| Ping non sécurisé | **80 ms** = 40 ms x (1 request+1 reply) |
| Avec tunnel VPN configuré manuellement | **Quelques minutes** de configuration manuelle + 240 ms (dont 4 paquets d'initialisation IKEv2) |
| SCOUT6 en mode Host-Host ou en mode Router-Host | **320 ms** (dont les 80 ms de découverte SCOUT6 présentés figure 5.13) |
| SCOUT6 en mode Host-Router ou en mode Router-Router (sans perte de paquet IPv6) | **400 ms** : >DOq, <ICMPpp, >RAq, <DOr, >IKE_SA_INITi, <INITr, >AUTHi, <AUTHr, >ESP{echo request}, <ESP{echo reply} |

**Table 5.5.:** Prévision des délais selon la sécurisation

ce qui nous a permis d'établir que le temps moyen de traitement d'un paquet est de l'ordre de 500 nanosecondes. La différence des ordres de grandeur entre l'écart-type des distributions figure 5.13 (quelques millisecondes) et le temps de traitement d'un paquet SCOUT6 s'explique par l'imprécision de l'horloge utilisée pour établir les distributions figure 5.13 : la fonction de temps utilisée alors retourne une durée avec une précision de l'ordre de quelques millisecondes. Une implémentation sur une plateforme temps-réel et avec des techniques de métrologie plus perfectionnées nous permettrait de mesurer plus précisément la durée de traitement en environnement réel.

Nos expérimentations n'ont pas révélées de surcharge supplémentaire du routeur mesurable au niveau du CPU engendrées par le traitement des différents paquets SCOUT6 générés lors de la négociation des canaux sécurisés.

## 5.3.2. Complexité des opérations avec et sans SCOUT

### 5.3.2.1. Complexité dans un réseau centralisé

Afin d'évaluer de manière qualitative les bénéfices d'utilisation de SCOUT6, la table 5.6 résume les coûts (en nombre de lignes de configuration) pour un réseau centralisé sur le modèle du Client/Serveur. Les coûts sont évalués lors de l'ajout et de la suppression d'un client et d'un serveur. On suppose ici que le client est l'initiateur du canal sécurisé et qu'il initie un canal par serveur contacté. N désigne le nombre de clients, potentiellement élevé.

### 5.3.2.2. Complexité dans un réseau décentralisé

Dans un réseau pair-à-pair avec n nœuds interconnectés, chaque nœud peut communiquer avec (n-1) autres nœuds. Dans le pire cas, cela représente un total de n(n-1)/2 canaux bidirectionnels sécurisés à configurer puis établir. Sans le protocole

| | Sans le protocole SCOUT6 | Avec le protocole SCOUT6 |
|---|---|---|
| Ajouter un client (N+1) | Configurer ce (N+1)ème client et configurer le serveur **O(2)** | Installer SCOUT6 sur le client **O(1)** |
| Supprimer le client N ou modifier les informations de sécurité | Reconfigurer les deux extrémités sur le client N et le serveur **O(2)** | Reconfigurer le client N et éventuellement le serveur (pour accepter les nouveaux certificats...) **O(2)** |
| Ajouter un nouveau serveur | Sur le serveur ajouté, configurer un canal par client potentiel et sur chaque client, configurer le canal **O(2N)** | Installer SCOUT6 sur le serveur **O(1)** |
| Enlever un serveur | Reconfigurer chaque client **O(N)** | Rien à faire **O(0)** |

**Table 5.6.:** Evaluation des coûts dans le modèle client/serveur

SCOUT, chaque canal devant être configuré aux deux extrémités, cela représente un total de ($n^2$-n) canaux à sécuriser.

Avec SCOUT, l'administrateur doit installer et configurer le dæmon sur chaque nœud, sans préciser à chaque fois les (n-1) extrémités de canaux potentiels. Ainsi, la configuration doit être effectuée seulement n fois et non plus ($n^2$-n) fois.

De plus, avec SCOUT, seuls les canaux réellement utilisés sont établis, au moment du premier échange : si deux nœuds ne communiquent jamais ensemble, aucun canal n'est créé, ce qui préserve les ressources réseau et de calcul.

## 5.4. Sécurité du protocole SCOUT6

Le protocole SCOUT a été élaboré pour rechercher les possibilités de sécurisation des nœuds du réseau et établir, quand c'est possible, un canal sécurisé entre les nœuds. La sécurisation des données est assurée par un protocole d'établissement (PE), mais qu'en est-il de la sécurité de notre protocole SCOUT ? L'objectif de cette section est de donner des éléments de réponse à cette interrogation.

### 5.4.1. Injection de paquets

L'injection de paquets valides dans un système par un autre système ne participant pas légitimement à la communication est une attaque courante appelée « injection de

| Type | Réception par un nœud supportant SCOUT6 | Réception par un nœud NE supportant PAS SCOUT6 |
|---|---|---|
| **DOq** | Un DOr est envoyé en réponse ✓ | Un ICMP Parameter Problem contenant le DOq est renvoyé ✓ |
| **RAq** | Un DOr est envoyé et contient l'adresse du routeur ✓ | Le RAq reçu est ignoré par le récepteur ✓ |
| **DOr** | Le DOr reçu est ignoré par le récepteur ✓ | Le DOr reçu est ignoré par le récepteur ✓ |

**Table 5.7.:** Injection de paquets contre SCOUT6

paquets » [Schoen 2007]. Dans le cas de SCOUT6, trois types de paquets peuvent être injectés : les DOq, les RAq et les DOr. Ces paquets peuvent être émis à destination de systèmes où un dæmon SCOUT6 est à l'écoute pour répondre aux requêtes ou à destination de systèmes configurés avec le protocole IPv6 mais sans support pour le protocole SCOUT6. La table 5.7 résume les 6 types d'injections et leurs impacts.

L'injection de paquets DOq ou RAq à un système supportant le protocole SCOUT6 conduit ce système à répondre avec un paquet DOr, donc à ajouter une légère surcharge au réseau (64 octets sont envoyés). Ce paquet contient comme information une liste de protocoles de sécurisation supportés par le destinataire, information qu'un attaquant obtient plus généralement par la technique de « scan de ports ». L'injection d'un RAq peut aussi permettre de connaître l'adresse du routeur voisin si ce dernier supporte SCOUT6.

Injecter un paquet DOr à un système avec un dæmon SCOUT6 n'a aucun effet : le système receveur n'ayant envoyé aucune requête (DOq ou RAq), il considère le paquet injecté comme orphelin (pas de tentative d'établissement d'un canal sécurisé pouvant justifier la réception du DOr) et l'ignore en conséquence.

Injecter un paquet DOq à un système sans dæmon SCOUT6 peut conduire le receveur soit à ignorer ce paquet qu'il ne sait pas traiter, soit à répondre par l'émission d'un paquet « ICMP Parameter Problem ». Ces deux comportements sont conformes au protocole SCOUT et sont utilisés pour déclencher la deuxième phase du protocole SCOUT, la tentative d'établissement avec le routeur voisin du destinataire.

Injecter un paquet RAq ou DOr à un système IPv6 sans dæmon SCOUT6 n'a aucun effet : conformément à la spécification du protocole IPv6 [Jankiewicz *et al.* 2011], les paquets avec ces options qui sont inconnues du destinataire doivent être silencieusement ignorés, sans émission de paquet indiquant une erreur, contrairement au cas précédent des DOq.

Il est important de noter qu'un système sans dæmon SCOUT6 recevant un paquet DOr peut être considéré comme probablement attaqué : le DOr normal n'est envoyé qu'en réponse à un DOq ou à un RAq, et ces paquets ne sont eux-mêmes émis que par des systèmes SCOUT6 !

### 5.4.2. Rejeu de paquets

Un attaquant pourrait écouter les paquets SCOUT6, en intercepter certains et les injecter à nouveau ultérieurement pour déstabiliser les systèmes destinataires. Mais la conception du protocole SCOUT6 met déjà en œuvre le rejeu de paquets pour compenser l'absence de garantie d'acheminement des paquets par le protocole IPv6 et donc le risque de perte. Les requêtes DOq et RAq sont ainsi émises jusqu'à trois fois (par défaut) ou jusqu'à la réception de la réponse DOr correspondante. Un système peut alors recevoir plusieurs DOq ou RAq, il émettra à chaque fois un paquet DOr.

Le paquet DOr contient les adresses IP de l'initiateur, du répondeur et du destinataire, adresses que le dæmon SCOUT6 exploite pour identifier en interne le flux à sécuriser et donc définir les paramètres utilisés pour configurer le PE et établir le canal sécurisé adéquat. Quand le paquet DOr est reçu, le dæmon SCOUT6 vérifie d'abord qu'il est bien la réponse à une tentative d'établissement en cours. Lors d'une tentative d'établissement en cours, le premier DOr reçu associé à cette tentative indique le succès de la tentative, entraîne l'invocation du PE et la tentative est alors terminée. Il n'y a ensuite plus de tentative «en cours» à associer aux DOr suivants qui arriveraient plus tard, donc ces derniers seraient orphelins et ignorés. Les DOr rejoués sont donc ignorés par le récepteur.

### 5.4.3. Paquets volés, altérés et attaque de l'homme du milieu

Un attaquant peut intercepter les messages SCOUT6 durant leur acheminement sur les réseaux. Il peut silencieusement supprimer les paquets, empêchant ainsi le dæmon SCOUT6 de négocier l'établissement d'un canal sécurisé.

Mais si un attaquant a la capacité de bloquer l'acheminement des paquets SCOUT6, alors il a aussi la possibilité de bloquer l'acheminement des paquets échangés par le protocole d'établissement (par exemple les paquets IKEv2) d'une manière identique. Protéger le protocole SCOUT6 contre cette attaque est donc inutile, l'attaquant se reporterait en aval de SCOUT6 et bloquerait alors le protocole d'établissement, contournant ainsi la protection de SCOUT6.

L'attaquant peut utiliser sa capacité d'interception pour essayer l'attaque de l'homme du milieu, en s'identifiant auprès de l'initiateur et du répondeur comme l'entité duale d'extrémité de la connexion à sécuriser et ainsi tenter d'établir deux canaux sécurisés, un avec l'initiateur et un avec le répondeur, pour pouvoir observer «en clair» les informations échangées de manière chiffrée via les deux canaux.

SCOUT6 n'est pas protégée contre cette attaque car son rôle est limité à l'identification des systèmes d'extrémité du canal à sécuriser. L'authentification est assurée par le protocole d'établissement. Intégrer en plus des mécanismes d'authentification à SCOUT6 aurait donc été redondant, complexe à mettre en œu-

vre et à configurer et n'aurait pas apporté de garantie supplémentaire en terme de sécurité.

L'authentification assurée par le protocole d'établissement invoqué par SCOUT6 garantit que l'attaque de l'homme du milieu contre SCOUT6 ne peut pas aboutir, l'attaquant serait démasqué par les deux extrémités légitimes (l'initiateur et le répondeur) à la phase d'authentification, faisant ainsi échouer l'établissement du canal sécurisé.

L'attaquant peut altérer le contenu du message DOr, c'est-à-dire l'adresse IPv6 du destinataire. Dans ce cas, le message DOr modifié ne contient plus l'information nécessaire pour que l'initiateur associe le paquet DOr à une tentative d'établissement en cours, donc le DOr est considéré comme orphelin par le nœud du réseau qui le reçoit et qui le rejette silencieusement en conséquence.

### 5.4.4. Falsification d'adresses IP

Un attaquant peut altérer ou créer des messages DOq et RAq en imposant une adresse source « volée » au paquet IPv6. Alors, le paquet DOr éventuel émis en réponse est adressé à un nœud pouvant ou non supporter le protocole SCOUT6. Dans le premier cas, le DOr est alors considéré par le dæmon SCOUT6 qui le reçoit comme orphelin, le nœud n'ayant aucune tentative d'établissement en cours à associer au DOr reçu. Dans le cas où l'adresse modifiée est celle d'un nœud IPv6 sans dæmon SCOUT6, le nœud ignore le DOr reçu, comme expliqué section 5.2.4.3.

Un attaquant peut aussi falsifier l'adresse de destination d'un message SCOUT6. Si le paquet est un DOq ou un RAq, alors le cas est similaire à une découverte légitime SCOUT6, donc le destinataire nouvellement ciblé renverra, selon les cas, un ICMP Parameter Problem, un DOr ou encore aucune réponse. Si le paquet émis est un DOr, alors il sera considéré comme orphelin par le destinataire.

### 5.4.5. Déni de Service (DoS)

Un attaquant pourrait envisager d'utiliser des paquets SCOUT6 pour surcharger le travail de certains nœuds du réseau. Les dæmons SCOUT6 peuvent servir de miroir : à la réception d'un DOq ou RAq, ils peuvent renvoyer un DOr. Cependant, les paquets SCOUT6 sont petits (48 ou 64 octets, selon le type de message, DOq, RAq ou DOr). Un attaquant peut facilement forger des paquets ICMPv6 bien plus gros ou de gros paquets avec des extensions IPv6 non standardisées pour « surcharger » un réseau, indépendamment de la présence ou de l'absence de dæmon SCOUT.

Un nœud peut être amené à accepter l'établissement d'un trop grand nombre de canaux sécurisés, nombre supérieur aux capacités du nœud. Ce n'est pas une vulnérabilité du protocole SCOUT mais plutôt une conséquence négative du passage à l'échelle rendu possible par SCOUT. Cette vulnérabilité peut être évitée par la

configuration du système en limitant le nombre maximal de canaux sécurisés simultanés. Si cette limitation n'est pas une solution acceptable, la vulnérabilité peut être réduite en utilisant des algorithmes efficaces pour gérer les canaux (par exemple avec un algorithme de recherche dichotomique plutôt qu'un parcours linéaire de table) et en supprimant les canaux inutilisés (par exemple avec l'algorithme de gestion de cache « Least Recently Used »).

### 5.4.6. Une attaque fonctionnelle contre SCOUT6 et sa conséquence

Un attaquant peut influencer le comportement de SCOUT6 en bloquant silencieusement les paquets DOq et en laissant passer les paquets RAq et DOr. Si l'hôte de destination et son routeur voisin sont tous deux compatibles SCOUT6, alors l'initiateur va tenter d'établir le canal sécurisé avec le routeur voisin du destinataire et non avec le nœud destinataire lui-même. Dans ce cas, le chemin entre le destinataire et son routeur voisin ne sera pas sécurisé par SCOUT6, malgré le support de SCOUT6 par le destinataire.

C'est pourquoi la première hypothèse d'utilisation de SCOUT6 de la section 5.2.2 est « le réseau local est sûr ». Cette attaque peut être empêchée si le dæmon SCOUT6 de l'initiateur est configuré pour refuser de fonctionner avec un répondeur de type routeur, donc pour ne pas envoyer de RAq et refuser les modes Host-Router et Router-Router. Mais cette solution réduit les possibilités de découverte et de sécurisation offertes par l'usage du protocole SCOUT6.

### 5.4.7. Conclusion sur la sécurité du protocole SCOUT6

Nous avons énuméré dans cette section différentes attaques contre notre protocole de découverte des capacités de sécurisation avec IPv6, le protocole SCOUT6. Les attaques par injection et par rejeu de paquets SCOUT6 sont inopérantes : le protocole SCOUT6 repose en effet sur des mécanismes similaires pour son fonctionnement, les paquets de l'attaquant sont alors gérés de la même manière que les paquets légitimes, sans conséquence néfaste pour la sûreté des systèmes. De la même manière, les paquets attaquant le dæmon SCOUT6 avec des adresses IP erronées ou falsifiées sont rejetés silencieusement.

Nous avons vu que le protocole SCOUT6 est vulnérable aux attaques de vol et d'altération de paquets. Cependant, le protocole d'établissement du canal sécurisé est lui aussi vulnérable à ces attaques. Ainsi, un attaquant capable de réaliser ce type d'attaque en amont durant la découverte SCOUT6 est capable de bloquer en aval l'établissement du canal, rendant inutile une protection en amont contre ces attaques, au niveau du protocole SCOUT6. De manière similaire, nous n'avons pas protégé le protocole SCOUT6 contre l'attaque de l'homme du milieu, afin d'éviter

la redondance avec les protections mises en place par le protocole d'établissement et de limiter la complexité du fonctionnement de notre protocole SCOUT6. La petite taille des paquets utilisés par notre protocole limite fortement l'impact de l'exploitation du dæmon SCOUT6 pour les attaques par déni de service.

Le protocole SCOUT6 est prévu, dans sa configuration par défaut, pour accepter la sécurisation des données entre des routeurs. Dans ce cas, les données ne sont pas protégées sur les liens des réseaux locaux liant les hôtes d'extrémités aux routeurs qui assurent la sécurisation, ces liens sont alors potentiellement vulnérables. Nous avons présenté deux pistes de solutions pour réduire cette limite de la sécurité apportée par notre protocole SCOUT6.

Pour conclure, le protocole SCOUT6 apporte globalement la sécurisation automatique des données échangées, tout en étant peu vulnérable aux attaques. Concernant les performances, nous avons identifié durant nos expérimentations un certain nombre d'améliorations potentielles qui sont l'objet de la section suivante.

## 5.5. Perspectives d'évolutions du protocole SCOUT

Le protocole SCOUT6 présenté dans les sections précédentes a été implémenté et validé avec succès sur des réseaux réels connectés à l'Internet. Cependant, certaines particularités de ce protocole peuvent être incompatibles avec des contraintes imposées par certains fournisseurs d'accès, acteurs incontournables pour les connexions longues distances à travers le réseau Internet. Cette section présente quelques uns de ces problèmes et deux solutions sont proposées pour y remédier.

### 5.5.1. Le problème des Router Alert rejetés par les fournisseurs d'accès

Certains fournisseurs d'accès Internet ajoutent des règles de filtrage afin de détruire dès l'entrée dans leur réseau tous les paquets porteurs de l'extension Router Alert (RA). Cela permet d'éviter de réduire les performances du réseau, les routeurs n'ayant plus à gérer « en profondeur » les paquets IP complétés de cette extension. Ce filtrage est aussi une protection contre les attaques qui exploiteraient l'option RA comme vecteur d'attaque. Mais ce filtrage des RA entraîne la destruction des paquets RAq utilisés pour la découverte de la seconde phase de SCOUT6.

Nous avions pensé initialement à créer un nouveau type de paquet SCOUT6 dérivé du DOq, mais l'optimisation développée dans la section suivante résout le problème d'une manière plus efficace.

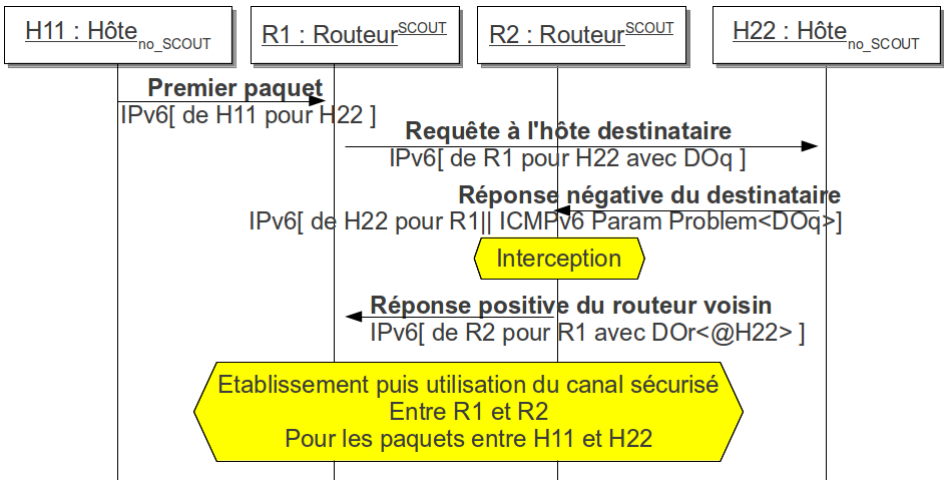



**Figure 5.14.:** Séquence optimisée pour SCOUT6

### 5.5.2. Accélération de la seconde phase par interception du résultat de la première phase

Dans le cas où l'hôte destinataire (H22) ne supporte pas SCOUT6 et où le routeur voisin de cet hôte (R2) supporte SCOUT6, la réception du DOq entraîne l'émission par H22 d'un paquet ICMPv6 Parameter Problem contenant ce DOq. Le déroulement classique de la découverte de SCOUT6 présenté plus haut en section 5.2.3 passe d'abord par la réception par l'initiateur de cet ICMP Param Problem<DOq>, du déclenchement de la deuxième phase et de l'émission d'un RAq vers le routeur qui répond alors avec un DOr.

La figure 5.14 montre comment raccourcir cette séquence. Le paquet ICMPv6 Parameter Problem<DOq> transite forcément par le routeur destinataire (R2). Celui-ci peut intercepter ce message, le traiter comme s'il avait reçu un RAq et donc renvoyer immédiatement un DOr adéquat. Cela permet de sauter les étapes de réception de la réponse négative de la première phase et d'émission de la requête de la deuxième phase. La figure 5.14 représente une découverte aboutissant au mode **SCOUT6 Router-Router** avec cette optimisation.

Les avantages de ce raccourci sont les suivants. Tout d'abord, il n'est plus nécessaire d'envoyer de RAq, ce qui résout le problème de filtrage par certains fournisseurs d'accès. Cela permet d'économiser l'acheminement de deux paquets (un retour pour la réponse négative puis un aller pour le RAq) sur le réseau.

De plus, le temps nécessaire à la découverte est réduit : dès le premier DOq transmis, l'initiateur est notifié soit par un DOr venant du nœud destinataire (H22) ou de son routeur (R2) soit par un ICMP Parameter Problem, ce dernier indiquant que SCOUT n'est pris en charge ni par H22 ni par R2 et donc l'échec de la découverte. Ainsi, un seul aller-retour est nécessaire pour tous les résultats possibles de la découverte.

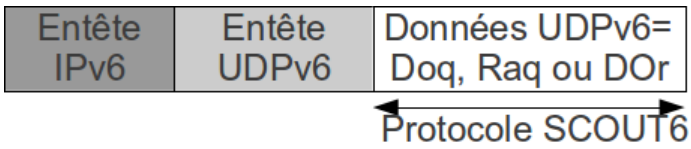



**Figure 5.15.:** Encapsulation des options pour SCOUT6 dans des datagrammes UDP

Cependant, des règles de filtrage spécifiques ou un système d'exploitation partiellement compatible avec la spécification IPv6 pourraient supprimer les paquets contenant les Destination Option (notamment les DOq) et ainsi ne pas renvoyer de paquet ICMP Parameter Problem. Dans ce cas de figure, cette optimisation est inapplicable.

### 5.5.3. Adaptation de SCOUT6 avec le protocole UDPv6

Afin de pallier les problèmes de filtrage des paquets contenant des extensions hop-by-hop et destination option, nous avons envisagé d'utiliser une encapsulation de l'option dans un datagramme UDP, comme présenté dans la figure 5.15.

L'avantage de cette solution est évidemment de garantir une « apparence normale » aux paquets IP et donc de permettre leur traitement avec les techniques couramment utilisées de filtrage des datagrammes UDP sur les routeurs, pare-feux et autres équipements réseaux. De plus, le protocole UDP étant très proche en IPv4 et IPv6, cette solution ouvre des perspectives d'application du protocole SCOUT avec le protocole IPv4. Cependant, cela pourrait poser des problèmes de mise en œuvre, notamment pour la traversée des Network Address Translation (NAT) qui s'avèrent peu compatibles avec IPsec [Kivinen *et al.* 2005].

## 5.6. Résumé du chapitre

Nous avons présenté dans ce chapitre le protocole SCOUT qui découvre automatiquement les possibilités de sécurisation sur un nœud d'un réseau. Le protocole SCOUT peut non seulement invoquer l'établissement d'un canal sécurisé entre deux hôtes mais aussi déclencher cet établissement avec les routeurs de proximité de ces hôtes si les hôtes ne disposent pas des capacités nécessaires à la sécurisation.

De plus, le protocole SCOUT réduit la charge d'administration de la sécurité des réseaux et facilite le passage à l'échelle des solutions de sécurité réseau. La concrétisation du protocole SCOUT avec IPv6 complète le protocole SCOUT générique avec un protocole concret nommé SCOUT6. Nos expérimentations indiquent une surcharge négligeable pour le réseau en terme de données échangées et de délais

additionnels et valident la mise en place dynamique et réactive d'un canal sécurisé de communication.

Cependant, la complexité des besoins aéronautiques dans ce domaine est trop faible pour justifier pour l'instant l'intégration de SCOUT au routeur SNG, mais la méthodologie de développement utilisée permet aisément l'ajout futur de modules complémentaires au routeur SNG tel qu'une classe de partition mettant en œuvre SCOUT6.

Le protocole SCOUT6 a été mis en œuvre sur un système basé sur le système d'exploitation Debian. L'annexe B contient la documentation liée au programme d'installation et de déploiement du dæmon SCOUT6, disponible à l'adresse http://www.recherche.enac.fr/~avaret/scout6.

# 6. Évaluation des performances du routeur SNG

Dans les chapitres précédents, nous avons vu les différentes phases d'élaboration du routeur SNG. Le chapitre 2 a présenté le contexte d'utilisation ciblé ainsi que les fonctionnalités attendues et les contraintes s'appliquant au routeur. Le chapitre 3 a présenté une méthodologie novatrice de développement logiciel, puis son application au routeur SNG a été présentée dans le chapitre 4.

Ce sixième et dernier chapitre résume la dernière phase de l'élaboration du routeur : l'évaluation et la validation des performances du routeur SNG. Dans ce chapitre, nous commencerons par présenter différentes topologies réseaux envisagées, puis nous continuerons par les différentes métriques permettant de quantifier les performances réseaux de notre routeur SNG et les outils utilisés pour les mesures. Enfin les résultats numériques seront présentés avec à chaque fois un rappel méthodologique et une interprétation des résultats.

Étant donné que la mise en œuvre du protocole IPv6 reprend l'ensemble des fonctionnalités mises en œuvre pour le protocole IPv4 et ajoute en plus d'autres fonctionnalités telles que la sécurité (confidentialité, intégrité et authentification), nous ne présenterons que les résultats d'évaluation pour ce protocole IPv6.

## 6.1. Cadre d'expérimentation

### 6.1.1. Topologies réseaux d'études

Avant d'effectuer des mesures, il est nécessaire de mettre en place un cadre d'expérimentation avec les différents systèmes, cadre devant permettre d'isoler les phénomènes observés pour mieux les mettre en évidence. Les objectifs sont de quantifier les performances et capacités de traitement du routeur SNG et de quantifier les performances d'une architecture réseau contenant un ou plusieurs routeurs SNG.

Pour cela nous nous sommes posés la question de la topologie à installer et configurer. La légende des topologies présentées ici est indiquée dans la figure 6.1a.

La première topologie présentée sur la figure 6.1b est constituée d'un hôte nommé H1, connecté physiquement par un lien Ethernet à un hôte R1. Cette topologie a

été utilisée dans un premier temps pour vérifier et valider le fonctionnement correct de la mise en œuvre de certaines fonctionnalités, elle nous a ainsi permis de valider la réponse aux ping (service ICMP ECHO) et le filtrage, ainsi que de valider partiellement les fonctionnalités de routage et le protocole NDP du routeur SNG.

Cependant, la topologie de la figure 6.1c est plus adaptée pour valider le routage. Constituée de trois hôtes et trois sous-réseaux logiques, elle ne contient cependant qu'un seul routeur SNG et ne permet donc pas d'évaluer les performances de la sécurisation entre routeurs SNG.

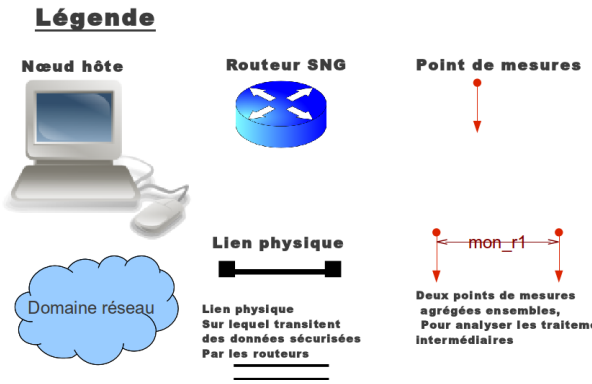
C'est pourquoi nous lui avons préféré la topologie de la figure 6.1d, constituée de deux hôtes d'extrémités H1 et H2 servant respectivement de source et de puits ou encore de source et de miroir (renvoyant des réponses adaptées aux requêtes de la source) et comprenant un unique chemin traversant deux routeurs SNG. Ces routeurs voisins peuvent établir, selon la configuration, un canal sécurisé entre eux pour valider les fonctionnalités de sécurité. La topologie est ainsi symétrique :
H1 – R1 ==«symétrie»== R2 – H2.

Cette topologie permet aussi de valider le routage : les routeurs routent les paquets entre deux réseaux connectés à deux de leurs interfaces réseau. De plus, la configuration mutualise le lien entre les deux routeurs R1 et R2 pour les connecter simultanément à quatre réseaux logiques, chacun étant associé à un niveau de sécurisation différent, comme présenté figure 6.1e.

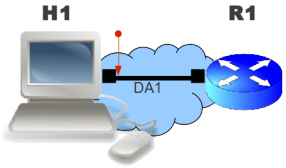
Une quatrième et dernière topologie a été envisagée et est présentée figure 6.1f, mais sa lourdeur de mise en place (elle nécessite d'utiliser encore plus de matériel) n'est pas justifiée, car la topologie 6.1e permet déjà d'évaluer ce que nous souhaitions quantifier et qui est détaillé dans les sections suivantes : délais, débits, pertes et disponibilités, selon les types et nombres de flux, la sécurité, la charge simultanée sur le réseau ; cette dernière topologie 6.1f n'a donc pas été utilisée.

THALES AVIONICS nous a présenté deux topologies avioniques réalistes, illustrées dans la figure 6.2. Ces topologies exploitent les fonctionnalités de routage et de filtrage de nos routeurs SNG entre les différents réseaux avioniques, des tunnels sécurisés permettent d'interconnecter certains réseaux disjoints en traversant d'autres réseaux vulnérables. Ces topologies utilisent donc les mêmes fonctionnalités que celles déjà validées et évaluées à l'aide de la topologie 6.1d, nécessitant cependant plus de matériel et de temps pour ces topologies de la figure 6.2.

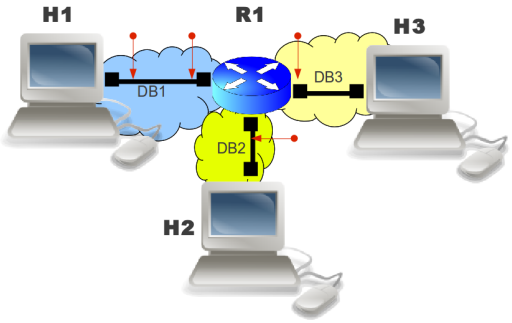
Nous avons donc privilégié la topologie 6.1d pour tester et stresser les routeurs SNG. Ceux-ci pourront ensuite être intégrés aux autres topologies telles que celles de la figure 6.2, en reconfigurant les tables de routage et de filtrage et les informations de sécurisation (clefs de chiffrement, certificats...).



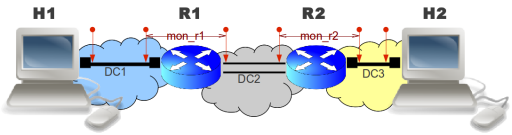

**(a)** Légende utilisée pour les topologies réseau



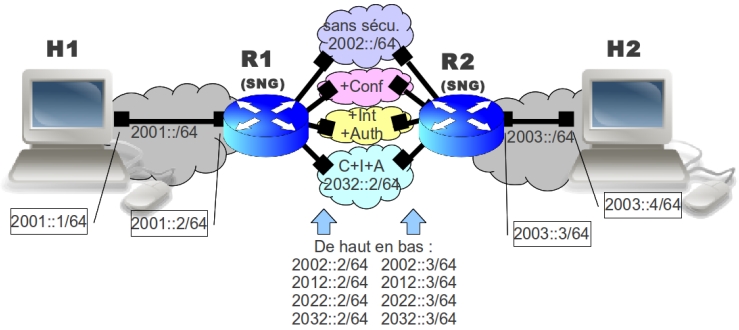

**(b)** Topologie "A" à un hôte et un routeur



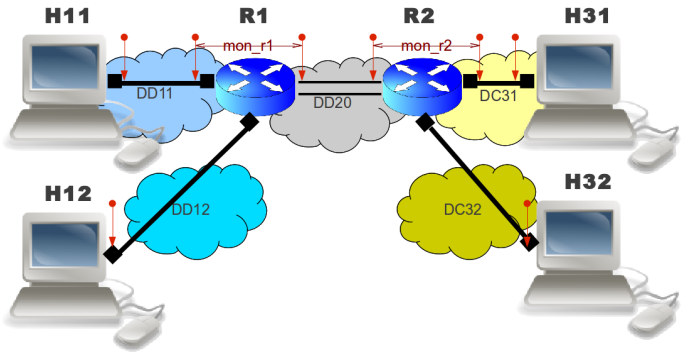

**(c)** Topologie "B" à trois hôtes et un routeur




**(d)** Topologie "C" à deux hôtes et deux routeurs

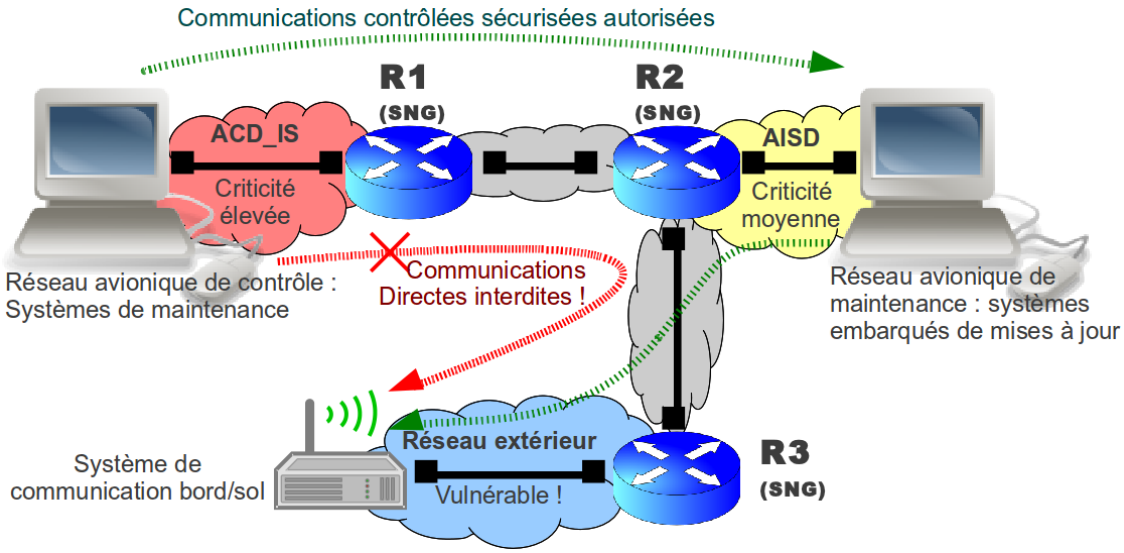



**(e)** Réseaux logiques permis par la topologie "C"

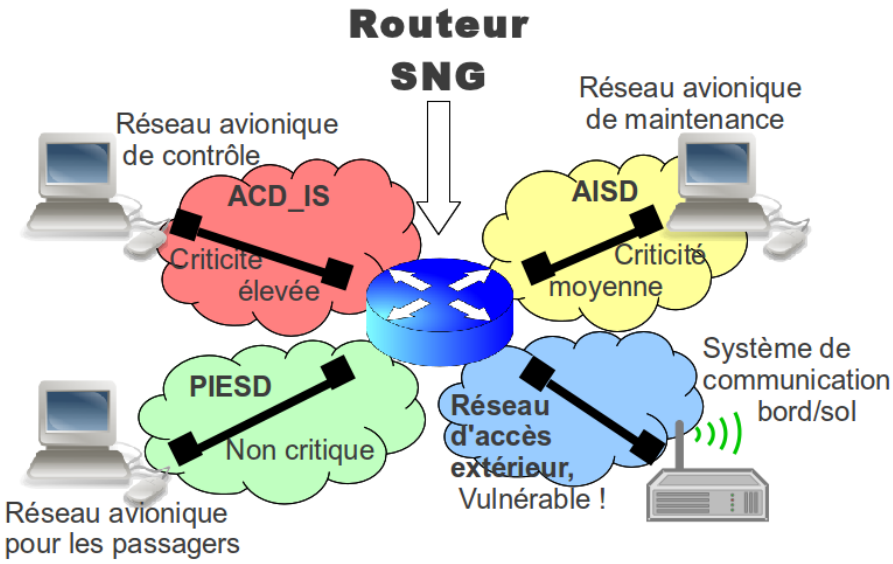



**(f)** Topologie "D" à quatre hôtes et deux routeurs

**Figure 6.1.:** Les différentes topologies réseau envisagées




**(a)** Topologie avionique proposée par THALES AVIONICS (cas n°1)




**(b)** Topologie avionique proposée par THALES AVIONICS (cas n°2)

**Figure 6.2.:** Topologies avioniques associées aux cas d'utilisation de THALES AVIONICS

### 6.1.2. Matériel mis en œuvre

Pour réaliser la topologie 6.1d, nous avons utilisé différents systèmes.

Les routeurs SNG embarqués ont fonctionné sur des architectures classiques de type PC, équipés de processeurs Intel® Xeon® E5603 cadencés à 1,60GHz. Ces processeurs 64 bits quadri-core équipés de caches L3 de 4 Mo de mémoire interne à 1,60GHz ont en fait été sous-exploités puisque le système de la plateforme IMA était configuré pour n'utiliser qu'un cœur et en 32 bits. De même, les 2 Go de mémoire RAM de type DDR2 cadencés à 800 MHz étaient sous-exploités : notre logiciel du routeur SNG nécessite moins de 2 Mo à l'exécution pour fonctionner.

Ces routeurs étaient chacun équipés d'une carte réseau Ethernet Gigabit quadri-ports Intel® PRO/1000 PT, carte basée sur deux contrôleurs 82571GB et connectée à un port PCI express (ce port «PCIe» offre un débit minimal de 2 Gbps pour les communications avec le reste du matériel). Ces cartes étaient configurées en mode 100base-Tx-FD, c'est-à-dire que les transmissions s'effectuaient à la vitesse de 100 Mbps en Full Duplex (soit 200 Mbps en cumulant les deux directions). Les systèmes étaient reliés entre eux par des câbles RJ45 croisés de catégorie 6, afin de limiter les pertes de données. Le Full Duplex a été choisi pour éviter les pertes et délais de retransmissions par collisions de trames Ethernet.

Les hôtes d'extrémités et les moniteurs étaient des postes de travail classiques équipés de systèmes Debian Lenny (5.0) avec un processeur dual core 32 bit à 2.6 GHz et avec 4 Go de RAM. Leur carte réseau était là aussi configurée manuellement en 100base-Tx-FD et nous avons vérifié durant les tests que ces hôtes n'étaient pas les causes des limites des performances mesurées durant les tests.

## 6.2. Métriques et outils

Parallèlement à la détermination de la topologie réseau et de sa mise en place dans notre laboratoire, nous avons choisi un certain nombre de métriques pour évaluer notre routeur SNG et nous avons aussi choisi les outils logiciels à utiliser pour quantifier ces métriques.

### 6.2.1. Métriques sélectionnées

Le domaine de la métrologie réseau nous fournit différentes métriques, mesurables à différents endroits de notre topologie réseau.

La première métrique est la **capacité de traitement** du routeur SNG, évaluée en octets par seconde. Cette métrique est plus souvent appelée « Débit [maximal] du routeur » ou encore le « débit opérateur ». Elle est mesurée en sortie du routeur SNG. Il est possible d'utiliser d'autres unités pour qualifier ce débit : en bit par

seconde (8 bits = 1 octet) ou en paquet par seconde lorsque les mesures se font avec des paquets dont la taille est constante.

La topologie utilisée étant ici la topologie de la figure 6.1d, il est intéressant d'étudier le débit entre les hôtes d'extrémité H1 et H2, aussi appelé « **débit utilisateur** ». Dans cette topologie, une relation relie ces débits «utilisateurs» et «opérateurs» : le chemin étant unique, le débit utilisateur est limité par les éléments du réseau, plus exactement par l'élément le plus limitant, le «goulet d'étranglement». Dans notre cas, nous avons utilisé des cartes Ethernet Full Duplex à 100 Mbps et les résultats présentés dans la section 6.3 montrent que l'élément limitant est notre routeur SNG. Ainsi, nous avons donc la relation « débit utilisateur entre H1 et H2 » = « débit opérateur/capacité de traitement du routeur SNG ».

Une troisième métrique est le **délai de traitement** des paquets par le routeur SNG. Ce délai induit une latence, mesurable entre H1 et H2 sous le nom de délai OWD (« One Way Delay »). Toutefois, ce délai OWD est complexe à mesurer : il impose de synchroniser les temps des deux systèmes d'extrémité et de vérifier et compenser les dérives des horloges durant l'expérimentation.

C'est pourquoi nous lui avons préféré la métrique du délai RTT, Round Trip Time, délai aller-retour. En effet, notre topologie étant symétrique et en mode Full Duplex, le délai RTT est donc le double du délai OWD. Cela résout le problème précédent de synchronisation des horloges. Le problème de dérive temporelle a été écarté aussi en raison de la différence d'ordres de grandeur des mesures de RTT (de l'ordre de la milliseconde) et de la dérive et de la précision des horloges utilisées (de l'ordre de la microseconde, les valeurs ayant été mesurées sur les systèmes observateurs).

La **gigue** est une autre métrique, dérivée des valeurs résultant des mesures de RTT.

Nous avons aussi mesuré le **taux de pertes** du routeur. Celui-ci n'a de sens qu'avec le protocole de transport UDP et non avec TCP ; ce dernier dispose en effet de mécanismes de détection et compensation des pertes, il n'y a donc aucune perte du point de vue de l'application qui utilise une connexion TCP pour communiquer.

Une dernière métrique que nous avons sélectionnée est la **disponibilité** du routeur. La majorité des systèmes avioniques est en effet censée fonctionner sans interruption ou se réinitialiser automatiquement en cas de panne récupérable.

## 6.2.2. Outils choisis

Pour quantifier ces métriques, différentes classes d'outils s'offrent à nous : les outils actifs de mesures et les outils passifs de mesures.

### 6.2.2.1. Outils de métrologie passive

Les outils passifs exploitent les données transitant sur les réseaux pour en extraire des informations. Les «sniffeurs» réseaux en font parti : ils écoutent les données et

les analysent à la volée, ou bien copient ces données sur disque pour une analyse ultérieure hors ligne.

Ainsi, le sniffeur réseau Wireshark [Combs 2013b] effectue l'écoute et l'analyse en ligne, produisant rapidement les résultats, mais il consomme beaucoup de ressources mémoire et processeur, ce qui peut fausser les résultats : paquets perdus car le processeur est trop chargé, arrêt brutal du logiciel par le système d'exploitation et instabilité globale quand la mémoire vive est saturée.

Une alternative que nous avons aussi utilisée est tcpdump [Van Jacobson & McCanne 2013], ou encore dumpcap [Combs 2013a] utilisé en interne par Wireshark. Ces autres sniffeurs réseau écoutent le réseau et sauvegardent dans des fichiers les données capturées.

Nous avons enfin utilisé le programme socat [Rieger 2011], surnommé le « nouveau couteau suisse du réseau », qui offre de nombreuses possibilités de relayage réseau : relais de données applicatives, de flux TCP de données, de paquets IPv4 et IPv6, de trames Ethernet, et passerelle entre ces différentes couches protocolaires mais aussi passerelle de type «proxy SOCKS » [Leech *et al.* 1996] ou encore redirection vers des fichiers et pseudo-fichiers. Nous l'avons utilisé pour dupliquer à bas coûts les paquets vers une interface réseau reliée à l'hôte externe dédié à l'observation de réseaux.

#### 6.2.2.2. Outils de métrologie active

La deuxième classe d'outils de mesure est la classe des outils actifs. En effet, dans certains cas, aucune donnée ne transite sur les réseaux ou alors les données transitant ne permettent pas d'extraire des informations utiles ou/et de qualité. Dans ce cas, les outils actifs émettent des données sur le réseau, ce qui peut donc impacter le réseau. L'émission doit donc être réalisée pour être la moins intrusive possible pour le réseau.

Ainsi, l'outil pathload [Dovrolis & Jain 2002, Constantine Dovrolis 2006] émet des rafales de paquets pour déduire la capacité du chemin testé. Lorsque ces rafales transitent sur le réseau, les éléments intermédiaires peuvent être congestionnés et donc perdre des paquets. Toutefois nos mesures avec cet outil se sont avérées trop variables et trop instables, non reproductibles, nous avons donc cherché d'autres outils de mesures.

Un autre outil actif que nous avons utilisé est l'outil «iperf» [Mark Gates 2013]. Cet outil permet de déterminer le débit sur un chemin, donc le débit de H1 à H2 via R1 et R2 dans notre cas. Il peut fonctionner en TCP, chargeant au maximum le chemin et indiquant alors le débit maximal avec une connexion TCP. Il peut aussi fonctionner en UDP et émet un débit constant de datagrammes, indiquant les taux de pertes et le débit de réception à l'arrivée. Cet outil est donc très intrusif.

Les mesures des délais RTT ont été effectuées à l'aide de l'outil standard «ping6». Cet outil envoie des petits paquets « ICMPv6 ECHO request » avec l'heure d'envoie.

| Métrique | Unité | Outils de mesure |
|---|---|---|
| Débit « utilisateur » | {_,k,M,G}{bit,octet,paquet}/s | iperf, Wireshark |
| Débit « opérateur » | {_,k,M,G}{bit,octet,paquet}/s | |
| Délai RTT | {_,m,µ,n}seconde | ping6, tcpdump |
| Délai de traitement du routeur | {_,m,µ,n}seconde | |
| Gigue | {_,m,µ,n}seconde | |
| Pertes | ( % ) | iperf en UDP |
| Disponibilité | | |

**Table 6.1.:** Synthèse des métriques et des outils

Lors de la réception du paquet « ICMPv6 ECHO response » en réponse à la requête, l'outil ping6 calcule le délai entre la requête et la réponse, délai qui est le Round-Trip Time. En mesurant de nombreux RTT, il est possible de déterminer la variance du RTT, appelée gigue. Cet outil, bien qu'actif, surcharge peu le réseau (104 octets envoyés par seconde sur un chemin de capacité supérieur à 10 Mbps dans notre cas), son intrusivité a donc un impact que nous avons jugé négligeable sur notre réseau expérimental.

Le tableau 6.1 résume les différentes métriques évaluées et leurs unités, ainsi que les outils utilisés pour les mesurer.

## 6.2.3. Trafics de charge

### 6.2.3.1. Les différentes charges

Nous avons déjà évoqué dans la section précédente le problème du changement des performances du réseau, selon qu'il y ait ou non des données en transit sur ce réseau. Ainsi, la mesure de la capacité maximale de traitement du routeur peut être sous-évaluée si un flux de données consomme une partie de la capacité simultanément à l'évaluation de capacité, par exemple lorsque plusieurs outils intrusifs évaluent la capacité dans un même temps.

Mais nous verrons dans les résultats que le groupe des mesures le plus impacté par l'état du réseau est le groupe de mesures de délais. Lors des mesures sur un réseau chargé par de nombreux flux de données, les paquets utilisés par notre outil actif ping6 de mesures ne sont pas prioritaires et doivent donc passer par des files d'attentes à l'entrée de chaque système ou sous-système. En conséquence, le délai aller/retour est fortement pénalisé par ces attentes, par rapport aux mêmes mesures lorsque le réseau n'a rien à acheminer.

Nous avons choisi d'étudier les performances du routeur SNG dans trois cas de charges de trafic. Le premier cas est celui où aucune donnée utilisateur ne circule sur le réseau, nous l'appellerons « **sans charge** ». Ce cas d'un réseau vide empêche l'utilisation des outils passifs de métrologie, mais permet d'obtenir les résultats

les plus fiables concernant les capacités maximales (débits) du réseau et de ses constituants.

À partir de cette capacité maximale mesurée sans charge, où l'élément limitant est à sa limite de saturation, nous pouvons mettre en œuvre le deuxième cas de charge que nous appellerons « **charge maximale** » ou encore « **pleine charge** ». Ainsi, le réseau est toujours entre les cas « sans charge» et «à charge maximale». Nous pouvons donc nous attendre à ce que, en utilisation réelle, les performances du routeur SNG soient bornées entre les performances «sans charge» et les performances «à charge maximale».

Nous avons aussi effectué des mesures avec un troisième cas de trafic de charge, la «**charge prévisionnelle**» ou «**charge nominale**», représentant un trafic que nous avons étudié pour être le trafic du contexte d'utilisation avionique du routeur SNG présenté section 2.2.

#### 6.2.3.2. Le trafic de charge nominale envisagée

Nous avons effectué une étude pour déterminer les types de flux susceptibles de transiter à travers le routeur SNG. Comme présenté dans la section 2.2.1, trois classes principales de trafic sont susceptibles de transiter sur les réseaux liés au routeur SNG : les flux ATSC, AOC+AAC et APC.

Ces flux sont respectivement ceux des systèmes hautement critiques nécessaires au vol (Air Traffic Services), des systèmes embarqués gérés par la compagnie pour les aspects commerciaux, de sécurisation, de régulation et d'optimisation du vol (Aeronautical Operational Control + Aeronautical Administrative Communication) et enfin ceux des futurs flux « passagers » (Aeronautical Passenger Communication) constitués des données échangés par les passagers vers les réseaux au sol via les infrastructures embarquées dans l'avion (par exemple, le Wifi fourni à bord par la compagnie).

Pour ces classes de flux, nous avons modélisé, selon les types de protocoles employés (TCP ou UDP), les délais entre paquets (appelés **Doff**, en secondes) et les tailles de flux (appelés **Don**, en octets), ainsi que leurs distributions mathématiques et leurs paramètres. Ainsi, **Doff** désigne la distribution des intervalles de temps entre les débuts de flux et **Don** désigne la distribution des durées des flux.

Les flux ATSC, AOC et AAC existent déjà, ils utilisent exclusivement le protocole de transport UDP [Ben Mahmoud 2012]. Nous avons déterminé à partir du COCR [ICAO 2006] les tailles minimales et maximales des paquets IPv4 échangés. Étant donné qu'il a été prouvé que le choix de la distribution a un impact négligeable sur les résultats [CNES 2009], nous avons choisi d'utiliser des distributions uniformes pour les Don et les Doff. Ainsi, nous avons modélisé ces flux, comme décrit dans la table 6.2.

Le cas des flux passagers est plus complexe. Bien qu'il existe quelques cas de services Internet embarqués, par satellites (voir la sous-section 2.1.3 à ce sujet), peu de

| Flux UDP | | Paramètres | Air vers Sol | Sol vers Air |
|---|---|---|---|---|
| ATSC | Don : Uniforme [octets] | Min | 88 | 88 |
| | | Max | 2500 | 4000 |
| | Doff : Uniforme [seconde] | Min | 0 | 0 |
| | | Max | 2 | 2 |
| AOC+AAC | Don : Uniforme [octets] | Min | 90 | 90 |
| | | Max | 5000 | 125 |
| | Doff : Uniforme [seconde] | Min | 0 | 0 |
| | | Max | 2 | 2 |

**Table 6.2.:** Modélisation par distributions mathématiques des flux ATSC, AOC et AAC

données publiques quantitatives existent et elles sont inadaptées aux moyens de communications et aux débits envisagés. En effet, THALES AVIONICS nous a demandé d'utiliser la valeur de 500 kbps (62.5 kBps) pour les débits montant vers l'avion et 300 kbps (37.5 kBps) pour les débits descendant vers le sol. Ces valeurs de débits se retrouvent chez certains fournisseurs [GogoInf 2013]. De plus, nous ferons abstraction du moyen physique de communication (satellite, ondes hertziennes ou autre [Besse 2013]) pour des raisons de secret industriel.

Ces débits futurs partagés par tous les passagers de l'appareil sont insuffisants pour des usages consommateurs de type échanges de fichiers pair-à-pair mais permettraient aux passagers de consulter leurs courriers électroniques et de disposer d'un accès (lent mais fonctionnel) aux services Web, comme c'est déjà le cas pour les fournisseurs actuels d'Internet embarqué [Oi 2013, GogoInf 2013, SwiftBB 2013] et avec les équipements de type Smartphone via les réseaux 3G [ARIB 2013].

Afin de modéliser le trafic APC, nous avons repris les conclusions d'études sur des trafics «Internet» [Steffen Gebert 2012, Olivier & N.Beameur 2001, Ben Mahmoud 2012, Oi 2013]. Ainsi, nous avons modélisé les Doff (temps entre flux) à l'aide de lois de Weibull($\lambda$, k, max), jugées aujourd'hui les plus réalistes par [Olivier & N.Beameur 2001]. Nous avons cependant ajusté les valeurs de plafond « max » pour que le débit total soit conforme à ce qui est attendu.

La loi de Pareto($X_m$, $\alpha$, max) a été utilisée pour modéliser les Don (durées des flux), avec comme paramètre Xm=4 octets (c'est la taille minimale observée dans les données échangées, hors entêtes TCP/UDP/IP). Le paramètre $\alpha$ est ajusté pour que les moyennes correspondent à ce qui est observé par [Olivier & N.Beameur 2001]. Enfin, le paramètre «max» est une borne maximale définie par les tampons internes du système d'exploitation ; sans ce plafond, la loi de Pareto a naturellement tendance à fournir des valeurs élevées de l'ordre du millier de milliard d'octets, rendant la modélisation inexploitable pour la génération de trafic réseau dans notre cas.

La table 6.3 résume les valeurs utilisées pour modéliser les flux TCP et UDP constituant l'APC.

| | | Paramètres | Air vers Sol | Sol vers Air |
|---|---|---|---|---|
| Flux TCP de l'APC | Don : Pareto [octets] | $X_m$ | 4 | 4 |
| | | $\alpha$ | 0.32 | 0.32 |
| | | Max | 1 000 000 | 1 000 000 |
| | | *(moyenne)* | *(27 400)* | *(27 400)* |
| | Doff : Weibull [seconde] | $\lambda$ | 0.55 | 0.55 |
| | | k | 0.7 | 0.7 |
| | | Max | 5.34 | 3.05 |
| | | *(moyenne)* | *(0.975)* | *(0.560)* |
| Flux UDP de l'APC | Don : Pareto [octets] | $X_m$ | 4 | 4 |
| | | $\alpha$ | 0.36 | 0.36 |
| | | Max | 131 072 | 131 072 |
| | | *(moyenne)* | *(4 800)* | *(4 800)* |
| | Doff : Weibull [seconde] | $\lambda$ | 0.28 | 0.28 |
| | | k | 0.74 | 0.74 |
| | | Max | 4 682 | 2 675 |
| | | *(moyenne)* | *(0.534)* | *(0.307)* |

**Table 6.3.:** Modélisation par distributions mathématiques des flux APC

## 6.2.4. le programme sourcesonoff

### 6.2.4.1. Les insuffisances des outils existants

Les tables 6.2 et 6.3 présentent la modélisation des trafics de l'APC, l'AOC, l'AAC et l'ATSC, ces quatre modélisations décrivant ensemble le trafic de charge nominal que nous souhaitons avoir entre les hôtes H1 et H2 via nos routeurs SNG R1 et R2 de notre topologie de tests de la figure 6.1d.

Afin de pouvoir avoir un trafic réseau conforme à cette modélisation, nous avons cherché des outils de génération de trafic en environnement réel. Cependant aucun des outils que nous avons trouvé ne répondait à nos contraintes. En effet, les outils existants sont généralement des outils d'enregistrement de trafic capables de rejouer les trafics préalablement capturés, comme tcpreplay [Turner 2000], cependant ces outils ne sont pas conçus pour générer un trafic qui n'existe pas encore tel que le trafic APC. De plus ils nécessitent des captures auxquelles nous n'avons pas accès, tel que le trafic ATSC.

Suite à nos recherches, les outils trouvés et capables de générer des trafics sur des réseaux réels se sont avérés limités à la génération d'un trafic constant, comme iperf [Mark Gates 2013], constitué souvent d'un flux unique TCP ou UDP, parfois de plusieurs flux simultanés mais tous similaires entre eux. Or, les études sur les trafics des réseaux locaux et Internet montrent que ce type de trafic artificiel n'est pas représentatif des observations de trafics réels [Steffen Gebert 2012, Olivier & N.Beameur 2001, Leland W. E. & V. 1994, Willinger W. & V. 1997].

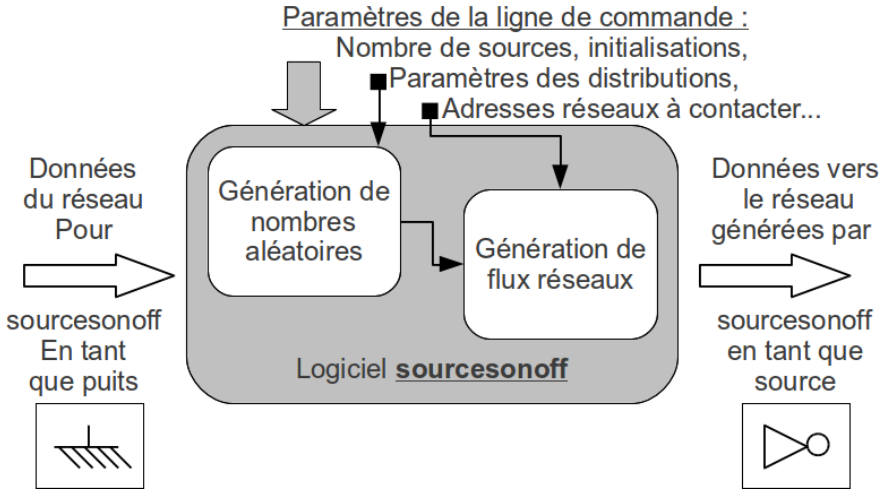



**Figure 6.3.:** Le logiciel sourcesonoff et sa structure interne

Nous nous sommes alors tourné vers les outils de génération plus réalistes. Les simulateurs réseaux (par exemple NS2 [Sandeep Bajaj 2013] et OmNet++ [Comm. 2013]) proposent ces outils capables de générer des flux aléatoires probabilistiquement contrôlés. Cependant, ce sont des simulateurs et non des systèmes générant des trafics réels, or notre routeur SNG est un système réel et non un système simulé. Les domaines des expérimentations environnement réel et en simulation, bien que voisins, s'avèrent cloisonnés dans la pratique.

En conséquence, nous avons choisi de développer notre propre outil de génération de trafics réseaux, outil que nous avons appelé sourcesonoff.

#### 6.2.4.2. Conception et développement de sourcesonoff

Le programme, développé en langage C et utilisé sur des systèmes Linux Debian, nous permet de générer le trafic de charge nominal pour les expérimentations. Les 3000 lignes de code sont organisées en deux parties, comme présenté sur la figure 6.3. La première partie permet, à partir des distributions et de leurs paramètres, de générer des ensembles de nombres aléatoires. Ces ensembles sont utilisés par la deuxième partie pour générer des flux TCP et UDP, plus exactement pour définir les temps entre créations de flux de données (les temps Doff) et les quantités de données à émettre à chaque flux (les valeurs Don).

Sur la topologie d'évaluation des performances, nous avons utilisé H1 pour les sources des flux «à bord» et H2 pour les sources «au sol». H2 reçoit les flux de H1 et est un puits : il reçoit les données et les jette silencieusement aux oubliettes. H1 est aussi un puits pour les flux émis dans H2.

Les distributions mises en œuvre sont les distributions uniformes, gaussiennes, de Poisson, exponentielles, de Weibull et de Pareto. Le détail de la génération des nom-

bres aléatoires sous forme d'entiers 64 bits est cependant un problème d'ingénierie qui sort du cadre de ce mémoire.

Il en est de même pour la mise en œuvre basée sur l'API socket compatible POSIX-2001 de la génération des ensembles simultanés de flux TCP et UDP. Notons juste que cette seconde partie a été conçu pour mettre en œuvre des sources ON/OFF, comme décrit par [Leland W. E. & V. 1994, Willinger W. & V. 1997], ce mécanisme étant d'après son auteur (et nous-même) générateur d'un trafic de charge plus réaliste qu'une génération uniforme et reproduisant les observations de réseaux opérationnels locaux (LAN, Local Area Networks) et internationaux (WAN, Wide Area Networks). Le choix de ce mécanisme a fait que nous avons nommé notre outil «sourcesonoff». Cet outil est public et diffusé gratuitement en Open Source sous licence libre GPLv3 [(FSF) 2007] .

Nous avons validé cet outil en comparant un trafic capturé sur un réseau local à 50 utilisateurs de l'ENAC avec un trafic généré par notre outil avec les paramètres adéquats. La validation a montré que les trafics sont similaires au niveau des couches réseau (IP) et de transport (TCP) : mêmes répartitions des durées des flux et des durées inter-flux, mêmes effets de mémoire à long terme sans mémoire à court terme, effets quantifiables avec des calculs d'autocorrélation et les estimations des valeurs de Hurst qui sont dans des intervalles associés aux données chaotiques et autosimilaires. Cette validation est l'objet d'une publication en cours d'écriture.

## 6.3. Résultats des mesures de performance

Le cadre expérimental et les métriques à mesurer ont été détaillés dans les sections précédentes. Nous avons effectué plusieurs campagnes de mesures pour évaluer les différentes caractéristiques du routeur selon différents paramètres. Les résultats exhaustifs sont trop nombreux pour être indiqués ici, de plus une certaine partie de ces résultats est redondante. C'est pourquoi nous avons choisi de ne présenter que les résultats les plus significatifs, montrant quels effets entraînent la variation de tel ou tel paramètre.

En effet, les performances varient en fonction de plusieurs paramètres.
- Tout d'abord le protocole de transport utilisé pour acheminer les données. Nous avons utilisé deux protocoles courants, les protocoles UDP et TCP, le premier orienté message et le second orienté flux continu de données, le second étant le seul à garantir l'absence de perte.
- Un deuxième paramètre est la charge imposée sur le réseau pendant les mesures. Nous disposons ici de trois charges détaillées section 6.2.3 : réseau à vide, réseau en charge nominale (charge prospective avionique) et réseau en charge maximale.
- Un troisième paramètre concerne le nombre de flux en parallèle utilisés pour les données actives des tests. Nous nous attendons en effet à ce que les performances du routeur soient différentes selon que les données cheminent via un unique flux TCP ou via une multitude de flux TCP. La même question se pose pour UDP, bien que la notion de flux soit plus difficile à déterminer en raison de l'absence de datagramme d'initialisation et de finalisation de «flux UDP».
- Un quatrième point concerne la sécurité, à savoir le coût d'utilisation des mécanismes de confidentialité et d'intégrité+authentification. Ces deux derniers étant intimement liés dans la mise en œuvre du routeur SNG (ils utilisent tous deux la même signature), nous considérerons ce couple comme indissociable.
- Un cinquième point concerne la taille des paquets IP échangés.
- Nous nous sommes aussi posé la question du coût engendré par la position de l'entrée de routage dans la table de routage.
- De même, nous avons vérifié la disponibilité du routeur sur une «longue période» d'une semaine, alors que la majorité des autres mesures prenait un temps de l'ordre de la minute voir de l'heure.
- Nous avons aussi étudié l'impact de l'utilisation d'un émulateur à la place d'un système réel.
- Rappelons que nous avons mesuré des délais, des débits et d'autres métriques.
- Notons que chaque résultat présenté ici est le fruit d'au minimum 5 répétitions de la mesures, parfois plus, cela nous permettant de vérifier la stabilité du résultat et sa reproductibilité.

Nous pouvons résumer ces paramètres indépendants dans un hypercube à 10 dimensions décrits par la table 6.4, hypercube que nous avons parcouru pour aller de l'origine à différents sommets résumés dans les sections suivantes.

Pour rappel, les résultats présentés ici ont été effectués sur la topologie présentée

| Dimension | Valeurs possibles |
|---|---|
| Métrique | {Délais de RTT [s], Débits/capacités de traitement [Mbps, pps], ... } |
| Protocole de transport | {TCP, UDP} |
| Charge pendant le test | {À vide, nominale (800 kbps), maximale (48 Mbps)} |
| Nombre de flux actifs | {1, 4, 8, 16}⊂ $\mathbb{N}^\star$ |
| Sécurité | {sans sécu., conf., intég.+auth., conf.+intég.+auth.} |
| Taille des paquets IP | {60, 1200, 1500}⊂ $\mathbb{N}^\star$ |
| Position dans la table de routage | {1, 2, 3, 4, 5}⊂ $\mathbb{N}^\star$ |
| Durée du test | De 60 secondes à 7 jours |
| Répétition du test | ≥ 5 fois |
| Matériel | { réel (cible embarquée), émulé (qemu)} |

Unités: pps=paquet par seconde, kbps=kilobit par seconde, Mbps=megabit par seconde

**Table 6.4.:** Les dimensions de l'espace parcouru pour l'évaluation

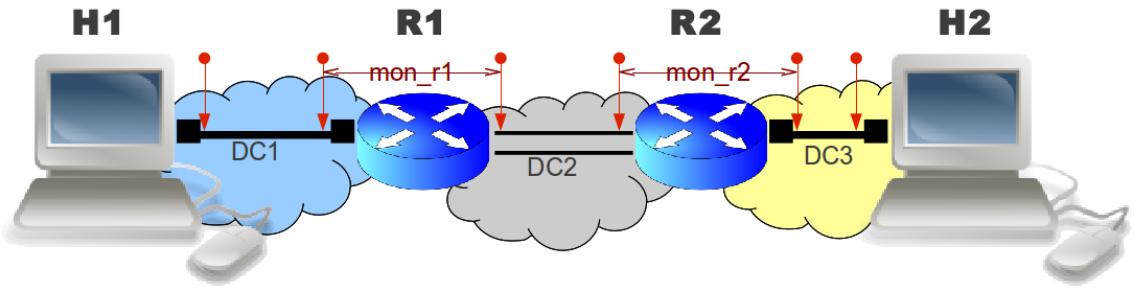



**Figure 6.4.:** Topologie principale utilisée pour l'évaluation des routeurs SNG

figure 6.1d et recopiée dans la figure 6.4.

## 6.3.1. Premiers résultats préparatoires

Les premiers résultats que nous présenterons ont été effectués avec un seul flux en UDP puis en TCP, sans sécurité. Nous ajouterons la sécurité et le multi-flux dans la sous-section suivante.

### 6.3.1.1. Capacité du routeur SNG avec un flux UDP

La première série de mesures a consisté à émettre depuis la source H1 un flux de données en UDP de débit constant et à mesurer en entrée du puits H2 la quantité de données qui a traversée avec succès les routeurs SNG intermédiaires R1 et R2.

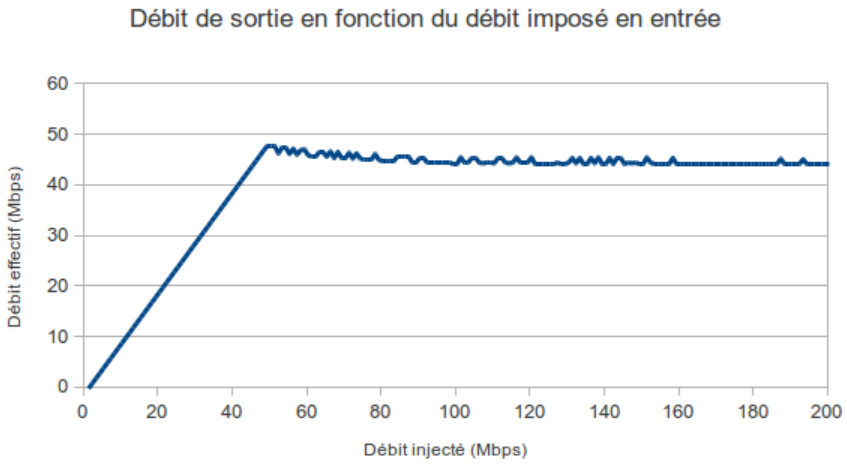



**Figure 6.5.:** Capacité du routeur SNG avec un flux UDP

Dans ces expérimentation comme dans toutes les suivantes relatives aux mesures de capacités, nous avons utilisé des paquets IPv6 de taille maximale, la MTU étant ici fixée à 1500 octets.

Les capacités mesurées sont les débits «utilisateurs», mesurées au niveau de la couche applicative, et non des débits IP ou Ethernet. Les mesures de capacité exploitant ici toujours la taille maximale de paquets IP, il est possible de passer du débit applicatif au débit IP en appliquant un ratio de [ 1 500 / (1 500 - 40 (ipv6) - 8 (udp) ] = 1 500/1 452 ~ 1.033, les entêtes IPv6, UDP ou encore TCP étant respectivement de 40, 8 et 20 octets inclus dans chaque paquet de 1500 octets.

Le flux est émis de manière continue pendant quelques secondes pour s'assurer de la régularité de l'arrivée des données sur H2. Dans les résultats présentés ici, nous avons maintenu le flux à débit constant pendant environ 30 secondes et à l'arrivée les données sont mesurées en kilobits par seconde.

Nous avons fait varier le débit d'émission, appelé «Débit injecté», de 0 à 200 Mbps par pas de 100 kbps, chaque pas durant 30 secondes pour s'assurer du régime permanent de la transmission. Le débit reçu est appelé ici «Débit effectif» et est représenté sur la figure 6.5.

La courbe obtenue est linéaire entre 0 et 48,2 Mbps, puis après ce plafond le débit reçu redescend et tend vers une asymptote à 44 Mbps, soit 91 % du maximum.

Pour comparaison, la courbe théorique à laquelle nous nous attendions était

$$D_{effectif} : D_{injecté} \mapsto \begin{cases} D_{injecté} & si\ D_{inj} \leq \widehat{D_{udp}} \\ \widehat{D_{udp}} & sinon \end{cases} \tag{6.1}$$

où $\widehat{D_{udp}}$désigne le débit maximal de traitement du routeur SNG pour un flux UDP. Cette courbe théorique traduit bien la montée linéaire du débit effectif, mais nous

n'avions pas envisagé la stabilisation asymptotique à une valeur différente du débit maximal. Cette redescente peut s'expliquer par les débordements des différentes files d'attente des routeurs SNG.

Étant donné que les éléments du réseau autres que les routeurs SNG sont surdimensionnés et ont été validés avec des débits supérieurs, nous en déduisons que l'élément limitant est ici le routeur SNG. Le routeur SNG est donc capable de traiter jusqu'à $\widehat{D_{udp}} = 48{,}2$ Mbps (soit environ 6 Mo/s) pour un flux UDP. Nous utiliserons cette valeur comme base pour la «charge maximale» des expérimentations suivantes.

Il faut aussi noter que les routeurs que nous avons utilisés disposent de 4 interfaces réseaux. Dans cette expérience, les flux sont monodirectionnels, entrent par une interface et sortent par une deuxième. Nos expérimentations pour utiliser d'autres interfaces, voir utiliser plusieurs interfaces d'entrée simultanément n'ont montré aucune différence : le débit total que peut gérer le routeur SNG reste $\widehat{D_{udp}} = 48{,}2$ Mbps, quelque soit la ou les interfaces d'entrées et de sorties, validant le fonctionnement de notre ordonnanceur de paquets mettant en œuvre un round-robin paquet par paquet, comme décrit dans la section 4.2.3.

#### 6.3.1.2. Étude des pertes de paquets en UDP

Parallèlement à cette expérimentation, nous avons mesuré le taux de pertes de paquets sur H2, c'est-à-dire le ratio paquets reçus par H2 par rapport au nombre de paquets injectés par H1.

La courbe théorique a pour équation

$$f : D_{injecté} \mapsto \begin{cases} 0 & si\ D_{inj} \leq \widehat{D_{udp}} \\ \frac{D_{inj} - \widehat{D_{udp}}}{D_{inj}} & sinon \end{cases} \tag{6.2}$$

La courbe expérimentale représentée sur la figure 6.6 «colle» parfaitement à la courbe théorique que nous attendions : aucune perte en deçà du débit maximal de 48.2 Mbps, puis une courbe exponentielle retournée tendant vers 1 (=100%) au-delà. Le phénomène d'escaliers visible sur la courbe est un artéfact dû aux arrondis des résultats par notre outil de mesures actives, iperf.

#### 6.3.1.3. Capacité du routeur SNG avec un flux TCP

Après avoir mesuré la capacité de traitement d'un flux UDP par notre routeur SNG, qui est dans notre cas précis la capacité du chemin de H1 à H2 via R1 et R2, nous avons fait varier le protocole de transport et mesuré la capacité avec le protocole TCP.

Il est important de rappeler que le protocole TCP dispose de mécanismes ajustant le débit en fonction des pertes et des délais pour maximiser le débit de la connexion

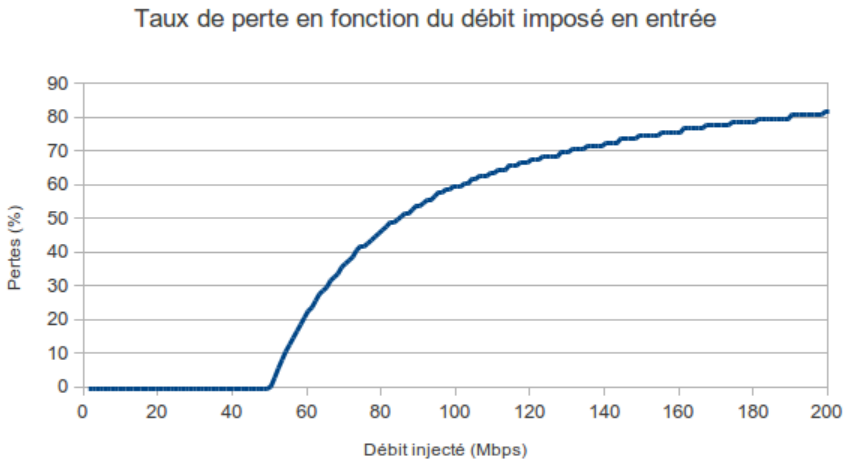



**Figure 6.6.:** Pertes de paquets avec un flux UDP

TCP tout en minimisant la surcharge sur le chemin. La version de TCP utilisée ici est le TCP Cubic [Sangtae Ha & Xu 2008, Sangtae Ha & Xu 2007], version actuellement par défaut pour les systèmes Debian et Ubuntu.

Ainsi, contrairement au cas précédent, nous n'imposons pas de débit d'émission à TCP ; nous nous contentons de lui fournir un flux applicatif de données suffisamment volumineux à transmettre et laissons le soin au protocole de découper le flux applicatif en segments et d'optimiser la transmission de ces segments.

La figure 6.7 illustre les deux approches en montrant pour TCP et pour UDP l'évolution des débits réseaux au cours du temps.

Ensuite, un outil de métrologie passive, tel que Wireshark, représente la courbe de débit instantanée, courbe se stabilisant rapidement à une valeur «plafond» que nous avons mesurée : $\widehat{D_{tcp}} = 42.8\,Mbps$. La courbe atteint très rapidement la valeur plafond puis est stable. La phase transitoire est très courte et ne nous intéresse pas ici, c'est pourquoi nous ne l'avons pas représentée.

Nous avons bien entendu répété les mesures et vérifié que l'élément limitant était le routeur SNG ; une erreur courante consiste en effet à utiliser un fichier lu sur un disque dur (lent) et à mesurer la vitesse de lecture du disque au lieu de mesurer la capacité du chemin réseau.

$\widehat{D_{tcp}} < \widehat{D_{udp}}$, car les mécanismes de régulation de débits sont imparfaits et bien qu'ils se rapprochent le plus possible de l'optimum, ils ne l'atteignent jamais. Nous verrons plus loin que plusieurs flux TCP coexistants sur le même chemin entraînent une dégradation des performances par rapport au cas mesuré ici où seul un flux sature les capacités des routeurs.

Un autre élément à prendre en compte concerne les entêtes des protocoles TCP et UDP : le protocole nécessite 20 octets tandis que l'entête UDP n'en prend que 8.

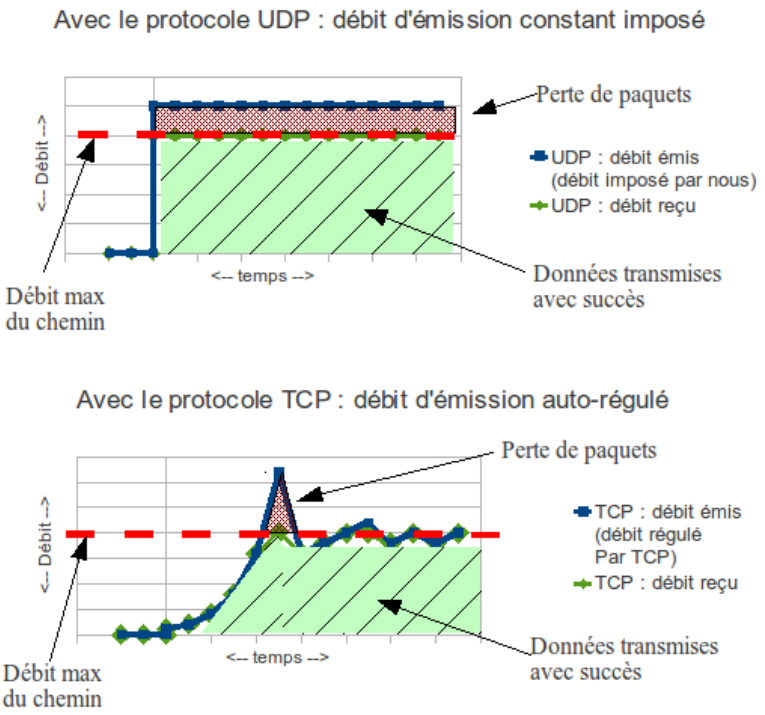


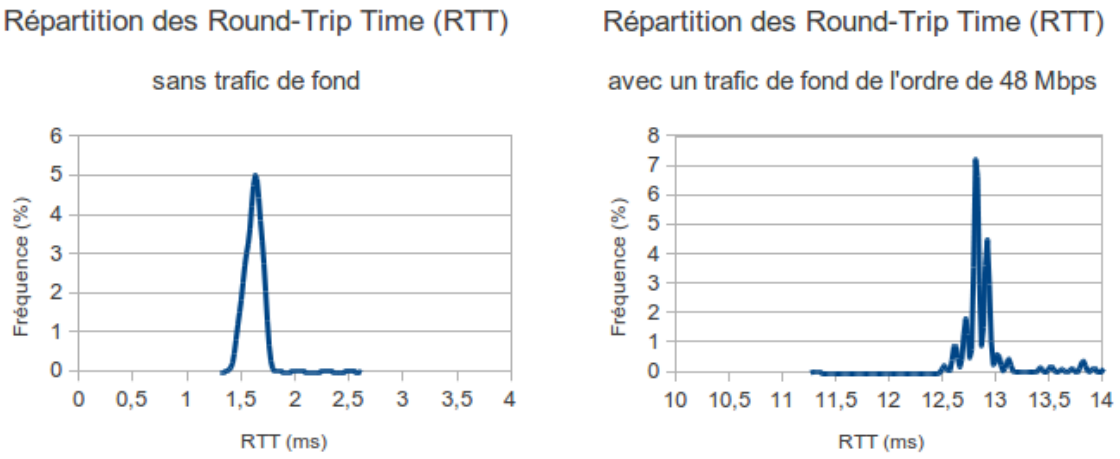

**Figure 6.7.:** Débits instantannés en TCP et en UDP au cours du temps

Les paquets IP étant de taille constante maximale, limitée par la MTU du chemin (ici configurée à 1500 octets), il en résulte que chaque datagramme UDP transporte 12 octets de données applicatives supplémentaires par rapport aux segments TCP. Cette différence de (12/1500=) 0.8 % est cependant négligeable ici. Rappelons que la spécification d'IPv6 [Jankiewicz *et al.* 2011] impose aux systèmes une MTU minimale de 1280 octets, la différence reste alors de l'ordre de 1%.

$\widehat{D_{tcp}}/\widehat{D_{udp}} = 42.8/48.2 = 89\%$, soit une « sous-efficacité » apparente, en terme de débit maximal, de TCP par rapport à UDP. Cependant TCP fournit la garantie d'acheminement des données ou de notification de non délivrance, sous forme d'un flot continu et non de messages, préservant l'ordre des données, il ne nécessite pas d'information préalable sur la capacité des chemins du réseau et ajuste dynamiquement son débit en cas de modification de cette capacité. Donc les protocoles TCP et UDP ne répondent pas aux mêmes besoins et sont plus complémentaires que concurrents.

#### 6.3.1.4. Répartition des délais RTT selon deux charges extrêmes

Après nous être intéressé aux capacités du chemin et donc des routeurs SNG, nous nous sommes intéressés à la latence induite par nos équipements réseaux. Pour cela, nous avons utilisé l'outil «ping6» qui permet de mesurer le temps d'aller-retour d'un




**(a)** Distribution des RTT à vide **(b)** Distribution des RTT en charge maximale

**Figure 6.8.:** Répartition des délais aller-retour (RTT)

paquet ICMPv6 entre les hôtes H1 et H2.

**Sans charge, mesures « à vide »** Nous avons d'abord mis en place des scripts pour automatiser l'émission d'une requête de 12 octets applicatifs (soit un paquet IPv6 de 60 octets) sur le réseau. Cette requête est émise une fois par seconde, pendant 1 000 secondes. Cette série a été répétée pour s'assurer de la stabilité et de la fiabilité des mesures.

Ensuite, nous avons tracé la courbe de fréquence des délais RTT. La courbe obtenue, visible sur la figure 6.8a, est proche d'une gaussienne, centrée à **1.61 ms** et d'écart type **0.11 ms**.

Il est important de noter que sur la topologie utilisée pour nos expérimentations,

1. les routeurs SNG R1 et R2 sont identiques matériellement et logiciellement,
2. les configurations sont symétriques,
3. le délai induit par les liaisons RJ45 croisés de 3 mètres sont supposés négligeables (de l'ordre de la nanoseconde),
4. les hôtes d'extrémités H1 et H2 sont supposés ne pas induire de latence supplémentaire*,
5. le routage des paquets est symétrique, les paquets font l'aller et le retour par le même chemin.

Ces 5 hypothèses nous permettent donc de postuler que le délai RTT est 4 fois le délai induit par le routeur SNG pour traiter un paquet ICMPv6 :

$$RTT = 4 \cdot T_{traitementRTsng} \tag{6.3}$$

| | Distribution des délais RTT | Distribution **majorant** les délais de traitement SNG |
|---|---|---|
| Réseau à vide | gaussienne (**1.61** ms, **0.11** ms) | gaussienne (**0.40** ms, **0.03** ms) |
| Réseau en charge nominale | gaussienne (**1.61** ms, **0.11** ms) | gaussienne (**0.40** ms, **0.03** ms) |
| Réseau en charge maximale | gaussienne (**12.80** ms, **0.20** ms) | gaussienne (**3.20** ms, **0.05** ms) |

Note : gaussienne(**moyenne, écart-type**)

**Table 6.5.:** Distributions des délais selon la charge

Les délais RTT suivant ici une distribution Gaussiennne(moyenne=1.61, écart-type=0.11), le routeur SNG induit donc, sur un réseau à vide, un délai dont la distribution mathématique est une distribution gaussienne(moyenne=**0.403** ms, écart-type=**0.028** ms).

En fait, la quatrième hypothèse marquée d'une «*» est fausse, la latence induite par les hôtes d'extrémité n'est pas négligeable (nos mesures indiquent qu'elles représentent jusqu'à 25% dans ce cas), donc la distribution gaussienne(moyenne=**0.403** ms, écart-type=**0.028** ms) est une surestimation des délais de traitement du routeur SNG.

**En charge maximale** La deuxième figure 6.8b montre la distribution des RTT lorsque le réseau est chargé, c'est-à-dire lorsque le trafic parasite est de l'ordre de 48 Mbps en plus des requêtes et réponses ICMPv6. Bien que la nouvelle courbe soit plus «dentée» et irrégulière, elle peut elle aussi être modélisée par une distribution gaussienne, centrée sur **12.8** ms avec un écart-type de **0.20** ms. L'équation précédente 6.3 nous donne donc une distribution du délai de traitement par le routeur SNG qui est, en charge maximale, majorée par une distribution gaussienne(**3.2** ms, **0.050** ms).

**En charge nominale** Nous avons aussi mesuré les latences en charge nominale, soit une charge de 500 kbps montants et 300 kbps descendants. Cependant nous n'avons pas représenté les latences car la courbe des fréquences des RTT se confond parfaitement avec celle mesurée « à vide », la distribution est la même que celle visible sur la figure 6.8a. Nous en avons déduit qu'une charge nominale donnait les mêmes tendances qu'aucune charge, ce qui peut s'expliquer par le fait que la charge nominale de 800 kbps équivaut à 1.6% de la charge maximale à 48.2 Mbps que peut traiter le routeur, elle est donc ici négligeable.

La table 6.5 synthétise les distributions discutées dans cette section.

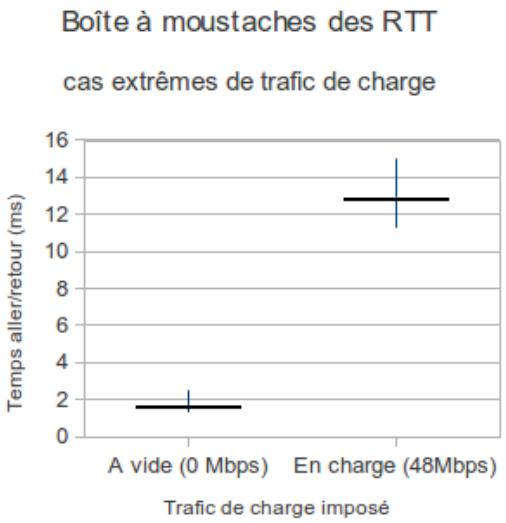



**Figure 6.9.:** Boîte à moustaches des RTT

**Comparaison des latences selon la charge** Nous avons choisi de comparer les cas extrêmes de charges sur un même diagramme, de type boîte-à-moustaches. Chaque série de donnée est représentée par un rectangle s'étendant entre les premier et troisième quartiles (Q1 et Q3), avec une barre centrale symbolisant la médiane. Ce rectangle est prolongé par deux traits s'étendant jusqu'aux premier et dernier déciles (D1 et D9). Ce diagramme permet donc de représenter et synthétiser rapidement des données statistiques.

Cependant, sur la figure 6.9 où sont représentés les médianes et quantiles des distributions des latences RTT hors charge et avec charge maximale, on ne peut distinguer les médianes des quartiles. Cela montre une propriété importante extraites de nos mesures : les variations des mesures de RTT, aussi appelées gigues, sont très faibles, toutes les mesures sont concentrées autour de la moyenne. Cela montre graphiquement que le rapport gigue/RTT est très faible. Nous y reviendrons dans la section 6.3.2.4.

La figure 6.9 montre aussi visuellement la différence de délais d'acheminement dans un réseau «vide» et «chargé». En effet, les RTT sont multipliés par 8 lorsque le réseau est en charge maximale. Cela s'explique par la mise en file d'attente des paquets en charge, les cartes réseaux et les routeurs ne pouvant traiter les paquets simultanément, ceux-ci sont stockés et traités dans leur ordre d'arrivée par les systèmes, ce qui impacte négativement les performances.

**Impact de la taille des paquets sur la latence** Pour terminer ce premier groupe de mesures, nous nous sommes aussi intéressé à l'impact de la taille des paquets sur les latences.

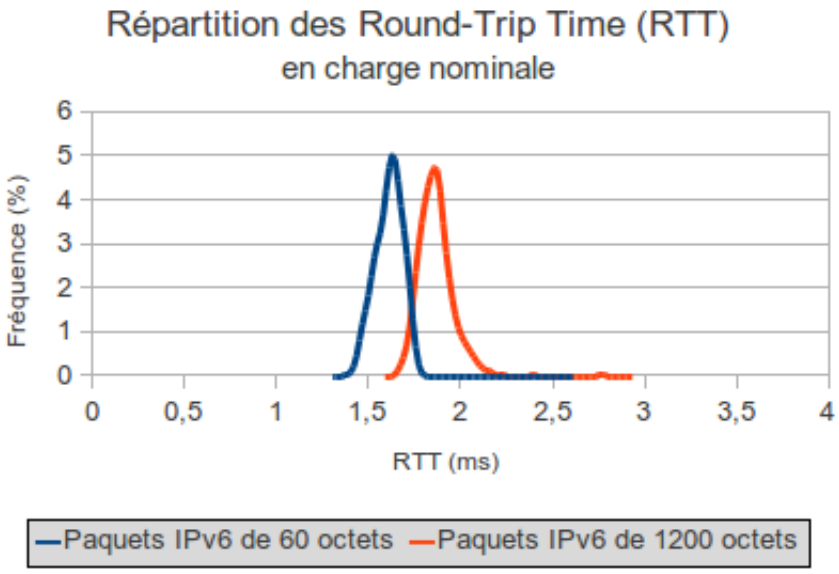



**Figure 6.10.:** Impact de la taille des paquets sur les délais RTT

La figure 6.10 montre les mesures effectuées sur notre réseau en charge nominale (800 kbps). La première série a été effectuée avec des paquets IPv6 de 60 octets (dont 12 octets «utiles», les 48 autres étant les entêtes IPv6 et ICMPv6). La seconde a été effectuée avec des paquets 20 fois plus volumineux, de 1200 octets (dont 1152 utiles).

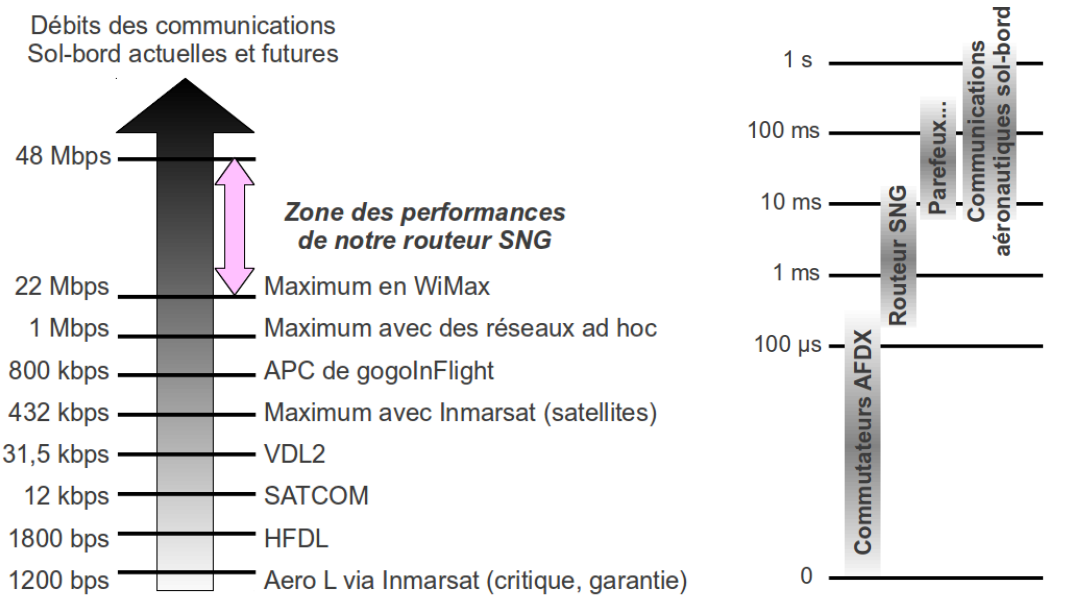
En dehors de distributions toutes deux gaussiennes, les mesures montrent une augmentation de 15% engendrée par l'usage de paquets plus gros. De plus, la seconde distribution, effectuée avec de plus gros paquets, présente une légère asymétrie : la moitié droite est plus «lourde» et descend moins vite que ne monte la moitié gauche de la courbe de fréquences.

#### 6.3.1.5. Contexte d'utilisation et validation des performances

Cependant, avant d'aller plus loin, il est nécessaire de nous interroger sur la pertinence de ces premiers résultats, de les comparer avec l'existant et les besoins d'un routeur embarqué dans un avion pour assurer des services de routage, de filtrage et de sécurisation de données potentiellement critiques.

Les documents [Aryoba 2013, Cisco 2009] contiennent des informations relatives aux performances «brutes» de routeurs Cisco, qui vont de quelques centaines de kbps à quelques dizaines de Gbps en débit de paquets IP routés.

S'il est vrai que les performances évaluées ici de notre routeur SNG sont inférieures à celles de la majorité des routeurs Cisco® professionnels, onéreux et de coût potentiel de certification prohibitif, actuellement les avions ne disposent pas de moyens de




**(a)** Besoins en terme de débits **(b)** Besoins en latence

**Figure 6.11.:** Besoins de performances pour les communications de données aéronautiques

communication de grande capacité, comme le montre les tables 2.2 et 2.1 du chapitre 2.

Ainsi, les débits des systèmes sol-bord sont de l'ordre du kilobit et les usages futurs (WiMax) montent au maximum à 22 Mbps, comme l'illustre la figure 6.11. Quant aux délais de transmission, ceux-ci se mesurent en dizaines de millisecondes, voir plus pour les communications par satellite. Ainsi, sans sécurité, le routeur SNG sur notre matériel embarqué est suffisant pour fournir des performances acceptables pour les communications sol-bord.

Pour les communications bord-bord, les débits sont de l'ordre de 100 Mbps avec des exigences élevées en terme de latences au niveau de la couche liaison (couche 2 AFDX). Par exemple, [Ian Land 2009] indique que les commutateurs AFDX doivent avoir une latence inférieure à 100 µs et que la gigue maximale est de 500 µs pour l'acheminement d'un paquet entre deux systèmes avioniques.

Cependant, au niveau de la couche réseau (couche 3 IPv6), aucune limitation ni objectif n'a pour l'instant été défini pour les équipements d'interconnexion de réseaux, tel que notre routeur SNG. Les interconnexions entre réseaux sont pour l'instant très limitées pour garantir la sûreté et la sécurité des différents réseaux avioniques.

Suite aux réunions avec THALES AVIONICS, nous avons établi que les informations à échanger via le routeur SNG représentent un faible volume, de l'ordre de quelques mégaoctets en dix secondes pour les téléchargements de maintenance (soit un débit moyen de 1 Mo/s, donc 8 Mbps). Concernant la latence, là encore, les besoins

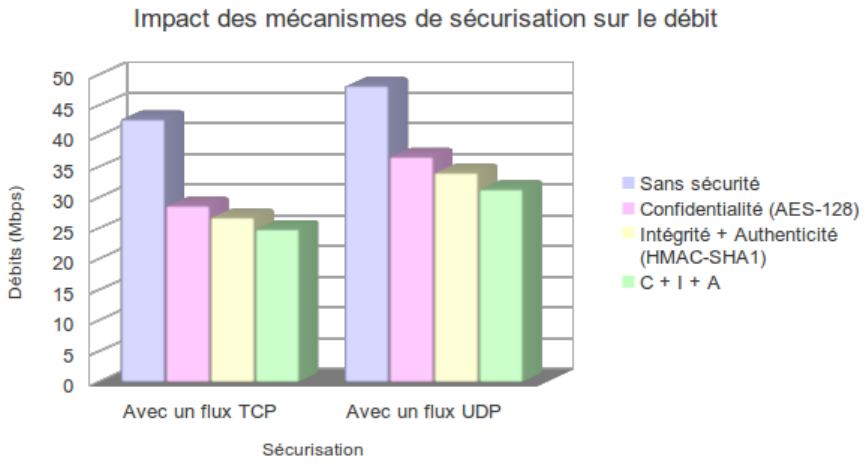



**Figure 6.12.:** Capacité du routeur SNG avec un flux sécurisé

opérationnels de l'ordre de 100 ms sont largement couverts par notre routeur SNG.

Toutefois, ces premiers résultats ont étudié les performances du routeur SNG (débits maximum et temps de traitement du routeur) en l'absence des mécanismes de sécurisation et avec le cas - optimal ? - d'un seul flux de données. Comment évoluent ces performances lorsque la sécurité est activée et que plusieurs flux partagent les mêmes canaux de communication ? C'est l'objet de la section suivante.

## 6.3.2. Étude de l'impact des mécanismes de sécurisation

Dans la section précédente, nous avons mesuré les performances de paquets IP routés, sans exploiter les mécanismes de sécurisation mis en œuvre par le routeur. Dans cette section, nous étudions l'impact de l'activation de la sécurisation sur les performances du routeur.

Ces mécanismes permettent l'encapsulation des données dans un tunnel à l'aide du protocole ESP[Kent 2005b], le chiffrement par l'algorithme AES-CBC-128[NIST 2001] pour assurer la confidentialité des données et la signature par l'algorithme HMAC-SHA1[NIST 2012, NIST 2008] pour en assurer l'intégrité et l'authenticité. Ces mécanismes de sécurisation ont été introduits dans la section 4.4.

### 6.3.2.1. Capacité du routeur SNG avec un flux sécurisé

Nous nous sommes d'abord intéressé à l'impact de la sécurisation sur le débit. Nous avons donc repris les expérimentations présentées dans les sous-sections 6.3.1.1 et 6.3.1.3 relatives aux mesures des capacités du routeur avec respectivement un flux

UDP et un flux TCP, puis nous avons réeffectué ces expériences en faisant varier la sécurisation : sans sécurité (déjà mesurées précédemment, ces capacités maximales de traitement $\widehat{D_{udp}}$ et $\widehat{D_{tcp}}$ servent donc de références), avec chiffrement (confidentialité), avec signature (intégrité et authentification) et enfin avec chiffrement ET signature (donc tous les services de sécurité activés).

Les résultats de nos 8 mesures sont représentés par le graphique 6.12.

Nous pouvons à nouveau constater que les débits en TCP sont toujours légèrement inférieurs (-10%) à leurs équivalents avec UDP, comme déjà constaté et justifié dans la section 6.3.1.3 ; la sécurisation ne change pas cette tendance.

La sécurité a ici un impact mesurable directement. Quelque soit le protocole de transport utilisé, le chiffrement entraîne une réduction de 7.5% de la capacité maximale de traitement du routeur SNG, tandis que la signature la pénalise de 14 %. Notre mise en œuvre de l'algorithme HMAC-SHA1 est en effet plus consommateur en ressources que celle de AES-CBC-128 [Alshamsi 2005].

Dans les deux cas, il est nécessaire d'encapsuler les données sécurisées dans un autre paquet IPv6 et de décapsuler les données à l'arrivée. Ces opérations, regroupées sous le terme de «tunnellisation», ont un coût engendrant une baisse de 13% en UDP et 22% en TCP.

Cette différence entre ces deux protocoles de transport peut s'expliquer par les mécanismes de régulation de débit de TCP. UDP ne disposant d'aucun mécanisme de ce type, nous mesurons alors l'impact «pur» de la tunnellisation : -13% sur les débits. À cet impact «pur» s'ajoute aussi les effets de l'augmentation du délai de traitement des paquets (que nous quantifierons plus loin) : le protocole TCP mesure le RTT des données échangées et adapte son débit en conséquence. Ici, l'augmentation de délai entraîne une baisse de débit de 22%. Nous étudierons les délais plus en détails plus loin dans la sous-section 6.3.2.3.

Pour résumer, l'ensemble des services de sécurité entraîne une baisse des débits de l'ordre de -35% à -43%.

#### 6.3.2.2. Impact du nombre de flux sur la capacité du routeur SNG

Jusqu'à présent, les mesures étaient basées sur l'étude d'un unique flux (TCP ou UDP) de données. La question à laquelle nous cherchons une réponse dans cette section est la suivante : y a-t-il une différence si nous employons plusieurs flux simultanés ?

**Avec des flux UDP** Nous présenterons d'abord les mesures de capacité effectuées en UDP. En plus des mesures effectuées avec un unique flux UDP saturant la capacité de traitement du routeur SNG (équivalent à saturer le chemin entre H1 et H2, les raisons de ce raccourci sont expliquées dans la section 6.3.1.1), nous avons réeffectué l'expérience en utilisant 4 flux en parallèles, puis 8 puis 16.

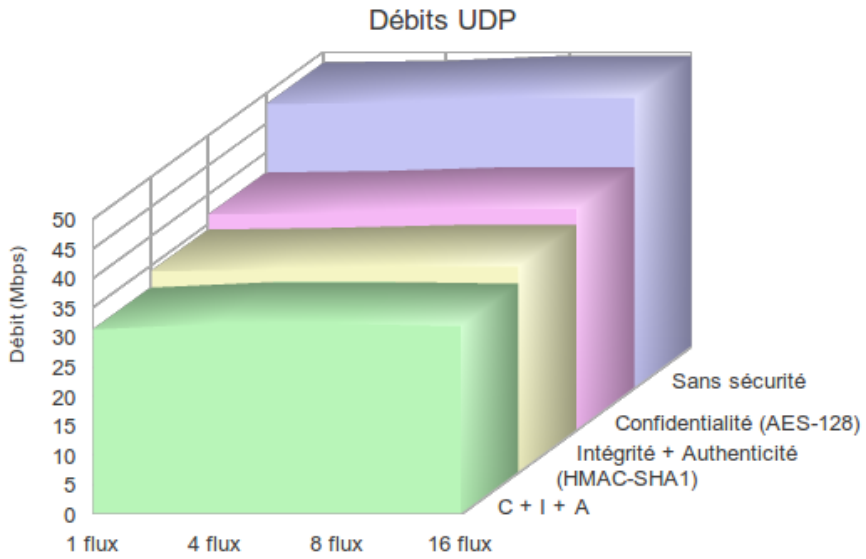



**Figure 6.13.:** Capacité du routeur SNG avec plusieurs flux UDP sécurisés

À chaque fois, nous avons reporté sur la figure 6.13 la quantité totale de données reçues par le puits H2, i.e. la somme des débits de réception pour les 4, 8 ou encore 16 flux simultanés.

En UDP, les flux coexistent mais sont indépendants. Ainsi, l'augmentation du nombre de flux n'entraîne pas d'augmentation du débit de réception, d'où l'apparence d'un escalier à marches plates de la figure 6.13.

En fait un grossissement des différences montre que l'augmentation du nombre de flux entraîne bien une légère augmentation du débit, mais négligeable (jusqu'à +2% pour 16 flux). Nous avons reproduit l'expérience sans routeur SNG, ce qui nous a permis de confirmer cette tendance, probablement due à une meilleure gestion des files d'attentes en sortie de l'interface réseau de H1 dans le cas multiflux.

Un autre point important concerne les interactions des effets du multiflux avec la sécurité. En effet, la figure 6.13 montre que ces deux paramètres sont indépendants. Cela se justifie par la mise en œuvre des fonctionnalités de sécurité du routeur qui manipule des paquets IP, indépendamment de leur contenu ; le routeur SNG ne travaille pas aux niveaux des couches supérieures à la couche réseau (3), il ne différencie donc pas les flux issus d'un même hôte logique (même adresse IP pour les flux).

**Avec des flux TCP** Nous avons ensuite renouvelé l'expérience en mesurant le débit atteint par des connexions TCP. La figure 6.14 présente les résultats sur un graphique de même catégorie que précédemment pour bien mettre en avance les différences entre TCP et UDP. Là encore nous avons représenté la somme totale des débits, plus utile que le débit obtenu par une connexion TCP parmi les 4, 8 ou 16.

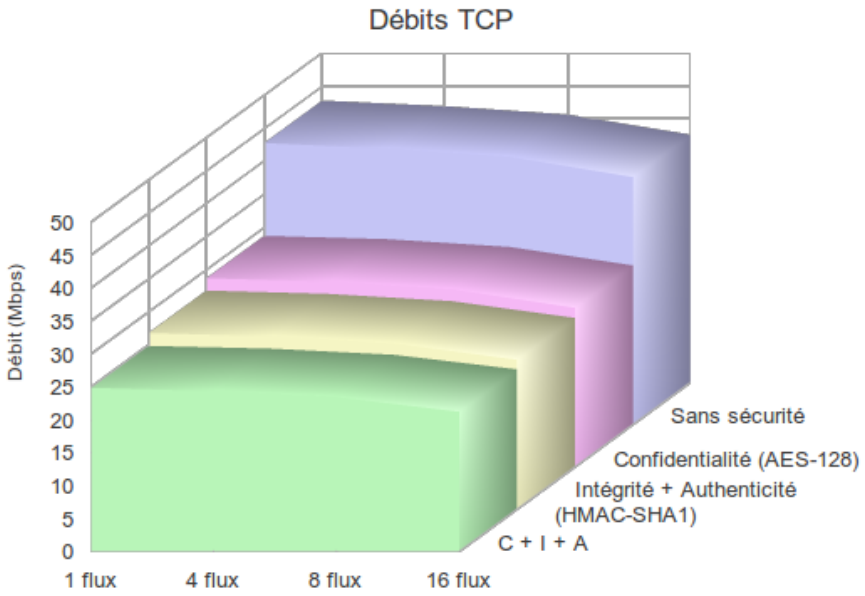


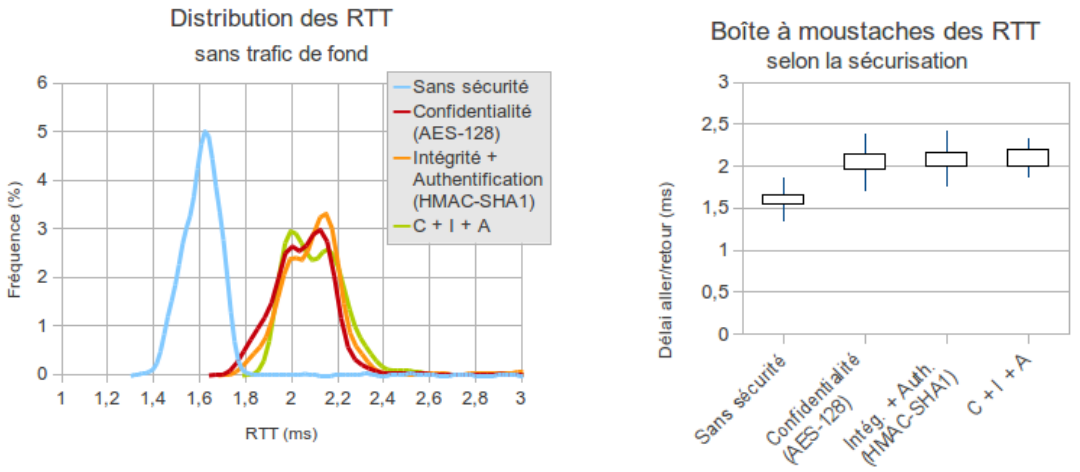

**Figure 6.14.:** Capacité du routeur SNG avec plusieurs flux TCP sécurisés

Avec ce second protocole de transport, TCP, l'augmentation du nombre de flux diminue significativement les performances du routeur SNG. Ainsi, entre les cas extrêmes mesurés de 1 et 16 flux, le débit TCP total diminue de 15%. Cette diminution semble exponentielle sur la courbe car l'échelle du nombre de flux l'est : 1, 4, 8 et 16. Nous n'avons cependant pas suffisamment d'information pour qualifier cette baisse (exponentielle, linéaire, logarithmique, parabolique).

Cette diminution des débits lorsque le nombre de flux augmente s'explique là encore par les mécanismes de régulation de débits de TCP. Ces mécanismes sont les plus efficaces lorsque la connexion TCP est seule à exploiter le chemin. Quand plusieurs connexions cohabitent sur le chemin, elles se dégradent mutuellement, car la congestion engendrée par une connexion TCP impacte aussi toutes les autres connexions, qui réduisent donc toutes leurs débits.

Les mesures de partage de la capacité de chemins réseaux sont l'objet de recherches dans la communauté ; il est ainsi établi que certaines versions de TCP ne sont pas «fair» («équitables»), contrairement à d'autres. La version TCP Cubic que nous avons utilisé a généralement un bon score de «fairness» (mesure d'équitabilité) [Sangtae Ha & Xu 2008].

En TCP comme en UDP, nous pouvons remarquer que la sécurisation est indépendante du multiflux, il y a une homothétie entre les courbes : quelque soit le mécanisme de sécurisation, l'augmentation du nombre de flux tend à diminuer la capacité du routeur SNG, de plus quelque soit le nombre de flux, la sécurisation tend à diminuer la capacité du routeur SNG. Par exemple, les valeurs avec chiffrement correspondent aux valeurs de référence sans sécurité, multipliées par un ratio de 0.70




**(a)** Distribution des RTT **(b)** Boîtes à moustaches des RTT

**Figure 6.15.:** Délais RTT en fonction de la sécurisation

(correspondant à une baisse de 30 %), il en est de même pour les autres modes de sécurisation, à multiplier par 0.65 pour les valeurs avec signature et 0.57 avec signature et chiffrement.

Il est important de noter que toutes ces mesures de débits restent supérieurs aux besoins de l'industrie aéronautique exprimés dans la sous-section 6.3.1.5, le débit minimal mesuré étant obtenu avec 16 flux TCP signés et chiffrés à un débit de 22 Mbps.

#### 6.3.2.3. Distribution des RTT en fonction de la sécurisation

Après nous être intéressé à l'impact de la sécurisation sur la capacité de traitement du routeur SNG et donc sur le débit maximal du chemin entre H1 et H2, nous étudions ici l'impact de la sécurisation sur les délais de traitement du routeur et donc sur les RTT et OWD du chemin H1-H2.

Pour cette expérimentation, nous avons mis en place le trafic nominal de charge (800k bidirectionnel), afin de mesurer des latences dans un scénario le plus réaliste possible, c'est-à-dire le scénario de routeur SNG embarqué avec une charge nominale (exposé dans la section 6.2.3.2).

Le premier graphique présenté sur la figure 6.15a reprend la courbe des fréquences des RTT dans la figure 6.3.1.4, nommée ici «sans sécurité» et servant de référence. Les trois autres courbes sont mesurées avec les services de sécurité cités précédemment : avec chiffrement, avec signature, avec chiffrement et signature de chaque paquet.

Ces trois courbes se superposent, comme le montre aussi les boîtes à moustaches de la figure 6.15b. Les valeurs numériques sont représentées dans la table 6.6.

| Sécurisation | Délai RTT moyen | Ecart-type | Coût de la sécurité sur le RTT |
|---|---|---|---|
| Sans sécurité | 1.60 ms | 0.10 ms | (réf) 100 % |
| Confidentialité (chiffrement) | 2.09 ms | 0.15 ms | 130.6 % |
| Intégrité + Authentification (signature) | 2.12 ms | 0.16 ms | 132.5 % |
| Confidentialité + Intégrité + Auth. | 2.16 ms | 0.15 ms | 135 % |

**Table 6.6.:** Valeurs numériques des RTT mesurés selon la sécurité en charge nominale

Elles confirment que les délais sont bien concentrés autour de 2.12 ms lorsque la sécurisation est activée, quelque soit les services activés. La variation de ces délais est de l'ordre de 150 µs avec sécurisation.

Les calculs sur ces valeurs nous permettent de déduire ici que la tunnellisation impacte de +28% le RTT, le chiffrement représentant +2.5% et la signature +4.5%. Ces ratios sont spécifiques à notre mise en œuvre du routeur SNG.

Rappelons que les valeurs présentes dans la table 6.6 sont des valeurs Aller-Retour via R1 et R2, les requêtes et réponses ICMPv6 ECHO utilisées par notre outil ping6 traversent 4 fois les routeurs SNG. Comme expliqué dans la section 6.3.1.4, il est donc possible de majorer la latence induite par les routeurs en divisant par 4 les valeurs de la table.

#### 6.3.2.4. Distribution de la gigue

Après avoir mesuré les RTT en charge nominale, nous nous sommes intéressé aux cas de réseaux chargés à la limite de saturation et à vide (sans charge).

Les résultats des RTT suivent les tendances logiques auxquelles nous pouvons nous attendre suite à la lecture des sections 6.3.2.3 (RTT en charge nominale), 6.3.1.4 (RTT sans sécurité suivant les 3 profils de charge) et 6.3.2.1 (impact de la sécurisation sur la capacité du routeur).

En effet, les délais RTT sans charge et en charge nominale sont équivalents (ratio de 1), la charge nominale représentant 1.7% de la charge maximale admissible par le routeur. De même, les délais en charge maximale correspondent aux délais sans charge multipliés par un ratio d'environ 8, quelque soit la sécurisation (ou l'absence de sécurité).

Cependant, une autre métrique est intéressante à étudier : la gigue, calculée en étudiant les variations des délais de transmission des paquets. [Demichelis & Chimento 2002] expose différentes définitions associées au terme de

«gigue» et explique ainsi qu'il faut mieux employer les termes de «variation des délais», ce que nous ferons dans la suite de ce manuscrit.

En pratique, nous utiliserons la valeur moyenne de variation de délais, calculée par l'outil ping6, qui approxime cette variation de délais par le calcul de l'écart type des délais RTT mesurés.

En plus de la variation des délais $\sigma$, nous avons aussi calculé le rapport entre cette variation des délais et la moyenne $\bar{x}$ des RTT associés. Les résultats sont présentés dans la table 6.7 pour plus de clarté.

L'augmentation des délais en raison de la charge RTT s'accompagne d'une augmentation des variations des délais, mais cette augmentation des variations est moins régulière (ratio de 2 à 4) et inférieure à l'augmentation des délais (ratio de 8 à 9). Ainsi, si les variations sont non négligeables lorsque le réseau est vide (de l'ordre de 7 %), elles le deviennent lorsque le routeur est saturé (les variations représentent alors de 1 à 3% du délai).

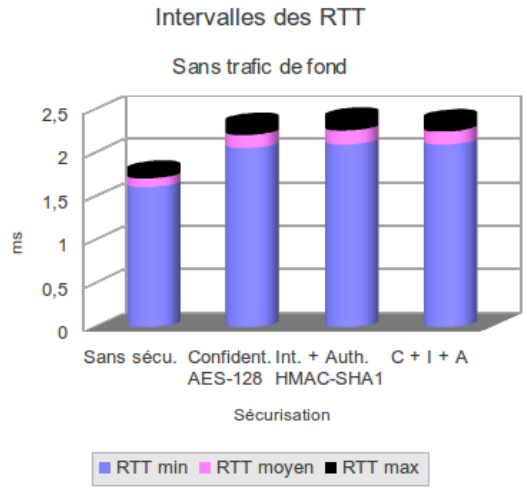
En valeur, les variations de délais RTT restent faibles, comprises entre 100 et 600 µs. L'absence de contrainte explicite de l'industrie aéronautique concernant les variations de délais nous empêchent de statuer sur l'acceptabilité de ces mesures.

Ces tendances sont visibles sur les graphiques des figures 6.16a et 6.16b. En effet sur ces figures, la variation est symbolisée par les deux couvercles posés sur les cylindres des délais RTT minima. Ainsi, pour chaque colonne de cylindres, la partie du dessous représente les délais minima observés, la partie intermédiaire culmine à la valeur moyenne des délais RTT et le haut de la colonne monte jusqu'aux valeurs maximales des délais mesurés.

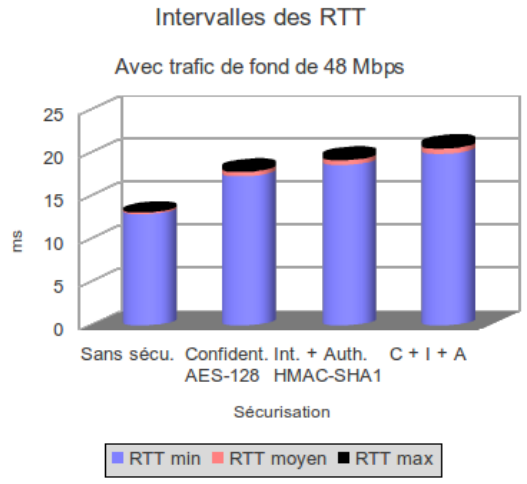
Les couvercles supérieurs apparaissent moins épais sur la figure 6.16b par rapport à la figure 6.16a car sur un réseau chargé la variation des délais est plus petite par rapport aux délais eux-mêmes. Les valeurs sont en fait de 2 à 4 fois plus importantes.

| Sécurisation | | Délai RTT moyen | Variation des délais | Rapport $\sigma/\bar{x}$ | Rapport charge max/à vide | |
|---|---|---|---|---|---|---|
| | | | | | Délais RTT | Variations des délais |
| Sans sécurité | À vide | 1.61 ms | 0.10 ms | 6 % | 8.01 | 2 |
| | charge max | 12.9 ms | 0.20 ms | 1.5 % | | |
| Confidentialité (chiffrement) | À vide | 2.06 ms | 0.15 ms | 7 % | 8.40 | 3.33 |
| | charge max | 17.3 ms | 0.50 ms | 2.9 % | | |
| Intégrité + Auth. (signature) | À vide | 2.10 ms | 0.16 ms | 7 % | 8.86 | 3.44 |
| | charge max | 18.6 ms | 0.55 ms | 3.0 % | | |
| Conf. + Intég. + Auth. | À vide | 2.10 ms | 0.15 ms | 7 % | 9.48 | 4 |
| | charge max | 19.9 ms | 0.60 ms | 3.0 % | | |

**Table 6.7.:** Variation des délais en charge et à vide, selon la sécurité




**(a)** Variabilité sur un réseau sans charge




**(b)** Variabilité à charge maximale

**Figure 6.16.:** Colonnes représentant les délais RTT et la variabilité des mesures

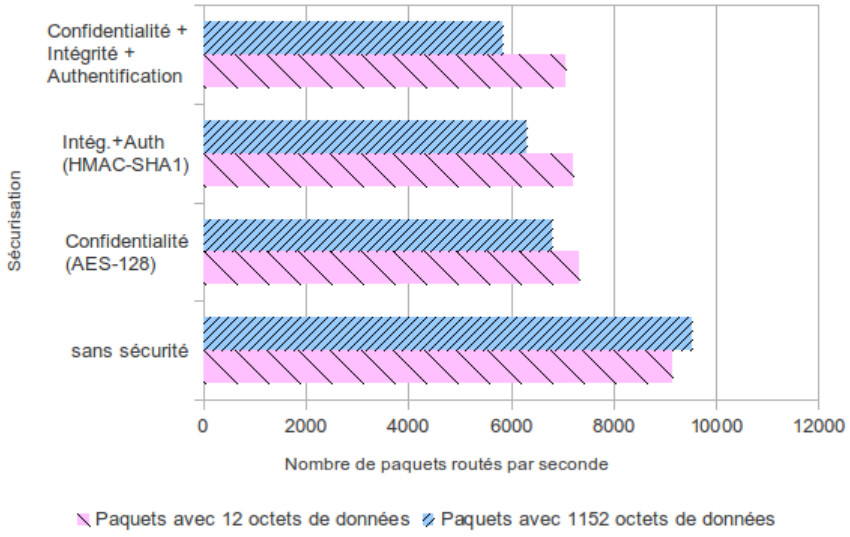



**Figure 6.17.:** Débits (paquets par seconde) selon la sécurisation

#### 6.3.2.5. Impact de la sécurisation sur le nombre de paquets traités par le routeur SNG

Un dernier paramètre étudié est le nombre de paquets par unité de temps qui peuvent être traités par le routeur SNG. Ces mesures sont complémentaires des mesures de débit, car ces dernières sont calculées à l'aide de paquets de taille maximale, donc de la MTU fixée dans notre topologie de la figure 6.4 à 1 500 octets.

Toutefois, nous avons mesuré ici le nombre de paquets non seulement pour des paquets volumineux, mais aussi pour des « petits » paquets. Ainsi, en plus de la variation de la sécurité, nous avons à chaque fois mesuré le nombre de paquets routés par seconde pour des paquets de 12 et 1152 octets de données applicatives encapsulées en UDP, soit des paquets IPv6 de respectivement 60 et 1200 octets (de 1 à 20 fois).

La figure 6.17 montre les résultats des mesures.

L'impact de la taille des paquets est nettement visible sur les six séries sécurisées : la dégradation des performances est d'autant plus forte que les paquets sont gros, la sécurisation impacte peu les petits paquets et beaucoup les gros paquets. Ces mesures nous indiquent que le chiffrement entraîne -7 % de gros paquets traités par rapport à des petits paquets, la signature -13% et la tunnellisation seulement -4 %.

Cela s'explique par le fonctionnement des mécanismes de sécurité : les algorithmes de chiffrement et de signature manipulent toutes les données des paquets, consommant un temps proportionnel à la quantité de données à sécuriser. Par contre, la tunnellisation ne manipule que les entêtes et devrait affecter donc de manière similaire tous les paquets, petits et grands. Mais la tunnellisation entraîne aussi la copie

| [Référence 100% = sans sécurisation] | Débits (octets/seconde) | | Délais RTT | Débits (paquets/seconde) | |
|---|---|---|---|---|---|
| | UDP | TCP | ICMP | UDP (60 o) | UDP (1200 o) |
| Confidentiaté : chiffrement AES-CBC-128 | -7.5% | | -2.5% | -1.6% | -5% |
| Intégrité et Authentification : signature HMAC-SHA1 | -14% | | -4.5% | -2.8% | -10% |
| Tunnellisation : encapsulation avec le protocole ESP | -13% | -22% | -28% | -19% | -23% |
| Coût total | -35% | -43% | -35% | -23% | -38% |

**Table 6.8.:** Synthèse de l'impact de la sécurisation sur les débits, délais RTT et nombre de paquets traités par seconde

de données vers le paquet ESP, elle dépend donc elle aussi de la quantité de données à sécuriser.

Par contre, la comparaison des performances sans et avec sécurisation montre bien une baisse du nombre de paquets routés par seconde, de l'ordre de -23% à -38%.

Il apparaît sur la courbe que, sans sécurité, le routeur SNG route plus de gros paquets par seconde que de petits paquets. Nous soupçonnons la perte de petits paquets d'engendrer de nombreux petits «trous» dans les transmissions, réduisant le taux de remplissage du chemin. Cependant, la précision des horloges de nos systèmes ne nous ont pas permis de prouver ou invalider cette hypothèse.

#### 6.3.2.6. Conclusions sur l'impact de la sécurisation

Afin de synthétiser l'impact selon les métriques utilisées, nous avons choisi de regrouper les résultats principaux dans la table 6.8. Celle-ci montre que si les tendances sont les mêmes quelque soit la métrique, les diminutions des valeurs restent spécifiques à la métrique utilisée. Ainsi, la mise en œuvre par notre routeur SNG de l'algorithme de chiffrement AES-CBC-128 est moins coûteuse en temps de calcul que l'algorithme de signature HMAC-SHA1. La tunnellisation représente par contre toujours la majorité de la consommation de calculs, quelque soit les services de sécurité utilisés.

### 6.3.3. Autres paramètres étudiés

Dans cette dernière partie sont présentés les résultats d'autres expérimentations complémentaires, quantifiant l'impact d'autres paramètres sur les performances du

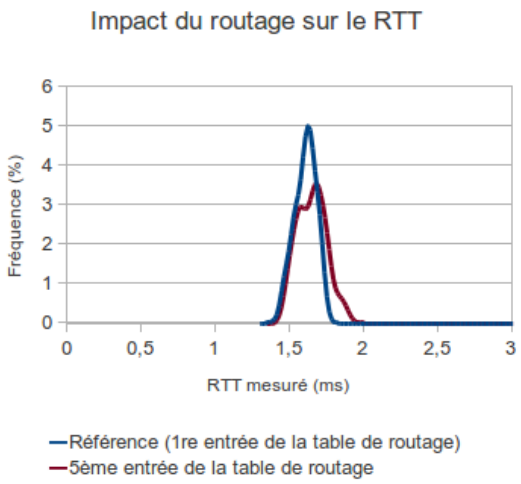



**Figure 6.18.:** Impact du routage sur les délais RTT

routeur SNG.

#### 6.3.3.1. Impact du routage sur le RTT

Le routeur utilise une table de routage statique pour router les paquets, table dont la structure et l'exploitation sont décrites en détails dans la section 4.2.4. L'algorithme utilisé pour router les paquets est un parcours linéaire de la table, i.e. une recherche en temps linéaire.

Nous avons donc utilisé deux configurations, la première définie pour que les paquets de H1 à H2 soient routés par la première entrée de la table, la seconde définie pour utiliser la cinquième (les n°2 à 4 étaient utilisé pour d'autres expérimentations et nous avons préféré mutualiser la configuration pour plusieurs expériences afin de réduire la charge de reconfiguration).

Ensuite, nous avons réalisé les deux séries de mesures et représenté les résultats dans la figure 6.18 montrant les courbes de fréquences des délais RTT. Cette figure montre peu de différences entre les deux courbes, quasiment superposées. Les valeurs numériques indiquent un décalage de 40 µs (la médiane des délais passe de 1,61 à 1,65ms), ce qui nous permet de calculer qu'à chaque ligne supplémentaire lue dans la table de routage pour déterminer la route d'un paquet correspond une augmentation du délai de traitement du paquet par le routeur de 0.4% (cette valeur étant spécifique à notre mise en œuvre et au matériel que nous avons utilisé).

Ces valeurs semblent ici négligeables. Nous ne les avons laissé dans ce mémoire que pour rappeler que la gestion de la table de routage statique n'est algorithmiquement pas optimisée.

Tant que la table contient un «petit nombre» d'entrée, il est inutile de l'optimiser, mais si la table est utilisée avec un «grand nombre» d'entrées, les paquets corres-

pondants aux dernières entrées pourraient retarder significativement le traitement du flot de paquets et ainsi impacter négativement les performances.

Par exemple, en supposant que la dégradation de performances est linéaire (conformément à la mise en œuvre), il est possible d'estimer le délai de traitement de la 101ème entrée de routage : le RTT d'un flux passant par la 101ème entrée serait de 2,61 ms [1], soit 62% d'augmentation par rapport au RTT d'un flux passant par la 1ère entrée.

L'optimisation de la table de routage peut être réalisée par exemple en utilisant un autre algorithme (arbre binaire de recherche ou recherche dichotomique si tous les masques de sous-réseau sont identiques) ou en triant à priori la table de routage statique utilisée par le routeur, en mettant les entrées de routage les plus utilisées en premier et en mettant celles rarement utilisé en dernier.

#### 6.3.3.2. Résultats de capacités TCP et UDP obtenus sur émulateur

Avant d'expérimenter le routeur SNG sur un matériel cible embarqué, nous avons utilisé un émulateur, qemu, qui nous a permis de valider à un stade précoce les fonctionnalités mises en œuvre par le routeur SNG. Le chapitre sur la méthodologie expose les raisons de ce choix, la méthodologie permet en effet de rapidement et facilement passer de l'un à l'autre entre cible réelle et cible émulée.

Les figures de la section 6.3.2.2 présentaient les résultats de mesures de capacité de traitement du routeur SNG (débits maximaux), selon le protocole de transport (TCP ou UDP), le nombre de flux et la sécurisation éventuelle (sans, chiffrement, signature, tout).

Nous avons aussi effectué ces mesures mais sur émulateur. Comme le montre la figure 6.19, les résultats nous avaient semblé peu enthousiasmants : les débits sur émulateur sont compris entre 400 kbps et 1250 kbps, acceptables pour les besoins aéronautiques actuels mais pas pour les besoins futurs, tandis que les débits sur cible réels s'avèrent bien supérieurs, acceptables pour les besoins actuels et futurs.

En plus d'être plus faibles, les performances sur émulateur sont aussi plus irrégulières. Avec UDP, les courbes sont lisses sur cible réelle tandis que sur émulateur, les résultats chutent vertigineusement lorsque le nombre de flux augmente. Une recherche approfondie de la cause de cette chute nous a montré que l'outil employé ici, iperf, est défaillant lorsque les débits sont inférieurs à 500 kbps, rendant inexploitables les résultas avec 8 et 16 flux UDP. Pour les cas à 1 et 4 flux, le ratio de débits cible réelle/cible émulée est de l'ordre de 20 avec le protocole de transport UDP.

Avec le protocole TCP, l'irrégularité des courbes nous empêche de bien montrer l'impact du multiflux sur la cible émulée, mais les données étaient déjà suffisantes

1. 1.61 + ( (1.65-1.61)/4 * 100) = 2.61 ms

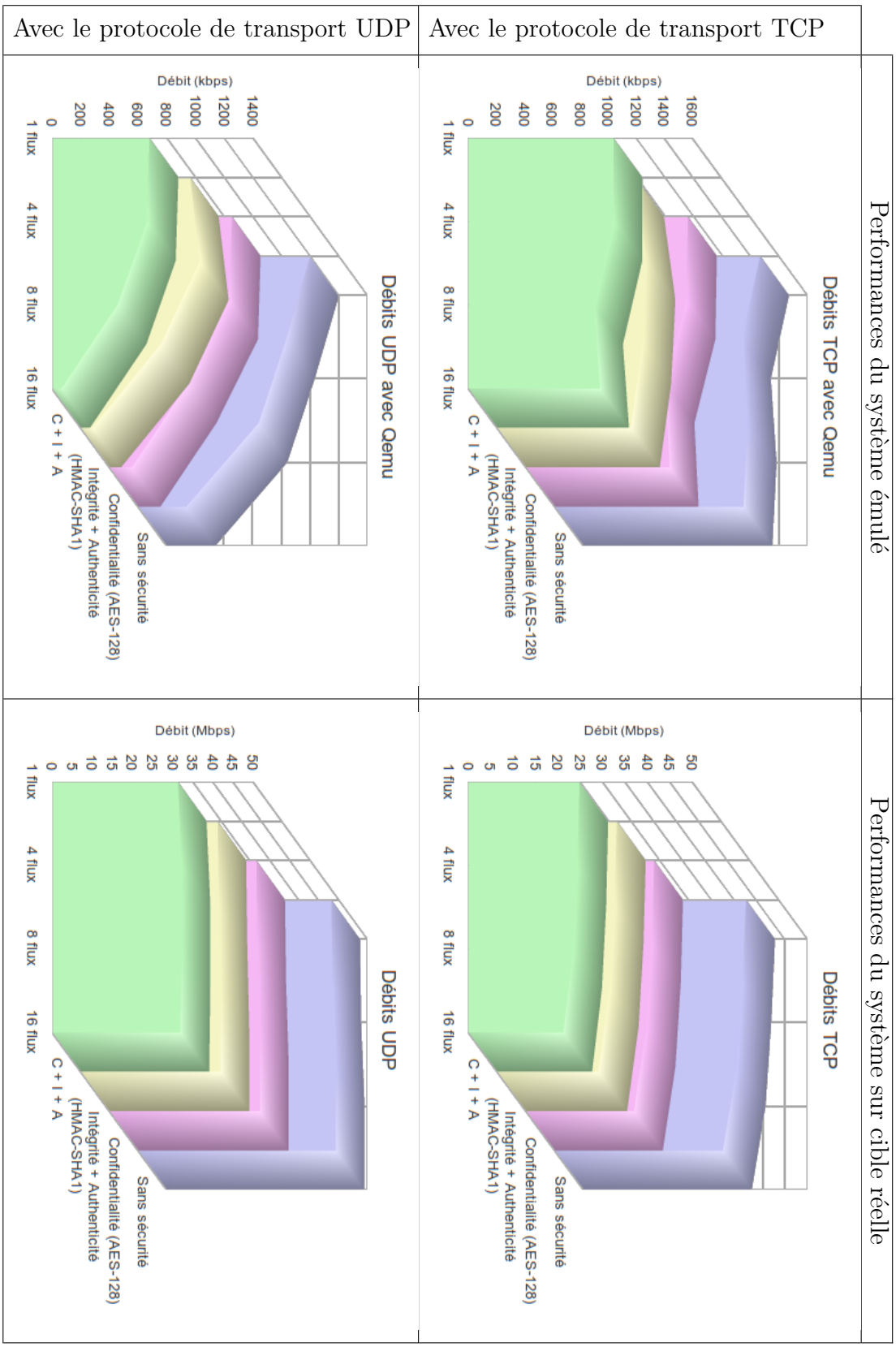



**Figure 6.19.:** Débits avec TCP et UDP obtenus sur émulateur

pour statuer sur l'impact de la sécurisation. Ainsi, avec TCP, les mesures de performances ont permis de qualifier et quantifier l'impact du multiflux et de confirmer celui de la sécurisation. Le ratio de débits entre cibles réelles et émulées est de l'ordre de 40 avec le protocole de transport TCP.

Pour conclure, l'usage de l'émulateur nous a permis de valider et déverminer rapidement les différentes fonctionnalités du routeur SNG, mais l'usage d'une cible matérielle réelle s'est avéré incontournable pour valider les performances du routeur : les résultats préliminaires sur émulateur étaient globalement mauvais et peu encourageants, au contraire des résultats sur cible réelle.

#### 6.3.3.3. Disponibilité du routeur sur une semaine

Un autre élément que nous souhaitions étudier concerne la disponibilité du routeur SNG sur le «long terme». Nous nous sommes donc intéressé au comportement du routeur SNG sur une période d'une semaine. En pratique, les gros porteurs commercialisés sont réinitialisés chaque matin pour la première rotation de la journée, mais l'appareil peut rester exceptionnellement en fonctionnement jusqu'à une centaine d'heures dans le cas des vols long courriers. La durée d'exploitation reste donc inférieure à une semaine, d'où notre choix.

Nous avons donc mis en place la topologie réseau à 2 hôtes et 2 routeurs de la figure 6.4, l'avons configuré pour activer les services de sécurité (chiffrement et signature) et avons généré un trafic de charge de 20 Mbps en UDP, volontairement inférieur au trafic maximal de 22 Mbps que peut atteindre le routeur SNG avec ces paramètres (cf. section 6.3.2.2). Nous avons alors étudié les pertes durant cette expérience de 168 heures.

Sur les 1 004 155 245 paquets émis par H1, 2 685 n'ont pas été reçus par H2. Cela représente donc un taux d'intégrité des communications supérieur à 99.999% (1 - 2.67e-6), conforme aux exigences les plus contraignantes des communications aéronautiques critiques définies par les normes internationales ([ICAO 2006]). Le routeur a toujours été fonctionnel durant cette semaine, soit une disponibilité parfaite.

Notre méthodologie prévoit l'utilisation d'un système compatible IMA (cf. 3.1.4), donc d'un système équipé de mécanismes de détection des fautes logicielles et matérielles et de correction (en redémarrant le système, en le stoppant ou encore en modifiant sa configuration) tels que le gestionnaire de santé « Health Manager » et le chien de garde «watchdog» intégrés au système d'exploitation Sysgo PikeOS que nous utilisons sur nos routeurs SNG. Dans le cas du routeur SNG, ces mécanismes avaient été configurés pour redémarrer automatiquement en cas de défaillance. Cependant, l'absence de panne de notre routeur SNG nous a empêché de valider le fonctionnement de ces mécanismes.

Nous avons tenté d'identifier la source des pertes de paquets mesurées. Cependant, celles-ci semblent aléatoires dans le temps, les délais entre deux pertes sont pour nous imprédictibles.

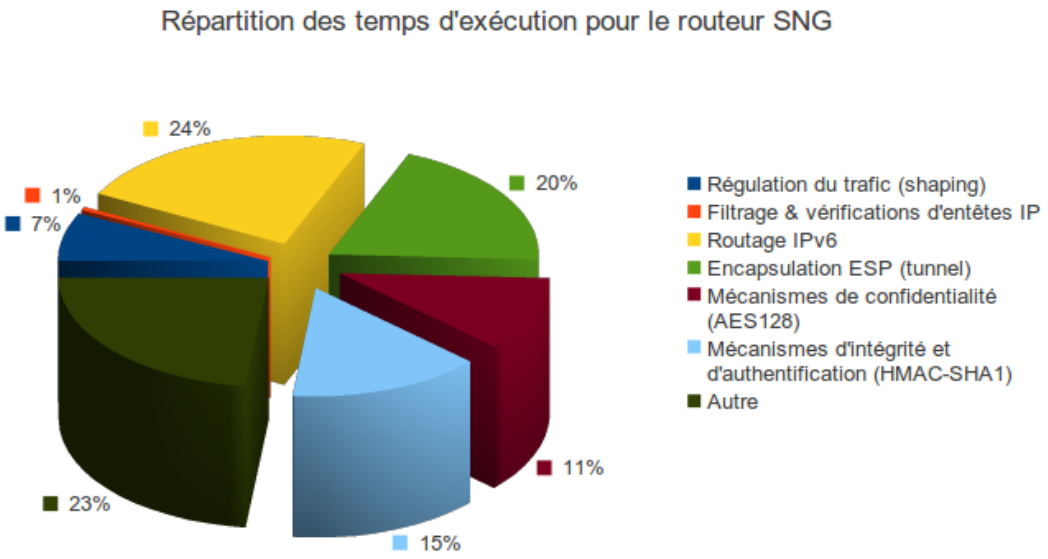



**Figure 6.20.:** Répartition des temps de calcul selon les tâches du routeur SNG

Les sources envisagées sont les deux routeurs SNG (leur logiciel et leur matériel), les liaisons RJ45 catégorie 6 de 3 mètres, les hôtes H1 et H2 (des systèmes Linux Debian). Des mesures sur H1 et H2 nous ont permis de déterminer que le taux de perte de paquets UDP de tailles maximales par ces hôtes d'extrémité était de l'ordre de 1e-8.

Pour pouvoir établir une majoration du taux de perte de paquets par un routeur SNG, nous avons pris comme hypothèse que les liaisons par câble RJ45 étaient parfaites. Cela nous a permis d'établir que le taux de perte de paquets par un seul routeur SNG est majoré par 1.34e-6.

Cependant cette valeur indicative doit être relativisée : elle a été calculé pour une topologie à un seul chemin et à trois sous-réseaux, ne transportant que des données avec des datagrammes UDPv6, les paquets étaient toujours de la taille MTU de 1500 octets et la charge de calcul imposée aux routeurs SNG avoisinait 90% de ses capacités maximales mesurées précédemment.

#### 6.3.3.4. Répartition des temps de calcul selon les fonctions du routeur SNG

Les derniers résultats que nous présenterons concernent la mise en œuvre des fonctionnalités du routeur SNG. La figure 6.20 représente la répartition des temps d'exécution passés par chacune des tâches nécessaires au routage, au filtrage et à la sécurisation des paquets IPv6.

Ce diagramme montre que les fonctionnalités de sécurisation sont les plus consommatrices en temps de calcul, suivi par la fonctionnalité de routage. En effet, pour

sécuriser, il est nécessaire d'encapsuler les données dans des paquets et de les décapsuler, et la tunnellisation représente 20% du temps de calcul. À cela il est nécessaire d'ajouter 11 % pour le chiffrement ou/et 15 % pour la signature. Ainsi la sécurité représente entre 31 et 46 % du temps de calcul.

Le routage IPv6 suit de prêt ces chiffres, il représente 24%. Cela s'explique par le grand nombre de «cas particuliers» à traiter ; par exemple, les adresses de lien et multicast nécessitent un traitement particulier, avec à chaque fois le masquage et la comparaison des 16 octets d'adresses IPv6.

La troisième catégorie «Autre» représente le temps d'exécution consommé par des fonctions très diverses, allant de la réinitialisation de variables temporaires entre deux paquets aux fonctions de résolutions d'adresses de couche liaison (ici les adresses MAC) par notre mise en œuvre du protocole NDP (cf. section 4.3.2).

Dans nos mesures, une seule règle de filtrage était présente, expliquant le poids négligeable de 1% du «filtrage et vérifications d'adresses IP».

Ce diagramme est spécifique à notre mise en œuvre, à nos outils et à notre matériel. Ainsi, utiliser une autre version des transformateurs et compilateurs changerait mécaniquement la répartition des temps de calcul présentée dans cette figure 6.20.

Ce diagramme permet d'orienter les programmeurs pour l'optimisation du code et des modèles afin d'améliorer les performances du routeur SNG.

Ainsi, le poids du chiffrement AES pourrait être réduit en exploitant les instructions assembleur adéquates fournies par notre processeur : la gamme Intel® inclut en effet désormais une dizaine d'instructions pour accélérer les calculs de chiffrement avec l'algorithme AES. Des considérations similaires s'appliquent à l'algorithme HMAC-SHA1 : un support matériel via des instructions spéciales du processeur ou des puces de calcul dédiées pourrait compléter voir remplacer le code actuel en C.

Le routage effectue de nombreux traitements sur les adresses de 16 octets. Ces traitements sont actuellement gérés systématiquement octet par octet par le transformateur qui insère de nombreuses boucles telles que «{ for(i=0;i<16;i++) j[i]=k[i]; }». Le transformateur peut être optimisé pour éviter certaines recopies de variables inutiles (nous en avons identifié un certain nombre durant la lecture manuelle du code source). De même, le code assembleur associé à ces copies peut parfois être optimisé ; ainsi, les processeurs actuels disposent de jeux d'instructions SSE capables de manipuler de manière atomique (et rapide) des variables de 128 bits (donc nos 16 octets d'adresses IPv6).

De plus, le modèle utilisé pour le routage n'est pas optimal : certains traitements pourraient être modélisés avec un autre algorithme, accélérant le traitement de la majorité des paquets au prix de la pénalisation du traitement de certains paquets spéciaux mais plus rares, en réorganisant les différentes conditions dans l'ordre inverse de leur probabilité d'occurrences...

Les mises en œuvre du filtrage et du routage exploitent des tables parcourues linéairement. Ainsi, si la condition validant le paquet est en début de table, alors le routeur

passe rapidement à la phase suivante de traitement. Cependant, si la condition est vers la fin de la table ou n'est pas dans la table, le routeur pourrait consommer un temps non négligeable pour le parcours de ces tables.

Le cas de l'impact du parcours de la table de routage a été traité dans la section 6.3.3.1. Bien que nous n'ayons pas étudié spécifiquement le cas de la table de filtrage, le même algorithme de parcours est utilisé. Les quelques possibilités d'optimisation énoncées dans la section 6.3.3.1 sont donc aussi applicables au filtrage : remplir la table de manière à ce que les conditions les plus souvent vérifiées soient au début et celles les plus rares à la fin, changer l'algorithme de parcours (pour une recherche dichotomique par exemple, quand c'est possible) voir adapter la structure de la table elle-même pour permettre d'utiliser des algorithmes plus efficaces.

Il y a aussi d'autres axes d'optimisations. Ainsi, les communications entre partitions sont utilisées dans notre routeur SNG, nous avons choisi d'utiliser des Files de Messages (Message Queue) par soucis de simplicité, mais d'autres solutions pourraient se révéler plus efficaces : zones mémoires partagées, Sample Queues... De même, lors de l'élaboration de l'architecture (cf. 3.3.1), le choix d'assigner une classe de partition à un processus ou à une partition IMA n'est pas anodin en terme de performances, car les mécanismes d'ordonnancement sont différents et ont donc un impact sur les performances.

Les systèmes des réseaux avioniques sont dimensionnés pour éviter les congestions et nous avons vu que notre routeur SNG valide les exigences de performance. Toutefois, afin d'augmenter les performances de la mise en œuvre du routeur SNG et de traiter une éventuelle congestion, il est possible de travailler sur l'optimisation de l'ordonnancement et du dimensionnement des files d'attentes. La méthodologie de développement permet, de plus, d'ajouter aisément au routeur des mécanismes plus élaborés de Qualité de Service pour limiter les congestions, tel que le mécanisme Random-Early-Detection[Floyd & Jacobson 1993].

Pour résumer les différents vecteurs d'optimisation que nous avons identifié, nous disposons de :
- la modélisation : algorithmes et «astuces d'optimisations», mais aussi choix du langage de modélisation ;
- les outils : transformateurs, compilateurs, assembleurs, lieurs ;
- l'élaboration de l'architecture du logiciel : comment sont traduites les classes de partition sur l'architecture IMA, le choix des IPC;
- le partitionnement : comment sont regroupés les fonctionnalités ;
- le système d'exploitation embarqué ;
- l'écriture de la configuration qui sera utilisée par le système embarqué ;
- le matériel : processeur, coprocesseurs éventuels, mémoires... ;

Nous retrouvons ainsi les étapes de la méthodologie de développement présentée au chapitre 3 ; la boucle est bouclée !

Pour conclure sur l'optimisation des fonctionnalités, il ne faut pas oublier qu'il y a toujours un goulet d'étranglement des performances. L'éliminer implique qu'un

autre endroit deviendra le nouveau goulet d'étranglement, déplaçant ainsi indéfiniment le problème au fur et à mesure des optimisations.

Par contre, l'optimisation conduit souvent à complexifier la mise en œuvre, ce qui peut bloquer d'autres activités autour du développement, telles que la certification du code évoquée à la section 2.4 et son évaluation développée dans la section 2.5.

Il est donc nécessaire de se fixer des objectifs de performance et d'arrêter les itérations d'optimisation lorsque les objectifs sont atteints.

## 6.4. Résumé du chapitre

Nous avons présenté quelques mesures de performances du routeur SNG dans ce chapitre, après avoir listé les métriques intéressantes pour l'évaluation de notre routeur et détaillé les topologies de réseaux que nous avons utilisées. Cette évaluation nous a conduit à établir un profil des trafics avioniques et aéronautiques destinés à transiter par notre routeur, ainsi que de développer un logiciel spécifique pour générer les données à échanger sur le réseau.

Les résultats de nos mesures expérimentales peuvent être résumés par la figure 6.21. Loin d'être exhaustif en raison du grand nombre de paramètres susceptibles d'influer sur les performances de notre routeur SNG embarqué, nous avons tenté de couvrir un maximum de possibilités pour valider les fonctionnalités et surtout permettre d'extraire de nos résultats des tendances importantes, i.e. l'impact de ces paramètres sur les différentes métriques mesurables.

Cependant, la majorité de ces mesures sont spécifiques à notre version du logiciel, issue de nos modélisations des fonctionnalités précédemment décrites dans le chapitre 4. Ces mesures sont aussi spécifiques à notre matériel d'expérimentation, décrit dans la section 6.1.2. Ainsi, comme exposé dans la section précédente, un changement de matériel ou/et des modèles peut fortement impacter les valeurs brutes issues des mesures. Toutefois, les tendances devraient rester, par exemple les débits TCP devraient rester inférieurs aux débits UDP, la sécurisation devrait toujours réduire les performances...

Les performances de notre système sont amplement acceptables vis-à-vis des contraintes aéronautiques industrielles. Ainsi, de notre point de vue, notre produit, le routeur SNG, pourrait accéder aux phases industrielles suivantes de développement d'un produit et quitter le stade de prototype. Il a prouvé que la méthodologie présentée dans le chapitre 3 permet de réaliser rapidement un système complet et fonctionnel.

De plus, ce chapitre a été conçu de manière à guider une seconde étude de performance de routeur. Ainsi, si une seconde version du routeur était mise en œuvre avec par exemple une architecture matérielle différente basée sur un autre processeur, une évaluation des performances suivant les principes exposés dans ce chapitre devrait

permettre de démontrer de manière approfondie l'impact des changements matériels sur les performances. Cela reste cependant une perspective que nous n'avons pas explorée, étant limité par les moyens matériels disponibles.

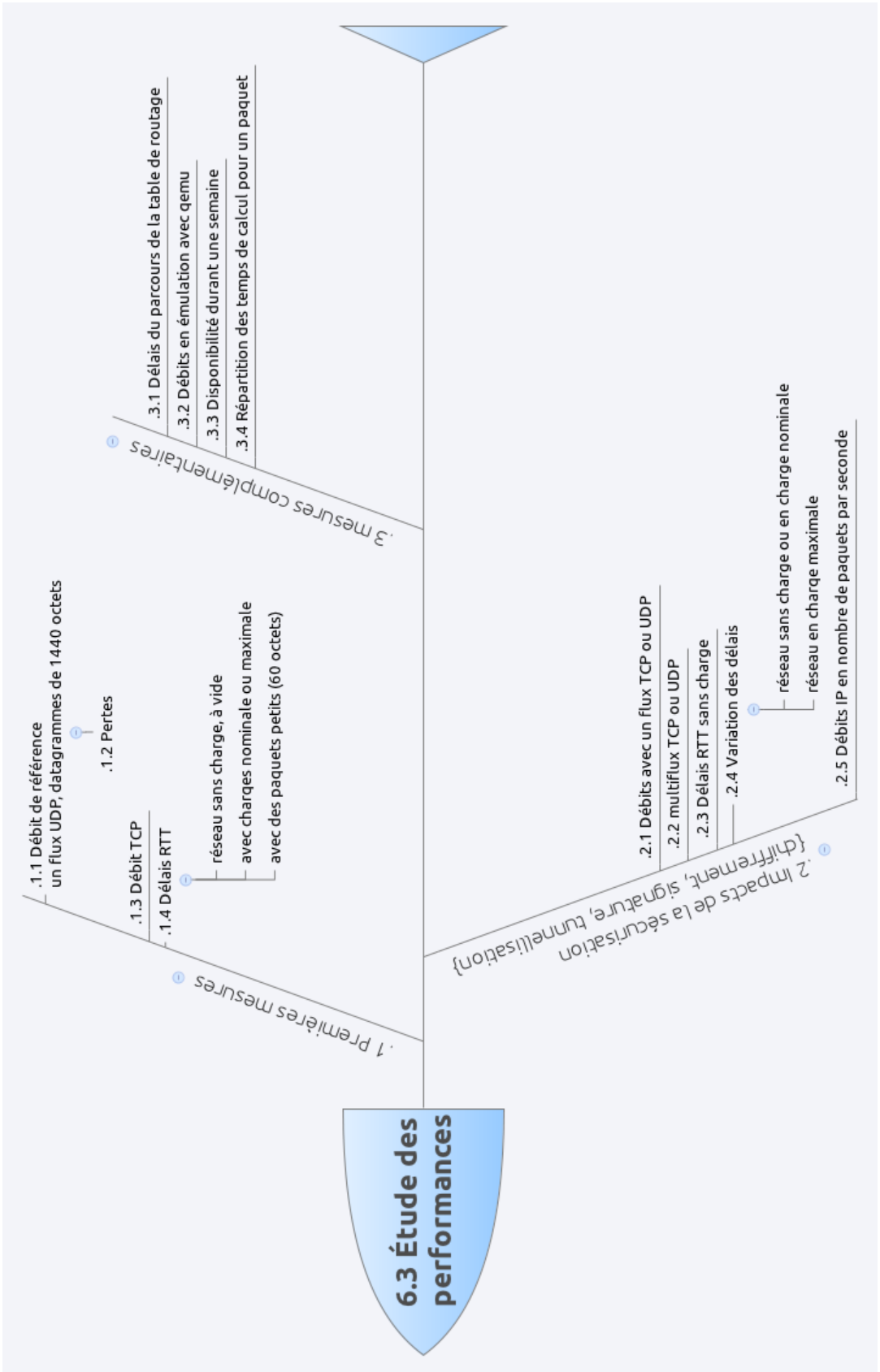



**Figure 6.21.:** Vision globale des mesures effectuées et de la logique qui les organise

# 7. Conclusions et perspectives

## 7.1. Bilan des avancées et des contributions scientifiques

L'objectif initial d'élaboration d'une architecture avionique novatrice mutualisant les différents canaux de communications aéronautiques tout en assurant la sécurité des échanges de données a rapidement mis en exergue le besoin d'un système essentiel des réseaux. Cet élément central ayant guidé les travaux effectués durant cette thèse est un **routeur sûr et sécurisé de nouvelle génération** pour les besoins aéronautiques actuels et futurs, routeur inexistant dans l'ensemble des systèmes avioniques embarqués actuels, jusqu'à aujourd'hui en 2013.

**Contexte contraignant de système sûr et sécurisé** Nous avons commencé dans ce mémoire par la présentation des contextes aéronautiques et avioniques, ainsi que de leurs spécificités. De très nombreux standards et normes sont employés dans l'aéronautique, domaine où les systèmes modernes côtoient des systèmes plus anciens, ainsi par exemple les technologies utilisées dans l'A380 ont été développées entre 1975 et 2005 et elles devront être maintenues jusqu'en 2050.

La sûreté des communications par la technologie est une problématique globalement nouvelle dans le monde de l'aéronautique, les réseaux ayant jusqu'à présent toujours été isolés des autres réseaux publics de communication tel que l'Internet. Cependant, les industriels étudient l'ouverture de ces réseaux aéronautiques, cela posant des questions sur la confidentialité, l'intégrité et l'authentification des échanges de données ainsi que sur la disponibilité des systèmes et des moyens à mettre en œuvre pour garantir ces services. Une phase d'évaluation garantissant l'immunité des systèmes avioniques permet ainsi de s'assurer de la sûreté des systèmes.

Les systèmes avioniques sont critiques : leur défaillance peut menacer de mort les êtres vivants transportés par les avions. Cela impose aux concepteurs des avions d'apporter des garanties quant à l'innocuité des systèmes utilisés, de certifier ces systèmes comme sécurisés.

Ainsi, après avoir présenté les différents réseaux aéronautiques et avioniques, le chapitre 2 a introduit les notions de sûreté [1] et de sécurité [2], nous les avons appliquées

1. La **sûreté** dans les transports a ici le sens du mot anglais «**security**».
2. La **sécurité** dans les transports a ici le sens du mot anglais «**safety**».

durant l'élaboration de notre routeur SNG par l'écriture de deux documents, le Profil de Protection et les spécifications d'exigences logicielles du routeur SNG.

**Réflexions sur le processus de développement** Les nombreuses contraintes du développement de logiciel avionique critique nous ont poussé à réfléchir au processus à adopter dans notre cas de l'élaboration d'un système novateur de type «routeur». Cela nous a conduit à créer une méthodologie de développement et de prototypage, s'étendant de la phase initiale d'écriture des exigences systèmes à la phase de recette, i.e. à la vérification et à la validation du système sur émulateur ou sur système réel.

Exploitant la puissance de la virtualisation, du développement basé sur les modèles, de la sécurité apportée par l'IMA, de la sûreté amenée par les concepts MILS, la méthodologie guide l'équipe de développement jusqu'aux expérimentations, accélérant de nombreuses tâches en automatisant les étapes tout en tenant compte des contraintes d'évaluation et de certification du système final produit, contraintes couramment imposées dans le contexte de logiciels embarqués critiques.

Afin d'être utilisable pour le plus grand nombre d'application et non réservé uniquement au cas de notre logiciel d'un système de réseau informatique, les sept étapes de la méthodologie conservent de la souplesse, par exemple en permettant le choix parmi de nombreux outils différents pour réaliser chacune des étapes, ainsi que les variations nécessaires pour s'adapter aux spécificités de chaque développement logiciel.

D'autres points de la méthodologie complètent les avantages apportés par son application. Ainsi, nous avons créé les concepts de classes de partition et d'instances de partition pour faciliter l'élaboration de l'architecture du logiciel, puis ensuite la réutilisation des modélisations afin de générer rapidement des variantes de logiciel.

L'usage d'outils Open Source apporte encore d'autres avantages lorsqu'ils sont utilisés avec la méthodologie, comme énuméré au chapitre 3.

Enfin, nous avons terminé la phase de réflexion sur le processus de développement en développant «Metrix», un outil qui nous permet de qualifier et de quantifier la qualité des codes sources intermédiaires de la méthodologie.

**Application de la méthodologie au routeur SNG** Après avoir réfléchi sur la méthodologie de développement, nous l'avons appliquée au cas de notre routeur SNG dans le chapitre 4.

Ainsi, nous avons créé des modèles pour les différents protocoles réseaux utilisés ou provisionnés pour l'aéronautique : IPv4 et IPv6 (ordonnancement, routage et filtrage des paquets), IKEv2, NDP, ICMP et ICMPv6 (service ECHO), ESP et ISAKMP ; la figure 7.1 représente ces protocoles mis en œuvre par notre routeur SNG. À notre connaissance, c'est la première fois que des modèles de haut niveau sont utilisés pour générer le code source les protocoles IKEv2 et ESP en plus de permettre leur validation formelle.

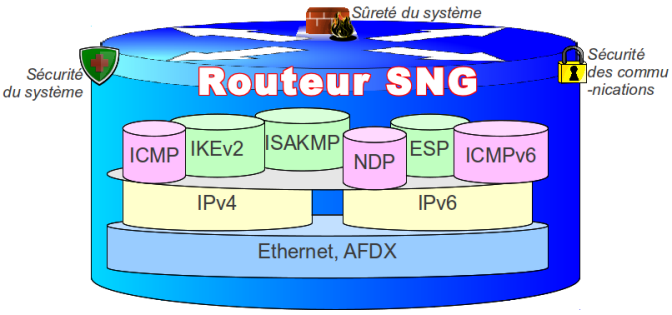



**Figure 7.1.:** Le routeur SNG met en œuvre de nombreux protocoles réseau

Cette mise en œuvre de protocoles s'étend de la modélisation à l'aide de machines à états-transitions des protocoles jusqu'à leurs tests et leur utilisation sur un système réel. Elle s'étend des couches «hautes» de fonctionnalités complémentaires aux mécanismes de routage (par exemple l'établissement de canaux sécurisés de communication à l'aide du protocole IKEv2 et leur utilisation pour sécuriser les paquets IP à l'aide du protocole ESP) jusqu'aux couches «basses» physiques d'échanges de données entre systèmes via des signaux électroniques, en tenant compte des spécificités de notre contexte avionique (tel que l'absence de protocole ARP avec l'AFDX par exemple).

De plus, notre routeur SNG prouve concrètement l'applicabilité de notre méthodologie : cette dernière a été élaborée à l'aide d'idées et de concepts et a été validée en permettant de développer un routeur SNG complet et fonctionnel, disposant de mécanismes de sécurisation des données, fourni avec des informations nécessaires à son évaluation de sûreté et à sa certification de sécurité.

**Complément des travaux sur la sécurité des réseaux : les protocoles SCOUT et SCOUT6** Le routeur SNG met en œuvre les protocoles IKEv2 et ESP pour respectivement établir des canaux de sécurité et utiliser ces canaux. Cependant, le protocole d'établissement (ici IKEv2) a besoin de connaître les extrémités des canaux de communication pour les sécuriser. Cela passe actuellement par la configuration à priori de ces canaux, un administrateur décrivant à l'avance l'ensemble des possibilités de sécurisation.

L'automatisation de cette tâche de communication est apparue comme un objectif à forte valeur ajoutée, surtout dans notre domaine avionique. Ainsi, nous avons développé le protocole SCOUT qui découvre de manière réactive les possibilités de sécurisation et invoque automatiquement le mécanisme d'établissement d'un canal sécurisé adéquat.

Ce protocole SCOUT est générique, il peut potentiellement fonctionner indépendamment du protocole réseau qu'il sécurise, quelque soit ce protocole. Nous l'avons mis en œuvre avec le protocole réseau IPv6, protocole en cours de déploiement destiné à remplacer le protocole IPv4 ; cette mise en œuvre passe par l'exploitation

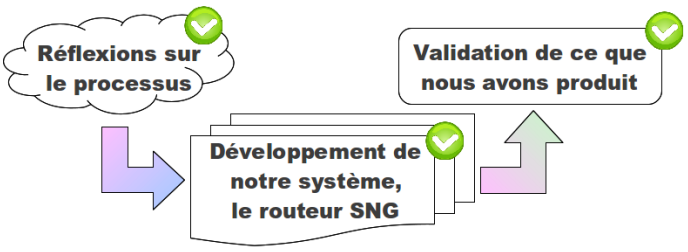



**Figure 7.2.:** Les trois groupes de tâches pour élaborer le routeur SNG

de mécanismes spécifiques à IPv6 et donc nous a conduit à nommer SCOUT6 le protocole SCOUT instancié en IPv6.

Ces protocoles ont la possibilité non seulement de sécuriser les flots de données applicatifs pour les sécuriser, mais en plus ils peuvent être mis en œuvre sur un équipement tiers pour sécuriser les flots transitant par cet équipement intermédiaire. Cette possibilité a été étudiée pour permettre la sécurisation de données lorsque les hôtes communiquant ne peuvent sécuriser eux-mêmes les données qu'ils échangent.

Elle est une contrainte indispensable dans notre contexte avionique, car le coût d'ajout d'un système nouveau assurant la sécurité est inférieur au coût de mise à jour des systèmes existant pour leur ajouter la sécurisation. De plus, si une vulnérabilité sur un protocole ou un algorithme de sécurité est découverte, il est plus facile et moins coûteux de remplacer cet équipement intermédiaire de sécurité que de remplacer l'ensemble des systèmes avioniques hôtes.

Le protocole SCOUT6 a été mis en œuvre et validé expérimentalement sur une topologie réseau composée de systèmes réels Debian Linux permettant de reproduire un comportement plus réaliste que des simulations, voir des émulations de systèmes.

**Évaluation des performances du routeur SNG** La phase de réflexion du processus de développement du routeur SNG du chapitre 3 puis sa mise en œuvre présentée au chapitre 4 est complétée par l'évaluation des performances de notre routeur SNG dans le chapitre 6. Cela suit le schéma logique présenté figure 7.2.

Pour cette phase d'évaluation de performances, nous avons étudié les trafics réseaux pour l'aéronautique et développé un outil logiciel générique générant du trafic réseau, appelé «sourcesonoff» et utilisé pour créer une charge réaliste sur le réseau, correspondante à nos prévisions des communications avioniques et aéronautiques à l'horizon 2020. Nous avons ensuite expérimenté suivant différentes topologies réseau et mesuré différents critères de performances de notre routeur SNG.

Les résultats de notre évaluation des performances sont positifs : les mesures de débits, de délais, de variation des délais et de disponibilités sont acceptables vis-à-vis des contraintes existantes et de celles prévues pour un futur proche dans notre domaine aéronautique, et ce quelque soit le protocole de transport testé (TCP et

UDP), quelque soit la sécurisation mise en œuvre (chiffrement et signature des messages), quelque soit le nombre de flux réseau testés (impacts étudiés pour 1 à 16 flux simultanés), quelque soit la taille des paquets échangés (bornés par 60 octets et une MTU à 1500 octets).

L'évaluation des performances se conclut par un profilage du système mis en œuvre, i.e. par l'étude des blocs fonctionnels du routeur SNG et de la performance de leur mise en œuvre.

## 7.2. Perspectives continuant nos recherches

**Perspectives méthodologiques** À l'issue de cette thèse, le concept d'un routeur SNG en tant que système avionique a été étudié, développé, complété et évalué en laboratoire dans un cadre expérimental. Cela fait de ce projet industriel un projet avancé au niveau TRL 4 («Technology Readiness Level 4»), prêt à passer à la phase TRL 5 d'expérimentation dans un environnement simulant une architecture avionique réelle, avec des flux AFDX etc. Cette nouvelle étape est elle-même une étape intermédiaire entre l'étape actuelle et celle d'utilisation du système final en situation opérationnelle (TRL 8), objectif ultime pour notre système.

Ce travail nous a permis d'approfondir les concepts IMA et MILS, leurs points communs et leurs différences. Notre approche dans notre méthodologie de développement exploite simultanément ces deux concepts, mais ceux-ci restent attachés à deux ensembles de standards et de normes, un pour l'IMA et l'autre pour le MILS, exploités par deux communautés différentes (la communauté aéronautique et la communauté militaire). Pourtant ces deux concepts sont partiellement redondants : bien que leurs objectifs soient différents, la sûreté pour le MILS et la sécurité pour l'IMA, les mécanismes les concrétisant sont similaires. Ainsi, une évolution notable serait la réunification de ces deux concepts, formalisée par l'écriture d'une norme (et non un standard qui est propriétaire et non consensuel).

Notre méthodologie présentée au chapitre 3 est ouverte à l'utilisation de nouveaux outils et de nouveaux langages intermédiaires. Ainsi, nous avions pensé utiliser une chaîne «outils de programmation fonctionnelle» basée sur le langage CaML mais nous n'avons pas approfondi cette piste faute de besoin associé à notre projet du routeur SNG. Pourtant une chaîne fonctionnelle pourrait s'avérer bien plus adaptée que notre chaîne «programmation impérative» pour les projets de recherche basés sur les mathématiques fondamentales et issus du domaine de la recherche opérationnelle.

Notre méthodologie effectue un découpage fort entre les classes de partition et les instances de partition, découpage analogue à une classe d'objet et une instance d'objet en Model-Based Design. Ce découpage a été prévu notamment pour préparer la mutation prochaine d'architectures basées sur des processeurs monocore et gérant de manière séquentielle les fils d'exécution à des architectures futures multicore ou multiprocesseurs, voir massivement multicore (GPGPU), où chaque

instance peut être associée à son cœur. Les systèmes multicore ont montré leur efficacité dans différents domaines, mais le monde aéronautique semble encore frileux à les intégrer en raison de considérations de sûreté de fonctionnement. Les projets de développement de logiciels avioniques ayant suivi notre méthodologie devraient être adaptables facilement et à moindre coût aux futures architectures parallèles dès que les GPGPU seront disponibles sur le marché des plateformes matérielles pour l'aéronautique. Ceci sera rendu possible en évitant de recommencer l'ensemble du processus de développement et de validation du logiciel.

Enfin, nous avons testé la méthodologie sur le routeur SNG en utilisant le langage C. Bien que nous disposions des outils pour utiliser le langage Ada comme langage intermédiaire, nous nous sommes contenté d'évaluer les performances sur une version générée en langage C. Pourtant, le langage Ada présente potentiellement de nombreux avantages, aussi bien en termes de performances et d'optimisation qu'en termes de sûreté et de sécurité de fonctionnement logiciel. Il mériterait aussi un complément d'étude pour évaluer l'impact du langage intermédiaire sur l'application de la méthodologie.

Concernant notre outil «Metrix» d'évaluation de la qualité du code source, son extension aux langages des modèles permettrait d'étendre l'analyse au-delà du langage intermédiaire. Capable actuellement de quantifier différentes métriques sur des codes en langages C et Ada, il pourrait analyser les machines à états-transitions Stateflow, les diagrammes de blocs Simulink et Scicos et ainsi fournir des informations sur la qualité de l'ensemble des langages de la chaîne d'outils appliquée aux développements logiciels.

L'évaluation des performances de notre routeur SNG a validé les contraintes aéronautiques pour lesquelles il a été conçu. Cependant, d'autres domaines plus exigeant pourraient nécessiter d'améliorer ces performances. Le chapitre 6 a énuméré un certain nombre de points pour lesquels des optimisations sont possibles, à toutes les étapes de la méthodologie : meilleurs algorithmes, meilleur usage des possibilités de modélisation, optimisation des dimensionnements des IPC et de l'ordonnancement des instances de partition, amélioration des outils, meilleur usage des possibilités offertes par le support matériel, voir élaboration de processeurs et d'autres matériels optimisés spécifiquement pour améliorer les performances.

**Perspectives fonctionnelles pour le routeur SNG** Le développement du routeur SNG a abouti à un système capable de sécuriser des flux de données et de router des paquets en suivant les informations de configuration définies statiquement à l'installation du routeur SNG dans l'avion. Certains protocoles ont été partiellement mis en œuvre, tel le protocole ICMP dont seule la partie relative au service ECHO est incluse dans le routeur SNG. La mise en œuvre de ces protocoles pourrait être complétée avec par exemple d'autres fonctionnalités du protocole ICMP telles que les messages de réseaux inaccessibles («ICMP unreachable network»). Le routeur pourrait aussi être complété par de nouveaux protocoles pour apporter de nouvelles

fonctionnalités, par exemple le protocole OSPF pour un routage dynamique pour les réseaux non critiques, le protocole IGMP pour gérer les flux multicast. À notre connaissance, la modélisation de ces protocoles par des modèles servant à la vérification formelle du protocole et à la génération de code n'a pas encore été effectuée et serait donc une avancée scientifique pour la communauté réseau.

La sécurité du routeur SNG est actuellement limitée à un seul ensemble d'algorithmes : bien que le routeur SNG mette en œuvre la possibilité de supporter différents algorithmes de sécurité, nous n'avons pas intégré d'algorithme supplémentaire, mais une extension du routeur SNG avec d'autres algorithmes permettrait de s'assurer d'une meilleure compatibilité avec les mises en œuvre d'autres systèmes de sécurité. En plus d'offrir des alternatives aux algorithmes actuellement utilisés, cela permettrait d'envisager d'expérimenter de nouveaux algorithmes novateurs de la cryptographie, dont la mise en œuvre serait dérivée automatiquement de modèles abstraits formellement prouvés avec les plus hauts niveaux d'assurance d'immunité.

La méthodologie permet d'intégrer des classes de partition supplémentaires dédiées à des tâches complexes. Ainsi, il est possible d'adjoindre au routeur SNG la fonctionnalité de «Deep Packet Inspection» pour filtrer de manière plus efficace les flux non critiques. Ces fonctionnalités avancées de filtrage sont actuellement étudiées par les avionneurs et leurs équipementiers, mais leur intégration est ralentie par les contraintes de sûreté et de sécurité ; la méthodologie permet de faciliter cette intégration en garantissant des niveaux élevés d'innocuité et d'immunité et permet donc d'envisager cette intégration à court terme.

Les protocoles SCOUT et SCOUT6 ont été présentés au chapitre 5 avec des perspectives d'évolution, permettant de réduire le nombre de messages échangés et de contourner les problèmes des nœuds non conformes à la spécification IPv6. La validation de ces évolutions reste cependant d'actualité.

De plus, le routeur SNG ne met pas en œuvre le protocole SCOUT, alors que SCOUT a été inventé en tenant compte du contexte contraignant des communications aéronautiques. Il serait intéressant d'effectuer cette intégration de SCOUT avec le routeur SNG et de confirmer (ou d'infirmer) les bénéfices de SCOUT dans la configuration des systèmes aéronautiques, ainsi que les gains et les menaces sur la sécurité des communications aéronautiques.

Une autre perspective qui est en cours d'étude dans notre laboratoire concerne la caractérisation du trafic Internet que nous avons appliquées à notre routeur, pour améliorer encore le réalisme des expérimentations réseau.

> Ce travail de thèse a montré qu'en exploitant un ensemble d'outils judicieusement choisis, il est possible de produire rapidement des logiciels de qualité, répondant à des contraintes fortes de certification et d'évaluation. De plus, les avionneurs peuvent désormais réaliser une architecture avionique avec non seulement les équipements réseaux existants des couches physique (1) et liaison (2), mais aussi exploiter les

possibilités offertes par la couche réseau (3) via le routeur SNG aéronautique. Enfin, le chemin de la modélisation de protocoles de sécurité a franchi un nouveau pas par le double usage d'un même modèle : la validation des protocoles et la génération automatisée de sa mise en œuvre dans des systèmes opérationnels.

Au-delà du travail déjà effectué de l'élaboration méthodologique et technique d'un nouvel équipement avionique embarqué, notre **routeur SNG**, l'ensemble des perspectives issues de ces travaux ouvre la voie à de nouvelles recherches pour créer de nouvelles possibilités et améliorer l'efficacité des systèmes aéronautiques, tout en maintenant les niveaux élevés de sûreté et de sécurité du transport aérien.

# Publications

## Conférences internationales avec comité de lecture

- Mohamed Slim Ben Mahmoud, Nicolas Larrieu, Alain Pirovano, Antoine Varet. **An Adaptative Security Architecture for Future Aircraft Communications,** 29th Digital Avionics Systems Conference (DASC), Salt Lake City, Utah, USA, 2010.
- Antoine Varet, Nicolas Larrieu. **New methodology to develop Certified Safe and Secure Aeronautical Software - an Embedded Router Case Study,** 30th Digital Avionics Systems Conference (DASC), Seattle, Washington, USA, 2011.
- Antoine Varet, Nicolas Larrieu. **Security Capability discovery protocol Over Unsecured IP-based Topology,** 7ème Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information (SAR SSI), Cabourg, France, 2012.
- Antoine Varet, Nicolas Larrieu, Christophe Macabiau. **Design and Development of an Embedded Aeronautical Router With Security Capabilities,** Integrated Communication, Navigation and Surveillance Conference, Washington DC, Colombia, USA, 2012.
- Antoine Varet, Nicolas Larrieu. **Le protocole de découverte de sécurisation SCOUT,** Conférence sur les NOuvelles TEchnologies de la REpartition, Colloque Francophone sur l'Ingénierie des Protocoles, NOTERE/CFIP, Anglet, France, 2012.

## Conférences internationales sans comité de lecture

- Antoine Varet, Nicolas Larrieu. **METRIX: a new tool to evaluate the quality of software source codes,** Conférence InfoTech@Aerospace, Boston, Massachusetts, USA, 2013.

## Conférences nationales

- Antoine VARET. **Un routeur IP embarqué et sécurisé pour l'aéronautique,** 13ème Congrès des Doctorants EdSys, Toulouse, France, 2012.

## Rapports techniques et délivrables industriels

- Antoine VARET. **Software Requirements Specification of the SNG Router**, “SRS RT SNG”, 2011.
- Antoine VARET. **Profil de Protection du Routeur Sûr de Nouvelle Génération**, “PP RT SNG”, 2011.

## Logiciels produits

- Antoine VARET. **Dæmon SCOUT6**, 2012.
  - Code source, documentation et paquets Debian disponibles à http://www.recherche.enac.fr/~avaret/scout6/.
  - Publié sous licences CeCILL-C_v1 et LGPL2.1.
- Antoine VARET. **Metrix, logiciel d’évaluation quantitative de la qualité de codes sources**. 2012.
  - Code source, documentation et paquets Debian disponibles à http://www.recherche.enac.fr/~avaret/metrix/.
  - Publié sous licence GPLv3.
- Antoine VARET. **sourcesonoff, programme de génération de trafic réseau par des sources ON/OFF suivant des distributions mathématiques de durées des flux et d’intervalles entre flux**. 2013.
  - Code source disponible à http://www.recherche.enac.fr/~avaret/sourcesonoff.
  - Publié sous licence GPLv3.

# Liste des symboles

| | |
|---|---|
| AAC | Aeronautical Administrative Communication |
| ACARS | Aircraft Communication Addressing and Reporting System |
| ACD | Aircraft Control Domain |
| AEEC | Airlines Electronic Engineering Committee, of the ARINC company |
| AES | Advanced Encryption Standard |
| AFDX | Avionic Full-DupleX switched Ethernet |
| AH | Authentication Header |
| AISD | Airline Information Service Domain |
| AOC | Aeronautical Operational Control |
| APC | Aeronautical Passenger Communication |
| APEX | Application Executive |
| ARINC | Aeronautical Radio, Incorporated |
| ARP | Address Resolution Protocol |
| ARP (4754) | Aerospace Recommended Practice ARP4754A |
| ATC | Air Traffic Control |
| ATN | Aeronautical Telecommunication Network |
| ATN/IPS | Aeronautical Telecommunication Network with Internet Protocol Suite |
| ATSC | Air Traffic Services Communication |
| AUTH | Authentication |
| Caml | Categorical Abstract Machine Language |
| CC | voir CC ITSEC |
| CC ITSEC | Norme ISO CEI 15408 d'évaluation de la sûreté de systèmes, appelée «Common Criteria for Information Technology Security Evaluation» |
| CEI | voir IEC |

| | |
|---|---|
| COCR | Communications Operating Concepts and Requirements |
| COTS | Commercial Off-The-Shelf |
| CPU | Central Processing Unit |
| Critères Communs | voir CC ITSEC |
| DAL | Design Assurance Level |
| DGAC | Direction Générale de l'Aviation Civile |
| DHCP | Dynamic Host Configuration Protocol |
| DNS | Domain Name System |
| DNSSEC | Domain Name System Security Extensions |
| DOq | Destination Option query (cf. SCOUT6) |
| DOr | Destination Option response (cf. SCOUT6) |
| EAL | Evaluation Assurance Level, défini dans la norme CC ITSEC, de EAL-1 (le plus faible niveau d'assurance) à EAL-7 |
| EAP | Extensible Authentication Protocol |
| EASA | European Aviation Safety Agency, Agence Européenne de la Sécurité Aérienne |
| eLoC | Effective Lines of Code |
| ENAC | École Nationale de l'Aviation Civile |
| ESP | Encapsulating Security Payload |
| EUROCAE | European Organisation for Civil Aviation Equipment |
| FAA | Federal Aviation Administration |
| FAI | Fournisseur d'Accès à l'Internet |
| FIS | Flight Information Services |
| GNU | GNU's Not UNIX |
| Go | Giga-octet |
| GPGPU | General-Purpose computing on Graphics Processing Units |
| GPL | GNU General Public License |
| GRE | Generic Routing Encapsulation |
| HDR | Header |
| HF | High Frequency |
| HFDL | High Frequency Datalink |

| | |
|---|---|
| HMAC | Keyed-Hashing for Message Authentication |
| ICMP | Internet Control Message Protocol |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IFE | In-Flight Entertainment |
| IGMP | Internet Group Management Protocol |
| IHM | Interface Homme-Machine |
| IKE | Internet Key Exchange |
| IMA | Integrated Modular Avionics |
| IP | Internet Protocol |
| IPC | Inter-Process Communications ou Inter-Partition Communications |
| IPsec | Internet Protocol Security |
| IPv4 | Internet Protocol, version 4 |
| IPv6 | Internet Protocol, version 6 |
| ISAKMP | Internet Security Association and Key Management Protocol |
| ISO | Organisation internationale de normalisation - International Organization for Standardization |
| ISO CEI 15408 | Voir CC ITSEC |
| ITSEC | Information Technology Security Evaluation Criteria, cf CC ITSEC |
| KINK | Kerberized Internet Negotiation of Keys |
| LAN | Local Area Network |
| LGPL | Lesser GNU General Public License |
| lLoC | Logical Lines of Code |
| LoC | Lines Of Codes |
| LRU | Line Replacable Unit |
| LSB | Least Significant Bit |
| MAC | Media Access Control |
| MBD | Model-Based Development |
| MILS | Multiple and Independent Levels of Security |

| | |
|---|---|
| MISRA | Motor Industry Software Reliability Association |
| MSB | Most Significant Bit |
| MTU | Maximum Transmit Unit |
| NA | Neighbor Advertisement (NDP) |
| NASA | National Aeronautics and Space Administration |
| NAT | Network Address Translation |
| NDP | Neighbor Discovery Protocol |
| NS | Neighbor Sollicitation (NDP) |
| NS2 | The Network Simulator 2 |
| OACI | Organisation de l'Aviation Civile Internationale |
| OE | Opportunistic encryption |
| OS | Operating System, Système d'Exploitation |
| OSI | Open Systems Interconnection |
| OSPF | Open Shortest Path First |
| OWD | One-Way Delay |
| PC | Personnal Computer |
| PCI, bus PCI | Peripheral Component Interconnect |
| PCIe | Peripheral Component Interconnect Express |
| PE | Protocole d'Etablissement (d'un canal sécurisé) |
| Pfr | Partition de Filtrage et de Routage |
| PIESD | Passenger Information Entertainment Services Domain |
| Pif | Partition d'interfaçage |
| POD, PODD | Passenger-Owned (Devices) Domain |
| POSIX | Portable Operating System Interface |
| PP | Protection Profile, défini dans la norme CC ITSEC |
| PPP | Point-to-Point Protocol |
| PPTP | Point-to-Point Tunneling Protocol |
| Pse | Partition de sécurisation |
| PXE | Preboot eXecution Environment |
| QdS, QoS | Qualité de Service (Quality of Service) |

| | |
|---|---|
| RA | Router Alert (IPv4, IPv6), Router Advertisement (NDP) |
| RAD | Rapid Application Development |
| RAM | Random Access Memory |
| RAq | Router Alert query (cf. SCOUT6) |
| RFC | Request For Comments |
| ROM | Read-Only Memory |
| RS | Router Sollicitation (NDP) |
| RTCA | Radio Technical Commission for Aeronautics |
| RTT | Round-Trip Time |
| SA, SAi, SAr | Security Association (initiator, responder) |
| SADB | Security Association Database |
| SAR | Security Assurances Requirements, défini dans la norme CC ITSEC |
| SATCOM | COMmunications SATellite |
| SC | Special Commity |
| SCOUT | Security Capabilities discovery protocol Over Unsecured network Topologies |
| SCOUT6 | Security Capabilities discovery protocol Over Unsecured network Topologies with Internet Protocol version 6 |
| SE | Système d'Exploitation |
| SFR | Security Functional Requirements, défini dans la norme CC ITSEC |
| SHA | Secure Hash Algorithm |
| SNG | Secure Next Generation |
| SOCKS | SOCKet Secure protocol |
| SPI, SPIi, SPIr | Security Parameter Index (initiator, responder) |
| SRS | Software Requirement Specifications |
| SSE | Streaming SIMD Extensions |
| SSH | Secure SHell |
| SSL | Secure Sockets Layer |
| ST | Security Target, défini dans la norme CC ITSEC |
| TCP | Transmission Control Protocol |

| | |
|---|---|
| TCPv6 | Transmission Control Protocol for IPv6 |
| TFTP | Trivial File Transfer Protocol |
| TLS | Transport Layer Security |
| TRL | Technology Readiness Level |
| TTL | Time To Live |
| UDP | User Datagram Protocol |
| UDPv6 | User Datagram Protocol for IPv6 |
| UIT | Union Internationale des Télécommunications |
| VDL | VHF Datalink |
| VHF | Very High Frequency |
| VM | Virtual Machine, Machine Virtuelle |
| VPN | Virtual Private Network |
| WAN | Wide Area Network |
| WG | Working Group, Groupe de Travail d'une organisation de normalisation |
| XMPP | Extensible Messaging and Presence Protocol |

# Bibliographie


[Aboba *et al.* 2004] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson and H. Levkowetz. *Extensible Authentication Protocol (EAP).* RFC 3748 (Proposed Standard), June 2004. Updated by RFC 5247.

[Abrial 1996] Jean-Raymond Abrial. The b-book, assigning programs to meanings. Cambridge University Press, 1996.

[AdaCore 2013] AdaCore. *GNAT Pro, Ada compiler by AdaCore.* http ://www.adacore.com/, 04 2013.

[Adams *et al.* 2005] M. M. Adams, P. B. Clayton and N. J. Tudor. *CLAWZ : Cost-Effective Formal Verification for Control Systems.* In 26th DASC, pages 10.D.6–1–10.D.6–10, 2005.

[Agrou *et al.* 2011] Hicham Agrou, Pascal Sainrat, Marc Gatti, David Faura and Patrice Toillon. *A design approach for predictable and efficient multi-core processor for avionics.* In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 7D3–1–7D3–11. IEEE, October 2011.

[Ákos Horváth 2010] Dániel V. Ákos Horváth. *Model-Driven Development Of ARINC 653 Configuration Tables.* In Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th, pages 5.A.5–1–5.A.5–115. Budapest University of Technology and Economics Department of Measurement and Information, Budapest, Hungary, GMV, Lisbon, Portugal, 2010.

[Almeida José 2010] Vatrinet F. Almeida José. *Safe and Secure Virtualization : Answers for IMA Next Generation and Beyond (article).* In DASIA. SYSGO, 2010.

[Alshamsi 2005] AbdelNasir Alshamsi. *A Technical Comparison of IPSec and SSL.* AINA '05 Proceedings of the 19th International Conference on Advanced Information Networking and Applications, vol. 2, pages 395–398, 2005.

[Andreasson & Blanc 2008] Oskar Andreasson and Marc Blanc. Iptables Tutorial 1.2.2. Philippe Latu philippe.latu(at)linux-france.org, September 2008.

[Aoun & Davies 2007] C. Aoun and E. Davies. *Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status.* RFC 4966 (Informational), July 2007.

[Arends *et al.* 2005] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose. *DNS Security Introduction and Requirements.* RFC 4033 (Proposed Standard), March 2005. Updated by RFC 6014.

[ARIB 2013] CCSA ETSI TTA TTC ARIB ATIS. *The 3rd Generation Partnership Project*. http ://www.3gpp.org/, 06 2013.

[ARINC429P1-18 2012] Airlines Electronic Engineering Committee (AEEC) ARINC429P1-18. *ARINC Specification 429P1-18 Digital Information Transfer System (DITS), Part 1, Functional Description, Electrical Interfaces, Label Assignments and Word Formats*, 11/2012.

[ARINC429P2-16 2004] Airlines Electronic Engineering Committee (AEEC) ARINC429P2-16. *ARINC Specification 429P2-16 Mark 33 Digital Information Transfer System (DITS), Part 2 - Discrete Data Standards*, 12/2004.

[ARINC429P3-19 2009] Airlines Electronic Engineering Committee (AEEC) ARINC429P3-19. *ARINC Specification 429P3-19 Mark 33 Digital Information Transfer System (DITS) - Part 3 - File Data Transfer Techniques*, 06/2009.

[ARINC429P4 2012] Airlines Electronic Engineering Committee (AEEC) ARINC429P4. *ARINC Specification 429P4 Digital Information Transfer System (DITS), Part 4, Archive of ARINC 429 Supplements*, 11/2012.

[ARINC618-6 2006] Airlines Electronic Engineering Committee (AEEC) ARINC618-6. *ARINC Report 618-6 Air/Ground Character-Oriented Protocol Specification*, 06/2006.

[ARINC619-3 2009] Airlines Electronic Engineering Committee (AEEC) ARINC619-3. *ARINC Report 619-3 ACARS Protocols for Avionic End Systems*, 06/2009.

[ARINC620-7 2012] Airlines Electronic Engineering Committee (AEEC) ARINC620-7. *ARINC Report 620-7 Data Link Ground System Standard and Interface Specification (DGSS/IS)*, 06/2012.

[ARINC622-4 2001] Airlines Electronic Engineering Committee (AEEC) ARINC622-4. *ARINC Report 622-4 ATS Data Link Applications Over ACARS Air-Ground Network*, 10/2001.

[ARINC623-3 2005] Airlines Electronic Engineering Committee (AEEC) ARINC623-3. *ARINC Report 623-3 Character-Oriented Air Traffic Service (ATS) Applications*, 04/2005.

[ARINC629-1 1999] Airlines Electronic Engineering Committee (AEEC) ARINC629-1. *ARINC Report 629-1 Part 1-5 - Multi-Transmitter Data Bus, Part 1-Technical Description*, 03/1999.

[ARINC629-2 1999] Airlines Electronic Engineering Committee (AEEC) ARINC629-2. *ARINC Report 629-2 Part 2-2 Multi-Transmitter Data Bus, Part 2-Application Guide*, 02/1999.

[ARINC653P1-3 2010] Airlines Electronic Engineering Committee (AEEC) ARINC653P1-3. *ARINC Report 653P1-3 Avionics Application Software Standard Interface, Part 1, Required Services*, 11/2010.

[ARINC653P2-2 2012] Airlines Electronic Engineering Committee (AEEC) ARINC653P2-2. *ARINC Report 653P2-2 Avionics Application Software Standard Interface, Part 2, Extended Services*, 06/2012.

[ARINC653P3 2006] Airlines Electronic Engineering Committee (AEEC) ARINC653P3. *ARINC Report 653P3 - Avionics Application Software Standard Interface, Part 3, Conformity Test Specification*, 10/2006.

[ARINC653P4 2012] Airlines Electronic Engineering Committee (AEEC) ARINC653P4. *ARINC Report 653P4 Avionics Application Software Standard Interface, Part 4, Subset Services*, 06/2012.

[ARINC664P1-1 2006] Airlines Electronic Engineering Committee (AEEC) ARINC664P1-1. *ARINC Report 664P1-1 Aircraft Data Network, Part 1, Systems Concepts and Overview*, 06/2006.

[ARINC664P2-2 2009] Airlines Electronic Engineering Committee (AEEC) ARINC664P2-2. *ARINC Report 664P2-2 Aircraft Data Network, Part 2 - Ethernet Physical and Data Link Layer Specification*, 01/2009.

[ARINC664P3-2 2009] Airlines Electronic Engineering Committee (AEEC) ARINC664P3-2. *ARINC Report 664P3-2 Aircraft Data Network, Part 3 - Internet-Based Protocols and Services*, 02/2009.

[ARINC664P4-2 2007] Airlines Electronic Engineering Committee (AEEC) ARINC664P4-2. *ARINC Report 664P4-2 Aircraft Data Network, Part 4 - Internet-Based Address Structure Assigned Numbers*, 12/2007.

[ARINC664P5 2005] Airlines Electronic Engineering Committee (AEEC) ARINC664P5. *ARINC Report 664P5 Aircraft Data Network, Part 5, Network Domain Characteristics and Interconnection*, 04/2005.

[ARINC664P7-1 2009] Airlines Electronic Engineering Committee (AEEC) ARINC664P7-1. *ARINC Report 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*, 09/2009.

[ARINC664P8-1 2010] Airlines Electronic Engineering Committee (AEEC) ARINC664P8-1. *ARINC Report 664P8-1 Aircraft Data Network, Part 8, Interoperation with Non-IP Protocols and Services*, 11/2010.

[ARINC811 2005] Airlines Electronic Engineering Committee (AEEC) ARINC811. *ARINC Standards 811 Commercial Aircraft Information Security Concepts of Operation and Process Framework*, 12/2005.

[ARINC821 2008] Airlines Electronic Engineering Committee (AEEC) ARINC821. *ARINC Standards 821 Aircraft Network Server System (NSS) Functional Definition*, 12/2008.

[Arkko & Pignataro 2009] J. Arkko and C. Pignataro. *IANA Allocation Guidelines for the Address Resolution Protocol (ARP)*. RFC 5494 (Proposed Standard), April 2009.

[ARP4754A 2010] ARP4754A. *Aerospace Recommended Practice (ARP) ARP4754A Guidelines For Development Of Civil Aircraft and Systems*, December 2010.

[Arthur H. Watson 1996] Thomas J. McCabe Arthur H. Watson. *Structured Testing : A Testing Methodology Using the Cyclomatic Complexity Metric.* http ://www.mccabe.com/iq.htm, 1996.

[Aryoba 2013] Aryoba. *Cisco Equipment Performance (per pps and Mbps).* http ://www.dslreports.com/faq/13434, 03 2013.

[Aubry 2010] Claude Aubry. Scrum, le guide de la méthode agile la plus populaire. Dunod, 2010.

[Bagnulo *et al.* 2011] M. Bagnulo, P. Matthews and I. van Beijnum. *Stateful NAT64 : Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers.* RFC 6146 (Proposed Standard), April 2011.

[Baker 2011] Kirk Baker. *Filling the FAA guidance and policy gap for systems integration and safety assurance.* In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 1B4–1–1B4–4. IEEE, October 2011.

[Bellard 2013] Fabrice Bellard. *QEMU, logiciel libre et générique d'émulation de processeur et de machine virtuelle.* http ://www.qemu.org/, 04 2013.

[Ben Mahmoud 2012] Mohamed S. Ben Mahmoud. *Addressing Security Challenges in Emerging Data-based Aeronautical Communications.* PhD thesis, Institut National des Sciences Appliquées de Toulouse (INSA Toulouse), February 2012.

[Besse 2013] Frédéric Besse. *Réseaux ad hoc aéronautiques.* Phd thesis, Ecole Nationale de l'Aviation Civile (ENAC), Toulouse, 02 2013.

[Besson 2011] Maxime Besson. Virtualisation et cloud. Open Source Solutions, April, Ploss, 2011.

[Bhatt *et al.* 2005] D. Bhatt, B. Hall, S. Dajani-Brown, S. Hickman and M. Paulitsch. *Model-Based Development and the Implications to Design Assurance and Certification.* In Digital Avionics Systems Conference, DASC 2005. The 24th, pages 10.D.3–1–10.D.3–13, 2005.

[Boeing 2013] Boeing. *Boeing 777 Commercial Airplanes (official website).* http ://www.777.newairplane.com/, 06 2013.

[Bonnet 2008] Lucas Bonnet. Livre blanc sur la virtualisation. Bearstech, 2008.

[Busser *et al.* 2004] R. D. Busser, M. R. Blackburn, A. M. Nauman and T. R. Morgan. *Reducing cost of high integrity systems through model-based testing.* In Digital Avionics Systems Conference, 2004. DASC 04. The 23rd, volume 2, pages 6.B.1–61–13, 2004.

[byBoe 2013] The Boeing Company byBoe. *Connexion-by-Boeing.* http ://fr.wikipedia.org/wiki/Connexion_by_Boeing, 02 2013.

[Canu *et al.* 2011] Antoine Canu, David Faura, Patrice Toillon, Marc Gatti and Philippe Benabes. *A versatile input interface for avionic computers.* In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 7A3–1–7A3–11. IEEE, October 2011.

[cc1 2009] cc1. *Common Criteria for Information Technology Security Evaluation Part 1 : Introduction and general model*, July 2009.

[cc2 2009] cc2. *Common Criteria for Information Technology Security Evaluation Part 2 : Security functional components*, July 2009.

[cc3 2009] cc3. *Common Criteria for Information Technology Security Evaluation Part 3 : Security assurance components*, July 2009.

[cc4 2009] cc4. *Common Methodology for Information Technology Security Evaluation Evaluation methodology*, July 2009.

[Cheshire 2008] S. Cheshire. *IPv4 Address Conflict Detection*. RFC 5227 (Proposed Standard), July 2008.

[Cisco 2009] Cisco. *Portable Product Sheets - Routing Performance*. http ://www.cisco.com/web/partners/downloads/765/tools/quickreference/routerperformance.pdf, 11 2009.

[Clarke *et al.* 1996] Edmund M. Clarke, Jeannette M. Wing and Et Al. *Formal Methods : State of the Art and Future Directions*. ACM Computing Surveys, vol. 28, page 4, December 1996.

[ClearSy 2013] ClearSy. *Atelier-B*. http ://www.atelierb.eu/, 04 2013.

[CNES 2009] CNES. *SATCOM for ATM : Estimation of Capacity Required for AMS(R)S Communications Around 2020 Over European Area*, 07 2009.

[Combs 2013a] Gerald Combs. *dumpcap - Dump network traffic*. http ://www.wireshark.org/docs/man-pages/dumpcap.html, 06 2013.

[Combs 2013b] Gerald Combs. *Wireshark, Go Deep*. http ://www.wireshark.org/, 01 2013.

[Comm. 2013] Comm. *OMNet++ Network Simulation Framework website*. http ://www.omnetpp.org/, 01 2013.

[Constantine Dovrolis 2006] Manish Jain Constantine Dovrolis. *Pathload, a measurement tool for the available bandwidth of network paths*. http ://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bw-est/pathload.html, 03 2006.

[Conta *et al.* 2006] A. Conta, S. Deering and M. Gupta. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. RFC 4443 (Draft Standard), March 2006. Updated by RFC 4884.

[da Silva Stanisce Correa *et al.* 2011] Guilherme C. da Silva Stanisce Correa, Adilson M. da Cunha, Luiz A. Dias and Osamu Saotome. *A comparison between automated generated code tools using model based development*. In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 7E4–1–7E4–9. IEEE, October 2011.

[De Cerchio & Riley 2011] Raymond De Cerchio and Chris Riley. *Aircraft systems cyber security*. In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 1C3–1–1C3–7. IEEE, October 2011.

[Deering & Hinden 1995] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 1883 (Proposed Standard), December 1995. Obsoleted by RFC 2460.

[Deering & Hinden 1998] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[Delio 2004] Michelle Delio. *Linux : Fewer Bugs Than Rivals*. http ://www.wired.com/software/coolapps/news/2004/12/66022, December 2004.

[Demichelis & Chimento 2002] C. Demichelis and P. Chimento. *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*. RFC 3393 (Proposed Standard), November 2002.

[Descartes 1637] René Descartes. Discours de la méthode - pour bien conduire sa raison et chercher la vérité dans les sciences. imprimerie Jan Maire, 1637.

[Despres 2010] R. Despres. *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)*. RFC 5569 (Informational), January 2010.

[Dia 2013] Dia. *Dia, éditeur libre de diagrammes*. http ://live.gnome.org/Dia/, 04 2013.

[Dierks & Rescorla 2008] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[Digia 1995] Digia. *The Qt development framework*. http ://qt-project.org/, 1995.

[DO-136 1968] SC-110 DO-136. Universal air-ground digital communication system standards. Radio Technical Commission for Aeronautics, 3/7/1968.

[DO-163 1976] SC-131 DO-163. Minimum performance standards-airborne hf radio communications transmitting and receiving equipment operating within the radio-frequency range of 1.5 to 30 megahertz. Radio Technical Commission for Aeronautics, 03/19/1976.

[DO-169 1979] SC-140 DO-169. Vhf air-ground communication technology and spectrum utilization. Radio Technical Commission for Aeronautics, 7/20/1979.

[DO-178 1982] SC-167 DO-178. Software considerations in airborne systems and equipment certification. Radio Technical Commission for Aeronautics, 1982.

[DO-178A 1986] SC-167 DO-178A. Software considerations in airborne systems and equipment certification. Radio Technical Commission for Aeronautics, 8/1986.

[DO-178B 1992] SC-167 DO-178B. Software considerations in airborne systems and equipment certification. Radio Technical Commission for Aeronautics, 12/1/1992.

[DO-178C 2011] SC-205 DO-178C. Software considerations in airborne systems and equipment certification. Radio Technical Commission for Aeronautics, 12/13/2011.

[DO-248B 2001] SC-190 DO-248B. Final annual report for clarification of do-178b software considerations in airborne systems and equipment certification. Radio Technical Commission for Aeronautics, 10/12/2001.

[DO-248C 2012] SC-205 DO-248C. Supporting information for do-178c and do-278a. Radio Technical Commission for Aeronautics, 12/13/2012.

[DO-254 2000] SC-180 DO-254. Design assurance guidance for airborne electronic hardware. Radio Technical Commission for Aeronautics, 4/19/2000.

[DO-265 2000] SC-188 DO-265. Minimum operational performance standards for aeronautical mobile high frequency data link (hfdl). Radio Technical Commission for Aeronautics, 12/14/2000.

[DO-281A 2005] SC-172 DO-281A. Minimum operational performance standards for aircraft vdl mode 2 physical, link and network layer. Radio Technical Commission for Aeronautics, 11/08/2005.

[DO-281B 2012] SC-214 DO-281B. Minimum operational performance standards (mops) for aircraft vdl mode 2 physical link and network layer. Radio Technical Commission for Aeronautics, 03/21/2012.

[DO-297 2005] SC-200 DO-297. Integrated modular avionics (ima) development guidance and certification considerations. Radio Technical Commission for Aeronautics, 11/08/2005.

[DO-326 2010] SC-216 DO-326. Airworthiness security process specification. Radio Technical Commission for Aeronautics, 12/08/2010.

[DO-330 2011] SC-205 DO-330. Software tool qualification considerations. Radio Technical Commission for Aeronautics, 12/13/2011.

[DO-331 2011] SC-205 DO-331. Model-based development and verification supplement to do-178c and do-278a. Radio Technical Commission for Aeronautics, 12/13/2011.

[DO-332 2011] SC-205 DO-332. Object-oriented technology and related techniques supplement to do-178c and do-278a. Radio Technical Commission for Aeronautics, 12/13/2011.

[DO-333 2011] SC-205 DO-333. Formal methods supplement to do-178c and do-278a. Radio Technical Commission for Aeronautics, 12/13/2011.

[Dovrolis & Jain 2002] Constantine Dovrolis and Manish Jain. *Pathload, a measurement tool for end-to-end available bandwidth.* In PAM, 01 2002.

[Droms 1997] R. Droms. *Dynamic Host Configuration Protocol.* RFC 2131 (Draft Standard), March 1997. Updated by RFCs 3396, 4361, 5494, 6842.

[Dubrawsky 2003] Ido Dubrawsky. *Firewall Evolution - Deep Packet Inspection.* Security Focus, vol. 7, no. 1716, page 16, 08 2003.

http ://www.symantec.com/connect/articles/firewall-evolution-deep-packet-inspection.

[EASA 2013] EASA. *European Aviation Safety Agency - Agence Européenne de la Sécurité Aérienne - Europaeische Agentur fuer Flugsicherheit.* http ://www.easa.europa.eu/, February 2013.

[eclipse 2013] eclipse. *eclipse, an open development platform comprised of extensible frameworks, tools and runtimes for building.* http ://www.eclipse.org/, 04 2013.

[FAA 2007] FAA. *Special Conditions : Boeing Model 787-8 Airplane ; Systems and Data Networks Security-Isolation or Protection From Unauthorized Passenger Domain Systems Access.* https ://www.federalregister.gov/articles/2007/04/13/E7-7065/special-conditions-boeing-model-787-8-airplane-systems-and-data-networks-security-isolation-or, 04 2007.

[Farkhani & Razzazi 2006] T.R. Farkhani and M.R. Razzazi. *Examination and Classification of Security Requirements of Software Systems.* In Information and Communication Technologies, 2006. ICTTA '06. 2nd, volume 2, pages 2778–2783, 2006.

[Floyd & Jacobson 1993] Sally Floyd and Van Jacobson. *Random Early Detection (RED) gateways for Congestion Avoidance.* In IEEE/ACM, editeur, IEEE/ACM Transactions on Networking, volume 1, pages 397–413. IEEE, ACM, 08 1993.

[Foundation 2011] Linux Foundation. *The Network Emulation webpage.* http ://www.linuxfoundation.org/collaborate/workgroups/networking/netem, 12 2011.

[Frankel *et al.* 2003] S. Frankel, R. Glenn and S. Kelly. *The AES-CBC Cipher Algorithm and Its Use with IPsec.* RFC 3602 (Proposed Standard), September 2003.

[Freier *et al.* 2011] A. Freier, P. Karlton and P. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0.* RFC 6101 (Historic), August 2011.

[(FSF) 2007] Free Software Foundation (FSF). *Licence publique générale GNU.* http ://www.gnu.org/licenses/gpl-3.0.html, 2007.

[Gallet & Brinton 2011] Bart Gallet and Chris Brinton. *Massively parallel processing for dynamic airspace configuration.* In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 3A5–1–3A5–9. IEEE, October 2011.

[Gaska *et al.* 2010] Thomas Gaska, Brian Werner and David Flagg. *Applying virtualization to avionics systems - The integration challenges.* In Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th, pages 5.E.1–1–5.E.1–19. IEEE, October 2010.

[GCC 2013] GCC. *gcc, the GNU C Compiler.* http ://gcc.gnu.org/, 04 2013.

[Ghafari 2010] Naghmeh Ghafari. *Applying Formal Methods to Airborne Software*. In 29th DASC proceedings. 29th DASC, 29th DASC, October 2010.

[GogoInf 2013] Gogo LLC (Aircell) GogoInf. *Gogo Inflight Internet*. http ://www.gogoair.com, 02 2013.

[Graphviz 2000] Graphviz. *Graph Visualization Software*. http ://graphviz.org/, 2000.

[Gérard Huet 1985] Ascánder Suárez-Pierre Weis Michel Mauny Gérard Huet Guy Cousineau. *The Categorical Abstract Machine Language (CaML)*, 1985.

[Haley *et al.* 2008] Charles Haley, Robin Laney, Jonathan Moffett and Bashar Nuseibeh. *Security Requirements Engineering : A Framework for Representation and Analysis*. IEEE Transactions on Software Engineering, vol. 34, no. 1, pages 133–153, 2008.

[Hamzeh *et al.* 1999] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little and G. Zorn. *Point-to-Point Tunneling Protocol (PPTP)*. RFC 2637 (Informational), July 1999.

[Hanks *et al.* 1994] S. Hanks, T. Li, D. Farinacci and P. Traina. *Generic Routing Encapsulation (GRE)*. RFC 1701 (Informational), October 1994.

[Harkins & Carrel 1998] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*. RFC 2409 (Proposed Standard), November 1998. Obsoleted by RFC 4306, updated by RFC 4109.

[Hearn 2004] J. Hearn. *Does the common criteria paradigm have a future ?* IEEE Security & Privacy Magazine, vol. 2, no. 1, pages 64–65, January 2004.

[Huitema 2006] C. Huitema. *Teredo : Tunneling IPv6 over UDP through Network Address Translations (NATs)*. RFC 4380 (Proposed Standard), February 2006. Updated by RFCs 5991, 6081.

[Huyck 2010] Patrick Huyck. *SKPP Conformance - Activities and Considerations to Achieve Certification*. In 2010 IEEE/AIAA 29th Digital Avionics Systems Conference, pages 4.A.6–1–4.A.6–8, October 2010.

[Ian Land 2009] Jeff Elliott Ian Land. *Architecting ARINC 664, Part 7 (AFDX) Solutions*. Rapport technique, Xilinx - XAPP1130 (v1.0.1), 05 2009.

[IANA 2012] IANA. *Internet Assigned Numbers Authority website*. http ://www.iana.org, 03 2012.

[IBM 2013] IBM. *IBM Rational DOORS*. http ://www-01.ibm.com/software/awdtools/doors/productline/, 04 2013.

[ICAO 2006] EUROCONTROL/FAA ICAO. *Communications Operating Concept and Requirements (COCR) for the Future Radio System, version 2.0*. http ://legacy.icao.int/anb/panels/acp/repositoryf, 03 2006.

[Ichbiah 1983] Jean Ichbiah. *The Ada Language*. http ://www.adaic.com/, 1983.

[IEEE1233 1998] IEEE1233. *Guide for Developing System Requirements Specifications Std 1233*. Rapport technique, Software Engineering Standards Committee and the IEEE Computer Society, Institute of Electrical and Electronics Engineers, 1998.

[IEEE830 1998] IEEE830. *Recommended Practice for Software Requirements Specifications Std 830*. Rapport technique, Software Engineering Standards Committee and the IEEE Computer Society, Institute of Electrical and Electronics Engineers, 1998.

[Inc 2008] Wind Rivers Inc. *ARINC 653 : An Avionics Standard for Safe, Partitioned Systems*. electronic, June 2008.

[Ingham 2002] Stephanie Ingham Kenneth ; Forrest. *A History and Survey of Network Firewalls*. http ://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf, 2002.

[Inmarsat 2013] Inmarsat. *Site web d'Inmarsat - The mobile satellite company(TM)*. http ://www.inmarsat.com/, 06 2013.

[Innotek 2011] Innotek. *Oracle VM VirtualBox*. http ://www.virtualbox.org/, 12 2011.

[INRIA 1996] Jérôme Vouillon Damien Doligez Didier Rémy INRIA Xavier Leroy. *Objective Caml*. http ://caml.inria.fr/, 1996.

[INRIA 2013] Metalau Project INRIA. *Scicos, simulateur graphique de systèmes dynamiques*. http ://www.scicos.org/, 04 2013.

[Intel & Systemsoft 1999] Intel and Systemsoft. *Preboot eXecution Environment (PXE)*. http ://download.intel.com/design/archives/wfm/downloads/pxespec.pdf, 1999.

[Iridium 2013] Iridium. *Site relatif à la constellation "Iridium" de satellites, pour l'aéronautique*. http ://www.iridium.com/IridiumConnected/IridiumAtWork/Aviation.aspx, 06 2013.

[iso iec 2008] iso iec. *ISO/IEC 27005 Information technology - Security Techniques - Information security risk management*. Rapport technique, ISO, June 2008.

[ISO7498-1 1994] ISO7498-1. *ISO/IEC 7498-1 Open Systems Interconnection (OSI) model*, 1994.

[Izerrouken *et al.* 2008] N. Izerrouken, X. Thirioux, M. Pantel and M. Strecker. *Certifying an Automated Code Generator Using Formal Tools : Preliminary experiments in the GeneAuto project.* In IRIT, University of Toulouse, France, 2008.

[Jaeger 2010] Eric Jaeger. *Study of the Benefits of Using Deductive Formal Methods for Secure Developments*. PhD thesis, LIP6, 104 boulevard Kennedy, 75016, March 2010.

[Jankiewicz *et al.* 2011] E. Jankiewicz, J. Loughney and T. Narten. *IPv6 Node Requirements*. RFC 6434 (Informational), December 2011.

[Jean-Louis Bénard 2004] Régis Médina Dominic Williams Jean-Louis Bénard Laurent Bossavit. L'extreme programming. Eyrolles, 2004.

[Jim Alves-Foss & Taylor 2005] Paul Oman Jim Alves-Foss W. Scott Harrison and Carol Taylor. *The MILS Architecture for High-Assurance Embedded Systems.* International Journal of Embedded Systems, vol. 1, pages 1–9, 2005.

[Jobredeaux *et al.* 2011] Romain Jobredeaux, Timothy E. Wang and Eric M. Feron. *Autocoding control software with proofs I : Annotation translation.* In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 7C1–1–7C1–13. IEEE, October 2011.

[Joyce 2010] Jeffry Joyce. *Formal Methods in RTCA, DO-178C.* In 29th DASC proceedings. 29th DASC, 29th DASC, October 2010.

[Kaufman *et al.* 2010] C. Kaufman, P. Hoffman, Y. Nir and P. Eronen. *Internet Key Exchange Protocol Version 2 (IKEv2).* RFC 5996 (Proposed Standard), September 2010. Updated by RFC 5998.

[Kaufman 2005] C. Kaufman. *Internet Key Exchange (IKEv2) Protocol.* RFC 4306 (Proposed Standard), December 2005. Obsoleted by RFC 5996, updated by RFC 5282.

[Keblawi & Sullivan 2006] F. Keblawi and D. Sullivan. *Applying the common criteria in systems engineering.* IEEE Security & Privacy Magazine, vol. 4, no. 2, pages 50–55, March 2006.

[Keir Fraser 2003] Ian Pratt The Xen Project XenSource Inc. Keir Fraser Steven Hand. *Xen, hypervisor.* http ://www.xen.org/, 2003.

[Kenneth Hess 2010] Amy Newman Kenneth Hess. Virtualisation en pratique. Collection Reference, 1re édition. Collection Référence, April 2010.

[Kent & Seo 2005] S. Kent and K. Seo. *Security Architecture for the Internet Protocol.* RFC 4301 (Proposed Standard), December 2005. Updated by RFC 6040.

[Kent 2005a] S. Kent. *IP Authentication Header.* RFC 4302 (Proposed Standard), December 2005.

[Kent 2005b] S. Kent. *IP Encapsulating Security Payload (ESP).* RFC 4303 (Proposed Standard), December 2005.

[Kerberos 2011] Kerberos. *Website Kerberos : The Network Authentication Protocol.* http ://web.mit.edu/kerberos/, 12 2011.

[Kerren & Jusufi 2009] Andreas Kerren and Ilir Jusufi. *Novel Visual Representations for Software Metrics Using 3D and Animation.* In SOFTWARE ENGINEERING - WORKSHOPBAND, 2009.

[Kivinen *et al.* 2005] T. Kivinen, B. Swander, A. Huttunen and V. Volpe. *Negotiation of NAT-Traversal in the IKE.* RFC 3947 (Proposed Standard), January 2005.

[Kocher 1996] Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems, volume 1109. Springer Berlin / Heidelberg, 1996. http ://dx.doi.org/10.1007

[Labs 2009a] Trusted Labs. *Profil de Protection Routeur avec Elément De Confiance Embarqué v4.0*. Rapport technique, FR/PM/SGDN/ANSSI, March 2009.

[Labs 2009b] Trusted Labs. *Rapport de certification ANSSI-CC-PP-2009/03 Profil de Protection Routeur avec élément de confiance embarqué (référence PP RECE, version 4.0)*. Rapport technique, FR/PM/SGDN/ANSSI, March 2009.

[Leech *et al.* 1996] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas and L. Jones. *SOCKS Protocol Version 5*. RFC 1928 (Proposed Standard), March 1996.

[Leiner & Schlager 2007] Bernhard Leiner and Martin Schlager. *A comparison of Partitioning Operating Systems for Integrated Systems*. In Springer-Verlag B. Heidelberg, editeur, Springer, 2007.

[Leland W. E. & V. 1994] Willinger W. Leland W. E. Taqqu S. M. and Wilson D. V. *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 2, NO. 1, 02 1994.

[Leroy 2006] Xavier Leroy. *Formal Certification of a Compiler Back-end*. In ACM CE. ACM, January 2006.

[LibreOffice 2013] LibreOffice. *Communauté LibreOffice francophone, LibreOffice*. fr.libreoffice.org/, 04 2013.

[Linux 2000] Linux. *Linux Kernel Coding Style*. https ://www.kernel.org/doc/Documentation/CodingStyle, 2000.

[LLC 2012] M Squared Technologies LLC. *Metrics Definitions*. http ://msquared-technologies.com/m2rsm/docs/rsm_metrics_narration.htm, Jun. 2012.

[Madson & Glenn 1998] C. Madson and R. Glenn. *The Use of HMAC-SHA-1-96 within ESP and AH*. RFC 2404 (Proposed Standard), November 1998.

[Mankins 1995] John C. Mankins. *Technology Readiness Levels : A White Paper*. Rapport technique, NASA, Office of Space Access and Technology, Advanced Concepts Office., 06 1995.

[Marcil 2011] Luc Marcil. *MBD code generation : A cost-effective way to speed up HMI certification*. In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 8B1–1–8B1–10. IEEE, October 2011.

[Mark Gates 2013] Jim Ferguson Jon Dugan Feng Qin Kevin Gibbs John Estabrook Mark Gates Ajay Tirumala. *Iperf, a modern alternative for measuring TCP and UDP bandwidth performance*. http ://iperf.fr/, 06 2013.

[Martin Pinzger 2005] Michele Lanza Martin Pinzger Harald Gall. *Visualizing Multiple Evolution Metrics*, 2005.

[MathWorks 2013a] The MathWorks. *Matlab R2012b*. http ://www.mathworks.fr/, 04 2013.

[MathWorks 2013b] The MathWorks. *Simulink, logiciel de modélisation système multiphysique*. http ://www.mathworks.com/products/simulink/, 04 2013.

[MathWorks 2013c] The MathWorks. *Stateflow, logiciel de modélisation de diagrammes à états*. http ://www.mathworks.com/products/stateflow ?s_cid=wiki_stateflow_2, 04 2013.

[Matolak *et al.* 2011] David W. Matolak, Qiong Wu, Juan J. Sanchez-Sanchez and M. C. Aguayo-Torres. *LTE performance in the airport surface area channel*. In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 4D3–1–4D3–7. IEEE, October 2011.

[May & Woods 1979] T.C. May and M.H. Woods. *Alpha-particle-induced soft errors in dynamic memories*. IEEE Trans Electron Devices, vol. 2, no. ED-26, page 0, 1979.

[Mellado *et al.* 2006] Daniel Mellado, Eduardo Fernández-Medina and Mario Piattini. *A Comparative Study of Proposals for Establishing Security Requirements for the Development of Secure Information Systems*. In Computational Science and Its Applications - ICCSA 2006, volume 3982 of *Lecture Notes in Computer Science*, chapitre 109, pages 1044–1053. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006.

[Microsoft 2005] Microsoft. *Internet Key Exchange*. http ://technet.microsoft.com/en-us/library/cc78499401 2005.

[MIL1553 1975] MIL1553. *Command/Response Multiplex Data Bus, revision 1975/04/30*, 1975.

[MIL1553B 1978] MIL1553B. *Command/Response Multiplex Data Bus, revision 1978/09/21*, 1978.

[MIL1553w 2013] MIL1553w. *The MIL-STD-1553 official website*. http ://www.milstd1553.com/resources-2/history-of-mil-std-1553/, 06 2013.

[MISRA-C 1998] MISRA-C. *Motor Industry Software Reliability Association*. http ://www.embedded.com/columns/beginerscorner/9900659 ?_requestid =427335, 1998.

[Morimoto & Cheng 2005] S. Morimoto and Jingde Cheng. *Patterning Protection Profiles by UML for Security Specifications*. In Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, volume 2, pages 946–951, 2005.

[Motré & Téri 2000] Stéphanie Motré and Corinne Téri. *Using Formal and Semi-formal Methods for a Common Criteria Evaluation*. In Eurosmart, Marseille (France), June 2000.

[Narten *et al.* 2007] T. Narten, E. Nordmark, W. Simpson and H. Soliman. *Neighbor Discovery for IP version 6 (IPv6).* RFC 4861 (Draft Standard), September 2007. Updated by RFC 5942.

[Nilsson 2005] Ulf Nilsson. *Static Analysis methods and tools.* by Linköping University, May 2005.

[NIST 2001] FEDERAL INFORMATION PROCESSING STANDARDS Publication NIST. *ADVANCED ENCRYPTION STANDARD (AES).* http ://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 11 2001.

[NIST 2008] FEDERAL INFORMATION PROCESSING STANDARDS Publication 198-1 NIST. *The Keyed-Hash Message Authentication Code (HMAC).* http ://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf, 07 2008.

[NIST 2012] FEDERAL INFORMATION PROCESSING STANDARDS Publication NIST. *Specifications for the Secure Hash Standard (SHS).* http ://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf, 03 2012.

[OACI ] OACI. *Annexe 10 à la Convention relative à l'aviation civile internationale - Télécommunications aéronautiques.*

[Oi 2013] The ARINC Company Oi. *GLOBALink Onboard Internet data sheet.* http ://www.arinc.com/downloads/product_collateral/globalink-oi-datasheet.pdf, 02 2013.

[Olive *et al.* 2006] M. L. Olive, R. T. Oishi and S. Arentz. *Commercial Aircraft Information Security-an Overview of ARINC Report 811.* In 2006 IEEE/AIAA 25th Digital Avionics Systems Conference, pages 1–12, October 2006.

[Olivier & N.Beameur 2001] P. Olivier and N.Beameur. *Flow level ip traffic characterization.* ITC 17, Salvador, Brazil, 12 2001.

[OpenSWAN 2012] OpenSWAN. *The OpenSWAN project website.* https ://www.openswan.org/projects/openswan/, 03 2012.

[Partridge & Jackson 1999] C. Partridge and A. Jackson. *IPv6 Router Alert Option.* RFC 2711 (Proposed Standard), October 1999. Updated by RFC 6398.

[Piper 1998] D. Piper. *The Internet IP Security Domain of Interpretation for ISAKMP.* RFC 2407 (Proposed Standard), November 1998. Obsoleted by RFC 4306.

[Plummer 1982] D. Plummer. *Ethernet Address Resolution Protocol : Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware.* RFC 826 (INTERNET STANDARD), November 1982. Updated by RFCs 5227, 5494.

[Polgar 2011] B. Polgar. *Model-based Integration, Execution and Certi cation of Development Tool-chains.* In Formal Methods for Automation and Safety in Railway and Automotive Systems, pages pp 227–235. Department of Measurement and Information Systems, Budapest University of Technology and Economics Magyar Tudosok krt. 2, Budapest, Hungary, H-1117, 2011.

[Postel 1980] J. Postel. *User Datagram Protocol*. RFC 768 (INTERNET STANDARD), August 1980.

[Postel 1981a] J. Postel. *Internet Control Message Protocol*. RFC 792 (INTERNET STANDARD), September 1981. Updated by RFCs 950, 4884, 6633.

[Postel 1981b] J. Postel. *Internet Protocol*. RFC 791 (INTERNET STANDARD), September 1981. Updated by RFCs 1349, 2474.

[Postel 1981c] J. Postel. *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD), September 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[Rayadurgam & Heimdahl 2001] S. Rayadurgam and M. P. E. Heimdahl. *Coverage based test-case generation using model checkers*. In ECBS, pages 83–91, 2001.

[Richardson & Redelmeier 2005] M. Richardson and D.H. Redelmeier. *Opportunistic Encryption using the Internet Key Exchange (IKE)*. RFC 4322 (Informational), December 2005.

[Rieger 2011] Gerhard Rieger. *Multipurpose relay (SOcket CAT), Relay for bidirectional data transfer between two independent data channels*. http ://stackoverflow.com/tags/socat/info, 12 2011.

[Ritchie 1972] Dennis Ritchie. *The C Language*, 1972.

[Rugina & Dalbin 2010] A. E. Rugina and J. C. Dalbin. *Experiences with the GENE-AUTO Code Generator in the Aerospace Industry*. In Astrium+Airbus, 2010.

[Rugina *et al.* 2008] Ana-Elena Rugina, Dave Thomas, Xavier Olive and Guillaume Veran. *Gene-Auto : Automatic Software Code Generation for Real-Time Embedded Systems*. In S. ASTRIUM Satellites SA and Thales A. Space, editeurs, DASIA, 2008.

[Rushby 1981] J. M. Rushby. *Design and verification of secure systems*. SIGOPS Oper. Syst. Rev., vol. 15, no. 5, pages 12–21, December 1981.

[Saint-Andre 2011] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP) : Core*. RFC 6120 (Proposed Standard), March 2011.

[Sakane *et al.* 2006] S. Sakane, K. Kamada, M. Thomas and J. Vilhuber. *Kerberized Internet Negotiation of Keys (KINK)*. RFC 4430 (Proposed Standard), March 2006.

[Sandeep Bajaj 2013] Deborah Estrin Sandeep Bajaj Lee Breslau. *The Network Simulator - ns-2,*. http ://www.isi.edu/nsnam/ns/, 01 2013.

[Sangtae Ha & Xu 2007] Injong Rhee Sangtae Ha Long Le and Lisong Xu. *Impact of Background Traffic on Performance of High-speed TCP Variant Protocols*. In Computer Networks : The International Journal of Computer and Telecommunications Networking, volume 15, pages 852–865, 2007.

[Sangtae Ha & Xu 2008] Injong Rhee Sangtae Ha and Lisong Xu. *CUBIC : A New TCP-Friendly High-Speed TCP Variant*. In ACM SIGOPS, editeur, ACM SIGOPS Operating System Review, volume 42, pages 64–74. ACM SIGOPS, ACM, 07 2008.

[Savola & Patel 2004] P. Savola and C. Patel. *Security Considerations for 6to4*. RFC 3964 (Informational), December 2004.

[SCADE 2013] SCADE. *Software Critical Application Development Environment (SCADE), a lustre-based IDE generating C-code*. http ://www.esterel-technologies.com/products/scade-system/, 04 2013.

[Schoen 2007] Seth Schoen. *Detecting Packet Injection : A Guide To Observing Packet Spoofing by ISPs*. https ://www.eff.org/sites/default/files/packet_injection_0.pdf, 11 2007.

[Seward 2002] Julian Seward. *Valgrind, a GPL licensed programming tool for memory debugging, memory leak detection, and profiling*. www.valgrind.org, 2002.

[Silberschatz 2010] Peter B. ; Gagne-Greg Silberschatz Abraham ; Galvin. *Process Scheduling : 5.3.4 Round Robin Scheduling*. Operating System Concepts (8th ed.), vol. 1, page 194, 2010.

[Singh *et al.* 2010] H. Singh, W. Beebee and E. Nordmark. *IPv6 Subnet Model : The Relationship between Links and Subnet Prefixes*. RFC 5942 (Proposed Standard), July 2010.

[Skaves 2011] Peter Skaves. *Cyber security issues related to aircraft systems, paper #260*. In 2011 IEEE/AIAA 30th Digital Avionics Systems Conference, pages 1–35. IEEE, October 2011.

[Smith 2013] Brett Smith. *Guide rapide de la GPLv3*. http ://www.gnu.org/licenses/quick-guide-gplv3.fr.html, 02 2013.

[Sollins 1992] K. Sollins. *The TFTP Protocol (Revision 2)*. RFC 1350 (INTERNET STANDARD), July 1992. Updated by RFCs 1782, 1783, 1784, 1785, 2347, 2348, 2349.

[Srisuresh & Egevang 2001] P. Srisuresh and K. Egevang. *Traditional IP Network Address Translator (Traditional NAT)*. RFC 3022 (Informational), January 2001.

[Steffen Gebert 2012] Daniel Schlosser-Klaus Heck Steffen Gebert Rastin Pries. *Internet Access Traffic Measurement Analysis*. http ://tma2012.ftw.at/TMA/pres/TMA2012-talk3.pdf, 03 2012.

[Stephens 2004] B. Stephens. *Security architecture for aeronautical networks*. In The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576), volume 2, pages 8.E.2–81–19. IEEE, 2004.

[StrongSWAN 2012] StrongSWAN. *Website of Strongswan, the Open Source IPsec-based VPN Solution*. http ://www.strongswan.org/, 03 2012.

[Stroustrup 1983] Bjarne Stroustrup. *The C++ Language*, 1983.

[Subbiah & Nagaraj 2003] S. Subbiah and S. Nagaraj. *Issues with object orientation in verifying safety-critical systems*. In Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pages 99–104, 2003.

[Swift64 2013] SatCom Direct Swift64. *Swift 64 Overview*. https ://www.satcomdirect.com/main/aviation/swift-64/default.aspx, 02 2013.

[SwiftBB 2013] SatCom Direct SwiftBB. *SwiftBroadband Overview*. https ://www.satcomdirect.com/main/aviation/swiftbroadband-swift-broadband-sbb/, 02 2013.

[Sysgo 2010] Sysgo. *PikeOS, Safe and Secure Virtualization*. http ://www.sysgo.com/products/pikeos-rtos-and-virtualization-concept/, 2010.

[Sysgo 2012] Sysgo. *CODEO, Eclipse based IDE*. http ://www.sysgo.com/products/pikeos-rtos-and-virtualization-concept/eclipse-based-codeo/, 2012.

[Thaler *et al.* 2010] D. Thaler, S. Krishnan and J. Hoagland. *Teredo Security Updates*. RFC 5991 (Proposed Standard), September 2010.

[Thomas Graf 2011] Remco van Mook Thomas Graf Greg Maxwell. *Linux Advanced Routing and Traffic Control*. http ://lartc.org/, 07 2011.

[Tonu Naks IB 2010] K. Tonu Naks IB. *Gene-Auto : Automatic Software Code Generation for Real-Time Embedded Systems D2.55 Gene-Auto installation and usage manual*, June 2010.

[Toom *et al.* 2008] A. Toom, T. Naks, M. Pantel, M. Gandriau and Indrawati. *Gene-Auto : an Automatic Code Generator for a safe subset ofSimulink/Stateflow and Scicos*. In Akadeemia IB Krates OÜ, University of Toulouse IRIT-ENSEEIHT, Alyotech F. CRIL Technologies and Tallinn University of Technology, editeurs, ERTS, 2008.

[Toom *et al.* 2010] A. Toom, N. Izerrouken, T. Naks, M. Pantel and Ssi Yan Kai. *Towards Reliable Code Generation with an Open Tool : Evolutions of the Gene-Auto toolset*. In Institute of Cybernetics at Tallinn University of Technology, Université Irit-Enseeiht, S. Continental Automotive France SA, Mäealuse IB Krates OÜ and Department of Computer Control of Tallinn University of Technology, editeurs, ERTS, 2010.

[Toshiyuki Asahi & Schneiderman 2005] David Turo Toshiyuki Asahi and Ben Schneiderman. *Vizual decision-making : Using treemaps for the Analytic Hierarchy Process*. In ACM SIGCHI, Special Interest Group on Computer-Human Interaction, 2005.

[Tschofenig *et al.* 2008] H. Tschofenig, D. Kroeselberg, A. Pashalidis, Y. Ohba and F. Bersani. *The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method*. RFC 5106 (Experimental), February 2008.

[Turner 2000] Aaron Turner. *tcpreplay - Replay network traffic stored in pcap files*. http ://linux.die.net/man/1/tcpreplay, 03 2000.

[Uchenick 2009a] Gordon M. Uchenick. *Introduction to Security for Integrated Modular Avionics*. In 28th DASC proceedings. 28th DASC, 28th DASC, October 2009.

[Uchenick 2009b] Gordon M. Uchenick. *Multiple Independant Levels of Safety and Security : High Assurance Architecture for Integrated Modular Systems*. In 28th DASC proceedings. 28th DASC, 28th DASC, October 2009.

[uit 2013] uit. *UIT : Engagée à connecter le monde - site web officiel*. http ://www.itu.int/fr/, 06 2013.

[USA 2007] NSA/IA Director USA. *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness (SKPP)*. http ://www.niap-ccevs.org/cc-scheme/pp/pp.cfm/id/pp_skpp_hr_v1.03/, 06 2007.

[Van Jacobson & McCanne 2013] Craig Leres Van Jacobson and Steven McCanne. *tcpdump - dump traffic on a network*. http ://www.tcpdump.org/, 06 2013.

[Varet & Larrieu 2011] Antoine Varet and Nicolas Larrieu. *New methodology to develop Certified Safe and Secure Aeronautical Software - an Embedded Router Case Study*. In 30th Digital Avionics Systems Conference Proceedings. IEEE, 10 2011.

[Varet & Larrieu 2012a] Antoine Varet and Nicolas Larrieu. *Le protocole de découverte de sécurisation SCOUT*. In Conférence Francophone de l'Ingénierie des Protocoles 2012, 05 2012.

[Varet & Larrieu 2012b] Antoine Varet and Nicolas Larrieu. *Security Capability discovery protocol Over Unsecured IP-based Topology*. In SAR SSI 2012 7ème Conférence sur la Sécurité des Architectures Réseaux et Systèmes d'Information, 10 2012.

[Varet & Larrieu 2013] Antoine Varet and Nicolas Larrieu. *METRIX : a new tool to evaluate the quality of software source codes*. In AIAA Infotech@Aerospace 2013 Conference Proceedings. AIAA, 08 2013.

[Varet *et al.* 2012] Antoine Varet, Nicolas Larrieu and Christophe Macabiau. *Design and Development of an Embedded Aeronautical Router With Security Capabilities*. In Integrated Communication, Navigation and Surveillance Conference Proceedings. IEEE, 04 2012.

[Verifysoft 2012] Verifysoft Technology GmbH Verifysoft. *Halstead Metrics*. http ://www.verifysoft.com/en_halstead_metrics.html, Jun. 2012.

[Vickoff 2009] Jean-Pierre Vickoff. Méthode agile, les meilleures pratiques, compréhension et mise en oeuvre. QI, 2009.

[Wall 1987] Larry Wall. *The Perl language*. http ://www.perl.org/, 1987.

[Weber 2001] Chris Weber. *Using IPSec in Windows 2000 and XP, Part 1*. http ://www.symantec.com/connect/articles/using-ipsec-windows-2000-and-xp-part-1, 12 2001. by Symantec.

[Wettel & Lanza 2007] Richard Wettel and Michele Lanza. *Visualizing Software Systems as Cities*. In Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on, pages 92–99, 2007.

[Willinger W. & V. 1997] Sherman R. Willinger W. Taqqu M. S. and Wilson D. V. *Self-Similarity Through High-Variability : Statistical Analysis of Ethernet LAN Traffic at the Source Level*. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 5, NO. 1„ 02 1997.

[WindRiver 2009] WindRiver. *The Wind River VxWorks RTOS*. http ://www.windriver.com/products/vxworks/, 2009.

[Ylonen & Lonvick 2006] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Protocol Architecture*. RFC 4251 (Proposed Standard), January 2006.

[Yuhua 2008] Guo Yuhua. *Implementation of 3D Kiviat Diagrams*, Oct. 2008.

[Zuse 1997] Horst Zuse. A framework of software measurement. Walter De Gruyter Inc, 1997.

# Annexes

# A. Protection Profile for the Secure Next Generation Router (PP RT SNG)

# PROTECTION PROFILE

# SECURE NEXT GENERATION ROUTER

**Table des matières**

**Table des tableaux**



**Table des figures**

# 1 Introduction au Profil de Protection

## *1.1 Identification du Profil de Protection (PP)*

Thalès Avioniques et l'Ecole Nationale de l'Aviation Civile ont lancé conjointement une étude autour de la conception d'un routeur sécurisé de nouvelle génération pour les besoins aéronautiques avec un niveau élevé d'assurance en terme de sûreté et de sécurité.

| | |
|---|---|
| *Titre* | **Profil de Protection du Secure Next Generation Routeur** |
| *Auteur* | Antoine Varet, ENAC, THALES |
| *Version* | 1.0 |
| *Date* | 16/09/2011 |
| *Référence* | -- PP SNG Router |
| *Version des CC* | 3.1 Révision 3 |

## *1.2 Vue d'ensemble de la cible d'évaluation (Target of Evaluation, TOE)*

La TOE est un noeud du réseau (un routeur IP). Son rôle principal est l'acheminement des informations entre les différents noeuds du réseau d'une manière sécurisée.

Ce PP définit les exigences de sécurité auxquelles la TOE doit se conformer en vue d'une évaluation de sécurité.

### 1.2.1 Utilisation de la TOE

Un flot individuel consiste en une séquence de paquets IP issus d'un utilisateur.

La TOE est un noeud du réseau. C'est un routeur dont le rôle est de gérer le routage des flots individuels d'une manière sécurisée.

### 1.2.2 Type de TOE

La TOE est un produit composé des éléments matériels et logiciels suivants :

- un routeur ;
- l'application embarquée sur le routeur assurant les fonctionnalités de base pour lesquelles l'équipement est prévu.

### 1.2.3 Caractéristiques de sécurité de la TOE

Afin d'améliorer la sécurité des infrastructures réseau, la TOE offre la possibilité de sécuriser la communication entre deux routeurs SNG, soit en assurant l'intégrité et l'authenticité des données échangées, soit en assurant la confidentialité, l'intégrité et l'authenticité des données échangées.

La TOE assure un niveau élevé de sécurité et de sûreté pour les utilisateurs du réseau. La TOE étant destinée au secteur aéronautique, il n'est pas prévu de mettre à jour le logiciel ni de l'administrer en exploitation.

### 1.2.4 Environnement de la TOE

La TOE est un noeud d'un réseau IP interagissant avec d'autres routeurs, SNG ou non. Ce réseau logique repose sur une infrastructure réseau physique.

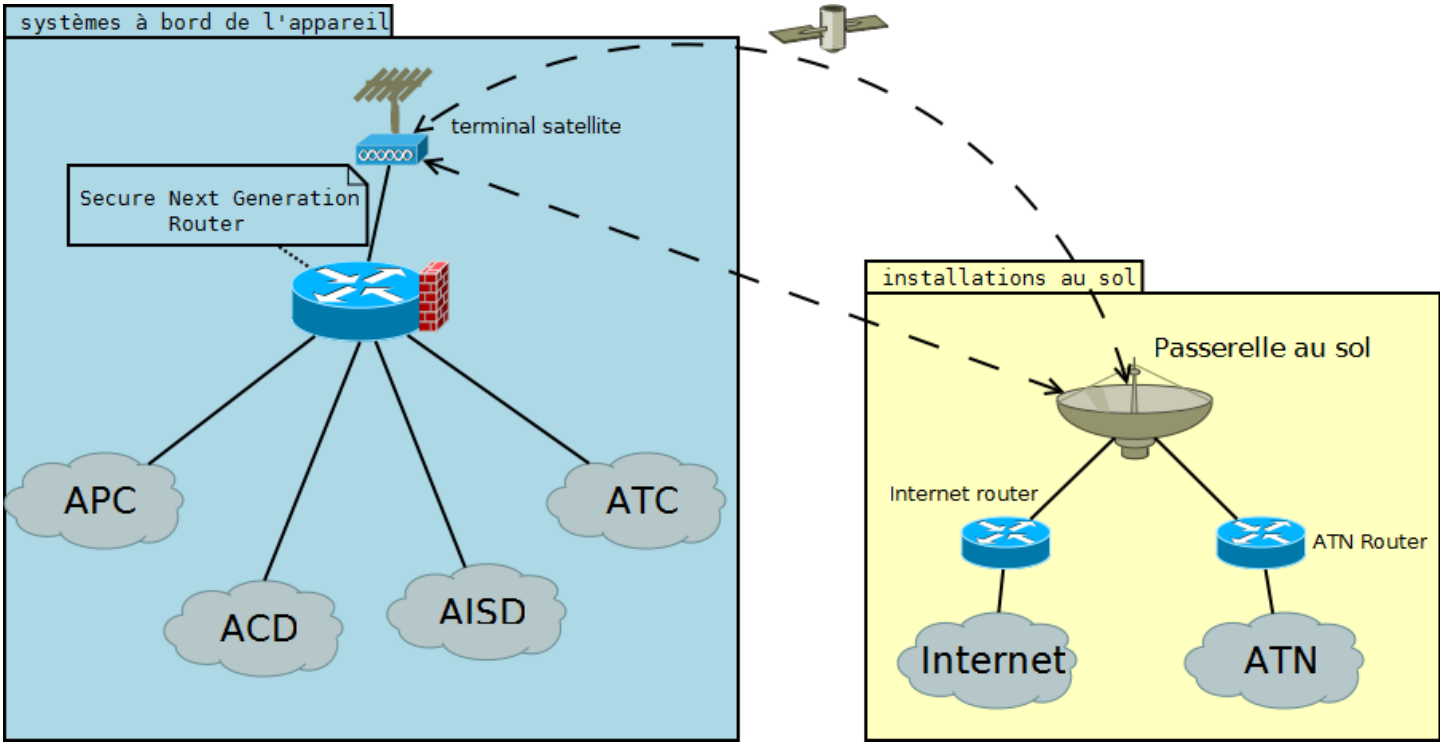



**Figure 1 Exemple d'environnement du routeur SNG**

Un exemple d'application du routeur SNG (présenté figure 1) consiste à l'utiliser à bord d'un avion pour router et sécuriser les communications avec les équipements au sol.

## 1.3 Description de la TOE

### 1.3.1 Architecture générale de la TOE

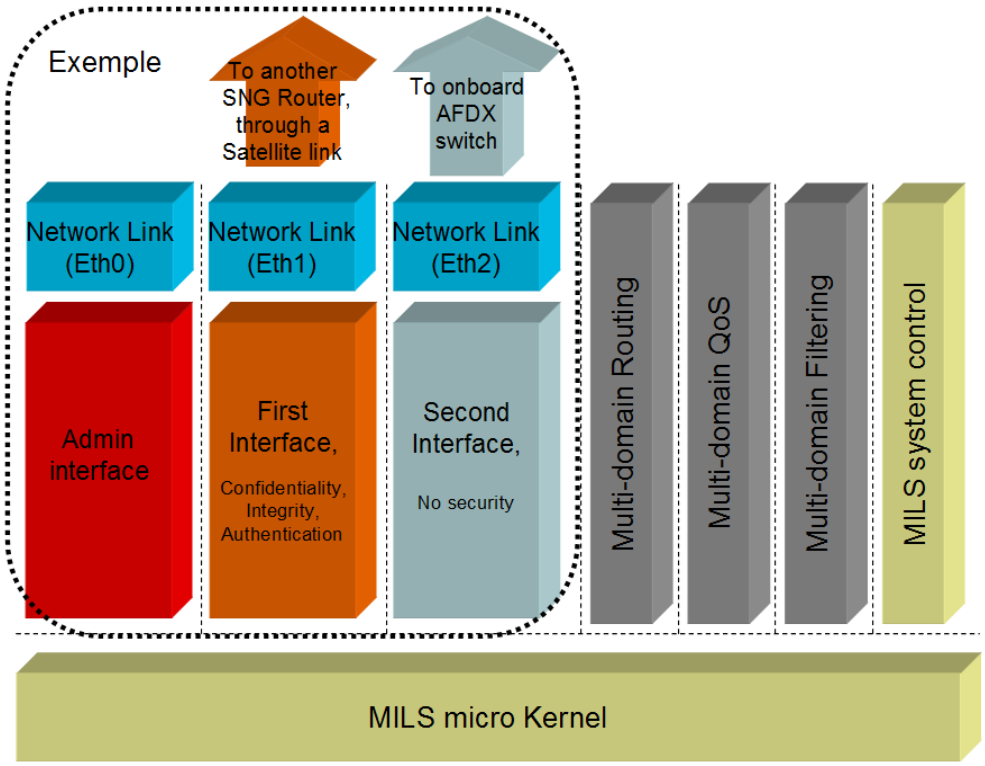



**Figure 2 Exemple d'architecture d'un routeur SNG**

L'architecture du routeur SNG repose sur une séparation logicielle des différents modules assurant les fonctionnalités proposées par le routeur. Les différents modules sont :

- un module d'entrée/sortie pour chaque interface du routeur assurant un premier filtrage au niveau de la couche liaison (le routeur possède 2 interfaces + 1 interface d'administration servant à la récupération des informations de journalisation et au téléchargement de configurations),
- un module de filtrage des paquets IP,
- un module de routage des paquets IP,
- un module de qualité de service,
- un micronoyau MILS assurant la ségrégation des modules et régulant les communications autorisées entre eux.

**Remarque :** La configuration du routeur présentée dans la figure 2 ne montre que 3 interfaces ethernet : une pour un réseau à bord, une sécurisée liée à un équipement de communication par satellite, une servant à la récupération des informations de journalisation. Certaines configurations du routeur comprennent des interfaces supplémentaires, ethernet ou autres, similaires aux "First-" et "Second Interface". Ainsi, la configuration utilisée dans le cas de la figure 1 présentant un exemple d'environnement pour le routeur dispose de 6 interfaces dont 4 sont destinées aux réseaux à bord.

### 1.3.2 Rôles

Le fonctionnement de la TOE dans son environnement opérationnel fait appel directement ou indirectement aux rôles décrits ci-dessous :

- Opérateur (chargé d'assurer le bon fonctionnement du routeur dans l'infrastructure du réseau tant que les conditions de sécurité restent réunies, en assurant par exemple le remplacement d'un matériel défectueux à bord de l'appareil)

- Utilisateur (les utilisateurs du routeur étant ici d'autres routeurs, commutateurs et équipements terminaux voisins communicant avec lui)

- Administrateur (son rôle consiste à consulter régulièrement l'état du routeur et de ses logs et à télécharger de nouveaux fichiers de configuration)

### 1.4 Environnement opérationnel de la TOE

Le routeur SNG peut se trouver dans une baie à bord d'un avion ou au sol.

### 1.5 Acronymes

| | |
|---|---|
| CC | (Common Criteria) Critères Communs |
| IP | Internet Protocol |
| PP | Profil de Protection |
| SEI | Secure Ethernet Interface |
| SFP | Security Function Policy |
| ST | Security Target |
| SNMP | Simple Network Management Protocol |
| TOE | Target of Evaluation |
| TSF | TOE Security Functionality |
| UEI | Unsecured Ethernet Interface |

### 1.6 références


| | |
|---|---|
| [CC1] | Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, Revision 3, July 2009. |
| [CC2] | Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1, Revision 3, July 2009. |
| [CC3] | Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements. Version 3.1, Revision 3, July 2009. |
| [CEM] | Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1, Revision 3, July 2009. |

# 2 Déclarations de conformité

Ce chapitre contient les sections suivantes :

- Déclaration de conformité de ce PP aux CC (2.1)
- Déclaration de conformité à un paquet d'exigences de sécurité (2.2)
- Déclaration de conformité à un PP (2.3)
- Déclaration pour la conformité à ce PP (2.4)

## *2.1 Déclaration de conformité aux critères communs*

Ce profil de protection est conforme à :

- la partie 2 des critères communs version 3.1 révision 3 de juillet 2009 (cf [CC2]),

- la partie 3 des critères communs version 3.1 révision 3 de juillet 2009 (cf [CC3]).

Aucune extension n'a été retenue, la conformité aux parties 2 et 3 est stricte. Aucun composant étendu n'a été défini.

## *2.2 Déclaration de conformité à un paquet d'exigences de sécurité*

Ce PP ne définit aucun ensemble d'exigence d'assurance.

## *2.3 Déclaration de conformité à un profil de protection*

Ce profil de protection ne s'appuie sur aucun autre profil de protection.

## *2.4 Déclaration pour la conformité à ce PP*

Les PP et ST conformes à ce PP peuvent annoncer un niveau de conformité **démontrable**.

# 3 Définition du problème de sécurité

## *3.1 Biens*

Les biens sensibles à protéger par la TOE se décomposent en deux catégories : les biens que la TOE sécurise et les biens dont elle dispose pour réaliser ses objectifs.

### 3.1.1 Biens protégés par la TOE

- Paquets transitant par une interface sécurisée : ils contiennent des données dont l'**intégrité**, l'**authenticité** et, selon la configuration de l'interface la **confidentialité**, doivent être protégées

Note : le routeur peut posséder des interfaces non sécurisées ; ces interfaces ne sont pas contraintes par les exigences de sécurité énumérées dans ce document.

### 3.1.2 Biens sensibles de la TOE

- Table de routage : elle comporte des informations sur l'aiguillage des paquets afin d'atteindre un équipement ou un réseau donné ; son **intégrité** doit être protégée
- Clés privées : ensemble des clés cryptographiques privées utilisées par le routeur pour assurer la communication sécurisée entre la TOE et un autre routeur SNG. Ces clés permettent le déchiffrement et la création de signature et doivent donc avoir leur **intégrité** et leur **confidentialité** protégées.
- Clés publiques : ensemble des clés cryptographiques publiques utilisées pour assurer la communication sécurisée entre la TOE et un autre routeur SNG. Ces clés permettent le chiffrement et la vérification de signature et doivent donc avoir leur **intégrité** protégée.
- Secrets d'administration : les clés partagées, privées, les mots de passe ou certificats pour s'identifier via l'interface d'administration auprès de la TOE doivent avoir leur **intégrité** et leur **confidentialité** protégées.
- Données administrables : les statistiques sur les paquets et la configuration doivent avoir leur **intégrité** protégée.

## *3.2 Menaces*

### 3.2.1 Menaces liées à la compromission des informations

Ces menaces ciblent la confidentialité des biens énumérés précédemment.

#### M.LISTEN_COM

Un attaquant capture et analyse les données qui entrent et sortent d'une ou plusieurs interfaces sécurisées de la TOE afin d'en extraire une partie ou toute information transportée.

#### M.LISTEN_AUTH

Un attaquant capture et analyse les données entrant et sortant de la TOE afin de récupérer des données d'authentification et rejouer ces messages pour avoir un accès inapproprié aux données sensibles.

#### M.LISTEN_SENSIBLE

Un attaquant capture et analyse les données entrant et sortant de la TOE afin de récupérer des informations sensibles et configurer la TOE d'une façon inappropriée.

#### M.DIVULG_SECRETS

Un attaquant parvient à accéder sur le système cible aux clés secrètes et autres secrets (mots de passe, certificats) utilisés pour protéger les données, ce qui invalide les services de sécurité offerts par le routeur (intégrité, authenticité et confidentialité).

### 3.2.2 Menaces liées à l'altération des informations

Ces menaces ciblent l'intégrité des biens énumérés précédemment.

**M.ALTER_SECRETS**

Un attaquant parvient à modifier les clés publiques, privées et autres secrets stockés dans la TOE.

**M.ALTER_LOGS**

Un attaquant parvient à modifier les statistiques stockées dans la TOE.

**M.ALTER_TABLE_DE_ROUTAGE**

Un attaquant parvient à modifier la table de routage stockée dans la TOE.

**M.ALTER_COM**

Un attaquant injecte des données dans la TOE, afin de faciliter la mise en oeuvre d'autres menaces.

### 3.2.3 Menaces liées à la compromission des fonctions

**M.USURPATION**

Un attaquant profite de la faiblesse du système d'authentification pour accéder aux fonctionnalités d'administration de la TOE.

**M.DENY_RESP**

Un attaquant renie avoir utilisé le routeur pour l'accès aux informations sensibles ou leur modification.

**M.DENY_OF_SERVICE_INTERFERENCE**

Un attaquant injecte des données illégitimes interprétables par la TOE afin de limiter voir d'empêcher la TOE d'effectuer son traitement des données légitimes.

**M.DENY_OF_SERVICE_SURCHARGE**

Un attaquant injecte une quantité de données suffisamment importante dans la TOE afin de l'empêcher d'effectuer son traitement des données légitimes.

**M.IP_SPOOFING**

Un attaquant émet des données vers la TOE en utilisant des informations d'identification (IP source) associées à des systèmes utilisateurs qu'il ne contrôle pas afin d'utiliser le routeur comme passerelle l'authentifiant.

**M.DYSFUNCTION**

Un attaquant met la TOE dans un état contribuant à empêcher la TOE d'effectuer son traitement ou permettant à l'attaquant d'effectuer d'autres menaces.

## *3.3 Politiques de sécurité organisationnelles (OSP)*

Ce profil de protection n'impose aucune politique de sécurité organisationnelle. Un PP ou un ST conforme à ce PP peut et devrait ajouter des politiques adaptées aux contextes d'utilisation du routeur.

### 3.4 Hypothèses

La matrice suivante présente les associations liant les menaces aux biens à protéger. Dans ce tableau, les attributs des biens sont les suivants : C=Confidentialité, I=Intégrité, D=Disponibilité et Au=Authenticité. Un "X" signifie une association conduisant à la perte de l'attribut du bien ciblé.

| | ***Paquets IP*** | | | ***Table de routage*** | ***Clefs privées*** | | ***Clefs publiques*** | ***Secrets d'admin*** | | ***Données admin*** |
|---|---|---|---|---|---|---|---|---|---|---|
| | C | I/D | Au | I | C | I | I | C | I | I |
| M.LISTEN_COM | X | | | | | | | | | |
| M.LISTEN_AUTH | | | X | | | | | | | |
| M.LISTEN_SENSIBLE | | | | | X | | | X | | |
| M.DIVULG_SECRETS | | | | | X | | | X | | |
| M.ALTER_SECRETS | | | | | | X | X | | X | |
| M.ALTER_LOGS | | | | | | | | | | X |
| M.ALTER_TABLE_DE_ROUTAGE | | | | X | | | | | | |
| M.ALTER_COM | X | X | X | | | | | | | |
| M.USURPATION | | | | | | | | | | X |
| M.DENY_RESP | | | | | | | | | | X |
| M.DENY_OF_SERVICE_INTERFERENCE | | X | | | | | | | | |
| M.DENY_OF_SERVICE_SURCHARGE | | X | | | | | | | | |
| M.IP_SPOOFING | | | X | | | | | | | |
| M.DYSFUNCTION | | X | | | | | | | | |

# 4 Objectifs de sécurité

## *4.1 Objectifs de sécurité pour la TOE*

### O.SECURE_DATA

La TOE doit fournir des fonctionnalités garantissant la confidentialité des données échangées à sécuriser.

### O.AUTH_AND_INT_DATA

La TOE doit fournir des fonctionnalités garantissant l'intégrité et l'authenticité des données échangées à sécuriser.

### O.FAIR_RSRC_USE

La TOE doit répartir les ressources nécessaire aux traitements des paquets de manière à pouvoir traiter toutes les sources de données sans en affamer certaines.

### O.REPLAY_DATA

La TOE doit fournir un moyen de détection et de refus du rejeu de l'authentification et d'autres messages sensibles.

### O.STATIC_INTEGRITE

La TOE doit garantir l'intégrité des données statiques nécessaires à son fonctionnement.

### O.ACCESS_ADMIN

La TOE doit authentifier les auditeurs avant qu'ils puissent accéder aux données stockées dans la TOE ou les modifier. La TOE doit détecter les attaques de type rejeu.

### O.CONFIDENTIAL_ADMIN

Les échanges avec les auditeurs doivent être confidentiels.

### O.ADMIN_INTEGRITE

L'intégrité des statistiques enregistrées et manipulées par la TOE doit être garantie.

## 4.2 Objectifs de sécurité pour l'environnement opérationnel

Toutes les menaces énumérées dans ce profil de protection sont couvertes par des objectifs de sécurité pour la TOE. En conséquence, il n'y a aucun objectif de sécurité pour l'environnement opérationnel. Une cible de sécurité (ST) conforme à ce PP peut ajouter des objectifs de sécurité spécifiques pour son environnement opérationnel.

## 4.3 Argumentaire des objectifs de sécurité

### 4.3.1 Menaces (TBD)

### 4.3.2 Politiques de sécurité organisationnelles (OSP)

N'ayant aucune politique de sécurité organisationnelle, cette section n'a aucun argument à développer.

### 4.3.3 Hypothèses

Dans la table suivante, un X représente la couverture d'une menace par un objectif de sécurité.

| | ***O.SECURE_DATA*** | ***O.AUTH_AND_INT_DATA*** | ***O.FAIR_RSRC_USE*** | ***O. REPLAY_DATA*** | ***O.STATIC_INTEGRITE*** | ***O.ACCESS_ADMIN*** | ***O.CONFIDENTIAL_ADMIN*** | ***O.ADMIN _INTEGRITE*** |
|---|---|---|---|---|---|---|---|---|
| M.LISTEN_COM | X | | | | | | | |
| M.LISTEN_AUTH | | X | | | | | | |
| M.LISTEN_SENSIBLE | | | | | | | X | |
| M.DIVULG_SECRETS | | | | | | X | X | |
| M.ALTER_SECRETS | | | | | X | X | | |
| M.ALTER_LOGS | | | | | | X | | X |
| M.ALTER_TABLE_DE_ROUTAGE | | | | | X | | | |
| M.ALTER_COM | X | X | | X | | | | |
| M.USURPATION | | | | | | X | X | |
| M.DENY_RESP | | | | | | X | | |
| M.DENY_OF_SERVICE_INTERFERENCE | | X | | | | | | |

| | *O.SECURE_DATA* | *O.AUTH_AND_INT_DATA* | *O.FAIR_RSRC_USE* | *O. REPLAY_DATA* | *O.STATIC_INTEGRITE* | *O.ACCESS_ADMIN* | *O.CONFIDENTIAL_ADMIN* | *O.ADMIN _INTEGRITE* |
|---|---|---|---|---|---|---|---|---|
| M.DENY_OF_SERVICE_SURCHARGE | | | X | | | | | |
| M.IP_SPOOFING | | X | | | | | | |
| M.DYSFUNCTION | X | X | | X | | | | |

### 4.3.4 Tables de couverture entre définition du problème et objectifs de sécurité (TBD)

| *Menaces* | *Objectifs de sécurité* | *Argumentaire* |
|---|---|---|

**Tableau 1Association menaces vers objectifs de sécurité**

| *Objectifs de sécurité* | *Menaces* | *Argumentaire* |
|---|---|---|

**Tableau 2Association objectifs de sécurité vers menaces**

| *Politiques de sécurité organisationnelles (OSP)* | *objectifs de sécurité* | *argumentaire* |
|---|---|---|

**Tableau 3Association politiques de sécurité organisationnelles (OSP) vers objectifs de sécurité**

| *objectifs de sécurité* | *Politiques de sécurité organisationnelles (OSP)* | *argumentaire* |
|---|---|---|

**Tableau 4Association objectifs de sécurité vers politiques de sécurité organisationnelles (OSP)**

| *Hypothèses* | *Objectifs de sécurité pour l'environnement opérationnel* |
|---|---|

**Tableau 5Association hypothèses vers objectifs de sécurité pour l'environnement opérationnel**

| *Objectifs de sécurité pour l'environnement opérationnel* | *Hypothèses* |
|---|---|

**Tableau 6Association objectifs de sécurité pour l'environnement opérationnel vers hypothèses**

# 5 Exigences de sécurité

## *5.1 Exigences de sécurité fonctionnelles*

### 5.1.1 SFR applicables aux interfaces utilisateurs

### FDP_IFC.2 Information Flow Control Policy

The TSF shall enforce the ***confidential authenticated flow policy*** on ***transmitted IP packets*** through a ***confidential authenticated SEI*** and all operations that cause that information to flow to and from subjects covered by the SFP.

The TSF shall enforce the ***authenticated flow control policy*** on ***transmitted IP packets*** through an ***authenticated-SEI*** and all operations that cause that information to flow to and from subjects covered by the SFP.

The TSF shall enforce the ***no flow control policy*** on ***transmitted and received IP packets*** and all operations that cause that information to flow to and from subjects covered by the SFP.

**Note d'application :** les trois exigences précédentes permettent de formaliser les trois niveaux de sécurité possibles pour les paquets devant être émis par le routeur : UEI, auth-SEI, conf-auth-SEI. Les paquets non sensibles ne nécessitent pas de gestion particulière des flots et sont donc du niveau UEI (Unsecured Ethernet Interface). Les paquets qui nécessitent des vérifications d'authenticité uniquement sont classés au niveau de sécurité auth-SEI (authenticated Secure Ethernet Interface) tandis que ceux qui nécessitent simultanément de l'authenticité et de la confidentialité sont du niveau conf-auth-SEI (confidential authenticated Secure Ethernet Interface).

The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

**Note d'application :** l'exigence précédente précise que tout paquet transitant par le routeur appartient forcément à l'un des trois niveaux précités.

### FDP_IFF.1 Information Flow Control Functions

The TSF shall enforce the ***confidential authenticated flow control policy*** based on the following types of subject and information security attributes: ***IP packet to be transmitted through confidential authenticated SEI***.

The TSF shall enforce the ***authenticated flow control policy*** based on the following types of subject and information security attributes: ***IP packet to be transmitted through authenticated SEI.***

The TSF shall enforce the ***no flow control policy*** based on the following types of subject and information security attributes: ***IP packet to be transmitted***

***through UEI or IP packet whose transmission interface type is undefined***.

**Note d'application :** les trois exigences précédentes permettent d'associer à chaque paquet transitant par le routeur l'un des trois niveaux de sécurité précisée.

The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: ***-all received packets change for no flow control policy; -all packets to transmit change for the flow control policy assigned to the exiting interface***.

The TSF shall enforce the ***none***.

The TSF shall explicitly authorise an information flow based on the following rules: ***none***.

The TSF shall explicitly deny an information flow based on the following rules: ***none***.

**Note d'application :** les quatre exigences précédentes explicitent qu'un paquet peut changer de niveau de sécurité lors de son traitement, le nouveau niveau dépendant de décisions internes au routeur.

## FDP_UCT.1 Inter-TSF User data Confidentiality Transfer protection

The TSF shall enforce the ***confidential authenticated flow control*** to ***transmit*** user data in a manner protected from unauthorised disclosure.

**Note d'application :** l'exigence précédente associe la niveau de sécurité confidentielle et authentifiée à l'application impérative de mécanismes de protection de la confidentialité des données (ici les paquets échangés).

## FDP_UIT.1 Inter-TSF User data Integrity Transfer protection

The TSF shall enforce the ***confidential authenticated flow control and the authenticated flow control*** to transmit user data in a manner protected from ***modification, deletion, insertion, replay*** errors.

The TSF shall be able to determine on receipt of user data, whether ***modification, deletion, insertion, replay*** has occurred.

**Note d'application :** l'exigence précédente associe les niveaux de sécurité conf-auth-SEI et auth-SEI à l'application impérative de mécanismes de protection de l'intégrité des données (ici les paquets échangés). L'authenticité des messages échangés ne peut être garantie sans vérifier leur intégrité. Ici, l'intégrité consiste à vérifier que le message n'a pas été modifié, qu'il n'a pas été complété par des informations supplémentaires ou tronqué. Le routeur doit aussi détecter les attaques de type rejeu. En cas de détection d'informations altérées ou rejouées, ces informations sont ignorées.

## FTP_ITC.1 Import from outside of the TOE

The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

The TSF shall permit ***the TSF or another IT-product*** to initiate communication via the trusted channel.

The TSF shall initiate communication via the trusted channel for ***transmission of packet***.

**Note d'application :** l'application de mécanismes de sécurité tels que l'authentification et la confidentialité nécessitent de mettre en place des "canaux de communication" dédiés au transport sécurisé d'information. En pratique, ces canaux pourraient être des tunnels bidirectionnels.

## FCO_NRO.2 Non-Repudiation of Origin

The TSF shall enforce the generation of evidence of origin for transmitted ***IP packets through SEI*** at all times.

**Note d'application :** l'exigence précédente formalise le besoin d'authentification des données sensibles (confidentielles ou non, toutes nécessitent l'authentification). Ce besoin impose, à l'émission des paquets, d'adjoindre aux données sensibles les informations d'authenticité de l'émetteur (ici, le routeur) et des informations pour contrôler ultérieurement l'intégrité des paquets transmis.

The TSF shall be able to relate the ***authenticity*** of the originator of the information, and the ***body of the packet*** of the information to which the evidence applies.

The TSF shall provide a capability to verify the evidence of origin of information to ***recipient*** given ***immediate***.

**Note d'application :** les deux exigences précédentes permettent de formaliser le besoin d'authentification des données sensibles. Ce besoin impose, à la réception, de vérifier dès la réception du paquet les informations adjointes d'authenticité de l'émetteur.

## FIA_UID.2 User Identification

The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

**Note d'application :** l'authenticité d'une source nécessite des informations d'identification de cette source. Cette exigence formalise ce besoin d'identité associé aux mécanismes d'authenticité. Dans le cas du routeur, les paquets IP entrant contiennent systématiquement une adresse IP source permettant cette identification.

## FRU_RSA.2 Minimum and maximum quotas

The TSF shall enforce maximum quotas of the following resources ***time slots*** that ***partitions*** can use ***over the CPU time frame.***

The TSF shall ensure the provision of minimum quantity of each ***time slots*** that is available for ***partitions*** to use ***over the CPU time frame.***.

**Note d'application :** le routeur est conçu en exécutant des partitions partageant le même processeur. Cette concurrence est rendue possible en partitionnant le temps d'exécution attribué à chaque partition. Chaque créneau temporel est appelé "time slot" et l'ensemble des créneaux successifs forme une partition d'une fenêtre temporelle appelée "CPU time frame", permettant aux partitions d'être exécutées de manière périodique. Les deux exigences précédentes imposent au partitionnement de la fenêtre temporelle des contraintes sur la

répartition des time slots des partitions : une partition doit disposer d'un minimum et d'un maximum de time slots pour garantir son exécution et l'empêcher d'affamer les autres partitions.

### 5.1.2 SFR applicables aux données stockées par la TSF

## FDP_SDI.2 Stored Data Integrity

> The TSF shall monitor user data stored in containers controlled by the TSF for ***invalid integrity checksum*** on all objects, based on the following attributes: ***pre-calculated reference value stored in the configuration***.
>
> Upon detection of a data integrity error, the TSF shall ***halt the corrupted partition if it is not the partition for administration, halt the TSF (then halt all partitions) if the corrupted partition is the partition for administration***.

**Note d'application :** les deux exigences précédentes imposent au routeur de vérifier l'intégrité des informations qu'il utilise afin d'éviter d'employer des informations erronées. La validité des données n'est pas traitée ici, la configuration stockée dans le routeur étant supposée correcte. Seule l'absence de changement entre le moment de chargement et d'utilisation des données de configuration est vérifiée à l'aide d'une somme de contrôle validant l'intégrité de ces données. La dernière exigence permet d'empêcher l'utilisation de données non intègres en désactivant les sujets (partitions) associées aux données corrompues. Ainsi, il est garanti que les données utilisées sont intègres : les données corrompues sont inutilisées car les partitions associées sont arrêtées.

## FPT_TST.1 TSF Self-Test

> The TSF shall run a suite of self tests ***during initial start-up, after downloading a configuration file, before uploading audit data*** to demonstrate the correct operation of the TSF.

**Note d'application :** afin de garantir l'intégrité des données utilisées, il est nécessaire de mettre en place un mécanisme de contrôle (cf FDP_SDI.2) et de préciser quand est utilisé ce mécanisme. L'exigence précédente précise ce dernier point.

> The TSF shall provide authorised users with the capability to verify the integrity of ***TSF data***.
>
> The TSF shall provide authorised users with the capability to verify the integrity of ***TSF***.

**Note d'application :** les deux exigences précédentes imposent au routeur de fournir une fonctionnalité de vérification de l'intégrité pouvant être déclenchée à la demande de l'administrateur (seul utilisateur autorisé). L'intégrité du routeur signifie en fait l'intégrité des données de configuration qui y sont stockées. Le mécanisme d'audit (qui renvoie les statistiques) accessible par l'administrateur est tout à fait adapté pour valider ces deux exigences : l'intégrité doit être vérifiée à l'appel des fonctionnalités d'audit afin de renvoyer le résultat dans les statistiques renvoyées par l'appel.

### 5.1.3 SFR applicables à l'interface d'administration

## FTP_TRP.1 Trusted Path

> The TSF shall provide a communication path between itself and ***local*** users that is logically distinct from other communication paths and provides assured

identification of its end points and protection of the communicated data from ***disclosure***.

**Note d'application :** la sécurité du routeur ne peut être garantie qu'en dissociant les opérations d'administration du routeur (notamment le chargement de nouvelles configurations) des opérations d'utilisation du routeur (routage sécurisé des paquets). Cette dissociation est assurée physiquement par l'utilisation d'une interface d'administration dédiée, indépendantes des interfaces "utilisateur" du routeur. Cette interface doit être sécurisée en terme de confidentialité.

The TSF shall permit ***local users*** to initiate communication via the trusted path.

**Note d'application :** le routeur n'étant pas prévu pour interagir activement durant l'administration, seuls des utilisateurs extérieurs peuvent déclencher les fonctionnalités d'administration et donc initier la communication avec le routeur.

The TSF shall require the use of the trusted path for ***all administrative tasks (configuration and statistics)***.

**Note d'application :** L'administration via les interfaces "utilisateurs" ne doit pas être autorisée.

### FPT_RPL.1 Replay Detection

The TSF shall detect replay for the following entities: ***administration session***.

The TSF shall perform ***ignoring replayed data*** when replay is detected.

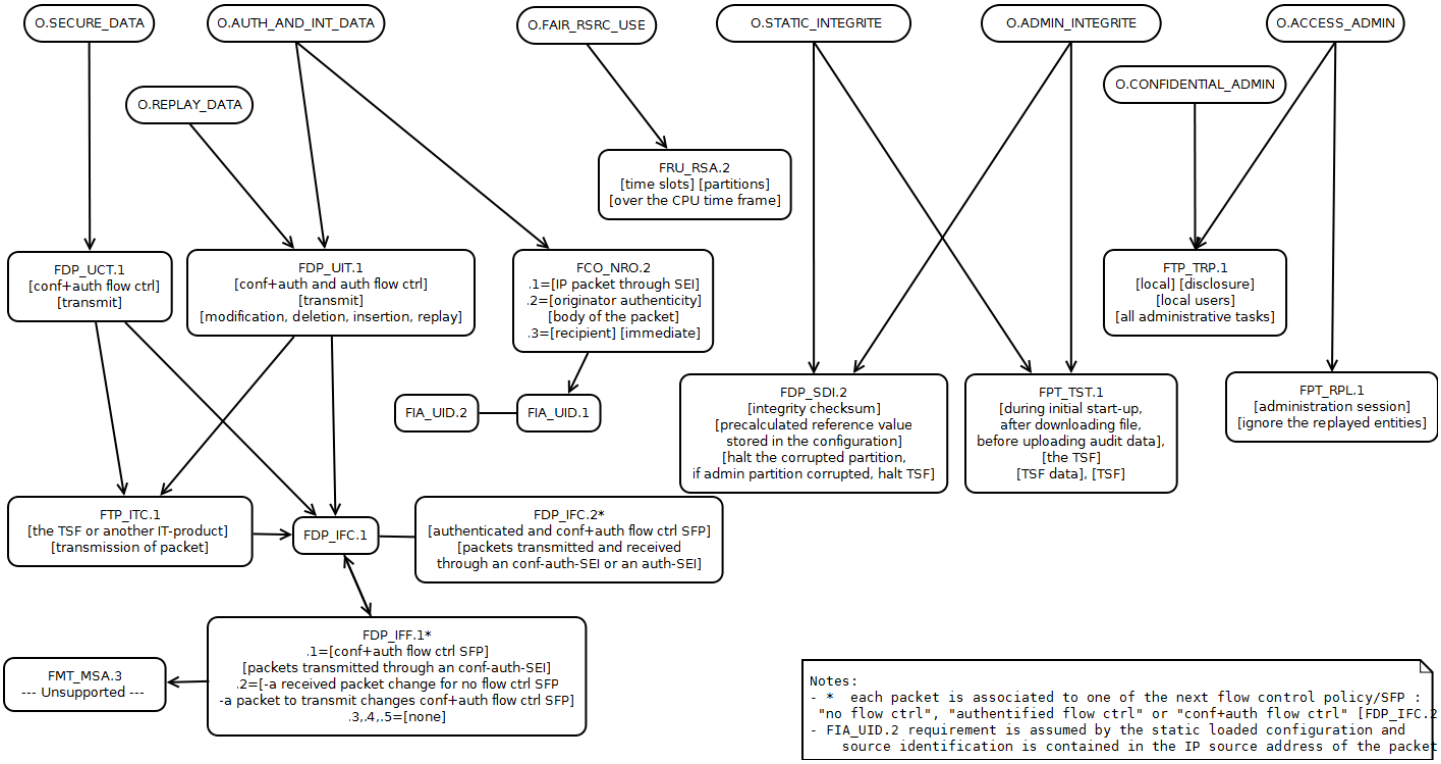
**Note d'application :** Le routeur doit détecter les attaques de type rejeu sur son interface d'administration. Les informations étant détectées dans ce cas doivent être ignorées. Les attaques de type rejeu sur les autres interfaces sont traitées dans FDP_UIT.1.

## *5.2 Exigences de sécurité d'assurance*

Ce document étant un Profil de Protection, il n'est pas nécessaire de définir d'exigence de sécurité d'assurance particulière.

## 5.3 Argumentaire des exigences de sécurité (TBD)

### 5.3.1 Objectifs de sécurité pour la TOE




### 5.3.2 Tables de couverture entre objectifs et exigences de sécurité

| Objectifs de sécurité | Exigences fonctionnelles pour la TOE | Argumentaire |
|---|---|---|

**Tableau 7 Association objectifs de sécurité de la TOE vers les exigences fonctionnelles**

| Exigences fonctionnelles pour la TOE | Objectifs de sécurité |
|---|---|

**Tableau 8 Association exigences fonctionnelles vers objectifs de sécurité de la TOE**

### 5.3.3 Dépendances des exigences de sécurité fonctionnelles

| Exigences | Dépendances CC | Dépendances Satisfaites |
|---|---|---|

**Tableau 9 Dépendances des exigences fonctionnelles**

+ Argumentaire pour les dépendances non satisfaites ...

### 5.3.4 Dépendances des exigences de sécurité d'assurance

| Exigences | Dépendances CC | Dépendances Satisfaites |
|---|---|---|

**Tableau 10 Dépendances des exigences d'assurance**

### 5.3.5 Argumentaire pour les exigences de sécurité d'assurance

# B. Mise en œuvre du dæmon SCOUT6 version beta

# MISE EN OEUVRE DU DÆMON SCOUT6

Antoine Varet

## Présentation de ce programme

Le dæmon SCOUT6 permet de rechercher la possibilité d'établir un canal sécurisé. Cette version fonctionne avec IPv6 sur des PC Linux.

Son exécution ainsi que (presque) toutes les commandes présentées ici nécessitent les privilèges maximaux sur un système Linux ; cette implémentation de SCOUT6 a été conçue et testée avec Debian et des machines virtuelles.

Les protocoles SCOUT et SCOUT6 ont été présentés à la conférence SAR SSI 2012 à Cabourg (France) dans un article scientifique intitulé « Security capability discovery protocol over unsecured IP-based topologies ».

## Installation et configuration du dæmon

Afin de le mettre en œuvre, il est nécessaire de disposer d'un système Linux, du programme binaire scout6 et du script scout6d. Dans ce qui suit, les commandes sont à exécuter en tant que superutilisateur (root), dans un terminal, dans le répertoire où l'archive SCOUT6 a été décompressée.

SCOUT6 nécessite l'installation de la libipq (installée avec le paquet Debian iptables-dev) :

```
(root)# make prepare_compile
```

Le dæmon SCOUT6 peut alors être compilé puis installé :

```
(root)# make install
```

Les fichiers temporaires peuvent alors être supprimés :

```
(root)# make clean
```

Le fichier de configuration de SCOUT6 se nomme `/etc/scout6/scout6.conf` . Bien que facultatif, il peut être présent et ainsi surcharger les variables d'environnement et les valeurs par défaut que SCOUT6 utilise pour sa configuration.

Dans le cas d'un routeur interceptant des paquets non sécurisés pour leur imposer de transiter via un tunnel sécurisé, il peut être nécessaire de spécifier quelles adresses ou plages d'adresses doivent être sécurisées. Cela s'effectue via le fichier intercept_ra_for_addresses.conf, à raison d'une ligne par adresse ou couple d'adresse. Par défaut, le dæmon tentera de sécuriser tout paquet transitant (`::/0`).

# Environnement de fonctionnement

Le dæmon peut fonctionner de différentes manières. Prenons le réseau minimal suivant :

```
Ha — Ra — Rb — Hb
```

avec la configuration IPv6 suivante :

```
 IPv6 node Ha
[2001::1/64 on eth1]
 ||
[2001::2/64 on eth1]
 IPv6 router Ra
[2002::2/64 on eth2]
 ||
[2002::3/64 on eth2]
 IPv6 router Rb
[2003::3/64 on eth1]
 ||
[2003::4/64 on eth1]
 IPv6 node Hb
```

Ce réseau permet de mettre en évidence les 4 modes de résultats découverts par SCOUT :

- Mode Host-Host (Ha == Hb)
- Mode Router-Host (Ha – Ra == Hb)
- Mode Host-Router (Ha == Rb – Hb)
- Mode Router-Router (Ha – Ra == Rb – Hb)

# Premier démarrage

Pour la suite, nous supposerons que nous avons le réseau minimal et que le dæmon SCOUT est installé sur Ha et sur Hb (mode Host-host).

Les fichiers et dossiers importants sont :

- /opt/scout6/scout6 (le programme binaire SCOUT6)
- /opt/scout6/scout6d (le daemon qui appelle le programme binaire)
- /etc/scout6/scout6.conf (le fichier de configuration)
- /etc/scout6/proto2 (les commandes pour établir un tunnel SSH, expérimental)
- /etc/scout6/proto4 (les commandes pour établir un tunnel IKEv2, plus stable)

## *Commandes pour configurer les hôtes*

### Sur Ha :

```
    (user)@ha$ su
# Ajouter l'IPv6 2001::1 à l'interface eth1 qui est liée à Ra
    (root)@ha# ifconfig eth1 add 2001::1/64 up
# Puis ajouter une route pour les paquets passant par Ra
    (root)@ha# route -6 add 2000::/8 gw 2001::2
```

### Sur le routeur Ra :

```
    (root)@ra# ifconfig eth1 add 2001::2/64 up
    (root)@ra# ifconfig eth2 add 2002::2/64 up
# Envoyer par défaut à Rb
    (root)@ra# route -6 add 2000::/8 gw 2002::3
# Activer le routage IPv6
    (root)@ra# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

## De même sur Rb :

```
(root)@rb# ifconfig eth1 add 2003::3/64 up
(root)@rb# ifconfig eth2 add 2002::3/64 up
(root)@rb# route -6 add 2000::/8 gw 2002::2
(root)@rb# echo 1 >  /proc/sys/net/ipv6/conf/all/forwarding
```

## Et enfin sur Hb :

```
(root)@hb# ifconfig eth1 add 2003::4/64 up
(root)@hb# route -6 add 2000::/8 gw 2003::3
```

A noter que dès à présent, Ha et Hb doivent pouvoir communiquer sans sécurité :

```
(user)@ha$ ping6 2003::4  (...ok)
```

On peut désormais lancer SCOUT6 sur Ha et Hb :

```
(root)#  /opt/scout6/scout6d start eth1
```

A partir de ce moment, les paquets ICMPv6 émis par ping6 et les paquets UDPv6 seront interceptés et déclencheront la découverte de tunnel puis l'invocation de celui-ci. A noter qu'il est intéressant de se placer sur l'un des routeurs (par exemple) et d'observer le comportement du réseau à l'aide d'un analyseur réseau tel que Wireshark.

Ainsi, la commande sur l'hôte Ha :

```
(root)@ha#  ping6 -I eth1 2003::4 -c 5
```

permet de voir sur le routeur Ra les paquets SCOUT6 :

1. un Doq de Ha vers Hb,
2. suivi d'un Dor de Hb vers Ha.
3. Le tunnel IKEv2 est alors établi entre Ha et Hb :
   1. 2 paquets IKE_SA_INIT
   2. suivis de 2 IKE_SA_AUTH
4. Ensuite, les paquets ICMPv6 sont échangés entre Ha et Hb de manière sécurisés : ils sont encapsulés dans des paquets ESP et n'apparaissent donc pas « en clair » sur les routeurs intermédiaires Ra et Rb.

Il est important de remarquer que le premier paquet ICMP est systématiquement perdu. Ce problème s'explique par sa destruction lors de l'établissement du tunnel sécurisé.

L'état des dæmons peut être demandé à tout moment à l'aide de

```
(root)#  /opt/scout6/scout6d status
```

Les dæmons peuvent être arrêtés à tout moment à l'aide de

```
(root)#  /opt/scout6/scout6d stop
```

La commande suivante de démarrage permet d'être beaucoup plus verbeux :

```
(root)#  VERBOSE="scout6d_parle" cmd_line_plus="-vvv"
/opt/scout6/scout6d start eth1
```

La variable `VERBOSE` demande au script de démarrage de détailler les étapes, le « `-vvv` » rend bavard le programme scout6 en le faisant passer du niveau d'affichage 0 (quiet) à 3 (info) (le maximum est le niveau 4 debug avec « `-vvvv` »).

D'autres expériences peuvent être effectuées en arrêtant l'un des dæmons, en exécutant SCOUT6 sur Ra, Rb ou les deux routeurs et en l'arrêtant sur Ha, Hb ou les deux hôtes d'extrémité...

## Autres notes à l'attention de développeurs...

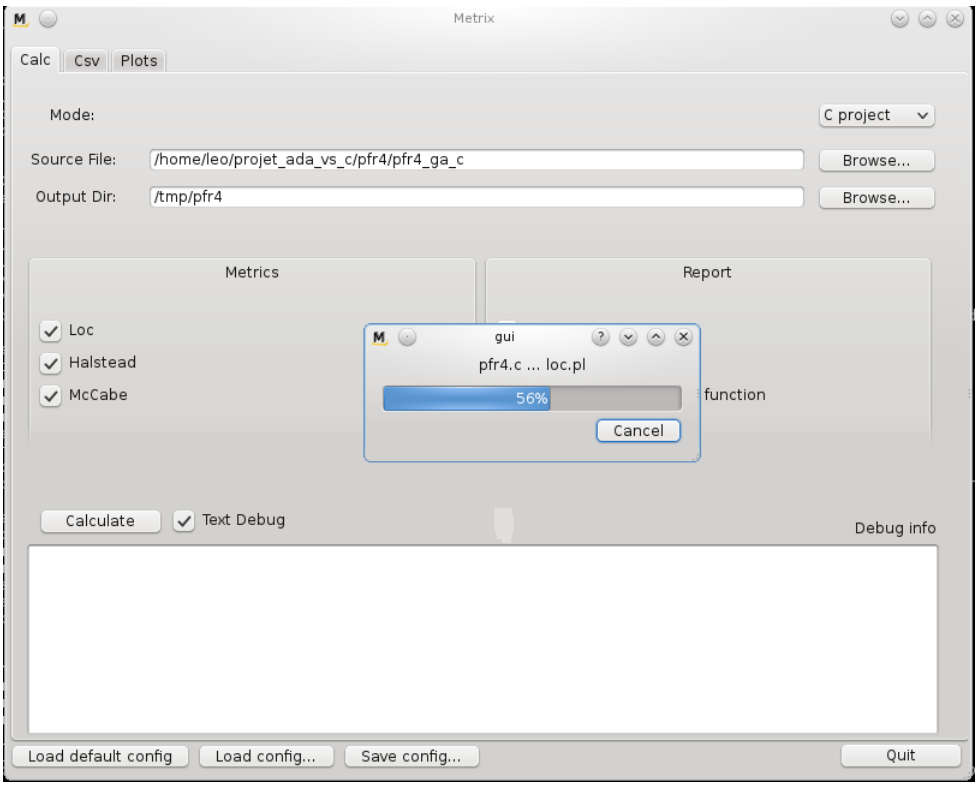
- Le fichier `scout6.c` a été adapté pour être parsé avec **doxygen** qui rend les 1950 lignes du code source plus accessibles en générant une documentation (html, pdf...) conviviale.
- En mode router-router, on a {Ha=inspiring node, Ra=initiator node, Rb=responder node, Hb=targeted node}
- En mode host-host, on a {Ha=inspiring node=initiator node, Hb=responder node=targeted node}
- Lors de la phase de découverte, si plusieurs protocoles sont compatibles entre les nœuds, c'est le protocole ayant le n° le plus élevé qui est sélectionné. Ainsi, proto4 sera privilégié par rapport à proto2.
- Les protocoles sont invoqués via des scripts placés dans /etc/scout6/protoXX, où XX est un numéro arbitraire (actuellement 2=ssh et 4=ikev2). Ces scripts sont basés sur des modèles présents et commentés dans `/etc/scout6/proto_template`.
- Les commandes invoquées sont suivies de 4 paramètres qui sont respectivement les **@IPv6 inspiring, initiator, responder** puis **targeted**. Par exemple, le binaire peut invoquer la commande : « `/etc/scout6/proto4/invoke_channel.sh 2001::1 2002::2 2003::4 2003::4` »
- Merci de me contacter pour m'indiquer les bugs éventuels, les suggestions d'amélioration, les bouts de code utiles, quel(s) mode(s) du dæmon sont utilisés...

avaret@recherche.enac.fr

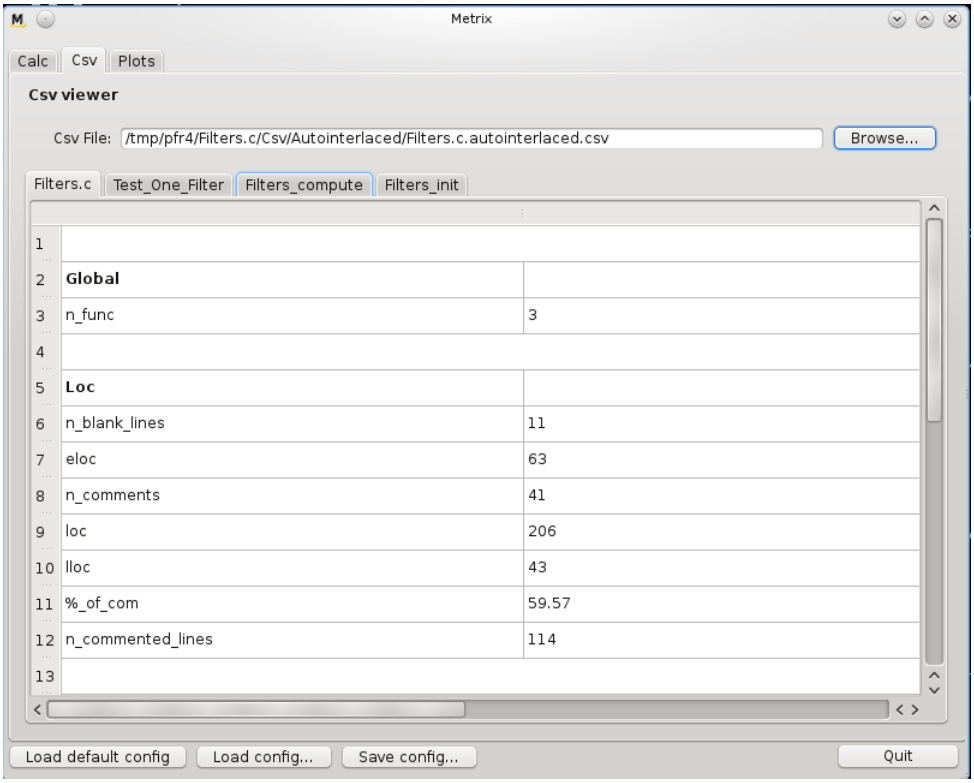
# C. Interface Homme-Machine de Métrix

Notre logiciel d'évaluation quantitative de la qualité du code source Métrix est fourni avec une interface homme-machine, aussi appelée interface graphique utilisateur, écrite en C++[Stroustrup 1983] avec Qt4[Digia 1995]. Cette interface dispose de trois onglets, représentés dans les figures C.1a, C.1b, C.2a et C.2b.

1. L'onglet «Calc» permet de définir le dossier contenant les fichiers sources à analyser et d'indiquer où mettre le rapport à générer.
2. L'onglet «Csv» fournit à l'utilisateur un moyen rapide de visualiser la sortie textuelle des scripts de mesures, donc d'afficher fichier par fichier les résultats d'analyse.
3. L'onglet «Plots» propose de visualiser graphiquement les résultats, soit sur un diagramme de Kiviat soit à l'aide d'un diagramme CityMap.



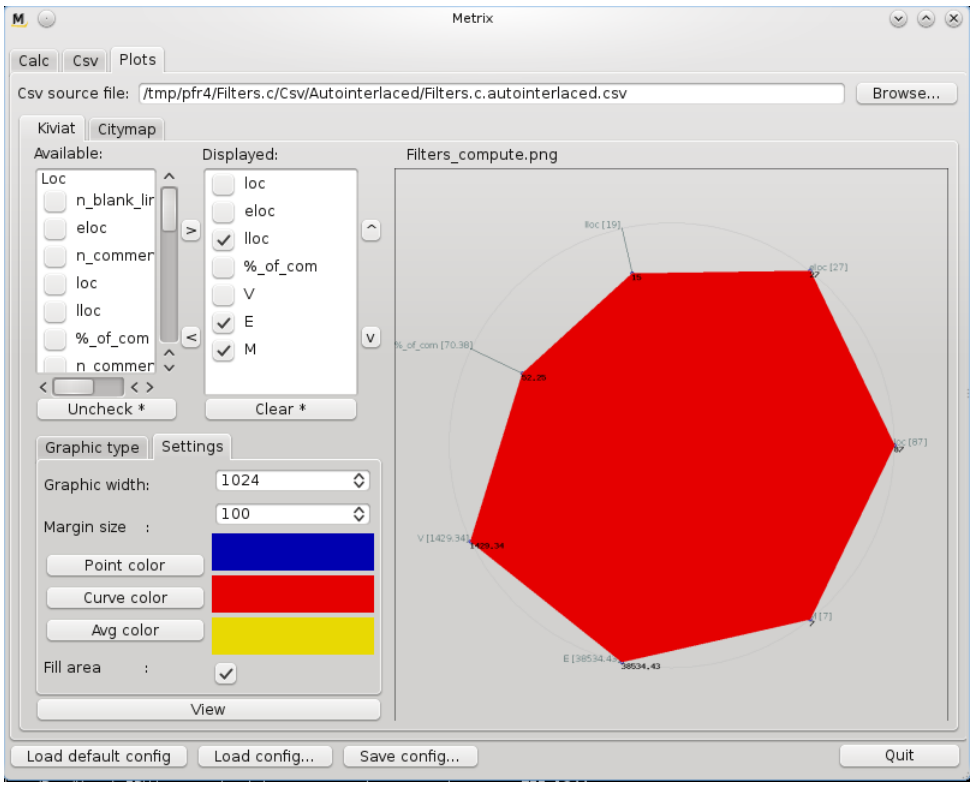

**(a)** Onglet Calc de Métrix



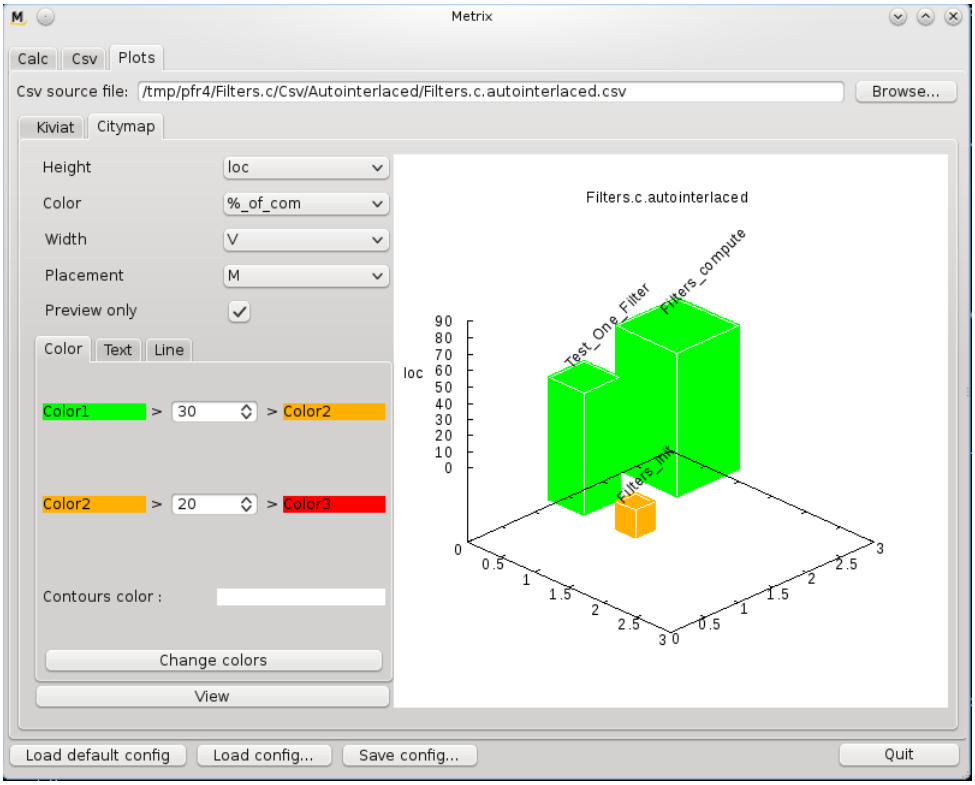

**(b)** Onglet Csv de Métrix

**Figure C.1.:** Onglets Calc et Csv de Métrix


254




**(a)** Onglet Plots de Métrix avec un Kiviat




**(b)** Onglet Plots de Métrix avec une citymap

**Figure C.2.:** Onglet Plots de Métrix : représentation graphique des résultats

1

1. Ce document a été rédigé avec LyX qui utilise LaTeX.