Rapport de Stage

(PremiŁre page : cf moodle)

Remerciments

//TODO

Table des matiŁres

A propos de OpenWide

Historique du groupe

Open Wide est une SSLL (SociØtØ de Service en Loficiel Libre) crØØe en 2001 par Patrick Benichou dont le siŁge social se situe Lyon. C'est une entreprise d'expertise dans le domaine des technologies Open Source et plus particuliŁrement autour de l'environement GNU/Linux. Pour cela, une de ses missions est de constituer un observatoire technologique et de sØlectionner puis de qualifier des solutions libres, suffisamment stables et normatives afin de les intØgrer industriellmeent au sein des systŁmes d'informtions de grandes entrprises et du secteur publique. Cepandant dans le r le d'architecte et de conseiller technologique en environnement hØtØrogŁnes, le champs des logiciels propriØtaires ne peut Œtre exclu totalement.

Fin 2002, l'entreprise s'engage vers la rØalisation de solutions clØen main avec engagement de rØsulats et deviens l'un des meilleurs architecte et untØgrateur Open Source en France .

DŁs 2003 Open Wide noue un partenariat avec Accelance MSP, une sociØtØspØcialisØe dans l'administration, la supervision et l'exploitation de plate-forme Open Source. Cela aboutira au rachat de Accelance MSP par Open Wide et la crØation du p le Open Wide Outsourcing.

En 2006, un nouveau p le d'activitØest crØer afin de ciblier les industriels, particuliŁrement dans le secteur de l'informatique embarquØe : Open Wide ingØnieurie (anciennement OS4i). C'est sous l'impulsion de Pierre Ficheux, spØcialiste reconnu de Linux embarquØe et directeur technique de la sociØtØet Øgalement sur la demande de grands groupes industriels tels que Thales, EADS, Airbus et bien d'autres, que OWI a vu le jour.

En 2007 Open Wide est en pleine croissance et crØe avec une sociØtØTunisienne un filiale commune appelØe Open Vision.

En 2010, la filiale Open Wide Technologies est crØe pour diffuser la plateforme Improve Foundation, dØdiØe au dØveloppement de projets Java.

En fØvrier 2015, le groupe rachŁte la soucitØ Neopixl, spØcialisØe dans le applications smartphone et tablette.

Fin 2015, Open Wide est rachetØpar le numØro 1 europØen de l'Open Source, Smile.

Illustration 1: DiffØrents clients du groupe Open Wide

Les diffØrents p les d'OpenWide

Open Wide SystŁmes d'information (OWSI) est l'intØgrateur dØdiØ aux applications mØiers, aux solutions de travail collaboratif et aux solutuions de GED.

Open Wide Outsourcing (OWT) est la sociØtØd'infogØrance du groupe. Elle propose des infrastructures dØdiØes ainsi que des services de Cloud privØs ou hybrides.

Open Wide Technologies (OWT) assure les missions d'expertise et de support sur le volet systŁme d'information. Elle est Øgalement Øditeur de la plate-forme Improve Foundations.

Open Wide IngØnierie est le spØcialiste leader en France des technologies Linux embarquØe et temps rØel. La sociØtØpropose Øgalement des compØence forte dans les domaines du traitement d'image, de la mobilitØet de l'IoT (Internet of things).

PrØsent sur Paris, Grenoble et Toulouse, l'entreprise est spØcialiste des plate-forme Linux embarquØe. Mon stage s'est dØroulØdans les locaux parisiens d'Open Wide ingØnierie

Organigramme

Dans le contexte du rachat de l'entreprise, il est compliquer de fournir un organigramme car celui ci n'est pas fixØ

Sujet du stage :

Le projet Sisell Box :

Open Wide propose des solutions d'analyse vidØo dans le cadre par exemple de la vidØosurveillance . L'enregistreur Sisell Box est un outil permettant de rØcupØrer les flux vidØos en provenance de divers camØra. L'utilisation d'algorithme de dØtection de mouvement ou de reconnaissance de forme (dØtection de visage par exemple) permet d'ajouter divers informations au flux vidØo sous forme de mØta-donnØes. Ce flux peut alors Œtre envoyer via un rØseau un enregistreur distant afin de conserver ses informations dans une base de donnØe.

Sujet avant arrivØet sujet rØel

Le sujet qui m'a ØtØconfiØavant mon arrivØe Øtait un projet de recherche et dØveloppement concernant la refonte de la plate-forme de vidØo-surveillance Sisell Box. Cet outils permet d'analyser un flux vidØo pour en extraire certaines information (visage, mouvement, etc.). Ce sujet dØfini correspondait bien au sujet rØel de mon stage, auquel est Øgalement venu ce greffer un sujet annexe de plate-forme de streaming bas dØbit embarquØe pour ThalŁs.

Etat de l'art :

Sisell box est une solution existante proposØpar Open Wide et dØj en place. Cependant, celle-ci Øtant vieillissante et devenant difficilement maintenable, il a ØtØ dØcidØ de refondre complŁtement la solution pour repartir se de nouvelles bases plus Øvolutives. Pour cela, la nouvelle version de Sisell Box se base sur le framework Gstreamer. Lors de mon arrivØe sue le projet, un streamer et un enregistreur basique Øtaient dØj rØalisØs ainsi que le portage de certaines briques de traitement d'image de l'ancienne solution vers la nouvelle architecture de Sisell.

Travail rØaliser :

L'objectif du stage est de revoir l'architecture de la Sisell Box et de l'outil Sisell Search (simillaire Sisell Box, mais fonctionnant sur la base d'enregistrements vidØo et non des sources live). Des travaux doivent Œtre menØ sur les ØlØments gØrant l'Ømission, le traitement et l'enregistrement des flux vidØo, ainsi que sur des outils permettant le contr le de ses diffØrentes opØrations.

De plus, l'ancienne version de la Sisell Box se prØsentait sous la forme d'un boitier contenant les outils d'enregistrements (carte embarquØe et disque dur) sur laquelle se connectait des camØras analogique. Cette solution se rØvŁle peu pratique et chŁre produire. C'est pourquoi nous souhaitons vers Øvoluer la Sisell Box pour utiliser des camØras numØriques et avoir une gestion plus souple de la plate-forme, qui pourrais Œtre soit embarquØavec la camØra (grâce des cartes de type Raspbery Pi ou autre) soit sur une machine distante de la camØra qui traite un flux reçu via un rØseau.

Planning prØvisonnel :

Les diffØrentes Øtapes du projet :

Gstreamer

La version original de la Sisell Box Øtait basØe sur ffmpeg, qui posait de nombreux problŁmes techniques et de maintenabilitØ Le framework GStreamer Øtant une solution relativement plus simple mettre en place et extrŒmement flexible, il a ØtØdØcider se baser dessus pour la refonte du projet.

La premŁre semaine de mon stage donc ØtØconsacrØ la dØcouverte et la familliarisation avec le framework Gstreamer sur lequel se base toute la structure de Sisell. C'est pourquoi il me semble important de dØtailler le fonctionement de ce framework.

Gstreamer est un framework multimedia sous licence libre publiØpour la premŁre fois le 31 octobre 1999. Il est Øcrit en C est initialement pensØ pour Œtre un solution concurrente QuickTime et DirectShow sur GNU/Linux. Aujourd'hui, gstreamer est disponible pour de nombreux systŁmes d'exploitations (GNU/Linux, OpenSolaris, BSD, Android, OSX, iOS, Windows, OS/400). De plus, des binding existent pour utiliser Gstreamer dans une multitude de langages de programmation en plus du C (C++, Java, Python, Vala etc..)

Principes technique

Gstreamer est basØsur un systŁme de pipeline. DiffØrents ØlØments sont connectØentre eux par des tuyaux (pipe). Chaque ØlØment possŁde des « pad » servant les connecter aux autres. Ces pads peuvent Œtre soit des entrØes (sink) soit des sorties (source).

Exemple de pipeline Gstreamer :

Cette pipeline lit un fichier *.ogg, en sØpare les pistes et dØcode sa piste audio pour ensuite la jouer.

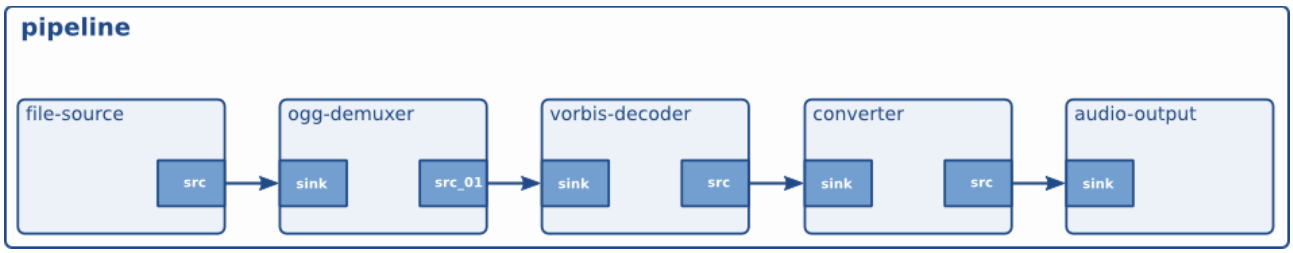

Illustration 2: Une pipeline gstreamer

SisellBox Streamer :

La partie streamer de Sisell Box gŁre la capture vidØo, l'analyse de l'image et son Ømission via un serveur RTSP.

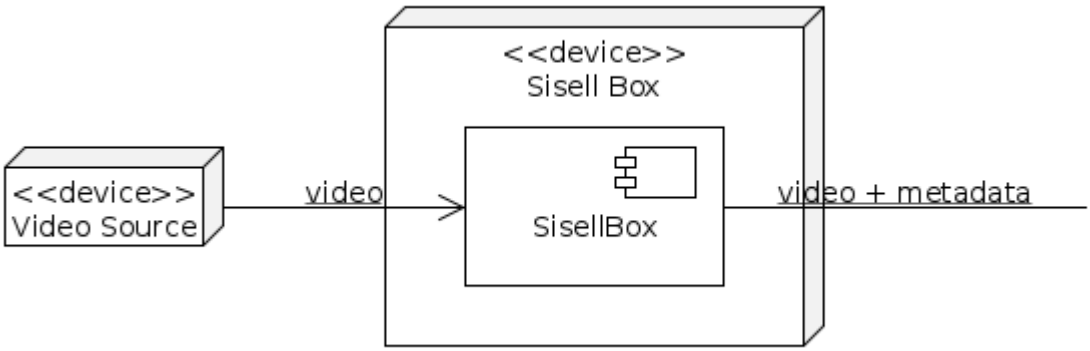

Illustration 3: Sisell box user case

La brique d'analyse est un ØlØment interchangable effectuant un traitement particulier sur l'image afin d'en extraire des informations (detection de visage, reconaissance de forme, mouvement, etc..). Actuellement, seul quelques algorithmes d'analyse (dØtection de visage) ont ØtØimplØmentØs pour la nouvelle architecture. Le flux vidØo peut Œtre encoder avec les formats H.264 et MJPEG, afin de les transmettre sur le rØseau via un des protocoles RTP, RTSP ou HTTP. Le streamer doit aussi gØrer des sources non-live (fichiers vidØo) de diffØrents formats (*.mkv, *.ts, *.avi).

Du c tØ de l'implØmentation, les birques algorithmiques de traitement d'image sont dØveloppØen C++ et en Vala. Le reste est dØveloppØen Python et en utilisant les bindings Python pour GStreamer afin d'utiliser le framework et ses plugins dØveloppØen C.

Sisell Box Recorder :

Le recorder rØcupŁre les flux vidØo et de mØta- donnØes gØnØrØpar le streamer. Les mØta-donnØes dont enregistrØ en base de donnØe ainsi que divers informations (localisation des

enregistrements vidØo etc.). La vidØo est multiplexØavec les mØta-donnØes (en utilisant la piste de sous-titre) puis enregistrØe sur le disque. Le recorder peut Øgalement rØcupØrer des images (format jpeg) d'une zone de la vidØo rØcupØrØ(par example un visage dØtectØen amont) via un ØlØment « Snapshooter ».

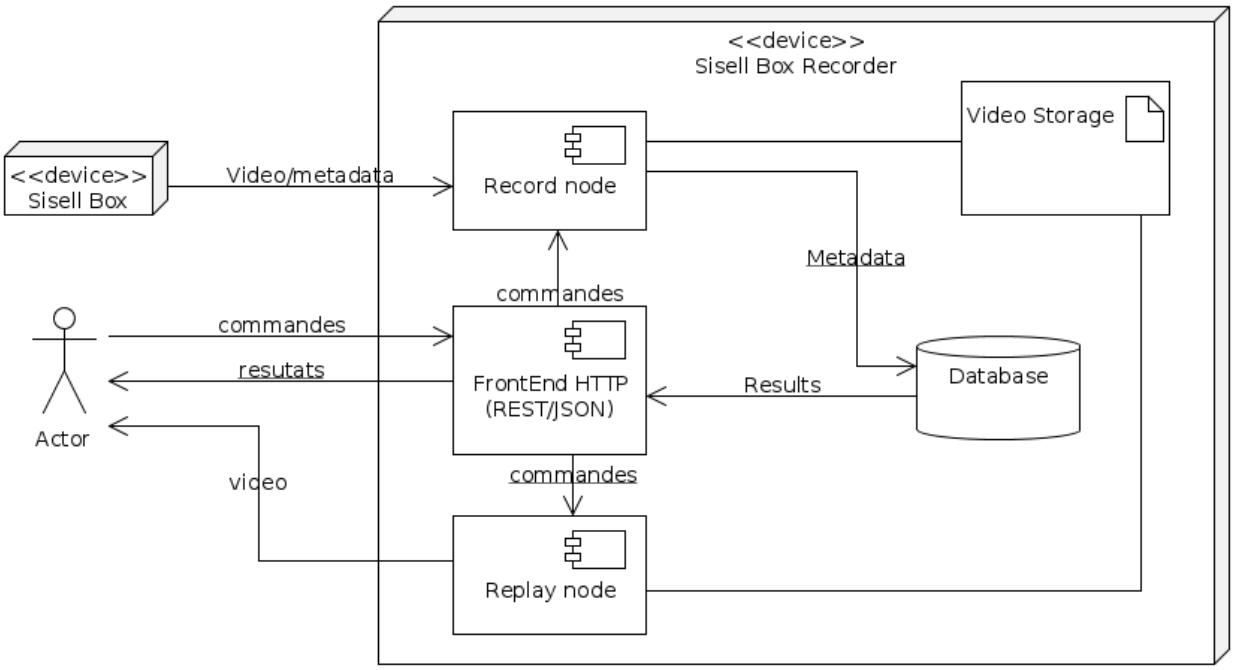

Illustration 4: Sisell box recorder user case

Pour la base de donnØe, nous utilisons PostgreSQL sur une machine Debian distante. Pour la phase de dØveloppement, cette machine est virtualisØvia un conteneur LXC (LinuX Containers).

Le problŁme Matroskamux

Lors des travaux sur le recorder, nous avons fait fasse une difficultØ imprØvu avec le multiplexage de la vidØo et des mØta-donnØes. Ce multiplexage ce fait l'aide du plugin Gstreamer « matroskamux » qui prends plusieurs pistes (video/audio/sous-titres) pour en faire un flux unique au format video/matroska (enrigistrØdans un fichier *.mkv). Cepandant, cela causait un bloquage complet de la pipeline. Cette phase de dØbugage nous amener effectuer de nombreuses vØrifications sur le fonctionnement de la pipeline (synchronisation des flux, timestamp des buffers etc.) alors que le problŁme venait de l'ØlØment matroskamux lui mŒme. L'ØlØment et la

documentation indiquaient tord le support d'un format (subitile/unknown) que nous utilisions pour les mØta-donnØes. Nous avons donc changØ pour passer du text/utf8 l'ØlØment. Ceci crØa un nouveau problŁme lors de la sØparation des flux, car l'ØlØment matroskademux renvoi en sortie la piste texte au format pango-markup. Hors le text ØchappØ ce format n'est plus utilisable en l'Øtat (le xml Øchappøn'est plus lisible en tant que tel). Des modifications du code et la recompilation de l'ØlØment matroskademux ont donc ØtØnØcessaire.

Modification des architectures du streamer et du recorder

Les parties streamer et recorder Øtait l'origine deux entitØs totallement sØparØ Cepandanant, leurs architectures et leur fonctionnement ne sont pas si diffØrents. Les deux prennent un flux vidØo, effectuent des traitement et le tranfert sur une sortie (Ømission rØseau ou stockage). Ces deux programmes Øtant tout les deux composØ d'une pipeline GStreamer, tout les deux comportait des ØlØments commun (gestion du status de l'Øtat du flux, gØnØration de graph reprØsentant la pipeline etc.). Les rØpØitions de code Øait donc relativement nombreuse.

Une des premiŁres partie de mon stage donc consister travailler sur l'architecture du streamer et du recorder afin de rØpondre plusieur problŁmatiques :

- Factoriser l'architecture et le code (python2) des deux programmes en une base commune pouvant ensuite se spØcialiser.
- Rendre le code le plus modulaire possible afin de pouvoir gØnØrer et modifier simplement les pipelines
- Permettre l'ajout et la suppression d'ØlØments chaud (pour pouvoir par exemple ajouter/supprimer/remplacer des briques algorithmiques de traitements d'image sans relancer l'application).
- CrØer des ØlØments permettant de gØrer les diffØrentes sources, encodage, protocoles d'Ømission souhaitØ
- Faire un minumum de transcodage, du flux vidØo afin d'optimiser le temps de traitement (le transcodage Øtant une opØration coßteuse).

Sboxcommon :

La premiŁre Øtape de ce ramnimant ØtØ de crØer un module python « sboxcommon » qui contiendra tout les sous-modules communs au streamer et au recorder. Les modules python prØsents en double dans les deux programmes y sont transfØrØ et tout les modules crØØ par la suite pour qui ne sont pas spØcifiques l'une des deux t che y sont Ølement placØ

Nous crØons un programme python nommØ « sisellbox.py » qui sera chargØ d'initialiser et de monitorer des pipelines Gstreamers (qui seront soit « streamer » soit « recorder »).

pipelineManager

Le sous module pipelineManager contenant une classe unique du mŒme nom contient les fonctions de surveillance et de contr le de la pipeline. Le pipeline manager Øcoute les messages Ømits et gŁre les potentiels erreurs, la fin et/ou l'arrŒt du flux de donnØe dans la pipeline. Cette classe permet Øgalement de gØnØrer des fichiers au format *.dot permettant de rØaliser une reprØsentation graphique de l'Øtat de la pipeline lors de ses changements d'Øtats.

PipelineGenerator

Ce sous module contient la classe generatedBin, hØritant des bins GStreamer. Un bin est un conteneur gstreamer permettant de considŁrer un ensemble d'ØlØment comme un ØlØment unique. Dans notre cas, nous crØeons un bin contenant l'intØgralitØ de la pipeline. Ainsi nous n'avons qu'un ØlØment ajouter dans la pipleline dans tout les cas de figure dans l'application sisellbox.py, les problŁmatiques d'ajout/suppression, connexion d'ØlØments etc. sont dØlØguØs la classe generatedBin. Les diffØrents ØlØments du bin sont crØer par une classe generatedBinFactory. La classe generatedBin tant qu'a elle gŁre l'ajout des ces ØlØments dans le bin, leur connexion et leur suppression ainsi que la modification des diffØrentes propriØtØs des ØlØments Gstreamers contenue dans le bin.

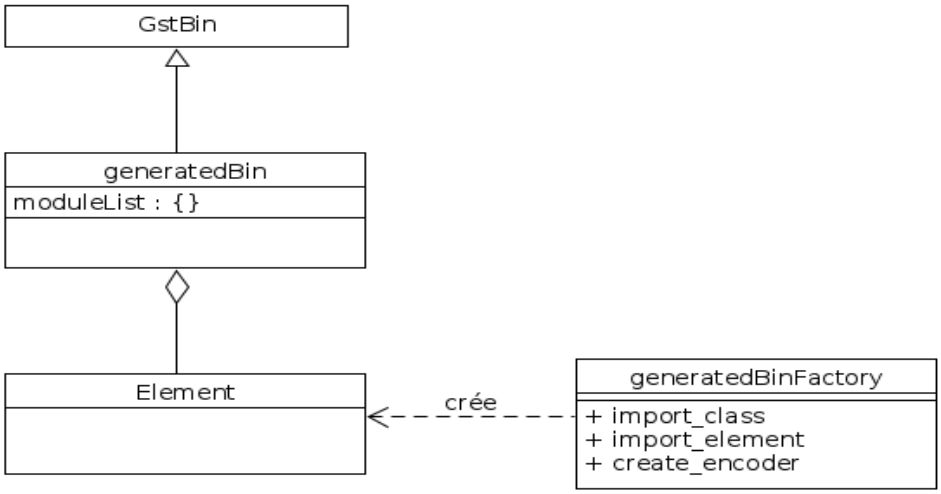

Illustration 5: generatedBin: Diagramme de classe simplifiØ

L'ØlØment source du bin emet un flux avec un certain format (raw, h.264, jpeg, etc..) qui n'est pas nØcessarirement celui requis par les autres ØlØments. Il a donc falu concevoir la mØthodologie de connexion des ØlØments afin que les flux soit transcoder (uniquement) lorsque cela est nØcessaire. Ainsi, lors de l'ajout d'une source un ØlØment « tee » est ajoutØderrirŁre celle ci et est

conservØdans une liste de « tee ». Cet ØlØment permet de multiplier un flux pour le distribuer plusieurs ØlØments. Lors de l'ajout d'un ØlØment (non source), le generatedBin regarde dans la liste de tee si un d'entre eux fourni un format que peut utiliser l'ØlØment. Si oui, celui-ci est branchØau tee correspondant. Dans le cas contraire un ØlØment rØalisant le transcodage vers le format souhaitØ est ajoutØet un nouveau tee lui est greffØ Ainsi un nouvel ØlØment pourra Øgalement bØnØficier du flux transcodØsans rØaliser une nouvelle fois l'opØration (cf pipelines en annexe).

Serveur de relecture :

Afin de pouvoir retrouver une sØquence vidØo et la relire, nous ajoutons aux ØlØments « streamer » et « recorder » un ØlØment « replay » afin de rØemettre le flux d'une vidØo enregistrØe. Dans son architechture, le serveur de relcture est un player. Il ouvre des fichiers vidØo, peut les lire (la lecture consisite ici Ømettre un flux sur le rØseau et non afficher localement la vidØo) mettre le flux en pause, l'arrŒtØetc. Cet ØlØment doit Œtre contr ler via une interface web (cf partie frontend web). Le serveur de relecture est constituØde deux ØlØments principaux : La classe ReplayFileSrc contient les diffØrents ØlØments GStreamer constituant la pipeline et la classe Replay qui monitore les fonctions du service tel que la gestion du flux (play/pause) ou l'ouverture d'un fichier.

Frontend web :

Afin de pouvoir intØragir avec les diffØrents ØlØment de la solution Sisell, nous mettons en place des APIs de contr le au travers d'une interface web-service HTTP. Ces APIs suivent un modŁle REST savoir :

- Les ressources sont accŁssible par leur URL, chaque URL reprØsentant une ressource en particulier.
- Les requŒtes et les rØponses seront encodØe au format « application/json », les dates sont au format iso8601 ou en timestamp, les gØomØtries sont au format GeoJSON
- Des actions de type Create Read Update Delete (CRUD), reprØsentØes par les « verbes » HTTP POST/GET/PUT/DELETE, permettent de manupuler les ressources.
- Les codes d'erreur HTTP permettent d'indiquer la bonne exØcution des requŒtes, la sØmantique de ces codes sera conservØe.
- Les API sont versionnØes (api/v1/path/to/ressource), un changement d'API reprØsentant une incompatibilitØavec la version prØcØdente. Les changements d'API sont limiter au maximum.

Par exemple, pour rØcupØrer la liste de tous les enregistrement nous devons utiliser la requŒte HTTP GET comme suit :

```
curl -X GET -H 'Content-Type:application/json'
http:://127.0.0.1:5000/api/v1/records
```

Pour gØrer la communication en interne entre les composants et les applications sisell, nous utilisons un mØcanisme de RPC (Remote Procedure Call). Pour cela, nous utilisons l'API gRPC de Google. Bien que rØcente (sortie en 2015 et toujours en phase de bØ) cette API l'avantage d'Œtre lØgŁre et simple d'utilisation. De plus elle est basØ sur protobuf, dØj utiliser dans le projet pour sØrialiser les mØta-donnØe. Cela nous permet donc de garder une cohØrence dans les technologies utilisØes. Pour cela, nous devons crØer un fichier protobuf contenant les requŒtes rpc que nous souhaitons implØmenter et les messages protobuf associØ(un message contient 0 n variable sØrialiser). Les requŒtes rpc de gRPC ne peuvent prendre en paramŁtre et ne retourner que des messages protobuf.

Travail sur les mØtadonnØes :

Afin de transportØ les mØtadonnØes comme n'importe quel flux avec Gstreamer, il nous a fallu les sØrialiser. Pour cela, le protocole de sØrialisation de Google, le protocol buffer (ou protobuf) ØØchoisi. La structure de donnØe protobuf pour les mØtadonnØes Øait la suivante :

```
message Frame {
        repeated Rect rect = 1;
}

enum ClassType {
        UNKNOWN_CLASS = 1;
        FACE_CLASS    = 2;
        MOTION_CLASS  = 3;
}

message Rect {
        required uint32 id = 5;
        required uint32 left   = 1;
        required uint32 width  = 2;
        required uint32 top    = 3;
        required uint32 height = 4;

        optional ClassType class_type = 6 [default = UNKNOWN_CLASS];
}
```

Ces mØta-donnØes sont sØrialisØe en utilisant le protocole de Google : protobuf (Protocol Buffer). En fin de traitement, ces mØta-donnØes sont enregistrØ dans une base de donnØe sous le format XML et correspondant la norme ONVIF (Open Network Video Interface Forum). //TODO : dØtailler ONVIF

Au cours du stage, cette structure ØØremaniØafin de rØpondre plusieurs problŁmatique. Tout

d'abord, une notion de timestamp ØÔajoutØafin de pouvoir simplement retrouver le moment de la vidØo source correspondant la donnØe. Pour cela, nous avons ajouter au pipeline un ØÔment GStreamer Øcrit en vala qui copie le timestamp du buffer contenant la mØtadonnØe dans un champs

```
required uint64 timestamp    = 2;
```

en plus du rectangle dans la frame.

Ce plugin est placØjuste aprŁs la brique d'analyse afin que le timestamp du buffer de mØtadonnØe corresponde celui de la bonne frame du flux vidØo.

Un autre travail important sur les mØtadonnØe ØÔ de modifier le format des trames protobuf et des fichiers XML obtenus afin de mieux correspondre la norme ONVIF. Ainsi, les trames ne doivent plus contenir d'objet de type « Rectangle » mais une structure « GØometrie » plus gØnØrique. La nouvelle structure souhaitØpour les trames est donc :

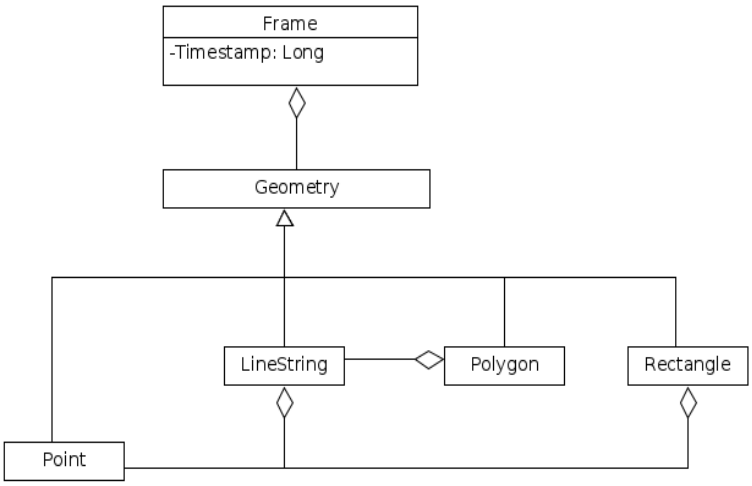

Illustration 6: Frame: diagramme de classe simplifiØ

La norme ONVIF ne contient normalement pas de gØomØtrie de type rØctangle. Ceux ci-doivent Œtre reprØsentØpar un polygone. Cependant, un polygon Øtant formØe d'une « LineString » fØrmØe (suite de points formant une ligne brisØe). Cette mØthode nous forcant stocker cinq points pour un rectangle, la ou deux pourrait suffire (les coins opposØes du rectangle). Cette forme Øtant largement la plus utilisØpar les outils d'analyse de Sisell Box, cette lØgŁre entorce la norme ONVIF s'avŁre justifiØau vu du gain apportØpour le transport et le traitement de ces rectangles.

La nouvelle structure de nos mØta-donnØes deviens donc :

```
message Frame {
        repeated Geom geom = 1;
        optional uint64 timestamp_ms = 2;
```

```
}

enum ClassType {
        UNKNOWN_CLASS = 1;
        FACE_CLASS    = 2;
        MOTION_CLASS  = 3;
}

 message Geom {
        optional uint32 id = 1;
        optional Point point = 2;
        optional LineString linestring = 3;
        optional Rectangle rectangle = 4;
        optional Polygon polygon = 5;
        optional MultiPoint multipoint = 6;
        optional MultiLineString multilinestring = 7;
        optional MultiPolygon multipolygon = 8;
        optional GeometryCollection geometrycollection = 9;
        optional uint32 wscale = 10;
        optional uint32 hscale = 11;
        optional ClassType class_type = 12 [default = UNKNOWN_CLASS];
}

message Point {
        required uint32 coordx = 1;
        required uint32 coordy = 2;
}
 message LineString {
        repeated Point coordinates = 1;
}

message Rectangle {
        repeated Point coordinates = 1;
}
 message Polygon {
       repeated LineString coordinates = 1;
}

message MultiPoint {
        repeated Point coordinates = 1;
}

message MultiLineString {
        repeated LineString coordinates = 1;
}

message MultiPolygon {
        repeated Polygon coordinates = 1;
}

message GeometryCollection {
        repeated Geom geom = 1;
}
```

Le code des briques de traitement d'image Øcrit en C++ et vala ont Øgalement du Œre remaniØafin de fournir des mØta-donnØes correspondant ce nouveau format.

Autres sujets :

Etude de faisabilitØpour Nexter

RØalisation d'une application de streaming sur plate-forme imx6 pour ThalŁs

L'objectif de ce projet est de rØaliser une plate-forme de streaming bas dØbit pour une carte wandboard imx6. Ce projet se dØcompose en trois lots :

- Lot 1 : RØalistation d'une application de streaming bas dØbit embarquØe
- Lot 2 : RØalisation d'un lecteur pour rØcupØrer le flux vidØo sur PC
- Lot 3 : RØalisation d'un interfacage avec le protocole SNMP afin de pouvoir paramŁtrer l'application sur la carte depuis une machine distante.

Lot 1 : Application de streaming

L'application est basØe sur GStreamer et doit rØcupØrer un flux vidØo en provenance d'une camØra et le transfØrer via RTP aprŁs avoir appliquer certains traitements au flux au travers d'un plugin de « pre-traitement ». Ce plugin de pre-traitement ne fait qu'un redimensionnement de la vidØo, mais sert principalement de coquille vide que le client pourra utiliser pour implØmenter ses propre traitements. De plus, l'application Øtant destinØ un systŁme embarquØ il nous faut Œtre prudent sur les librairies externes en essayant d'en utiliser le moins possible.

L'application est Øcrite en C et doit permettre d'utiliser plusieurs encodeurs :

- openjpegenc : Encodeur Jpeg2000
- openh264enc et x264enc : deux encodeurs h264
- imxvpuenc_h264 : afin de rØaliser un encodage hardware en utilisant le vpu intØgrØ la carte.

Tout les paramŁtres d'encodage ainsi plusieurs autres (adresse de destination, port, etc.) doivent pouvoir Œtre modifiØs de deux fa on diffØrentes : par arguments en ligne de commande et par un fichier de configuration. Pour le fichier de configuration, nous avons choisi d'utiliser libconfig qui permet de simplement rØcupØrer et manipuler les informations contenu dans le fichier sans rØØcrire un parser en intØgralitØ

Pour gØnØrer la toolchain de cross compilation, nous utilisons buildroot qui permet de simplement sØlectionner les packages que nous souhaitons installer sur notre systŁme. Un package pour notre application doit donc Øgalement Œtre crØer. Pour cela, nous crØeons un fichier *.mk qui contient les rŁgles de compilation pour notre application. Ensuite nous pouvons gØnØrer l'image de notre systŁme et l'installer sur la carte.

Lot 2 : rØalisation d'un lecteur vidØo

Le lecteur vidØo rØalisØ avec Qt pour gØrer l'interface graphique et GStreamer pour la partie gestion du flux vidØo. Certains paramŁtres peuvent Œtre mis modifiØ depuis cette interface, mais le plus gros du paramØrage sera fait dans le lot 3. Mon r le dans cette Øtape Øtait de rØaliser la partie traitement du flux avec Gstreamer. Tout comme pour la partie streaming, cette application est livrØe avec un plugin de « post-traitement » faisant du redimentionement et destinØ Œtre modifier par le client. Cette pipeline GStreamer Øtait dØvelopper sous la forme d'une librairie partagØe afin de l'importer dans notre interface graphique Qt.

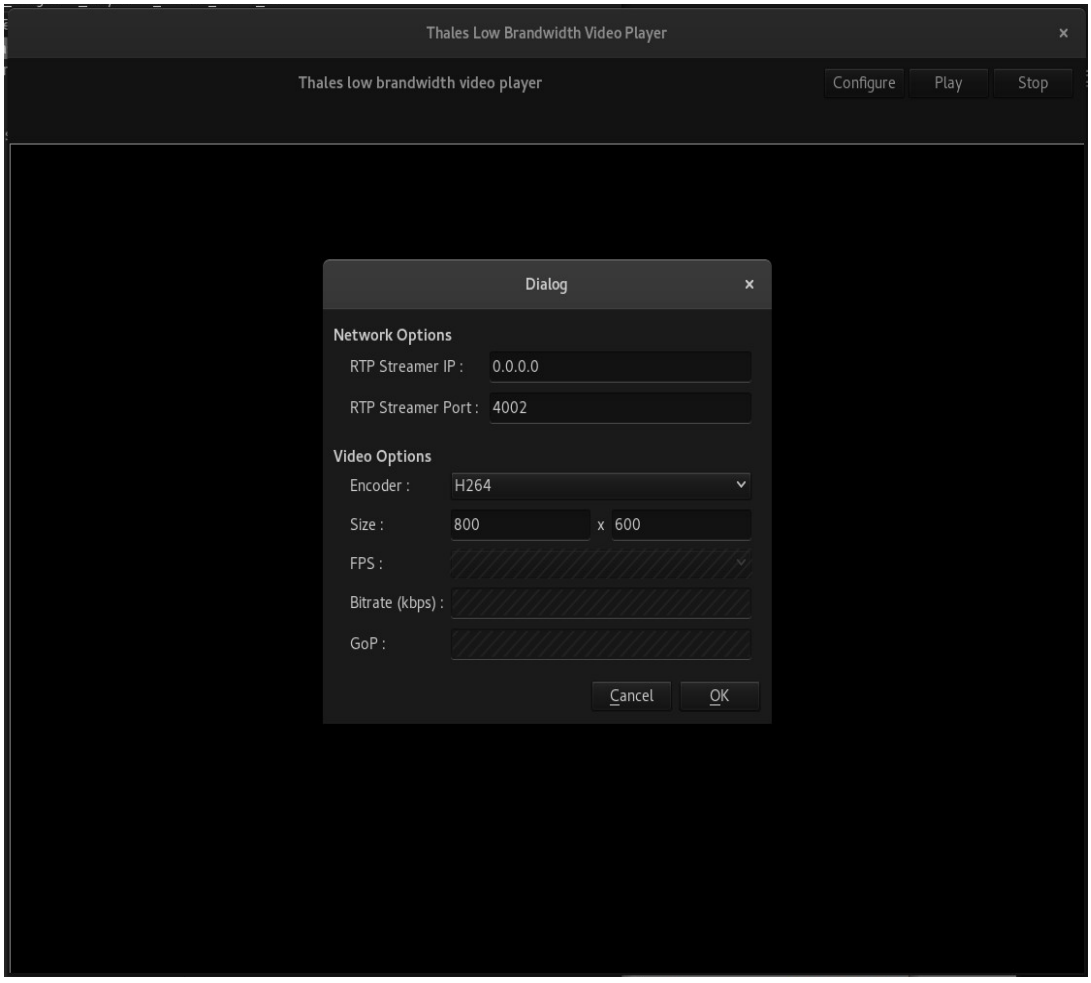

Illustration 7: interface graphique du lecteur vidØo

Conclusion

tant moi mÖme partisant de la philosophie liØau logiciel libre, travailler dans ce milieu ØØ… . Le fait de travailler sur de nombreuses technologies (gstreamer, protobuf, grpc, vala, ansible, etc.) que je dØcouvrait pour une large partie d'entre elle ØØpour moi trŁs enrichissant et m'a permis de progresser Ønorm Ø ment.