Documentation technique

Windows/Linux

Version 0.6

VLVC est un projet de fin d'études réalisé à EPITECH
http://www.vlvc.net
http://www.epitech.net

Date de publication: Décembre 2006

Sommaire

1 Présentation

1.1 La vidéoconférence

La vidéoconférence est un mode de communication qui utilise une camera et un microphone pour transmettre l'image et le son à chacun des participants.

Elle permet à un groupe de plusieurs personnes situées à différents endroits géographiques d'avoir une conversation mêlant l'audio et la vidéo.

1.2 VLC

VLC est un lecteur multimédia, qui intègre déjà toute une panoplie de différents codecs, il gère également très bien le streaming multimédia, l'acquisition de périphériques…
Le tout sur une multitude de plateformes.
Vous pouvez vous renseigner sur http://videolan.org

1.3 Pourquoi ce projet ?

La vidéoconférence est actuellement en plein essor avec des logiciels comme MSN Messenger, Skype et iChat.
Toutefois, il n'existe pas, à l'heure actuelle, de logiciel qui soit a la fois gratuit, open source et multi plateformes.

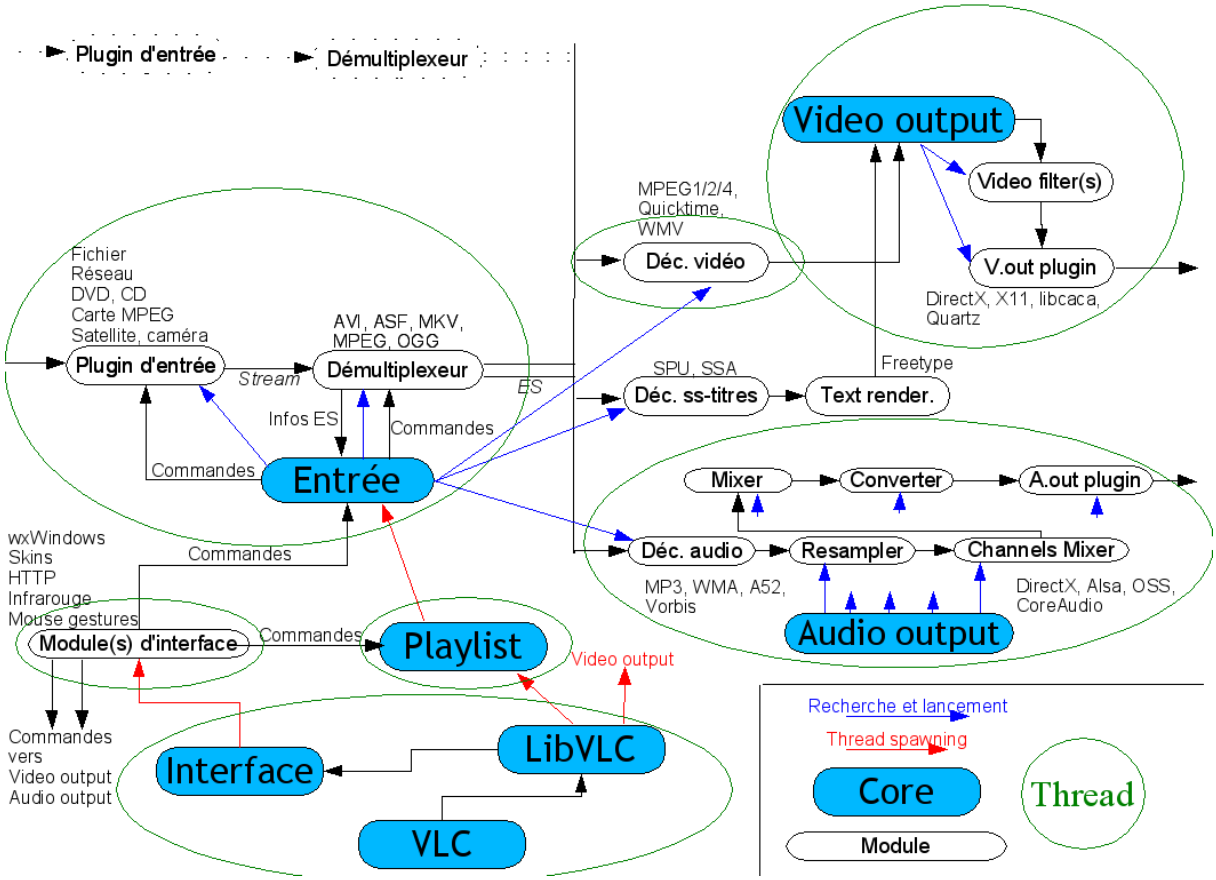
Nous avons choisi de nous intégrer à VLC pour des raisons de facilité. Si l'on avait fait une application à part entière, nous aurions eu à gérer toutes ces parties, chose irréalisable sur la période du PFE.

La majeure partie de notre travail a donc été de comprendre le fonctionnement de VLC, et apprendre à l'utiliser pour nos besoins, en organisant les flux de telle sorte qu'ils fonctionnent pour pouvoir utiliser VLC comme un outil de vidéoconférence.

2 Modularité

VLC est extrêmement modulaire.

Voici un schéma de fonctionnement global :



Source : videolan.org

Comme nous le voyons ici, VLC est simplement un gestionnaire de modules. Il charge des modules, qui s'occupent de faire les traitements.

Il est intéressant de noter qu'il y a une hiérarchie entre les modules, ici sur le schéma les modules en bleu sont les modules cœur (principaux), d'autres modules, en blanc, les utilisent pour leur fonctionnement.

2.1 Nos modules

VLVC est donc un module de VLC, mais avec certaines limitations. Il n'est pas un module au sens où l'on ne peut pas simplement rajouter la bibliothèque VLVC dans un répertoire de VLC, et VLC dispose de la vidéoconférence.

VLVC a une partie intégrée au cœur de VLC, et a nécessité de très légères modifications dans VLC, spécifiquement dans la partie graphique.

Il est donc un module au sens où il s'ajoute à un logiciel existant pour lui ajouter une fonctionnalité.

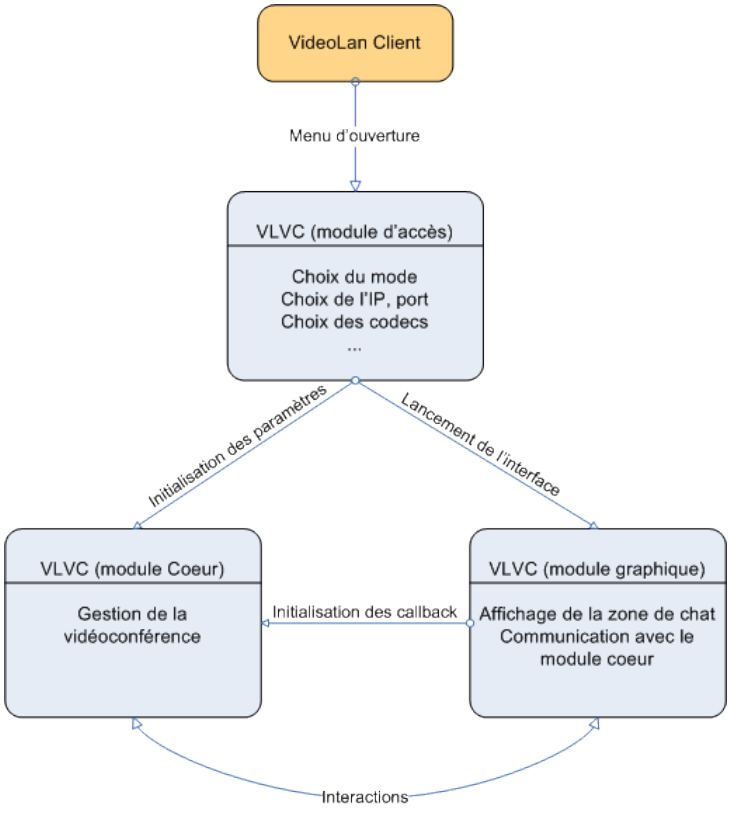
Plus techniquement, VLVC se compose en 3 modules :

- Module cœur : pièce maîtresse de VLVC, qui gère la vidéoconférence, les clients, les flux audio/vidéo à démarrer ou arrêter, les messages…

- Module d'accès : partie, correspondant plus à un hack. Permet à VLVC d'être dans les onglets du menu d'ouverture de VLC. Plus de détails sont donnés dans la partie détails techniques.

- Module graphique : composant graphique du projet. Permet l'affichage de la fenêtre de gestion de la conférence / chat. N'effectue aucun traitement, permet simplement de faire le lien entre l'utilisateur et le module cœur. Prévu de manière générique afin que le module cœur n'ait pas à être dépendant de l'interface pour fonctionner. Cela permet d'avoir le même module cœur sur toutes les architectures, seule l'interface graphique étant à refaire (et éventuellement le module d'accès à repenser).

Schéma de l'organisation des modules de VLVC au sein de VLC.



2.2 VideoLan Manager

VLM est le module qui gère tous les modules de VLC. C'est lui que nous utilisons pour faire fonctionner la vidéoconférence.

Il s'utilise via des commandes passées sous forme de chaînes de caractères.
Pour plus de détails, le mieux est de consulter la documentation de VLC disponible à l'adresse http://www.videolan.org/doc/

Voici un exemple utilisant des commandes VLM suivi de leur explication :

Pour acquérir une Webcam avec DirectShow, la transmettre vers 192.168.0.1, port 2222, plus un affichage local, les commandes VLM suivantes sont exécutées :

1. new vlvcClient01 broadcast enabled
2. setup vlvcClient01 input dshow://
3. setup vlvcClient01 option dshow-size=320x240
4. setup vlvcClient01 output #transcode{vcodec=DIV3,vb=512,scale=1,acodec=mpga,ab=96,channels=2}:duplicate{dst=std{access=rtp,mux=ts,dst=192.168.0.1:2222},dst=display }
5. setup vlvcClient01 play

Détails :

1. Création du contrôle nommé vlvcClient01, de type standard pour lire/envoyer un flux
2. On indique qu'on lit le flux d'entrée en local sur la webcam en utilisant DirectShow
3. Exemple d'option pouvant être ajoutée au contrôle, ici la taille de capture
4. Paramétrage de la sortie, avec les paramètres de compression et de sortie
5. Une fois le contrôle paramétré, on lance la lecture.

3 Technologies utilisées

3.1 Vidéo/audio

VLC supporte une multitude de différents codecs audio et vidéo :

Vidéo	**Audio**
MPEG-1 vidéo	MPEG Layer 1/2/3 audio
MPEG-2 vidéo	AC3 (i.e. A52)
MPEG-4 vidéo	MPEG-4 audio (i.e. AAC)
DivX 1/2/3 vidéo	Vorbis/Speex
WMV ½	FLAC
H/I 263	
MJPEG	
Theora	
H.264/MPEG-4 AVC	

Pour la vidéo, nous avons choisi d'utiliser le DivX 3 par défaut car il s'est avéré être le plus stable et le plus intéressant en termes de rapport qualité / bande passante.

Pour l'audio, nous avons choisi d'utiliser le mpeg audio ou mpga.

Note :
Il est possible de choisir n'importe quel codec supporté par VLC pour la vidéoconférence.

3.2 Réseau

VLVC utilise différents moyens de communication réseau.

Les flux audio et vidéo sont gérés par VLM, aucune gestion directe n'est faite dans nos modules. L'utilisateur peut choisir pour le protocole de transport entre UDP et RTP.

La communication entre le serveur et les clients se fait via un socket TCP. C'est cette gestion qui va être détaillée paragraphe 4.1 Protocole réseau.

L'intégralité de la gestion réseau de VLVC se passe dans le module cœur, les modules graphique et d'accès n'ont strictement aucune action réseau.

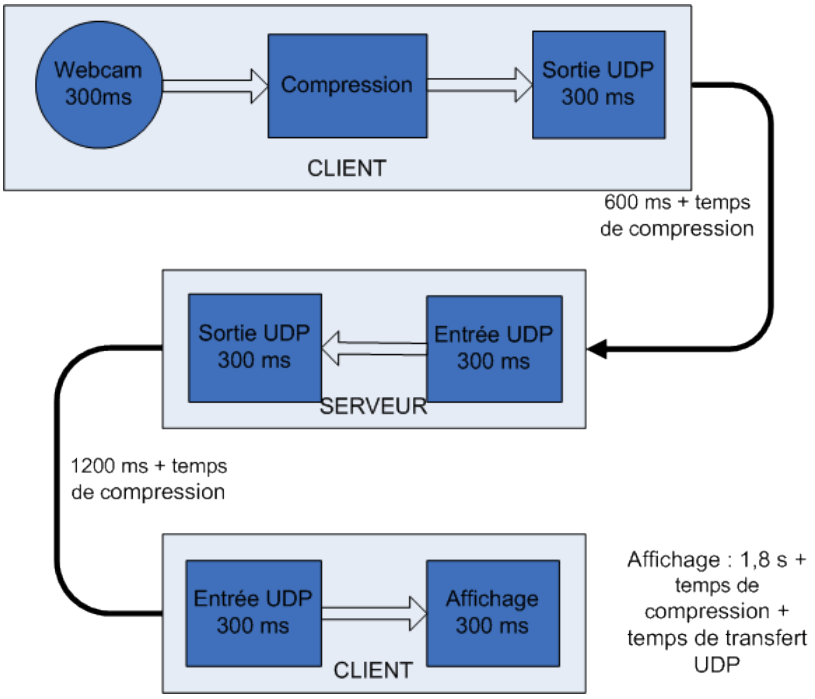
Pour le mode client, un seul socket est ouvert, il n'y a donc qu'une boucle de gestion de la réception de messages provenant du serveur.

Pour le serveur une gestion plus poussée est nécessaire.
Actuellement, nous utilisons un select pour vérifier tous les sockets ouverts, celui d'écoute de l'arrivée de nouveaux clients, et ceux de tous les clients actuellement connectés.

La gestion du select ne s'effectue que sur la lecture sur les sockets, pas l'écriture. Lorsque select() prévient de l'arrivée de données sur un socket, une action est déclenchée.

Dans une précédente version, le serveur disposait simplement d'une boucle d'accept, et lors de l'arrivée d'un nouveau client, un thread spécifique était créé pour s'en occuper. Cette méthode fonctionnait correctement, mais des comportements imprévus survenaient parfois. N'ayant pas pu déterminer la provenance exacte de ces comportements exceptionnels, nos doutes se sont portés sur le contexte multi threadé, les accès concurrentiels qu'il entraîne, etc.

Le choix de select() a donc naturellement était fait, permettant de n'avoir plus qu'un seul thread.



Ce schéma explique les différentes étapes du processus de streaming et l'apparition de latence durant la transmission des flux.

3.2.1 Transmission des flux multimédia

Trois possibilités sont disponibles sur un réseau TCP/IP pour transmettre des flux multimédia.

3.2.1.1 L'unicast

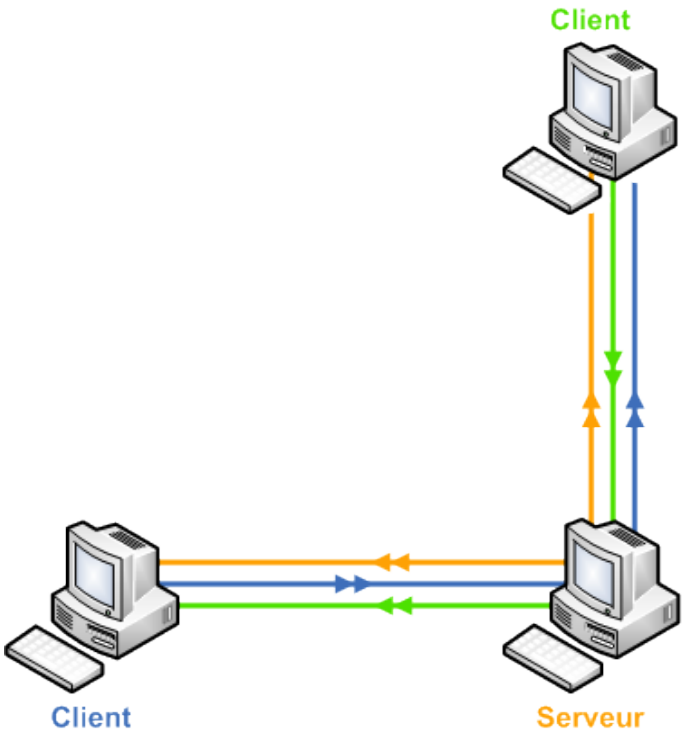
Tous les clients envoient leurs flux au serveur, qui les retransmet à son tour à tous les participants.

Avantages :

- Seul le serveur a besoin d'une bande passante importante.
- Simplification de la gestion des flux des clients côté serveur (play/stop sur les flux).

Inconvénient :

- Implique un temps de latence supplémentaire.



3.2.1.2 Le Peer to Peer

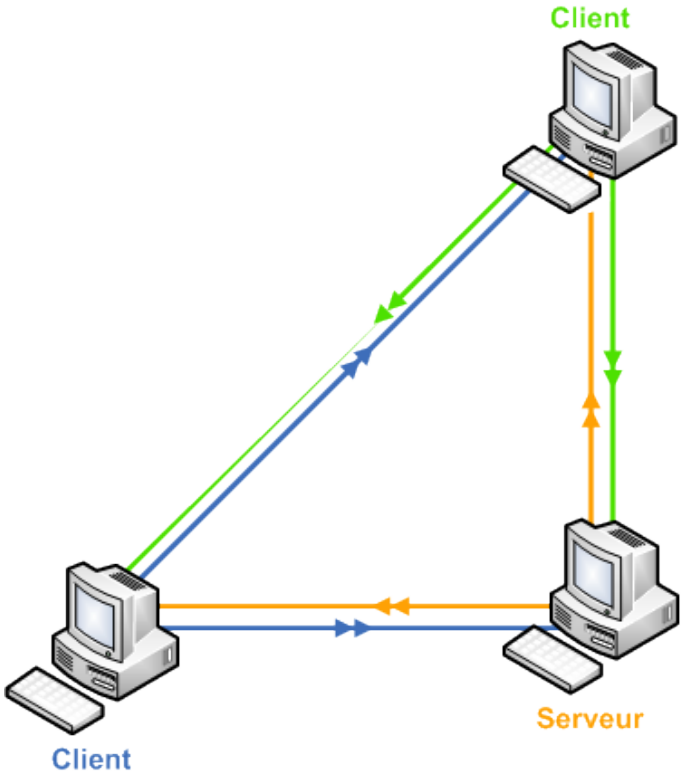
Chaque client envoie ses flux à tous les autres participants.

Avantage :

- Envoi direct, pas d'introduction de latente supplémentaire.

Inconvénients :

- Tous les clients ont besoin d'une bande passante importante.
- Nécessite une gestion des flux clients plus poussée (play/stop sur les flux).



3.2.1.3 Le multicast

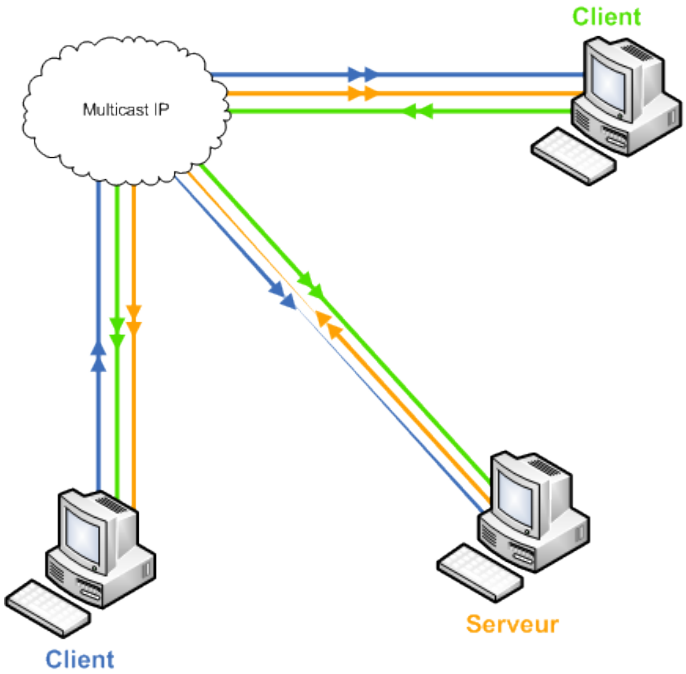
Utilisation d'une adresse réseau spécifique pour la transmission des flux.

Avantages :

- Envoi direct, pas d'introduction de latence supplémentaire.
- La bande passante nécessaire est peu importante.

Inconvénients :

- Fonctionne uniquement en LAN.



3.2.1.4 Notre choix

Nous avons choisi d'utiliser le mode unicast. De ce fait, une seule des machines participant à la conférence (le serveur) a besoin d'avoir une importante bande passante. Les clients ont par contre besoin d'un minimum de ressources.
Bien que cette solution entraîne une latence supplémentaire, elle demeure la plus pratique.

4 Normes

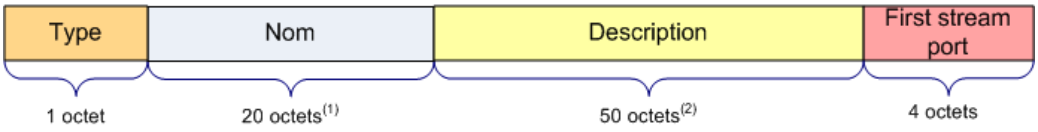
4.1 Protocole réseau

VLVC implémente son propre protocole de gestion des clients. Le protocole est basique, et se compose en 2 parties : le type de message, suivi des données spécifiques au message (optionnelles).

Voici la liste de ces messages ainsi que leur description :

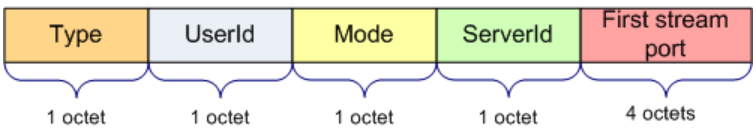
4.1.1 Hello

C'est une commande envoyée par le client lors de sa connexion au serveur.
Type : VLVC_NET_COMMAND_HELLO
Nom[1] : Nom du client qui se connecte
Description[2] : Description du client qui se connecte
First stream port : Premier port sur lequel le serveur doit envoyer les flux des clients. Le port d'un client correspond à ce port de base plus l'identifiant du client.



4.1.2 Welcome

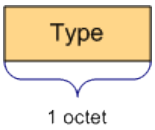
C'est une commande envoyée par le serveur a un client qui vient d'envoyer un HELLO pour lui signifier qu'il est accepté sur le serveur.
Type : VLVC_NET_COMMAND_WELCOME
UserId : Numéro identifiant l'utilisateur sur le serveur
Mode : mode de conférence du serveur
ServerId : Numéro identifiant de l'utilisateur ayant lancé le serveur
First stream port : Premier port utilisé par le serveur pour recevoir les flux des clients. Le port correspondant à un client correspond à ce port de base plus son identifiant.



[1] Taille définie par la macro VLVC_MAX_USER_NAME_LENGTH (voir include/vlc_vlvc.h)
[2] Taille définie par la macro VLVC_MAX_USER_DESC_LENGTH (voir include/vlc_vlvc.h)

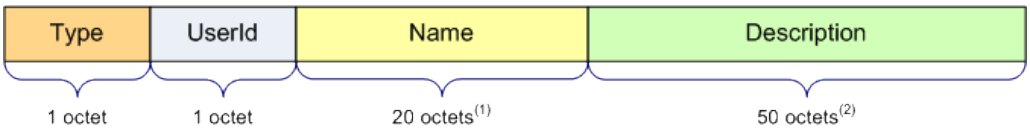
4.1.3 Not welcome

C'est une commande envoyée par le serveur a un client qui vient d'envoyer un HELLO pour lui signifier qu'il est refusé sur le serveur.
Type : VLVC_NET_COMMAND_ NOT_WELCOME



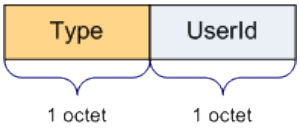
4.1.4 Add user

C'est une commande envoyée par le serveur lorsqu'un utilisateur a rejoint la conférence.
Type : VLVC_NET_COMMAND_ADD_USER
UserId : Numéro identifiant de l'utilisateur qui vient de rejoindre la conférence
Name[1]: Nom de l'utilisateur qui vient de rejoindre la conférence
Description[2] : Description de l'utilisateur qui vient de rejoindre la conférence



4.1.5 Leave user

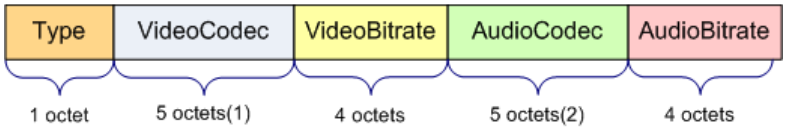
C'est une commande envoyée par le serveur lorsqu'un utilisateur a quitte la conférence.
Type : VLVC_NET_COMMAND_LEAVE_USER
UserId : Numéro identifiant de l'utilisateur qui vient de quitter la conférence



[1] Taille définie par la macro VLVC_MAX_USER_NAME_LENGTH (voir include/vlc_vlvc.h)
[2] Taille définie par la macro VLVC_MAX_USER_DESC_LENGTH (voir include/vlc_vlvc.h)

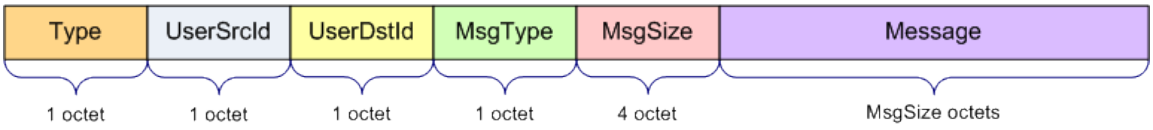
4.1.6 Codecs

Le serveur dispose d'une option lui permettant de forcer les codecs utilisés par les clients.
Type : VLVC_NET_COMMAND_CODECS
VideoCodec[1] : nom du codec vidéo à utiliser
VideoBitrate : bitrate à appliquer au codec vidéo
AudioCodec[2] : nom du codec audio à utiliser
AudioBitrate : bitrate à appliquer au codec audio



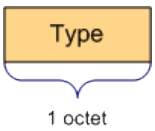
4.1.7 Message

C'est une commande envoyée aussi bien par le serveur que par le client.
Type : VLVC_NET_COMMAND_MESSAGE
UserSrcId : Numéro identifiant de l'utilisateur qui envoie le message.
UserDestId : Numéro identifiant de l'utilisateur qui doit recevoir le message.
MsgType : Type de message.
MsgSize : Taille du message a la suite.
Message : Contenu texte du message.



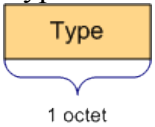
4.1.8 Ping

Cette une commande n'est pas utilisée pour le moment.
Type : VLVC_NET_COMMAND_PING



4.1.9 Quit

C'est une commande envoyée par le client lorsqu'il quitte le serveur.
Type : VLVC_NET_COMMAND_QUIT



[1] Taille définie par la macro VLVC_MAX_VIDEO_CODEC_LENGTH (voir include/vlc_vlvc.h)
[2] Taille définie par la macro VLVC_MAX_AUDIO_CODEC_LENGTH (voir include/vlc_vlvc.h)

4.2 Code

La norme d'écriture est soumise à celle de VLC qui est disponible à l'adresse : http://developers.videolan.org/vlc/vlc/doc/developer/html/ch01.html#id288821.
De plus, un soin tout particulier a été apporte à la longueur ainsi qu'à la clarté des fonctions. Il est donc nécessaire de conserver cette norme interne.

4.3 Commentaires

Une norme particulière interne a été ajoutée au niveau des commentaires, ce qui permet de générer des fichiers html grâce a un script Perl. Voici un exemple de cette norme :

```
/*****************************************************************************
 * Name: NetWrite
 *
 * Parameters:
 *          vlvc_t *p_vlvc: a pointer to the vlvc object
 *          int i_fd: the fd to whom we want to send data
 *          void *p_content: data to be sent
 *          int i_size: size of the data to be sent
 *
 * Returns:
 *          int: number of bytes written
 *
 * Description:
 *          Wrapper for net_Write (sends data on the network)
 *****************************************************************************/
```

Le seul champs obligatoire est "Name :" et doit être suivi du nom de la fonction.
Ensuite le nom des sections est détecté automatiquement en prenant le texte avant les ':'. Le contenu de chaque section correspond au texte entre deux sections.

Les sections obligatoire sont : Parameters, Returns, Description. Il est aussi possible d'ajouter des sections comme Comments. Ces fichiers sont fournis en annexe. Le script Perl utilisé pour les générer est disponible sur le subversion.

5 Communication

Voici le « déroulement technique » lorsque l'on utilise VLVC :
Sélection du menu d'ouverture par l'utilisateur : la partie déclaration des variables du module d'accès est utilisée par VLC pour afficher tous les champs nécessaires.

L'utilisateur appuie sur le bouton OK une fois tous les paramètres choisis :
Dans l'événement OnOK()[1] de la fenêtre, l'interface de VLVC est lancée. Du code a du être ajouté à cet endroit pour pouvoir le faire. En effet, le module cœur et le module d'accès sont en C, ils ne peuvent donc pas instancier un objet C++.

Une fois lancée, l'interface graphique récupère une instance du module cœur, et attend qu'il soit initialisé par le module d'accès.

VLC appelle la méthode Open()[2] du module d'accès, où les paramètres sont récupérés, et le module cœur initialisé.
Une fois le module cœur initialisé par le module d'accès, le module graphique renseigne les callbacks dans le module cœur[3].

Une fois correctement initialisé, le module cœur lance un thread correspondant au mode choisi (client ou serveur), et la vidéoconférence est lancée.

Il est nécessaire de lancer un thread pour ne pas bloquer l'appelant de vlvc_Start()[4]

5.1 Communication entre les modules

Pour communiquer, le module cœur et le module graphique utilisent 2 méthodes différentes :
Le module cœur peut être appelé grâce à des fonctions « exportées » globalement.

VLC dispose d'une macro VLC_EXPORT qui permet aux fonctions exportées par ce moyen d'être visibles et utilisables absolument partout. Pour utiliser le module cœur, le module graphique passe donc par ces fonctions exportées.

Pour la communication du module cœur au module graphique, le module graphique initialise des callbacks dans le module cœur qui peut alors communiquer avec lui. Ceci est utilisé pour gérer des événements comme la connexion / déconnexion des utilisateurs ainsi que les messages à afficher.

[1] Fichier modules/gui/wxwidgets/dialogs/open.cpp
[2] Fichier modules/access/vlvc/vlvc.c
[3] Voir modules/gui/wxwindows/dialogs/vlvc/vlvc_frame.cpp
[4] Fonction de lancement du module cœur, voir src/misc/vlvc.c

5.2 Evénements dans l'interface graphique

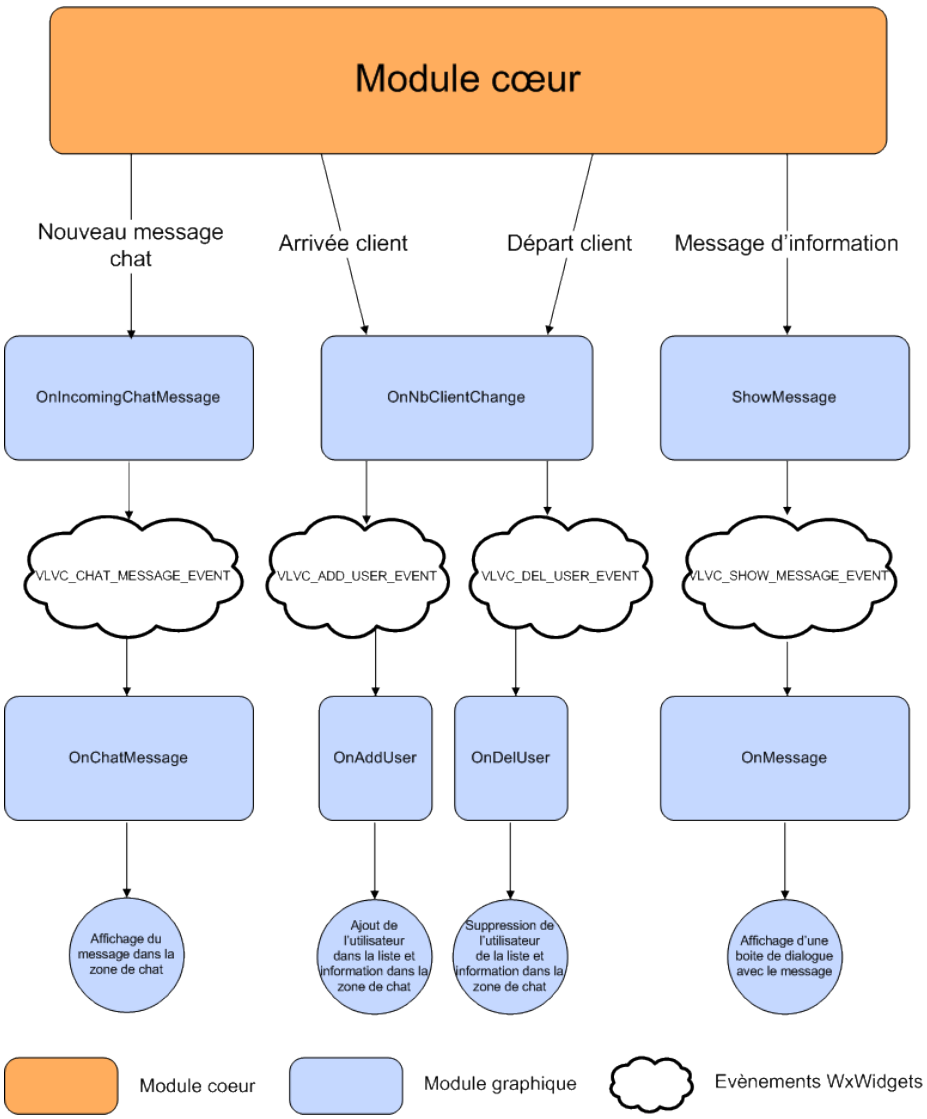
Au niveau de l'interface graphique, nous sommes soumis à une gestion événementielle dans un contexte threadé.
Il est un problème courant lorsque l'on développe une interface graphique : on ne peut pas faire n'importe quoi n'importe quand.

Par exemple, lorsque le module cœur reçoit un message chat d'un autre utilisateur, il en informe l'interface. Si l'on cherche à mettre à jour directement la zone de texte correspondant au chat, il y a de grandes chances que l'interface se bloque. Cela est dû au fait que dans un contexte événementiel, l'interface est régulièrement occupée par des messages systèmes, et faire des modifications pendant qu'elle est occupée conduit directement à un problème.

Pour tout ce qui agit sur l'interface, notamment la mise à jour des contrôles, il faut donc passer par un système d'événements.
Les callbacks appelées par le module cœur génèrent donc un événement, qui une fois appelé peut mettre à jour les contrôles sans risquer le moindre ennui.

Ces fonctions de gestion d'événements obtiennent un mutex préalablement à la modification pour éviter que plusieurs modifications surviennent en même temps.

Voici un schéma explicatif des évènements:



6 Compilation

Pour Windows et Linux, un script de compilation buildAll-[linux|win32]-[debug|release].sh est disponible sur le SVN.

6.1 Dépendances

Voici les librairies nécessaires pour compiler VLC

- automake1.9
- autoconf2.13
- cvs
- gettext
- libtool
- libmad0-dev
- ffmpeg
- libavcodec-dev
- libmpeg2-4-dev
- libwxgtk2.6-dev
- libavformat-dev
- libfreetype6-dev
- libncurses5-dev
- xlibs-static-dev
- libpostproc-dev
- liblivemedia-dev
- libsdl1.2-dev
- libdvdread-dev
- libdvbpsi4-dev
- libflac-dev
- libflac++-dev
- libid3tag0-dev
- libdvdnav-dev
- libspeex-dev
- libxml2-dev

6.2 Linux

Note : dans les lignes ci-dessous, le prompt du shell est représenté par "$>", il ne faut pas taper ces caractères avant les commandes.

Tout d'abord il faut faire un bootstrap :
Aller dans le répertoire ou se trouve VLVC puis taper :

```
$> ./bootstrap
```

Puis lancer configure :

```
$> ./configure --enable-shout --enable-v4l --enable-ncurses --enable-xosd \
        --enable-goom --enable-x11 --enable-xvideo --disable-gtk \
        --enable-sdl --enable-ffmpeg --enable-mad --enable-libdvbpsi \
        --enable-a52 --enable-dts --enable-libmpeg2 --enable-dvdnav \
        --enable-vorbis --enable-ogg --enable-theora --enable-faac \
        --enable-mkv --enable-freetype --enable-fribidi --enable-speex \
        --enable-flac --enable-caca --enable-skins --enable-skins2 \
        --enable-alsa --disable-kde --disable-qt --enable-wxwindows \
        --enable-ncurses --enable-debug --enable-vlvc
```

Si cette erreur apparait lors du configure : "liveMedia/libliveMedia.a in /usr/lib/live... no"
faire :
$> mkdir -p /usr/lib/live/liveMedia ; ln -s /usr/lib/libliveMedia.a /usr/lib/live/liveMedia

Enfin il ne reste plus qu'a compiler :

```
$> make
```

Si erreur lors du make : "file not found libpostproc/postprocess.h"
faire :
$> ln -s /usr/include/postproc /usr/include/libpostproc

si erreur : "ld: cannot find –lXext"
faire :
$> ln -s /usr/X11R6/lib/libXext.a /usr/lib/

si erreur : "ld: cannot find -lX11"
faire :
$> ln -s /usr/X11R6/lib/libX11.a /usr/lib/

6.3 Windows

Important : la version Windows est compilée depuis Linux avec des options spécifiques.

On a besoin de mingw32, donc on commence par :
$> apt-get install mingw32 mingw32-binutils mingw32-runtime tofrodos

Prendre la contrib du 28 avril 2006 :
http://download.videolan.org/pub/testing/win32/contrib-20060428-win32-bin-gcc-3.4.5-only.tar.bz2

L'extraire à la racine :
$> tar xvjf contrib-20060428-win32-bin-gcc-3.4.5-only.tar.bz2 -C /

Faire des liens symboliques :
$> ln -s /usr/lib/libraw1394.a /usr/win32/lib/
$> ln -s /usr/lib/libgsm.a /usr/win32/lib/
$> ln -s /usr/lib/libdc1394_control.a /usr/win32/lib/

Voici le script de compilation :

```
$> ./bootstrap && \
PKG_CONFIG_LIBDIR=/usr/win32/lib/pkgconfig \
CPPFLAGS="-I/usr/win32/include -I/usr/win32/include/ebml" \
LDFLAGS=-L/usr/win32/lib \
CC=i586-mingw32msvc-gcc CXX=i586-mingw32msvc-g++ \
./configure --host=i586-mingw32msvc --build=i386-linux \
    --disable-gtk \
    --enable-nls --enable-sdl --with-sdl-config-path=/usr/win32/bin \
    --enable-ffmpeg --with-ffmpeg-mp3lame --with-ffmpeg-faac \
    --with-ffmpeg-zlib --enable-faad --enable-flac --enable-theora \
    --with-wx-config-path=/usr/win32/bin \
    --with-freetype-config-path=/usr/win32/bin \
    --with-fribidi-config-path=/usr/win32/bin \
    --enable-live555 --with-live555-tree=/usr/win32/live.com \
    --enable-caca --with-caca-config-path=/usr/win32/bin \
    --with-xml2-config-path=/usr/win32/bin \
    --with-dvdnav-config-path=/usr/win32/bin \
    --disable-cddax --disable-vcdx --enable-goom \
    --enable-twolame --enable-dvdread \
    --enable-release --enable-vlvc &&\
make
```

Au moment de la compilation de libstream_out_switcher_plugin.a, si une erreur intervient faire :

```
$> cd modules/stream_out/ && \
        i586-mingw32msvc-gcc -Wsign-compare -Wall -mms-bitfields \
        -pipe -o libstream_out_switcher_plugin.dll -g -shared \
        -u _vlc_entry__0_8_5 -L/usr/win32/lib \
        libstream_out_switcher_plugin.a -L/usr/local/lib \
        -lavformat -lpostproc -lavcodec -lavutil \
        /usr/win32/lib/libvorbisenc.a /usr/win32/lib/libvorbis.a \
        /usr/win32/lib/libtheora.a /usr/win32/lib/libogg.a -ldts \
        /usr/win32/lib/liba52.a -lm -lraw1394 -ldc1394_control \
        -lgsm -lz -lmp3lame -lfaac && \
        cd ../.. && \
        make
```

pour faire un zip :
$> make package-win32-zip

6.4 MacOSX

La version MAC OS X est indisponible pour le moment.