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 27 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

Table des figures

Liste des tableaux

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.

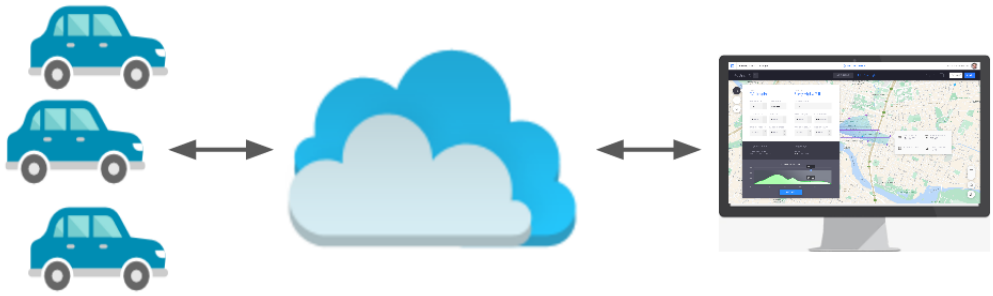

Fig.2: Schema représentant le circuit des données remontés

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.

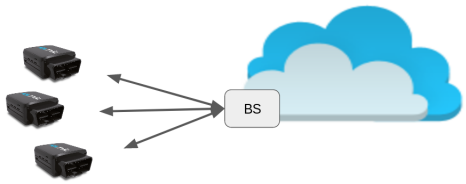

Fig.3: BS le seul point d'entrée des clouds

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. MD30 sera le nouveau protocol de communication OBD-cloud qui a comme objectifs :

- 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.

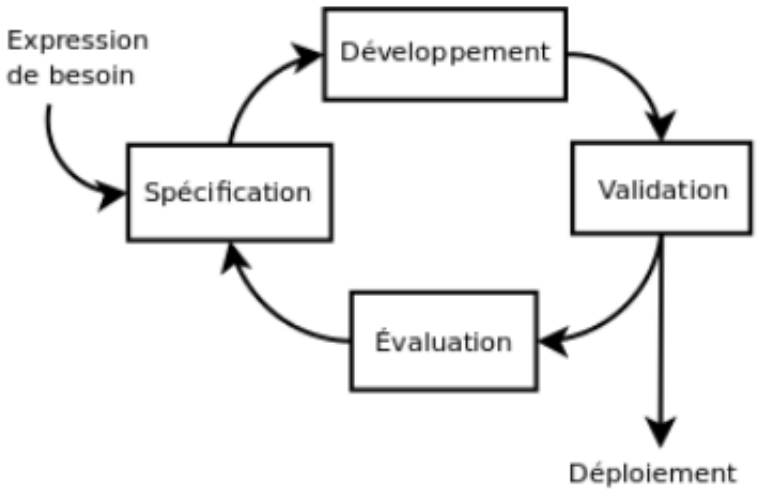

Fig.4: Le processus du cycle itératif

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 des méthodologies : 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 est un travail d'équipe suivi

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. C'est pour cela qu'une réunion hebdomadaire de synchronisation s'effectue chaque fin de semaine. Le suivi de ce projet se fait à travers la création des ticket en utilisant l'outil Asana.

Asana est un outil de gestion de projet web et mobile qui donne une visibilité sur .....
figure : What asana dashboard looks like

3 Etude de l'exisant

Cette partie concerne le plus sur l'architecture globale des serveurs clouds de MDI, leurs évolutions au cours du temps ce qui éclaircirera le besoin de la création du nouveau composant point d entrée.

3.1 Evolution des clouds MDI

Les boîtiers OBD ont la particularité de s'intégrer dans tout type de véhicule et permettent l'émission, la réception, le traitement et la représentation de données diverses comme l'état de la batterie du véhicule, les données standards, les données du propriétaire du véhicule ou des données des capteurs du boîtier même (localisation GPS, l'accéléromètre, le gyroscope, la consommation de la batterie ...)et un tas d'autres informations complémentaires. Ces données là ont besoin d'être traité pour pouvoir en sortir de l'information utile, d'où le besoin de services divers, regroupés dans un cloud.

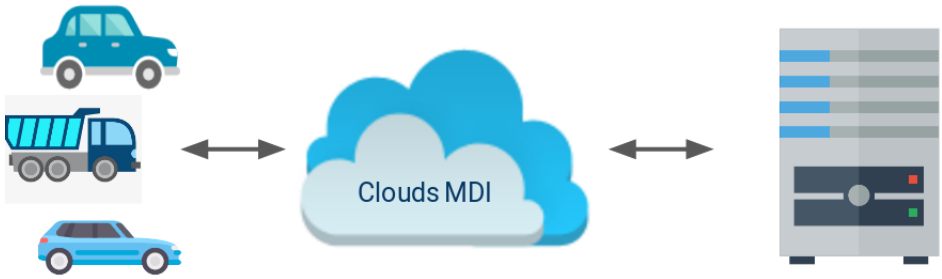

Fig.5: Communication avec le cloud

Le Cloud collecte les informations des véhicules récoltés par ces produits intégrés, traite et stocke ces derniers et communique avec les serveurs clients externes.

3.1.1 Le premier Cloud de MDI

Le CloudConnect - ou CC - est le premier cloud fait maison de MDI qui englobe un ensemble de services pour les clients. Créé en ..et hebergés dans des serveurs d'OVH, CC traite jusqu'à présent un nombre important du traitement des données. CloudConnect propose également un écosystème d'applications partenaires (gestion de flottes, ecodriving...) mais aussi des services connectés par l'intermédiaire de partenaires.

L'architecture globale de ce cloud est présenté par le shema de la figure 6 de la page 9.

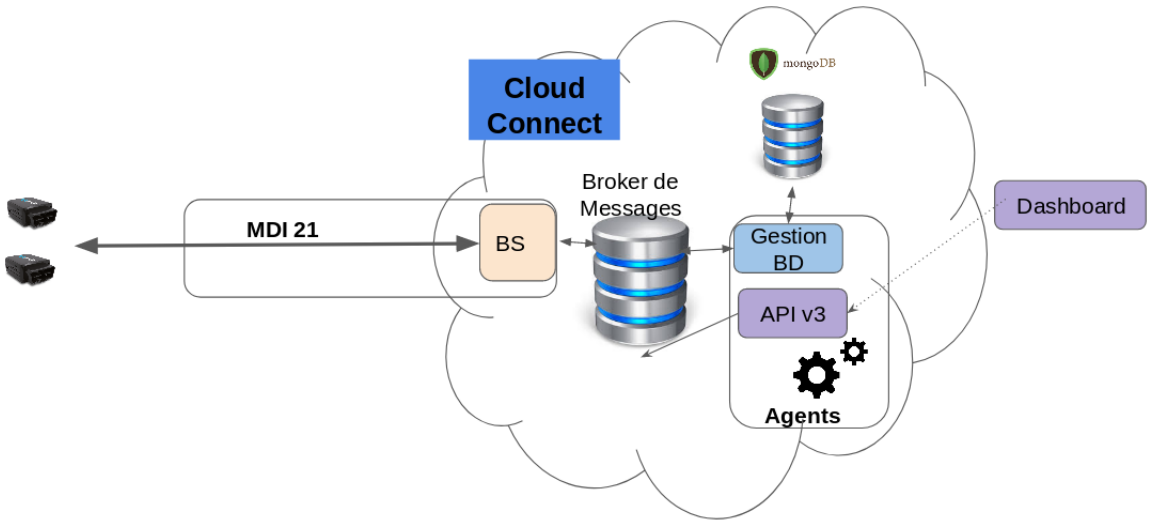

Fig.6: L'architecture de CloudConnect - CC

Pour des raisons de confidentialité ce schema a été simplifié. Néanmoins, il présente les principaux composants de CloudConnect. Parmi ces coposants on trouve :

- un **BS** qui est le composant d'échange de messages entre le cloud et les OBD. Il présente le point d'entrée du cloud et représente le composant en question. Il supporte MD21, le protocole de l'entreprise conçu pour la communication entre boîtiers et Cloud. Nous détaillerons bien ce composant à la fin de ce chapitre.
- un **broker de messge :** pour les systèmes Big Data. C'est un système distribué très répandu dans le domaine de BigData centralisant le flux de données provenant des différents systèmes de l'entreprise tout en gardant le découplage entre les systèmes producteurs de ceux consommateurs de données.
  Le point fort de cet outil est sa capacité de supporter des débits très importants que ce soit pour la publication ou la lecture des données. D'autres part, ce broker permet la consommation des données en temps réel et en mode batch, en persistant les données. Il est à noter que les données ne sont pas stockés dans ce système, ils sont là jusqu'à leurs consomattion. D'où le besoin d'une base de données.
- une **base de données NoSQL** orienté documents pour le stockage des données jusqu'à 3 mois.
- un ensemble de services qui sont accéder à partir de **API-V3** :
- un **dashboard** pour l'affichage des données et l acces visuel des services au clients

Il est bien de noter que l'équipe serveur de MDI veille à ce que sa politique de récupération des données respecte la nouvelle réglementation de GDPR[3].

3.1.2 De CC à CN

Bien que le CloudConnect parraît complet. Il présente quelques problématiques au niveau 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.

Comme solution à ces problèmes, MDI a eu recours à la création d'un autre Cloud afin de revoir l'architecture logicielle et technique de CloudConnect 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.
L'entreprise a créé alors sa propre infrastructure adaptée aux données de la télématique appelée « CloudNext » - ou CN dont l'objectif est de se diriger vers une infrastructure orientée containers qui rend plus facile l'intégration de solutions développées par les clients de l'entreprise et permet de diversifier ses activités.

L'architecture simplifié du CN est présenté par la figure 7 dasn la page 11.

Dans le nouveau cloud on trouve d'autres composants nouveaux. Parmi eux on trouve :

- **Shovel :** Il traduit les informations du format "MessagePack" au format "Protobuf".
- **Routeur :** Il assigne chaque track à un agent.
- **Connecteur :** Il gère les états des stateful agents
- **Pod :** Les pods de Kubernetes
- **Butler :**
- **Agent :** Un service Cloud

Pour faciliter la communication entr les agents, CC migre vers un format de message de

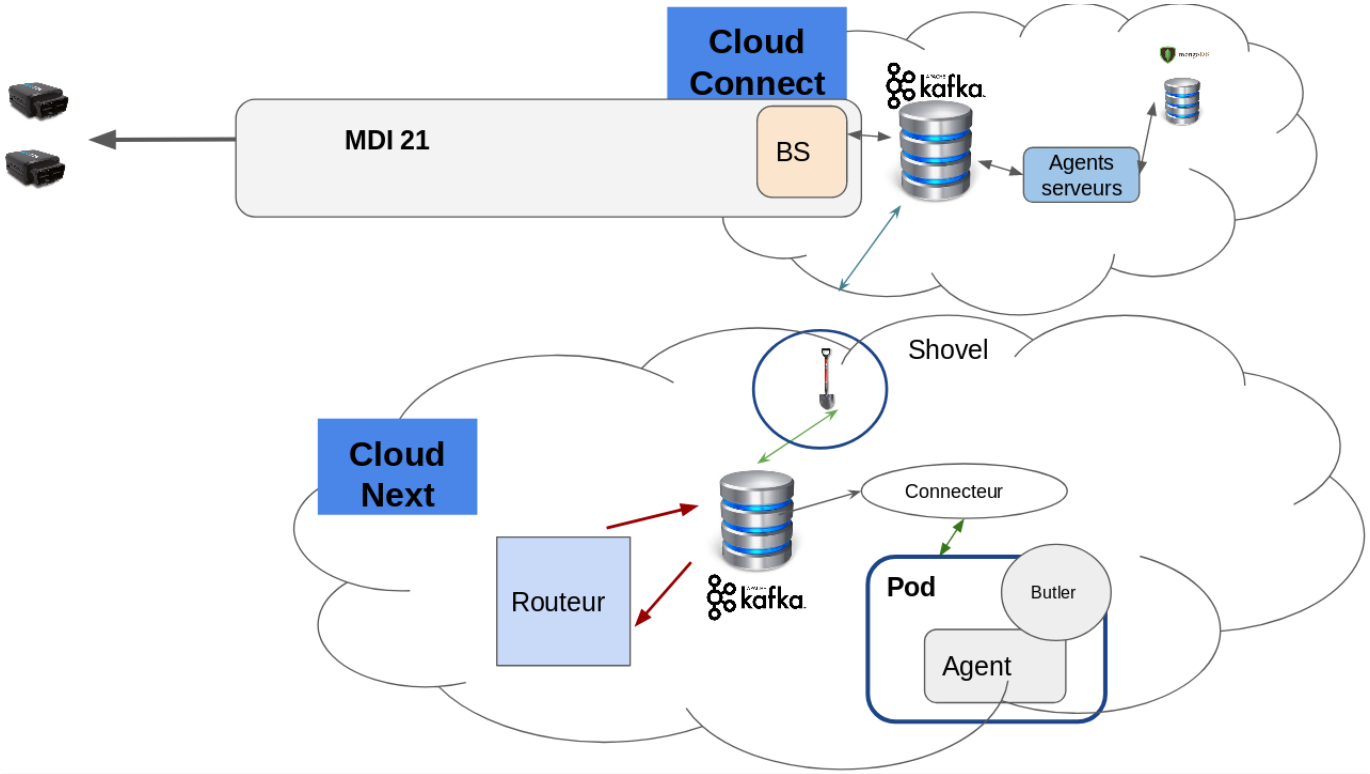

Fig.7: L'architecture du cloud avec CloudNext

MsgPack vers ProtoBuffer

**********Expliquer différence entre MsgPack et ProtoBuffer *******

3.2 le BS MD21 de plus pres

3.2.1 technologies

Redis , Indigen ..., fonctionnalités
Cet agent est codé en Erlang, c'est un problème pour la garantie de la maintenance future. Erlang perds de son intérêt pour 2 raisons : L'élasticité de la VM n'est plus utile avec Docker et Kubernetes Le chargement à chaud des nouvelles versions et la coexistence avec les anciennes version est aussi disponible avec Docker et Kubernetes

Le BS ecrit dans RQueue et le Shovel copie de RQueue vers Kafka. donc il n'écrit pas directement dans le broker de message de CloudNext.

Les informations de session sont dans redis (socket permanente). Les informations statiques (fields, channels, paramètres des accounts) sont récupérés depuis les apis (web services) - c'est une action rare. De plus les changements en DB sont notifiés au BS via Kafka.

3.2.2 Format de données

Les informations remontées par les boîtiers sont considérées comme des évènements. Ces évenement peuvent contenir des tracks , des messages et/ou des présences.

- **Track :** informations géolocalisées envoyés depuis le boitier au Cloud et qui sont les données remontés des capteurs traités par les OBD.

- **Message :** informations plus compléte qui se dirige vers un channel précis, invoqué depuis le serveur vers le boitier.

- **Presence :** état du boitier, si il est connecté au serveur ou pas.

3.2.3 Transmission boitier-serveur

La transmission se fait en utilisant le protocole MD21. MD21 est une logique in-house de séquence ainsi qu'un encodage décrit en ASN.1
Ce protocole fonctionne en TCP et en UDP. L'UDP n'est que rarement utilisé étant donné que les opérateurs téléphonique ne le favorise pas, voire rendent son utilisation impossible correctement. L'utilisation d'UDP nécessite en parallèle un lien TCP.
Le lien est à l'initiative du boîtier,c'est à dire que le boitier qui doit se connecter en premier au serveur en passant par le BS La sécurité du lien est demandée par le boîtier au serveur et supporte les modèles récents (TLS 1.2).
Une limitation actuelle, c'est que les transaction ne sont pas identifiées et que le serveur ne peut donc pas indiquer au boîtier un index unique du dernier track traite. De ce fait, le boitier re-envoi tous les track qui n'ont pas été acknowledged et on a des rejeu de données.

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 standard des objets connectés : le MQTT.

...

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.

4.2.1 La couche physique

v2x définit un protocole dérivé du wifi : le 802.11p, nommé 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, 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 pour cette couche.

4.2.2 La couche transport

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

4.3 Specifications techniques

La spécification téchnique peut être divisée en deux parties :

- La spec technique du protocole
- La spec ajoutée par MDI pour adapter

4.4 Implémentation

Imlémenter c est faciiile ! !

4.4.1 Recherche & état de l'art

4.4.2 Développement effectué

Le développement de ce projet est divisé

4.5 Tests & Performance

Paragraphe sur le language Go : Go 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 .

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 Itération2 : nouveau BS englobant TCP serveur

5.1 Spec

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.

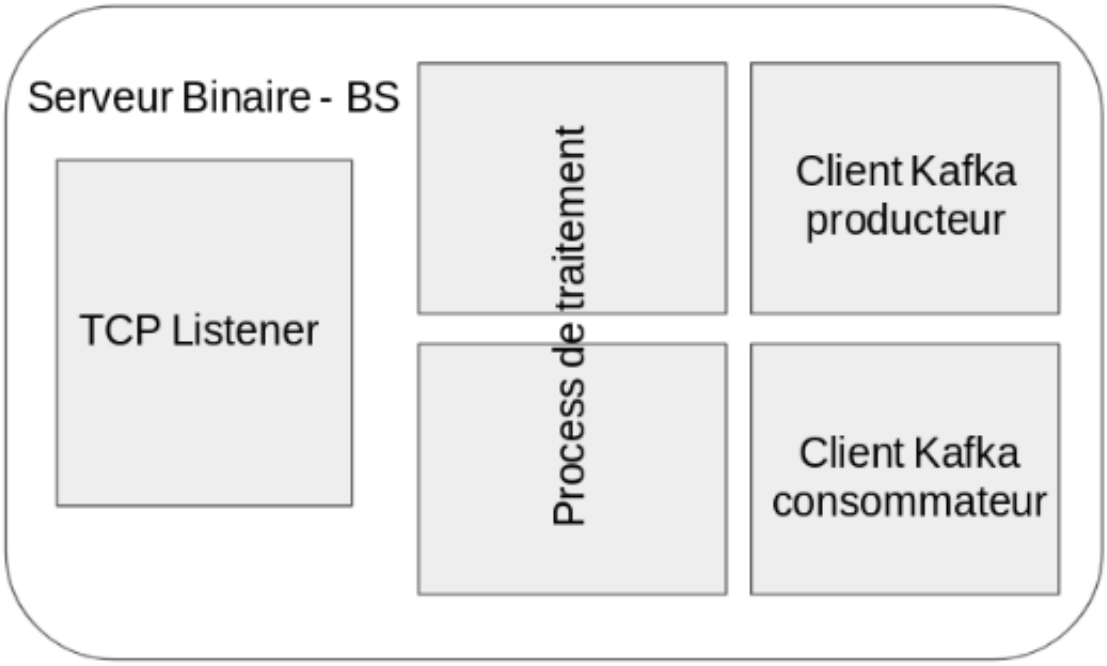

Fig.8: La conception de l'architecture du BS de l'itération 2

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.

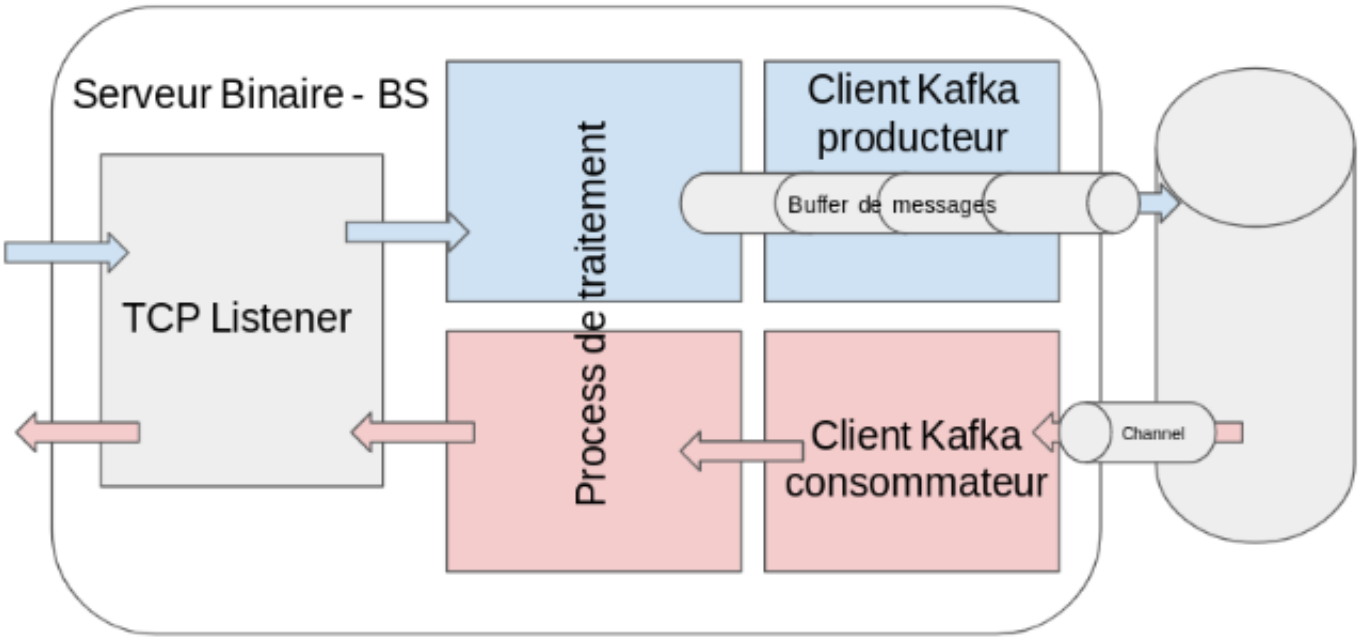

Fig.9: Le flux de données du BS de l'itération 2

* 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.

5.2.2 Go un language pas différent

Avantages
- basé sur les librairies basiques
- développé par Google donc un bon support
- Evolution du language

Inconvénients
-
- Pas de programmation orientée objets ou support pas standard

5.3 Test

Apprendre Go était une très belle expérience. La réference principale pour moi était .... m'a servi de réference pour comprendre des concepts sur le fonctionnement interne du language. Mais ça m'a aidé à comprendre un certain nombre de problèmes auquels j'ai fait face, notament :

- La notion de variance de structures[**variance_wiki**].

- 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 Go,

Mais la piste principale pour l'apprentissage reste la pratique. Et avec le projet MD30 j'ai eu la chance de faire des fautes, réitérer, apprendre des concepts et des patterns, et corriger, dans un cycle qui a duré 5 mois.

5.4 Amélioration à effectuer

Data plane ..

6 Conclusion

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

7 Perspectives

Le développement du projet demande encore du travail à fourninr et des " features" à développer.

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.gdpr-info.eu*. Documentation officielle de la réglementation générale sur la protection des données.

Glossaire

ASN.1 Abstract Syntax Notation One: Syntxe de grammaire utilisé pour décrire les protocoles de communication.. 12

BS Binary Server: service de communication entre Cloud et boîtiers. 5, 9, 12

CC CloudConnect. II, IV, 8–10

CN CloudNext. II, 10

GDPR General Data Protection Regulation - Réglement Général sur la protection des données[3] . 9

MD21 Mobile Device 21 : le protocole de communication actuel de MDI. II, 5, 9, 11

MD30 Mobile Devices 30 : le nom du nouveau protocol de communication . 5, 18

MDI Mobile Devices Ingénierie. II, 2, 4, 5, 8–10, 13

OBD ON Board Diagnostics. 4, 5, 8