Transmission de données multimédia

pOLYTECH’ NICE SOPHIA ANTIPOLIS |Master SSTIM / VIM

TP3 flux video et audio

Guénon Marie / Favreau Jean-Dominique / Tanguy Arnaud

2014

Table des matières

[Initialisation 2](#_Toc379525472)

[1. La salle 310 2](#_Toc379525473)

[2. Logiciels 2](#_Toc379525474)

[Flux d’image 3](#_Toc379525475)

[1. Caméra 3](#_Toc379525476)

[Avec un navigateur 3](#_Toc379525477)

[Avec gstreamer 4](#_Toc379525478)

[Décomposition du flux 4](#_Toc379525479)

[Ce flux en réseau 5](#_Toc379525480)

[2. Un autre streaming 6](#_Toc379525481)

[Avec decodebin 6](#_Toc379525482)

[On décompose plus finement 6](#_Toc379525483)

[Sauvegarde directe 6](#_Toc379525484)

[Sauvegarde et visualisation 7](#_Toc379525485)

[3. Fichier 8](#_Toc379525486)

[Sons 9](#_Toc379525487)

[1. Fichier 9](#_Toc379525488)

[Lecture 9](#_Toc379525489)

[Envoi UDP 9](#_Toc379525490)

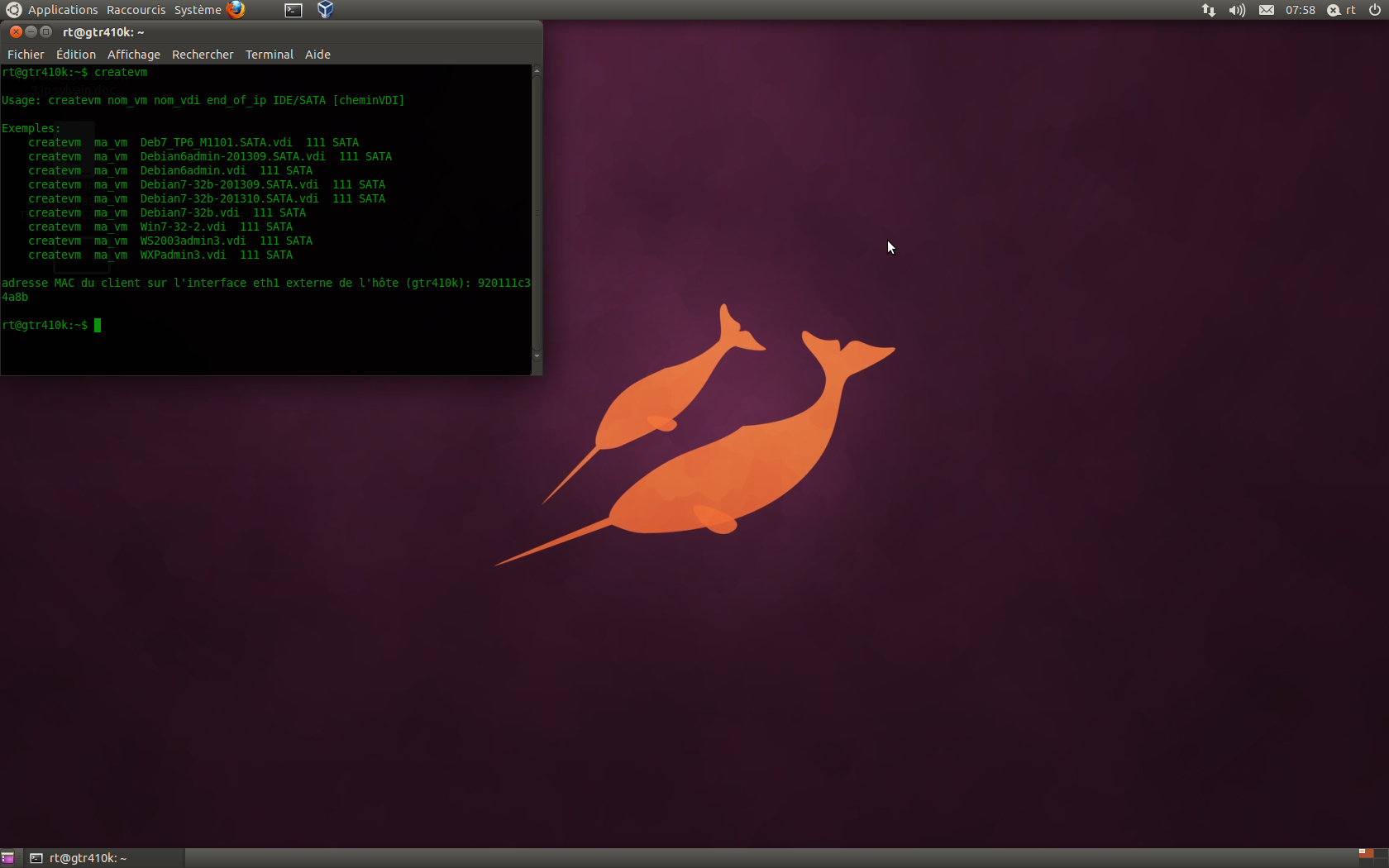
[Envoi RTP/UDP 10](#_Toc379525491)

[Vidéo 11](#_Toc379525492)

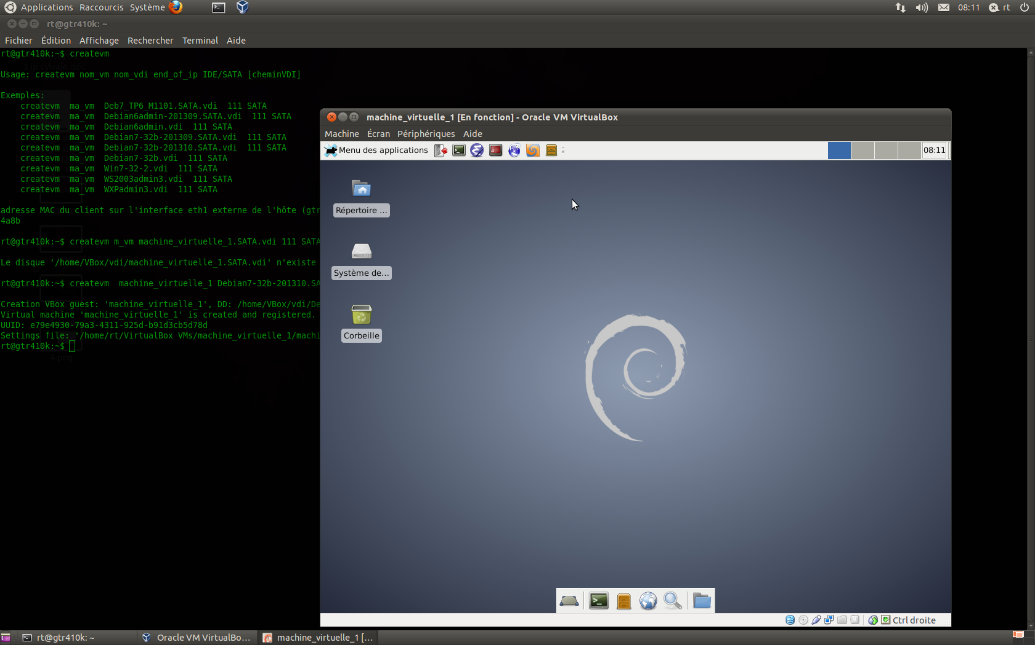
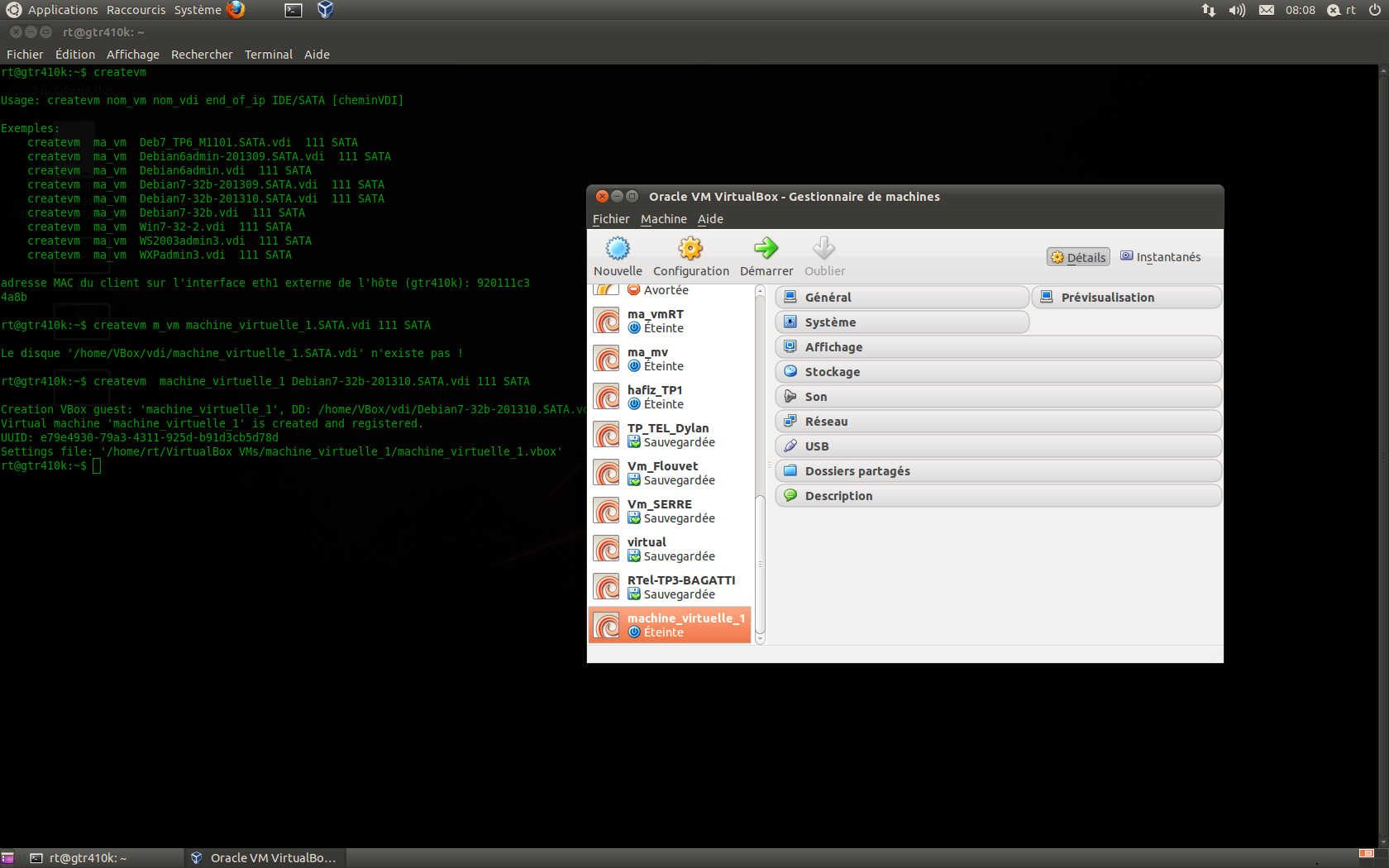
# Initialisation

## La salle 310

Grâce à la commande *creatvm* nous créons une machine virtuelle :

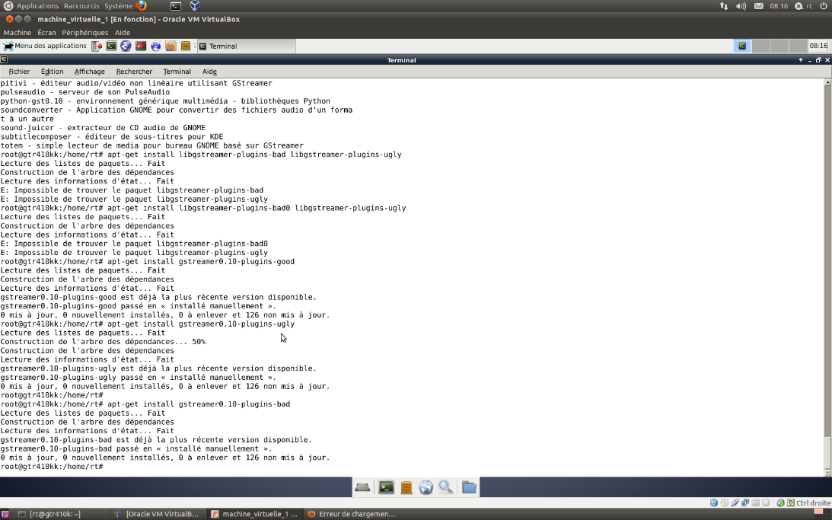


Puis nous lançons la machine virtuelle crée précédemment.



## Logiciels

Sur la machine virtuelle fraichement crée, les logiciels dont nous allons avoir besoin par la suite ne sont pas à jour. Nous avons donc dû les mettre à jour ainsi qu’installer les logiciels non présents, tels que *vlc, gstreamer*,…



# Flux d’image

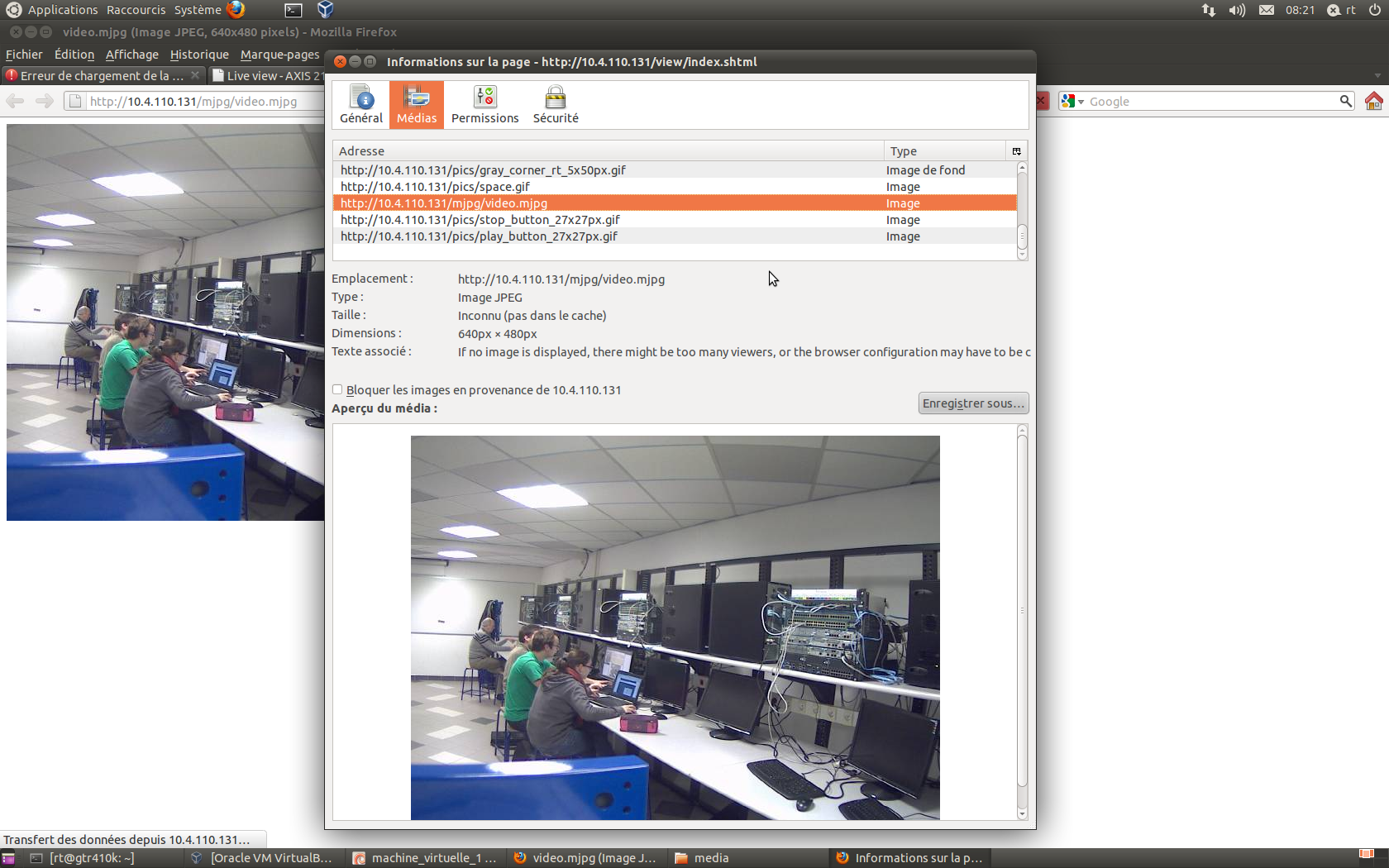
## Caméra

Une fois la machine virtuelle lancée et à jour, nous avons eu à récupérer et afficher les images fournies en temps réel par la caméra Axis. Pour cela nous avons utilisé deux méthodes :

### Avec un navigateur

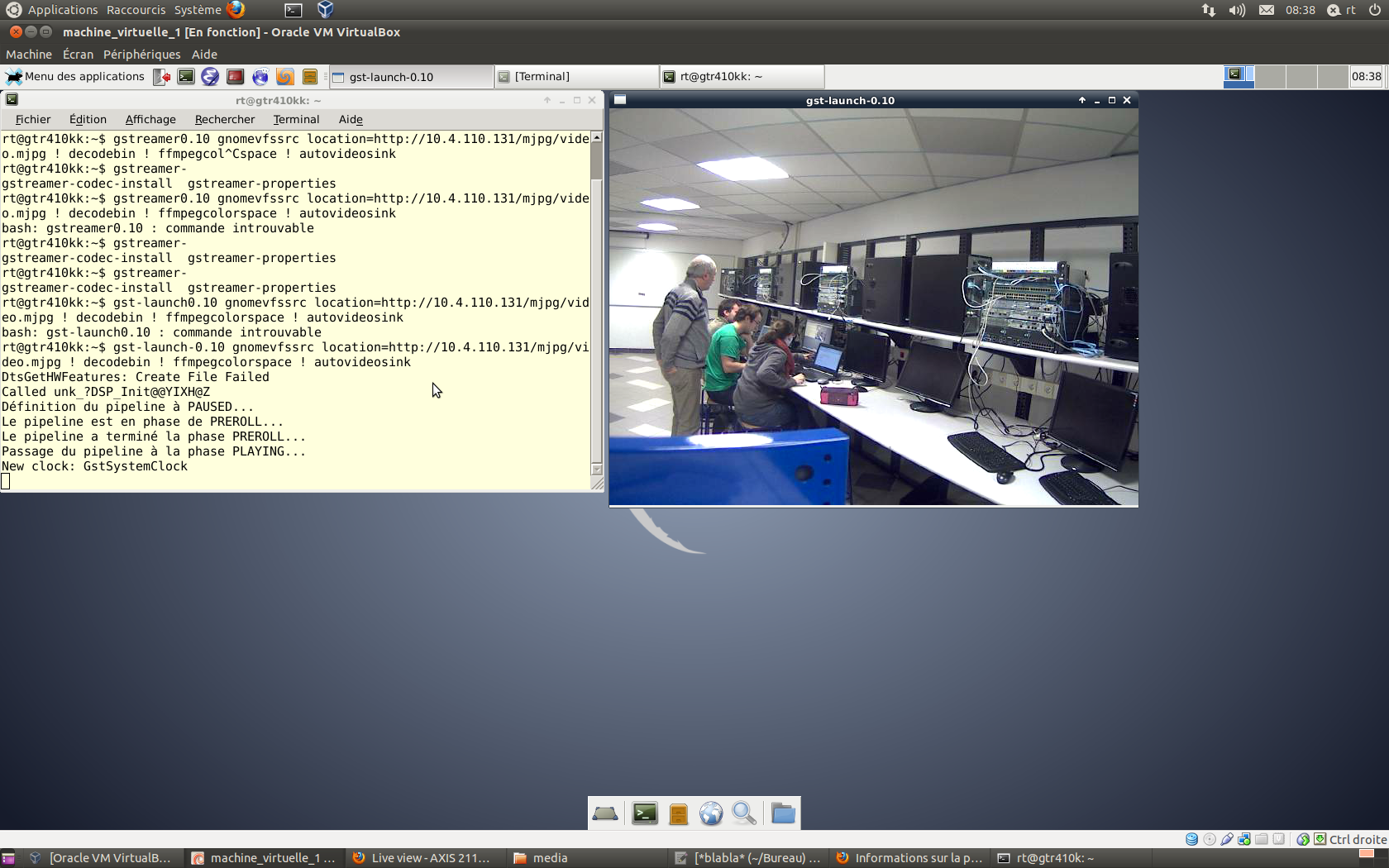
Notre première méthode a été de récupérer les images envoyées par la caméra grâce un navigateur web. Pour cela, nous pouvons accéder à deux adresses différentes et récupérer deux flux de type différents.

* A l’adresse <http://10.4.110.131/mjpg/video.mjpg>, nous récupérons un flux encapsulé en MJPEG. C’est-à-dire une suite d’images statiques affichées et rafraichies suffisamment souvent pour que l’œil humain croie à des mouvements fluides.
* A l’adresse <rtps://10.4.110.131:554/mpeg4/media.amp>, nous récupérons un flux encapsulé en MPEG4. C’est-à-dire une vidéo.



### Avec gstreamer

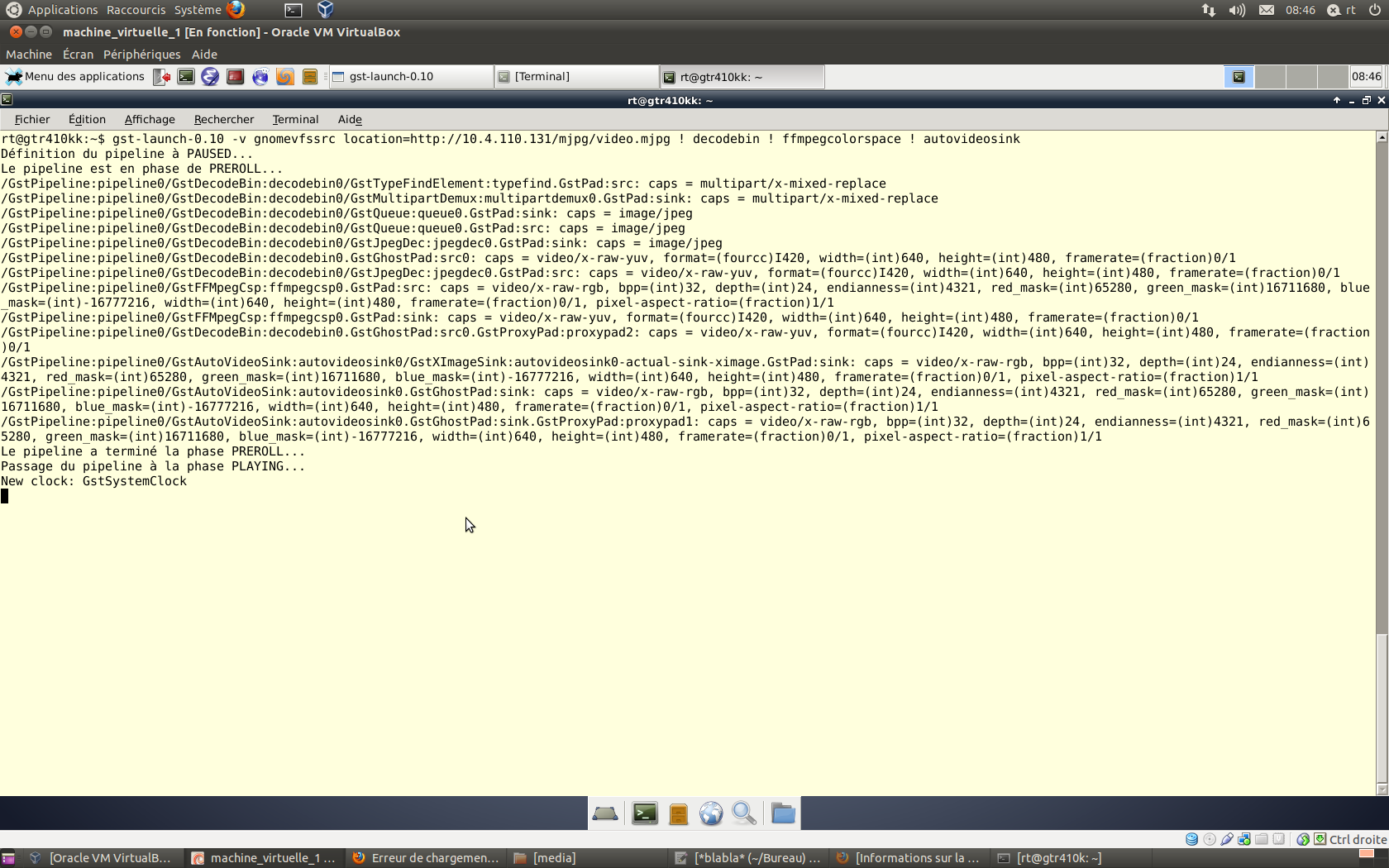
La deuxième méthode que nous avons utilisée consistait à utiliser la commande *gstreamer* pour construire une chaîne qui traite notre flux vidéo en agençant les traitements les uns après les autres.



### Décomposition du flux

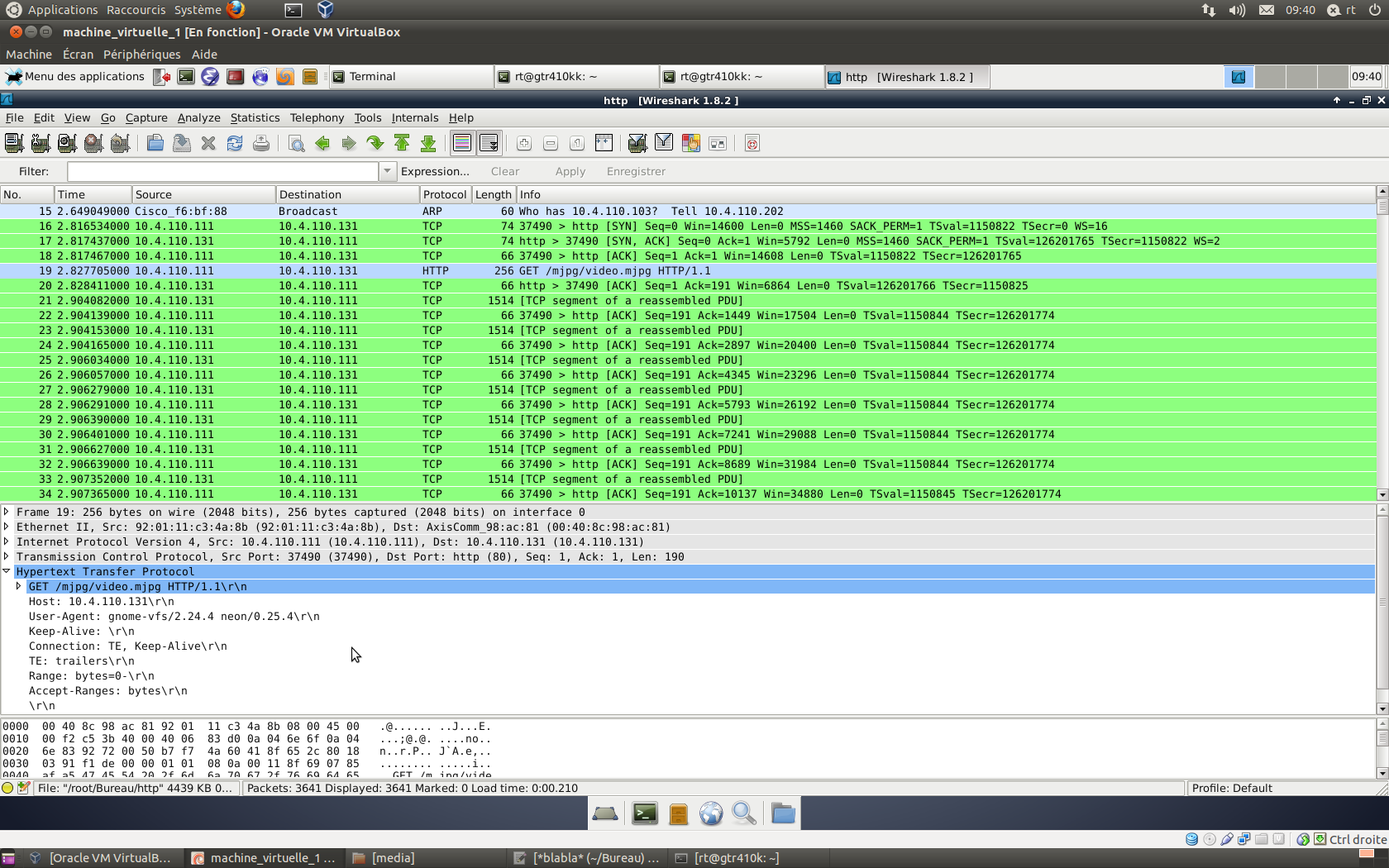
Lorsque l’on décompose la commande *gstreamer*, on se rend compte que les plugins utilisés sont *jpegdec0, ffmpegCsp, autovideosink0.*

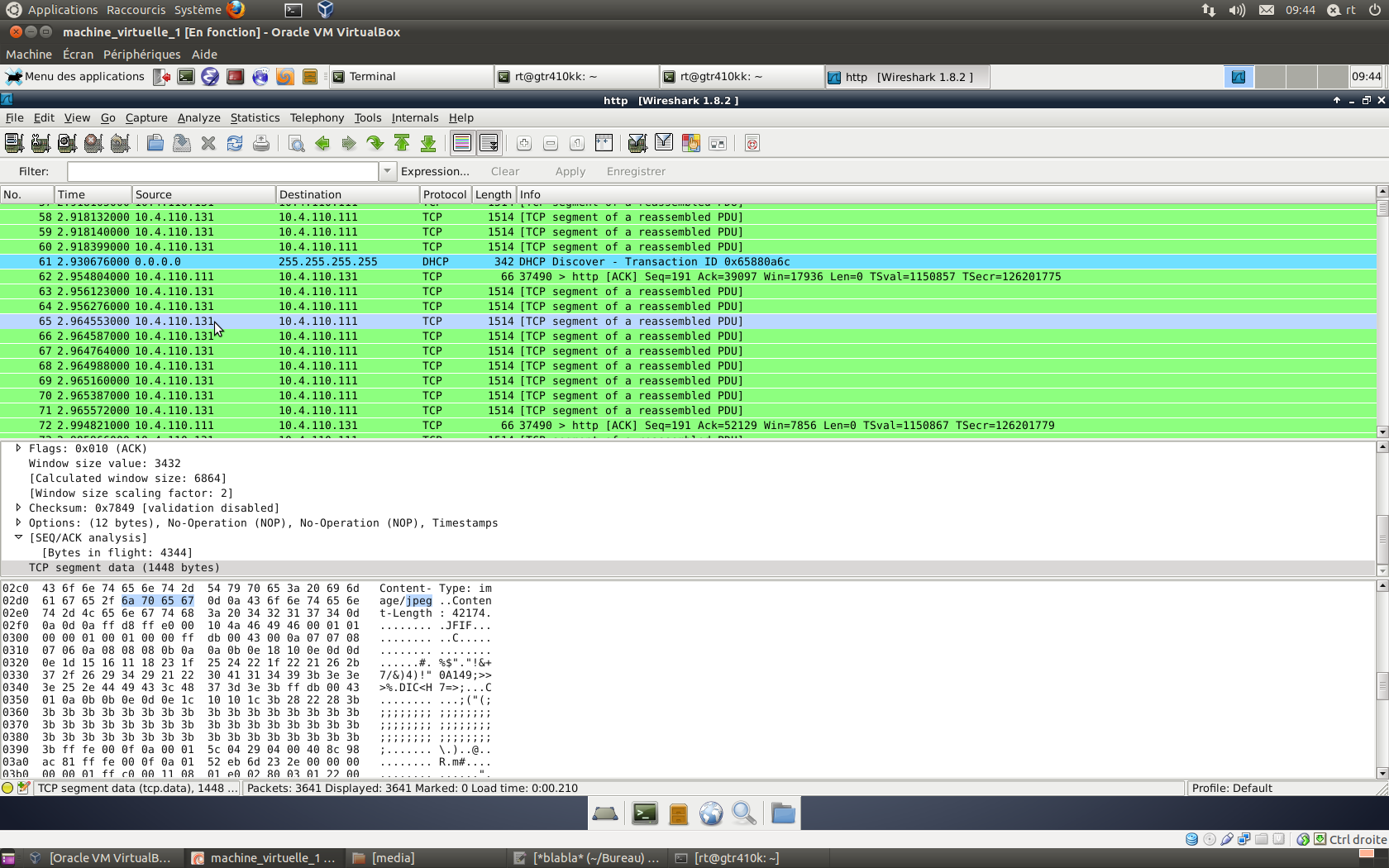
De plus, nous avons pu voir que les images envoyées par la caméra sont des images jpeg codées en YUV (luminance/chrominance). Ceci est lié au fait que ce sont les variations sur ces paramètres qui sont le mieux perçues par l’œil humain. Au contraire, les images sont reconverties pour être affichées en RGB, car c’est ce type d’affichage qui est le mieux géré par l’ordinateur.



### Ce flux en réseau

Le but était ici d’analyser les protocoles utilisés pour envoyer la vidéo de la caméra à l’ordinateur.  
Nous avons pu constater que la connexion est initialisée par une requête HTTP, après quoi les frames en jpeg sont envoyées par TCP.

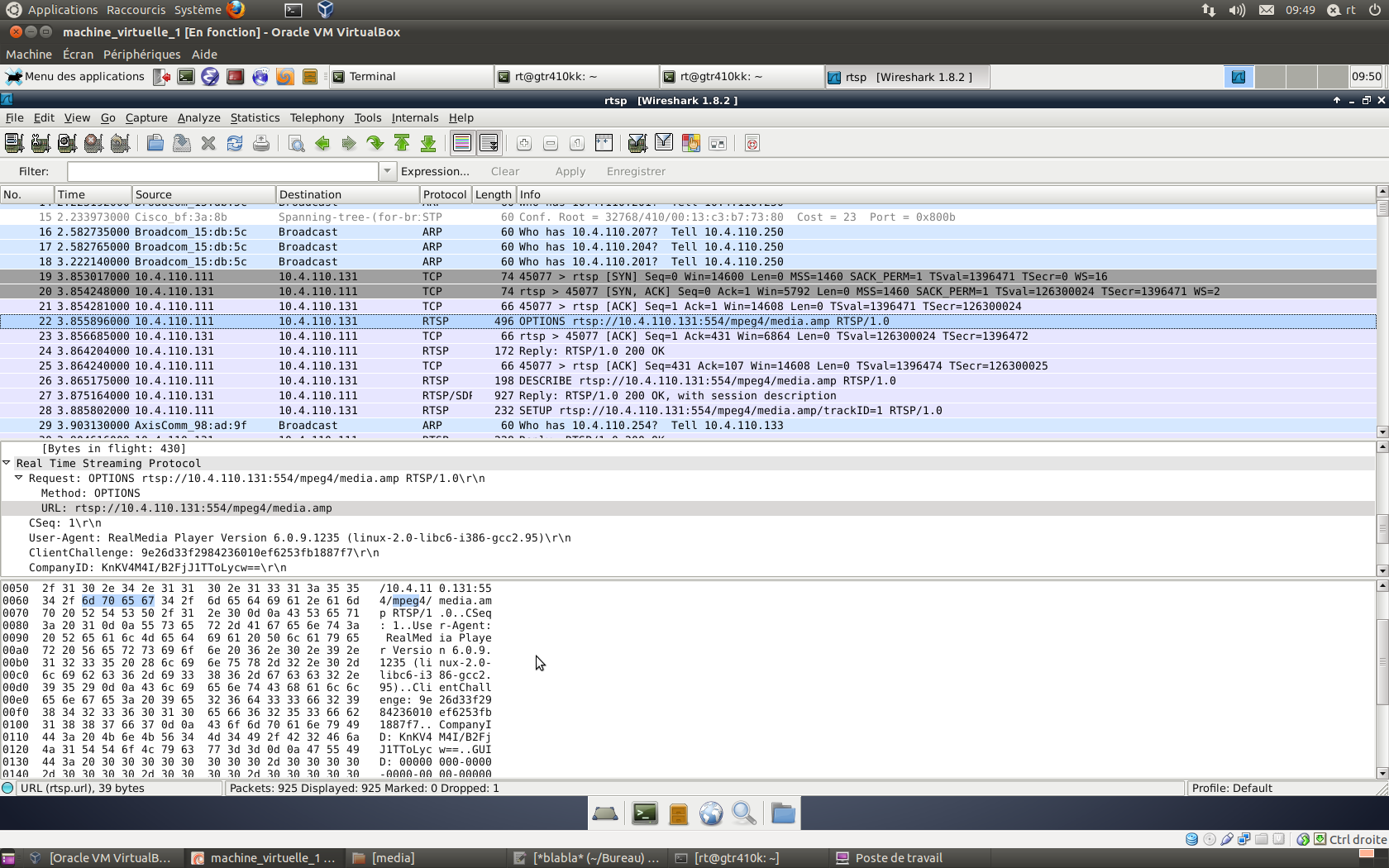




## Un autre streaming

### Avec decodebin

Nous avons lancé la lecture de la vidéo avec *gstreamer* et avec le plugin *decodebin*, puis nous avons analysé les flux envoyés grâce à *wireshark*.

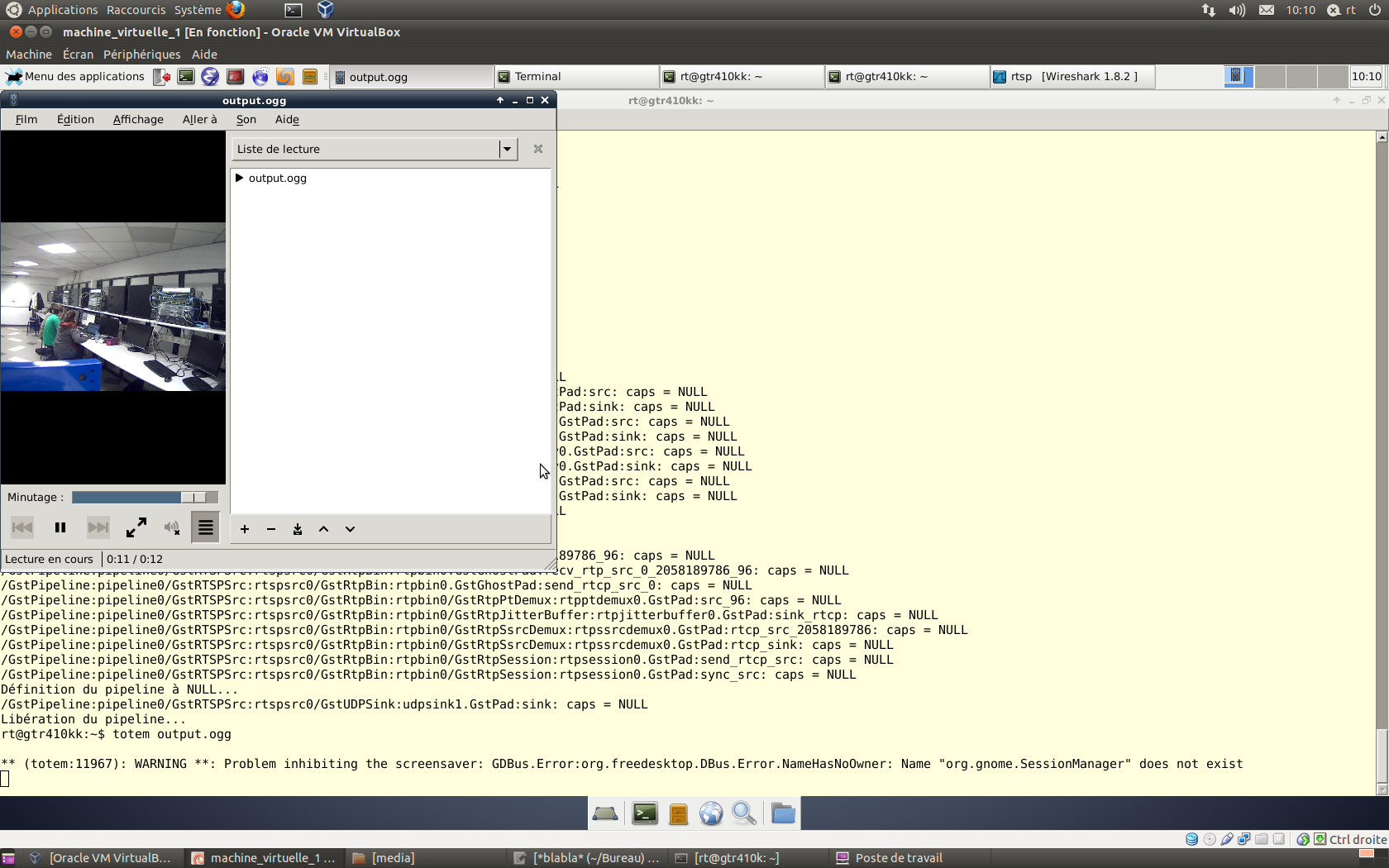


Nous pouvons par ailleurs constater que le transport utilisé ici est le protocole RTP.

### On décompose plus finement

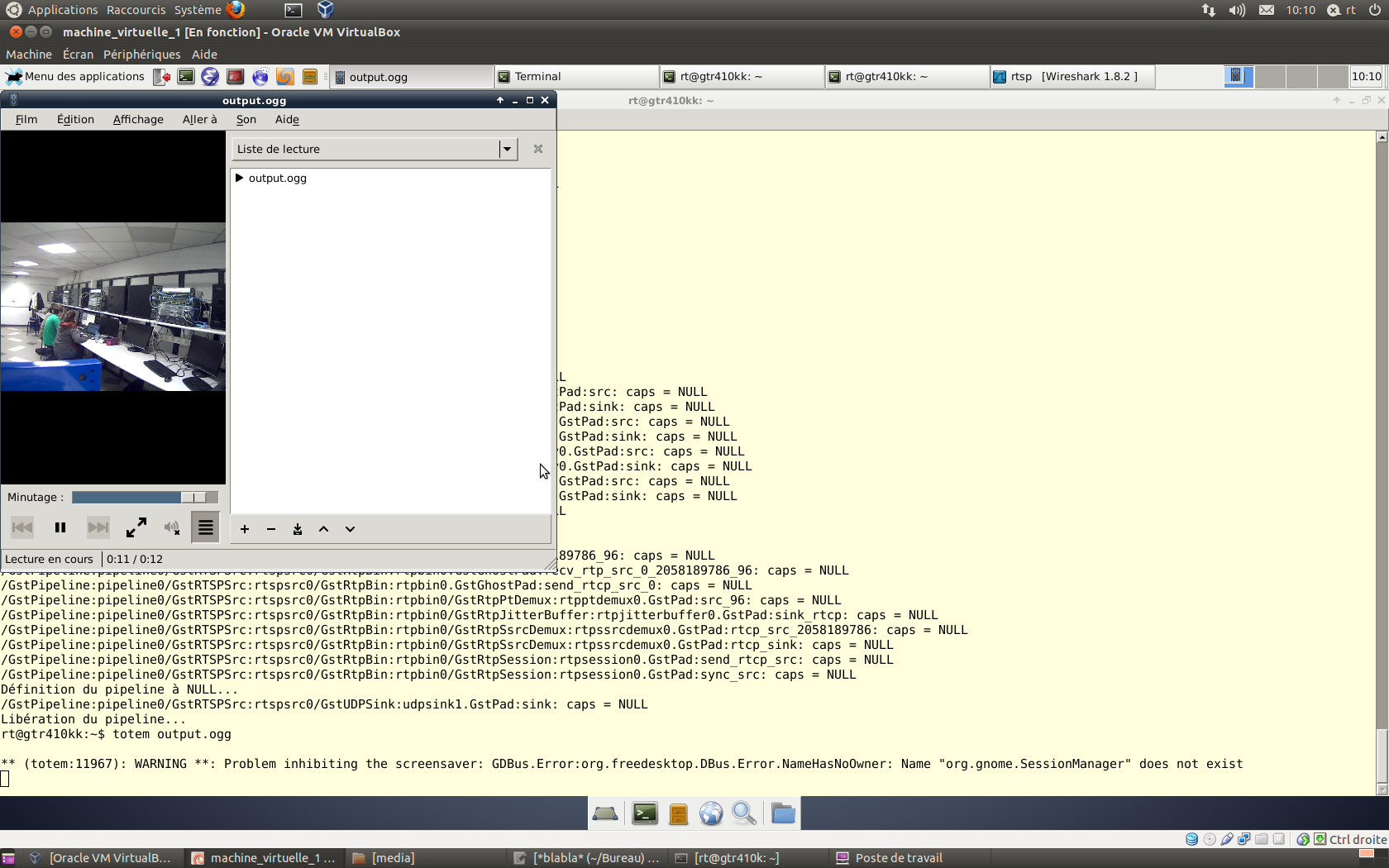
### Sauvegarde directe

Dans cette partie, nous avons enregistré le flux de sortie de la caméra dans un fichier *output.ogg* grâce à la commande *gst-launch*. Il est important de noter dans cecas que l’on ne voit pas directement le flux vidéo que l’on enregistre, mais l’on vérifie à posteriori l’efficacité de la méthode en lisant la vidéo avec un lecteur de film.



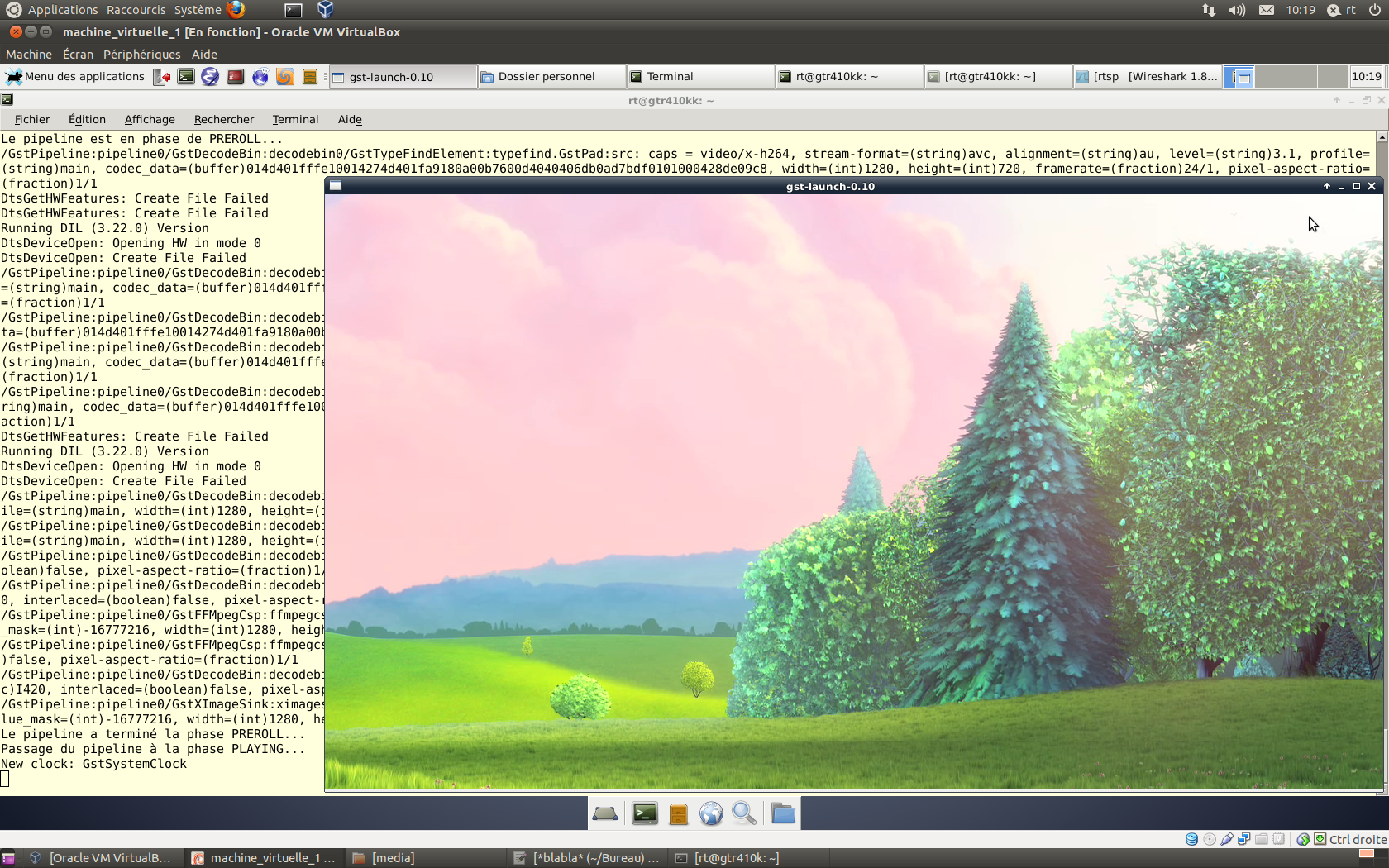
### Sauvegarde et visualisation

Nous avons fait évoluer notre manière de sauvegarder le flux vidéo. En effet, maintenant nous voyons les images sortant de la caméra en même temps que la sauvegarde se fait.



## Fichier

Nous avons récupéré par *ftp* le fichier *big\_bunny\_720p\_h264.mov* et nous l’avons visualisé (sans le son) grâce à *gstreamer.*



# Sons

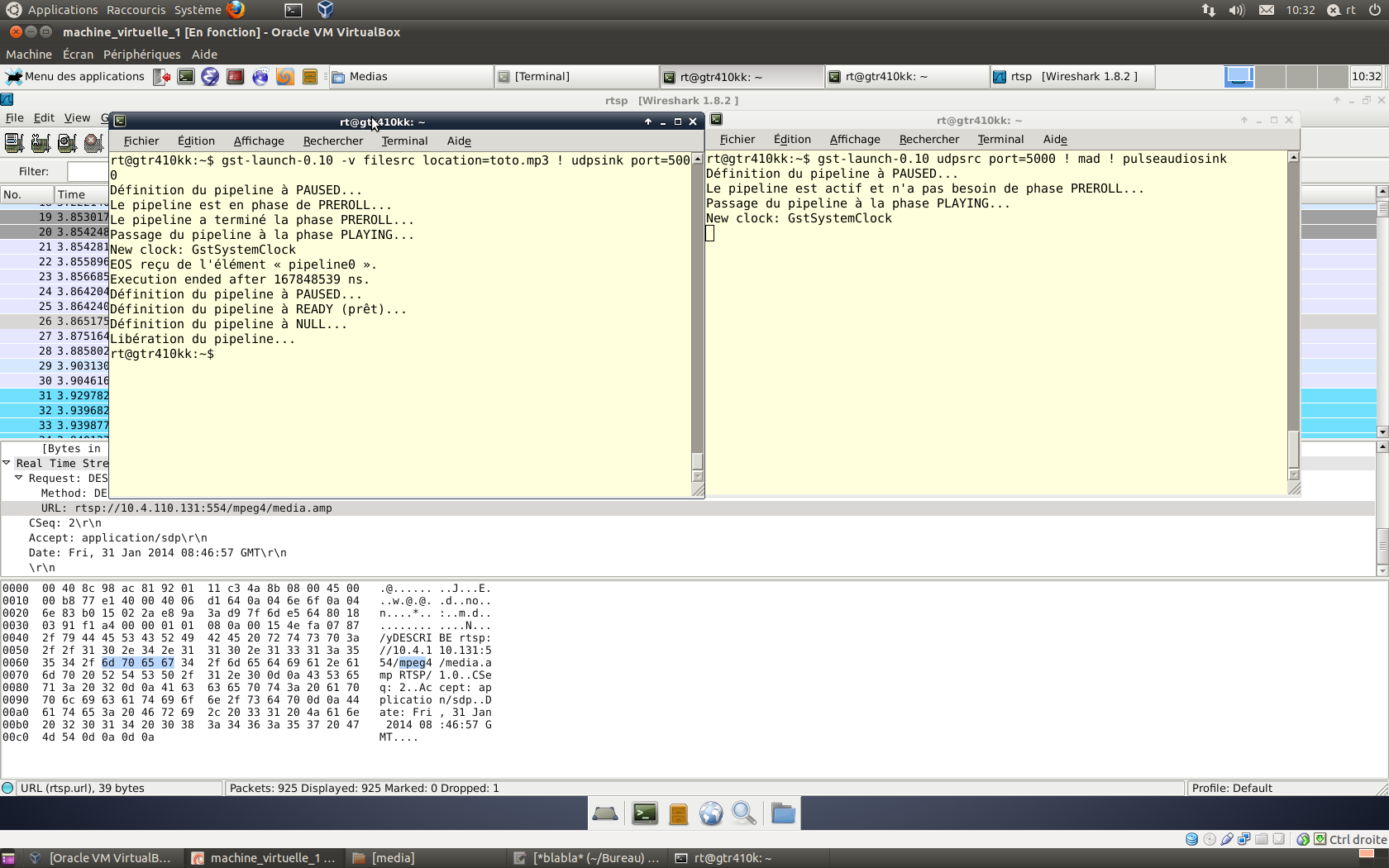
## Fichier

### Lecture

La lecture du son faite avec *gstreamer* mais sans contrôle du flux (il lit les informations au fur et à mesure qu’elles arrivent et sans se préoccuper des pertes) nous donne un son de qualité médiocre.

### Envoi UDP

Lors de l’envoie du son à un autre ordinateur, il faut faire attention à lancer la réception avant l’envoi. A l’inverse, le son n’a rien pour être reçu et n’est donc pas entendu.  
Par ailleurs, le flux reçu n’étant toujours pas décodé par UDP/RTP, on constate toujours une mauvaise qualité audio.



### Envoi RTP/UDP

Dans ce cas, nous avons ajouté une entête RTP à l’envoie du flux vidéo ainsi qu’à la réception. Ce qui signifie que les données audio ne sont pas lues au fur et à mesure de leur arrivée mais bien dans leur ordre de lecture normale. Nous avons constaté que le son entendu à la réception était de bien meilleur qualité bien qu’il y ait une phénomène de gigue (« trous » dans le son) perceptible.

# Vidéo

En combinant toutes les commandes vues précédemment, nous avons lu la vidéo *big\_bunny\_720p\_h264.mov* tout en ayant le son correspondant.

