
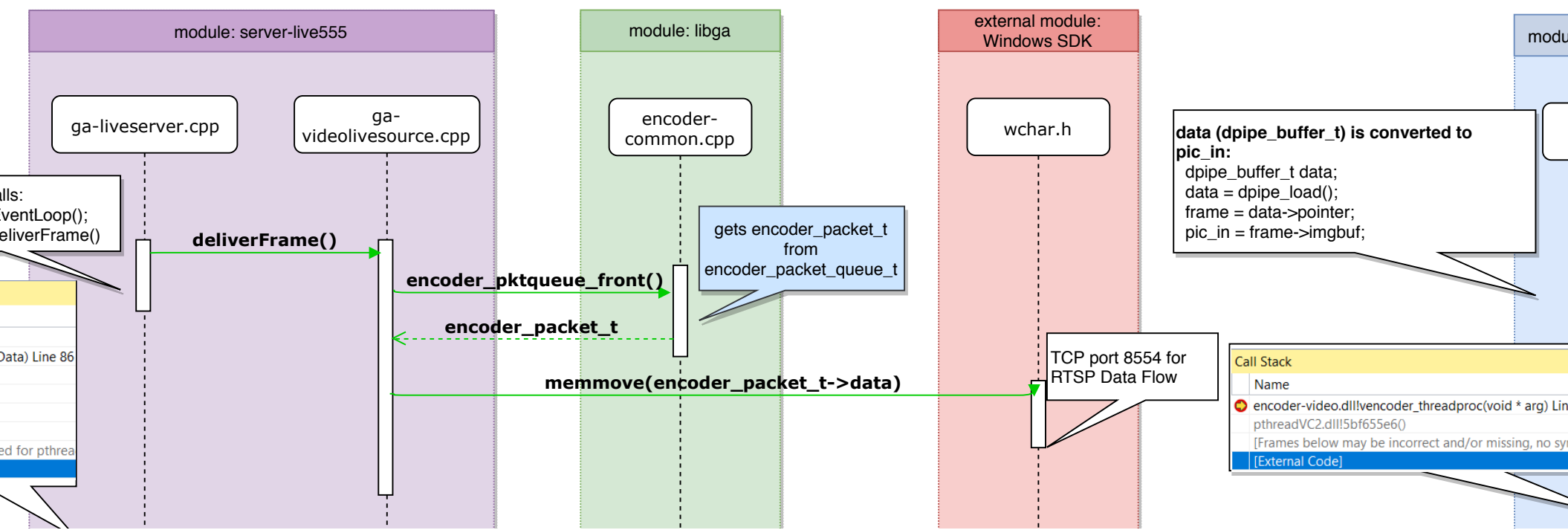
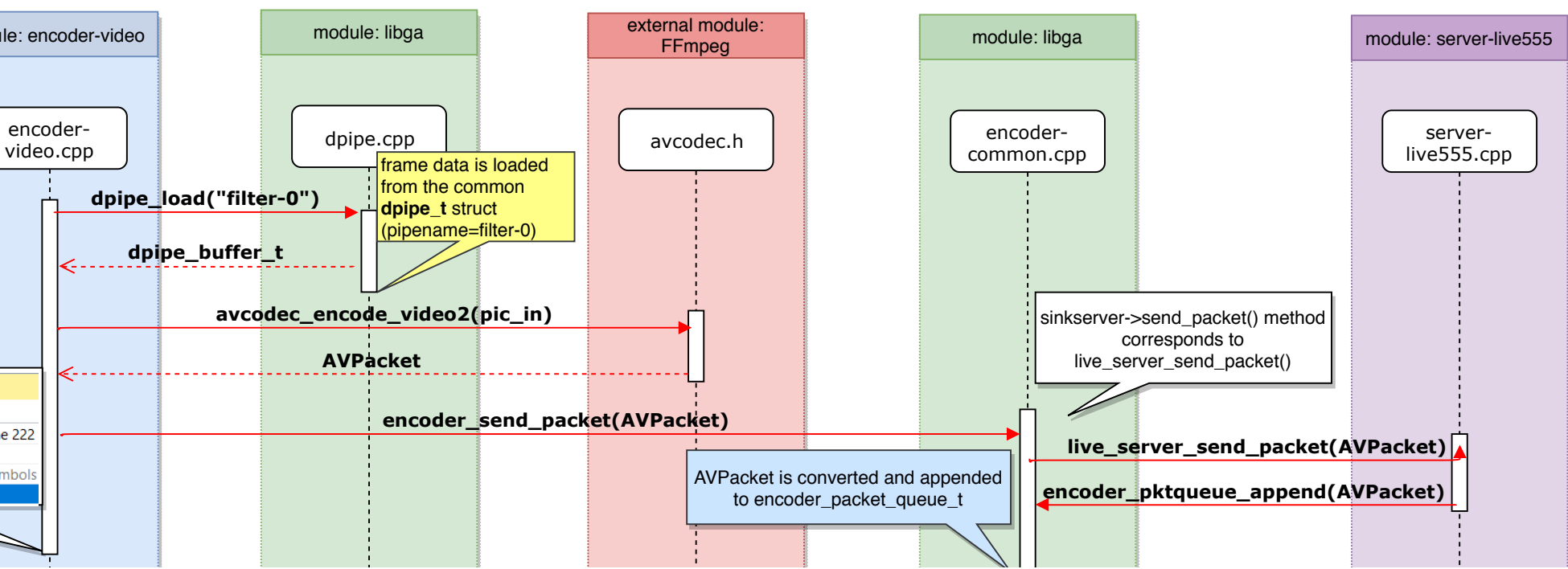
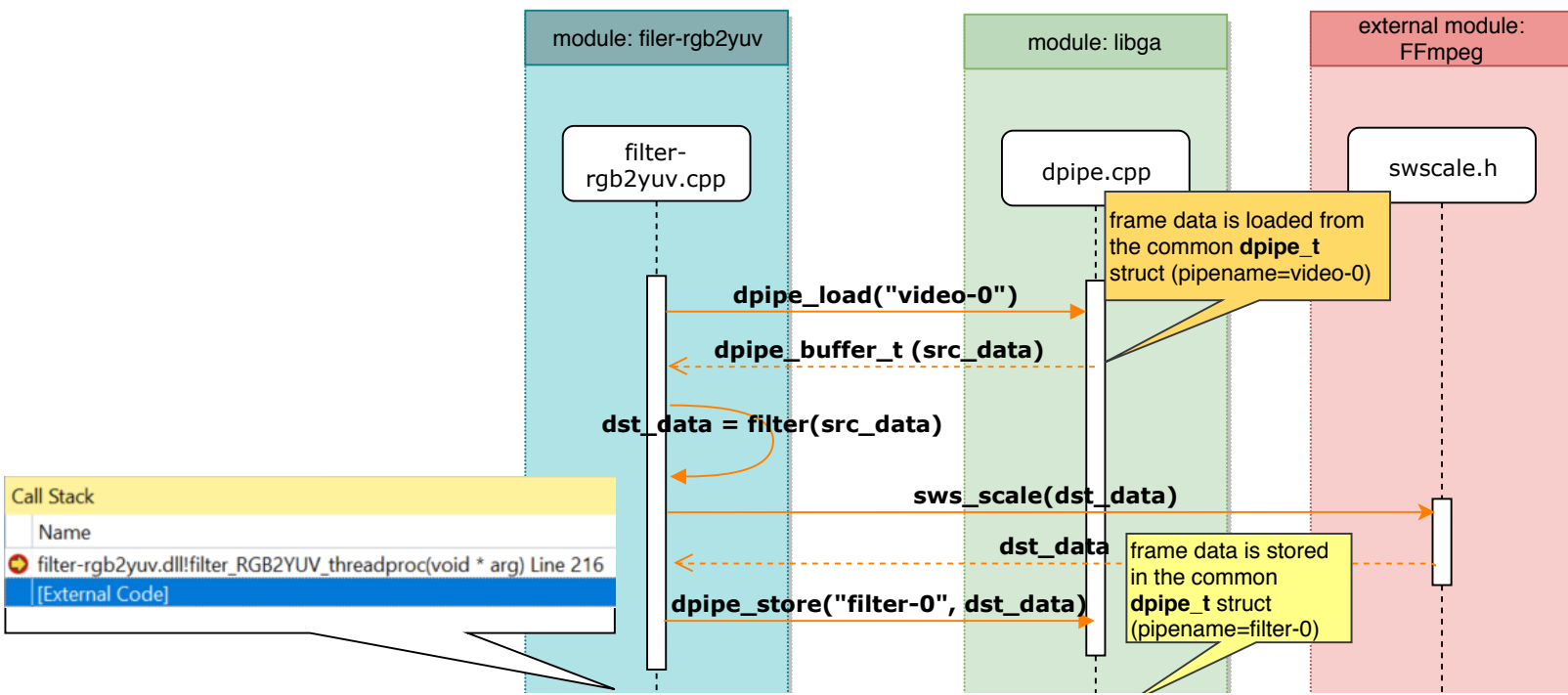


internally ca
taskScheduler().doE
then eventloop calls d

Call Stack	
Name	
 server-live555.dll!GAVideoLiveSource::deliverFrame() Line 145	
server-live555.dll!GAVideoLiveSource::deliverFrame0(void * clientD	
server-live555.dll!BasicTaskScheduler::SingleStep() Line 185	
server-live555.dll!BasicTaskScheduler0::doEventLoop() Line 81	
server-live555.dll!liveserver_main(void * arg) Line 87	
pthreadVC2.dll!5bf655e6()	
[Frames below may be incorrect and/or missing, no symbols loaded]	
[External Code]	

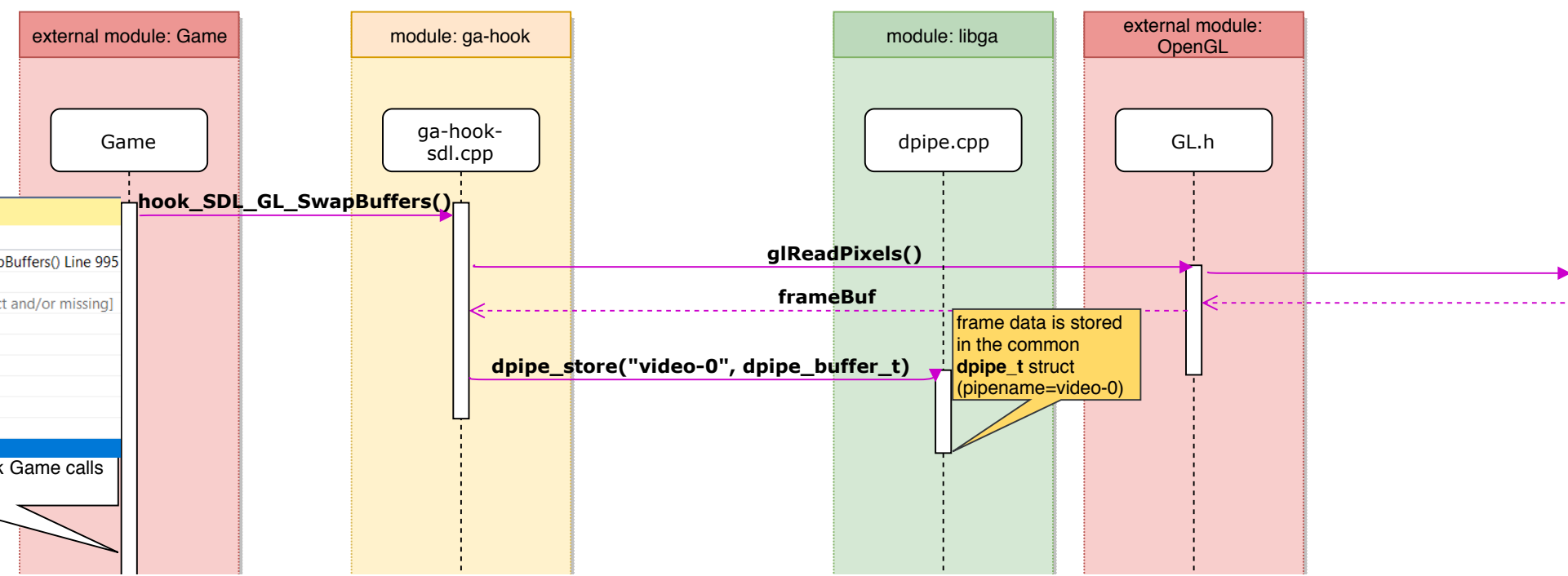
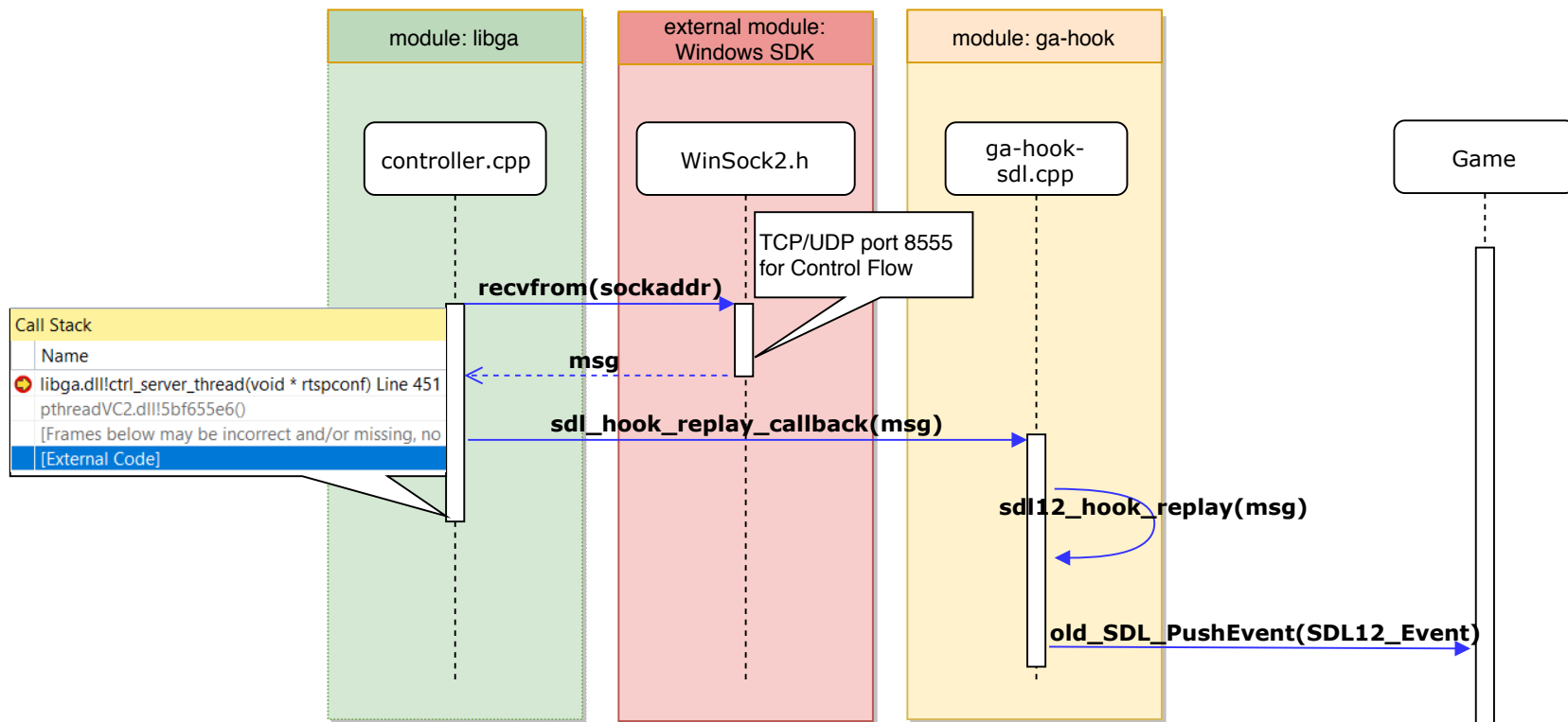


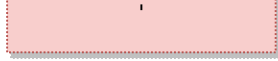
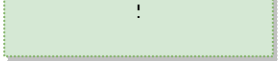




Call Stack	
Name	
filter-rgb2yuv.dll!filter_RGB2YUV_threadproc(void * arg) Line 216	
[External Code]	

Call Stack	
Name	
ga-hook.dll!hook_SDL_GL_Swap	
[External Code]	
[Frames below may be incorrect]	
neverball.exe!004234ca()	
neverball.exe!00423df7()	
neverball.exe!00423f89()	
neverball.exe!00423ae9()	
neverball.exe!004010a7()	
neverball.exe!00401123()	
[External Code]	
According to the Call Stack this function	





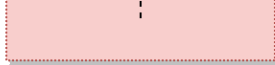
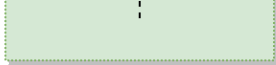
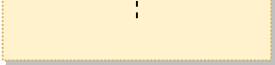
- purpose of server-live.cpp: servers are defined as ga modules (see. struct ga_module_t), module object has the function send_packet, which is to be implemented only by servers. server-live555.cpp is an implementation of server module, another implementation is server-ffmpeg.cpp. server modules register themselves to encoder-common (by calling encoder_register_sinkserver(&m)), encoder-common does not know about which implementation will be registered. It only calls send_packet() method, the method will be called on the currently registered server. server-live555.cpp only calls encoder_pktqueue_append() method of encoder-common but server-ffmpeg has a different implementation.
- dpipe and encoder_packet_queue: dpipe is used for storing the raw data from the GL library. It has an input pool (free frames) and an output pool (occupied frames) for data storage. encoder_packet_queue is used for storing the encoded data in a queue.
- uint8_t: it is used as "uint8_t* data" in AVPacket struct, in short it stores the raw compressed frame data. uint8_t* means that a pointer points to the beginning of data array, size of this array is stored in size field of AVPacket struct. For video, AVPacket contains 1 compressed frame.
- RTSP-Server: server-live555 module is the implementation of RTSP server according to the specifications of live555 streaming media library. rtspserver.cpp file in server-ffmpeg module is a complete implementation of an RTSP server from ground up, but it is not being used.









Important structs

```
typedef struct ga_module_s {
    HMODULE handle;    /**< Handle to a module */
    int type;          /**< Type of the module */
    char *name;        /**< Name of the module */
    char *mimetype;     /**< MIME-type of the module */
    int (*init)(void *arg);    /**< Pointer to the init function */
    int (*start)(void *arg);   /**< Pointer to the start function */
    //void * (*threadproc)(void *arg);
    int (*stop)(void *arg);    /**< Pointer to the stop function */
    int (*deinit)(void *arg);  /**< Pointer to the deinit function */
    int (*ioctl)(int command, int argsize, void *arg); /**< Pointer to ioctl function */
    int (*notify)(void *arg);  /**< Pointer to the notify function */
    void * (*raw)(void *arg, int *size); /**< Pointer to the raw function */
    int (*send_packet)(const char *prefix, int channelId, AVPacket *pkt, int64_t encoderPts,
        struct timeval *ptv); /**< Pointer to the send packet function: sink only */
    void * privdata;          /**< Private data of this module */
} ga_module_t;

typedef struct dpipe_s {
    int channel_id;    /**< channel id for the dpipe */
    char *name;        /**< name of the dpipe */
    //
    pthread_mutex_t cond_mutex; /**< pthread mutex for conditional signaling */
    pthread_cond_t cond;        /**< pthread condition */
    //
    pthread_mutex_t io_mutex;    /**< dpipe i/o pool operation mutex */
    dpipe_buffer_t *in;          /**< input pool: pointer to the first frame buffer in
        input pool (free frames) */
    dpipe_buffer_t *out;         /**< output pool: pointer to the first frame buffer in
        output pool (occupied frames) */
    dpipe_buffer_t *out_tail;    /**< output pool: pointer to the last frame buffer in
        output pool (occupied frames) */
    int in_count;                /**< number of unused frame buffers */
    int out_count;               /**< number of occupied frames */
} dpipe_t;

typedef struct encoder_packet_s {
    char *data;    /**< Pointer to the data buffer */
    unsigned size; /**< Size of the buffer */
    int64_t pts_int64; /**< Packet timestamp in a 64-bit integer */
    struct timeval pts_tv; /**< Packet timestamp in \a timeval structure */
    // internal data structure - do not touch
    int padding; /**< Padding area: internal used */
    unsigned char commandId; // prsc commandId
} encoder_packet_t;
```



-  ctrl_server_thread
-  vencoder_threadproc
-  liveserver_main
-  game sdl thread
-  filter_RGB2YUV_threadproc
-  read/write to common: dpipe (pipename=filter-0)
-  read/write to common: dpipe (pipename=video-0)
-  read/write to common: encoder_packet_queue

