Technische Universiteit Delft

TI3800 Bachelorproject

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

Requirements Analysis

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

Supervisor:
Cor-Paul Bezemer

August 22, 2013

Contents

1 Requirements **2**
1.1 Introduction 2
1.2 Functional requirements 2
1.2.1 Must have 2
1.2.2 Should have 3
1.2.3 Could have 3
1.2.4 Would have 3
1.3 Nonfunctional requirements 4
1.3.1 Must have 4
1.3.2 Should have 4
1.3.3 Could have 4
1.3.4 Would have 5
1.4 Constraints 5

Chapter 1

Requirements

1.1 Introduction

In this chapter, the functional-, nonfunctional requirements and constraints are elucidated. The functional requirements are requirements that pertain to the way the system should function. The nonfunctional requirements describe how the system should operate, using terms such as speed, design, user-friendliness and optimization of costs. The last section: constraints, describe the limits of the development process and proposed system. It can include constraints such as the date on which the system must be ready. The functional- and nonfunctional requirements are prioritized according to the MoSCoW method, as explained in the Plan of Action (section 3.6.2).

1.2 Functional requirements

1.2.1 Must have

The prototype must have the following features:

1. *Play the selected video;*
   When the user has selected a video from the before mentioned list, the user can then press a play button. After pressing the play button, the prototype will play the selected video.

2. *Pause the video;*
   While playing the video, the user can press a pause button which will pause the video. The video can then be resumed by pressing the play button.

3. *Seek in the video;*
   A slider will be at the bottom of the video, with which the user can set the slider to a certain part of the video. The prototype then resumes play at that part of the video.

4. *Libtorrent support;*
   The prototype will support the Libtorrent transport protocol for its P2P communication.

1.2.2 Should have

5. *Search for a video;*
   The prototype must have a search bar in which the user types the name of the video the user is looking for, then the prototype should show a number of names of videos which closely, if not fully, resemble the searched name.

6. *Download the selected video;*
   The user can choose to download a video by means of pressing a download button so the user can watch it later instead of watching it immediately.

7. *Single click installer;*
   The prototype application should have a single click installer, meaning that all components and dependencies should be intergrated into one single APK installation package.

8. *Support as many multimedia codecs and file formats as Tribler;*
   The prototype application should support as many multimedia codecs and file formats as the desktop version of Tribler.

1.2.3 Could have

9. *Dispersy support;*
   The prototype application could support Dispersy to spread data bundles over the internet in a fully decentralized way, as to facilitate the creation of 'channels' of bundled torrents.

10. *Browse through 'channels' for videos;*
    The desktop version of tribler has channels that allows users to browse through a collection of videos, a similar feature could be implemented in the prototype.

1.2.4 Would have

11. *Libswift support;*
    The prototype will support the Libswift transport protocol for its P2P communication.

12. *Anonymous tunneling/Subset of Tor protocol;*
    The prototype will support anonymous tunneling or a subset of the Tor protocol in order to facilitate anonymous data traffic.

13. *Seed a video;*
    While watching and downloading the video is only leeching, also seeding the already downloaded pieces of the video would increase the availability of pieces in the swarm.

14. *Upload a video;*
    The user can upload videos from his own gallery.

15. *Make the application available for other mobile platforms;*
    The prototype will only be made on the mobile platform: Android. It would however, be good to branch out to other mobile platforms such as iOS and Windows mobile in order to attract more users.r

1.3 Nonfunctional requirements

1.3.1 Must have

1. *The prototype must not introduce much extra lag on top of the start-up time before playing a video, in comparison to the desktop version of Tribler;*
   The time between pressing the play button and the video actually starting is decided by how fast the system gets the pieces it needs for continuous playback. In Tribler this strongly depends on the connection speed and how many seeders exist in the swarm. It can range from about eight seconds to three minutes. The prototype must not significantly increase this time.

2. *The prototype must be fully non-centralized;* No central servers can be used to facilitate the downloads.

1.3.2 Should have

3. *The playback should look smooth, no visible lag should occur;*
   Sometimes the playback can stall because it has not yet received the pieces needed for playback, it should not however, stutter.

1.3.3 Could have

4. *The prototype could have low power consumption in comparison to other VoD applications such as Youtube;*
   The power consumption of Youtube[1] and a simple implementation of the Libswift[2] protocol has been measured in [1]. A same set-up as explained in the paper can be achieved for the prototype, after which optimization of the prototype could lead to lower power consumption.

1.3.4 Would have

5. *Optimize the start-up time;*

1.4 Constraints

1. *The prototype must be ready before the second of October 2013*;

[1] http://www.youtube.com/
[2] http://libswift.org/

Bibliography

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