

Transmission de données multimédia

TP3 FLUX VIDEO ET AUDIO

Guéron Marie / Favreau Jean-Dominique / Tanguy Arnaud

Table des matières

Initialisation	2
1. La salle 310.....	2
2. Logiciels.....	2
Flux d'image	3
1. Caméra.....	3
Avec un navigateur	3
Avec gstreamer.....	4
Décomposition du flux	4
Ce flux en réseau	5
2. Un autre streamin.....	6
Avec decodebin	6
On décompose plus finement	6
Sauvegarde directe.....	6
Sauvegarde et visualisation	7
3. Fichier	8
Sons	9
1. Fichier	9
Lecture.....	9
Envoi UDP.....	9
Envoi RTP/UDP	9
Vidéo	10

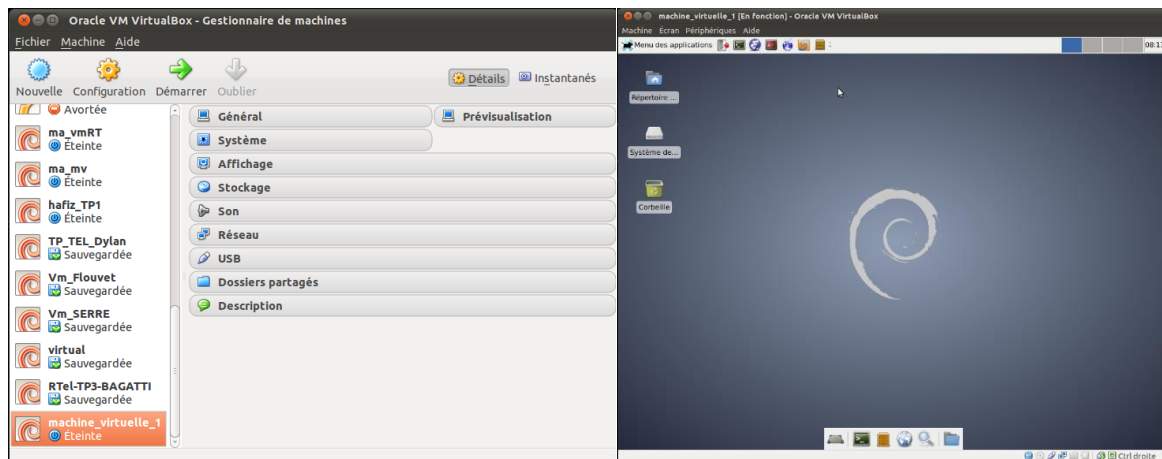
Initialisation

1. La salle 310

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

[illegible]

Puis nous lançons la machine virtuelle créée précédemment.



2. Logiciels

Sur la machine virtuelle fraîchement créée, 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*, *qstreamer*,...

```

# Root: /bin Affichage Recherche Terminal Aog
# Titre : editeur multimédia non linéaire utilisant GStreamer
# Description : serveur de son Politecadio
# Auteur : glib 2.0
# Licence : licence publique générale multi-média : bibliothèque Python
# Développement : Application GNOME pour convertir des fichiers audio d'un format
# 1 à 8 bits
# son2-jaisir - extracteur de CD audio de GNOME
# sublimescape - éditeur de sous-titres pour GNOME
# tottem - simple lecteur de média pour bureau GNOME basé sur GStreamer
# rcastel@rpi3:~/rpi3$ apt-get install libgstreamer-plugins-bad libgstreamer-plugins-ugly
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet libgstreamer-plugins-bad
rcastel@rpi3:~/rpi3$ apt-get install libgstreamer-plugins-bad libgstreamer-plugins-ugly
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet libgstreamer-plugins-bad
E: Impossible de trouver le paquet libgstreamer-plugins-ugly
rcastel@rpi3:~/rpi3$ apt-get install gstreamer-1.0-plugins-good
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
gstreamer-1.0-plugins-good est déjà la plus récente version disponible.
gstreamer-1.0-plugins-good peut être installé manuellement.
E: 0 mis à jour, 0 nouvellement installés, 0 à enlever et 126 non mis à jour.
rcastel@rpi3:~/rpi3$ apt-get install gstreamer-1.0-plugins-bad
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
gstreamer-1.0-plugins-bad est déjà la plus récente version disponible.
gstreamer-1.0-plugins-bad peut être installé manuellement.
E: 0 mis à jour, 0 nouvellement installés, 0 à enlever et 126 non mis à jour.
rcastel@rpi3:~/rpi3$
rcastel@rpi3:~/rpi3$ apt-get install gstreamer-1.0-plugins-bad
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
gstreamer-1.0-plugins-bad est déjà la plus récente version disponible.
gstreamer-1.0-plugins-bad peut être installé manuellement.
E: 0 mis à jour, 0 nouvellement installés, 0 à enlever et 126 non mis à jour.
rcastel@rpi3:~/rpi3$

```

Flux d'image

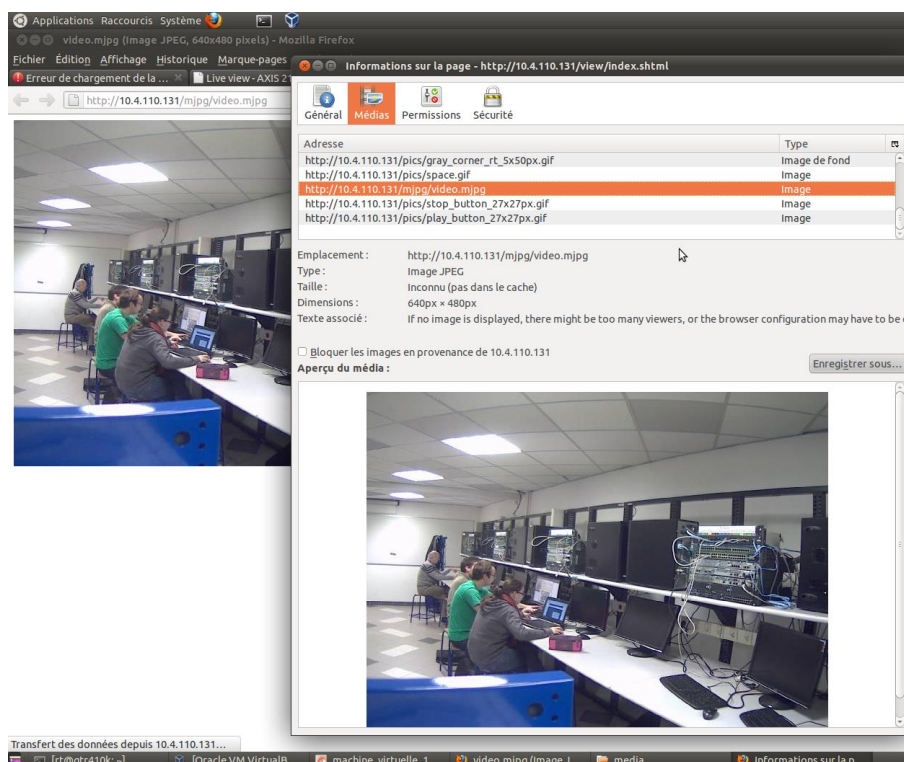
1. Caméra

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

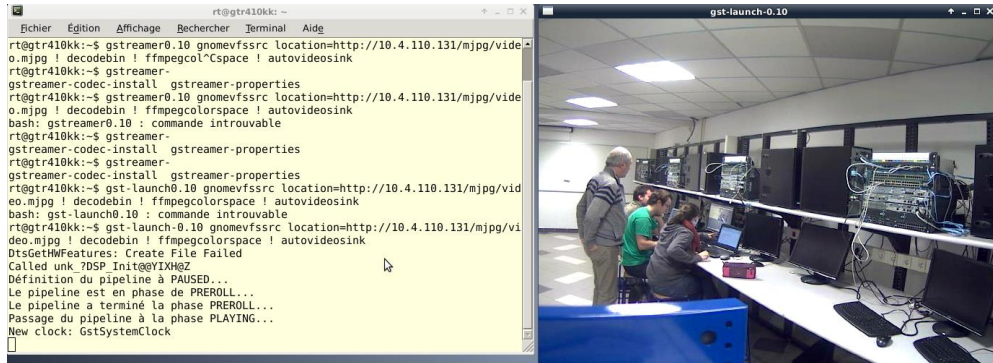
Notre première méthode a été de récupérer les images envoyées par la caméra grâce un navigateur web. Pour cela, nous pouvons accéder à deux adresses différentes et récupérer deux flux de type différents.

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



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 agencant les traitements les uns après les autres.



Décomposition du flux

Lorsque l'on décompose la commande *gstreamer*, on se rend compte que le(s) plugin(s) utilisé(s) est (sont) *jpegdec0*.

ligne de commande spécifique :

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.

```
rt@gtr410kk:~$ gst-launch-0.10 -v gnomevfsrc location=http://10.4.110.131/mjpg/video.o.mjpg ! decodebin ! ffmpegcol^Cspace ! autovideosink
Définition du pipeline à PAUSED...
Le pipeline est en phase de PREROLL...
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstTypeFindElement:typefind.GstPad:src: caps = multipart/x-mixed-replace
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstMultipartDemux:multipartdemux0.GstPad:sink: caps = multipart/x-mixed-replace
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstQueue:queue0.GstPad:sink: caps = image/jpeg
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstQueue:queue0.GstPad:src: caps = image/jpeg
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:sink: caps = image/jpeg
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-yuv, format=(fourcc)I420, width=(int)640, height=(int)480, framerate=(fraction)0/1
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-yuv, format=(fourcc)I420, width=(int)640, height=(int)480, framerate=(fraction)0/1
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-rgb, bpp=(int)32, depth=(int)24, endianness=(int)4321, red_mask=(int)16711680, blue_mask=(int)16777216, width=(int)640, height=(int)480, framerate=(fraction)0/1, pixel-aspect-ratio=(fraction)1/1
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-yuv, format=(fourcc)I420, width=(int)640, height=(int)480, framerate=(fraction)0/1
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-yuv, format=(fourcc)I420, width=(int)640, height=(int)480, framerate=(fraction)0/1
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-rgb, bpp=(int)32, depth=(int)24, endianness=(int)4321, red_mask=(int)16711680, blue_mask=(int)16777216, width=(int)640, height=(int)480, framerate=(fraction)0/1, pixel-aspect-ratio=(fraction)1/1
/GstPipeline:pipeline0/GstDecodeBin:decodebin0/GstJpegDec:jpegdec0.GstPad:src: caps = video/x-raw-rgb, bpp=(int)32, depth=(int)24, endianness=(int)4321, red_mask=(int)16711680, blue_mask=(int)16777216, width=(int)640, height=(int)480, framerate=(fraction)0/1, pixel-aspect-ratio=(fraction)1/1
Le pipeline a terminé la phase PREROLL...
Passage du pipeline à la phase PLAYING...
New clock: GstSystemClock
```

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.

The screenshot shows a Wireshark capture of an HTTP GET request. The packet list on the left shows a series of TCP segments from 10.4.110.131 to 10.4.110.131. The selected packet is a GET request for /mjpg/video.mjpg. The packet details pane shows the following information:

- Frame 19: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits) on interface 0
- Ethernet II, Src: 92:01:11:c3:4a:8b (92:01:11:c3:4a:8b), Dst: AxisComm 98:ac:81 (08:40:8c:98:ac:81)
- Internet Protocol Version 4, Src: 10.4.110.111 (10.4.110.111), Dst: 10.4.110.131 (10.4.110.131)
- Transmission Control Protocol, Src Port: 37490 (37490), Dst Port: http (80), Seq: 1, Ack: 1, Len: 190
- Hypertext Transfer Protocol
 - GET /mjpg/video.mjpg HTTP/1.1\r\n
 - Host: 10.4.110.131\r\n
 - User-Agent: gnome-vfs/2.24.4 neon/0.25.4\r\n
 - Keep-Alive: \r\n
 - Connection: TE, Keep-Alive\r\n
 - TE: trailers\r\n
 - Range: bytes=0-\r\n
 - Accept-Ranges: bytes\r\n
 - \r\n

The packet bytes pane shows the raw data of the request, including the GET line and headers.

The screenshot shows a Wireshark capture of a TCP segment. The packet list on the left shows a series of TCP segments from 10.4.110.131 to 10.4.110.131. The selected packet is a TCP segment from 10.4.110.131 to 10.4.110.131. The packet details pane shows the following information:

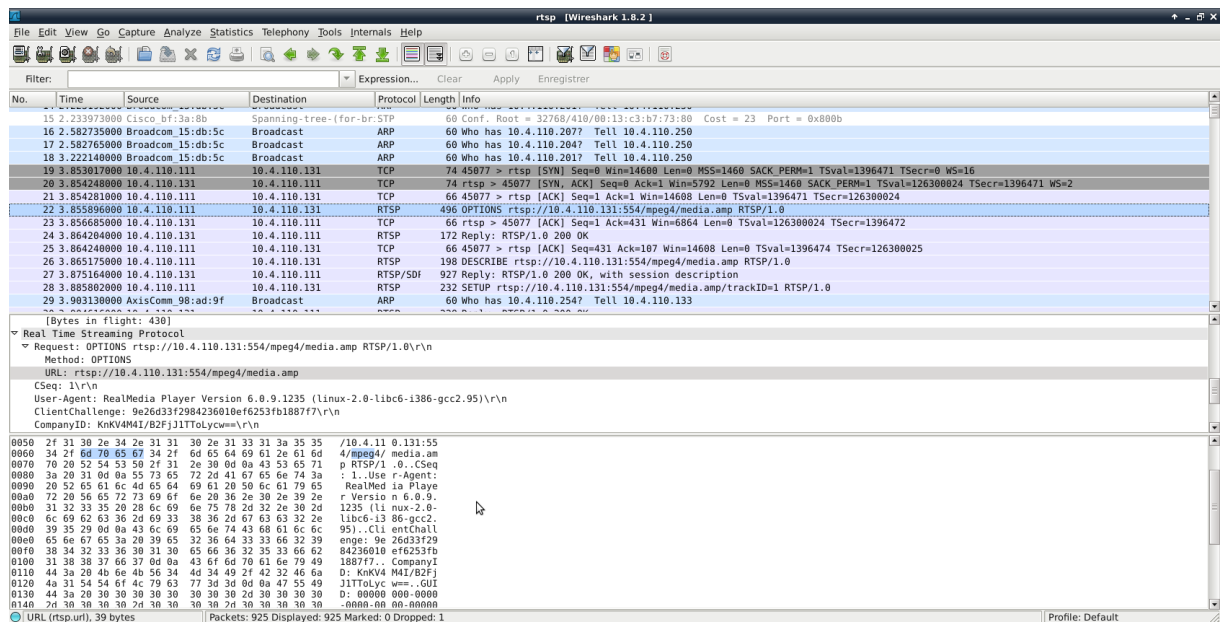
- Flags: 0x010 (ACK)
- Window size value: 3432
- [Calculated window size: 6864]
- [Window size scaling factor: 2]
- Checksum: 0x7849 [validation disabled]
- Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
- SEQ/ACK analysis
 - [Bytes in flight: 4344]
- TCP segment data (1448 bytes)

The packet bytes pane shows the raw data of the TCP segment, including the flags, window size, and sequence number.

2. Un autre streaming

Avec decodebin

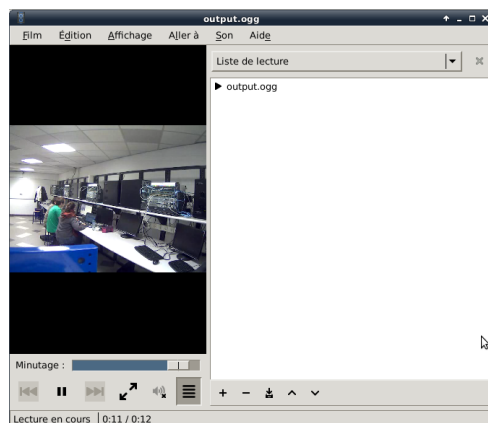
Nous avons lancé la lecture de la vidéo avec *gstreamer* et avec le codec(?) decodebin, puis nous avons analysé les flux envoyés grâce à *wireshark*.



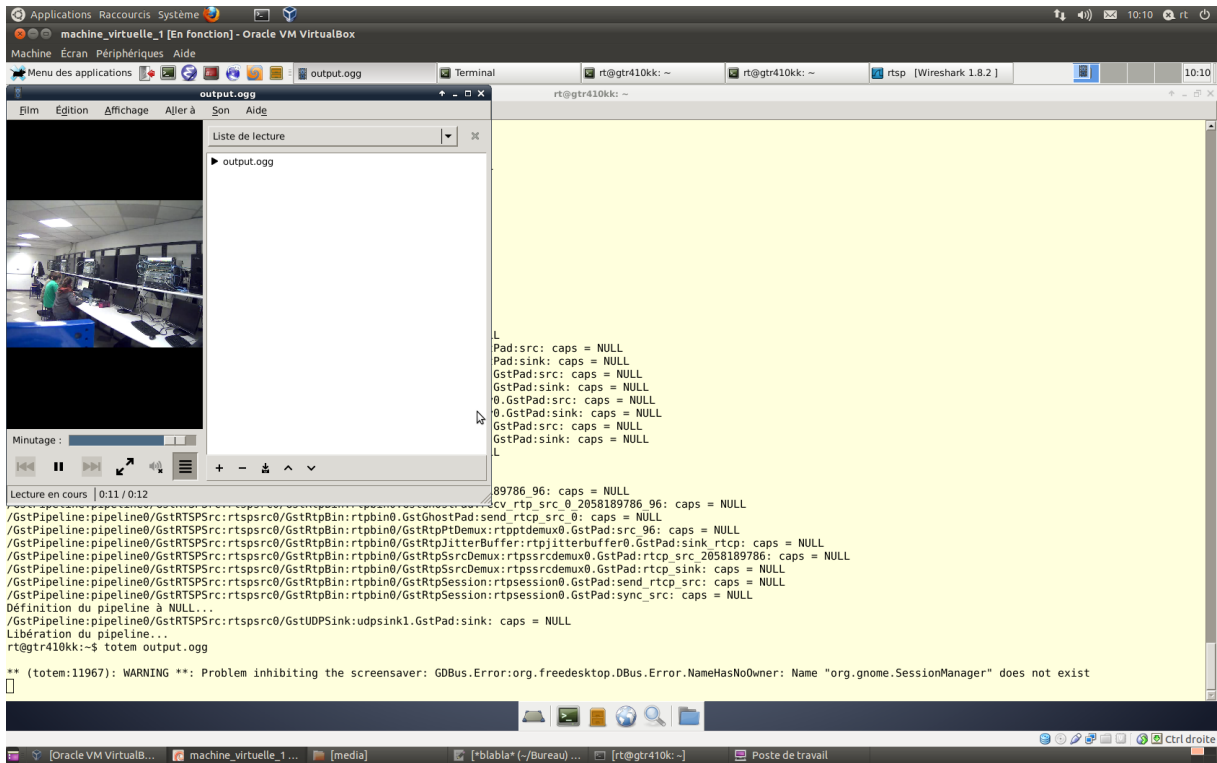
Nous pouvons par ailleurs constater que le transport utilisé ici est le protocole RTP.

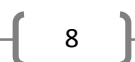
On décompose plus finement

Sauvegarde directe



Sauvegarde et visualisation





Sons

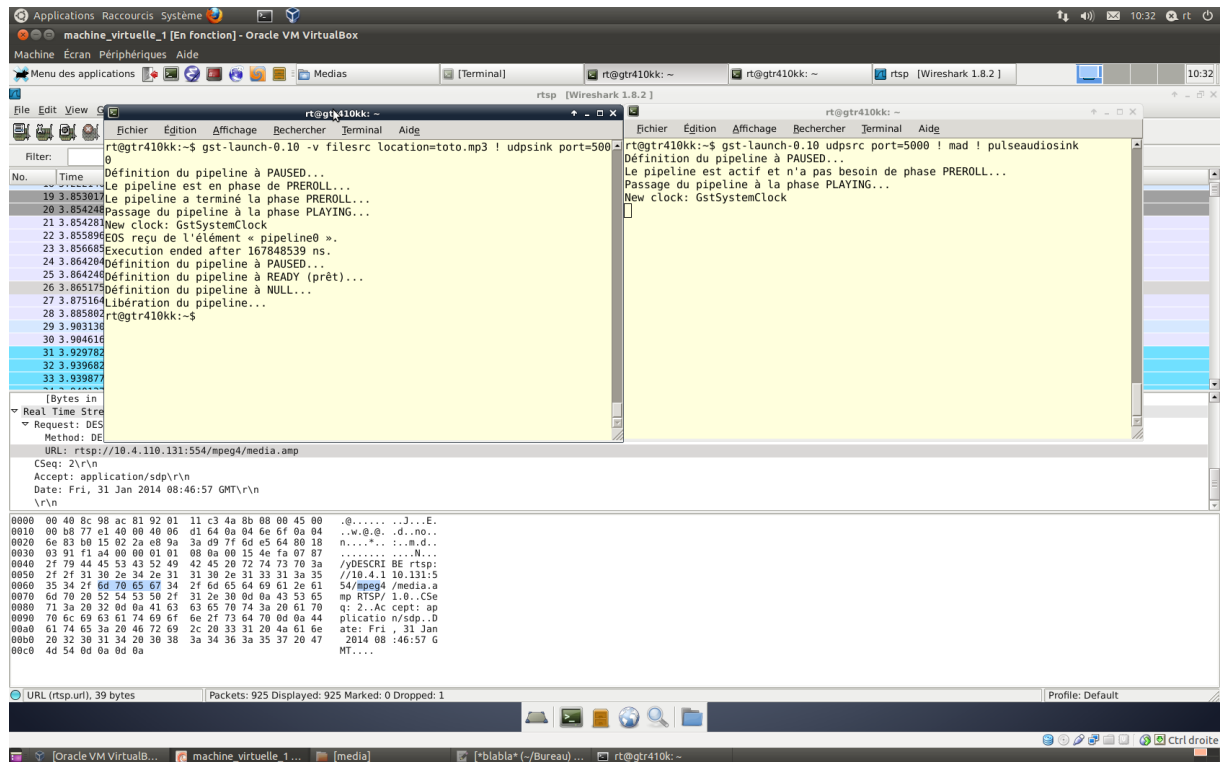
1. Fichier

Lecture

Envoi UDP

Attention, il faut lancer la réception avant l'envoi (à l'inverse, le son n'a rien pour être reçu et n'est donc pas entendu)

Mauvaise qualité audio



Envoi RTP/UDP

Vidéo

