INSA Lyon – Département Télécommunications
Année universitaire : 2025 – 2026

Analyse architecturale et optimisation de la transmission d'écran via WebRTC sur plateforme ARM (Raspberry Pi 4)

Encadrant :
Stéphane Frenot - Damien Reimert

Réalisé par :

- Orhon Gabriel
- Chkoundali Yasmine
- Mohammi Islam
- Abidi Jean
- Adjami Axel

Sommaire

1. Introduction

La projection d'écran sans fil via WebRTC sur plateforme embarquée soulève des contraintes à la fois algorithmiques, matérielles et protocolaires. Le principal verrou technique provient du décalage structurel entre :
- Les stratégies d'encodage sophistiquées des navigateurs modernes (Chrome/Chromium), optimisées pour des machines x86 puissantes
- Les capacités réelles de décodage matériel du SoC Broadcom BCM2711 équipant le Raspberry Pi 4.

Ce document propose une analyse rigoureuse des mécanismes internes des codecs vidéo, du processus de négociation SDP, ainsi que des causes profondes de la dégradation du rendu des contenus textuels (tableurs, interfaces bureautiques). L'objectif est d'expliquer *pourquoi* le système échoue par défaut et *comment* contraindre la chaîne WebRTC pour obtenir un flux stable, peu latent et visuellement fidèle.

2. Analyse structurelle et mathématique des codecs vidéo

Un codec vidéo n'est pas un simple algorithme de compression : il repose sur une modélisation mathématique de la redondance spatiale et temporelle de l'image.

2.1 H.264 / AVC (Advanced Video Coding)

Le codec H.264 repose sur une décomposition spatiale de l'image en macroblocs (jusqu'à 16×16 pixels), eux-mêmes subdivisables en sous-blocs adaptatifs.

Principe mathématique
- Chaque bloc est transformé via une Transformée en Cosinus Discrète (DCT), projetant les valeurs spatiales dans le domaine fréquentiel.
- Les coefficients de haute fréquence (détails fins) sont fortement quantifiés ou supprimés, conformément aux limites de la perception visuelle humaine.

Profiles et Levels
- Le standard H.264 définit des *profiles* (Baseline, Main, High) et des *levels* encadrant la complexité maximale (résolution, débit, nombre de références).
- Le Raspberry Pi 4 dispose d'un décodeur matériel optimisé pour H.264 High Profile – Level 4.1.
- Toute négociation au-delà (ex. Level 5.x) force un décodage logiciel sur CPU ARM, entraînant une explosion de la latence et une chute drastique du framerate.

Filtre de déblocking: Afin de masquer les discontinuités entre macroblocs, H.264 applique un filtre adaptatif de lissage. Ce mécanisme, bénéfique pour la vidéo naturelle, est nuisible aux interfaces graphiques : les contours fins (polices, grilles Excel) perdent leur netteté.

2.2 VP8 et VP9 (Google / AOMedia)

VP9: Introduit des superblocks jusqu'à 64×64 pixels, découpables récursivement (quadtree asymétrique). Cette flexibilité améliore l'efficacité de compression mais augmente fortement la complexité algorithmique.
VP8: Codec à prédiction intra/inter plus simple, utilisant 10 modes de prédiction spatiale et moins coûteux que VP9 mais moins robuste face aux mouvements rapides.

Limitation matérielle critique
- Le SoC Broadcom du Raspberry Pi 4 ne possède aucune unité de décodage matériel VP9.
- Le décodage repose donc exclusivement sur le CPU ARM Cortex-A72.
- Chaque itération de découpage, de prédiction et d'inversion de transformée est exécutée en calcul arithmétique pur, saturant rapidement les quatre cœurs.
→ Conséquence directe : surcharge CPU, montée thermique (>80 °C), *thermal throttling*, instabilité globale du système.

3. Négociation SDP et logique de sélection des codecs

Le Session Description Protocol (SDP) constitue le contrat formel entre l'émetteur et le récepteur. Il ne transporte pas de données multimédia, mais décrit comment celles-ci doivent être échangées.

3.1 Structure pertinente d'un SDP vidéo

Les champs déterminants sont :
- `m=video` : média, port et protocole (RTP/SAVPF).
-`a=rtpmap` : association Payload Type ↔ codec (ex. `H264/90000`).
-`a=fmtp` : paramètres fins du codec.
Pour H.264, `fmtp` inclut notamment : `packetization-mode et profile-level-id`
Ces champs conditionnent directement la compatibilité avec le décodeur matériel du Raspberry Pi.

3.2 Échec de la négociation automatique

Les navigateurs Chromium appliquent une logique de préférence interne : VP9 est priorisé pour son efficacité de compression et le H.264 est relégué en solution de repli.

Scénario réel :
- Le PC annonce : VP9 (priorité haute), H.264 (priorité basse).
- Le Raspberry Pi accepte VP9 car il est théoriquement supporté.
- Le flux démarre en VP9 → décodage logiciel → saturation CPU → effondrement des performances.
→ Ce comportement est conforme au standard, mais inadapté à une plateforme embarquée.

4. Rendu textuel, chrominance et sous-échantillonnage

4.1 Représentation YUV et perception humaine

Les flux vidéo sont encodés en **YUV** :
- **Y (Luma)** : intensité lumineuse.
- **U/V (Chroma)** : composantes colorimétriques.

Le sous-échantillonnage **4:2:0** conserve :
- 100 % de la luminance.
- seulement 25 % de l'information de couleur.

→ Cette approximation est acceptable pour la vidéo naturelle, mais destructrice pour les interfaces graphiques.

4.2 Cas pathologique des tableurs

Un tableur présente : des contrastes abrupts (noir/blanc), des motifs fins et répétitifs et des aplats colorés adjacents.
En 4:2:0 : la chrominance est interpolée spatialement et les transitions de couleur débordent sur les pixels voisins.
→ Effet visuel : halos colorés, contours flous, texte « baveux ».

4.3 Rôle du QP et du contentHint

Le Quantization Parameter (QP) contrôle le niveau de perte :
- QP élevé → forte compression, perte de détails.
- QP faible → fidélité accrue, débit plus élevé.

Le paramètre `contentHint = "text"` informe l'encodeur WebRTC que le flux contient des structures fines et statiques à fort contraste, ce qui l'amène à réduire la quantification et à préserver les contours, améliorant significativement la lisibilité des contenus bureautiques malgré le sous-échantillonnage chromatique.

5. Conclusion

Les dysfonctionnements observés ne relèvent pas d'un bug logiciel, mais d'une inadéquation architecturale entre les choix par défaut de la chaîne WebRTC et les capacités réelles de la plateforme embarquée. WebRTC et les navigateurs modernes sont principalement optimisés pour la transmission de vidéo fluide, ce qui conduit Chrome à privilégier automatiquement le codec VP9 pour son efficacité de compression. Or, le Raspberry Pi 4 repose quasi exclusivement sur le décodage matériel H.264, le décodage VP9 étant entièrement logiciel et donc prohibitif en termes de charge CPU et de latence. Cette divergence est accentuée par la nature des contenus projetés, majoritairement statiques et fortement textuels, qui ne correspondent pas aux hypothèses de conception des encodeurs vidéo temps réel. Une solution robuste consiste dès lors à forcer l'utilisation du codec H.264 compatible matériellement par modification du SDP, à éliminer VP9 de la négociation, et à contraindre l'encodeur via le paramètre *contentHint* = *"text"*, afin de préserver les contours et la lisibilité. Cette approche permet de réaligner l'ensemble de la chaîne WebRTC avec les contraintes matérielles et fonctionnelles du système embarqué.