

# Simulation Simplifiée d'un Flux AFDX (ARINC 664)

Ethernet/AFDX Simulé

## Introduction

Ce document présente une implémentation logicielle permettant de simuler de manière simple le fonctionnement d'un flux **AFDX (Avionics Full-Duplex Switched Ethernet)**, basé sur la norme ARINC 664. L'objectif principal est de comprendre la logique des **Virtual Links (VL)** ainsi que l'encapsulation des données en trames réseau pouvant être analysées via un outil comme *Wireshark*.

Dans ce projet, j'utilisone des sockets UDP classiques.

## 1 Concept du Projet

L'architecture AFDX repose sur :

- un réseau Ethernet commuté et déterministe ;
- des flux simplex logiques appelés **Virtual Links (VL)** ;
- des contraintes de débit (BAG) et de taille (Lmax) ;
- une redondance obligatoire (réseaux A et B).

Dans ma simulation simplifiée, je ne modélise pas toute la complexité d'un réseau avionique. Toutefois, plusieurs principes clés sont reproduits :

- séparation des flux : chaque VL est identifié par un port UDP unique ;
- encapsulation personnalisée : construction d'une trame avec VL\_ID, taille et données ;
- transmission unidirectionnelle : un programme *sender* envoie des trames vers un *receiver* ;
- analyse des trames via *Wireshark*.

## 2 Architecture du Projet

Le projet se divise en plusieurs fichiers :

- `include/*` : décrit le format des entêtes de Ethernet, ip, udp;
- `afdex_trame.c` : construit une trame AFDX simulée ;
- `sender.c` : envoie une trame AFDX simulée ;
- `receiver.c` : écoute sur un port UDP et affiche la trame reçue ;

L'objectif est d'envoyer une trame depuis le *sender*, de la recevoir dans le *receiver*, puis de l'analyser dans *Wireshark* via un filtre tel que :

```
udp.port == 20001
```

## 3 Utilisation et Compilation

Compilation :

```
make
```

Lancement du receiver dans un Terminal :

```
./receiver
```

Lancement du sender dans une autre Terminal :

```
./sender
```

Capture Wireshark :

```
sudo wireshark
select loopback
udp.port == 20001 : comme filtre
```