



MOVIQS

Module pour de la vidéo sur Internet avec qualité de service

1.0

SOMMAIRE

1	INTRODUCTION	4
2	FONCTIONNALITES DE LA DLL « CR_INRIA.DLL »	5
3	ARCHITECTURE GENERALE DE LA DLL	6
3.1	TRANSPORT DE LA VIDEO SUR RTP	7
3.2	CONTROLE DE CONGESTION EN POINT A POINT (UNICAST)	7
3.2.1	<i>Contexte codage temps réel</i>	7
3.2.2	<i>Source FGS</i>	8
3.3	CONTROLE DE CONGESTION MULTIPPOINT (MULTICAST).....	8
3.4	CONTROLE DE PERTES	8
4	DESCRIPTION DES INTERFACES	9
4.1	DESCRIPTION DE L'INTERFACE C.....	9
4.1.1	<i>Création d'un canal</i>	9
4.1.2	<i>Description d'un canal</i>	9
4.1.3	<i>Description du flux de données associé au canal</i>	9
4.1.4	<i>Connexion</i>	10
4.1.5	<i>Attente de connexion</i>	10
4.1.6	<i>Emission de données</i>	10
4.1.7	<i>Attente de réception de données</i>	10
4.1.8	<i>Libération d'un canal</i>	11
4.2	DESCRIPTION DE L'INTERFACE R.....	11
4.2.1	<i>Fonction d'information sur le débit</i>	11
4.2.2	<i>Fonction d'information sur le taux de perte</i>	11
5	PARAMETRAGES DE LA DLL	12
5.1	PARAMETRES GENERAUX	12
5.2	PARAMETRES SERVEURS.....	13
5.3	PARAMETRES CLIENTS.....	14
6	EXEMPLE D'UTILISATION DES INTERFACES	15
6.1	ETABLISSEMENT D'UNE CONNEXION	15
6.2	ECHANGE DE DONNEES.....	15
6.3	ACCES AUX INFORMATIONS UTILES AU CONTROLE DE CONGESTION	16
	ANNEXE A REFERENCES	17
	ANNEXE B ABREVIATIONS	18
	ANNEXE C RECONSTRUCTION DE LA DLL “CR_INRIA.DLL”	19

LISTE DES FIGURES

Figure 1 : Modules principaux de la DLL "CR_Inria.dll"	6
Figure 2 : Exemple de fichier de configuration.....	12

1 Introduction

Ce document présente une description haut niveau des principales fonctions de MOVIQS¹, module proposé sous la forme d'une DLL « CR_Inria.dll ». Il est organisé en cinq parties principales. La première introduit les fonctionnalités offertes par la DLL. Elles s'appuient sur des modules principaux qui sont présentés dans la seconde. La DLL s'utilise à l'aide de deux interfaces et d'un fichier de configuration qui sont respectivement présentés dans les troisième et quatrième parties . Enfin, les grandes lignes de l'utilisation de la DLL sont présentées à l'aide d'un exemple dans la cinquième et dernière partie.

¹ MOVIQS: module pour de la vidéo sur Internet avec qualité de service.

2 Fonctionnalités de la DLL « CR_Inria.dll »

La DLL « CR_INRIA.dll » offre des mécanismes :

- de transport de la vidéo sur RTP/RTCP en mode unicast et multicast,
- de contrôle de congestion et de régulation de débit,
- de contrôle de pertes fondé sur l'ajout de redondance générée par des codes correcteurs d'erreur.

Elle est utilisée par les applications serveurs et clients associées. La mise en œuvre de ces mécanismes s'appuie sur deux interfaces de programmation :

- l'interface de communication ou « Interface C » utilisée par les applications serveurs et clientes pour dialoguer avec les interfaces réseau,
- l'interface de régulation ou « Interface R » utilisée par les applications serveur pour mettre en œuvre une stratégie de contrôle de congestion en fonction de l'analyse du réseau.

3 Architecture générale de la DLL

Le schéma ci-dessous présente les modules mettant en œuvre les fonctionnalités listées dans le chapitre précédent et les échanges extérieurs via les interfaces R et C respectivement nommées ici I_R et I_C.

Le fonctionnement général de la DLL est le suivant. Les flux vidéo sont émis par l'application serveur vers la DLL en utilisant l'interface C selon un débit calculé à partir des informations obtenues à l'aide de l'interface R.

Les données vidéo sont alors paquetisées avec, éventuellement, des informations de redondances calculées par le module FEC². Ces paquets sont ensuite encapsulés dans des paquets RTP³, marqués éventuellement pour tirer parti des mécanismes de différenciation de service.

En retour, l'application cliente émet vers le serveur des informations de QoS à l'aide de paquets RTCP RR. Ces informations servent à déterminer à l'aide du module de contrôle de congestion les valeurs utilisées pour la régulation du débit et accessibles par l'application depuis l'interface R.

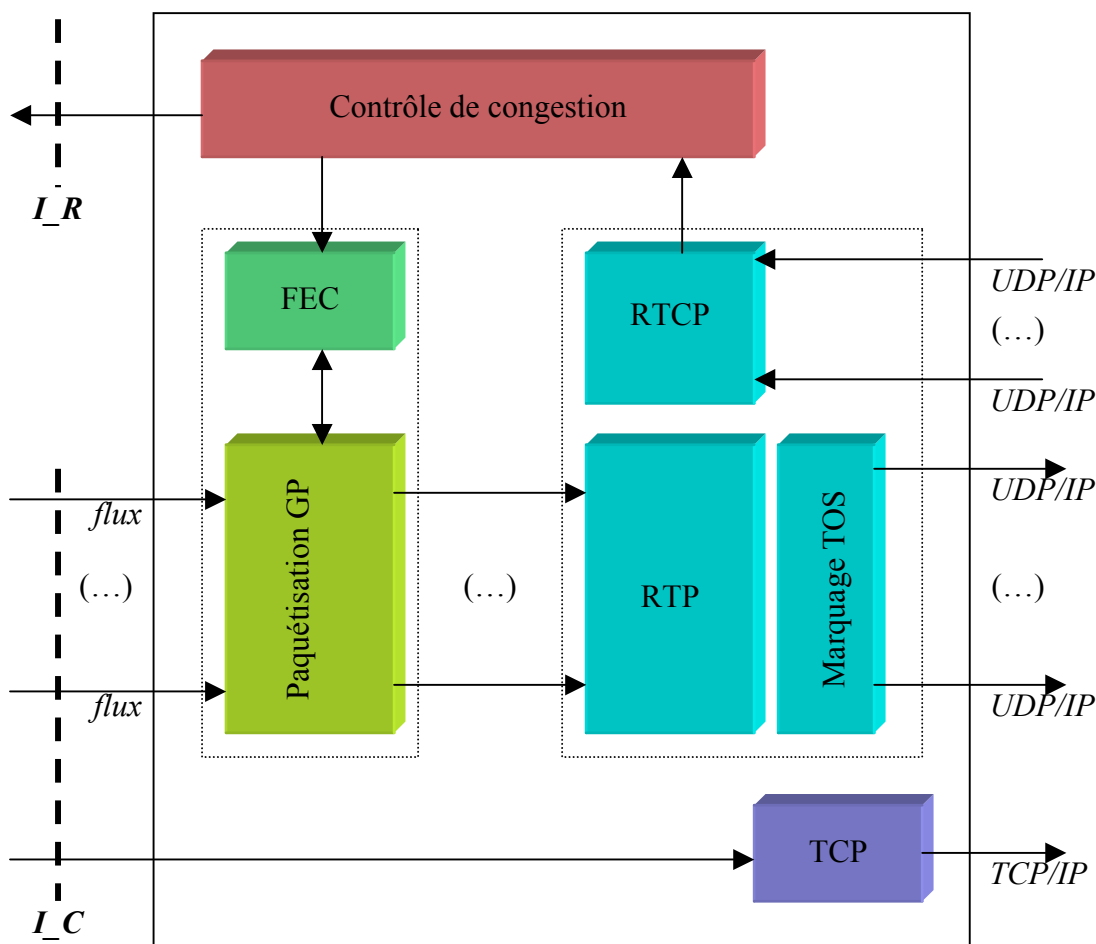


Figure 1 : Modules principaux de la DLL "CR_Inria.dll"

² FEC:Forward Error Correction.

³ [RFC 1889].

Le schéma montre aussi que l'interface C peut être utilisée pour établir des connexions en mode TCP. Par exemple, cela peut être le cas lorsqu'une connexion RTSP⁴ est établie ou qu'il est, d'une façon générale, nécessaire de transmettre des informations sur un canal fiable. Chacune de ces fonctions est détaillée dans les chapitres suivants.

3.1 Transport de la vidéo sur RTP

La DLL « CR_Inria.dll » transporte des flux vidéo MPEG-4 et H.263+ selon le format spécifié dans [DRAFT GENRTP]. Ce format, Generic Payload ou GP, permet de :

- transporter des flux élémentaires (MPEG-4 ES⁵) et des paquets SL-PDU⁶ incluant des informations de synchronisation,
- regrouper au sein d'un même paquet des AU⁷ de différents types comme des trames audio ou des images vidéo,
- regrouper les données utiles et les données de redondance (duplications de données d'en-tête nécessaires à la configuration du décodeur ou codes correcteurs).

La zone utile, ou payload, d'un paquet RTP est structurée en objet. Ces objets sont :

- soit des données classiques, relatives à une image par exemple (AU ou fragment d'AU),
- soit d'autres types de données comme celles de redondance.

3.2 Contrôle de congestion en point à point (UNICAST)

Le principe de base sur lequel repose le contrôle de congestion (point à point) est un processus adaptatif impliquant émetteur et récepteur, destiné en fonction du réseau, à contrôler le débit d'émission.

Les caractéristiques du réseau considérées sont pour chaque flux appartenant à un session :

- le débit,
- le taux de perte de paquets.

Pour calculer ces valeurs qui seront accessibles via l'interface R (cf. 4.2), le module de contrôle de congestion traduit les rapports RTCP RR relatifs à chaque canal RTP en consignes de débits.

Deux modes de contrôle de congestion en point à point sont possibles selon que l'application est dans un contexte codage temps réel ou avec une source FGS.

3.2.1 Contexte codage temps réel

Dans le cas d'un contexte codage temps réel, l'application qui utilise la DLL « CR_Inria.dll », répartit la bande passante estimée du réseau entre la source vidéo et l'information de redondance (cf. 3.4), en utilisant en utilisant par exemple un algorithme d'optimisation débit-distorsion afin d'optimiser la qualité du signal reconstruit.

⁴ RTSP : Real Time Streaming Protocol [RFC 2326].

⁵ ES : Elementary Stream.

⁶ SL-PDU : Synchronization Layer Protocol Data Unit.

⁷ AU : Access Unit.

3.2.2 Source FGS

Dans le cadre d'une source FGS, l'application qui utilise la DLL « CR_Inria.dll » peut tirer partie des informations fournies par la DLL pour, par exemple, transmettre la couche de base (BL), à débit constant et mettre en œuvre le contrôle de congestion au niveau du codage de la ou des couches d'amélioration.

La DLL offre la possibilité de transmettre les flux relatifs à la couche de base et aux couches d'amélioration dans autant de sessions RTP séparées. L'algorithme de régulation du débit de l'application serveur appelant la DLL devra tenir compte des informations de retour associées à chaque session RTP.

3.3 Contrôle de congestion multipoint (MULTICAST)

La DLL « CR_Inria.dll » permet de diffuser des flux vidéos en multipoint ou mode multicast. Toutefois, les communications multicast posent des problèmes liés à l'hétérogénéité des conditions de transmission dans le réseau mais aussi aux capacités de décodage et de traitement des récepteurs.

L'extension des approches point à point à des communications multipoint résiste mal au facteur d'échelle. En particulier, la multiplication des rapports de réception entraîne une charge inacceptable pour le réseau et le serveur.

Aussi, la DLL permet à l'application de mettre en œuvre une technique de contrôle de débit multipoint hybride source/récepteurs. Par exemple, un principe possible consiste en un système d'abonnement/désabonnement des récepteurs et en un mécanisme d'agrégation des retours. La DLL ne fournit pas d'agent d'agrégation. Par contre, la bibliothèque RTP/RTCP utilisée a été enrichie pour les supporter. Les paquets RTCP RR comprennent un vecteur de champs r_i c_i , où r_i et c_i sont respectivement les débits requis par les récepteurs et c_i le nombre de clients demandant le débit r_i . Ces vecteurs doivent être construits par des agents d'agrégation répartis sur le réseau.

3.4 Contrôle de pertes

La DLL « CR_Inria.dll » comprend un mécanisme de contrôle de perte qui réagit aux congestions du réseau par une adaptation conjointe du débit des sources et de la redondance nécessaire à la protection.

Le niveau de protection est dynamique et la valeur initiale choisie en début de session (cf. §5).

4 Description des interfaces

La DLL « CR_Inria.dll » s'utilise à l'aide de deux interfaces :

- l'interface C pour accéder aux fonctions de communication,
- l'interface R pour accéder aux fonctions de régulation.

Ces deux fonctions exposées par ces deux interfaces sont détaillées dans les chapitres suivants.

4.1 Description de l'interface C

L'interface C est utilisée par les applications serveurs et clientes pour accéder aux fonctions réseau. Elles permettent de créer et paramétrer un canal de communication puis de l'utiliser pour émettre et recevoir des données.

Les chapitres suivants décrivent ces fonctions de programmation d'un canal.

4.1.1 Création d'un canal

La création d'un canal est la première opération proposée par l'interface à effectuer. Elle s'effectue en créant un objet de type « C_Channel_Inria ».

```
C_Channel_Inria* pChannel = new C_Channel_Inria()
```

4.1.2 Description d'un canal

La description d'un canal s'effectue à l'aide de la méthode suivante :

```
DWORD C_Channel_Inria::Setup(C_DescrChannel * ptDescrChannel)
```

Avec comme paramètre, un pointeur sur un descripteur de canal créé par l'appelant contenant, en particulier, des informations sur le type de canal (TCP ; UDP ou GP/RTP) et les caractéristiques de la socket associée :

- le numéro de port local,
- le numéro de port distant,
- l'adresse IP locale,
- l'adresse IP distante.

La classe « C_DescrChannel » est décrite dans le fichier « CR_classes.h ».

4.1.3 Description du flux de données associé au canal

Le flux de données associé au canal est décrit à l'aide de la méthode suivante :

```
DWORD C_Channel_Inria::SetupStream(C_StDataStream * ptStDataStream)
```

Avec pour paramètre un pointeur sur un objet de type « C_StDataStream » contenant des informations sur le flux :

- l'identifiant de session,
- le type de flux(e.g. IOD, OD, BIFS, VIDEO),
- le numéro de la couche pour la vidéo (0 = BL, 1, 2, 3).

La classe « C_StDataStream » est définie dans le fichier « CR_classes.h ».

4.1.4 Connexion

La connexion d'un canal de communication s'effectue à l'aide de la méthode suivante :

```
DWORD C_Channel_Inria::ChannelAdd()
```

4.1.5 Attente de connexion

Un canal de communication peut être mis en attente de connexion à l'aide de la méthode suivante :

```
DWORD C_Channel_Inria::ReceiveChannelAdd(DWORD pDelay)
```

Où l'entier « pDelay » représente le nombre de millisecondes pendant lesquelles une connexion entrante est attendue. Si cette valeur vaut NULL, l'attente d'une connexion est illimitée et bloquante.

4.1.6 Emission de données

L'émission de données sur un canal paramétré et connecté s'effectue avec l'une des deux formes suivantes de la méthode :

```
DWORD C_Channel_Inria::SendData(C_StBuffer *ptStBuffer)
```

```
DWORD C_Channel_Inria::SendData(C_StBuffer *ptStBuffer, C_StDataLst *ptStDataLst)
```

Avec pour paramètres des pointeurs sur des objets de type :

- C_StBuffer, qui contient les données à émettre,
- C_StDataLst qui contient une liste d'objet de type « C_StData » contenant des informations vidéos.

Les trois classes, « C_StBuffer », « C_StDataLst », « C_StData » sont décrites dans le fichier « CR_classes.h ».

La seconde forme la méthode est, par exemple, utilisée des pompes vidéos qui émettent les informations vers la DLL « CR_Inria.dll ».

4.1.7 Attente de réception de données

La réception de données sur un canal paramétré et connecté s'effectue avec l'une des trois formes suivantes de la méthode :

```
DWORD C_Channel_Inria::ReceiveData(C_StBuffer *ptStBuffer, DWORD pDelay)
```

```
DWORD C_Channel_Inria::ReceiveData(C_StBuffer *ptStBuffer, C_StDataLst *ptStDataLst, DWORD pDelay)
```

```
DWORD C_Channel_Inria::ReceiveData(C_StBuffer *ptStBuffer, C_StData *ptStData, DWORD pDelay)
```

Avec pour paramètres des pointeurs des objets de type:

- C_StBuffer, contenant un buffer de réception et des informations associées (e.g. nombre de caractères lus),
- C_StDataLst contenant une liste d'objet de type « C-StData » contenant des informations vidéos.

Le dernier paramètre « pDelay » représente le nombre de millisecondes pendant lesquelles les données sont attendues. Si cette valeur vaut NULL, l'attente de données en réception est illimitée et bloquante.

Les trois classes, « C_StBuffer », « C_StDataLst », « C_StData » sont décrites dans le fichier « CR_classes.h ».

4.1.8 Libération d'un canal

Un canal est libéré à la l'aide de la méthode suivante :

```
DWORD C_Channel_Inria::ChannelDelete()
```

4.2 Description de l'interface R

L'interface de régulation est utilisée par les applications serveur pour lire les consignes de débit calculées par le module de contrôle de congestion de la DLL. Elle permet pour la couche de base et les couches d'amélioration d'obtenir deux informations caractéristiques :

- le débit,
- le taux de perte.

Les appels aux deux fonctions d'information associées de l'interface sont décrits dans les chapitres suivants.

4.2.1 Fonction d'information sur le débit

La DLL « CR_Inria.dll » peut fournir à l'application utilisatrice le débit possible pour chacune des couches d'une même session à l'aide de la fonction suivante :

```
CR_INRIA_API DWORD R_GetRateSession_Inria(  
    BYTE    IdSession,           // Identifiant de ma session  
    DWORD * BL_Rate,            // Débit pour le base layer  
    DWORD * EL1_Rate,           // Débit pour l'enhance layer 1  
    DWORD * EL2_Rate,           // Débit pour l'enhance layer 2  
    DWORD * EL3_Rate);         // Débit pour l'enhance layer 3
```

Avec pour paramètres :

- l'identifiant de la session pour laquelle on cherche à connaître le débit des différentes couches,
- des pointeurs sur des entiers dans lesquels seront stockées les valeurs des débits possibles pour chacune des couches.

4.2.2 Fonction d'information sur le taux de perte

La DLL « CR_Inria.dll » peut fournir à l'application utilisatrice le taux de perte pour chacune des couches d'une même session à l'aide de la fonction suivante :

```
CR_INRIA_API DWORD R_GetPacketLostSession_Inria(  
    BYTE    IdSession,           // Identifiant de ma session  
    DWORD * BL_PacketLoss,       // Perte de paquet pour la couche base layer  
    DWORD * EL1_PacketLoss,      // Perte de paquet pour la couche enhance layer  
    DWORD * EL2_PacketLoss,      // Perte de paquet pour la couche enhance layer 2,  
    DWORD * EL3_PacketLoss       // Perte de paquet pour la couche enhance layer 3  
);
```

Avec pour paramètres :

- l'identifiant de la session pour laquelle on cherche à connaître le débit des différentes couches,
- des pointeurs sur des entiers dans lesquels seront stockées les valeurs des taux de pertes pour chacune des couches.

5 Paramétrages de la DLL

Le comportement de la DLL « CR_Inria.dll » est paramétré en fonction d'un fichier de configuration, « cr_inria.cfg », exploité à chaque chargement de la DLL. Il est structuré en trois sections qui décrivent les paramètres généraux et ceux propres à une utilisation côté serveur ou client de la DLL. Un exemple de fichier de configuration est donné ci-après. Les différents paramètres sont détaillés dans les chapitres suivants.

```
[General]
TMM = 1
Multicast = 0
CCType = 1
Aggregator_IP = 139.100.18.73
Aggregator_Port = 3300
[Server]
ToSType = 0
Regul = 1
Loss = 0
EG_Q = 0.0
EG_P = 0.0
InitialBW = 1024000
Protect = 0
ProtectRate = 0.0
ELDebit = 768000
BLInitBitRate = 256000
EL1InitBitRate = 768000
EL2InitBitRate = 0
EL3InitBitRate = 0
[Client]
Correct = 0
InitialBandWidth = 1024000
MaxBW = 1024000
MinBW = 256000
```

Figure 2 : Exemple de fichier de configuration

5.1 Paramètres généraux

La section « [General] » du fichier de configuration contient les paramètres suivants :

Nom	Valeur	Rôle
TMM	0 ou 1	Type de codec : 0 : la DLL est en mode H263+ 1 : la DLL est en mode FGS
Multicast	0 ou 1	Mode de diffusion : 0 : mode unicast 1 : mode multicast
CCType	0 ou 1	Type de l'algorithme de contrôle de congestion : 0 : les statistiques des couches appartenant à la même session sont traitées séparément. 1 : les statistiques réseaux des couches appartenant à la même session sont consolidées pour estimer la bande passante disponible.

Aggregator_IP	Une adresse IP sous la forme X.Y.Z.T	Adresse IP de la machine hébergeant l'agent d'agrégation. Mode multicast uniquement.
Aggregator_Port	Entier non signé 16 bits	Numéro de port de l'agent d'agrégation dédié à la réception des rapports RTCP.

5.2 Paramètres serveurs

La section « [Server] » du fichier de configuration contient les paramètres suivants :

Nom	Valeur	Rôle
TosType	0, 1,2	Marquage des paquets via le champ ToS : 0 : pas de marquage de paquets. Dans le cas de l'utilisation de routeurs mettant en œuvre « Diffserv » : 1, 2 : classes de services dans le cas FGS. La couche de base (BL) contient des cadres I,P, B et la couche d'amélioration (EL) des cadres EI, EP, EB. Les couches BL et EL sont respectivement marquées comme la classe Diffserv AF1 et AF2. I,P,B sont marqués comme des sous classes de AF1 : AF11, AF12, AF12. De même, EI, EP et EB sont marqués comme des sous-classes de AF2 : AF21, AF22 et AF23.
Regul	0 ou 1	Exploitation des informations calculées par le contrôle de congestion par le serveur : 0 : ne sont pas prise en compte, 1 : prises en compte par le serveur.
Loss	0 ou 1	Simulation de perte dans la DLL selon un algorithme fondé sur Elliot-Gilbert : 0 : simulation activée, 1 : simulation désactivées.
EG_Q/EG_P	0<=valeur<1	Paramètres de l'algorithme Elliot-Gilbert.
InitialBW	Entier non signé 16 bits	Valeur de la bande passante initialement utilisée par le serveur.
Protect	0,1,2	Définition du taux de protection des données à l'aide de FEC. 0 : pas de protection de données 1 : « equal error protection », protection équitablement répartie entre les différents types de données 2 : « unequal error protection », certains

		types de données sont privilégiés et protégés (e.g. cadres I).
ProtectRate	0<valeur<1	Valeur initiale du taux de protection. Cette valeur correspond au pourcentage de place occupée par les données redondante par rapport aux données vidéos dans un paquet RTP. Une valeur faible, inférieure à 0.2, est conseillée.
ELDebit	Entier non signé 16 bits	Débit de la couche d'amélioration. Dans le cas où le mécanisme de régulation est désactivé (Regul = 0) et que la DLL est en mode FGS, cette valeur correspond au débit fixe de la couche d'amélioration.
BLInitRate EL1InitRate EL2InitRate EL3InitRate	Entiers non signés 16 bits	Débits initiaux des couches de base (BL) et d'amélioration (EL1, EL2, EL3).

5.3 Paramètres clients

La section « [Client] » du fichier de configuration contient les paramètres suivants utiles à la DLL côté client :

Nom	Valeur	Rôle
Correct	0 ou 1	Présence de FEC : 0 : pas de FEC, 1 : présence de FEC.
InitialBandWidth	0 ou 1	Valeur initiale de la bande passante côté client.
MaxBW/MinBW	0 ou 1	Bornes maximales et minimales de la bande passante déterminée par l'algorithme de contrôle de congestion.

6 Exemple d'utilisation des interfaces

Ce chapitre présente un exemple d'utilisation des interfaces « C » et « R ». Il rassemble les séquences d'appel à la DLL « CR_Inria.dll » nécessaire pour :

- établir une connexion avec un serveur,
- échanger des données,
- accéder aux informations utiles au contrôle de régulation.

6.1 Etablissement d'une connexion

Pour qu'une application client puisse établir une connexion, il est nécessaire qu'auparavant l'application serveur soit en attente d'une connexion. Pour cela, l'application serveur utilise l'API de la DLL « CR_Inria.dll » pour :

- décrire un canal de communication en créant un objet de type « C_DescrChannel » puis en précisant les valeurs de ses attributs, par exemple, le type de canal (C_TYPE_TCP, C_TYPE_RTP_UDP, C_TYPE_GP_RTP_UDP), les numéros de la socket... (cf. 4.1.2),
- créer puis initialiser un canal de communication de type « C_Channel_Inria » à l'aide de la méthode « C_Channel_Inria ::SetUp(C_DescrChannel* ptDescrChannel) » (cf. 4.1.2),
- attendre une connexion sur le canal de type « C_Channel_Inria » à l'aide de la méthode « C_Channel_Inria ::ReceiveChannelAdd(DWORD pDelay) » (cf. 4.1.5).

La classe « C_DescrChannel » est décrite dans « CR_classes.h ».

Une fois que le serveur est en attente de connexion, une application cliente utilise l'API de la DLL « CR_Inria.dll » d'une façon symétrique pour :

- décrire un canal de communication en créant un objet de type « C_DescrChannel » puis en précisant les valeurs de ses attributs, par exemple, le type de canal (C_TYPE_TCP, C_TYPE_RTP_UDP, C_TYPE_GP_RTP_UDP), les numéros de la socket... (cf. 4.1.2),
- créer puis initialiser un canal de communication de type « C_Channel_Inria » à l'aide de la méthode « C_Channel_Inria ::SetUp(C_DescrChannel* ptDescrChannel) » (cf. 4.1.2),
- se connecter au serveur à l'aide de la méthode « DWORD C_Channel_Inria::ChannelAdd() ».

6.2 Echange de données

Les applications serveurs et clients peuvent s'échanger des données. Par exemple, un canal ouvert à l'aide de l'API de la DLL « CR_Inria.dll » peut servir à transporter des requêtes « RTSP » du client vers le serveur et un autre transporte des données du serveur vers le client.

Aussi, pour présenter la phase d'échange de données, les termes plus génériques « émetteur » et récepteur » sont utilisés.

Le récepteur procède à la réception de données en deux étapes en :

- préparant la zone de réception en créant un objet de type « C_StBuffer » et en précisant la valeur de ses attributs notamment le pointeur sur la zone allouée pour la réception et la taille de cette zone,
- en attendant une réception de donnée sur le canal voulu à l'aide de la méthode « DWORD C_Channel_Inria::ReceiveData(C_StBuffer *ptStBuffer, DWORD pDelay) » ou d'une des ses formes associées (cf. 4.1.7).

L'émetteur procède symétriquement en :

- préparant la zone d'émission en créant un objet de type « C_StBuffer » et en précisant la valeur de ces attributs notamment le pointeur sur la zone allouée pour la réception et la taille de cette zone.
- émettant les données à l'aide de la méthode « DWORD C_Channel_Inria::SendData(C_StBuffer *ptStBuffer) » ou l'autre forme associée (cf. 4.1.6),
- libérant les zones allouées une fois que l'émission est effective.

Si un canal n'est plus utile, il est nécessaire de le libérer à l'aide de la méthode « DWORD C_Channel_Inria::ChannelDelete() ».

6.3 Accès aux informations utiles au contrôle de congestion

Les deux fonctions fournies par l'interface « R » (cf.4.2) de la DLL « CR_Inria.dll » sont appelables à tout moment par l'application serveur pour disposer des informations utiles à la mise en œuvre de la politique de contrôle de régulation de débit choisie. Toutefois, la fréquence de ces prises de mesures doit être :

- compatible avec la charge et les fonctions du moment du serveur (e.g. l'émission d'images liées),
- inférieure à la fréquence de rafraîchissement de ces informations.

Annexe A Références

- [DRAFT GENRTP] Guillemot C., Christ P., Wesner S., “RTP Payload for MPEG-4 with Scaleable & Flexible Error Resiliency”, draft-guillemot-genrtp-01.txt, INRIA, RUS. June 2,1999 Expires:December 1,1999.
- [RFC 1889] Schulzrinne H., Casner S., Frederick R., Jacobson V., « RTP : A Transport Protocol for Real-Time Applications », RFC 1889, GMD Fokus ; Precept Software, Xerox Palo Alto Research Center, Lawrence Berkeley National Laboratory. January 1996.
- [RFC 2326] Schulzrinne H., Rao A., Lanphier R. « Real Time Streaming Protocol », RFC 2326, Columbia University, Netscape, RealNetworks. April 1998.

Annexe B Abréviations

API	Application Programming Interface
AU	Access Unit
BL.....	Base Layer
DLL	Dynamic Link Library
EL	Enhancement Layer
ES	Elementary Stream
FEC.....	Forward Error Correction
FGS.....	Fine Grain Scalability
GP.....	Generic Payload
IP	Internet Protocol
MOVIQS	Module pour de la vidéo sur Internet
QoS.....	Quality of Service
RTCP	Real Time Streaming Protocol
RR.....	Receiver Report
RTP.....	Real Time Protocol
SL-PDU	Synchronization Layer Protocol Data Unit.
TCP.....	Transport Control Protocol
TOS	Type Of Service
UDP.....	User Datagram Protocol

Annexe C Reconstruction de la DLL “CR_Inria.dll”

Les sources de la DLL « CR_Inria.dll » sont regroupés dans une structure arborescente de répertoires dont la racine est le répertoire « DepotAPP ».

La DLL peut être reconstruite à l’aide du fichier « dsp » se trouvant sous le répertoire « CR_Inria ». Ce fichier « dsp » fait appel à d’autres projets dont la localisation est relative au répertoire racine « DepotAPP ».