INSAT

Mobile Devices Ingénierie

Rapport de stage d'ingénieur

Mise en place d'un service de communication au standard MQTT hébergé au sein d'un environnement kubernetes

---

Génie Logiciel

- Ihsen Charfi

- Diplôme préparé : Ingénieur en Génie Logiciel
- Encadrant de stage en entreprise : Jeremy HEATER
- Responsable de stage à l'INSAT : Riadh ROBENNA
- Durée du stage : Du 13 mars 2018 au 31 août 2018
- Date de remise du rapport : Le 24 août 2018

Remerciements

J'aimerai, au debut de ce rapport, remercier tous ceux qui ont contribué à rendre ce stage une expérience riche et épanouissante, en particulier, mon encadrant coté entreprise Jeremy HEATER et mon encadrant coté INSAT Riadh ROBENNA.

Je tiens aussi à remercier toutes les personnes qui m'acompagnaient au cous du parcours à l'INSAT, mes binomes mes amis les plus proches qui se reconnaissent et toute personne qui m'a soutenu.

Enfin, je remercie.

Table des matières

1 Introduction Générale

Paragraphe sur la télématique des voitures , la voiture connectée, la tendance des clouds et systèmes big data ...

La télématique est née de la convergence des progrès de l'informatique embarquée et des télécommunications. Dans le domaine des transports, elle désigne les dispositifs permettant la production, l'émission, la réception, le traitement et la représentation des données de manière automatisées.

2 Présentation du contexte du projet

2.1 Mobile Devices Ingénierie

2.1.1 Qui est MDI ?

Mobile Devices Ingénierie[1], un des leaders mondiaux de la télématique, est une entreprise française spécialisée dans les systèmes embarqués pour l'automobile. L'entreprise dont le siège est situé en région parisienne (à Villejuif) a depuis à peu près 15 ans, conçu et développé des boîtiers connectés aux véhicules par port OBD (On Board Diagnostics).

Actuellement, MDI comprend une cinquantaine d'employés répartis dans plusieurs pays dont la France, les États-Unis et l'Afrique du sud. Avec un chiffre d'affaire de plusieurs dizaines de millions d'euros et plus d'un million de boîtiers déployés dans le monde.

((((((((((((((((((((((((((((figure : carte du monde montrant les pays où se trouvent MdI (fr , USA , AS) et les pays ou les OBD sont déployés (les clients : (Autriche, Belgique, Bulgarie, Croatie, Chypres, République Tchèque, Danemark, Estonie, ETats unies, Finlande, France, Allemagne, Grèce, Hongrie, Ireland, Italie, Lettonie, Lituanie, Luxembourg, Malte, Pays bas, Pologne, Portugal, Romanie, Slovaquie, Slovénie, Espagne, Suède, Suisse, Grande Bretagne))))))))))))))))))))))))))))))))

Fig.1: Les pays où circulent des véhicules ayant des OBD déployés

Les clients de Mobile Devices Ingénierie sont ainsi réparties partout dans l'Europe et les Etats Unies. En effet, ils occupent différents secteurs d'activités tels que :

- Des applications embarquées de gestion de flottes de taxis

- Des compagnies d'assurances
- Des compagnies de navigation et d'info trafic
- Des revendeurs qui adaptent les produits MDI à leurs services et qui disposent d'un accès à certains marchés de par leur histoire ou leur image de marque.

2.1.2 Domaine d'activité et Stratégie

L'objectif de l'entreprise est de devenir le leader mondial pour l'acquisition, le traitement, l'enrichissement et l'échange des données techniques de véhicules connectés.

Mobile Devices Ingénierie a pour secteur d'activité la télématique embarquée pour l'automobile. Concrètement, cela consiste à proposer des solutions techniques permettant l'échange d'informations entre un ou plusieurs systèmes de gestion centralisés et une flotte de véhicule qui y est rattachée et connectée en temps réel. Ces informations sont ensuite récupérées en temps réel et peuvent être envoyées vers les serveurs de l'entreprise ou des clients.

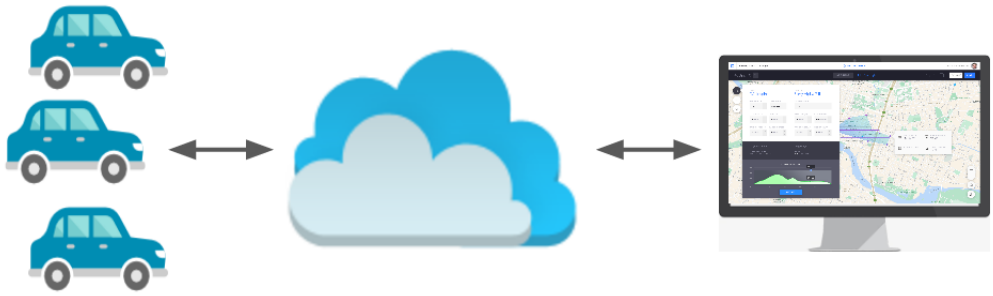

Fig.2: Schema représentant le circuit des données remontés

L'activité principale de Mobile Devices Ingénierie se résume à la mise en place de solutions techniques afin d'effectuer le suivi de véhicules et la gestion de flottes. Les produits de l'entreprise permettent de récupérer des données telles que la vitesse, la consommation, le type de conduite et même de prévenir des accidents. Ces mêmes données permettent ensuite, par exemple pour certaines entreprises, d'étudier la conduite des véhicules et leurs différents paramètres comme suivre un déplacement et/ou quantifier la consommation dans le cas d'une gestion de flotte.
Ces données peuvent aussi servir à étudier le type de conduite dans le cas d'une compagnie d'assurance. La société compte aussi d'autres activités comme la commercialisation des boîtiers pouvant offrir l'aide à la navigation servant par exemple aux Taxis. Les entreprises clientes peuvent aussi reprogrammer une partie de leurs boîtiers afin de récupérer des données spécifiques et les transférer vers des serveurs distants.

Par la nature même de l'activité, l'entreprise apporte à ses clients des solutions innovantes permettant de concilier le développement économique et la préservation de l'environnement.

En effet, ses solutions permettent de mettre en œuvre le contrôle de baisse de la consommation énergétique, de l'empreinte carbone des véhicules et du risque environnemental associé.

Dans le cadre de son développement durable, la protection de l'environnement est une préoccupation fondamentale pour MDI. L'éthique, l'équité et la diversité sont des facteurs de progrès qui contribuent à améliorer les résultats économiques ainsi que le climat social, éléments clefs d'un développement durable pour Mobile Devices Ingénierie.

2.1.3 Organisation et produits

L'entreprise suit un schéma d'organisation hiérarchique fonctionnelle composée d'une division classique du travail par fonctions. Elle est constituée d'un dirigeant et ensuite d'une équipe de collaborateurs ayant différentes fonctions : commerciale, comptable, financière, ressource humaine, production, etc. Quant au côté technique, MDI articule plusieurs types d'équipes :

- L'équipe **Production** en charge de la production et la livraison des produits
- L'équipe **Core** en charge de « Morpheus » (le système d'exploitation applicatif du boîtier)
- L'équipe **Système** en charge du système d'exploitation linux
- L'équipe **Pre-Sales** en charge du support clients et accompagement des commerciaux
- L'équipe **Serveur** en charge du cloud, au sein de laquelle j'ai effectué mon stage

Ces équipes travaillent sur différents projets autour des produits hardwares et softwares :

Les produits Hardwares	Les produits Softwares
• munic.io • municMax • OBD Dongle	• Morpheus OS • CloudConnect • CloudNext

Table 1: Tableau des produits MDI

Les produits Hardwares sont intégrés dans les véhicules et récolte des informations sur la vitesse, la position, le niveau de carburant, la pression des pneus, etc.. à un instant t donnée. En général nous appelons ces produits hardwares "les boîtiers" qui doivent être intégrés dans les véhicules et déployés dans le système cloud.

Quant aux produits Softwares, il y a le Morpheus OS qui présente le système d'exploitations des boitiers. D'autre part,les clouds CloudConnect et CloudNExt qui offrent un ensemble de services aux clients et qui sont développés et maintenus par l'équipe Serveur. On détaillera plus sur ces deux produits dans le prochain chapitre.

2.2 Présentation du projet

2.2.1 Contexte du projet

Mon sujet principal de stage est la conception et développement du nouveau composant du Cloud. Ce composant représente le point d'entrée des données au cloud. Il assure la communication entre le cloud et les boitiers. Le composat actuel porte le nom de BS - acronyme de Binary Server. Il est intégré seulement dans CloudConnect.

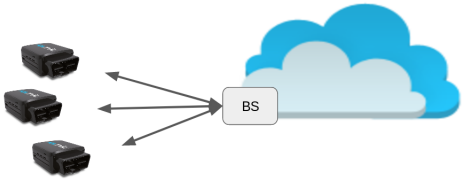

Fig.3: BS le seul point d'entrée des clouds

2.2.2 Problématique

Plusieurs facteurs rendent le développement d'un noouevau composant cloud une necessité.
Dans l'architecture actuelle, les deux cloud sont totalement indépendants mais ils reçoivent les données de la même source BS malgrès la différence entre les deux formats de messages.

D'autre part , pour une raison commercial, le nouveau protocol migre d'un protocol fait maison vers un protocol standardisé.
Un standard permet de bénéficier des librairies existantes sur le marché et donc de gagner en temps d'intégration puisque il' n y aura pas le besoin de faire du développement à zéro. MQTT n'étant pas propriétaire, les partenaires ou clients savent également comment celui-ci est conçu. Ce qui n'est pas le cas pour MD21 le protocole actuel fait maison donc cela permet de rassurer le client. Le "standard MQTT" permets également de laisser d'autres boitiers se connecter à notre plate-forme plus facilement plutot que d'implémenter notre protocole propriétaire dont ils n'ont meme pas les sources.
Donc la possibilité de gagner la confiance de plus de clients et de s'ouvrir à des marchés potentiels.

2.2.3 Objectifs

Suite à cette problématique MDI a mis en place le projet MD30 pour mettre en place la solution. Les objectifs du projet consiste à :

- concevoir et réaliser un nouveau protcole de communication entre boitier et cloud
- mettre en place un nouveau composant BS
- faire une étude de faisabilité et de performance de l'intégutilisation du protocol MQTT comme protocol de communication.
- Le support du format de données CloudNext par le nouveau BS

2.3 Méthodologies de travail

2.3.1 Cycle itératif

Le mode itératif est une méthodologie de développement différente des méthodes classiques(modèle en cascade, modèle en V). Elle tente de formaliser une méthode plus pragmatique et maniable que ces derniers.

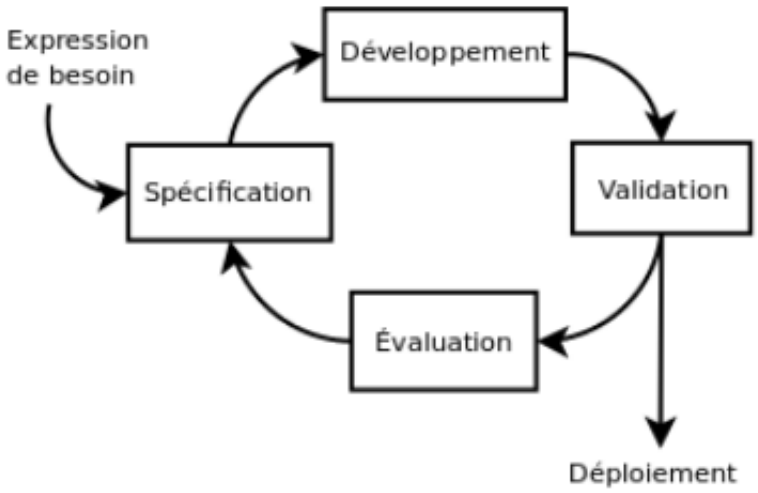

Fig.4: Le processus du cycle itératif

Cette méthode présente 6 étapes marquante :

- L'expression de besoin : où les exigences sont mis en place. L'idée reste que les informations en entrée peuvent être modifiées par la suite du processus itératif.
  Le cercle du processus itératif :
  — La spécification technique du besoin
  — Le développement : la période où on implémente la spec
  — La validation : une étape qui est marqué par les tests.C'est l'ensemble des tests qui permettent de s'assurer que le développement effectué correspond bien à ce qui était attendu.
  — L'évaluation : Cette étape sert à effectuer un retour sur les fonctionnalités laissés. Ceci sert comme des informations d'entrée pour un nouveau cycle.
- Déploiement : l'étape finale qui consiste à ce que les livrables validés sont livrés.

Ce type de cycle de développement est le plus souple de tous ceux présentés ici : chaque itération permet de s'adapter à ce qui a été appris dans les itérations précédentes et le projet fini peut varier du besoin qui a été exprimé à l'origine.[2]

Vu la nature du projet qui part plutôt vers une étude de faisabilité avec des critères de performances l'expression des besoins peuvent être redéfinies pendant ce temps. Il est préférable de suivre une méthodologie souple d'où le choix de telle méthodologie.

2.3.2 MD30 un travail d'équipe

le projet MD30 présente le nouveau protocole de communication entre les boîtiers OBD et le CLoud et donc le projet invoque les boîtiers ainsi que le point d entrée du cloud.

Le projet MD30 engage 5 personnes dans deux différentes parties :

- 2 personnes sur MD30 côté Serveur
- 3personnes sur MD30 côté Core

Même si le coeur de développement des deux parties est totalement indépendant l'un de l'autre. Les deux parties doivent rester synchroniser sur l'avancement de chaque partie afin d'avancer dans le projet en phase.

3 Etat de l 'art

Cette partie concerne plus l architecture global des clouds. Leurs évolutions au cours du temps ce qui mettra le point sur le besoin de la création d un nouveau point d entrée.

3.1 Le premier Cloud de MDI

CC est le premier cloud fait maison de MDI qui englobe un ensemble de services pour les clients... L'architecture globale de ce cloud est comme le montre le shema suivant :

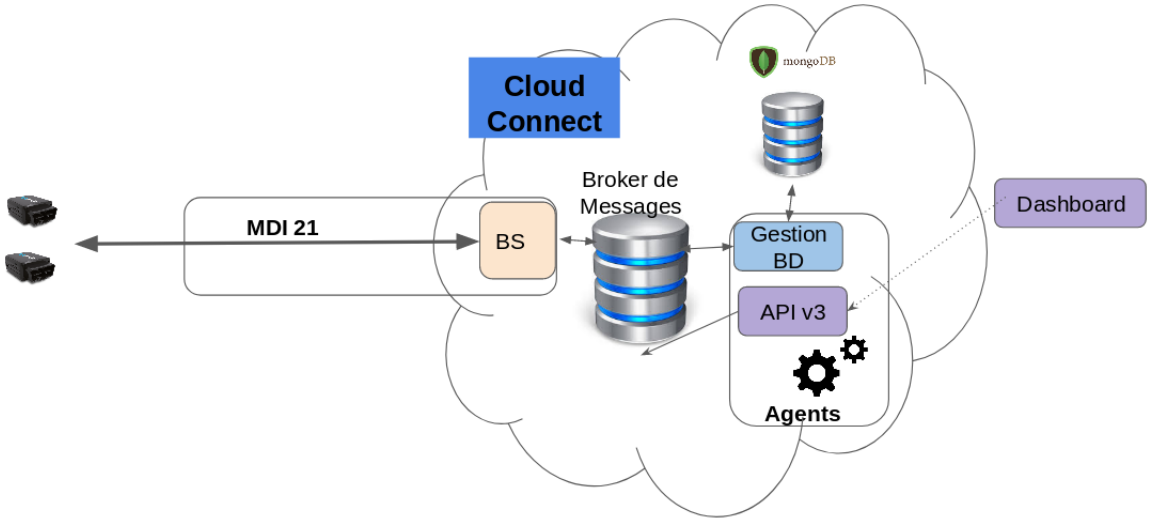

Fig.5: L'architecture de CloudConnect

CC comporte plusieurs composants :

- un BS qui est le composant d'échange de messages entre le cloud et les OBD. Il consiste le point d'entrée du cloud.
- un broker de message pour les systèmes Big Data
- un ensemble de services qui sont accéder à partir de APIV3
- une base de données pour le stockage des données
- un dashboard pour l affichage des données et l acces visuel des services au clients

MD21 est le protocole ASN1 fait maison, conçu pour la communication entre boîtiers et Cloud. //développé un peu sur MD21

3.2 De CC à CN

Exprimer le besoin du changement d'architecture. Autour de .. Développement des agents : Un Framework interne est utilisé pour simplifier le développement des différents agents en langage ruby. Ce framework fournit un ensemble de fonctions qui permettent, entre autres, d'accéder au courtier de messages en lecture et en écriture. Concernant les agents à état, le framework ne fournit pas de mécanismes pour simplifier la gestion de leurs états. La gestion de l'état est donc laissée aux développeurs.

Déploiement des agents : les agents sont exécutés de manière parallèle. En d'autres termes, un agent peut avoir plusieurs instances déployées dans différentes machines. Les différentes instances d'un même agent partagent le traitement du flux des événements mais ne communiquent pas entre eux. Le déploiement ainsi que la maintenance d'un agent se traduisent par une mise-à-jour des fichiers de configuration des machines exécutant l'agent via le logiciel Chef.

Solution : Création d'un autre Cloud afin de revoir l'architecture logicielle et technique de Cloud Connect afin de simplifier le développement des agents, leurs déploiements ainsi que leurs maintenances et de permettre à des développeurs externes de l'entreprise de coder leurs propres agents.

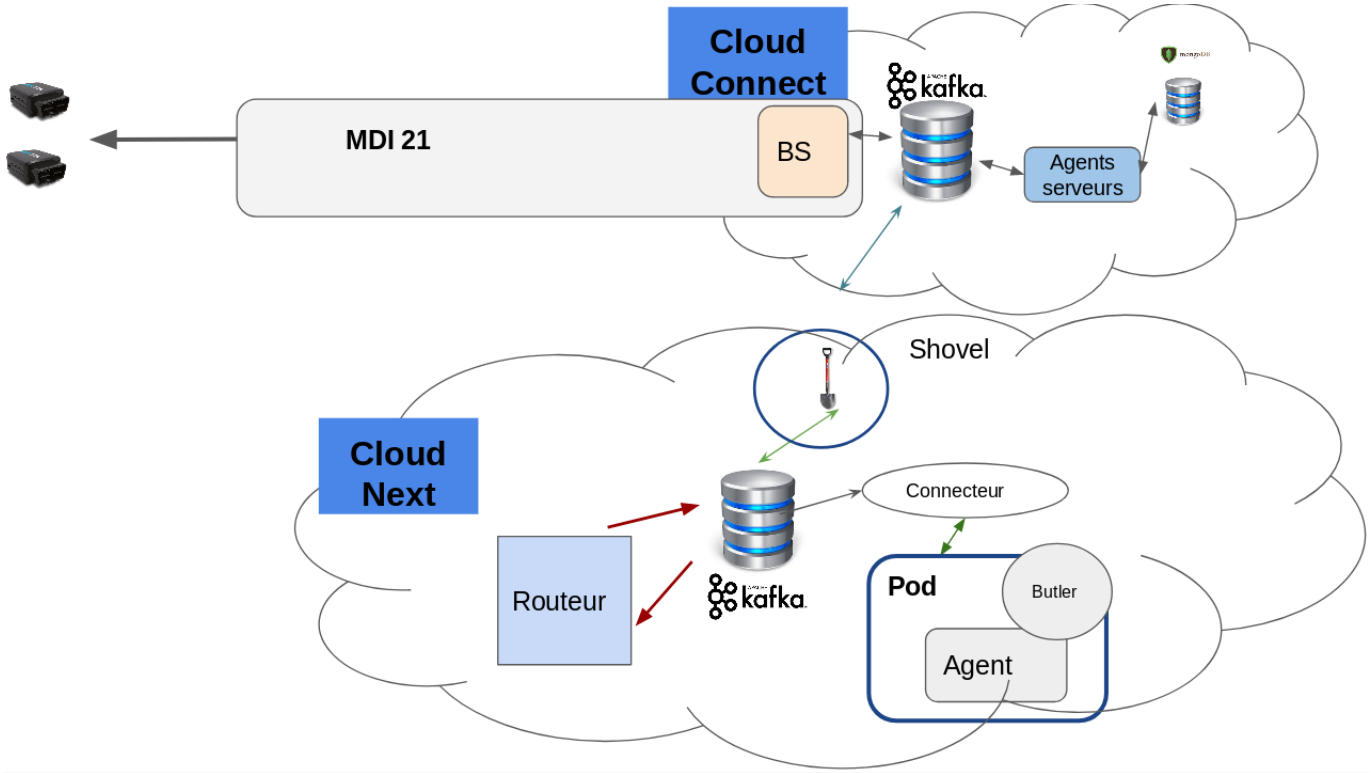

Fig.6: L'architecture du cloud avec CloudNext

3.3 format de données du cloud

Track : Message :
message track

Paragraphe sur le language Go : Go[**rust_site**] est un jeune language de programmation système fascinant. Ce language compilé hérite des idées des paradigmes de programmation impérative et fonctionnelle. Il définit des concepts et des règles qui assurent à l'utilisateur des binaires "safe".

4 itération 1 : nouveau BS au standard MQTT

4.1 Conception de la première itération

MDI, étant une entreprise innovante en télématique, doit toujours suivre et avancer les nouvelles technologies. Une de ces technologies est le protocol de communication standar des objets connectés : le MQTT.

... Le protocole- Vehicle to Everything - donne au véhicule la connaissance de son environement en collectant les informations des autres utilisateurs de ce protocole, et contribue aux connaissances collectives en diffusant les siennes.

Actuellement, MDI utilise une stack V2X developpée par Marben[3]. Cette stack, écrite en C++, ne répond pas aux demandes de performance de l'entreprise, et introduit une dépense évitable en coûts de license.

Le but de ce projet est de développer une stack V2X pour remplacer celle de Marben, et qui procure les performances nécessaires pour tourner sur les boitiers. Ce sujet ne sera pas trop détaillé vu qu'il a été abandonné par l'entreprise.

4.2 Descriptif du protocole :Le standard MQTT, l'essentiel à savoir

Voir les couches de MQTT Ce protocole s'intéresse à toutes les couches de communication du modèle OSI. Les spécifications exactes diffèrent entre l'Union Européenne, les Etats Unis, et le Japon.

4.2.1 La couche physique

v2x définit un protocole dérivé du wifi : le 802.11p, nommé WAVE. Ce protocole plus lent, mais plus adapté sert aux communications entre véhicules à haute vitesse, et entre ces véhicules et des stations intelligentes. Les connexions 3G et 4G ont l'inconvenient d'une grande latence, et sont donc éliminées comme vecteurs pour le protocole. En effet, V2X a besoin d'établir des connexions directes entre véhicules, pour assurer une latence petite, et bornée. Une communication par 5G, la C-V2X (Cellular V2X), est envisagée vers 2025 : les specs du 5G répondent aux demandes de latence pour V2X. Aujourd'hui, MDI utilise le module Vera-P174[4] pour cette couche.

4.2.2 La couche transport

V2X utilise les protocols TCP et UDP sur IPv6. Il utilise en plus le protocol WSMP[5]. Dans un premier temps, MDI s'intéresse à un modèle sans WSMP. Actuellement, la couche transport est implémentée par la stack Marben.

4.2.3 La couche applicative

La couche applicative de ce protocole consiste en un nombre d'applications essentielles et non essentielles. Les différentes stations V2X, ou station ITS, communiquent, en broadcast, plusieurs types de messages, dont les plus importants sont les CAMs et les DENMs.

Les CAMs signalent la présence de ces entités, et les DENMs notifient au cas d'un évenement. D'autre applications V2X peuvent être construites au dessus de ces applications de base.

4.3 Specifications techniques

La spécification téchnique peut être divisée en deux parties :

- La spec technique du protocole V2X[6].
- La spec ajoutée par MDI pour une stack sur ses dongles.

Tandis que les spécifications de la ETSI sont longues et restrictives, les spécifications additionnelles de MDI sont très simples : faire mieux que Marben ! Jeremy HEATER comptait six mois pour ce projet, pour aboutir à un prototype minimal.

4.4 Implémentation

Imlémenter c est faciiile ! !

4.4.1 Recherche & état de l'art

Une première étape dans ce projet, fut de passer sur toutes les recherches antérieures de l'entreprise sur le sujet. J'ai donc lu un grand nombre de spécifications V2X, ainsi que la documentation technique de la stack Marben. Celle-ci était intégrée avant le début de mon stage. J'ai lu, de même, la documentation du SoC Vera-P174[4], l'unité physique utilisée par MDI pour faire du V2X. Son utilisation est déléguée à la stack Marben. J'ai fait le "bring up" du composant et utilisé sa CLI pour faire des tests et se familiariser avec.

Fig.7: Prototype llc pour V2X - Vera P174

4.4.2 Développement effectué

Le développement de ce projet est divisé en deux parties :

- Développement des machines à états, un service Morpheus en Java.
- Développement de la stack qui remplacerait Marben, en Rust.

Les résultats de tests antérieurs avec VolksWagen étaient positifs pour les CAMs. Par contre, la gestion des DENMs nécessite des machines à états ne faisant pas partie de la stack Marben. La stack est, en fait, capable d'envoyer et recevoir les DENMs, mais pas de gérer leur logique. J'ai commencé donc le développement du service Morpheus contenant la logique de gestion des DENMs, comme décrit dans la figure 8.

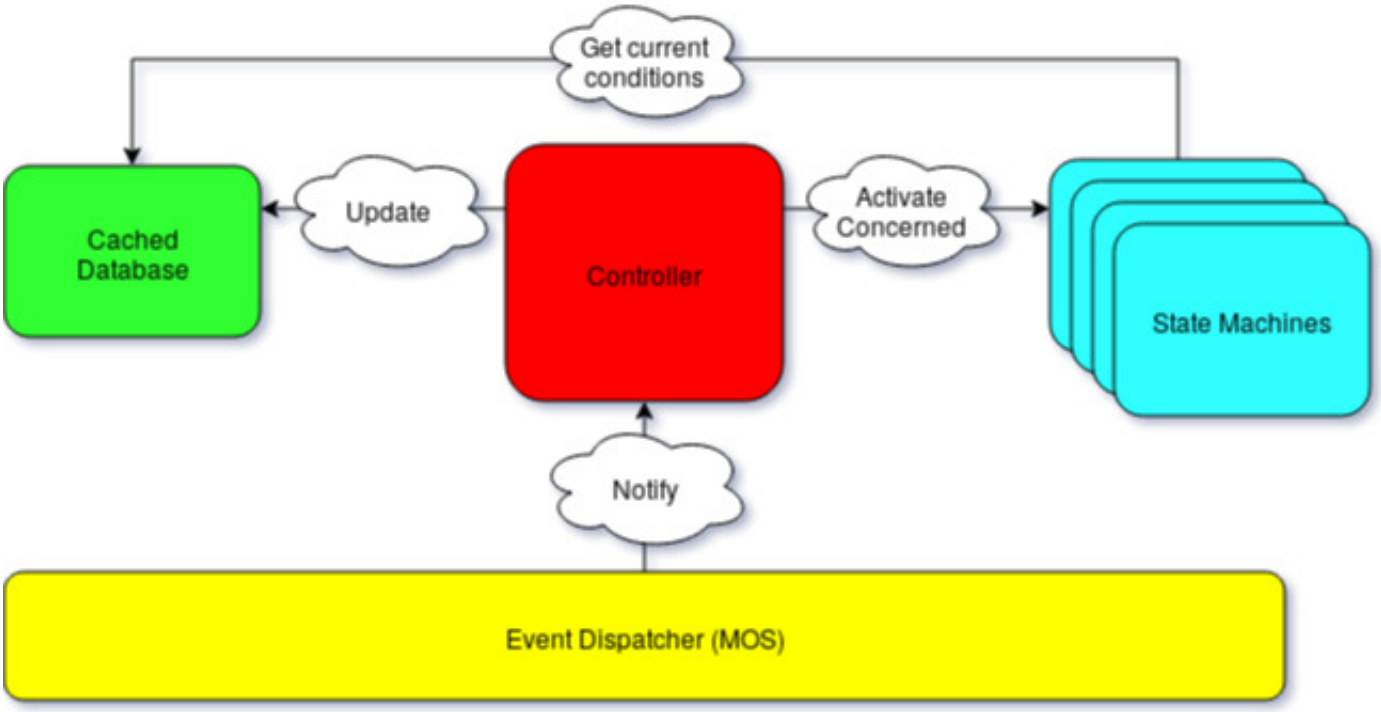

Fig.8: V2X State Machines

Quand un paramètre d'interêt pour une de nos machines change, le controlleur est notifié. Il met à jour son cache, et active les machines à états concernées. Celles-ci effectuent leurs transitions, et déclenchent les actions correspondantes.

4.5 Tests & Performance

Un déplacement en Allemagne, à Wolfsburg, nous a vu avancer dans ce projet. Malheureusement, il arrive à sa fin prématurée quand VolksWagen nous annonce que l'entreprise n'investira pas de ressources dans le projet avant une ou deux années. Le projet est alors marqué comme un succès, mais arrêté jusqu'à nouvel ordre.

5 Itération2 : nouveau BS englobant TCP serveur

5.1 Conception

Cette partie repose sur la modification des exigences et la nouvelle Conception

5.1.1 L'évolution des exigences

Après avoir fait l'évaluation des résultats de performance de la 1ere itération du projet, on a redéfinit les spécifications du projet. Le tableau suivant montre la différence des exigences entre les deux itérations :

Les changements des exigences			
	Itération1	Itération2	Avantage de cette évolution
Protocol de communication	MQTT	TCP avec le support de MQTT	Moins coûteux par rapport aux paquets échangés, tout en gardant la possibilité de connecter le boîtier à un serveur MQTT
Architecture du BS	Un cluster de brokers MQTT avec des subscribers des tracks et publishers d'events.	Un cluster de TCP serveur qui gèrent les connexions, traitent les message échangés et envoient/reçoivent les msg à/de kafka.	Moins coûteux par rapport aux instances à gérer. Plus facile à gérer un seul composant au BS. Pas de complexité à dispatcher les subscribers quand on monte de charge.

Table 2: Tableau de comparaison des exigences du projet

L'une des majeurs modifications est d'abandonner l'intégration du standard MQTT tel qu il est . On ne garde que le Header de MQTT comme le Header des messages échangés avec le nouveau tcp serveur. Ceci a pour but de garder la compatibilité avec MQTT pour satisfaire les besoins commerciaux.

5.1.2 Architecture de l'itération 2

L'architecture du nouveau BS est conçue d'une façon à ce qu il présente une solution aux problèmes rencontrés par l'architecture de la 1ere itération. La figure ci dessous montre les composants de la nouvelle architecture :

Le BS repose sur 3 grands composants : En premier temps, le TCP serveur qui gère la connexion avec le boitier. En deuxième temps,le traitement des données doit être gérer en parallèle. Ce traitement assure l'encodage du protobuf boitier au protobuf cloud et vis-versa.

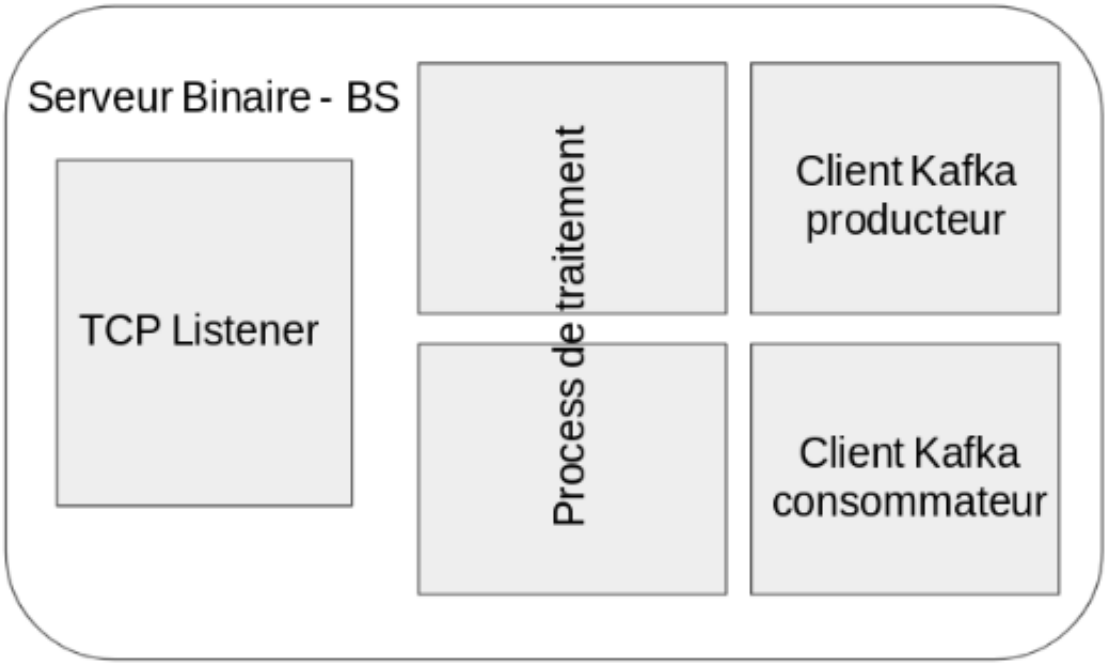

Fig.9: La conception de l'architecture du BS de l'itération 2

D'autre part, deux clients kafka, un producteur qui va publier les messages encodés et un autre consommateur qui va recevoir les messages d event.

5.2 Implémentation

Le nouveau BS lance les diverses traitements dans des traitements en parallèle. L'implémentation de ces processus parallèles s'effectuent grâce aux goroutines du langage go, ce qui englobe toutes les fonctionnalités du BS dans un même composant unifié.

5.2.1 Control plane

Concernant le flux de données ou Control plane, la communication entre ces processus se fait comme le présente la figure ci-dessous :

* Le boitier envoie un track pour le cloud :

La première connexion se fait avec le TCP Listener. Comme son nom l'indique, ce composant écoute, vérifie et maintient la connexion avec les boîtiers. Il est considéré comme le Générateur. Après avoir vérifier la connexion avec un boitier donné, il envoie le message reçu au processus de traitement dans un channel.
Ce dernier décode le format du message et l'encode en format des données du cloud. Ensuite il le met dans un buffer des messages prêts à la publication kafka. Puis le client kafka envoie tous les messages du buffer d'une manière périodique de quelques secondes.
L'envoi des messages d'une manière périodique minimise l'accès écriture dans la BD ce qui augmente l'efficacité du traitement global.

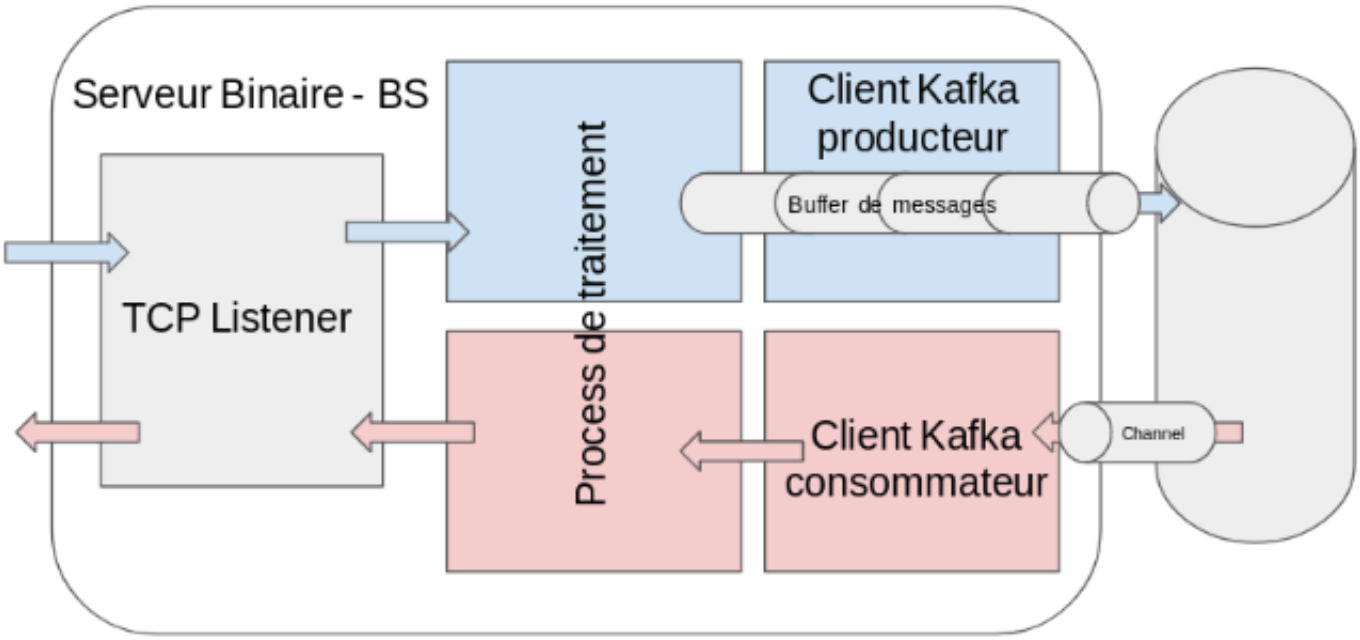

Fig.10: Le flux de données du BS de l'itération 2

* Le cloud envoie un event au boîtier : Le service qui a déclenché l'event va publier dans le broker des messages le message dans un topic spécifique. Le client consommateur est toujours en écoute sur ce topic, il reçoit le message et l'envoie directement au traitement. Le message s'encode avec le format protobuf du boitier est maintenant prêt à être envoyer au boîtier. Le TCP listener s'occupe alors de son envoi à son boitier.
Il est à noter que l'implémentation de cet envoi par rapport à plusieurs instances sera plus complexe.

- **Explicit Mutability** : Une variable est, par défaut, immuable (invariable ? :)). Pour la faire varier le code doit la déclarer explicitement mutable, avec le mot clé *mut*.

```
let a = 5;
a = 3; // Error, a not mutable
let mut a = 5;
a = 3; // OK
```

- **Ownership** : C'est un système de manipulation des données basé sur le fait que chaque donnée a un propriétaire. Elle ne peut pas avoir plusieurs propriétaires ; par contre elle peut changer de propriétaire, être consommée en quelque sorte ; cette opération s'appelle un move. Elle est effectuée sur les types qui n'implémentent pas le trait Copy.

```
let a = SomeStruct::new();
let b = a;
a.use(); // Error, use after move
```

- **Borrowing** : Une valeur *owned* par une certaine variable peut être "prêtée", par réference. Cela s'appelle le borrowing et a des règles très strictes, enforcées par le

BorrowChecker. Une variable peut avoir une réference mutable ou plusieurs références non-mutables.

```
let a = SomeStruct::new();
let b = &a; // Ok
let mut c = &a; // Ok, the borrow itself is not mutable
let d = &mut a; // Error, immutable borrow still in context
```

Le but de cette explication est de montrer la différence considérable avec d'autres langages de programmation. Un lecteur intéressé pourra apprendre le langage pour plus d'informations.

5.3 Implémentation

5.3.1 Go un language pas POO

Les bugs dans le code arrivent même avec les programmeurs les plus aptes. Mobile Devices Ingénierie ne fait pas exception à cette règle. Dans l'effort de minimiser ces bugs le plus possible, MDI a fait le choix de migrer vers Rust. Trois nouveaux projets sont développés en Rust :

- MD30, Un nouveau protocol de communication dongle-cloud
- MEP, Morpheus event processor, un outil de programmation graphique
- MSP, Morpheus signal processor, un moteur de traitement de signaux

Go présente un grand nombre d'avantages, mais aussi un grand nombre d'inconvénients. Pour une personne qui fait de la programmation orientée objets, s'adapter n'est pas trivial. Mais une fois on s'habitue aux dynamiques de ce languages, on retrouve qu'il est assez flexible.

Par contre, il faut s'attendre à se heurter à des bugs, de.... De même il faut être prêt à implémenter des bibliothèques rudimentales,..

Ci-dessous un bilan des principaux avantages et inconvénients de Rust par rapport à autres languages, notament C++.

5.3.2 Avantages

- Garanties thread-safety
- "Si ça compile, ça marche"
- Communauté active
- Bonne gestions de cas, pas de comportement indéterminé
- Performances très bonnes
- Gestion facile de bibliothèques (Cargo)
- Cross-compilation facile (Cargo)
- Evolution du language

5.3.3 Inconvénients

- Instabilités compilateur, parfois dans la chaine stable

- Manque d'expertises Rust en entreprise
- Known issues sans perspectives de fixes
- Manque d'optimisations spécifiques aux processeurs
- Complexité de codes simples dans autres languages
- Pas de programmation orientée objets
- Dynamic dispatch très limité
- Chaine de compilation non-optimale pour notre processeur
- Communauté très jeune
- Bibliothèques essentielles absentes

5.4 Test

Apprendre Rust était une très belle expérience. La réference principale pour moi était le Rust book[7]. J'ai lu ce livre entier, et suivi les examples. Il couvre assez succintement les détails du language.

Le Rustonomicon[8] m'a servi de réference pour comprendre des concepts sur le fonctionnement interne du language. Évidemment, pour profiter de Rust il m'a fallu éviter, le plus possible, les syntaxes **unsafe**. En effet, je n'en ai eu besoin que pour les s, et l'intégration avec .. Mais ce livre m'a aidé à comprendre un certain nombre de problèmes auquels j'ai fait face, notament :

- La notion de variance de structures[9].
- Les erreurs de lifetimes
- Coercions de types
- Le unwind de threads
- Les phantom data
- Drop et l'utilisation de **Option** dans des cas spéciaux

Il faut aussi dire que le bagage fourni par le cours de compilation à INSAT était d'une utilité exceptionnelle, surtout que programmer en Rust, c'est virtuellement une conversation avec le compilateur !

Mais la piste principale pour l'apprentissage reste la pratique. Et avec le projet MSP j'ai eu la chance de faire des fautes, réitérer, apprendre des concepts et des patterns, et corriger, dans un cycle qui a duré 4 mois. Les détails du code seront discutés dans la section concernant ce projet.

5.5 Amélioration à effectuer

Data plane .. Rust étant un langage très jeune, j'ai saisi l'opportunité pour contribuer à son développement, que ce soit en discutant avec la communauté, et en proposant des idées dans les issues (sur github), ou en publiant des crates sur crates.io[10]. Malgré mon désir de le faire, je n'ai pas eu le temps, jusqu'aujourd'hui, de contribuer au projet du language même.

Durant le projet MSP, j'ai publié le crate "ts-mem-pool"[11], développé dans le cadre de ce

projet. C'est une simple implémentation de memory pool léger, générique et avec des élements thread-safe. J'ai aussi écrit, avec un collègue, un binding en Rust pour la bibliothèque iir1[12], qui attend encore sa publication.

6 Conclusion

Une conclusion globale sur tous le projet, l experience ...

Références

[1] *www.mobile-devices.com*. Site officiel de Mobile Devices Ingénierie.

[2] *www.geek-directeur-technique.org*. Documentation technique sur le cycle itératif.

[3] *www.marben-products.com*. Site officiel pour les produits Marben.

[4] *www.u-blox.org*. Documentation technique de la serie Vera-P1.

[5] *www.indjst.org*. WAVE and WSMP description.

[6] *www.etsi.org*. Cet article définit les normes du protocole V2X.

[7] *Rust book*. The Rust Programming Language Book.

[8] *Rustonomicon*. A book about unsafe and advanced Rust.

[9] *Data structure Variance*. Article wikipédia sur la variance en informatique.

[10] *www.crates.io*. Répertoire principal des crates Rust.

[11] *ts-mem-pool*. Crate Memory pool développé pour MSP.

[12] *LibIir1*. Une bibliothèque contenant des implementations de filtres IIR divers.

Glossaire

BS Binary Server. 5, 8

CAM Common Awarness Message. 11–13

CC CloudConnect. II, 8, 9

CN CloudNext. II, 9

DENM Decentralized Environmental Notification Message. 11–13

ETSI European Telecommunications Standards Institute. 12

ITS Intelligent Transportation System. 11

MD21 Mobile Device 21 : le protocole de communication actuel de MDI. 5, 8

MD30 Mobile Devices 30 : le nom du nouveau protocol de communication . 5

MDI Mobile Devices Ingénierie. II, 2, 4, 5, 8, 11, 12, 17

MSP Morpheus Signal Processor. 18

OBD ON Board Diagnostics. 4

V2X Vehicle to Anything communication. 11, 12

WAVE Wireless Access in Vehicular Environments. 11

WSMP WAVE Short Message Protocols. 11