# Impact des ROI dans le streaming vidéo

Encadrant : **Olivier FOURMAUX**

Etudiants : **Sonia LOUNIS, Fabien MANSON et Alexandre MAZARS**

## Table des matières

# 1 Cahier des charges

## 1.1 Présentation du projet

Ce projet est réalisé dans le cadre de l'UE PRES de l'année 2019, Il est proposé par Mr. Olivier FOURMAUX et porte sur l'étude de l'impact du streaming vidéo adaptatif et la mise en place d'une démonstration de cette technologie.

## 1.2 Objectif du projet

Dans un premier temps nous allons étudier le fonctionnement de l'encodage différencié des régions d'intérêts ainsi que la transmission vidéo via http. Dans un second temps nous mettrons en place un système de codage et décodage adaptatif spatial. Enfin nous devrons présenter une démonstration de l'encodage différencié au travers d'un casque de réalité virtuelle ou via une émulation sur un écran et de son impact sur le réseau.

## 1.3 Contexte du projet

Ce projet s'inscrit dans la dynamique actuelle de la diffusion massive de contenu vidéo haute résolution qui ne cesse d'augmenter. Le développement de la réalité virtuelle et du contenu vidéo en VR représente une partie du futur de la diffusion de contenu vidéo. On peut trouver quelques études qui ont déjà travaillé sur ce sujet en proposant des méthodes d'optimisation de la transmission de vidéo UHD et du découpage de celle-ci en régions d'intérêt.

## 1.4 Contraintes liées au projet

Contrainte de temps le projet est à rendre en mai 2020 en prévision d'une soutenance de ce dernier un peu plus tard le mois. Les dates ne sont pas encore précisées. Les dates ont été changées à cause de l'épidémie du Covid-19, le rapport est à rendre le 10 juin 2020 et la soutenance aura lieu le 15 ou le 16 juin 2020 probablement en visio-conférence.

## 1.5 Documentation du projet

La documentation de la partie technique du projet sera minimaliste. Elle pourra se trouver sous l'une des deux formes suivantes : directement dans le code du projet sous la forme de commentaire de code très détaillé ou dans un fichier texte annexe (on peut même envisager un fichier PDF si le projet ne prend pas trop de retard).

## 1.6 Maintenant et évolution du projet

Le projet n'aura plus aucun suivi passé la date de soutenance. Il ne sera plus mis à jours ni même modifié. Cependant ou pourra retrouver l'intégralité du code source sur une page GitHub dédiée.

# 2 Plan de développement

La première partie du plan de développement portait sur l'analyse des différentes recherches menées par plusieurs équipes. Cette partie à aboutie à la création d'une bibliographie initiale de 13 articles que nous avons porté au nombre de 16 en ajoutant Des sources internet qui nous semblaient pertinentes. On retrouve entre autre la page officiel du logiciel GPAC ainsi que la page GitHub regroupant le code source du logiciel. La seconde partie s'oriente vers la mise en place de l'architecture de streaming DASH et la création d'un flux vidéo de démonstration. Il est évident que le plan de développement a changé par rapport à celui présenté dans le rapport intermédiaire. À cause de l'épidémie de Covid-19 les dates de rendu et de soutenance ont été modifiées pour nous laisser un peu plus de temps pour avancer le projet en confinement. C'est pourquoi le diagramme va jusqu'en Juin alors que la fin théorique était en Mai. Concernant le second semestre (figure 2) nous avons tous installé les logiciels requis pour le projet (encodeur, lecteur, serveur web, etc…) sur nos machines personnelles pour pouvoir réaliser de tests chacun de notre côté. Au final seul 2 de nos 3 installations se sont révélée suffisamment stable pour faire des tests dans de bonnes conditions. Une petite précision doit être ajoutée au diagramme du second semestre, la phase d'analyse de la documentation de GPAC et de Kvazaar s'est faite quasiment en parallèle

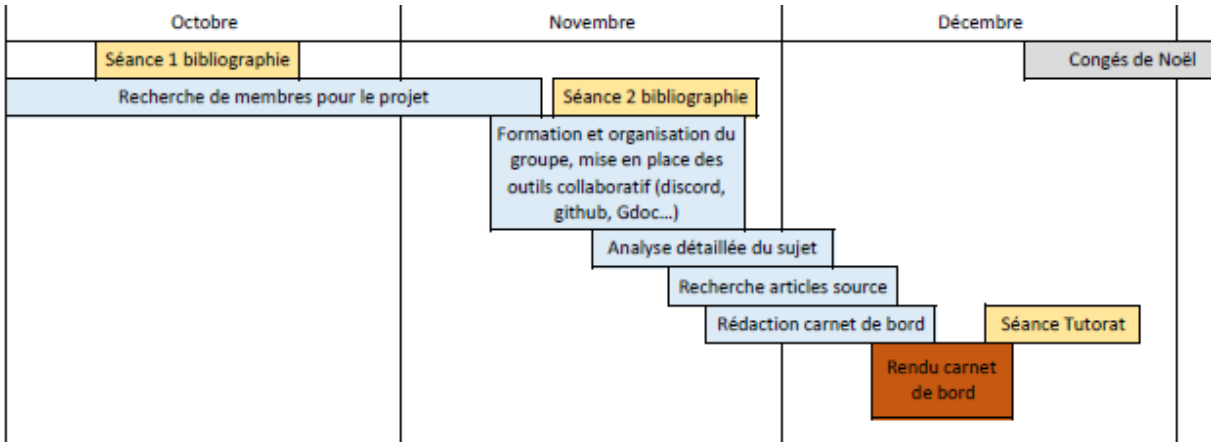


**Figure 1 Digramme de Gantt partie 1 (détail du premier semestre)**

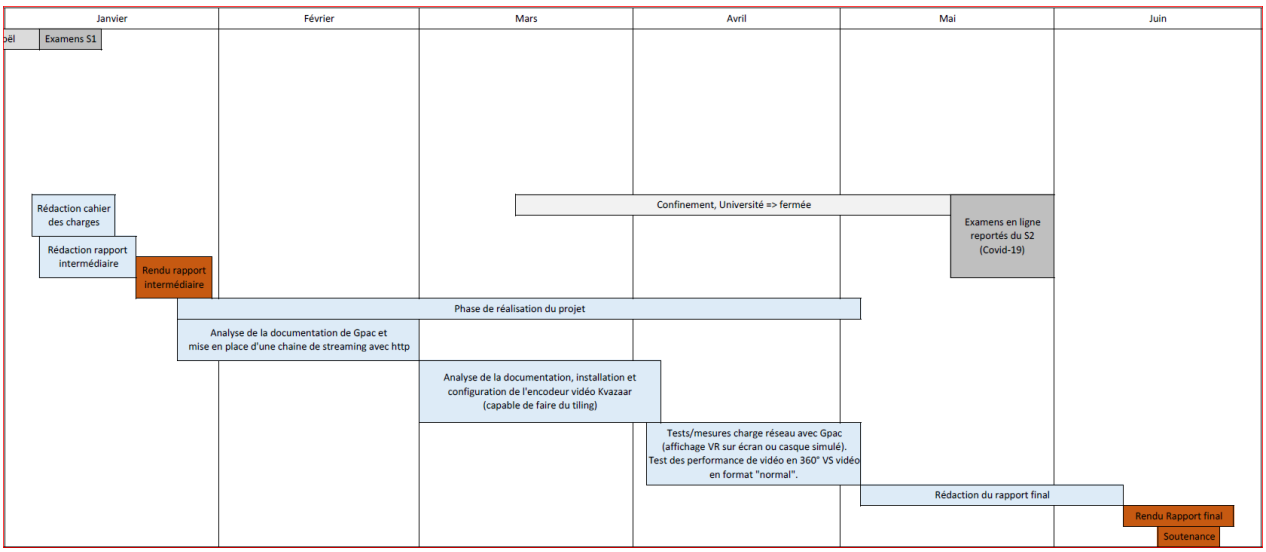


**Figure 2 Diagramme Gantt, fin du premier semestre (fin Janvier) et second semestre (Février à Juin)**

Les Items en Orange foncé sur le diagramme sont les différents rendus : carnet de bord, rapport intermédiaire et final.

# 3 Bibliographie

La bibliographie reste quasiment la même. Nous avons simplement rajouté quelques sites web tels que le site de GPAC ainsi que la page Wikipédia officielle de GPAC.

1] Tarek El-Ganainy and Mohamed Hefeeda. 2016. Streaming Virtual Reality Content. *arXiv:1612.08350 [cs]* (December 2016). Retrieved December 5, 2019

[2] Dan Grois and Ofer Hadar. 2011. Recent Advances in Region-of-interest Video Coding. In *Recent Advances on Video Coding*. DOI:https://doi.org/10.5772/17789

[3] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer! (September 2016). Retrieved December 5, 2019 from https://arxiv-org.accesdistant.sorbonne-universite.fr/abs/1609.08729v5

[4] JongBeom Jeong, Dongmin Jang, Jangwoo Son, and Eun-Seok Ryu. 2018. 3DoF+ 360 Video Location-Based Asymmetric Down-Sampling for View Synthesis to Immersive VR Video Streaming. *Sensors* 18, 9 (September 2018), 3148. DOI:https://doi.org/10.3390/s18093148

[5] Christopher Müller and Christian Timmerer. 2011. A VLC Media Player Plugin Enabling Dynamic Adaptive Streaming over HTTP. In *Proceedings of the 19th ACM International Conference on Multimedia* (MM '11), 723–726. DOI:https://doi.org/10.1145/2072298.2072429

[6] D. V. Nguyen, Huyen T. T. Tran, and Truong Cong Thang. 2019. A client-based adaptation framework for 360-degree video streaming. *Journal of Visual Communication and Image Representation* 59, (February 2019), 231–243. DOI:https://doi.org/10.1016/j.jvcir.2019.01.012

[7] Duc V. Nguyen, Huyen T. T. Tran, and Truong Cong Thang. 2019. Adaptive Tiling Selection for Viewport Adaptive Streaming of 360-degree Video. *IEICE Transactions on Information and Systems* E102.D, 1 (2019), 48–51. DOI:https://doi.org/10.1587/transinf.2018MUL0001

[8] Cagri Ozcinar, Ana De Abreu, and Aljosa Smolic. 2017. Viewport-aware adaptive 360° video streaming using tiles for virtual reality. *arXiv:1711.02386 [cs]* (November 2017). Retrieved December 5, 2019 from http://arxiv.org/abs/1711.02386

[9] Georgios Papaioannou and Iordanis Koutsopoulos. 2019. Tile-based Caching Optimization for 360° Videos. *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing* 171--180 (July 2019). Retrieved December 5, 2019 from http://graphics.cs.aueb.gr/graphics/docs/papers/MOBIHOC-2019.pdf

[10] Y. Sánchez de la Fuente, R. Skupin, and T. Schierl. 2017. Video processing for panoramic streaming using HEVC and its scalable extensions. *Multimed Tools Appl* 76, 4 (February 2017), 5631–5659. DOI:https://doi.org/10.1007/s11042-016-4097-4

[11] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems* (MMSys '11), 133–144. DOI:https://doi.org/10.1145/1943552.1943572

[12] Alireza Zare, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. In *Proceedings of the 24th ACM International Conference on Multimedia* (MM '16), 601–605. DOI:https://doi.org/10.1145/2964284.2967292

[13] Alireza Zare, Maryam Homayouni, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2019. 6K and 8K Effective Resolution with 4K HEVC Decoding Capability for 360 Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 15, 2s (July 2019), 68:1–68:22. DOI:https://doi.org/10.1145/3335053

[14] GPAC Multimedia Open Source Project, Telecom Paris, *https://gpac.wp.imt.fr*, (dernière mise à jour en Octobre 2019), site consulté entre Mars et Juin 2020

[15] GPAC Project on Advanced Content, Wikipedia The Free Encyclopedia, *https://en.wikipedia.org/wiki/GPAC_Project_on_Advanced_Content*, page créée en 2008, dernière mise à jour de la page en mars 2020, site consulté entre Mars et Juin 2020

[16] GPAC Code source, Github.com, *https://github.com/gpac/gpac*, Dernière mise à jour du code source le 3 Juin 2020, site consulté régulièrement entre Mars et Juin 2020

# 4 Analyse

Le concept de région d'intérêt a été mis en place dans le but de réduire la charge réseau tout en conservant un maximum de qualité vidéo. Dans ce projet on s'intéresse particulièrement à son impact sur les vidéos en 360 degré car c'est là où elle a le plus d'impact. En effet, lorsqu'on visionne une vidéo en 360 degré, notre champ de vision est réduit à une petite partie de la vidéo car cela évite de voir l'effet de déformation lié à la projection cylindrique équidistante (type de projection le plus courant pour passer d'une sphère a un plan). Sans utiliser les régions d'intérêt, toute la partie de la vidéo qui se trouve en dehors du champ de vision est quand même téléchargée depuis le serveur vers le client ce qui pose en problème de bande passante. La région d'intérêt permet de définir une région particulière (généralement la zone que regarde l'utilisateur) qui sera prioritaire par rapport aux autres zones. Un débit élevé et donc une meilleur qualité d'image sera attribuée aux régions ayant une forte priorité. Le reste des régions moins prioritaires auront un débit attribué plus faible permettant de réduire grandement la charge de la bande passante. Le choix de la région d'intérêt est libre et arbitraire, il peut s'agir de la région ou regarde l'utilisateur (capteurs d'un casque VR, pointeur de souris), d'une zone fixe ou encore d'une zone importante dans la vidéo par exemple une zone avec de nombreux détails nécessitant une meilleure qualité d'image. Ce principe de région d'intérêt est très important car il ne faut pas oublier qu'une vidéo en 360° est au minimum en 4k 30ips (ultra haute définition). La plupart du temps les vidéos ont une résolution bien supérieur à de la simple 4K, on trouve notamment de la 6K et même dans certains cas de la 8K (images de respectivement 6000 et 8000 pixels de large). Voici un exemple pour mieux comprendre l'intérêt du streaming dynamique et des régions d'intérêt. Nous disposons d'une vidéo en 4k60ips (image de 4096x2160 pixels) avec un débit binaire de 35Mbits/s. Nous créons un flux mpd avec différents bitrates, par exemple 15Mbits/s, 5 Mbits/s et 1Mbits/s. Dans le cas où le client dispose d'une bande passante inférieure à 35 Mbits/s et qu'il souhaite visionner la vidéo sans utiliser le streaming dynamique, son expérience de visionnage ne sera pas bonne. La vidéo sera saccadée car le débit de lecture est supérieur au débit de chargement de la vidéo. Une solution consiste à laisser charger la vidéo intégralement en mémoire pour la lire ensuite mais ce n'est pas envisageable pour des films qui, en très haute résolution, pèsent plusieurs dizaines de giga octets. La seconde solution intéressante est d'avoir recours au streaming dynamique. Certes le client ne dispose pas d'une bande passante suffisante pour visionner la vidéo dans sa résolution native, mais grâce aux différents bitrates encodés et disponibles via le flux décrit par le fichier .mpd, il peut visionner la vidéo de manière fluide. C'est le lecteur qui se charge de sélectionner le bitrate correspondant à la bande passante disponible. Les conséquences de

l'utilisation d'un tel système sont une qualité d'image dégradée et une charge d'encodage en amont élevée. Pas de secret, pour obtenir plusieurs versions d'une vidéo avec des bitrates différents il faut encoder la vidéo de multiples fois ce qui requiert une puissance et un temps de calcul élevée.

Le concept de tuile est tout aussi important que le concept de ROI pour pouvoir mettre en place un stream dynamique. Une tuile est une zone rectangulaire « découpée » dans une image et regroupant un certain nombre de pixels. Le nombre et la forme des tuiles découpées dans une image peuvent varier, on peut avoir des tuiles de taille identiques ou un ensemble dépareillé.

La méthodologie pour mettre en place un système de stream de vidéo utilisant le principe de région d'intérêt est toujours la même. Dans un premier temps, il faut trouver un média source. De préférence libre de droit est ayant une résolution d'image au moins égale à de la 4K avec une fréquence d'image de 30ips (images par secondes) au minimum. On peut Notamment utiliser le site Mettle.com qui propose quelques vidéos d'une trentaine de secondes en VR filmées depuis un drone ou une caméra statique. Une fois que l'on a téléchargé le média source, il faut appliquer un découpage en tuiles selon une grille de taille choisie, généralement une grille carrée. Ce découpage en tuiles peut se faire avec des encodeurs comme ffmpeg ou kvazaar. Le choix du découpage est totalement arbitraire, On peut choisir de découper des tuiles de même taille dans la vidéo on de découper de zones de taille et de formes variées. On peut par exemple ne découper qu'une zone dans la partie de la vidéo correspondant au bas/haut de la vidéo VR qui sont statistiquement les zones les moins regardées dans la sphère complète. Il faut aussi produire des clones du média source avec une qualité d'image volontairement dégradée. On prend par exemple la vidéo source en pleine résolution, et on crée d'autres clones, ayant chacun des résolutions de plus en plus faibles. Par exemple avec un média source ayant 4096p (pixels) de large, on peut créer des clones ayant les résolutions suivantes 2560p, 1920p, 720p, 360p. Une fois le découpage effectué, on obtient pour chaque tuile plusieurs segments de différentes résolutions. Il faut ensuite créer un fichier descripteur de flux mpeg-dash (.mpd) qui permettra de localiser les différentes tuiles et les différentes résolutions disponibles pour chaque tuile. Ensuite il ne reste plus qu'à trouver un lecteur qui permette d'ouvrir un flux mpeg-dash ce qui n'est pas chose aisée car la plupart des lecteurs sont des technologies d'entreprises privées comme YouTube ou Facebook. Il existe cependant un projet open source de lecteur dash en JavaScript et GPAC qui est un lecteur multimédia capable de lire ce type de flux. Le rôle du lecteur n'est pas si simple que cela car il doit non seulement afficher la vidéo de manière fluide (ce qui n'est pas trivial avec des très hautes résolutions) mais aussi télécharger les bonnes tuiles correspondant à la bande passante disponible. Le lecteur peut soit télécharger les tuiles depuis un serveur web soit directement en local si les fichiers son présent sur la machine du client. Le lecteur GPAC est donc une très bonne option car il est bien documenté et très complet c'est pourquoi nous l'utilisons pour la partie décodage et affichage du flux vidéo.

# 5 Conception

La méthodologie que nous avons suivie est quasiment identique à celle disponible sur la page GitHub[1] de GPAC, elle y est décrite assez succinctement mais avec suffisamment d'exemples pour que cela soit compréhensible mais surtout reproductible avec notre environnement de test.

Nous avons donc dans un premier temps récupéré un média source sur le site Mettle.com[2]. Nous avons ensuite modifié le média source en le copiant en miroir pour doubler sa durée, car 30 secondes ce n'est pas assez pour observer un changement de débit majeur (car la taille minimale recommandée du buffer d'images est de 1 seconde et que le lecteur met un certain temps à passer d'un flux à un autre). Le média résultant est une vidéo au format MP4 d'environ une minute de durée. Pour le découpage en tuiles, L'équipe GPAC recommande d'utiliser l'encodeur Kvazaar, Nous avons Donc installé et utilisé Kvazaar pour découper le média en tuiles. Nous avons découpé le média avec à chaque fois le même type de découpage en tuiles de même taille. Nous avons juste fait varier le nombre de tuiles en prenant des nombres de tuiles impair (3x3, 5x5, 7x7, etc…) pour toujours avoir une tuile centrale car un des algorithmes de démonstration de la zone d'intérêt implémenté dans le lecteur choisi le centre de l'image comme zone prioritaire. Il est ainsi plus facile de voir un changement de débit si une seule tuile se trouve au centre au lieu de deux. Le découpage en tuile est une opération extrêmement lourde au niveau encodage. A titre d'exemple sur un CPU dual Core 2.6Ghz découper 1 minute de vidéo en 4K 30ips prend plus de 1h30 et il ne faut pas oublier qu'il faut refaire l'encodage à chaque fois que l'on change la résolution ou le nombre de tuiles. Nous avons remarqué au cours de nos différents essais de découpage en tuile que certaines configurations ne sont pas stables. Au-delà de 49 tuiles par image (7x7 tuiles de même taille), le lecteur ne parvient pas décoder le flux correctement et plante après quelques secondes de vidéo. Pour éviter de devoir attendre pendant des heures et des heures la fin de l'encodage nous avons créé un script bash qui prend une vidéo source en entrée et génère le fichier descripteur de flux en découpant au préalable la vidéo en tuiles et en générant plusieurs clones avec des résolutions plus petites. Dans un premier temps, il faut convertir le média source (qui est au format mp4) au format YUV car Kvazaar ne prend pas en charge d'autre format que le YUV. Nous avons réalisé cette conversion en utilisant ffmpeg, un encodeur vidéo très puissant. La commande que nous utilisons pour convertir le média est la suivante :

```
ffmpeg –threads 4 –i media_source.mp4 media_source.yuv
```

Il est important de noter que nous n'utilisons aucune forme de compression de fichier. Cela permet de réduire le temps d'encodage et de préserver au maximum la qualité original du média source. L'option « -threads » permet d'utiliser le multithreading dans l'encodage d'un fichier. Elle est définie à 4 car nous ne disposons que de processeur à 2 ou 4 cœurs maximum.

La seconde étape consiste au découpage en tuiles du média source toujours au format

[1] *https://github.com/gpac/gpac/wiki/HEVC-Tile-based-adaptation-guide*
[2] https://vimeo.com/214402865

YUV. Les encodeurs kvazaar et ffmpeg sont tous les deux capables de réaliser un découpage en tuile cependant kvazaar est utilisé dans les démonstrations de l'équipe GPAC. C'est pourquoi nous avons décidé de l'utiliser. Nous utilisons donc kvazaar avec les options suivantes :

- la source : -i <media_src.yuv>
- la résolution d'entrée : --input-res largeur*hauteur
- la sortie : -o <media_out.hvc>
- le découpage en tuiles : --tiles nb_X*nb_Y
- le bitrate : -bitrate <valeur_en_bit>

Il est important de noter que les valeurs nb_X et nb_y qui correspondent au respectivement aux nombres de tuiles en largeur et en hauteur sont indépendants, on peut par exemple faire un découpage en 5x7 pour avoir 5 tuiles en largeur et 7 en hauteur. Il est important de noter que kvazaar encode chaque tuile dans un fichier hvc séparé. Cela veut dire que plus le nombre total de tuile est grand, plus le nombre de fichier créés est élevé ce qui, dans certains cas, peut poser problème comme nous le verrons dans la partie compte rendu de projet.

Après avoir découpé le media source et les différents clones en tuiles, il faut utiliser MP4Box pour rassembler les tuiles en différents segments de résolutions. Toutes les tuiles d'une même zone qui ont des résolutions différentes sont assemblées en un seul segment. Les segments seront eux même inclus dans le fichier descripteur de flux (mpd). Chaque segment a une taille de 1500 octets maximum.

Une fois que le descripteur de flux est créé, il faut trouver un lecteur média capable de décoder un tel fichier. Cela tombe bien car l'équipe GPAC a aussi développé un lecteur multimédia (MP4Client) capable de lire un descripteur de flux VR tout en utilisant un algorithme de sélection de tuiles. MP4Client est, à l'origine, développé sous MacOs mais y a plusieurs versions portées sous Linux et Windows. Aucun des membres du groupe ne possède de système Apple nous avons donc utilisé les deux autres versions. Dans un premier temps nous avons testé de lire directement le flux en local pour vérifier le bon fonctionnement du lecteur. Le lecteur MP4Client est capable d'ouvrir, de lire et d'adapter le débit vidéo à la bande passante. Un détail important de ce lecteur est qu'il est aussi disponible sur les plateformes mobiles et qu'il est capable d'afficher la vidéo au format casque VR (que l'on soit sur une version mobile ou PC) avec une séparation pour chaque œil. Le lecteur propose aussi une option très intéressante permettant de limiter la bande passante du lecteur pour simuler différentes configurations réseau.

Une fois que tout le traitement vidéo et la création du fichier descripteur de flux est terminée, nous avons mis en place un réseau de test. Nous avons deux architectures de test, la première un seul pc avec un serveur web local, la seconde deux pc connectés à un routeur l'un servant de serveur web l'autre de client. La bande passante en architecture client-serveur séparé est d'environ 4Mbps ce qui n'est pas suffisant pour nos tests. Nous avons donc opté pour le serveur local disposant d'une bande passante plus grande (on élimine donc ce facteur limitant)

Nous allons maintenant analyser nos résultats. Premièrement cette capture (fig. 3) nous servira de référence pour les autres mesures. Elle a été réalisée avec une vidéo ayant une résolution de 4096x2048 pixels et un bitrate 1.25Mbps (ce qui correspond à un débit d'une vidéo en 720p). La vidéo est découpée en 4x4 tuiles Attention les chiffres affichés sont ceux qui correspondent au dernier point de mesure donc dans ce cas à l'arrêt (la fin) de la vidéo. On peut remarquer que la vidéo charge bien, le buffer se charge et se décharge régulièrement, le nombre d'image par secondes est constant (= 30ips) tout comme la qualité et le débit. On voit donc que le lecteur arrive parfaitement à afficher la vidéo de manière fluide. Les tests sont réalisés avec un serveur local sur la même machine.

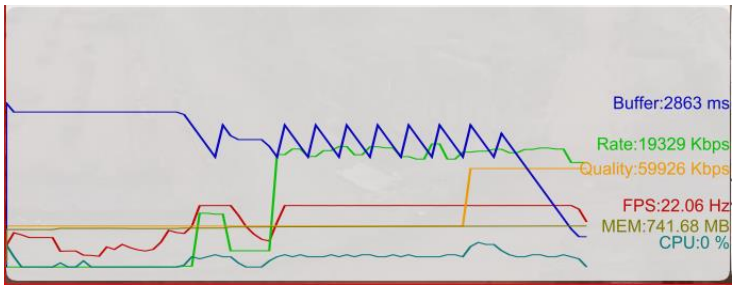


**Figure 4 : statistiques de lecture de la vidéo (1Mbps)**

**Figure 5 : échantillon de qualité d'image**

Passons maintenant aux tests réels. Tout d'abord il faut une explication du contexte, nous utilisons la même vidéo que précédemment, découpée en 16 tuiles de même taille (4x4) et encodée avec 6 bitrates différents 750Kbps, 1,5Mbps, 3Mbps, 6Mbps, 13Mbps et 45Mbps. Bitrates qui correspondent globalement aux résolutions suivantes : 256x144, 640x360, 1280x720, 1920x1080, 2560x1440 et 4096x2048. Nous utilisons toujours la machine la plus puissante que nous ayons à disposition avec un serveur en local.

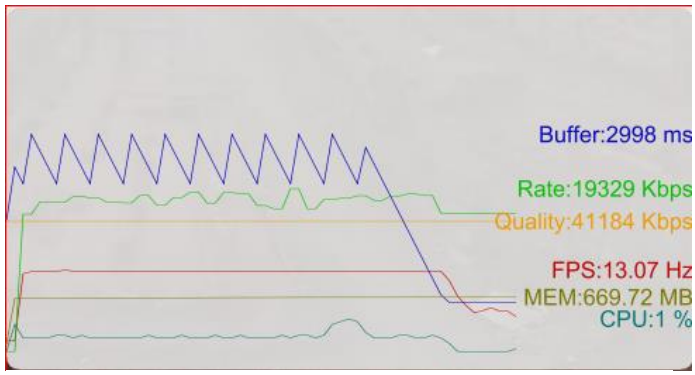


**Figure 3 : statisques de référence (sans multi bitrate)**

Cette première capture est réalisée avec une bande passante maximum de 1Mbps. On remarque que le lecteur ne change pas de qualité (courbe de qualité stable) car il ne dispose pas d'un bitrate de 1Mbps, il dispose de 750Kbps ou de 1,5Mbps. 1,5Mbps étant trop élevé

pour la bande passante disponible, la lecture serait saccadée. La qualité d'image est très basse, elle ressemble à de la 640p (fig. 5). On peut observer une légère perte d'ips au milieu de la courbe rouge cela est causé par le déplacement des fenêtres graphiques dans l'interface du lecteur, c'est un problème d'actualisation de l'interface du lecteur. Pour résumer cette capture, le lecteur s'adapte bien à la bande passante car il choisit de ne pas changer de qualité pour maximiser la fluidité du visionnage.

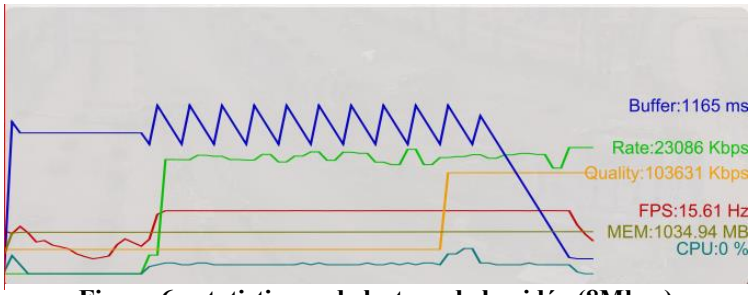


**Figure 6 : statistiques de lecture de la vidéo (8Mbps)**

**Figure 7 : échantillon de qualité d'image**

Avec une limitation de bande passante à 8 Mbps, le lecteur choisi un bitrate plus élevé. Il dispose d'un bitrate de 6Mbps et un autre de 13Mbps, il prend celui à 6Mbps qui correspond à une qualité d'image en fullHD (fig. 7). Les ips sont stables tout comme le chargement et le déchargement du buffer, le lecteur n'a aucun problème pour lire la vidéo : l'expérience est fluide.

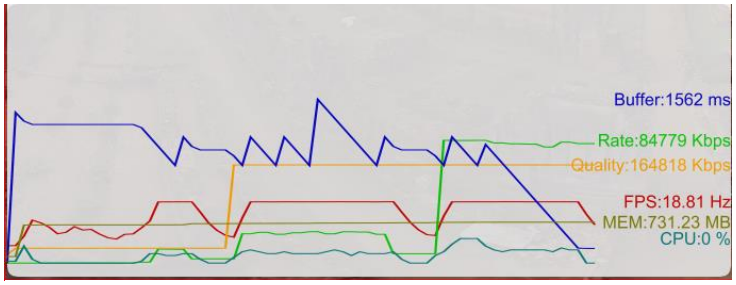


**Figure 8 : statistiques de lecture de la vidéo (pas de limite de bande passante)**

**Figure 9 : échantillon de qualité d'image**

Sans aucune limitation de bande passante, on observe que la qualité d'image est très élevée c'est de la 4k (4096p). Cependant on remarque que la courbe des ips n'est pas stable. Elle n'est pas stable pendant les premières secondes de chargement du lecteur (quand il commence à télécharger et afficher les images) elle est ensuite quasiment constante le reste du temps avec des arrêts sur image de temps en temps. C'est arrêt sont causé par une incapacité

du lecteur à gérer les très hautes résolutions. Le buffer est quasiment stable si l'on ne prend pas en compte le début ou sa taille s'adapte à la résolution et le pic inexplicable au milieu du visionnage. Il est possible qu'il ait tenté de téléchargé d'autres segments de différents bitrates en parallèle. Dans l'ensemble, l'expérience de visionnage est assez fluide pendant une grande majorité du temps.

# 6 Compte rendu de projet

Le but de ce projet était de mettre en place une chaine de streaming dynamique complète en utilisant l'encodage différencié avec les régions d'intérêt.

Nous avons réussi à mettre en place deux serveurs web basiques (apache), un sur la première machine (celle sous Debian 9) et l'autre sur la machine client qui permet de tester la lecture des fichiers directement en local. Il est possible d'accéder directement au fichier stocké sur disque depuis GPAC seulement si le dossier en contient plus de 10 000, le lecteur plante. La meilleure solution que nous avons trouvée pour contourner ce problème est de donner directement le lien du fichier se trouvant sur le serveur local du client. Cela permet d'éviter à GPAC de scanner de nombreux fichiers.

Une fois le serveur en place il a fallu trouver un media source, nous avons d'abord travaillée avec une vidéo de 30 secondes. Cependant après quelques tests rapides, nous nous sommes aperçus que ce n'était pas assez long pour observer la gestion automatique du bitrate. Nous avons donc utilisé un logiciel de montage vidéo basique pour dupliquer la vidéo un certain nombre de fois jusqu'à obtenir 2 medias source, le premier long de 1 minute et le second long de 3 minutes. Ces deux média nous ont servis de base pour le reste de nos tests.

L'encodage au format YUV est assez fastidieux car il est non seulement obligatoire pour travailler avec kvazaar mais il demande un temps et une puissance de calcul énorme. Par exemple pour notre média de test de 3 minutes, il a fallu plus de 40 minutes. De plus les fichiers générées au format YUV sont incroyablement lourds, environ 50 Go pour 3 minutes de MP4. C'est même à cause de cette taille de fichier que nous n'avons pas testé des vidéos de plus de 3 minutes. Apres avoir obtenu le fichier YUV, le découpage avec kvazaar est quasiment un jeu d'enfant, il suffit d'utiliser les paramètres souhaités et de lancer la commande. Nous avons développé un script qui automatise le découpage et le dashing des segments créés pour ne pas avoir à rester devant l'ordinateur pendant le découpage car bien qu'étant simple à réaliser, il prend du temps et beaucoup de ressource, surtout si on veut avoir un grand nombre de bitrate différents pour pouvoir tester différent paramètres de bande passante.

Nous avons rencontré un certains nombres de problèmes pendant ce projet, des problèmes d'encodage vidéo et des problèmes de lecture du flux généré. Les problèmes d'encodages vidéo par exemple des images mal encodées ou ayant un bitrate trop faible ont été réglés rapidement en testant de nombreux paramétrages de kvazaar. Les problèmes liés aux temps de calcul que cela soit avec ffmpeg ou kvazaar ont été résolu dans les limites de nos machines, nous ne pouvons pas monter au-dessus de 4 processus en parallèle pour décoder car nous n'avons tout simplement pas la configuration requise. Les problèmes

concernant le lecteur GPAC n'ont pas tous été résolus. Dans un premier temps il y a les limitations logicielles que nous avons rencontrées. En effet GPAC limite (malgré lui, le lecteur plante si on pousse les réglages plus haut) le nombre de tuiles que l'on peut découper dans une vidéo. Nous avons par exemple testé avec les découpages suivants 3x3, 4x4, 5x5, 6x6, 7x7, 9x9 et 11x11 tuiles. Le lecteur charge les tuiles sans aucun problème jusqu'à 7x7 tuiles au-delàs de cette limite, la vidéo ne charge tout simplement pas ou le lecteur crash et cela même si on diminue expressément le bitrate à une valeur ridicule pour réduire au maximum le temps de téléchargement de chaque segment. Il est important de préciser que cette limite est la même sur la version Windows et Linux. Nous nous sommes donc accommodés de cette limite c'est pourquoi le maximum de tuiles que nous avons testé est de 49. Un second problème que nous avons rencontré est la configuration du lecteur. Il dispose d'un fichier de configuration nommé « gpac.cfg » qui permet de régler les paramètres du streaming DASH, notamment l'algorithme de sélection des bitrates, l'adaptation par rapport à la bande passante disponible ou par rapport à la capacité du buffer. Nous avons testé plusieurs configurations différentes mais nous avons à chaque fois obtenu les mêmes résultats. Le lecteur force toujours un bitrate égal sur l'ensemble des tuiles et cela même forçant la priorité de certaines tuiles à l'encodage en spécifiant la QPMAP dans un fichier texte. La « quality priority map » est une simple matrice d'entiers positif donnant la priorité de chaque tuile, 0 étant la plus prioritaire. Elle permet par exemple de définir la zone centrale comme la zone la plus prioritaire (qui reçoit le bitrate le plus élevé). C'est deux limitations du lecteur GPAC nous ont poussé à chercher un autre lecteur plus stable. Nous avons fini par trouver un projet de lecteur DASH en JavaScript. Nous avons tenté de l'implémenter dans une page web en utilisant le navigateur Edge de Microsoft car il est le seul compatible nativement avec le projet. Ce fut un échec et une perte de temps même après avoir essayé de débugger le code source en JavaScript qui posait problème. Nous avons donc abandonné cette option et nous sommes revenus à GPAC. Après avoir longuement testé les différentes configurations, nous avons trouvé que la configuration en mode égal (toutes les tuiles ont le même bitrate) était la meilleure solution en termes de stabilité du lecteur, c'est celle qui ne faisait pas planter le lecteur. Dans la documentation du lecteur il est dit succinctement que seules les tuiles affichées à l'écran sont chargée et que ces dernières forment la ROI, c'est-à-dire qu'elles reçoivent un bitrate plus élevé. Cependant d'après nos observation, l'intégralité des tuiles est chargée dans le buffer même si celles si ne sont pas visibles car la bande passante minimale requise doit être supérieur ou égale au bitrate de la vidéo complète pas uniquement le bitrate d'une seule tuile. Cela ne reste qu'une supposition car il faudrait analyser intégralement le code source du lecteur pour être sûr de son comportement. Nous avons aussi réalisé une documentation sous forme d'un fichier texte, comme annoncé dans le cahier des charges, pour permettre aux futures personnes intéressées par le sujet de gagner un temps très important et de leur éviter de parcourir de sombres forums. La documentation ainsi que nos différents outils, scripts et paramétrages seront disponible pour une durée indéterminée (probablement jusqu'à fermeture du site GitHub, ce qui ne risque pas d'arriver de sitôt).

En conclusion nous avons réussi à mettre en place la chaine de streaming complète ainsi que la création, ou plutôt le traitement d'un média pour le rendre compatible avec le streaming DASH. Concernant l'adaptation dynamique du bitrate, nous avons réussi à avoir une vidéo qui charge différentes qualité en fonction de la bande passante disponible.

Cependant nous n'avons pas réussi à contrôler l'algorithme de manière précise, tous les paramétrages que nous avons testés ne semblent avoir aucun impact et le seul mode de l'algorithme qui semble fonctionner est celui qui attribue le même bitrate à toutes les tuiles. Pour la région d'intérêt, nous n'avons pas parfaitement maitrisé l'adaptation de la région d'intérêt par rapport à la zone de la vidéo qui est visible. De plus beaucoup d'inconnues perturbent nos résultats, la configuration réseau de test étant la plus importante. Notre environnement de test est un réseau domestique ou avec plusieurs appareils connectés au même routeur qui perturbent la bande passante locale. La seconde inconnue très importante est l'effet boite noir du lecteur GPAC. En effet bien qu'une documentation soit disponible, elle ne détaille pas le fonctionnement exact du lecteur quant au choix du bitrate et de la région d'intérêt. Il faudrait, pour avoir des résultats vraiment exploitables, mettre en place une configuration réseau de laboratoire avec une connexion client-serveur dédiée aux tests du lecteur. Un dernier point qu'il est important de souligner c'est le fait que dans toutes les démonstrations faites par l'équipe GPAC et disponibles sur internet, ils n'utilisent jamais les versions sous Windows et Linux, uniquement celle sous MacOs. Est-il possible que la version sous MacOs soit plus stable et que les versions portées sur les autres OS le soit moins ? Mettre en place un environnement de test sous MacOs pourrait être une piste de recherche pour approfondir le sujet et permettrait de vérifier que la stabilité du lecteur n'est pas responsable d'erreurs de mesure. Le découpage en Tuile est aussi un sujet à approfondir, pourquoi pas avoir un découpage encore plus fin, pourquoi pas de l'ordre du pixel!