QLectives – Socially Intelligent Systems for Quality
Project no. 231200

Instrument: Large-scale integrating project (IP)
Programme: FP7-ICT

Deliverable D.4.3.1
QMedia v1 - Short report

Submission date: 2010-03-01

Start date of project: 2009-03-01 Duration: 48 months

Organisation name of lead contractor for this deliverable: TUDelft

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidental, only for members of the consortium (including the Commission Services)	

Document information

1.1 Author(s)

Author	Organisation	E-mail
Nazareno Andrade	TU Delft	N.FerreiradeAndrade@tudelft.nl
Tamás Vinkó	TU Delft	T.Vinko@tudelft.nl
Johan Pouwelse	TU Delft	J.A.Pouwelse@tudelft.nl

1.2 Other contributors

Name	Organisation	E-mail
Adele Jia	TU Delft	L.Jia@tudelft.nl
Richard Gwin	TU Delft	R.W.Gwin@tudelft.nl
Alain van den Berg	TU Delft	A.M.Vandenberg@student.tudelft.nl

1.3 Document history

Version#	Date	Change
V0.1	4 January, 2010	Starting version, template
V0.2	1 February, 2010	Complete first draft
V0.3	11 Feburary, 2010	Complete revised version
V1.0	01 March, 2010	Approved version to be submitted to EU

1.4 Document data

Keywords	peer-to-peer, distributed mass media, decentralized channels, P2P widgets, graphical user interface
Editor address data	N.FerreiradeAndrade@tudelft.nl, T.Vinko@tudelft.nl
Delivery date	01 March, 2010

1.5 Distribution list

Date	Issue	E-mail
	Consortium members	QLECTIVES@list.surrey.ac.uk
	Project officer	Jose.FERNANDEZ-VILLACANAS@ec.europa.eu
	EC archive	INFSO-ICT-231200@ec.europa.eu

QLectives Consortium

This document is part of a research project funded by the ICT Programme of the Commission of the European Communities as grant number ICT-2009-231200.

University of Surrey (Coordinator)
Department of Sociology/Centre
for Research in Social Simulation
Guildford GU2 7XH
Surrey
United Kingdom
Contact person: Prof. Nigel Gilbert
E-mail: n.gilbert@surrey.ac.uk

University of Fribourg
Department of Physics
Fribourg 1700
Switzerland
Contact person: Prof. Yi-Cheng Zhang
E-mail: yi-cheng.zhang@unifr.ch

Technical University of Delft
Department of Software Technology
Delft, 2628 CN
Netherlands
Contact Person: Dr Johan Pouwelse
E-mail: j.a.pouwelse@tudelft.nl

University of Warsaw
Faculty of Psychology
Warsaw 00927
Poland
Contact Person: Prof. Andrzej Nowak
E-mail: nowak@fau.edu

ETH Zurich
Chair of Sociology, in particular
Modelling and Simulation
Zurich, CH-8092
Switzerland
Contact person: Prof. Dirk Helbing
E-mail: dhelbing@ethz.ch

Centre National de la Recherche Scientifique, CNRS
Paris 75006,
France
Contact person: Dr. Camille ROTH
E-mail: camille.roth@polytechnique.edu

University of Szeged
MTA-SZTE Research Group on
Artificial Intelligence
Szeged 6720, Hungary
Contact person: Dr Mark Jelasity
E-mail: jelasity@inf.u-szeged.hu

Institut für Rundfunktechnik GmbH
Munich 80939
Germany
Contact person: Dr. Christoph Dosch
E-mail: dosch@irt.de

QLectives introduction

QLectives is a project bringing together top social modelers, peer-to-peer engineers and physicists to design and deploy next generation self-organising socially intelligent information systems. The project aims to combine three recent trends within information systems:

- **Social networks** - in which people link to others over the Internet to gain value and facilitate collaboration
- **Peer production** - in which people collectively produce informational products and experiences without traditional hierarchies or market incentives
- **Peer-to-Peer systems** - in which software clients running on user machines distribute media and other information without a central server or administrative control

QLectives aims to bring these together to form Quality Collectives, i.e. functional decentralised communities that self-organise and self-maintain for the benefit of the people who comprise them. We aim to generate theory at the social level, design algorithms and deploy prototypes targeted towards two application domains:

- **QMedia** - an interactive peer-to-peer media distribution system (including live streaming), providing fully distributed social filtering and recommendation for quality
- **QScience** - a distributed platform for scientists allowing them to locate or form new communities and quality reviewing mechanisms, which are transparent and promote quality

The approach of the QLectives project is unique in that it brings together a highly inter-disciplinary team applied to specific real world problems. The project applies a scientific approach to research by formulating theories, applying them to real systems and then performing detailed measurements of system and user behaviour to validate or modify our theories if necessary. The two applications will be based on two existing user communities comprising several thousand people - so-called ”Living labs”, media sharing community tribler.org; and the scientific collaboration forum EconoPhysics.

Executive summary

This report accompanies and documents QMedia version 1.0. QMedia is a next-generation social media distribution platform intended to be one of QLectives' living labs. Its vision is to provide an alternative to mass media and apply the concept of self-organising personalised collectives.

QMedia version 1.0 is a special version of Tribler, an open-source media-sharing client. Furthermore, QMedia builds on the QLectives Platform version 1.0, documented in the QLectives Deliverable D.4.3.1. Tribler was taken as a starting point for QMedia and then extended and adapted to meet QLectives-specific requirements. This document focuses on such extensions and adaptations, namely the following:

- *CommentCast*, the prototype of a decentralized micropublishing system built around media-sharing channels;
- A complete *User Interface Redesign* that improves the interface formerly in place to improve the quality of the user experience, simplify the interaction model and to meet the requirements of the QMedia living lab.
- *P2P Widgets' User Interface and P2P Widgets Examples* to complement the P2P Widgets prototype platform (documented in the QLectives Deliverable D4.1.1) developed for the QLectives Platform. This deliverable introduces some simple, yet powerful, examples of such widgets to be used on QMedia. These are the ShoutBox, the New Torrent Notifier, the Hottest Torrent, and the Best Contributors widgets.

Contents

Chapter 1

Introduction

This report documents QMedia version 1.0, which is an experimental next-generation user-centric and social media distribution platform. It is built on the QLectives Platform, which is documented in the QLectives Deliverable D.4.3.1. QMedia's vision is to provide an alternative to mass media and apply the contept of self-organising personalised collectives.

As planned in the project proposal, QMedia and the QLectives Platform are based on the previously-existent Tribler codebase. Tribler an open-source software developed since 2006 by the Delft University of Technology in collaboration with several other universities and volunteers. Tribler serves as common codebase for two other FP7 projects, namely P2P-Next [1] and PetaMedia [2].

Both QMedia and the QLectives Platform are composed of core modules from Tribler extended and adapted to meet QLectives-specific requirements. We envision that such extensions and adaptations will lead to QMedia and the QLectives platform to become forks from the Tribler codebase in the future.

The contributions of this deliverable are the following:

CommentCast is a prototype of a decentralized micropublishing system. Micropublishing can be seen as a process where sources periodically publish short-form content that is delivered to a number of followers who may also publish content. Our CommentCast protocol gives each participant in QMedia network the possibility to post (and reply to) comments towards a channel (a particular media sharing source of the QLectives Platform), as well as the responsibility to help spread comments, in a distributed, efficient and secure way.

User Interface Redesign has been done in order to improve the graphical user interface of Tribler to improve the quality of the user experience, simplify the interaction model and to meet the requirements of the QMedia living lab.

P2P Widgets' User Interface with Examples complement our P2P Widgets prototype platform (documented in the QLectives Deliverable D4.1.1) developed for the QLectives Platform. This deliverable introduces some simple, yet powerful, examples of such widgets to be used on QMedia. These are the ShoutBox, the New Torrent Notifier, the Hottest Torrent, and the Best Contributors widgets.

The remaining of this document describes these contributions in details. Chapter 2 discusses the CommentCast protocol, Chapter 3 details the redesign of the graphical user interface of Tribler, while Chapter 4 gives the documentation of the P2P Widget examples.

Chapter 2

CommentCast

Twitter's huge popularity and the weaknesses in its server-based design triggered many researchers' interest to study decentralized micropublishing systems. In a general sense, micropublishing can be seen as a process where each source publishes certain content and is followed by a (usually different) number of followers who may also respond to the published content. Normally both the source and the published content can be quite diverse: it could be an every day user updating his daily life to his friend(s) or a commercial company broadcasting a new release of its product to its clients. With both friends and clients here can respond to the published content by posting some comments. In this way, a timeline of the source is established. CommentCast is a preliminary design of decentralized micropublishing. It gives each participant in QMedia network the possibility to post (and reply to) comments towards a Channel (a particular media sharing source of the QLectives Platform, for details see the QLectives Deliverable D.4.1.1., Appendix I), as well as the responsibility to help spread comments, in a distributed, efficient and secure way.

Each peer in the system keeps a private and a subjective shared comment history, which saves the comments posted by this peer, and by other peers. Periodically this peer exchanges a subset of these two comment histories with other peers. The period and candidate peers are decided by BuddyCast, the underlying gossip protocol of QLectives Platform. Considering security issues, peers only spread the comments belonging to the channels they have subscribed to (peers only subscribe to a channel when they believe that channel is in good quality, i.e. fewer spams). In this way, the more popular (in terms of number of subscrip-

tions) the channels are, the faster their comments are spread, and low quality channels will die off eventually, with and due to those malicious comments.

Intuitively, this self-organized design resembles the real human society very much and it could achieve a good performance for the system.

2.1 Local cache database

For CommentCast, each peer in the system keeps a *private comment history: (publisher id, my id, my nickname, comment txt, timestamp, signature)* which keeps a record of all the comments this peer has posted. Besides, each peer also maintains a *subjective shared comment history: (publisher id, commenter id, commenter nickname, comment txt, timestamp, signature)* which keeps a record of the comments posted by other peers.

Table 2.1: CommentCast table entry structure in local cache

filed	size	description
publisher_id	20 Bytes	the perm ID of the channel publisher
commenter_id	20 Bytes	the perm ID of the peer who posted the comment
commenter_nickname	max. 30 Bytes	the nickname of the commenter
time_stamp	4 Bytes	the time when the comment was posted
comment_txt	max. 280 Bytes	the content of the comment
signature	67 bytes	for security issue, generated from above four fields

2.2 Messages and functions

The CommentCast entries in local database are gained by exchanging CommentCast messages with other peers. Periodically,

1. each peer sends out a CommentCast message to some other peers. The period and candidate peers are decided by BuddyCast.

2. For each CommentCast message, it contains records of 10 *my_recent_comments*, 5 *my_random_comments*, 10 *others_recent_comments* and 5 *others_random_comments*;

3. upon receiving a CommentCast message, the recipient will update its local database.

Following basic functions are designed:

- **createCommentCastMessage** and **getRecentAndRandomComments**: retrieve comment records from local database and create the message;
- **createAndSendCommentCastMessage**: check the client version, check the CommentCast message validity and send out the message;
- **gotCommentCastMessage**: check the client version and the message validity;
- **handleCommentCastMessage** and **addComment**: check the local database of comment records: if there are new comments, add them to local database;

2.3 Full content pushing mode beats advertisement mode

In this design, we adopted full content pushing mode, instead of advertisement mode – which means the comment contents are contained in the CommentCast messages–, for the reason that:

1. New peers are always short of comment records. Under this situation, full content pushing is better. Otherwise eventually peers will request for the comments, then we are wasting the bandwidth for advertisements.
2. The probability of "second rendez-vous" is small (given BuddyCast's design), which reduces the probability of a large overlap of comment records between two peers.
3. In steady state, advertisement mode may be more suitable.
4. As claimed in [3], where a decentralized micropublishing (distributed twitter) prototype is brought forward, pushing entire contents of updates to subscribers is used, rather than merely notifying subscribers of new information, since this is the strategy adopted by many successful messaging systems, such as email and IM.

Chapter 3

User Interface Redesign

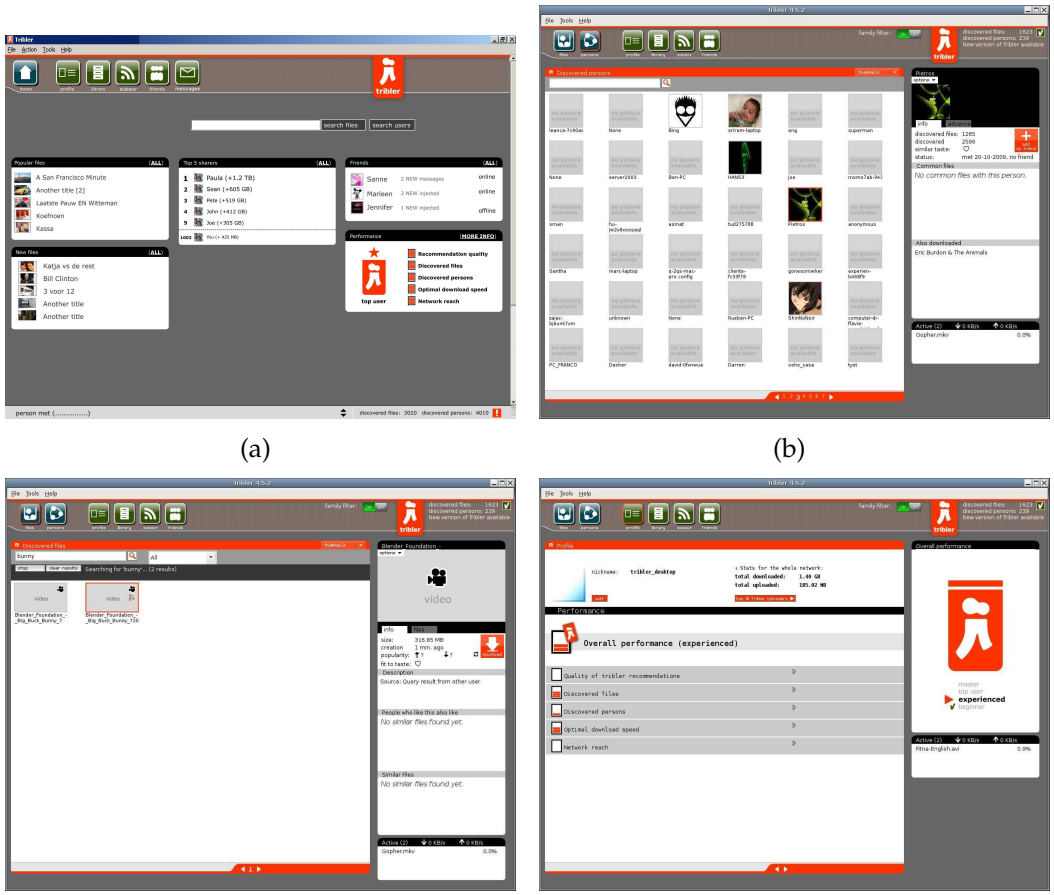
The success of the QMedia living lab depends on having a large number of loyal users in the deployed system. This observation, in turn, makes the usability of the system paramount. This section describes our work so far on improving the graphical user interface of Tribler to meet the requirements of the QMedia living lab.

More specifically, the elicited requirements are to have an interface for the system that is appealing and easily usable by a broad audience with minimal assumptions about their technical knowledge and skills. The approach chosen to make the overall user interface more appealing was to revamp the look-and-feel of the whole graphical user interface, giving it a more vivid and modern mood. As for usability, we opted to make Tribler's use as simple as possible, relying on the widespread known metaphor of a search engine portal, and to optimize for the central use-case in the system: search, acquire and watch, making it possible and easy to perform these three steps inside Tribler. We detail and discuss the rationale and implications of such choices below.

The starting point of our work is version 4.2 of Tribler, illustrated in Figure 3.1. User feedback pointed that this interface has a number of disadvantages:

- the starting window, displayed in Figure 3.1(a), is difficult to navigate: it contains many items, many items of the same importance and no clear cues on how to start interacting with the system;

- the interface exposes a number of concepts that are not obvious for users with limited technical knowledge;



(a) (b)

(c) (d)

Figure 3.1: Graphical user interface of Tribler before the redesign.

- the color palette used sets a serious and dark mood, evoking a technically complicated environment and hiding away the vibrant aspects of the media sharing experience;
- friends are exposed but there is no clear use for social contact in the interface.

The start window of Tribler after the interface redesign is show in Figure 3.2(a). There is now one central element of interaction that is exposed to the user: the search box. This search box is highly similar to that exposed by a number of other systems that virtually all users of the Web routinely rely on, such as Google, Yahoo! and Bing. There are two other components in the interface of this window: a sharing bar that indicates how much the user has shared with his peers compared to how much he has downloaded from them, and a set of links in the right

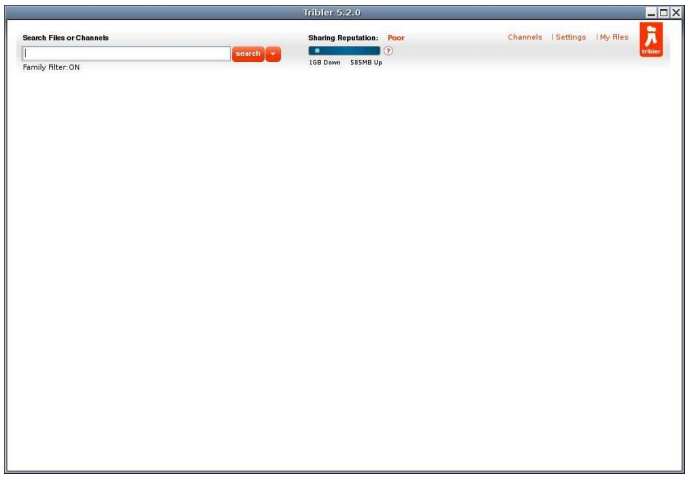
upper corner that allow the user to navigate to other tabs of the interface, which represent the different contexts in which the user might act. Furthermore, the interface redesign also led to a change in the colors used in the interface, relying on white to refer even more to the Web and search design contexts already familiar to users, and light colors to evoke a vibrant and friendly atmosphere.

The default functionality of the search box is to find files currently being distributed in the Tribler network. An example for search resultsis shown in Figure 3.2(b). The content item selected by the user is a media file, so next to it, the Tribler interface offers the user the option of starting the playback of the file straight away. This is possible due to extensions to the BitTorrent algorithms implemented in Tribler that allow the software to overlap the download of the file and its playback. This overlap, in turn, simplifies the management of files' life cycle from the perspective of users. Instead of managing the download, continuously checking its progress while waiting for the opportunity of enjoying a media file, the user can delegate this function to Tribler and wait only the minimum necessary time to watch the file. The playback of a movie file is illustrated in Figure 3.2(c).

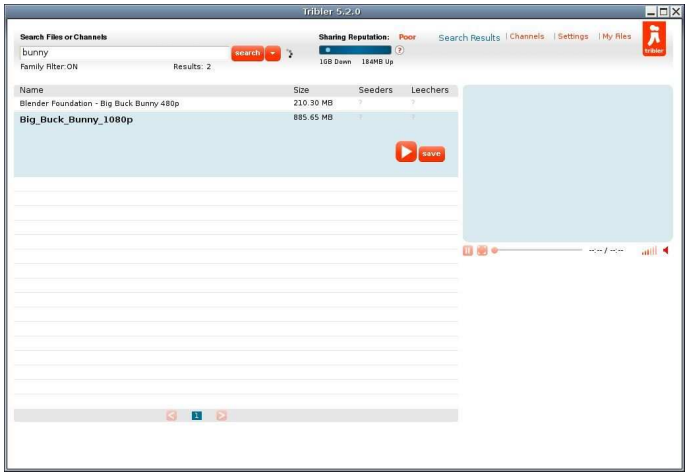
Besides the search results tab, there are three other contexts in which the user can interact with the system. There is a Settings tab, which allow him to configure the software in a user-friendly manner; there is a My Files tab, where the user can revisit files downloaded in the past and follow the progress of downloads he has started; and there is Channels tab. The Settings and MyFiles tabs are shown in Figure 3.3 and have a straightforward design. The channels tab centers around a larger innovation from the user perspective and is discussed in more details below.

Together with the change in the look-and-feel of the interface and its simplification, the addition of the Channel concept is the third major point in our interface redesign. A channel is a series of files with quality asserted by a user, and represent a first step towards the implementation of a wider social experience in Tribler. Channels are meant to help users discover quality content amidst the sea of available files; to collectively accredit users capable of identifying such quality as hubs of community activity and to eventually create digital spaces of shared interest that can seed group interaction and lead to community formation.

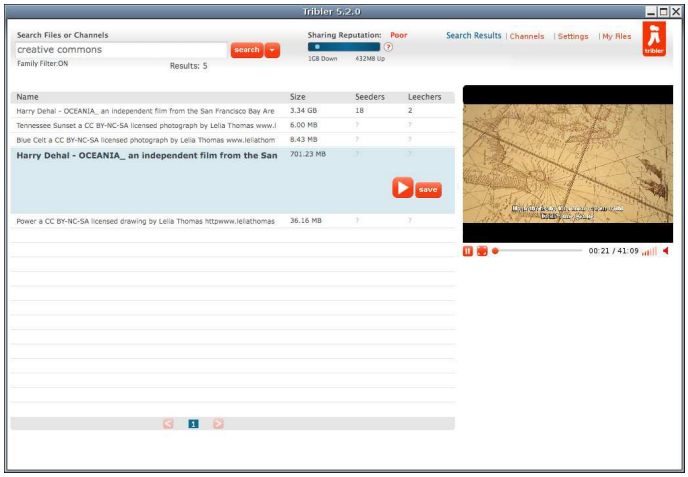
In the present version of Tribler each user has his own channel, whose man-



(a)

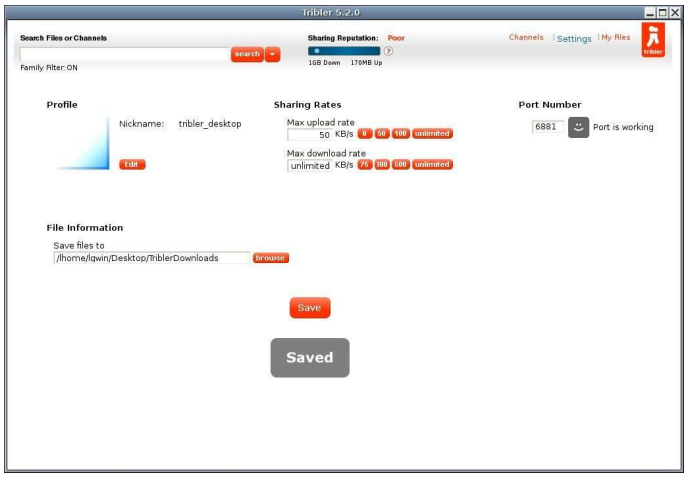


(b)

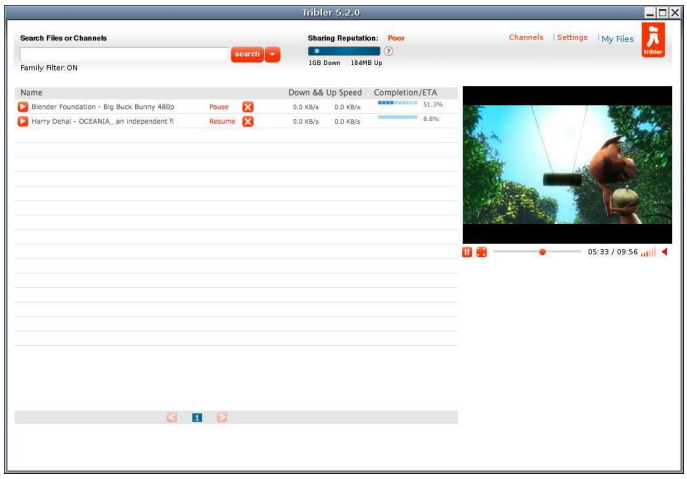


(c)

Figure 3.2: Overview of Tribler's graphical user interface after the redesign.

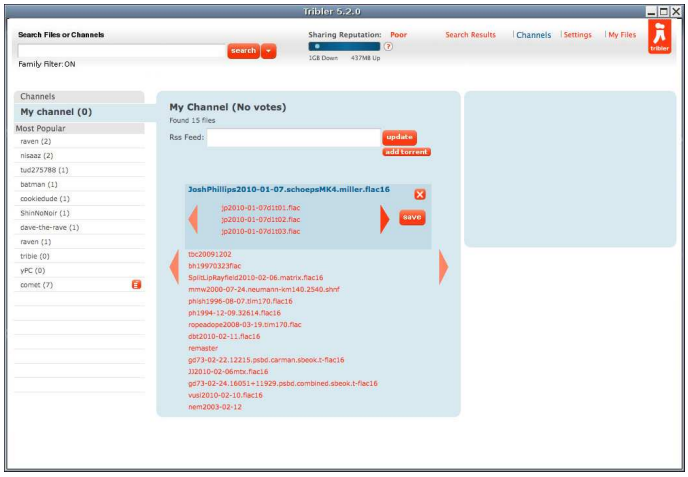


(a)

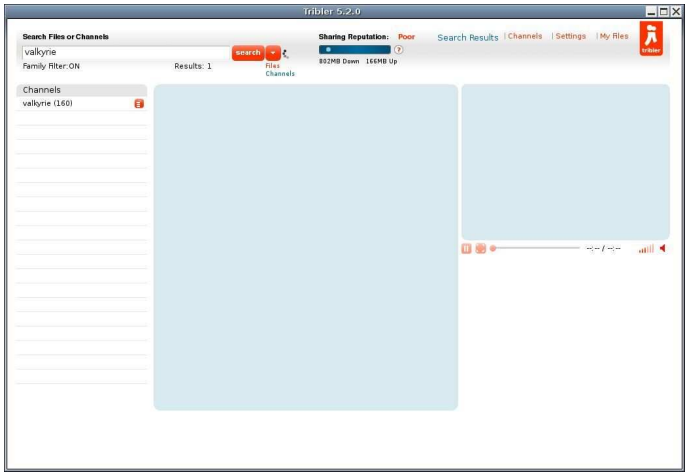


(b)

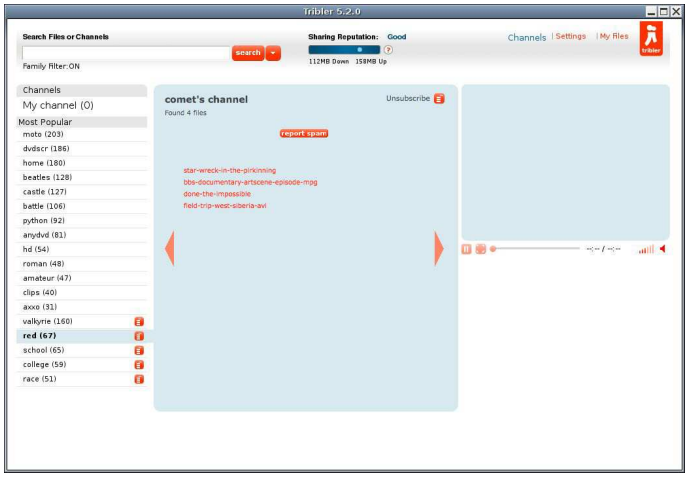
Figure 3.3: Settings and MyFiles tabs.



(a)



(b)



(c)

Figure 3.4: Interacting with channels in Tribler.

agement is illustrated in Figure 3.4(a). Channels are discovered by two means. The first option is for the user to find them similarly to files: the user selects the type of search he is performing in the drop menu next to the search box and uses the same mechanism as when searching for files. A search for channels is illustrated in Figure 3.4(b). The second option for channel discovery is to explore them based on popularity. The popularity of a channel is measured in terms of subscribers, which are users that decided they want to be informed of the addition of new items to a given channel. In this version of Tribler we take the popularity as measured by the number of subscribers in a channel as a proxy for the quality of this channel and order the channels a user is suggested to explore based on this metric. We envision, however, the evolution of the metric we use to gauge the quality of a channel in future versions of the software. This evolution is closely related and well-aligned with the research conducted in other streams of QLectives. The present way in which users can explore channels is shown in Figure 3.4(c).

Chapter 4

P2P Widgets Examples

In the QLectives Deliverable D.4.1.1 the full documentation of our P2P Widgets prototype is given. P2P Widgets are portable chunks of code that run within P2P networks, having access to functions the P2P network provides. In our implementation the Widget Engine belongs to the QLectives Platform, while QMedia provides the possibility of the actual usage of P2P Widgets. In the following we give some example applications which have been implemented for QMedia.

4.1 Widgets Graphical User Interface

The Widget Market is a P2P widget itself, which allows users to browse, install, rate and review widgets. Widget Market's primary focus is on the user interface and the dissemination of ratings and reviews. The Widget Market widget is not shown on the frontpage of QMedia, but has a menuitem called Add/Remove widgets. This saves space for the widgets the user really wants to see on the frontpage. The widget has a dialog with three tabs, where the user can perform several actions.

On the first tab, the widget market can be browsed, by viewing a sorted list of widgets. They can be sorted alphabetically or by average rating. There is an option to show every widget, because initially only whitelisted (trusted) widgets are shown in the list. Widgets can be selected to show more detailed information about the widget and the individual reviews and ratings of other users. The user can also add one review and rating per widget.

On the second tab, the installed widgets are listed and they can be removed

Figure 4.1: Graphical User Interface of the Widget Market in QMedia

easily by selecting a widget from the list and pressing the remove button.

The third tab enables widget developers to add their own widget to the Widget Market. They can select a widget file, and this file is then validated. The user can then choose to either test the widget or add it to the Widget Market. A overview of the graphical user interface is shown in Figure 4.1.

4.2 Example Widgets

To demonstrate the system's capabilities, we have created several widgets. For each widget, we will discuss what the widget does, the user interface, and the implementation of the widget. In Figure 4.2, the user interfaces of several widgets are shown.

4.2.1 ShoutBox Widget

The ShoutBox Widget is a widget that uses both the local storage and intra-widget communication to provide social interaction between P2P users. Users can leave a shout and the last ten shouts are displayed to everyone that uses the widget.

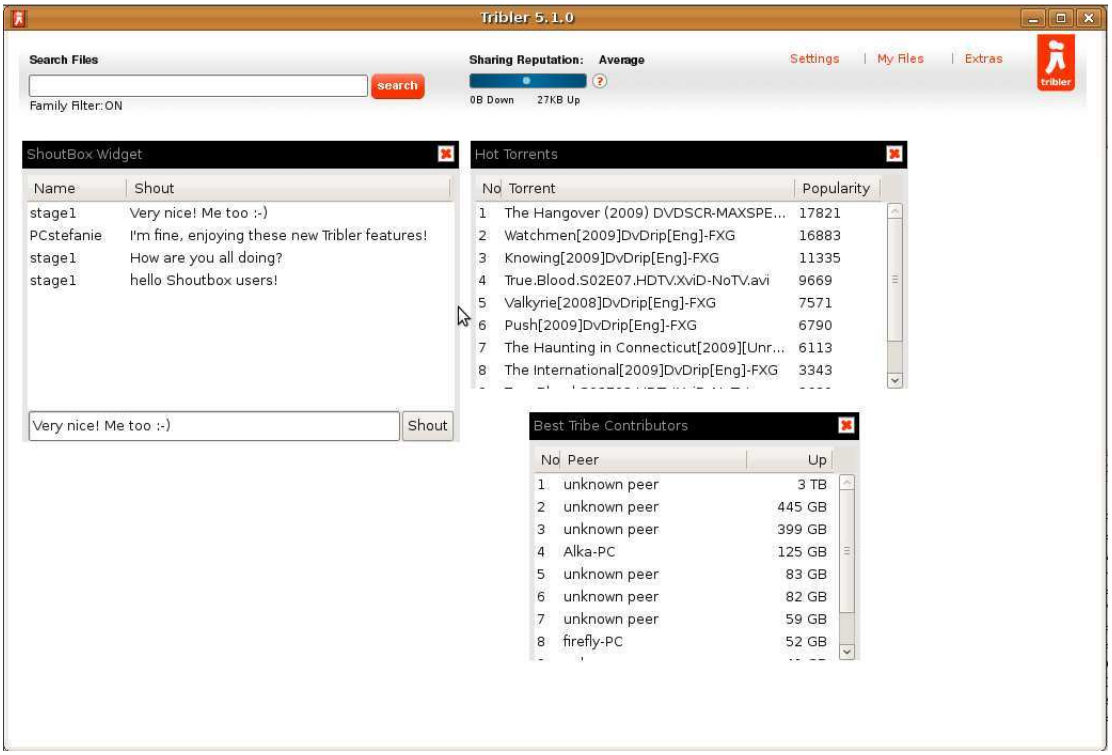


Figure 4.2: A screenshot of QMedia, including the ShoutBox, Hot Torrents and Best Tribe Contributors widgets.

The widget's user interface is quite simple, as most widgets interfaces are. It consists of a listbox with two columns, one for the name and one for the shout. Beneath the listbox, there is a textfield to type the shout and a button to send the message.

The local storage structure consists of three columns; the name, the shout and a clock value. The name and shout are self-explanatory. The clock is used to create a message order. When a new shout is inserted, the clock value is one higher than the current maximum clock value in the database. This way, messages that are replies to other messages get a higher clock value and are positioned higher than the others. Messages with the same clock value may be ordered differently on different computers. The gossip message consists of the ten most recent shouts. When a gossip message from someone else is received, its shouts are either inserted or ignored when they already exist.

4.2.2 New Torrent Notifier Widget

The New Torrent Notifier widget shows the most recently locally discovered torrents by BuddyCast. It does not use any local storage or intra-widget communication, but instead hooks into the notifying system of QPlatform.

The user interface is again very simple. The listbox, where the torrents are shown, is the most prominent GUI element and takes up most of the widget's space. It has one column, which shows the torrent name. A download button is placed beneath the listbox.

By registering an observer to be notified on widget insertion, the widget receives its torrents. The function that is called on insertion of a widget, receives the infohash of the torrent. We can then retrieve the torrent name and filename from the Tribler Torrent database. The name is inserted in the list and a mapping from torrent name to filename is stored, to be able to download the torrent. The list is then purged, such that at most 10 torrents are shown in the widget. When an item is selected to download, we retrieve the torrent filename and call the session to start the download. This widget implements the OnClose event handler, which removes the observer before the widget is removed.

4.2.3 Hottest Torrents Widget

The Hottest Torrents Widget displays the hottest torrents known at that time. It uses the information known locally, collected by BuddyCast, and combines this information with the information of other Hottest Torrents users using intra-widget communication for fast convergence.

Again, the widget displays the hottest torrents in a listbox, with three columns. First column shows the place of the torrent, the second the name of the torrent and the third the popularity. This is the same popularity value as is discussed in the QLectives Deliverable D.4.1.1.

Upon initialisation, the user interface is created. Upon OnPostInit, which is called by the runtime when the local storage is initialised, the top ten of most popular torrents is retrieved from the QPlatform database and stored in the local storage of the widget. Also, a timer is started to update the information of the widget with the QPlatform database every few minutes. Updating the database means that the top ten of torrents from QPlatform is merged with the top of the

widget: new torrents are added, old torrents are updated and finally the list is purged.

The local storage structure consists of the torrent infohash, torrent name, popularity and checktime, which is the local time of the peer when it retrieved the information from the QPlatform database. The torrent infohash is the primary key. Intra-widget communication is used to exchange the peers hot torrents and syn- chronise with with the other users. Upon creation of a gossip message, the ten hottest torrents are added to the message. Upon receipt of a message, this top ten from the other peer is merged, taking into account the checktime values.

The Hot Torrents Widget is built in such a way that the convergence is much faster than when only BuddyCast is used. In just a few minutes, it is possible to show the top torrents of the whole QMedia overlay. The more users there are, the faster the convergence. By using the checktime value, torrents that are becoming less popular, will eventually disappear from the list, because the torrents swarmsize is occasionally checked by QMedia peers.

4.2.4 Best Tribe Contributors Widget

This widget has basically the same architecture as the Hot Torrents widget. It takes a top ten, of BarterCast peers in this case, and synchronises it with other peers and occasionally with the local QPlatform database, converging to the global top ten of BarterCast's most altruistic peers.

This time, the interface shows the place of the user, his name and how much he has uploaded, according to QMedia. Because not every peer's nickname is known, a lot of names are filled with "unknown peers", but by synchronising with other peers, these names will eventually be filled with their real nicknames.

The local storage also consists of three columns: a permid, name and up. Up is the number of bytes uploaded. The primary key consists solely of the permid. Upon creation of a message, the current top ten is added to the message. When a message is received, the information in the message is merged with the currently known top. Either an entry is updated, meaning that up and name are taken from the message, or the entry is added. The highest value of up is always taken, since the value can only increase. A timer is used to merge the QPlatform BarterCast information into the widgets top every minute.

Chapter 5

Summary and Further Research Questions

This report documents the first release of QMedia, a media-sharing P2P system built on the QLectives Platform. Furthermore, this report describes the contributions of the QLectives project to the common code base that constitutes QMedia, Tribler and the QLectives Platform. The current version of the software is ready for deployment so as to serve as the QMedia Living Lab. A software release with this intention is planned for the first quarter of 2010. The experience with this release together with the research plan of QLectives will guide the evolution of the software towards version 2.0, planned for month 24 of the project.

Bibliography

[1] P2P-Next. http://www.p2p-next.org/.

[2] PetaMedia. http://www.petamedia.eu/.

[3] Daniel R. Sandler and Dan S. Wallach. Birds of a fethr: Open, decentralized micropublishing. In *8th International Workshop on Peer-to-Peer Systems (IPTPS '09) April 21, 2009, Boston, MA*, 2009.