Technische Universiteit Delft

TI3800 Bachelorproject

A non-centralized approach to Video on Demand on mobile devices

Architectural Design

Authors:
Martijn Breet
(1265458)
Jaap van Touw
(1380753)

Supervisor:
Cor-Paul Bezemer

August 27, 2013

Abstract

Contents

Chapter 1

Proposed Architecture

In this chapter the proposed prototype will be separated into different subsystems to give more insight into how to build the system. First the different subsystems will be elucidated, followed by an explanation on how the different subsystems interact with each other. The way the system stores data is clarified in section 1.2. Different threads might try to alter the same part of data which causes a concurrency issue, when this would be possible and how the team will attempt to solve this is described in section 1.3. How information flows from subsystem to subsystem in different use cases can be seen in section 1.4. Another important part of the system is how it deals with starting and stopping as well as crashes, how this will be implemented is explained in the final section.

1.1 System composition

In this section, the different subsystems are elucidated, followed by how they are combined together to form the proposed architecture of the prototype.

1.1.1 Subsystems

- **VLC**
  - LibVLCcore
    This core manages the threads, loading/unloading modules (codecs, multiplexers, demultiplexers, etc.) and all low-level control in VLC.
  - LibVLC
    On top of libVLCcore, a singleton class libVLC acts as a wrapper

class, that gives external applications access to all features of the core.

- VLC modules
  VLC comes with more than 200 modules including various decoders and filters for video and audio playback. These modules are loaded at runtime depending on the necessity. The modules communicate with the hardware directly without using the previously described LibVLC wrapper class.
- Buffer
  A buffer helps to ensure smooth playback, it pulls media data from the storage in the buffer to ready it for the VLC media player.
- VLC media player
  The media player from VLC will be the Graphical User Interface(GUI) when the user is watching a video. The GUI will be explained more in depth in the GUI subsystem.

• **Tribler**

- Core

- Libtorrent
  Libtorrent is a C++ implementation of the BitTorrent protocol, which Tribler uses to download the different pieces of a requested file. For Video-on-Demand(VoD) it will do this according to a download algorithm described by Petrocco et al[1]. The download algorithm discerns three priority tiers: high-, middle- and low-priority. The high priority section starts from the current playback position. First it downloads the pieces in this section in-order so that the user experiences continues playback. If no pieces can be downloaded from the high priority section, it will download the pieces in the mid priority section in a rarity first fashion to increase the availability of pieces in the swarm. If the middle priority pieces are also exhausted, it will download the low priority pieces in the same fashion.
- Video Player Control
  An important thing for Libtorrent is the current playback position because Libtorrent needs to get the right pieces for playback. The current playback position will be monitored by the Video Player Control.

- **GUI**
  - Start
  - VLC media player

1.1.2 Composition

In figure 1.1 a visual overview of the proposed architecture can be found. In this figure, the different subsystem are combined in to one system.

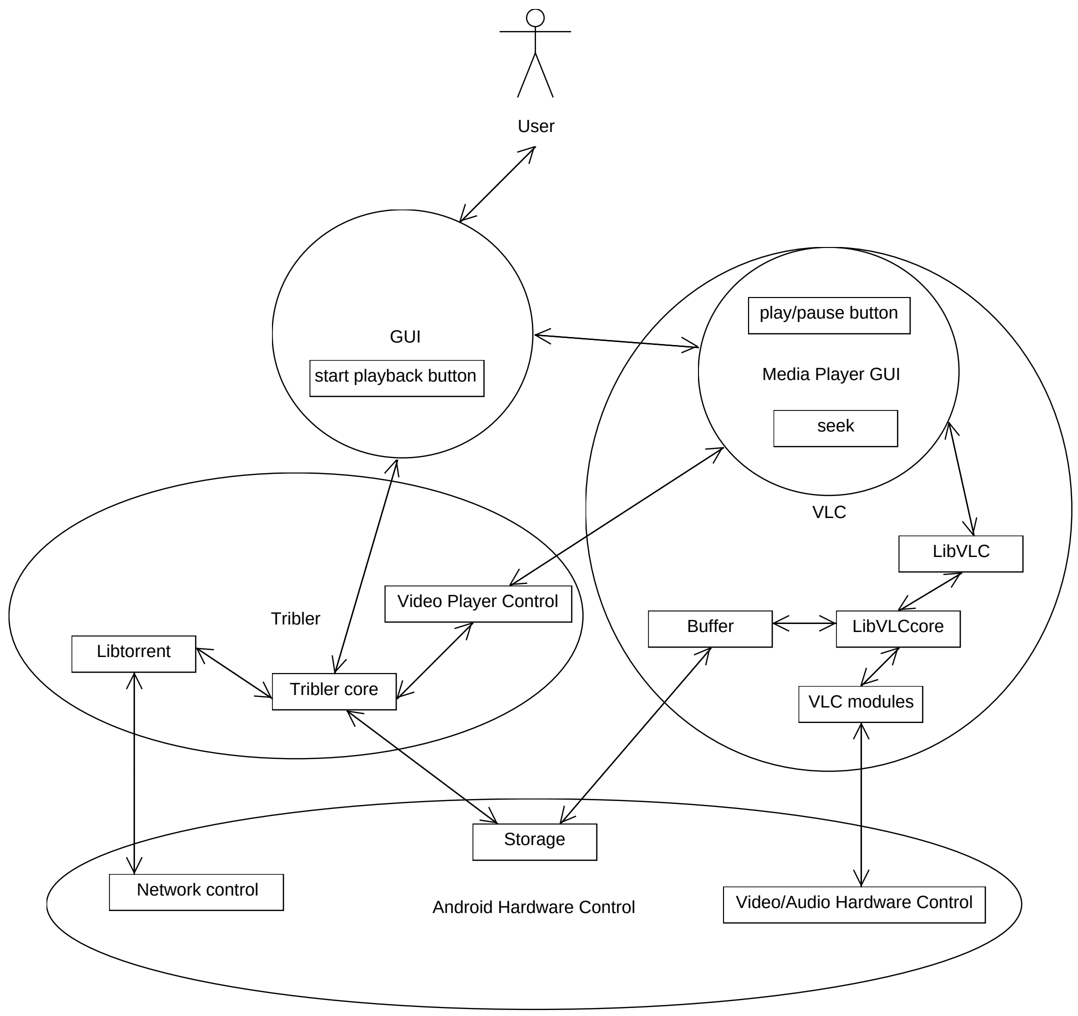

Figure 1.1: The proposed architecture of the prototype

1.2 Persistent data management

1.3 Concurrency

1.4 Software control

1.5 Boundary Conditions

Bibliography

[1] R. Petrocco, J. Pouwelse and D.H.J. Epema, *Performance Analysis of the Libswift P2P Streaming Protocol*, IEEE P2P, TU Delft, 2012.