H.264 LINUX Network Library API Specification

(For Linux)

(2004.11.13 newest Version: 3.2)

Modification of newest Version (3.2 Version)

Server: Support PV-260

client: Support PV-245、250 & 260.

V3.2Add:
Support audio and video data are transport in detachable mode.

We will only add functions in future Version rather than modify current functions any more.

Introduction

The Network Library provides a high level interface for developing applications that capture and playback audio/video via network. The behavior is defined as follow:

1. The Library includes server (libtmnets.so) and client (libtmnetc.so). The server runs with the H.264 Linux cards and must use with the SYS_SDK for linux(libtmsdk.so).The clients could run on the remote or even local machines. It supports connecting with windows use corresponding library.
2. The Library supports TCP, UDP, Multicast transfer mode.
   Now support TCP connect (max 16 connections per channel),
   UDP connect (max 16 connections per channel)
   MULTICAST connect (no limit);
3. The client can open several video display windows at the same time. The number of windows can be viewed depends on the performance of host CPU (for example P4 1.6G ----9 windows; P3 800----4 windows, etc).
4. If the client is running in MS-WINDOWS platform. The client use the library for Windows (proclient.dll).The display adapter in client must support DirectDraw Blt Shrink and Stretch by hardware if the client want to open several video display windows at the same time.
5. The server can only running on the system that has H.264 series cards installed.

Notes

1. After switching stream type for one channel (using Sys_sdk API SetStreamType()),client must reconnect;
2. You should use MP4_ServerSetStart and MP4_ServerSetStop before MP4_ServerStart;
3. BUFNUM is set to the default value (128), you could change it by using MP4_ServerSetBufNum before MP4_ServerStart;
4. MP4_ServerCheckIP and MP4_ServerCheckPassword should be used after MP4_ServerStart. If the Use checking and IP checking are not wanted, use MP4_ServerCheckIP(NULL) and MP4_ServerCheckPassword(NULL).
5. MP4_ClientSetNetPort should be used before MP4_ClientStartup MP4_ServerSetNetPort should be used before MP4_ServertStartup
6. Two parameters of MP4_ClientSetNetPort must be equal to the two parameters of MP4_ServerSetNetPort .

The Server End

Define for structure

```
1.typedef struct{char      m_datatype[MAX_CHANNEL_SUPPORT];//not use now
          char m_channum;  // number of channel
          int   m_waittime;   //not in use now
          int   m_bufnum;     //not in use now
}SERVER_VIDEOINFO,*PSERVER_VIDEOINFO;
```

Function Useing Order

```
int MP4_ServerSetNetPort（）;
int MP4_ServerSetBufNum（）;
int MP4_ServerSetStart（）;
int MP4_ServerSetStop（）;

        int MP4_ServerStart();

                int MP4_ServerGetState();

                int MP4_ServerCheckIP（）;
                int MP4_ServerCheckPassword（）;
                int MP4_ServerSetTTL（）;
                int MP4_ServerSetMessage ();


                int MP4_ServerWriteData();
                int MP4_ServerStringToClient();
                int MP4_ServerCommandToClient();
                int MP4_ServerResetChannel();

        int MP4_ServerStop();
```

Server Functions Define

This section gives a more detailed specification of the functions available in the server API.

1.int MP4_ServerSetNetPort(int iServerPort, int iClientPort);

Explanation:Set Server base port and Client base port;

Parameter: int iServerPort : Server base port

int iClientPort : Client base port

Return value: success return 0,fail return -1 ;

2.int MP4_ServerSetBufNum(int nChannel, int dBufNum);

Explanation:Set sending buffer size;

Parameter: int nChannel : channel num;

int dBufNum : buffer size .(unit is 8k , for example : if dBufNum = 10 , the buffer size = 80k . dBufNum = 30 default)

Return value: success return 0 , fail return –1;

3.int MP4_ServerSetStart(void (* StartCapCallBack)(int port));

Explanation: Set Callback Function at MP4_ServerStart,

int port : channel num

In MP4_ServerStart, StartCapCallBack will be called;

Return value: success return 0 , fail return –1;

4.int MP4_ServerSetStop(void (* StopCapCallBack)(int port));

Explanation: Set Callback Function at MP4_ServerStop,

int port : channel num

In MP4_ServerStop, StopCapCallBack will be called;

Return value: success return 0 , fail return –1;

5.int MP4_ServerSetMessage(void (* MessageCallBack)(char * buf, int iLen));

Explanation : Set Callback Function at Message arrive;

Parameter: buf : Message buffer Pointer;

iLen : Message length;

Return value: success return 0 , fail return –1;

6.int MP4_ServerCheckIP(int (* CheckIPCallBack)(int iChannel, char * sIP));

Explanation : Set Check IP Callback Function;

When client try to connect server , CheckIPCallBack will be called;

IChannel : channel num, sIP : client IP address

CheckIPCallBack return 0 means checking pass, or return –1;

Return value: success return 0 , fail return –1;

7.int MP4_ServerCheckPassword(int (* CheckPasswordCallBack)(char * usename, int namelen, char * password, int passlen));

Explanation: Set Check Password callback Function;

When client try to connect server , CheckPasswordCallBack will be called;

usename : channel num, namelen : length of name,password : user password,

passlen : length of password ; (now namelen = password = 50, they do not represent actual length),

CheckPasswordCallBack return 0 means checking pass, or return –1;

Return value: success return 0 , fail return –1;

8.int MP4_ServerStart(PSERVER_VIDEOINFO videoinfo);

Explanation : start server

Return value: success return 0 , fail return –1;

9.int MP4_ServerStop();

Explanation : stop server

Return value: success return 0 , fail return –1;

10.int MP4_ServerWriteData(int nPort, char * pPacketBuffer, int nPacketSize, int frameType, int breakable);

Explanation : write data to send buffer, net server sdk use this function to get data to send;

Parameter : nPort : channel num

PPacketBuffer : pointer of packet buffer;

NPacketSize : packet size;

FrameType : frame type of packet;

Breakable : (not use now)

Return value: success return 0 , fail return –1;

11.int MP4_ServerStringToClient(char * m_lAddrIP, char * m_sCommand, int miLen);

Explanation : send message to client;

Parameter : m_lAddrIP : client ip address;

M_sCommand : pointer of message;

M_wLen : length of message (<900)

Return value: success return 0 , fail return –1;

12.int MP4_ServerResetChannel(int nChannel);

Explanation : reset one channel of server (stop all connection to this channel);

Parameter : nChannel : channel num;

Return value: success return 0 , fail return –1;

13.int MP4_ServerGetState();

Explanation : get server state

Return value : -1 : server is not start ,other values means the num of conncetions to server;

14.int MP4_ServerSetTTL(unsigned char cTTLVal);

Explanation : set TTL of multicast

Parameter : cTTLVal : value of TTL (1-255,default is 32)

Return value: success return 0 , fail return –1;

15.int MP4_ServerCommandToClient(int iChannel,unsigned char cCommand);

Explanation : send command code to all clients that connect with iChannel of server;

Parameter : iChannel : channel num , cCommand : command code (0-10 is reserved for sys)

Return value: success return 0 , fail return –1;

The Client End

Note: When connect protocol is MULTICAST, client and server could not running in the same PC;

Define for structure

```
enum{PTOPUDPMODE=0, PTOPTCPMODE, MULTIMODE, ONLYAUDIOMODE,NOUSEMODE};
    PTOPUDPMODE：UDP connect type
    PTOPTCPMODE：TCP connect type
    MULTIMODE：    MULTICAST connect type
    ONLYAUDIOMODE: Audio and Video transport in detachable mode
    NOUSEMODE：  not in use now

typedef struct tagClientShowRect{
    unsigned int   uWidth;          // width of display window
    unsigned int   uHeight;         // height of display window
}CLIENTSHOWRECT;

typedef struct{
    char m_bRemoteChannel;          // channel num that want to connect
    char m_bSendMode;               // connect type
    char m_bRight;                  //not in use now
    char * m_sIPAddress;            // server IP address
    char * m_sUserName;             // user name
    char * m_sUserPassword;         // user password
    int m_bUserCheck;               // whether sending user name and pass to server(0/1)
    unsigned int subshow_x;         //start point (x) of display area
    unsigned int subshow_y;         //start point (y) of display area
    unsigned int subshow_uWidth;    //width of display area
    unsigned int subshow_uHeight;   //height of display area
}CLIENT_VIDEOINFO,*PCLIENT_VIDEOINFO;
```

Function Using Order

int MP4_ClientSetNetPort();

int MP4_ClientStartup();

int MP4_ClientGetServerChanNum();
int MP4_ClientCommandToServer();
int MP4_ClientCommandToServer_Handle();
int MP4_ClientShut();

```
int MP4_ClientStart();
        int MP4_ClientSetBufferSize();
        int MP4_ClientSetDelayBufferSize();

        int MP4_ClientAudioStart();
        int MP4_ClientAudioStop();
        int MP4_ClientAudioVolume();

        int MP4_ClientGetState();
        int MP4_ClientStartCapture();
        int MP4_ClientStartCaptureFile();
        int MP4_ClientStopCapture();

    int MP4_ClientStop();

int MP4_ClientCleanup();
```

Client Functions Define

1.int MP4_ClientSetNetPort(int iServerPort, int iClientPort);

See MP4_ServerSetNetPort

2.int MP4_ClientStartup(CLIENTSHOWRECT clientrect,
void MessageCallBack(int imark,
char command,
char * buf,
int iLen));

Explanation : Initial Client , set call back function at message arrive;

Parameter : clientrect : See structure CLIENTSHOWRECT;

MessageCallBack : when message or command code is arrive ,this function Will be called;

Imark : 0-command arrive, 1 –message arrive

Command :command c

Buf : pointer of message

ILen :length of message

Return value: success return 0 , fail return –1;

3.int MP4_ClientCleanup();

Explanation : release client.

Return value: success return 0 , fail return –1;

4. int MP4_ClientGetServerChanNum(char * m_sIPAddress);

Explanation : get number of channels of server;

Parameter : m_sIPAddress : server IP address

Return value: success return 0 , fail return –1;

5.int MP4_ClientStart(PCLIENT_VIDEOINFO pClientinfo,
void ReadDataCallBack(int StockHandle,
char * pPacketBuffer,
int nPacketSize));

Explanation : Start one client;

Parameter : pClientinfo : see structure PCLIENT_VIDEOINFO,
ReadDataCallBack : call back function at data arrive , if you do not
Want to deal with stream data, set this parameter to NULL;

Return value: success return client handle , fail return –1;

6.int MP4_ClientStop(int StockHandle);

Explanation : close client;

Parameter : StockHandle : client handle (return by MP4_ClientStart)

Return value: success return 0 , fail return –1;

7. int MP4_ClientGetState(int StockHandle);

Explanation : get state of client;

Parameter : StockHandle : client handle (return by MP4_ClientStart)

Return value : -1 : get state fail
1 : be connecting
2 : be receiving stream data
3 : abort by error
4 : finish receiving
5 : can not connect server
6 : connecting is denied by server

8. int MP4_ClientCommandToServer(char * m_sIPAddress,
char * m_sCommand,
int m_iLen);

Explanation : send string to server;

Parameter : m_sIPAddress : server IP address,
M_sCommand : string buffer pointer,
M_iLen : length of string (<900)

Return value: success return 0 , fail return –1;

9. int MP4_ClientCommandToServer_Handle(int StockHandle,
char * m_sCommand,
int m_iLen);

Explanation : same as MP4_ClientCommandToServer;

Parameter : StockHandle : client handle (return by MP4_ClientStart),
M_sCommand : string buffer pointer,
M_iLen : length of string (<900)

Return value: success return 0 , fail return –1;

10.int MP4_ClientStartCapture(int StockHandle);

Explanation : start stream data capture (ReadDataCallBack will be called when data arrive)

Parameter : StockHandle : client handle (return by MP4_ClientStart)

Return value: success return 0 , fail return –1;

11.int MP4_ClientStartCaptureFile(int StockHandle, char * FileName);

Explanation : start stream data capture (write to file)

Parameter : StockHandle : client handle (return by MP4_ClientStart)
FileName : file name;

Return value: success return 0 , fail return –1;

12.int MP4_ClientStopCapture(int StockHandle);

Explanation : stop stream data capture (do work to MP4_ClientStartCapture and MP4_ClientStartCaptureFile)

Parameter : StockHandle : client handle (return by MP4_ClientStart)

Return value: success return 0 , fail return –1;

13.int MP4_ClientShut(char * m_lAddrIP,char cChannel);

Explanation : reset cChannel of server (cut off all connections to cChannel of server)

Parameter : m_lAddrIP : server IP address
CChannel : channel num

Return value: success return 0 , fail return –1;

14. int MP4_ClientSetBufferSize(int StockHandle, int iBufsize);

Explanation : set size of receive buffer;

Parameter : StockHandle : client handle (return by MP4_ClientStart)
IBufsize : buffer size (50*1024<=iBufsize<=1000*1024, default is 1000*1024)

Return value: success return 0 , fail return –1;

Note: This function must be called immediately after MP4_ClientStart().

15.int MP4_ClientSetDelayBufferSize(int StockHandle, int iDelaybufsize);

Explanation : set delay data size (when receive data is add up to iDelaybufsize, start play)

Parameter : StockHandle : client handle (return by MP4_ClientStart)
IDelaybufsieze : delay size (0-1000*1024 ,default is 40*1024)

Return value: success return 0 , fail return –1;

Note: This function must be called immediately after MP4_ClientStart().

16.int MP4_ClientAudioStart(int StockHandle);

Explanation : play sound of one client;

Parameter : StockHandle : client handle (return by MP4_ClientStart)

Return value: success return 0 , fail return –1;

17.int MP4_ClientAudioStop();

Explanation : stop sound of one client;

Parameter : StockHandle : client handle (return by MP4_ClientStart)

Return value: success return 0 , fail return –1;

18.int MP4_ClientAudioVolume(int StockHandle, unsigned int sVolume);

Explanation : adjust volume;

Parameter : StockHandle : client handle (return by MP4_ClientStart)

SVolume : value of volume (0-100 , default is 100)

Return value: success return 0 , fail return –1;

V2.4 add new routes:

19. void MP4_SetSdlUse(unsigned int used)

Parameter: unsigned int used 0 or 1

Explanation: Set the mode of the video and audio play. If used set 1,then direct play by SDK, or call by yourself. Please call before MP4_ClientStartup()

Define Of Error Code

```
#define TMNetErrorCodeWrongOperation        0x00000001
#define TMNetErrorCodeInvalidArgs           0x00000002
#define TMNetErrorCodeServerNotAvailable    0x00000003
#define TMNetErrorCodeTimeOut               0x00000004
#define TMNetErrorCodeReceiveUnknowPacket 0x00000005
#define TMNetErrorCodeCreateThreadFail      0x00000006
#define TMNetErrorCodeConnectClientFail     0x00000007
#define TMNetErrorCodeAllocMemFail          0x00000008
#define TMNetErrorCodeRTPError              0x00000009
#define TMNetErrorCodeCreateFileFail        0x0000000a
#define TMNetErrorCodeNoSystemHeader        0x0000000b
#define TMNetErrorCodeConnectServerFail     0x0000000c
```