

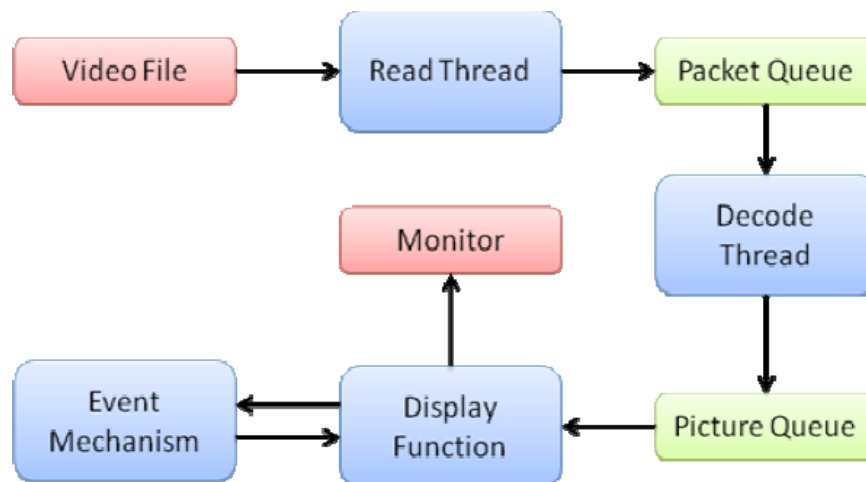
# Guidelines for MP4 Documentation

by

Hoang Nguyen (hnguyen5@illinois.edu)

Since this is your last MP, we would like you to have a more detailed documentation of your project. What follow are guidelines for documenting your MP4.

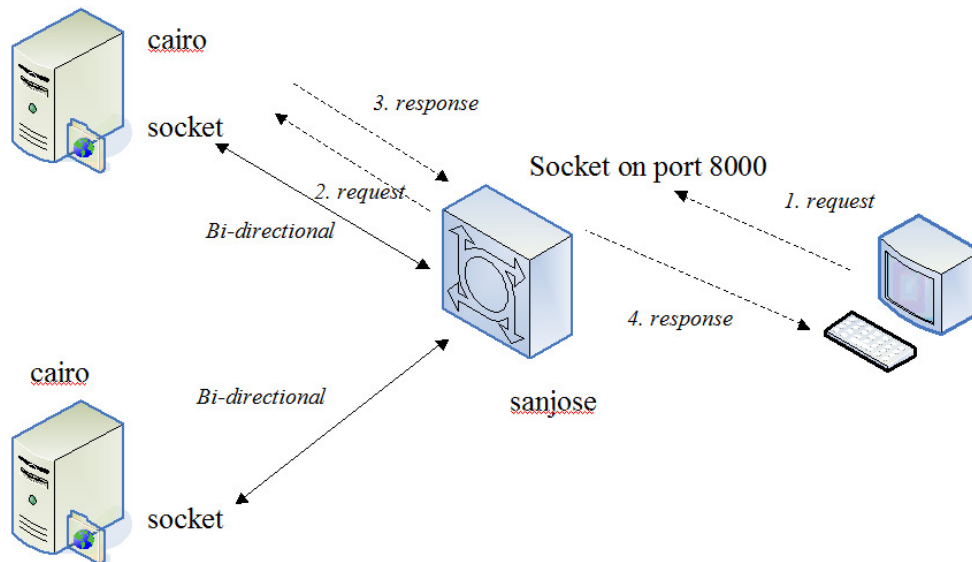
1. Using diagrams to illustrate program structure and program behaviors. Below are several guidelines.
  - a. Architecture diagram/Class diagram/Object diagram/Component diagram: Please use at least one of these diagrams to illustrate the structure of your program/system. The main purpose of these diagrams is to show classes/objects/components in the system and the relationship among them. If your project has been developed without using any classes or external components, you can draw a diagram of logical entities interacting with each other. Please draw it in the way that any future developers (including yourself!) wanting to understand your software, they can quickly capture the static structure of your program/system. Below is an illustration of a system architecture for video playback.



**Figure 1: System Architecture for Video Playback**

(Adopted from Group 5's documentation)

- b. Activity diagram/Sequence diagram: These diagrams are used to model activities/behaviors. The main use of these diagrams in this MP is to show the details of the protocols. You do not have to do this for all protocols. Please draw the important ones only. Once again, please draw the diagrams in the way that any future developers (including yourself!) wanting to understand your protocols, they can quickly capture the logic and the implementation of your protocols by looking at these diagrams.



**Handling a client request (adopted from MP2 help session)**

## 2. Important packet formats/ data structures

Please include all important packet formats and data structures used in your program. Include the details of control messages, data messages, circular buffer data structure and so on. Below is an illustration.

```
// Structure that will store shared data between the threads of the server
typedef struct {
    int controlSocket; // File descriptor of the TCP/IP socket of the control
    connection

    int port1, port2; // Ports on which the server makes the file 1/2 available

    // Parameters shared by the client and the server
    char fileName1[256], fileName2[256]; // 1 is video, 2 is audio (and is
    optional)
    int rate1; // Rate of the video (in fps)
    int duration1, duration2; // Duration to play of the files (duration2 == 0 if
    no audio)
    double audioPeriod; // Inverse of the sampling frequency
    // Buffers
    Buffer *audioBuffer, *videoBuffer;

    // ffmpeg data structures
    AVFormatContext *vFormatContext, *aFormatContext;
    AVCodecContext *vCodecContext, *aCodecContext;
    int aflow, vflow;

    // Mutexes
    pthread_mutex_t vneg_lock;
    pthread_mutex_t aneg_lock;
    pthread_mutex_t v_lock;
    pthread_mutex_t a_lock;
}
```

```
MediaList mediaList; // List of files available
} ServerData;
```

### 3. Important code snippets

Cut & paste important code snippets in your MP to the document. Important codes are those that implement key/most difficult/most challenging/trickiest techniques/algorithms. Several examples are synchronization algorithm or negotiation protocols. Below is an example of the code snippet that fixes the AVContext structure at the client side.

```
void fixCodecContext(AVCodecContext *s)
{
    // setup default pointers for AVCodecContext

    AVCodecContext temp;
    avcodec_get_context_defaults(&temp);

    s->av_class= temp.av_class;
    s->get_buffer= avcodec_default_get_buffer;
    s->release_buffer= avcodec_default_release_buffer;
    s->get_format= avcodec_default_get_format;
    s->execute= avcodec_default_execute;

    s->palctrl = NULL;
    s->reget_buffer= avcodec_default_reget_buffer;
}
```

### 4. Source code structure

Specify the structure of your source code. Directory structure, any version control information and filename hints are some basic information. It is also a good idea to have a diagram showing how these files are interacted. Below is an example.

#### 4. Important Classes:

Client.java: The client side implementation. On receipt of a stream request from the user it creates a StreamRequest object and calls AudioClient or VideoClient depending on the media type.

StreamRequest.java: Contains information about a streaming request (e.g., media type, server IP address, port, filename, function, duration, etc.)

AudioClient.java: Client side implementation of AudioClient. At first it creates a control channel with the server, negotiates the request, and on success it creates two threads: AudioQueueWriter and AudioPlayer.

(Adopted from Group 5's documentation)

### 5. Installation notes

Please include detailed instructions of how to install your software. Please do NOT assume users have any knowledge of libraries and software you use. Make sure to list all dependencies such as external libraries and softwares, where to obtain them (links, exact versions) and how to install them.

Include also possible known bugs that might happen during the installation. Specify platforms known to work fine with your software (operating system specification, sound libraries, gcc version and so on).

## 6. Execution Instructions

Please do not forget to include instructions of how to run your software! Leave important notes of what might happen during the execution.