Technische Universiteit Delft

TI3800 Bachelorproject

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

Final Report

Author:
Jaap van Touw
(1380753)

Supervisor:
Cor-Paul Bezemer MSc

September 24, 2013

Abstract

As mobile Internet traffic continues to consistently gain on desktop traffic in terms of volume, a desire for a mobile version of Tribler was expressed by the Client: Dr. Ir. Johan Pouwelse. This desire was formulated into a Bachelor project, which will be executed within a period of 11 weeks. The project is centered around answering the following research question: *"How can we make video-on-demand available for mobile devices using a non-centralized approach?"*. The goal of the project is to make a prototype of a mobile application for Android that features Video on Demand through a Peer-to-Peer network. During the design- and implementation phase of the project, several methods are used, such as Scrum, acceptance testing and MoSCoW prioritization. Additionally, Git is used to keep track of the history of all documents and source code. During the implementation phase, Libtorrent was implemented together with VLC to create the prototype. This prototype can stream videos on Android in a non-centralized way, and therefore answering the main research question. The application meets the 'must have' requirements as well as some of the should have requirements, which were elicited in the requirements.

Preface

This document entails the work of a Computer Science Bachelor project executed at the Delft Technical University within the Parallel and Distributed Systems group. During the different phases of this project I was helped by a lot of people and would like to offer my gratitude to the following people in particular:

Cor-Paul Bezemer, for his expert supervision, guidance and input.

Johan Pouwelse, for his positive support, input and visionary influence.

Jan-Willem van Velzen, for his fortunate mistake, partnership and his tireless effort in making a good application.

Egbert Bouman, for his sublime coding tip at a most fortunate time.

Contents

Part I

Introduction

The Parallel and Distributed Systems group[1] (hereafter: PDS) is a research group within the Software and Computer Technology[2] (hereafter: SCT) department, which is part of the Faculty Electrical Engineering, Mathematics and Computer Science[3] (hereafter: EEMCS) of the Delft University of Technology[4]. The research within the PDS group concentrates on the modeling, the design, the implementation, and the analysis of parallel and distributed systems and algorithms. Most of this research is experimental: the aim is to build prototypes of systems, preferably used in the real world, to demonstrate the quality of the proposed solutions. The main research areas of the PDS group are Peer-to-Peer (hereafter: P2P) systems and online social networks, massive multi-player online games, grids, clouds, multi-core architectures and parallel programming. P2P is considered by many as an efficient, reliable, and low cost mechanism for distributing any media file or live stream, and it is used extensively [1]. Much of the current research activities in P2P within the PDS group are centered around Tribler[5]. Tribler is an application that enables its users to find, consume and share content through a P2P network. Tribler builds on BitTorrent[6] and is available on desktop environments. Currently mobile internet traffic continues to consistently gain on desktop traffic in terms of volume[7] and mobile traffic is estimated to surpass traffic from wired devices in 2017[8]. In response to this growth and to meet the increasing demands of the market, the development of a mobile version of Tribler would be of great value.

First, the plan of action is described in which the project background, assignment, approach and design are described, as well as how the quality of the product will pursued. This is followed by the orientation phase, in which different tools, methodologies and risks are researched. Part IV describes the requirements which were set up together with the Client. The next part is about the test and implementation plan, which was set up to guide the developers towards a working prototype. How the system is designed beforehand is explained in Part VI. Part VII is a chronological report about how the process of the implementation phase developed. The final part includes the conclusion, what future work lies ahead and a personal reflection.

[1] http://www.pds.ewi.tudelft.nl/
[2] http://softechno.ewi.tudelft.nl/
[3] http://www.ewi.tudelft.nl/en/
[4] http://www.tudelft.nl/en/
[5] http://www.tribler.org
[6] http://www.bittorrent.com
[7] http://gs.statcounter.com/mobile_vs_desktopwwmonthly200812201306
[8] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.pdf

Part II

Plan of Action

Chapter 1

Introduction

This part describes the goal of the proposed Bachelor project and elaborates on how this goal will be accomplished, as well as the time frame in which it needs to be completed. The structure is as follows. The remaining part of this chapter will focus on preliminary background information that will explain the context in which the proposed Bachelor project will be performed. This will include a brief description of the research department and a background information on the topic of the project, as well as the causes that led to the project assignment. Chapter 2 will give a detailed description of the project. The project stakeholders are identified, as well as the main research question that the project will focus on answering. Additionally, a list of all the deliverables for the project will be covered. The last part of the chapter will cover the general requirements and constraints of the project, as well as the conditions under which the project will be performed. Chapter 3 will focus on the approach that will be used to answer the research question and will elucidate the project planning. The administrative part of the project will be covered in Chapter 4. Finally, Chapter 5 will focus on the measures that will be taken to assure the quality of the project.

1.1 Department description

The Parallel and Distributed Systems group is a research group within the Software and Computer Technology department, which is part of the Faculty Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. The research within the PDS group concentrates on the modeling, the design, the implementation, and the analysis of parallel and distributed systems and algorithms. Most of this research is experimental: the aim is to build prototypes of systems, preferably used in the real world, to demonstrate the quality of the proposed solutions. The main research areas of the PDS group are Peer-to-Peer systems and online social networks, massive multi-player online games, grids, clouds, multi-core architectures and parallel programming.

1.2 Background and Motivation

Distribution of radio and television programs, movies, music, ring-tones, games, and various data applications to the general public is possible today via a variety

of dedicated networks and special end-user terminals. As broadband Internet becomes ubiquitous in both desktop and mobile device environments, all content distribution services will be combined and conveyed to the general public via a common pipeline, the Internet. Today several technologies are used for the media distribution across the Internet: unicast, IP multi-cast, content distribution networks, and most recently Peer-to-Peer. P2P is considered by many as an efficient, reliable, and low cost mechanism for distributing any media file or live stream, and it is used extensively [1]. Much of the current research activities in P2P within the PDS group are centered around Tribler[1]. Tribler is an application that enables its users to find, consume and share content through a P2P network. Tribler builds on BitTorrent[2] and is available on desktop environments. Currently mobile Internet traffic continues to consistently gain on desktop traffic in terms of volume[3] and mobile traffic is estimated to surpass traffic from wired devices in 2017[4]. In response to this growth and to meet the increasing demands of the market, the development of a mobile version of Tribler would be of great value.

[1] http://www.tribler.org
[2] http://www.bittorrent.com
[3] http://gs.statcounter.com/mobile_vs_desktopwwmonthly200812201306
[4] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.pdf

Chapter 2

Project Assignment

In this chapter, key aspects of the project are set out such as the main research question that the project will be focused on answering, the goal of the project, as well as an exact formulation of the project assignment. General requirements and constraints for the project are also given here.

2.1 Client

The client is Dr. Ir. Johan Pouwelse, he is Assistant Professor at the Parallel and Distributed Systems Group of the Faculty of EEMCS, Delft University of Technology, and is also co-founder of Tribler. Moreover, he is Scientific director of several P2P research initiatives with a total budget of 26 Million Euro.

2.2 Stakeholders

The Client:

Dr. Ir. Johan Pouwelse
J.A.Pouwelse@tudelft.nl
+31 (0)15 27 82539
Room: HB 07.290
Mekelweg 4
2628 CD Delft

Project Supervisor:

Cor-Paul Bezemer, MSc.
C.Bezemer@tudelft.nl
+31 (0)15 27 82467
Mekelweg 4
2628 CD Delft

Bachelor Project Coordinator:

Dr. Martha A. Larson

M.A.Larson@tudelft.nl
+31 (0)15 27 87357
Room: HB 11.040
Mekelweg 4
2628 CD Delft

2.3 The Research question

The project will be focused on answering the following main research question:

"How can we make video-on-demand available for mobile devices using a non-centralized approach?"

In the Orientation Report, this question will be discussed in more detail.

2.4 Goal

The goal of the project will be to create a prototype application for mobile devices, that allows users to enjoy video-on-demand(VoD) through the Internet, using a non-centralized network architecture. The application will allow users to search for a video, which will start playing after the user presses a play button. Additionally, seeking functionality will be provided.

2.5 Assignment Formulation

In this project, a prototype of the P2P-based video-on-demand mobile application will be realized, that will allow users to search for videos on the Internet. Once the user has found a video to his or her liking, and issued a play command, the application will play the video by means of streaming it through a P2P-based network. The application will also feature the possibility to seek to different parts of a video, by means of a slider, that can be adjusted by the user.

2.6 Deliverables

During and after the completion of the project, the following products will be delivered:

1. Orientation Report.
2. Requirements Analysis Document
3. Architectural Design Document
4. Technical Design Document
5. Test- and Implementation Plan
6. Implementation

7. Source code evaluation by SIG[1]

8. Final report

The deliverables above are further defined in section 3.3.

2.7 Requirements and constraints

The final product will offer the features that are described in section 2.6 and should be considered a "prototype". The prototype will offer video-on-demand, including built-in search functionality. The development of the prototype will be targeted to the Android platform.

The exact functional- and nonfunctional requirements, as well as the constraints will be specified in the Requirements Analysis Document.

2.8 Conditions

The project members listed under Section 4.1 will deliver the products listed in Section 2.6 within the period starting from the 17th of July until the 2nd of October 2013.

The client will facilitate the project members by supplying them with all the resources needed for the development of the deliverables, such as (mobile) testing devices and required office space.

The project members will have weekly meetings with the supervisor (see Section 4.3) to discuss the status and progression of the project, as well as to receive feedback on completed work. These weekly meetings are also part of the development methodology which will be described in Section 3.5.

[1] http://www.sig.eu

Chapter 3

Approach

In this chapter insight is given into the approach that will be used to complete the project, including a brief description for every project phase and its deliverables. This chapter will also focus on the methodology that will be used during the project. Finally, the project planning will be elucidated.

3.1 Orientation Phase

In this phase, the research question is further crystallized into subquestions, which will create a better understanding of the problem as well its scope. After this is clarified, research will be conducted, by exploring existing (partial) solutions that can contribute towards a general solution for the problem. The findings of this research will be stated in the Orientation Report.

3.2 Design Phase

As a first step in the design phase, functional-, nonfunctional requirements and constraints for the prototype will be elicited. From these requirements, use-cases will be derived that convey how the system should interact with the user. Based on these requirements and use cases, an architectural design document will be created, consisting of a description of the proposed software architecture including subsystem decomposition, persistent data management, global resource handling, concurrency, software control and boundary conditions. Next, a detailed description of the packages, class diagram and a specification of the classes and methods is given in the Technical Design Document. Finally, a Test- and Implementation Plan is created that describes how the different features of the prototype will be tested and implemented.

3.3 Implementation Phase

Building the application is the main part of the project and consists of implementing the different components that were derived in the Design Phase. The source code of the prototype will be sent to the SIG for a thorough review on the

5th of September. The feedback provided by the SIG will be used to improve the code during the implementation phase.

3.4 Release Phase

In this final stage of the project, all the documents that were created during the previous phases of the project will be bundled into one final report. This will include a general conclusion and evaluation. After handing in the Final Report to the stakeholders, the source code of the prototype is sent to the SIG one more time for a final evaluation. The finished prototype will then be presented to the client, bachelor coordinator and supervisor.

3.5 Methodology

3.5.1 Scrum

During the project several methods will be put into practice. One of these methods, namely Scrum[1], will be used in all phases starting from the design phase. Scrum is an iterative and incremental Agile method. Scrum uses sprints in which the team goes through the process of adding functionality to the software, always maintaining a working version of the prototype. Given the relatively small timespan of the project, sprints of one week are chosen to ensure progress is measured frequently. Traditionally, the Scrum method defines a number of roles assigning different responsibilities to each of the team members. Since the team for this project solely consists of two persons, no specific roles are assigned. At the start of each sprint, a meeting; called a sprint planning, will be held. These meetings will be attended by the team members, as well as the supervisor. In this manner, the supervisor gets a better insight into what progress is made and is able to provide more meaningful feedback on the previous and upcoming tasks. In the sprint planning the following is discussed:

- Which tasks have been completed during the last sprint.
- Encountered impediments, if any.
- Decide on which tasks have to be done in the upcoming sprint.
- Determine the time it will take to complete the tasks and assign these to the team members.

The daily scrum meetings, also known as 'standups', will only be attended by the team itself. During these meetings, the team will briefly discuss what each person did on the day before, what each person is going to do and if there are any impediments that need to be overcome. Furthermore, bi-weekly demo sessions will be held with the client, to keep the client informed on the progression that is made.

[1] http://www.scrum.org

3.5.2 MoSCoW

The MoSCoW method was first developed by by Clegg et al.[2], and it has become a standard in prioritizing a list of requirements. The following categories are defined:

- Must have: requirements that must be satisfied in the final solution for the solution to be considered a success.
- Should have: high-priority requirements that should be included in the solution if possible.
- Could have: requirements which are considered desirable but not necessary.
- Would have: requirements that will not be implemented in a given release, but may be considered for the future.

The MoSCoW method will be used to prioritize the elicited requirements in the Requirements Analysis Document.

3.5.3 Acceptance Testing

Acceptance testing means that the team will see if the software in the current state meets the requirements by manually testing the functionality.

3.6 Planning

A Gantt chart of the project planning can be found in Appendix A. The chart is complemented with a timeline, for a quick overview.

Chapter 4

Project Design

This chapter will cover the administrative aspects of the project, such as the project members, reporting, financing and facilities.

4.1 Project members

The project members that will work on the project are Martijn Breet and Jaap van Touw. Both members are required to work at least 40 hours per week on the project. In addition to that, both members are required to make an equivalent contribution during every phase of the project, in order for both members to gain the same amount of experience with all types of activity (requirements analysis, design, implementation, etc.). How the work is divided, is included in Appendix E. A short introduction to each of the project members is given below, in which the previous and current activities of each member is briefly described, together with their current contact information.

Jaap van Touw
Jaap is a Computer Science bachelor student at the faculty of EEMCS. His main programming language is Java and before starting this project he worked at the DUT Racing Team as software engineer.

J.vanTouw@tudelft.nl
+31 (0)6 505 37 951
Room: HB 07.240
Mekelweg 4
2628 CD Delft

Martijn Breet[1]
Martijn is a Computer Science bachelor student at the faculty of EEMCS and has over 4 years of working experience in the field of software engineering. Before starting this project, he spent one year at the Delta Lloyd Solar Boat Team as full-time board member and software engineer.

[1] Martijn left the project before the implementation phase, see 16.1

M.S.Breet@student.tudelft.nl
+31 (0)6 158 86 822
Room: HB 07.240
Mekelweg 4
2628 CD Delft

4.2 Financing

All work within the project will be done on a voluntary basis. No specific budget is available for the project, and all the necessary work has to be be conducted using existing resources (see section 4.5).

4.3 Project Reporting

In order to keep the client up to date, both oral and written communication will be utilized. Informal meetings with the client will be held in person on a weekly basis. Written communication will be performed by means of using an IRC[2] channel , as well as per e-mail. Apart from the client, communication with the supervisor will also be done orally and in writing. Weekly sprint meetings will be held with the supervisor. The minutes of these meetings can be found in Appendix D. All project documentation will be written using the LaTeX[3] typesetting system, and will be released as LaTeX source code, as well as in PDF[4] format. All released project material, including source code and documentation will be uploaded to an online GitHub[5] repository which is publicly available at all times.

4.4 Resources

The project members each already own a laptop, which they will use for all the required software development and document creation. The client has arranged an office room at the department of the PDS group where the project members can work full-time. All the software that is required to perform the project tasks is open-source and freely distributed over the Internet. In case additional hard- or software is required during the project, the project members will ask the client for the required funding of these procurements.

[2] http://www.irchelp.org/
[3] http://www.latex-project.org/
[4] http://www.adobe.com/products/acrobat/adobepdf.html
[5] https://github.com/javto/bsc-project

Chapter 5

Quality Assurance

This chapter will focus on how different techniques and methods will be used to assure the quality of the final product and help improve the Bachelor project itself.

5.1 Quality

5.1.1 Documentation

During the overall process, several documents (see Section 2.6) will be created in which design choices are made and justified. These documents will be reviewed by both the supervisor and the client, after which the feedback will be processed by the project team.

5.1.2 Version Control

An existing version control system will be used during all phases of the project to keep track of the history of every created document and source code. One of the benefits of using a version control system is that in case an undesirable change is made to a document, one can revert back to any previous version of the document. The system of choice for this project will be Git[1], a widely used open-source distributed version control system. It is a system the team is already proficient with and also what the Tribler-team is using at the moment.

5.1.3 Code Review

During the implementation phase, the source code that is written by each project member will be briefly reviewed by the other member. This way, feedback can be obtained that can help improve the overall quality of the source code.
A complete source code review will be done by the SIG (see section 3.3), in which the quality of the software is professionally evaluated. By processing their feedback and recommendations, the project members can improve the quality of the software.

[1] http://git-scm.com/

5.1.4 Software Testing

During the implementation phase we will use acceptance testing as software testing method, as explained in Section 3.5.3.

5.1.5 Evaluation

After the project, an evaluation will be written by the project team to reflect on how the process went. In this reflection, focus is put on how the overall process, as well as the approach of each team member can be improved. Additional feedback on the bachelor project itself will also be provided. The evaluation does not necessarily increase the quality of the final product itself, but it does increase the quality of future work of the project members, the bachelor project and the final product in the long term by showing what more can be done to expand or improve the final product.

Part III

Orientation

Chapter 6

Introduction

Many of the current research activities in peer-to-peer within the Parallel and Distributed Systems group of the Delft University of Technology are centered around Tribler[1]. Tribler is an application that enables its users to find, consume and share content in a fully decentralized way so that no central server is needed. Another popular feature of Tribler is video-on-demand where users can immediately watch the video they are downloading. Currently, mobile internet traffic continues to consistently gain on desktop traffic in terms of volume[2] and mobile traffic is estimated to surpass traffic from wired devices in 2017[3]. In response to this growth and to meet the increasing demands of the market, the development of a mobile version of the VoD feature in Tribler would be of great value.
In this section, the problem statement and the research questions are presented.

6.1 Motivation

Tribler features VoD on desktop environments, with which people can search for a video and immediately play it. Similar features do exist on mobile devices, for example, YouTube[4] and ITV Player[5] but they all require central servers in order to watch the videos presented. A non-centralized approach ensures that the application does not depend on one central unit which could go offline, for example due to disasters or political intervention [6]. Applications that feature VoD are immensely popular. YouTube alone has one billion unique users who watch over six billion hours of video each month[6]. Combining VoD, non-centralization and the current market shift towards mobile devices is something to which, currently, no solution exists.

[1] http://www.tribler.org
[2] http://gs.statcounter.com/mobile_vs_desktop-ww-monthly-200812-201306
[3] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.pdf
[4] https://play.google.com/store/apps/details?id=com.google.android.youtube
[5] https://play.google.com/store/apps/details?id=air.ITVMobilePlayer
[6] http://www.youtube.com/yt/press/statistics.html

6.2 Problem Statement

The goal of our research is to come towards such a solution. The following research question is therefore central for this research:

"How can we make video-on-demand available for mobile devices using a non-centralized approach?"

This research question consists out of three main elements: VoD, mobile devices and a non-centralized approach.

VoD is the main feature that needs to be considered for implementation. Ideally, we can use an existing tool or library for VoD on mobile devices. In the first part of our research, we will focus on the following research question:

Research Question 1: *"Which solutions exist for VoD on (non-)mobile devices"*

If no solutions exist for mobile devices, there may be solutions for non-mobile devices which can be ported to those mobile devices. This leads to the following sub question:

(a) *"How can a port of an existing non-mobile solution be made to mobile devices?"*

Important to realize is, that a port of a solution might not have full functionality because of hardware issues or other incompatibilities. Therefore research has to be done into the following:

(b) *"What are the limitations of such a ported solution to mobile devices?"*

Interesting to see is which of these solutions, use a non-centralized approach. Non-centralization is key to getting a low-cost, highly reliable and community driven solution (see Section: 8.3.1). Therefore, in the second part of our research, the focus will be put on:

Research Question 2: *"Which solutions for VoD use a non-centralized approach?"*

There are an abundant number of different mobile devices, ranging from different sizes and processors to different platforms. Each different platform, such Apple's iOS, Android and Windows phone, has its advantages and disadvantages. These pros and cons have to be researched in order to see which platforms are best suitable for which solutions, hence, the following research question:

Research Question 3: *"Which mobile platforms are best suited to implement VoD in a non-centralized approach?"*

Videos are encoded to save space and bandwidth once transmitted. To decode these videos, decoding software has to be used. This software depends on the type of hardware it runs on and has to be included or linked to in order to play the video, research into this matter will be led by the following sub question:

Research Question 4: *"Which solutions exist for video decoding on the chosen platform?"*

The different mobile platforms are described in Chapter 7, covering Research Question 3. Research Question 1, about (non-)mobile VOD solutions, will be discussed in Chapter 8. In this same chapter, the non-centralization aspect (Research Question 2) as well as how a port can be made to mobile devices (Research Question 1(a)) are described. The limitations of these ports are also described in this section (Research Question 1(b)). Video decoding is described in Chapter 9. Finally an analysis of the risks involved is included in Chapter 10, which describes what to do when a situation arises that compromises a solution or part of it.

Chapter 7

Mobile Platforms

In this chapter, a brief overview of three different mobile platforms will be given. The mobile platforms in this chapter are chosen due to popularity and which run on current generation smart phones. The three mobile platforms are: Windows Phone[1], Apple iOS[2] and Google Android[3]. The findings are presented in Section 7.1.

7.1 Trade-off

The trade-off will be made according to several criteria. Table 7.1 is provided in which a number of criteria are set out. The market share of the different mobile platforms will be described in Section 7.1.1, followed by the proficiency of the team in relation to the different programming languages. Finally, a conclusion to which mobile platform is chosen for the prototype is provided.

7.1.1 Market share

In the second quarter of 2013, research was conducted by Strategy Analytics[4] in to the global market share of the different mobile platforms. The result is that Android holds a 79.5% market share, iOS has a 13.6% share and Microsoft has a share of 3.9%[5]. Android has the biggest share and this means that an application in Android will most likely have a larger user base than its iOS or Windows Phone counterpart.

7.1.2 Programming language

The team is proficient with C, C++, C#, Java and script languages such as Javascript, XML, PHP, etc. They also have some experience with making applications for Android. Further more they do not own a Mac computer or iPhone.

[1] http://www.windowsphone.com/
[2] http://www.apple.com/iphone/ios/
[3] http://www.android.com/
[4] http://www.strategyanalytics.com/
[5] http://news.cnet.com/8301-1035_3-57596548-94/android-nabs-record-80-percent-market-share-in-q2/

7.1.3 Conclusion

Concluding, Android is chosen as mobile platform because it has the largest market share and the team is more capable of developing applications for Android than for any other mobile platform.

Table 7.1: A trade off between three mobile platforms

	Windows Phone	Apple iOS	Google Android
Number of Applications Available	160.000[6]	900.000[7]	800.000[8]
Number of Total Application Downloads	? (200 million per month[9])	50 billion total[10]	48 billion total[11]
Number of Different Tablets and Phones	27[12,13]	19[14]	3997+[15]
Application Development Languages	C#[16], VB.NET[17], C++[18], Javascript[19], F#[20]	Objective-C[21].	Java[22], XML[23], C/C++
Market Share[24]	3.9%	13.6%	79.5%

[6] http://thenextweb.com/microsoft/2013/06/27/microsoft-windows-phones-160000-apps-see-more-than-200-million-downloads-monthly/

[7] http://www.apple.com/pr/library/2013/06/10Apple-Unveils-iOS-7.html

[8] http://www.appbrain.com/stats/number-of-android-apps

[9] via footnote 6

[10] http://ben-evans.com/benedictevans/2013/5/16/how-many-apps-do-android-and-ios-users-download

[11] via footnote 10

[12] http://www.windowsphone.com/en-us/phones

[13] http://reviews.cnet.com/8301-3121_7-57531284-220/windows-8-the-complete-new-pc-launch-list/

[14] 8 iPhone models, 5, iPod Touch models, 4 iPad models, iPad Mini and Apple TV

[15] http://techcrunch.com/2012/05/15/3997-models-android-fragmentation-as-seen-by-the-developers-of-opensignalmaps/

[16] http://www.microsoft.com/en-us/download/details.aspx?id=7029

[17] http://msdn.microsoft.com/en-us/vstudio/hh388573

[18] http://isocpp.org/

[19] https://developer.mozilla.org/en-US/docs/Web/JavaScript?redirectlocale=en-U0\&redirectslug=JavaScript

[20] http://fsharp.org/

[21] http://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

[22] http://www.java.com/

[23] http://www.w3.org/TR/REC-xml/

[24] http://news.cnet.com/8301-1035_3-57596548-94/android-nabs-record-80-percent-market-share-in-q2/

Chapter 8

Video on Demand on (non-)Mobile Devices

As mentioned in the motivation, Video-on-Demand(VoD) is a widely popular feature. A lot of different services exist to provide this feature, such as Youtube, ITV[1], Netflix[2] and Amazon Instant Video[3]. Another VoD service is Tribler [4], developed at the TU Delft by the Parallel and Distributed Systems(PDS) group. In essence, it is a BitTorrent[4] client featuring VoD. In this chapter the previously stated possibilities for VoD will be described. First, a description of the first four mentioned VoD services will be given. After this, Tribler will be presented. This will be followed by a trade-off in which the services will be compared to find the most suitable service for Android. Finally a conclusion of this trade-off is included.

8.1 Video on Demand services

In this section, the most popular VoD services are discussed:

8.1.1 ITV Player

ITV Player is a VoD service which shows the programs broadcast on the TV-cable channels ITV1, ITV2, ITV3, ITV4 and CITV. It provides playback for shows up to 7 days after they aired on TV and after a free registration this becomes 30 days. The programs shown are therefore limited to those that were aired; ranging from TV-shows to live shows such as X-factor. The ITV Player runs on its website[5], desktop environments (Mac, Windows and Linux), PlayStation 3, iOS and on Android[6]. The ITV player does show advertisements before playback of every chosen video.

[1] http://www.itv.com/
[2] http://www.netflix.com
[3] http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000663511
[4] http://www.bittorrent.com/
[5] http://www.itv.com
[6] https://www.itv.com/itvplayer/help/faq/1

8.1.2 Amazon Instant Video

Amazon has a VoD service in which the user pays a yearly fee($79), after which they can watch a lot of content for free, some releases may not be free and an additional price must be paid to buy or rent the item. The application with which to watch the videos is available for: Windows, Kindle Fire (HD), Apple devices, PlayStation 3, Xbox 360, Wii (U), a number of set-top boxes[7] and the so called: 'smart TVs'[8].

8.1.3 YouTube

YouTube started in 2005 and has quickly become the most popular VoD service. As said in section 6.1, one billion unique users visit YouTube every month, adding up to 6 billion hours of videos watched each month. The service is available for desktop environments, Android, iOS, Windows Phone, game consoles and smart TVs. The videos available for playback are all added by users themselves. Anyone can upload a video, for example, companies that want to add a video of their latest product or news stations covering the latest events, but also parents can upload the footage of their baby's first steps. However, there are limits to what is allowed, which is included in their policy[9].

8.1.4 Netflix

Netflix is a streaming VoD service that allows users to watch an unlimited amount of videos after paying a monthly fee ($7.99 per month, at the time of writing). It includes movies and TV programs and is available only in the United States, Canada, Latin America, the United Kingdom and Ireland. As of July 2013 it has 37.6 million users in total[10]. Netflix is available on Windows, Mac, Android, iOS, Windows Phone, set-top boxes, smart TVs and game consoles[11].

8.2 Tribler

Tribler is an application that enables its users to find, consume and share content through a peer-to-peer (P2P) network. The application is currently available for Windows, Mac and Linux. Tribler facilitates finding and sharing content in a fully decentralized way, which means that no central server is needed. Furthermore, to tackle the issue of people who only download(*leech*) content, Tribler makes use of a reputation system which encourages users to actively participate in uploading (*seeding*) content. Tribler also includes a VoD service which allows users to stream videos.
Figure 8.1 depicts the four components of Tribler:

- GUI: the graphical user interface.
- BTengine: An implementation of the BitTorrent protocol.

[7] http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000663511
[8] http://www.businessinsider.com/what-is-a-smart-tv-2010-12
[9] https://www.youtube.com/yt/policyandsafety/policy.html
[10] http://news.yahoo.com/numbers-netflix-subscribers-205626746.html
[11] http://www.netflix.com/NetflixReadyDevices
[12] http://sigmm.org/records/records1201/featured03.html

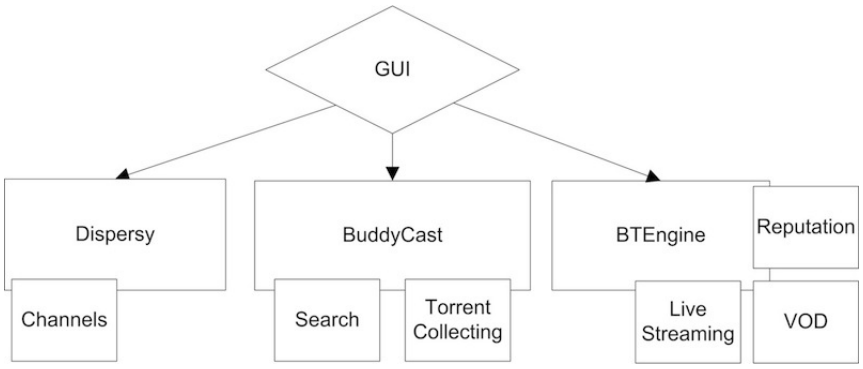


Figure 8.1: The architecture of Tribler[12]

- BuddyCast: a protocol used to find peers with the same taste as the user[4]
- Dispersy: a fully decentralized system for data bundle synchronization[5].

8.2.1 BTengine

BitTorrent is a protocol which allows for P2P file sharing. The protocol allows users to join a swarm of hosts to download and upload any file. For a user to share a file it can create a torrent descriptor file which contains information about the file. The torrent can then be distributed over the Internet via e-mail, a link on a website, etc. Other users with the torrent can connect to this host, called a seeder, and ask for pieces of this file. After all pieces are collected, the leecher becomes a seeder and other leechers can download from the new seeder. This way, the files are distributed over the Internet without needing any central server.
The BTengine in Tribler includes a reputation system, where the user is rated for their upload to download ratio. This helps to minimize the effects of free riding, where users only download, because the user with a low ratio will be given lower speed peers to connect to. As of Tribler 6.1, The BTengine uses an implementation of BitTorrent called: Libtorrent.
Libtorrent is a C++ implementation of BitTorrent for embedded devices as well as desktops[13].The interface is well documented and is designed to be CPU- and memory-efficient as well as easy to use.

8.2.2 BuddyCast

The BuddyCast protocol is used to find peers with similar taste, so Tribler can give recommendations to what the user might like and thus discover new content.

8.2.3 Dispersy

Dispersy is used to spread data bundles over the Internet in a fully decentralized way. This could potentially remove the need for central servers for services

[13]http://www.rasterbar.com/products/libtorrent/

such as Facebook or Wikipedia. In Tribler it is used for creating the so called 'Channels'. Each channel has different torrents bundled together to form a sort of play list about one topic such a single genre or the new 2013 releases. This makes the discovery of new files that the user might like easier.

8.2.4 VoD via P2P

By default, pieces of a file are downloaded in a rarity first fashion in the BitTorrent protocol. Tribler implements VoD via P2P by downloading the pieces in the following manner[3]: 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.

8.3 Trade-off

In this section the different criteria for what makes up a good VoD service are elucidated. Also described in this section is in how the different services perform on these criteria.

8.3.1 Non-centralization of servers

When no central servers are used, the service has low hardware maintenance costs. Furthermore in a central server way, if the server fails, data might be lost and the service is out of order. This can not happen in a non-centralized approach with enough users, making the non-centralized approach highly reliable. This also means that every user is part of the solution, making it a community driven approach, where every user has influence on the system(in terms of availability and range of content), the size of this influence depends on the number of users connected. Of the VoD services that were previously described, only Tribler currently uses a non-centralized approach. To implement non-centralization into one of the other services would be beyond the scope of this project due to the size and state of maturity of the those services and therefore the time it would take to switch to non-centralization.

8.3.2 Portability

As said in Chapter 7, Android is the platform of choice to provide the VoD functionality. Tribler and Amazon Instant Video are not yet available for this platform and must be ported to Android to make VoD possible. Amazon Instant Video does have an application on Android[14], but it works only on select devices and the ratings are quite low due to incompatibility. Since Amazon Instant

[14]https://play.google.com/store/apps/details?id=com.amazon.avod

Video is a closed-source project, porting it is nearly impossible. Tribler is open-source, but written in Python. There is a project on GitHub[15] that provides the functionality of running Python code on Android, but does this without being able to properly debug the code. Also, how video playback works on desktop environments is different than on Android, Chapter 9 will go into more detail about this.

8.4 Conclusion

In conclusion, Tribler is chosen as the VoD service to implement on Android. Although it is not yet available for Android, it can be ported and it uses a non-centralized approach, is free of costs to the user, has a large amount of videos available in HD and shows no advertisements.

[15] https://github.com/d3vgru/python-for-android/tree/tgs-android

Chapter 9

Video Decoding Frameworks

This chapter describes the three open-source video decoding frameworks that exist for the Android Platform, available on the Google Play Store. The architecture of each of the frameworks is briefly described, as well as the built-in features. The dependencies of each framework are also briefly analyzed. Finally, a trade-off will be made between these frameworks upon which a framework will be chosen for use in the project.

9.1 VLC for Android Beta

VLC for Android Beta is a popular open-source multimedia player and framework[1], with over 10.000.000 downloads. popular[2]. VLC for Android Beta is a ported version of VLC media player for desktop environments. It currently is in a beta phase.

9.1.1 Features

VLC for Android can decode many multimedia file formats[3], using a large number of free decoding libraries, including FFmpeg[4]. The current version of VLC for Android has support for devices with ARMv6[5], ARMv7, ARMv7+NEON[6] and x86 CPU architectures and is written almost entirely in C++. Additional multi-core decoding is supported for Cortex A7[7], A9[8] and A15[9] processor based devices. Other key features of VLC for Android are automatic screen rotation and touch gesture control.

[1] http://www.videolan.org
[2] https://play.google.com/store/apps/details?id=org.videolan.vlc.betav7neon
[3] http://www.videolan.org/vlc/features.html
[4] http://www.ffmpeg.org/
[5] http://www.arm.com/products/processors/instruction-set-architectures/index.php
[6] http://www.arm.com/products/processors/technologies/neon.php
[7] http://www.arm.com/products/processors/cortex-a/cortex-a7.php
[8] http://www.arm.com/products/processors/cortex-a/cortex-a9.php
[9] http://www.arm.com/products/processors/cortex-a/cortex-a15.php

9.1.2 Architecture

LibVLCcore

VLC is a modular framework that consists of one central core: LibVLCcore. This core manages the threads, loading/unloading modules (codecs, multiplexers, demultiplexers, etc.) and all low-level control in VLC.[10]

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. Modules on the other hand communicate directly with the core.

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. Given the modular nature of VLC's architecture, unnecessary modules can be taken out in order to reduce the footprint of the framework.

Multi-threading

VLC is a multi-threaded framework. One of the main reasons for using multi-threading is to warrant that an audio or video frame will be played at the exact presentation time without blocking the decoder threads.

9.1.3 Documentation

Ample documentation is available that covers both the core as well as the modules. Additional Android-specific documentation regarding the source code compilation and debugging process is also available[11]. An extensive knowledge base is also publicly available, in which codecs, file formats and protocols are documented.[12].

9.1.4 Updates

The VLC for Android application is backed up by a non-profit organization called VideoLAN[13] and at the time of writing this report, the source code is updated on almost a daily basis.[14]

9.1.5 Performance

A small performance test of VLC is done by playing two videos encoded in h.264, one with a resolution of 720p, and the other with 1080p. These are popular video formats used in encoding videos. This test is done on the Google Nexus 10" tablet device, which has VLC installed from the Google Play Store.

[10] http://wiki.videolan.org/Hacker_Guide/Core/
[11] https://wiki.videolan.org/AndroidCompile/
[12] https://wiki.videolan.org/Knowledge_Base/
[13] http://www.videolan.org/videolan/
[14] http://git.videolan.org/?p=vlc-ports/android.git;a=summary

The result is that VLC for Android features smooth HD playback, and no frame lag has been detected. The automatic screen rotation is also fast and smooth. Touch gesture control also feels responsive and looks smooth.

9.2 Stagefright

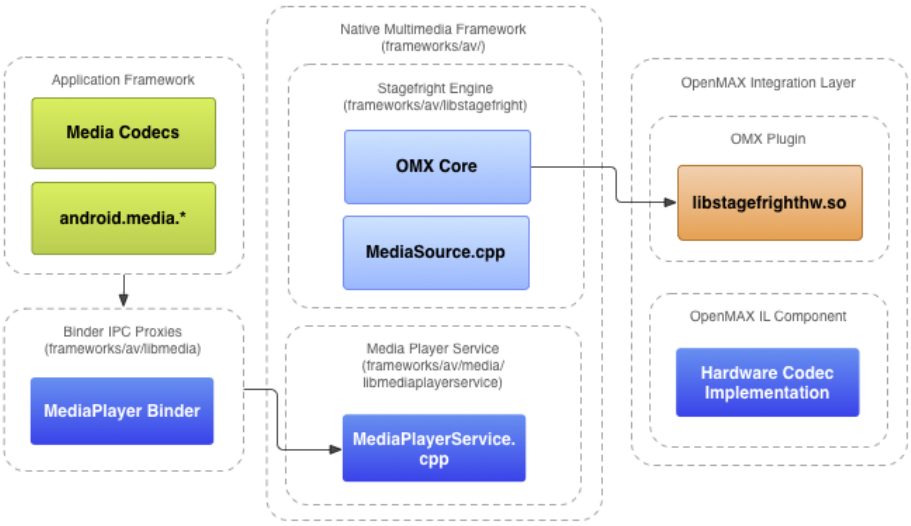
Stagefright is the primary open-source multimedia framework that comes built-in with the Android operating system[15]. It includes support for playing a variety of common media file formats.[16]

9.2.1 Features

Stagefright comes with built-in software-based codecs for several popular media formats. It also supports integration with hardware-accelerated OpenMax[17] codecs, and has HD 1080p video playback capabilities. Stagefright also supports session management, time-synchronized rendering, transport control and DRM.

9.2.2 Architecture

In the following figure an overview is given on the interaction between applications and the Stagefright framework, along with its external components.



Applications can utilize the Stagefright framework by using the classes inside the android.media APIs[18]. These classes communicate with the Stagefright framework by making use of the Binder[19] class which facilitates inter-process communication. As mentioned, Stagefright comes with built-in software codecs, and can be extended with hardware-based OpenMAX(OMX) codecs, called

[15] http://source.android.com/devices/media.html
[16] http://developer.android.com/guide/appendix/media-formats.html
[17] http://www.khronos.org/openmax/
[18] http://developer.android.com/reference/android/media/package-summary.html
[19] http://developer.android.com/reference/android/os/Binder.html

components. An additional shared library, an OpenMax plug-in, links these codecs to Stagefright.

9.2.3 Documentation

Extensive documentation, including API guides and references, is available through the official android developer portal.[20]

9.2.4 Updates

The Stagefright multimedia framework is updated with every new android version, and up-to-date hardware video codecs are supplied with it by the mobile device vendors.

9.2.5 Performance

Stagefright offers excellent performance as tested with the same tests as done with VLC in Section 9.1.5, most likely due to its native hardware acceleration support.

9.3 Dolphin Player

9.3.1 Introduction

Dolphin Player[21] is an open-source multimedia player for Android that is based on the FFmpeg multimedia framework, which allows it to decode most known multimedia file formats and compression methods. Dolphin Player is currently still a work in progress and contains many bugs and performance issues[22].

9.3.2 Features

Dolphin Player can decode most of the audio and video file formats. It supports android devices with an ARMv6, ARMv7-A, x86 and MIPS architecture. HD video files are reported to have lagging issues[23].

9.3.3 Architecture

The architecture of Dolphin Player can be divided into a native (C/C++) part and a Java part. The native part consists of the FFmpeg library, as well as several other smaller support libraries such as the SDL library[24] that provides low level access to audio and graphics hardware via OpenGL. Another example of one of the smaller libraries used is the bzip2[25] library which offers data compression services.

[20] http://developer.android.com/develop/index.html
[21] http://code.google.com/p/dolphin-player/
[22] http://code.google.com/p/dolphin-player/issues/list
[23] https://play.google.com/store/apps/details?id=com.broov.player
[24] http://libsdl.org/
[25] http://www.bzip.org/

9.3.4 Documentation

There is hardly any documentation available, except for brief instructions on how to compile the source code.

9.3.5 Updates

The last update of the source code was on July 3rd, 2012[26]. This shows that the source code of Dolphin Player is not recently maintained and that compatibility issues with current and future mobile devices are very likely to arise.

9.3.6 Performance

The current version of Dolphin Play for Android is suffering from frame lag issues, which results in frames being skipped, as well overall choppy playback. As a result of this, audio and video tracks can become out of sync. This was tested with the same method as described in Section 9.1.5.

9.4 Trade-off

In this section a trade-off will be made between the previously listed video decoding frameworks, resulting into a conclusion in which a framework is chosen that will be used in the prototype application of this Bachelor project. In the trade-off, the following criteria are taken into consideration:

- Number of supported multimedia file formats.
- Hardware decoding: to what degree does the framework support hardware accelerated decoding.
- Android version support.
- Performance: frame rate and image quality.
- Updates: how often is the framework updated and maintained.
- Documentation: quantity of documentation.
- Number of users.

In Table 9.1 an overview of the trade-off is given which shows how each of the frameworks scored against each of the listed criteria.
It should be noted that the supported file formats are overlapping. This means that Dolphin Player includes support for the all the file formats that are supported in Stagefright, plus an additional set of 15 file formats. VLC for Android supports all the file formats supported by the other frameworks, plus an additional set of 167 file formats.

[26] http://code.google.com/p/dolphin-player/source/list

9.5 Conclusion

Based on the findings in this section, VLC is selected as the multimedia framework for the prototype.

Table 9.1: Trade off between three video decoding platforms for Android

	VLC for Android	Stagefright	Dolphin Player
Number of Supported Multimedia File Formats(Video + Audio)	111+93	5+17	24+13
Hardware Decoding	Partly (only on some devices)	Yes	No
Android version support	Device Dependent, up to 4.3	2.0 and higher	2.1 and higher
Performance in playback	Fast and Smooth	Fast and Smooth	Choppy Playback
Updates	Frequent	Frequent	few to none
Documentation	Extensive(but less for Android specifically)	Extensive	Almost none
Number of Users	More than 10.000.000[27]	More than 900.000.000[28]	More than 100.000[29]

[27] based on number of installs in the Google Play Store
[28] http://www.forbes.com/sites/roberthof/2013/07/18/live-google-q2-earnings-fall-short-shares-down
[29] via footnote 27

Chapter 10

Risk Analysis

In this chapter various factors which may jeopardize the success of the project are identified and assessed by performing a risk analysis. This analysis involves identification of various countermeasures to successfully deal with these threatening factors and avoid project failure.

10.1 Risk Factors

10.1.1 VLC for Android incompatible with target device

Given the fact that VLC for Android is currently still in a beta phase, there is a chance that the framework will be incompatible with the target device of the project: the Google Nexus 10"[1].

Indications:

- Component fails to compile for target device architecture.
- Component fails to install on target device.
- Component fails to run on target device.
- Component crashes unexpectedly during runtime.

Countermeasure:
The VLC for Android component will be dropped and the native Android multimedia framework (Stagefright) will be used.

Timing of countermeasure action:
When the above listed indications appear during initial testing of the component on the target device, the countermeasure will take place. Initial testing of this component will be performed at the end of the Orientation phase.

Impact:

[1] http://www.google.com/nexus/10/

As a result of dropping the VLC for Android component, the number of supported multimedia formats in the prototype will be significantly reduced. However, the fewer codecs that are supplied with Stagefright are hardware accelerated and are guaranteed to work, as these hardware-based codecs are supplied by the hardware vendor. The 'must have' functionalities of the prototype will therefore not be compromised by this solution.

10.1.2 VLC for Android not working in combination with Tribler components

Given the bleeding edge nature of the combination of VLC for Android with P2P streaming data, there exists a risk that this video decoding framework will not work with key components of Tribler.

Indications:

- Component fails to play video data from Tribler components.

Countermeasure:
The VLC for Android component will be dropped and the native Android multimedia framework (Stagefright) will be used.

Timing of countermeasure action:
When there is no concrete evidence of a working combination of VLC for Android and the key components of tribler before the end of the first week of the implementation phase.

Impact:
As a result of dropping the VLC for Android component, the number of supported multimedia formats in the prototype will be significantly reduced. However, the fewer codecs that are supplied in combination with Stagefright are hardware accelerated and are guaranteed to work, as these hardware-based codecs are supplied by the hardware vendor. The 'must have' functionalities of the prototype will therefore not be compromised by this solution.

Python for Android

In case Python for Android does not work on the target devices, the Tribler core will not be able to be ported to Android, since this is completely written in Python.

Countermeasure:
A plug-in for VLC will be written that will make use of the libswift[2] library.

Impact:
Many of the unique Tribler functionalities will not be available, and the prototype application will not be easily extendable towards having Tribler func-

[2]http://libswift.org/

tionalities. The BitTorrent engine which Tribler relies on: Libtorrent, will be implemented in this case.

10.1.3 Frame rate performance issue

VLC

Given the fact that VLC for Android is currently still in a beta phase, the playback performance with some multimedia file formats can be unacceptable

Countermeasure:
Based on how many file formats are affected by the performance issue, the VLC component will be dropped and the native Android multimedia framework (Stagefright) will be used, or individual modules responsible for decoding will be adapted, if possible, or the formats will be reported as not functioning correctly and will not be supported in the prototype application.

Impact:
In the worst case, the number of supported multimedia codecs in the prototype will be significantly reduced. However, the fewer codecs that are supplied in combination with Stagefright will be hardware accelerated and are guaranteed to work, as these hardware-based codecs are supplied by the hardware vendor. The 'must have' functionalities of the prototype will therefore not be compromised by this solution.

In the best case, only a few uncommon multimedia file formats will be unsupported in the prototype application, which will hardly affect its video decoding capabilities.

Part IV

Requirements

Chapter 11

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 Section 3.5.2 of the Plan of Action.

11.1 Functional requirements

11.1.1 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.

11.1.2 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.

11.1.3 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.

11.1.4 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.

11.2 Nonfunctional requirements

11.2.1 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.

11.2.2 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.

11.2.3 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 [3]. 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.

11.2.4 Would have

5. *Optimize the start-up time;*

11.3 Constraints

1. *The prototype must be ready before the 2nd of October 2013*;

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

Part V

Test and Implementation Plan

Chapter 12

Introduction

In this document the planning for how and when the team will implement different functionality, will be elucidated. The implementation will be based on incrementally adding the different requirements as functionalities. Chapter 13 will explain how the functionalities will be tested. The next chapter describes the functionality that will be implemented and a planning of when each part is implemented. Also included in this planning is when the tests will be executed.

Chapter 13

Testing

This chapter provides insight into how testing will be executed in this project. The first section will explain how unit testing will be incorporated in the project, which is followed by a section about acceptance testing.

13.1 Unit Testing

Every function or method can be modeled as a black box; the black box processes the input and generates the output. To test if the correct output is generated, unit testing is used. Within a unit test the tester can define the input and what the expected output is, then the unit test runs to test if the function's output is the same as the expected output. If not, it will warn the tester that the test failed so the function can be changed to behave correctly. In this project the team will test all methods that have adequate logic in them. For example, getters and setters will not be tested. Methods from third-party software like Tribler, Libtorrent and VLC will also not be tested separately.

13.2 Acceptance testing

Acceptance testing means that the team will see if the software in the current state meets the requirements by manually testing the functionality. In the different sprints, functionality will be added to the prototype. Acceptance testing will be done in between the sprints to ensure that the prototype meets the requirements set to be implemented for that sprint.

Chapter 14

Implementation

In the Requirements Analysis, the requirements were prioritized according to the MoSCoW method. The priority in that document will be a guideline to plan the different stages of implementation. Every sprint planning, a number of functional requirements will be implemented as functionality to the prototype. For the explanation of the requirements, please see the Requirements Analysis. The following planning is subject to the risks involved, which are elucidated in the Orientation Phase and could therefore not be accurate later on in the project.

14.1 Sprint planning one

In sprint planning one, the aim will be to let a predetermined video, play on Android via VLC. This satisfies the requirement of playing a video. When this functions correctly, the requirements of pausing and seeking in a video are also satisfied because they depend on VLC. Acceptance testing will be used to test if this is indeed the case.

14.2 Sprint planning two

In this sprint planning the goal will be to incorporate the Tribler core in to the mix. Making the python-based program work on Android will be the challenge in this sprint planning. The Tribler core, which includes Libtorrent, should be capable of downloading a video on Android and together with the work done in sprint planning one, the prototype should now be able to download a predetermined video and let it play through the use of VLC. Testing will be done afterwards to check if the prototype meets the currently set requirements.

14.3 Sprint planning three

When the download algorithm is adapted for streaming, the prototype can stream the video directly to VLC instead of first downloading it. The prototype should have all the 'must have' requirements now, to test if this is the case, acceptance testing is used.

14.4 Sprint planning four

Tribler has a search function which searches for videos to download in a non-centralized way. In this sprint planning the team will attempt to incorporate the search function so users can search for a video, click play and then the video will be streamed to their device. Again, acceptance testing will be used before the next sprint planning to check if the prototype is still in a fully working state.

14.5 Sprint planning five

The last sprint planning will be to incorporate the requirement to optionally download a video. A single click installer will also be provided to incorporate VLC and Tribler into one single .apk file (the install file format for Android). After this, the prototype will be subjected to acceptance testing to check if everything works as intended.

Part VI

Architectural Design

Chapter 15

Proposed Architecture

In this chapter the proposed prototype will be separated into different subsystems to give more insight into how the team will 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 15.2. 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 15.3. How information flows from subsystem to subsystem in different use cases can be seen in Section 15.4. In the final section, we will explain how the system will deal with starting and stopping as well as crashes, how this will be implemented is explained in the final section.

15.1 System composition

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

15.1.1 Subsystems

- **GUI**
  - Start
    The GUI which is shown when the prototype is started, will show a start button. What happens when this button is pressed is further explained in the Section 15.1.2, which is about the composition of the different subsystems.
  - VLC media player
    The media player which comes with VLC for Android Beta has all the functionality needed for video playback control. It functions as the GUI when the user is watching a video. The user can use gesture controls to seek in the video, pause and then resume it and adjust the volume.
- **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 Control
    A buffer helps to ensure smooth playback, it puts media data from the storage in to 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
    The core is the main part of Tribler and includes the control of all modules including LibTorrent.
  - 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 the following download algorithm described by Petrocco et al[3]: 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.

15.1.2 Composition

In Figure 15.1 a visual overview of the proposed architecture can be found. In this figure, the different subsystem are combined in to one system. At the

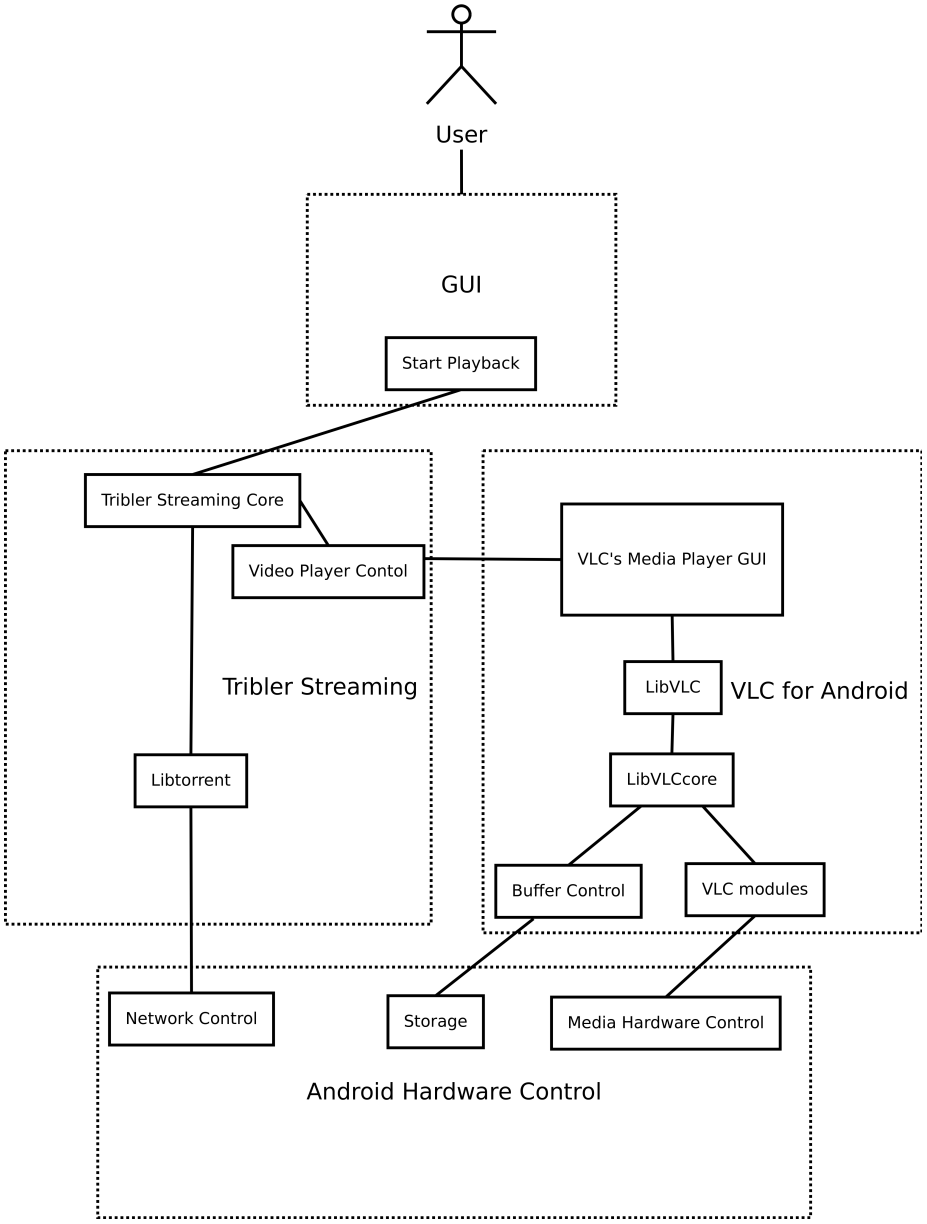


Figure 15.1: The proposed architecture of the prototype

top sits the user which issues the command to the GUI to start the process of streaming a video. It will then switch to another GUI, which VLC provides, for media playback (VLC's Media Player GUI). The user can play and pause the video as well as seek in the video via this GUI. After the user issues the command to start playback, the Tribler subsystem will download pieces from the video with the Libtorrent protocol. It will store those pieces so that the VLC subsystem can show them to the user when there are enough pieces for continues

playback. VLC does this by filling a buffer and letting the player use that for continues playback. The Media Player lets the Video Player Control know what the playback position is so that Libtorrent can download the proper pieces for playback as explained in the subsystems section. The Android Hardware Control subsystem is added to clarify that the interaction with hardware is mostly handled by the Android platform. To clarify, the part that the Team will code themselves, is the Tribler core and the video player control module. The rest may undergo a few minute changes, but will be kept as is to facilitate updates.

15.2 Persistent data management

15.2.1 Video Data

The prototype application will have to be able to stream high definition videos that in case of Blu-ray video, can have a bit rate of up to 40 Mbit/s[1]. In case a video is encoded in DivX Plus HD 1080p format, it can have a maximum bitrate of 20 Mbit/s[2]. In order to decode video streams with such bit rates in real-time, data will have to be downloaded into RAM, after which it can directly be decoded. The contiguous pieces that are downloaded from the peers will therefore be stored in a buffer that resides in RAM. At the same time, the piece can also be stored to non-volatile storage such as on board Flash Memory, or to an external SD-card, in order to share the piece with other peers later in time.

15.2.2 Peer Data

The peer data, such as peer IP addresses and the relationships between these addresses and available files and pieces, will be stored in a Distributed Hash Table that will reside in the RAM. Having this hash table reside in main memory allows for fast execution of queries, insertions and deletions.

15.2.3 Application Preferences

Preferences that can be set by the user of the prototype application will be stored persistently on the internal storage of the target device. These preferences will be private to the application and will persist across user sessions.

15.3 Concurrency

In order to avoid blocking any processes, all major I/O will run in separate threads. All network related transactions will run in a separate network thread (upload/download of pieces). The video decoding will be performed in several parallel threads that can each run on a separate CPU core. Finally, the GUI of the prototype application will also run in its own thread. It should be noted that there will not be any complex logic or potentially blocking (synchronous) calls performed in the GUI, as this could compromise the responsiveness of the GUI, as well as the overall user experience.

15.4 Software control

At start-up, the system will initialize the python environment, after which the Tribler core will be loaded. At the same time the VLC core will initialize and load the necessary video decoding modules. When all the required components have been initialized a GUI will be shown, containing a start button. The system will then perform the actions described in Section 15.1.2 and stream a video.

[1] http://www.blu-raydisc.com/Assets/Downloadablefile/BD-ROM-AV-WhitePaper_100604(1)-15916-18123.pdf

[2] http://www.divx.com/files/DivX_Plus_HD_Brochure.pdf

While streaming, the playback can be paused, or the user can seek to a different part of the video using a slider control, after which the system will continue streaming from there. It should be noted that when the playback is paused, the system will continue downloading pieces of the video in the background. In case of a system crash, the system will restore playback of the video that was playing at the moment of the crash, at a point in time close to what it was before the crash occurred.

15.5 Boundary Conditions

The prototype application will run on an Android-based mobile device and will communicate with other peers through the Internet using the BitTorrent protocol. Therefore in order for the prototype application to work, an Internet connection is required. The bandwidth of the Internet connection should be equivalent to an average broadband DSL connection (10-20Mbit/s). Once the system is started it can be closed either by selecting 'close' from a context menu, by terminating the application with the task manager in Android, or by rebooting the device. In order to facilitate smooth video playback, the device on which the prototype is run should contain hardware that is capable of playing back (decoding) 1080p at 30 fps.

Part VII

Implementation Phase

Chapter 16

Implementation

A vital part of Tribler is the protocol with which, it downloads the torrents. As explained in Section 8.2.1 this protocol is Libtorrent. Because the python code of Tribler proved too time-consuming to port to Android with the Python for Android project, the Team fell back on the risk analysis in Section 10 and choose to implement Libtorrent. This part describes how Libtorrent and VLC were combined into one prototype application to stream videos. Section 16.1 describes how the team formation changed going into the implementation phase. The following chapter describes how VLC was implemented on Android. How the Libtorrent libraries were built and included in the prototype is explained in Section 16.3. The following step was combining VLC and Libtorrent, how this was done is explained in Section 16.4. The final section describes how streaming was achieved.

16.1 Change in the Team

At the 29th of August, the Team had a meeting with the Supervisor in which Jaap van Touw brought to light that Martijn Breet had not done anything for the deadline that week. Martijn was given one more chance to make up for lost time, but he decided to stop his participation on the third of September. In this same timespan, the Client brought in the helping hand of Jan-Willem van Velzen. He was brought in as an external developer to help with programming. From then on he programmed approximately four days per week to help finish the prototype for the 2nd of October.

16.2 VLC

The first step into building a streaming application, is checking if VLC can be compiled for Android and if its source code can be modified. Otherwise, a different video decoding framework had to be included in the prototype. VLC was already available for Android, as well as its source code[1]. Later on, it is necessary to include VLC in the prototype (see Section 16.4). To do this, the source code is needed as well as a method to compile it into an application.

[1]source code can be obtained from: git://git.videolan.org/vlc-ports/android.git

VLC was compiled by following the guide in Appendix B. This guide was made by collecting information from different resources over the Internet, including the VLC wiki[2] and several forums. The guide is also made available on Github (see Section 4.3). Now, VLC has been tested to work on the target device and can be modified to meet the needs of the prototype.

16.3 Libtorrent

To build the Libtorrent libraries for Android, the Team first tested its functionality on Linux. It was easily compiled for Linux in which the Team ran a client test to see if a torrent file could be downloaded. Android however, has a different compiler, which comes with the Android NDK[3]. To compile Libtorrent with this compiler was not without errors however. This was due to Libtorrent's dependency on Boost.

16.3.1 Boost

This dependency meant that the Boost[4] libraries had to be build for Android as well. Boost is a popular set of C++ libraries, which include functions such as image processing, regular expressions and multi-threading. In the end, a github repository[5] was used which builds Boost for Android after calling a compile script, in which also the version of Boost can be specified.

16.3.2 RuTracker

After the Boost libraries were put in place, the Libtorrent library could still not be compiled with the compiler from the Android NDK. A lot of experimentation was done with different versions of Boost, different compile options and different versions of Libtorrent, but without satisfactory result. Then Egbert Bouman from within the Tribler team came with a tip to look into the simple torrent client by the name of: RuTracker[6], short for Russian tracker. This application for Android was open source and used Libtorrent for its download functionality. The Libtorrent libraries could then be build with the ndk-build command. The few errors still remaining were quickly resolved by changing one of Libtorrent's configuration options (see Appendix C). The guide on building Libtorrent and Boost is also made available on Github (see Section 4.3). The only drawback of using RuTracker's method is that it can not build the Libtorrent libraries for the 'Mips' and 'x86' architectures, meaning that some Intel and Mips tablets are not supported.
After building the libraries, the Team extracted these and put them in a separate client test project.

The application could, at this stage, download a torrent and send the downloaded file to the previously build, and separately installed, VLC for playback.

[2] https://wiki.videolan.org/AndroidCompile/
[3] http://developer.android.com/tools/sdk/ndk/index.html
[4] http://www.boost.org/
[5] https://github.com/MysticTreeGames/Boost-for-Android
[6] http://rutracker.org/forum/index.php

The torrent file itself must first be downloaded to the device so it can be selected in the application. At this stage, this works by using a separately installed file-browser, such as ASTRO[7].

16.4 Combining VLC and Libtorrent

The Team wanted to deliver the streaming application with VLC built-in. This way, the user can play a video without having to worry about installing the right media framework. To achieve this, the Team tried to let VLC act like a library from which the Libtorrent client application could call functions for playback. However, VLC for Android wasn't build with this purpose in mind, as the team discovered after some experimentation. The Team fused the two projects together by putting the source of the Libtorrent client application together with VLC in one project. The Team now had one .apk file, which can be installed. At this stage, it first downloads the file and then calls the VLC part of the prototype to play the media file.

16.5 Streaming

The next part to implement is to download the media file while VLC plays it at the same time. By making two threads, one for downloading and one for VLC, the application could download while the user watches the video. To make this a smooth experience, the Team implemented a buffer, which filled until the buffer was large enough, so that the user could continuously watch the video. This is based on the average speed and the file-size of the video to be streamed. In the end, the download algorithm wasn't implemented as described in Section 8.2.4. The prototype sets Libtorrent to download pieces in sequential mode as advised by the Tribler team.

[7] https://play.google.com/store/apps/details?id=com.metago.astro

Part VIII

Final phase

Chapter 17

Evaluation

17.1 Conclusion

In response to the growth of Internet traffic on mobile devices and to meet the increasing demands of the market, the Team set out to develop a mobile version of Tribler's Video on Demand. The following research question was central to the research and development of the sought-after prototype:

"How can we make video-on-demand available for mobile devices using a non-centralized approach?"

The solution is created in the form of a prototype Android application, which features a non-centralized Video on Demand service. As a result, the following contributions are made:

- An open source application on Android, which streams video and audio from torrents.
- An application, which can hardly be taken down by political intervention or other techniques. (see Section 8.3.1 for more advantages of a non-centralized approach)

17.1.1 Limitations

The application is still a prototype; a number of bugs and inherent limitations are still in the program including the following:

- It cannot play DVD's and Bluray DVD's, the video in a DVD is divided over multiple files, but the application can stream only one file. It will play one part of the DVD and stop after that.
- It doesn't support playlists yet, so if an album is downloaded, it will only play one file of that album.
- At some times, the file doesn't download and the application has to be restarted before it downloads again.
- The torrent file (.torrent) itself has to be downloaded externally.

- The download speed of a file is based on the availability in the swarm and upload speed of other peers as well as the maximum connection speed set by the user's Internet Service Provider. This greatly affects how long it takes before a file starts playing. This limitation is inherent to the disadvantages of using a non-centralized approach.

17.2 Future Work

There is much work that could be done after this project. In the requirements elicitation there were, for example, a lot of features that could be implemented. But, due to the limited time the Team had, they only implemented a subset of these requirements. Future work could entail Dispersy support, searching functionality, Libswift support and video uploading. Furthermore, the application could be extended to other mobile platforms such as iOS and Windows Phone. Lately there has been a lot of news about governments listening in on private conversations between people and watching their data traffic across the Internet[1]. To combat this, a release of a more polished version of the application in the Google Play store would be able to generate a lot of publicity for Tribler and its goal to create a censorship-free Internet[2]. Another feature that the Tribler team is working at, is anonymous tunneling, in which Tribler facilitates anonymous data traffic. To implement this in the Android application would be another huge step towards creating a censorship-free Internet.

17.3 Reflection

The project was not without stride and difficulties, but this was one of its many charms. In the end I think we delivered on satisfactory level, given the time and delays in the first stage of the project. When Martijn left, I understood that not a waking moment should pass without me working or thinking on the project, if I wanted to finish on the previously set deadline. Jan-Willem was of great help in trying to accomplish the latter. When I was busy trying to figure out how I could compile components such as Libtorrent and Boost, he went on to implement my ideas. I learned about many things such as native code on Android, Python, the Boost libraries, P2P networks and the inner workings of VLC. Also on a more personal level, I learned how to manage expectations and deal with people that don't deliver content on time. I had a lot of fun during the project and was happy to work on cutting edge technology with the possibility of creating a lot of impact in the world. An application such as the one here has not yet been seen on the market in a non-centralized way, which motivates me to continue working on this application.

[1] http://investigations.nbcnews.com/_news/2013/06/06/18809021-sources-us-intelligence-agencies-tap-servers-of-top-internet-companies

[2] https://github.com/Tribler/tribler/wiki

Bibliography

[1] Hendrik Schulze, Klaus Mochalski, *Internet Study 2008/2009*, Ipoque, Germany, 2009.

[2] Dai Clegg, Richard Barker, *Case Method Fast-Track: A RAD Approach* Addison-Wesley, 2004.

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

[4] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M. van Steen, H.J. Sips, *Tribler: A social-based Peer-to-Peer system*, TU Delft, Vrije Universiteit Amsterdam, 2006.

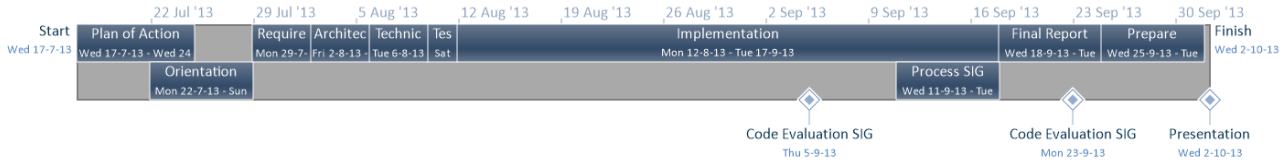
[5] Niels Zeilemaker, Boudewijn Schoon, Johan Pouwelse, *Dispersy Bundle Synchronization*, TU Delft, Parallel and Distributed Systems, January 2013.

[6] Andrew S. Tanenbaum, David J. Wetherall, *Computer Networks*, 5th edition, Pearson Education, 2010.

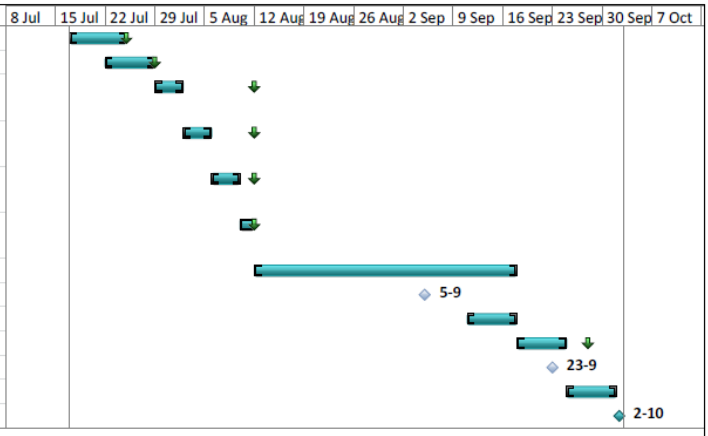
Appendices

Appendix A

Project Planning



ID	Task Name	Duration	Start	Finish
1	Plan of Action	1,14 wks	Wed 17-7-13	Wed 24-7-13
2	Orientation Report	1 wk	Mon 22-7-13	Sun 28-7-13
3	Requirements Analysis Document	4 days	Mon 29-7-13	Thu 1-8-13
4	Architectural Design Document	4 days	Fri 2-8-13	Mon 5-8-13
5	Technical Design Document	4 days	Tue 6-8-13	Fri 9-8-13
6	Test- and Implementation Plan	2 days	Sat 10-8-13	Sun 11-8-13
7	Implementation	5,29 wks	Mon 12-8-13	Tue 17-9-13
8	Code Evaluation SIG		Thu 5-9-13	
9	Process SIG Feedback	1 wk	Wed 11-9-13	Tue 17-9-13
10	Final Report	1 wk	Wed 18-9-13	Tue 24-9-13
11	Code Evaluation SIG		Mon 23-9-13	
12	Prepare presentation	1 wk	Wed 25-9-13	Tue 1-10-13
13	Presentation	2 hrs	Wed 2-10-13	Wed 2-10-13



Project: Mobile VoD

Task Start Deadline

Milestone Finish

Appendix B

Build Instructions VLC

ENVIRONMENT
install:

- Oracle JDK SE 'latest' (tested on Java version 7u25)
- Android ADT bundle (follow instructions on the download site)
- Android NDK R9 (including legacy toolchains)

SETUP
Edit environment variables:
sudo gedit /etc/environment
add ANDROID_SDK=/home/user/android/android-sdk-linux
add ANDROID_NDK=/home/user/android/android-ndk-r9
add ANDROID_ABI=armeabi-v7a

Add to PATH environment variable:
/home/user/android/android-sdk-linux/platform-tools:/home/user/android/android-sdk-linux/tools
install the following tools:
apache-ant (or ant), autoconf, automake, autopoint,cmake, gawk (or nawk), gcc, g++, ia32-libs, build-essential, libtool, m4, patch, pkg-config, ragel, subversion, and up-to-date versions of those tools.

On the command line:
sudo apt-get install ant autoconf automake autopoint cmake gawk gcc g++ ia32-libs build-essential libtool m4 patch pkg-config ragel subversion

Get the source code:
git clone git://git.videolan.org/vlc-ports/android.git VLC

COMPILING PROCESS
Go in to the folder VLC
gedit /compile.sh
Edit the mcpu flags for the appropriate target CPU (cortex-a15 for Nexus 10)

gedit /vlc-android/jni/Application.mk
Modify content into NDK_TOOLCHAIN_VERSION=4.8
sh compile.sh

Compiling should be done, go to device installation instructions

COMPILING TROUBLESHOOT
If the compiling stops unsuccessfully, run the following command: find ~/vlcsourcefolder/ -name "udiv.asm"
Edit each file from the search results, and remove the spaces between the [ ] brackets, if any.
sh compile.sh (start the compilation again, it will now successfully continue and finish)

TARGET DEVICE INSTALLATION INSTRUCTIONS:
cd /vlc-android/bin/ folder
Install the apk file with the 'adb install filename.apk' command

ECLIPSE
You can load the project into eclipse(preferably from the ADT bundle) by importing android project from existing source. (new project-Android-Android project from existing code).
Guide the wizard to your vlc folder and load the four projects that the wizard found.
Be sure to set the dependencies in the VLC project correctly by right-clicking on that project, go to the Android tab, and setting the three other projects as libraries for VLC.
You can now run and debug VLC from the Eclipse environment.

Appendix C

Build Instructions Libtorrent

BUILD BOOST
Tirst clone the boost for android project from github:
git clone https://github.com/MysticTreeGames/Boost-for-Android

Go to the directory in which the project resides, example:
cd git/Boost-for-Android

Then let boost-for-android, download and build boost by issuing the following command:
./build-android.sh

SETUP ENVIRONMENT
Add the following to environment (sudo gedit /etc/environment)
AndroidNDKRoot=path/to/android-ndk-r9_32
NDK_ROOT=path/to/android-ndk-r9_32
NDK_MODULE_PATH=path/to/Boost-for-Android
BOOST_ROOT=path/to/boost
BOOST_BUILD_PATH=path/to/boost/tools/build/v2

add the following to PATH:
path/to/Boost-for-Android

EXTRACT LIBTORRENT FROM RUTRACKER
Download the RuTracker source:
svn checkout http://softwarrior.googlecode.com/svn/tags/RutrackerDownloader/2.6.5.5/
extract all from the JNI folder.

MODIFY CONTENTS In Libtorrent source, adapt file.cpp:
Change *ifdef __NR_fallocate* to: *ifndef __NR_fallocate*

Add the previously build Boost libraries to libboost and make sure the filenames of these libraries in Android.mk are correct
In Application.mk, change or add APP_ABI to APP_ABI := armeabi armeabi-v7a

BUILD LIBTORRENT LIBS In a terminal:
Go to the folder where RuTracker resides: cd path/to/RuTracker
Issue the following commands:
ndk-build clean
ndk-build

After a while it should have created the libraries libtorrent.so for both armeabi and armeabi-v7a in the libs folder.

Appendix D

Minutes

D.1 July 26th, 2013

The meeting was opened at 15.39

1. Review agenda
   Agenda is accepted

2. Planning
   The supervisor said that the gantt-chart of the planning is good, but watch out with extending your own deadlines. It shouldn't happen that if we extend it one week now that it turns to six weeks at the end. The TU Delft will base the grade mostly on the report, the client on the product.The supervisor also said that another group first did the problem description,then the orientation report and after that the Plan of Action. The team said they are busy with the plan of action and will write the orientation report after this, so the problem description in the Plan of Action is kept broad on purpose.

   (a) Plan of action
   The supervisor asked the team to hand in the first draft on monday before 11.00 am. The team explained they are finalizing the last chapter. After a bit of talk about a risk analysis, the decided to put a reference in the plan of action, and a more detailed version in the orientation report.

   (b) Orientation report
   The supervisor asked the team to hand in the first draft on monday before 11.00 am. And the weekend after the final version, after the supervisor has reviewed the document. Trade-off's are good to have in an orientation report, explained the supervisor, also explain only the chosen one in detail and give a brief overview of the rest of the possibilities.

   (c) Planning of the rest of the project
   The supervisor recommended that one week to make a final report is not a lot of time, so spend less time on design (excl. requirements)

and change it later during implementation the team has more time for the implementation.

3. Other issues
   The supervisor recommended to keep the client up to date, especially about the requirements document. In the near future a meeting with the supervisor, client and the team should be planned, said the supervisor. Another issue he adressed was that the team should make sure the test-tablets come early; it shouldn't be so that the team has to wait two weeks when they want to start testing on a tablet.

There were no further questions and the meeting was ended at 16.05

D.2 August 1st, 2013

The meeting was opened at 10.00

1. Review agenda:
   Agenda is accepted

2. Review of prior minutes:
   No further remarks about the prior minutes are made.

3. Sprint planning

   - Completed tasks during last sprint:
     The team reports that more time is needed for the completion of the Orientation Report. The team states that it is still working on the video decoding sections, and that several other sections of the report still need to be extended and reviewed before release. The team says it can finish the remaining work by the end of the week. The supervisor adresses the fact that the team should be aware of time consumption during the current phase of the project; losing an additional week means losing 1/6th of the available implementation time. Implementation time, according to the Supervisor, is really critical for this project, and should be safeguarded as much as possible.

     Additional adivce is given by the Supervisor in relation to the (software) components that will be used in the project. The Supervisor advises the team to individually check each component to see if it works under Android. He also notes that the team first should only focus on being able to stream something. Only after this core functionality has been achieved, the team should focus on further improvements or additional features.

     The Supervisor re-addresses the fact that the team should do a sanity check on all components to really verify if these components will really run on an Android environment. The Supervisor also asks the team if there already is a component available to run Python code on Android. The team states that there is already a working component

written by Ed Knutson, that does exactly that. The team also says that Libswift runs on Android.

- Encountered impediments:
  Apart from the previously mentioned difficulties, no further impediments are reported by the Team.
- Tasks to do during upcoming sprint:
  The team and Supervisor both note that the Orientation Report needs to be finished, and a deadline is set on Sunday, August 4th (end of day). The Supervisor also notes that a task distribution, covering all of the project work, should be added to the Orientation Report as an appendix. The team notes that the design documents, consisting of the RAD document, Architectural Design Document, TDD, and Test- and Implementation Plan needs to be finished by the end of the next sprint. The Supervisor and the team both agree on the fact that the RAD document will take most of the time. The Supervisor also adds that the requirements should be validated in accordance with the Client. The Test- and Implementation Plan will only take around an hour, according to the Supervisor. The team and Supervisor both agree on setting the deadline for the design documents at Wednesday, August 7th. All design documents will be sent by e-mail to both the Supervisor and the Client. Finally, the Supervisor says he will try to set up the next progress meeting such that the Client can be present as well, as to facilitate the requirements validation.

4. Other Issues:
   No other issues were reported by the team nor the Supervisor.

There were no further questions and the meeting was ended at 10.35

D.3 August 8th, 2013

The meeting was opened at 10.00

1. review agenda
   Agenda is accepted
2. review of prior minutes
   No further remarks about the prior minutes made
3. sprint planning

   - Completed tasks during last sprint.
     The team went on with improving the orientation report after the Supervisor's feedback.
   - Encountered impediments
     Compiling VLC is quite difficult, the Client advised to look at the Tribler how-to compile VLC on Ubuntu. If difficulties remain and for further questions, mail VLC about the team's project for help.

- Requirements Analysis Document
  The following changes were advised by the Client and Supervisor:
  - The search function to: should have.
  - The nonfunctional requirement about the layout can be removed, smooth user experience is more important.

  The following requirements should be added according to the Client:
  - A single click installer as should have.
  - Libswift as would have.
  - Libtorrent as must have.
  - Dispersy as could have.
  - Support as much codecs as Tribler as should have.
  - Anonymous tunneling/Subset tor protocol as would have.

  The Client stressed that the team needs a wider view and look at what is really catchy or controversial, things that will make headlines at Tweakers.net, anonymous tunneling is one of those features.

  The Supervisor said that the team first has to check if python works, then they can look at VLC, without python, Tribler can't be implemented.
  Ed Knutson has Python working on Android, so check with him for details advised the Client.
  The client told to send Ed Knutson a mail about the current situation.

- Architectural Design Document
  The team said they will make this document now the requirements are set.

- Technical Design Document
  The team said they will make this document now the requirements are set.

- Test- and Implementation Plan
  The team said they will make this document now the requirements are set.

  It is of extreme importance that the Orientation Report is done the 11th of August.

The Supervisor will start reviewing the Plan of Action.

If Python can not be made to work on Android, the functionality must be reached through other ways, such as a plugin for VLC.

4. other issues
   No other issues were reported.

There were no further questions.

The meeting was closed at 10.30.

D.4 August 22nd, 2013

The meeting was opened at 10.00

1. Review agenda
   Agenda is accepted.
2. Review of prior minutes
   The previous minutes were accepted.
3. Sprint planning
   - Encountered impediments
     The team said that the Orientation Report takes up more time than originally thought, but it is in the stage for the first review.
   - Orientation Report
     The supervisor said to take up the timing in the risk analysis, it entails when a decision to, for example, use a different solution must be made.
     The supervisor stressed that at the next meeting the team should know if VLC will work or not.

     All documents should be done by Sunday.
   - Requirements Analysis Document

   - Architectural Design Document
     The team said that they have not yet started work on this document.
   - Technical Design Document
     The Supervisor and team agreed that this document should be pulled in with the architectural document.
   - Test- and Implementation Plan
     The plan to do test-driven development(TDD) should be revised, because it is not suitable for this project. Manual tests according to

the requirements should be done, and unit tests must be included. The plan of action should be revised to exclude TDD.

4. Other issues
   The supervisor stressed that more hours should be spent on the project, on average most people spend 45-50 hours per week.
   Both the Supervisor and the team were curious how SIG would evaluate the code that we include from other sources such as Tribler and VLC. The team should contact SIG to see how this will be evaluated.

There were no further questions.

The meeting was closed at 10.25.

D.5 August 29th, 2013

Opening at 10.00

1. Review agenda
   The agenda is accepted
2. Review of prior minutes
   The prior minutes are accepted
3. Sprint planning

   - Completed tasks during last sprint
     All the documents, except the Architectural Design Document were finished.
   - Encountered impediments

     i. Martijn did not contact his other teammate Jaap for a week, to which Jaap came to the Supervisor for. The Supervisor said to give Martijn one more chance, if a deadline is missed, Martijn will have to redo the Bachelor Project next year. Jaap will then finish the project by himself.
   - Architectural Design Document
     The deadline for the Architectural Design Document is due Sunday the first of September.
   - Playing a predefined video through VLC
     The Team made this work, but haven't tested it on the tablet yet. The Supervisor said to do this today to be sure VLC will be used for the prototype.

     Sunday is the deadline to check if python for Android works.

4. Other issues
   We should mail SIG with the question of how the code is reviewed when the team uses code from other projects such as Tribler and VLC.

There were no further questions.

The meeting was closed at 10.10

D.6 September 5th, 2013

The meeting was opened at 10.02.

1. Review agenda
   The agenda is accepted.

2. Review of prior minutes
   The prior minutes are accepted.

3. Sprint planning

   - Completed tasks during last sprint.

     - Architectural Design Document

     - Python for Android
       The team says that the documents are done and that they have python for Android working.

   - Encountered impediments
     none arose.

   - Playing a predefined video through VLC
     The Client says that to build m2crypto for android will be tough. The Team aims to have streaming working on Android working next Thursday.

   - External developer
     Jan-Willem is the external developer that will help develop the application, he is available for at least three days per week.

4. Other issues
   The Client stressed that the team must attempt to maximize the impact of the application, exposure is very important and any attention towards Tribler would be very good. So try to get more forks and likes at GitHub.com, get people talking about it; at Tweakers, New York Times, etc.

For magnet links, DHT must be looked at.

The Client says that the team should make a guide to compile the project with eclipse.

The following deadlines were created in accordance with the Supervisor, Client and Team:
20/9: hand in final report first version
24/9: hand in final report, final version (process feedback)

The team will ask the Bachelor coordinator about how SIG's role is.

There were no questions.

The meeting was closed at 10.21.

D.7 September 12th, 2013

The meeting was opened at 10.03.

1. Review agenda
   The agenda is accepted.
2. Review of prior minutes
   Prior minutes are accepted.
3. Sprint planning

   - Completed tasks during last sprint.

     - Boost
       Boost is compiled for Android, this library is needed for libtorrent. Thanks to Egbert, the Team can now compile libtorrent as well, through a russian tracker application for android.

   - Encountered impediments

     - SIG
       There is no code to hand in to SIG, so this wasn't done and the Supervisor ensured the Team that this was no issue.
     - Boost & libtorrent The team said that they took a long time finding out how to build boost and libtorrent so a quite some time was lost here. The Supervisor and Client understood this and as example they said that a professor took a full-time week to build libtorrent for Mac.

- Libtorrent
  The Team will now implement libtorrent in their own project to make it simple, fast and responsive. They will use it to download a torrent file and then let it play through a seperately compiled VLC for now.
- Streaming
  After making an app that downloads a torrent through the libtorrent wrapper, the Team will implement the streaming function. To get a video to stream, The Client said that Egbert wrote a wrapper class that does this on a desktop environment for python code. The Team will look at this wrapper to see how it is done and how they can achieve this for android.

4. Other issues
   No other issues were reported.

There were no questions.

The meeting was closed at 10.25.

D.8 September 17th, 2013

The meeting was opened at 15.00.

1. Review agenda
   The agenda is accepted.
2. Review of prior minutes
   The prior minutes are accepted.
3. Sprint planning

- Completed tasks during last sprint.

  - VLC and Tribler's VOD in one apk
    The Team reported that the VOD application and VLC are now in one package so users don't have to install extra media playback software.
  - Libtorrent
    The Team reported that they have Libtorrent working including the boost libraries thanks to a tip from Egbert to look how RuTracker did it.
  - Libtorrent sequential
    For video streaming it is wise to download the pieces in sequential mode, the team reported that they implemented this feature.

- Encountered impediments
  none reported

- streaming
  The Team will now work on implementing the streaming functionality. They will extend the video player class to implement buffering through information received from the Libtorrent library.

4. Other issues
   There were no other issues.

There were no other questions.

The meeting was closed at 15.10.

Appendix E

Work Division

This chapter clarifies who wrote the different chapters and it shows the division in the implementation phase. The division of the report is as follows:

E.1 Report

- Plan of Action
  Martijn and Jaap wrote this together.
- Orientation
  Martijn wrote the following sections:
  - Video Decoding Frameworks
  - Risk analysis

  Jaap wrote the rest:
  - Introduction
  - Mobile Platforms
  - Video on Demand on (non-)Mobile Devices
- Requirements
  Initially Jaap set this up. Martijn processed the feedback from the meeting with the client (see Appendix D.3).
- Test and Implementation plan
  Jaap wrote this.
- Architectural Design
  Jaap wrote the following section:
  - System composition

  And Martijn wrote the rest:
  - Persistent data management
  - Concurrency

  - Software control
  - Boundary conditions

- Implementation Phase
  Jaap wrote this.
- Final Phase
  Jaap wrote this.
- Appendices
  Martijn wrote the outline of the VLC compilation guide, Jaap finalized it. Jaap wrote the Libtorrent guide. Furthermore, Jaap wrote all the minutes, as well as the agendas for these meetings.

E.2 Implementation

Jan-Willem did about 40% of the work in the implementation phase, which contains most of the GUI and the combining of VLC and Libtorrent. Jaap did the other 60%, which includes figuring out how to compile the different libraries, such as Libtorrent, Boost and VLC.